Deliverable number:	D4.2
Deliverable title:	Ontology-based feedback on model quality

Delivery date:	2010/07/31
Submission date:	2010/08/31
Leading beneficiary:	Universidad Politécnica de Madrid (UPM)
Status:	Version 2.0 (final)
Dissemination level:	PU (public)
Authors:	Esther Lozano, Jorge Gracia, Asunción Gómez-Pérez, Jochem Liem, Carsten van Weelden, and Bert Bredeweg

Project number:	231526
Project acronym:	DynaLearn
Project title:	DynaLearn - Engaging and informed tools for learning conceptual system knowledge
Starting date:	February 1st, 2009
Duration:	36 Months
Call identifier:	FP7-ICT-2007-3
Funding scheme:	Collaborative project (STREP)

Abstract

This document describes the Ontology-Based Feedback process addressed to improve the quality of the models developed by learners. The core idea is to compare the knowledge contained in a learner model with the knowledge contained in a reference model (made by an expert) in order to get feedback on the quality of the former one. This is carried out by measuring how similar/dissimilar both models are and by identifying the ingredients and structures in which they differ. This information is used to construct suggestions of improvements which are shown to the user during the modelling process.

Comparisons are performed by using ontology matching techniques, which allow the alignment between learner and reference models. Some additional techniques are also applied, like semantic reasoning and model structure comparison, thus allowing a richer analysis of the different knowledge contained in the compared models and leading to the generation of different types of feedback.

Internal Review

- Elisabeth André, Michael Wißner, René Bühling, Multimedia Concepts and Applications, University of Augsburg (UAU), Germany.
- Petya Borisova, Institute of Biodiversity & Ecosystem Research, Bulgarian Academy of Sciences (CLGE/IBER), Bulgaria.

Acknowledgements

We would like to thank people from University of Brasilia for providing us the models for the evaluation of the ontology-based feedback. We specially thank Andreas, Maria and Anne Marie (from BOKU) for their participation in the evaluation experiment and their helpful comments. We would also like to thank Wouter for his help and support in the evaluation. We thank also our internal reviewers for their useful comments.

Document History

Version	Modification(s)	Date	Author(s)
0.1	Outline and general ideas	2010/07/14	Esther Lozano
0.2	First draft of main sections	2010/07/16	Esther Lozano
0.3	Complete draft of the document	2010/07/20	Esther Lozano
0.4	Review of the document	2010/07/20	Jorge Gracia
1.0	Draft for review by partners	2010/07/20	Esther Lozano
1.1	Review of the document	2010/07/25	Asunción Gómez-Pérez
1.2	Update of sections 1 and 2	2010/08/16	Esther Lozano
1.3	Review of the document	2010/08/24	Jorge Gracia
2.0	Final version	2010/08/26	Esther Lozano

Contents

1. Introduction

In the context of DynaLearn, providing learners adequate feedback on their models is crucial for successful teaching. For that reason, the **Ontology-Based Feedback** (OBF) module belonging to the Semantic Technology (ST) component in DynaLearn, provides semantic feedback on a single conceptual model created by a learner, supporting the learner in improving that model such that it best explains a phenomenon according to his/her personal ideas. This is done by comparing the **learner** model with a related model created by a teacher or expert and that we call **reference model**.

The ontology-based feedback proposed in DynaLearn combines ontology matching techniques, ontology-based reasoning and QR reasoning in order to discover the differences between the learner and the reference models. Each technique is being used in our algorithm with different purposes. The following briefly sketches how they are used and which type of problem they address.

One problem to solve is the use of different vocabularies in models, particularly when dealing with learners who are in the process of discovering canonical forms in the involved terminology. In order to establish a notion of similarity, we first need to determine how the terms used in a model relate to each other. This can be resolved by using **ontology matching** techniques [1, 2]. An ontology matching approach establishes which concepts in different ontologies correspond to each other, typically identifying equivalences (although other relations such as ‘more general’, ‘less general’, ‘disjoint’, etc are possible). Furthermore, the use of inference engines for **semantic reasoning** or **structure comparison**, allows us to obtain better feedback on the quality of the learner model. Semantic reasoners allow us to carry out comparison between the learner and reference (OWL based) models in order to detect missing structures or over specified ones in the learner model. Finally, we use QR reasoning to detect additional missing structures in the learner models using QR model patterns.

The document is organized as follows. First, an overview of the ontology-based feedback is presented in Section 2. Section 3 describes the necessary information for the communication between the components Semantic Technology (ST) and Conceptual Modelling (CM) during the generation of the OBF. In Section 4, the process of ontology matching and the techniques derived from it is discussed. Section 5 presents the techniques based on semantic reasoning. Section 6 describes the feedback based on structure comparison. Finally, Section 7 contains the conclusions, and Section 8 finishes the document with some additional discussion.

2. Overview of Ontology-Based Feedback

The semantic repository created in our system (see Deliverable D4.1 for further information) is intended to store and make accessible the pool of QR models created by users. This infrastructure is also utilized to support feedback during the model construction process. To that end, we devise the use of ontology matching techniques, semantic reasoning and QR specific comparisons between models. Depending on the particular technique, our system provides different types of feedback, as it can be seen in Subsection 2.1. Notice, however, that these types of feedback are not mutually exclusive and can be combined. The input of this process consists of two models: one corresponding to the **learner model** (the one under construction) and the other corresponding to the **reference model** that is made by an expert. As we explain later in this section, these input models must be in OWL language in order to perform the ontology matching between them. The output of the ontology-based feedback is the list of differences between the two models that is sent to the CM component in an XML representation. Eventually, this information will be shown to the user as suggestions of improvements. The interaction with the final user will be performed by CM and VC components and is out of the scope of this document (see Deliverable D5.4).

2.1. Types of Ontology-Based Feedback

The different types of Ontology-Based Feedback can be grouped by the type of mismatch that they solved (and thus by the technique used in the implementation). The following example illustrates the different types of mismatches can be found between two models:

The different types of mismatches can be classified as follows:

- **Terminological mismatches**. Two models of the same domain can use different terms to represent the same concepts and also can include a different number of concepts. Therefore, in this category we identify the following types of mismatches.
    - Improvements of terminology: differences in the labels or groundings used for equivalent terms.
    - Missing terms in the learner model: terms of the reference model without correspondence in the learner model.
    - Extra terms in the learner model: terms of the learner model without correspondence in the reference model.
- **Structural mismatches**. This process, described in Section 4, uses ontology matching for discovering equivalences between two models. The analysis of these differences addressed from different points of view leads to different types of feedback:
    - Missing hierarchical relationships: hierarchical relationships associated with a missing term (if the term is added, then its hierarchical relations must be taken into account).
    - Inconsistencies between hierarchies: differences between the two hierarchies that lead to an inconsistent situation.

- Missing QR model structures: missing elements in the learner model (instances of entities or quantities, dependencies, quantity spaces, etc.) that appear in some model fragment in the reference model.

In order to solve the possible mismatches, we use different techniques to address each type of mismatch. These techniques can be grouped as follow:

- **Ontology matching-based feedback**. This process, described in Section 4, uses ontology matching for discovering equivalences between two models. The analysis of these differences addressed from different points of view leads to different types of feedback:
  - Improvements of terminology (differences in the labels used for equivalent terms)
  - Missing terms in the learner model
  - Extra terms in the learner model
  - Missing hierarchical relationships
- **Semantic reasoning-based feedback**. By means of a semantic reasoner we can detect inconsistencies between the hierarchies of the two models that can be pointed out to the learner. More details are given in Section 5.
- **Model structure-based feedback**. From the QR point of view, we can exploit the particular semantics of the vocabulary to perform more specific comparisons between the models and to find model structures that are missing in the learner model. With this technique we can identify differences between the quantity spaces associated to two equivalent quantities, missing influences in the model, etc. This is described in Section 6.

2.2. Preconditions

The OBF process needs to receive a pair of models: one corresponding to the learner model (under construction) and other corresponding to a reference model made by an expert. However, these models have to fulfil certain preconditions before starting the process.

2.2.1. Grounding of models

To facilitate the work of the ontology matching technologies, and to improve the obtained results, both the learner model and the reference model should be previously grounded (see Deliverable D4.1 for a detailed explanation of the benefits of semantic grounding). Although this is not indispensable for the execution of the ontology-based feedback, the grounding of models provides additional information about the terms and better results can be obtained. If grounding of terms exists, we search for terms in the learner and the reference model that have the same grounding. Therefore, if two terms are grounded to the same concept, we can infer that they are equivalent. These equivalences between terms establish a preliminary set of mappings that is used during the ontology matching process as a useful starting point to find out new equivalences between the models. In addition, the ontology-based feedback can find equivalences between terms with different grounding, thus helping the learner to improve the grounding of his terms, by suggesting reusing the same groundings made by the expert.

2.2.2. Representation of models

The same model can be described in DynaLearn in several ways, depending on the requirements of the particular module that processes it (see Deliverable D3.2 for more details):

- **QR model**. This is the original model as it is made by the user in the DynaLearn workbench and is stored locally in a HGP file.
- **QR model in OWL**. This is the version of the QR model exported in OWL language. It contains all the information which characterise the model. The exported model can be stored locally in an OWL file and also be sent to the semantic repository to be stored.
- **Extracted vocabulary** (or domain vocabulary) of the model. This is an adapted representation of the QR model, containing the core semantic description of the model only, without most of the tool-specific constructions. This allows semantic processing techniques such as ontology matching or semantic reasoning to use the semantic of the ingredients to improve ontology matching performance. The extracted vocabulary is sent and stored into the semantic repository jointly with the QR model in OWL.

In the context of ST, we need to deal with QR models in OWL for storing and retrieving models in the Semantic Repository. Therefore, a transformation step has to be carried out to export the QR models into OWL [3] (this process is described in Deliverable D3.2).

However, for the ontology matching techniques used by the OBF module, the extracted vocabulary is required, owing to the peculiarities of the QR knowledge representation in contrast to the knowledge representation typically used in ontologies. In fact, ontologies tend to describe concepts based on their possible relationships (i.e. by specifying restrictions on concepts in OWL). However in DynaLearn QR models do not represent explicitly such properties for the model ingredients definitions. Instead, the way the ingredients are used in model fragments gives knowledge about their conceptual meaning (for example on the instance level).

2.2.3. Reference model

The reference model must be found among all models made by experts available in the semantic repository, according to some criteria that helps identifying the most suitable one for each case. In later stages of the project, when functionalities based on collaborative filtering will be developed (Deliverables D4.3 and D4.4), the reference model will be found automatically from the pool of models. To do this, the current meta-vocabulary of the model will contain the necessary data to do the collaborative filtering.

In the meantime, in the Ontology-Based Feedback module the reference model is provided directly by the user. There are two different operating modes:

a) Specific URI provided by the teacher

The teacher indicates to the learners the *target* model they should obtain, so learners can use it for getting the feedback.

b) Selection of the reference model among a ranked list

In this case, the learner has to select the best model among the available reference models stored in the repository. The system compares the learner model with all these models using specific metrics for ranking. In addition, the student should be able to decide how to apply these

metrics and, therefore, in which terms the model should be compared (i.e. the student could ask for the most similar model in terms of entities, quantities, etc).

Independently of how the reference model is selected, the reference model has to be previously stored in the repository. That means that both versions of the model, the QR model in OWL and its extracted vocabulary, must be present in the repository before the learner asks for feedback

.

2.3. Data flow

In order to describe the Ontology-Based Feedback (OBF) module in the context of the DynaLearn architecture, we have represented the data flow inside and outside the OBF component. We follow a top-down approach with two levels of detail: level 0 or context level, which represents the OBF component in the context of the other components of DynaLearn; and level 1, describing the data flow between the subcomponents of OBF.

2.3.1. Level 0: communication between DynaLearn components

Figure 1: Context level data flow diagram.

When a learner asks for quality feedback on the current model, CM sends the learner model and the URI of the reference model that is already stored in the semantic repository to the ST component, then the OBF process starts (see **¡Error! No se encuentra el origen de la referencia.**). The output is the list of differences between the two models that is sent back to the CM component. With this information, CM can send the adequate messages to VC, which will inform the student providing the necessary suggestions to get their models improved. All the communication between the ST and CM components is done through Web Services.

2.3.2. Level 1: communication between subcomponents of OBF

The two models (the learner model and the reference one) constitute the input of the Ontology-Based Feedback process. As **¡Error! No se encuentra el origen de la referencia.** illustrates, the first step in this process is the *grounding-based alignment*, where groundings of the two models are compared for discovering equivalent terms. These 'preliminary mappings' are sent to the *Ontology Matching* module in order to find the complete list of equivalent terms, which is sent to the next modules: *Semantic Reasoning* and *Structure Comparison*. The output of each function is addressed to the corresponding functionalities.

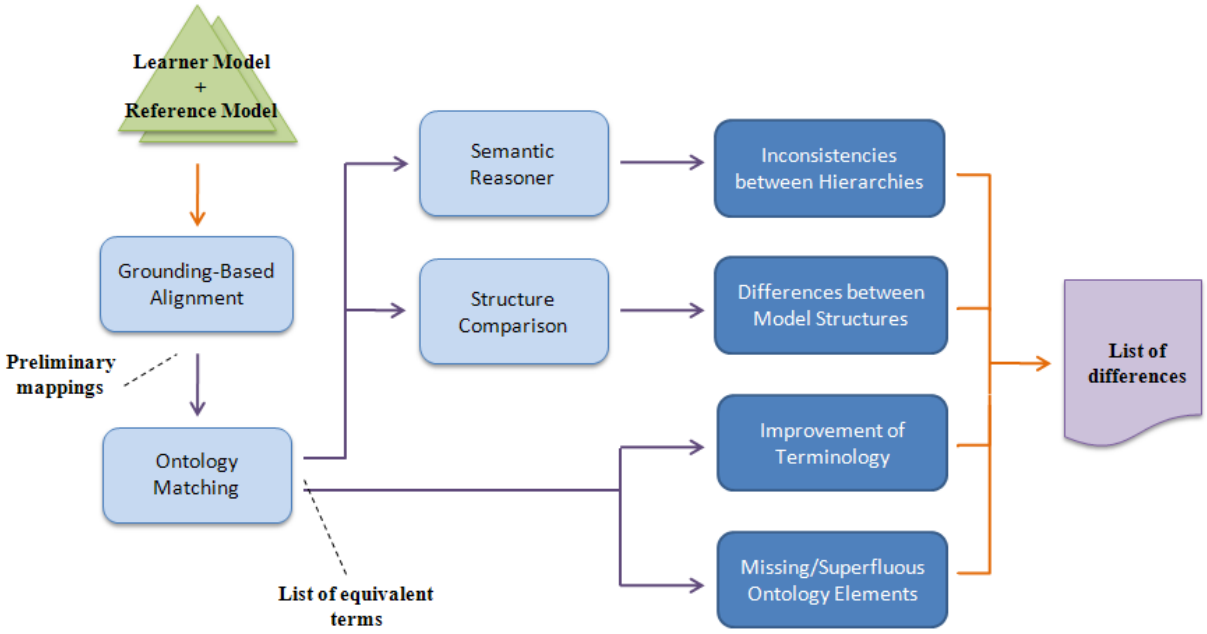


Figure 2: OBF data flow diagram.

2.4. General algorithm of the Ontology-Based Feedback

In the following sections we describe in detail each of the techniques implemented in the Ontology-Based Feedback. However, to understand the general behaviour of the OBF and to show how the different techniques are integrated in the whole process, we provide here the general algorithm of the OBF.

The formal algorithm of this process follows:

1. Retrieve the reference model from the semantic repository (model in OWL + exported vocabulary)
2. Store the learner model in OWL and its extracted vocabulary in the repository (if necessary)
3. Find the set of equivalent terms between the models (mappings)
    i. Obtain the preliminary mappings from the **grounding-based alignment**
    ii. Apply the **ontology matching** technique between the models
4. Calculate the similarity between the models
5. Find the differences of terminology between the matched terms (differences of label and/or grounding)
6. Detect the missing terms in the learner model
7. Detect the missing hierarchical relations for the missing terms in the learner model
8. Detect the extra terms in the learner model
9. Apply **semantic reasoning** to detect inconsistencies between the hierarchies of the models

10. Do **structure comparison** to find the differences between QR structures of both models
11. Remove the learner model from the semantic repository (if it was added at the beginning)

This algorithm matches with the data flow presented above in this section. Details of the input and output data are provided in Section 3, and the specific algorithms of each of the techniques, are described in Sections 4, 5 and 6.

3. Data exchange between CM and ST in the Ontology-Based Feedback

In Section **¡Error! No se encuentra el origen de la referencia.**2.3 have presented the data flow between the ST component and the CM during the execution of the Ontology-Based Feedback. In this section we provide a detailed description of these data as well as the format used in the representation.

3.1. Input data

The main actors in the OBF process are a model created by a learner and another one created by an expert that is taken as a reference model. Therefore, and according to the preconditions established in Section 2.2, the necessary information that has to be sent to the ST component is the following:

Learner model exported into OWL

The QR model developed by the learner is exported into OWL format, containing the information of all the model terms, as well as the groundings generated by the student. The learner creates the model locally and then has the option to store it into the semantic repository, although this is not compulsory. However, in order to apply semantic reasoners and other tools, the model has to be present (even unfinished) in the repository. For that reason, if the learner model has not been saved yet we temporary store it in the repository before executing the OBF. At the end of the process we remove it so the final status of the repository is not affected for the execution of the OBF.

Extracted vocabulary of the learner model

As mentioned in Section 2.2, we need an adapted representation of the model in order to apply ontology matching techniques on it. When the export model is stored in the repository, its extracted vocabulary is also provided and stored. This version, which is also in OWL, must be provided jointly with the QR model in OWL and stored in the semantic repository. If the learner model was not present in the repository before the execution of OBF (and thus the extracted vocabulary neither), this is also removed.

URI of the reference model exported into OWL

As mentioned above, the reference model has to be stored into the semantic repository (both QR model in OWL and its extracted vocabulary) before executing the OBF. Therefore, the only information we need to receive from the CM component is the URI of such a model. Based on that, we can retrieve the corresponding model from the repository. As the QR model in OWL and its extracted vocabulary are related internally in the repository, with this URI we are also able to retrieve the extracted vocabulary of the model (that has a different URI), needed for the ontology matching process.

3.2. Output data

The output data of the OBF process consists of the list of differences found between the learner model and the reference model, plus numerical values to indicate similarity between models. The list of differences contains, for each difference, specific information of the involved terms.

Information about the learner term

This will help the learner to identify the term to be improved, providing the necessary information for understanding the suggestion. This includes:

- URI of the term
- Label of the term
- QR type (entity, quantity, ...)
- URI of the grounding associated to the term

Information about the equivalent term in the reference model

In order to understand the suggestions and be able to decide if any change should be made, we need to show the information about the term used in the reference model and found equivalent to the learner term during the ontology matching.

- URI of the term
- Label of the term
- QR type (entity, quantity, ...)
- URI of the grounding associated to the term

Relation between the terms

In addition to the related terms discovered by the ontology matching technology, we have to provide the type of relation that states between the terms. Although most of the OBF techniques provide pairs of equivalent terms that differ in some aspects, in other cases like missing hierarchical relations the pair of terms do not share a relation of equivalence but a hierarchical one. To be aware the user of this situation, together with the information of the two terms we provide the type of relation between them: the two terms are equivalent; the reference term is subtype of the learner term; or the reference term is supertype of the learner one.

3.2.1. Representation format of the output data

As explained in the Deliverable D4.1, the communication between the CM and the ST components is done through a layer of web services provided by the ST component. In the case of the ontology-based feedback, the response of the corresponding web service (*getOBF*) contain (in XML format)

1. **Serialization of results in XML format**
   The results obtained from the different feedback techniques are transformed into XML format by the Web Service. Each individual result contains all the information regarding the affected term (see previous sections for a more detailed explanation about each type of result).

2. **OWL formalization (meta-model)**
   The output information is also provided in a formal way using OWL language and following a meta-model that provides the necessary vocabulary for representing the ontology-based feedback. This will be the input needed by Virtual Characters for constructing the final version of feedback.

4. Ontology Matching

As described in Deliverable D4.1, the use of ontology matching techniques fit naturally in the process of getting knowledge-based feedback, since we want to compare models or model fragments to reference models (expert models, teacher models, etc), and these models can be described in terms of an ontology language. Actually, these comparisons are favored by the relatively small size of the models (with respect to the utilized background ontologies, such as DBpedia), and by the similar nature of the compared ontologies: they share the same domain, are represented using the same meta-model, etc. In DynaLearn, we use CIDER system to discover the mappings between the learner and the reference models.

In this section we describe how the ontology matching technique is used for the OBF and the different types of feedback that can be generated. As explained in Section 2.2.2, we need to work with the extracted vocabularies, instead of the whole QR model, when working with ontology matching techniques. Therefore, along this section we assume that the reference and learner models are represented in terms of their extracted vocabularies. An evaluation plan for the techniques presented in this section can be found in the Appendix A of the document.

4.1. Ontology matching process for OBF

In order to select a suitable ontology matching tool for the OBF process, an evaluation with some state-of-the-art alignment systems was done in the context of QR modelling (details of this evaluation experiment can be found in Deliverable 4.1). After analysing the results, we decided to use CIDER [5] as ontology matching system (though not discarding other options in the future).

4.1.1. Grounding-based alignment

According to the OBF algorithm described in Section 2.4, the first step to find the equivalent terms between the learner and reference models is based on exploring their respective groundings. Since the concepts of both models are grounded to a common vocabulary, we use these relations to infer a preliminary set of mappings. Let us assume that the learner model has a concept labeled *Death* that is grounded to the DBpedia resource *Mortality rate*:

```
<owl:Class rdf:about = "&qrm;owl_ae_Death">
  …
  <owl:sameAs rdf:resource = "http://dbpedia.org/resource/Mortality_rate"/>
</owl:Class>
```

The reference model has a concept labeled Mortality that is also grounded to the same DBpedia resource *Mortality rate:*

```
<owl:Class rdf:about = "&qrm;owl_ae_Mortality">
  …
  <owl:sameAs rdf:resource = "http://dbpedia.org/resource/Mortality_rate"/>
</owl:Class>
```

Therefore, as they share the same meaning, we can infer that *Death* and *Mortality* are equivalent terms. In OWL, this can be expressed by establishing that both learner term and reference term are subjects of two *owl:sameAs* statements with the same DBpedia resource as object. In the grounding-based alignment, we transform these statements into *owl:equivalentClass* ones with the learner term as subject and the reference term as object.

```
<owl:Class rdf:about = "&qrm;owl_ae_Death">
  ...
  <owl:equivalentClass rdf:resource = "&qrm2;owl_ae_Mortality"/>
</owl:Class>
```

These new statements are included in a new owl file that will be also used as input for the ontology matching tool, together with the exported models in OWL.

4.1.2. Identification of equivalent terms

After identifying the preliminary set of mappings based on common groundings, we apply the ontology matching system. As result, we obtain a list of equivalences between learner terms and reference terms. This list is generated at the beginning of the OBF process and is used as starting point in all the feedback techniques described in this document.

These mappings are described in RDF following the *Alignment Format* [4]. Thus, each mapping is represented by a *cell* that contains the URI of the two related terms, the type of relation between them and a numerical value as a quantitative measure of how similar they are:

```
<map>
  <Cell>
    <entity1 rdf:resource='http://www.dynalearn.eu/models/Model23.owl#owl_ae_Death'/>
    <entity2 rdf:resource='http://www.dynalearn.eu/models/Expert6.owl#owl_ae_Mortality'/>
    <relation>=</relation>
    <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.4526</measure>
  </Cell>
</map>
```

4.1.3. Measuring similarity between models

The list of equivalent terms resulting from the ontology matching process is analyzed to obtain some quantitative values indicating how similar the models are. These values are obtained by means of different metrics with different expressiveness degree.

Algorithm

1. Get the number of equivalent terms found between the two models using CIDER: *nEquivalent*
2. Get the number of terms from the learner model analyzed during the matching process: *totalLearner*
3. Get the number of terms from the reference model analyzed during the matching: *totalRef*
4. Calculate the output values:

a. Percentage of terms in the learner model with an equivalence in the reference model:

$$pEquivalentLearner = \frac{nEquivalent}{totalLearner}$$

b. Percentage of terms in the learner model without an equivalence in the reference model:

$$pDifferentLearner = \frac{totalLearner - nEquivalent}{totalLearner}$$

c. Percentage of terms in the reference model with an equivalence in the learner model:

$$pEquivalentR = \frac{nEquivalent}{totalR}$$

d. Percentage of terms in the reference model without an equivalence in the learner model:

$$pDifferentR = \frac{totalR - nEquivalent}{totalR}$$

4.2. Solving terminological mismatches

As result of applying ontology matching techniques between a reference model and the learner model, we obtain a set of semantic correspondences (mappings) between the terms of both models. We analyze these correspondences to check whether the terms from the learner model are well defined regarding the terminology of the reference model or not. This type of mismatches can be found in models from all the learning spaces. We describe now the types of terminological mismatches that we can find using ontology matching.

4.2.1. Mismatches in terminology

Firstly, we check whether the learner terms are properly grounded and labeled. Two terms that have been deemed equivalent in the ontology matching process could share the same label and grounding. By comparing the learner terms with their equivalent reference terms we are able to detect these differences and suggest a better option to the student. As an example of this, if a learner has an entity labeled *Sustainable biomass* but the equivalent term in the reference model has the label *Carrying capacity*, the system will point out the difference so the learner can replace the label by the one used in the reference model. Differences between the groundings are also possible, since two terms with different grounding can be still found equivalent by the ontology matching.

Algorithm

For each one of the equivalences found between the models:

a. Compare the labels of the learner term and the reference term

b. Compare the groundings of the learner term and the reference term

c. If the labels and/or the groundings are different, add a new difference to the list of differences including the information of both terms. The relation between the terms is set to *equivalent*.

Figure 1 illustrates the interface that shows the results of this OBF technique in the current DynaLearn prototype. However, in future stages of the project the ontology-based feedback will be communicate to the user via VC.

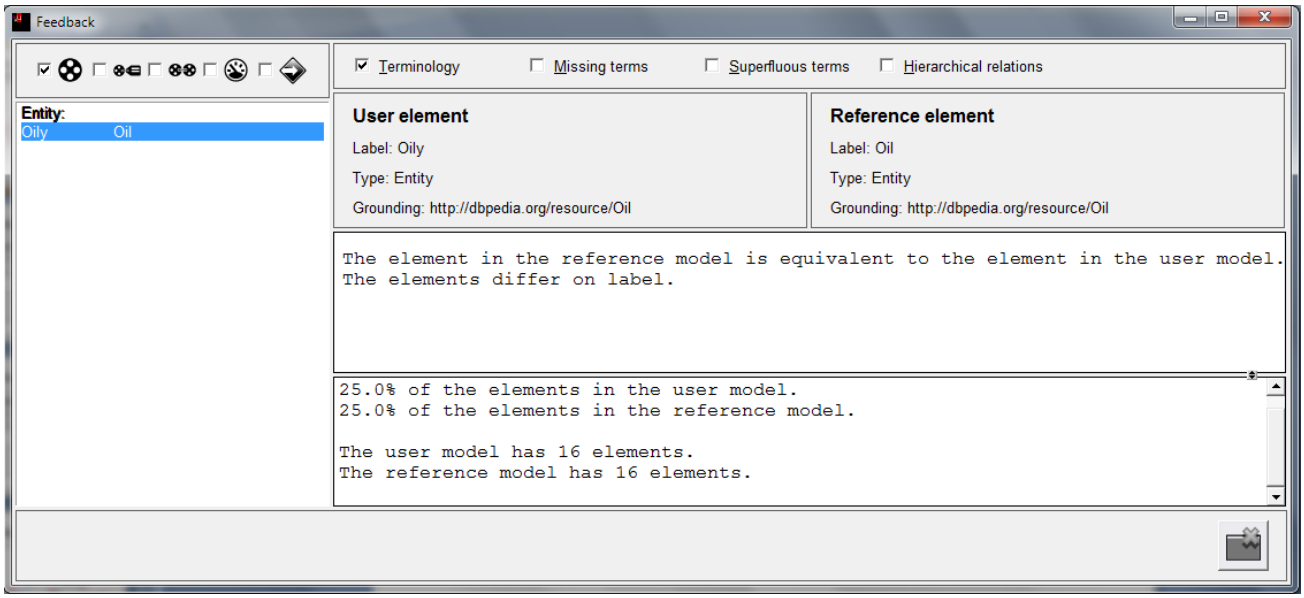


Figure 3 Mismatches in terminology in the OBF interface

4.2.2. Missing terms in the learner model

Aligning the two models and knowing how similar the terminology used by two models is, we can identify the terms that are present in the reference model but not in the learner one. Those terms, called *missing terms*, can be important for the right construction of the model. Therefore, they are presented to the learner who decides whether to include them in the model or not.

Algorithm

For each term from the reference model:

- Search the URI in the list of equivalent terms
- If not found, add the missing term to the list of differences between the models

In the current interface for the ontology-based feedback in DynaLearn the information is shown as Figure 4 illustrates.

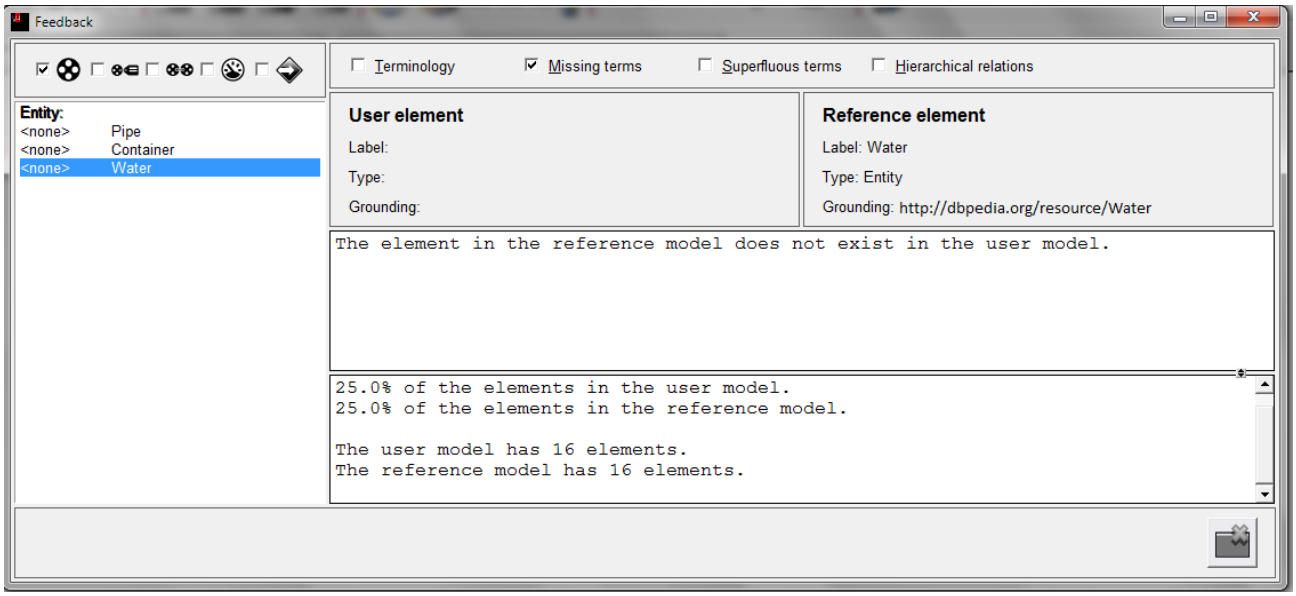


Figure 4 Missing terms in the OBF interface

4.2.3. Extra terms in the learner model

The learner model can contain terms which are not defined in the reference model. This might indicate that those terms are not needed and thus considered as *extra terms*. Those terms are shown to the learner who can remove them from the model if necessary.

Algorithm

For each term from the learner model:

- Search the URI in the list of equivalent terms
- If not found, add the extra term to the list of differences between the models

In the current interface for the ontology-based feedback in DynaLearn the information is shown as illustrated in Figure 5.

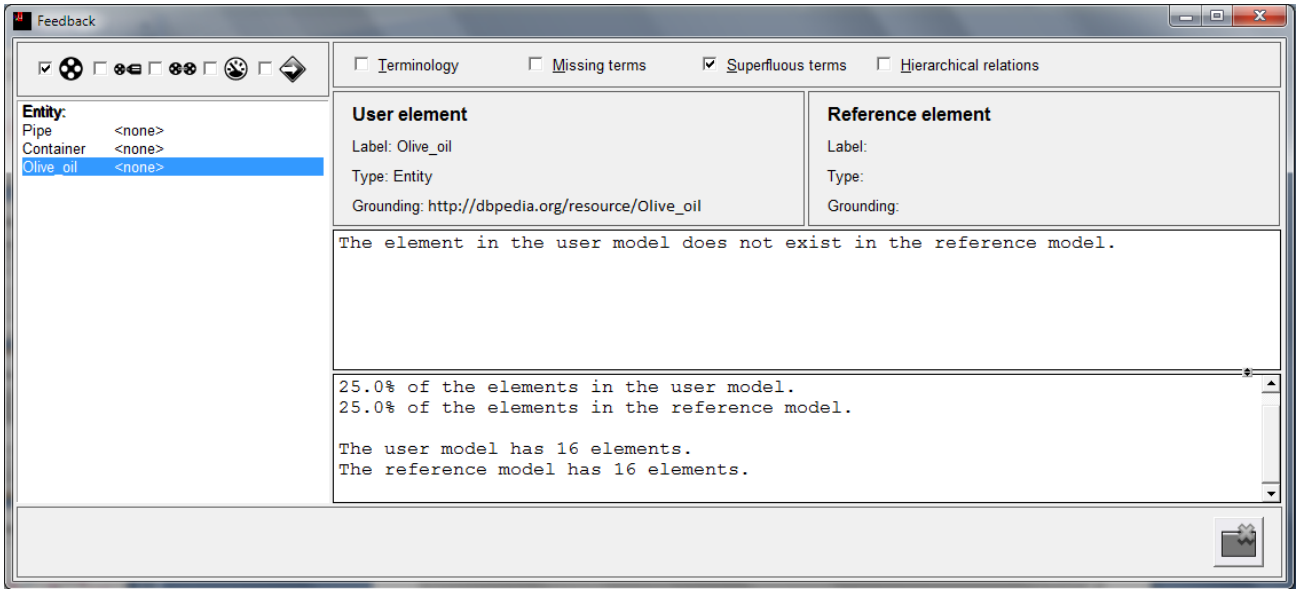


Figure 5 Extra terms in the OBF interface

4.3. Solving hierarchical mismatches

In addition to the differences between the terms used in the learner model and the reference model, we can also find the differences between the hierarchical structure of both models, thus discovering the missing hierarchical relationships for the found missing terms.

This technique can only be applied to models built in learning space 6, given that we need the hierarchy of entities to provide this type of feedback. This technique has to be executed after obtaining the list of missing term.

After aligning the two models and finding the missing terms, we can analyze the hierarchical relationships of those terms. The missing terms are hierarchically related with other terms in the reference model, as subtypes or supertypes. Even if a term is missing in the learner model, the terms related to it in the reference model could be present in the learner model. In that case, the hierarchical relation is suggested to the student together with the missing term.

As an example, Figure 6 shows the entity hierarchy of a learner model and Figure 7 the entity hierarchy of a reference model:

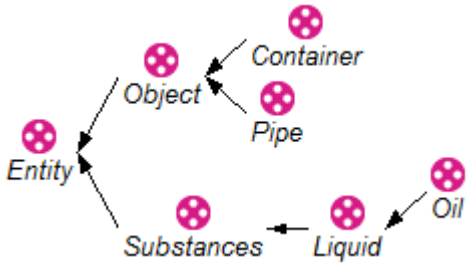


Figure 6 Entity hierarchy of a learner model about communicating vessels

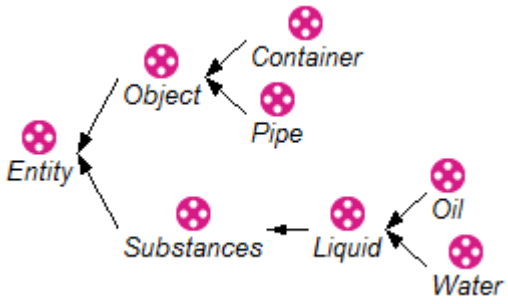


Figure 7 Entity hierarchy of a reference model about communicating vessels

In this example we can see that the term *Water* is missing in the learner model and thus can be suggested to be added to the model. In addition, we see that in the reference model *Water* is subclass of the entity *Liquid*, which has a correspondence in the learner model. Therefore, the learner term *Liquid* will be suggested as superclass of *Water* as a missing hierarchical relation.

In the case *Liquid* was the missing term in the learner model instead of *Water*, this OBF technique would produce three suggestions: *Liquid*, the missing term, should be added as subtype of the term *Substances* of the learner model; *Liquid* should be added as supertype of the term *Oil* of the learner model and *Liquid* should be added as supertype of the term *Water* of the learner model;

Algorithm

For each term in the reference model that is missing in the learner model:

1. Get the superclass of the term in the reference model (*refSuperclass*)
2. Look for the equivalent term of the *refSuperclass* in the learner model (*userSuperclass*)
3. If *userSuperclass* exists (is not a missing term), add to the list of differences. In this case the two terms included in the list of differences are:
    i. The missing term in the learner model (the information is obtained from the reference model)
    ii. *userSuperclass* (the term from the learner model)
    iii. The relation between the terms is set to *subClassOf* to indicate that the reference term (the missing term) should be added as subtype of the learner term (*userSuperclass*).
4. Get the subclass of the term in the reference model (*refSubclass*)
5. Look for the equivalent term of the *refSubclass* in the learner model (*userSubclass*)
6. If *userSubclass* exists (is not a missing term), add to the list of differences. In this case the two terms included in the list of differences are:
    i. The missing term in the learner model (the information is obtained from the reference model)

ii. *userSubclass* (the term from the learner model)

iii. The relation between the terms is set to *superClassOf* to indicate that the reference term (the missing term) should be added as supertype of the learner term *(userSubclass*).

In the current interface for the ontology-based feedback in DynaLearn the information is shown as illustrated in Figure 8.

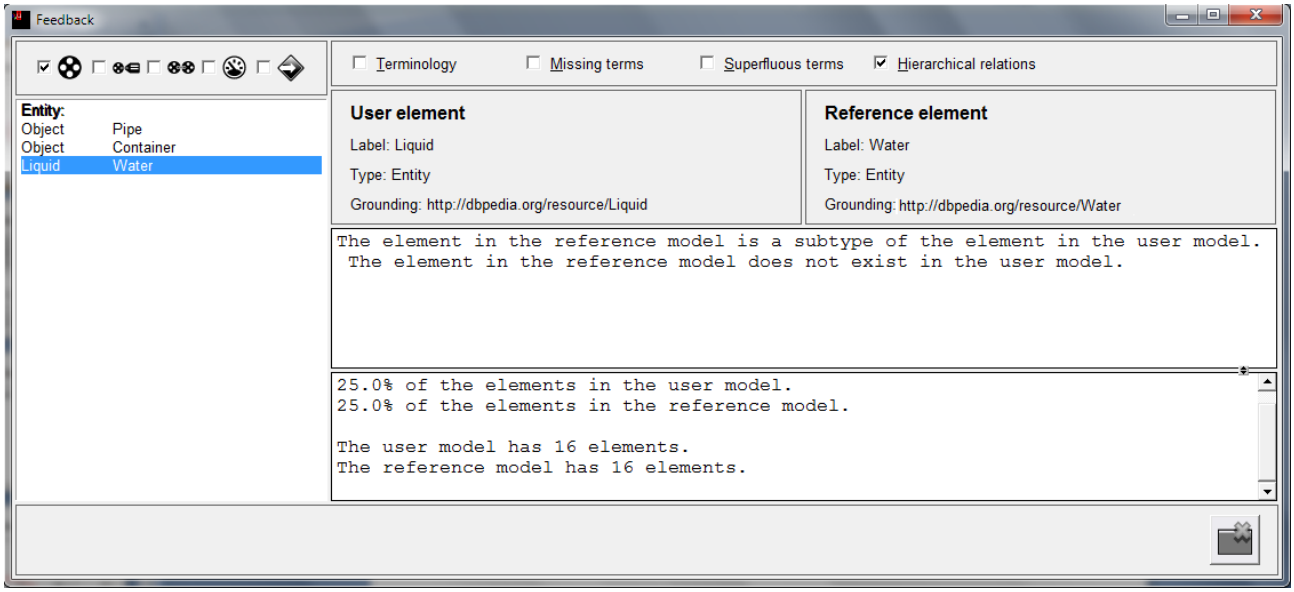


Figure 8: Missing hierarchical relationships in the OBF interface

5. Semantic Reasoning

In order to discover hierarchical inconsistencies between the models, we use semantic reasoning techniques by applying certain entailment rules. These rules allow us to detect inconsistencies between the learner and the reference models that go beyond of finding terminological mismatches and structural differences in the entities hierachy. The applied inference level has to be adapted, given that the higher the expressivity level is, the longer the reasoning time (though it also depends on the instances). Therefore, a trade-off between performance and inference level becomes necessary, taking into account that the potential volume of semantic content will be high.

In our current prototype we use Jena built-in reasoner[1], though any other reasoner can be used. Notice that in this document we are bringing forward the application of semantic reasoning techniques. However, this task will be approached later in the project (see Deliverable 4.4).

5.1. Inconsistencies between hierarchies

By using a semantic reasoner, we can search for differences between the hierarchies of the learner model and the reference model that lead to an inconsistent state. Notice that in this case we are also working with the extracted vocabulary of the models in order to apply the semantic reasoners.

Let us say the learner model defines the entity *Whale* as subclass of *Fish*. However, the reference model states that the equivalent term *Whale* is subclass of *Mammal*. As the entities *Mammal* and *Fish* have been declared as disjoint classes, these two statements result to be inconsistent. As a result, the system informs the student of the inconsistency so the hierarchy can be reviewed and changed accordingly.

Algorithm

1. For each pair of equivalent terms between the learner and the reference model:
    a. Get from the extracted vocabulary of the reference model all the statements related to the current reference term. The search covers all the superclasses of that term plus the properties in which the term appears as domain and/or range.
    b. Get from the extracted vocabulary of learner model all the statements related to the current learner term. The search covers all the superclasses of that term plus the properties in which the term appears as domain and/or range
    c. Add the sets of statements selected in both, learner and reference vocabularies, to a temporary ontology.
2. Once the temporary ontology contains the necessary statements from the two models, search for duplicate statements and remove them from the temporary ontology.
3. Apply a semantic reasoner to the temporary ontology.
4. Add the list of inconsistencies to the list of differences between the models.

[1] http://jena.sourceforge.net/inference/

6. Structure Comparison

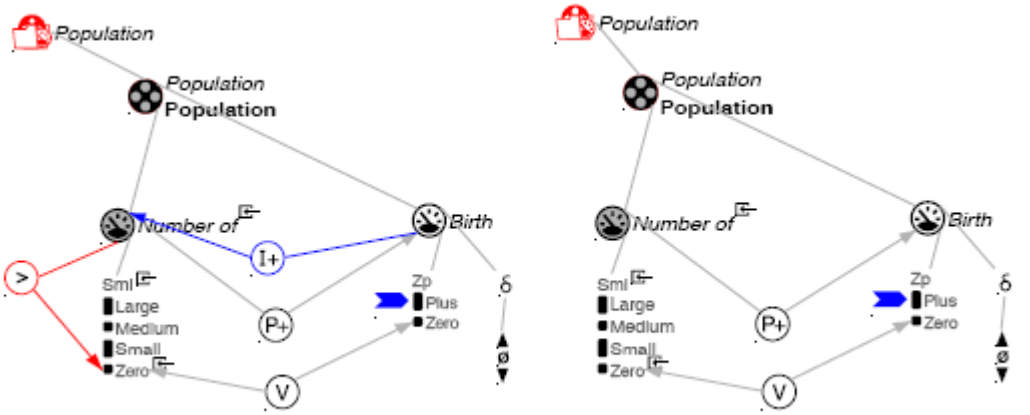
Besides the ontology-based comparisons described in previous sections, we also exploit the particular semantics of the QR vocabulary to perform more QR-specific comparisons between the models. In fact, we can identify common model structures that are present in the reference model but not in the learner model, thus revealing *differences between model structures*.

In this type of feedback, and opposite to the rest of feedback techniques, we need to deal with the QR model exported in OWL instead of doing it with its extracted vocabulary. That is because of the extracted vocabulary version does not contain the description of some ingredients like model fragments, influences or proportionalities, which appear in the model structures we want to compare.

6.1. Differences between model structures

This last functionality is about finding structures from the QR point of view that are present in reference model and are not in learner model. Even if there is not any inconsistency between processes, these *missing structures* can modify the final behavior of the model.

To detect these differences, we make a structural comparison between the models. First, patterns in the reference model are searched; then, by means of the set of mappings, we look for the same patterns in the learner model. This comparison process can detect that some model structures are missing in the learner model. Figure 9 illustrates this. In the example, an inequality property in *Number of* quantity and the positive influence between the quantities *Birth* and *Number of* are pointed out to allow the student to make the corresponding changes.



(a) Reference model fragment. (b) User model fragment with missing structures.

Figure 9: Example of feedback on missing model structures

Algorithm

For each pair of equivalent terms between the learner and the reference model:

1. Get the model fragments from the reference model where the reference term appears
2. For each model fragment found in the reference model:
    a. Extract the rest of terms that constitute the model fragment
    b. In the learner model, search for the most similar model fragment to the reference one
    c. Extract the ingredients involved in the learner model fragment
    d. Compare the two lists of ingredients, adding the missing ingredients (and the superfluous ones) to the list of differences.

7. Conclusion

This deliverable describes the current state and progresses on Task 4.2 “Ontology-based feedback on model quality”. First, we have presented the relation between the OBF component and the rest of DynaLearn components, as well as the preconditions that have to be accomplished before executing the process. Then, we have defined the different types of feedback grouped by the technology applied in each case. At the beginning of this document we present the format of the necessary data for starting the process (the input data). In addition, we describe the data provided as result (output data) form the communication elements between ST and CM components for the OBF process. The rest of the document describes in detail the behaviour of each type of feedback. The necessary techniques and algorithms have been presented and discussed.

In summary, the goal of Task 4.2 has been achieved, although further refinements are expected in the future, when user-based experiments will be carried out, and a significant amount of models will be available in the repository for experimentation. In the meantime, the OBF module already provides the basic mechanisms to analyse pairs of QR models in order to compute similarities and differences, aimed to suggest quality improvement during the modelling construction process.

8. Discussion

There are still some open questions related with the ontology-based feedback that will be solved in the future of the project. We have seen the benefits of grounding the terms before executing the ontology-based feedback. However, it is still unclear where we should put the boundary to decide that a model has enough terms grounded to ensure the ontology-based feedback will get better results. An option we are discussing is to ground at least all the entities and quantities of the model, Other types of ingredients, like configurations that represent relations between entities, can be more difficult to ground. Also, wrong groundings of terms can be damaging for the ontology-based feedback. A preliminary set of mappings is generated based on the groundings, so a wrong grounding could generate a wrong mapping between terms, Therefore, the models have to be grounded enough, but without introducing wrong groundings.

Finally, the ontology-based feedback compares a reference model and a learner model to point out the differences and help the student to have a better model. However, it is not clear yet the amount of information about the reference model that should be given to the student. An option is to allow the learners to see completely the reference model used, so they can analyze and understand the differences between the two models. On the other hand, this option also allows the student to copy the reference model without understanding the reasons of the differences.

References

1. Shvaiko, P. and Euzenat, J. “A survey of schema-based matching approaches”. Journal on Data Semantics, 4:146-171, 2005.

2. Euzenat, J. and Shvaiko, P. “Ontology Marching” Springer-Verlag, Heidelberg, 341 pages. 2007

3. Liem, J. and Bredeweg, B. “OWL and qualitative reasoning models” in C. Freksa, M. Kohlhase and K. Schill, editors, KI2006: Advances in Artificial Intelligence. 29th Annual German Conference on AI, number 4314 in Lecture Notes in Artificial Intelligence, pages 33-48, Bremen, Germany, 2207.

4. Euzenat, J. “An API for ontology alignment” in Proc. 3rd conference on International Semantic Web Conference (ISWC), Hiroshima, Japan, Lecture Notes in Computer Science 3298:698-712, 2004

5. Gracia, J. and Mena, E. "Ontology matching with CIDER: Evaluation report for the OAEI 2008," in Proc. of 3rd Ontology Matching Workshop (OM'08), at ISWC'08, Karlsruhe, Germany, vol. 431. CEUR-WS, October 2008, pp. 140-146.

6. Hu, W. and Qu, Y. "Falcon-AO: A practical ontology matching system," Journal of Web Semantics, vol. 6, no. 3, pp. 237-239, 2008.

7. d’Aquin, M. “Formally measuring agreement and disagreement in ontologies” in Proc. Of the 5th International Conference on Knowledge Capture (KCAP’09), pages 145-152, September 2009.

Appendix A: Evaluation plan of Ontology-Based Feedback

We need to evaluate how useful the OBF is during the learning process and how it can help the students to improve the quality of their models. These evaluations will involve the educational experts of this project and will take place in a short future. However, before doing that we need to evaluate the correctness of the results of the OBF. This first evaluation is the one we are describing in this document.

The hypothesis is that Ontology Matching techniques can be applied to QR models to find equivalences (or mappings) between terms. Therefore, we need to test the correctness of those mappings.

Description of the evaluation

The models to be used in the evaluation will be stored in the Semantic Repository. The models will be individually pre-grounded, so the evaluators do not have to ground them. This is because in this experiment we are not evaluating the grounding process

The evaluators will receive a list with the learner models and the reference models they have to use in each exercise. Evaluators will open from the repository one of the learner models and they will ask for feedback using the corresponding reference model. Then, they will go through all the types of feedback, checking all the suggestions made about the learner terms. For each suggestion, the evaluator will assess its correctness using specific buttons in the interface. The data will be automatically registered in a log file.

The evaluation will consist of two parts:

Part A

In this part, the learner models will consist of different versions of the reference model, created by making specific modifications over some terms to cover all the types of feedback. Given the high similarity between the two models to compare, the suggestions will be easy to understand and will thus be easy to evaluate as well. This part of the experiment will allow evaluators to familiarize themselves with the tool and the evaluation procedure.

Part B

In the second part, the two models will be made by different users that try to model the same topic. In this case, we think more differences will be generated and evaluation will take more time and effort. This part will be the most important, since it represents the actual use case of the OBF.

Metrics for each type of feedback

Mismatches in terminology

The terms of the learner model and the terms of the reference model found as correspondent by the Ontology Matching process are compared (the labels and the groundings) and the differences are shown to the user.

Values to measure:

- Number of times that evaluator agrees with using the suggested label (the one used in the reference term) for the learner term.
- Number of times that evaluator does not agree with the suggested label.
- Number of times that evaluator agrees with using the suggested grounding (the one used in the reference term) for the learner term.
- Number of times that evaluator does not agree with the suggested grounding.
- Number of times that evaluator considers the pair of equivalent terms as a *wrong mapping*.

Missing terms in the learner model

After matching the learner model with the reference one, the terms from the reference model that do not have an equivalence counterpart in the learner model are shown as missing terms.

Values to measure:

- Number of times that users agree that the suggested term should be added to the model:
  - Number of times that evaluator agrees with the label of the reference term.
  - Number of times that evaluator does not agree with the label.
  - Number of times that evaluator agrees with the grounding of the reference term.
  - Number of times that evaluator does not agree with the grounding.
- Number of times that evaluator considers that already has that term (*missing mapping* case).
- Number of times that evaluator considers that the term is not necessary in the learner model.

Extra terms in the learner model

After matching the learner model and the reference one, the terms from the learner model without equivalence in the reference model are shown as missing terms in the reference model.

Values to measure:

- Number of times that users agree that the *extra* term should be removed from the model.
- Number of times that users consider, after seeing all the terms in the reference model, that the term is actually present in the reference model (*missing mapping* case).

Missing hierarchical relationships

These missing terms are hierarchically related with other terms in the reference model, as subtype or supertype of those terms. Even if a term is missing in the learner model, the terms related to it in the

reference model could be present in the learner model. In that case, the hierarchical relation is suggested together with the missing term.

Values to measure:

- Number of times that evaluator agrees with the suggested term and also with the suggested relation.
- Number of times that evaluator does not agree with the suggested relation.
- Number of times that evaluator considers the suggested mapping as a *wrong mapping* between the superclass in the reference model and the equivalent term in the learner model.

e-mail: Info@DynaLearn.eu
website: www.DynaLearn.eu