



Deliverable number: D4.3

Deliverable title: **Model-based and memory-based collaborative filtering algorithms for complex knowledge models**

Delivery date: 2011/07/31 - 2011/10/31 (extended)

Submission date: 2011/11/15

Leading beneficiary: Universidad Politécnica de Madrid

Status: Version 2.0 (final)

Dissemination level: PU (public)

Authors: Esther Lozano, Jorge Gracia, Diego Collarana, Oscar Corcho, Asunción Gómez-Pérez, Boris Villazón, Sander Latour, and Jochem Liem

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

In DynaLearn, learners, teachers and domain experts create Qualitative Reasoning (QR) conceptual models that may store in a common repository. These models represent a valuable source of knowledge that could be used to assist new users in the creation of models with related topics. However, finding the appropriate models for this knowledge reuse can be a difficult task as the amount of models in the repository increases.

This document describes the task of recommending relevant models from the repository and its integration with the generation of semantic feedback. The recommendation process integrates both model-based and memory-based collaborative filtering algorithms. The recommended models are used as reference sources and are compared with the learner model. From the analysis of the differences between the models, a list of suggestions is generated and provided to the learner as feedback.

Finally, the document includes an appendix describing the User Management System and how the models in the repository can be organized in courses. These courses play an important role in the recommendation of models.

Internal review

- Bert Bredeweg, Human Computer Studies Laboratory, University of Amsterdam (UvA)
- Richard Noble, Hull International Fisheries Institute, University of Hull (UH)

Acknowledgements

We thank our internal reviewers for their careful review and useful comments.

Document History

Version	Modification(s)	Date	Author(s)
0.1	Initial draft	2011-09-25	Esther Lozano
0.2	Intro. feedback + recommendation	2011-10-04	Esther Lozano
0.3	Full version	2011-10-20	Esther Lozano
0.4	Review	2011-10-23	Jorge Gracia
0.5	Version for internal review	2011-10-27	Esther Lozano
0.6	Changes after UH internal review	2011-11-03	Esther Lozano, Jorge Gracia
1.0	Final version	2011-11-07	Esther Lozano
1.1	Appendix: UMS	2011-11-08	Diego Collarana
1.2	Review	2011-11-08	Jorge Gracia
1.3	Changes after UvA internal review and UH second review	2011-11-14	Jorge Gracia
2.0	Final version	2011-11-15	Esther Lozano, Jorge Gracia

Contents

Contents	4
1. Introduction	6
2. Overview of the semantic technologies	8
3. Recommendation of Relevant Models	9
3.1. Input	9
3.2. Criteria and computation of relevance	10
3.2.1. Multi-criteria attributes for recommendation	10
3.2.2. Utility function (or how to get the relevance value)	10
3.3. Recommendation process	11
3.3.1. Filtering of unrelated models by number of common groundings	12
3.3.2. Ranking of relevant models	12
3.3.3. Filtering of models based on relevance threshold	12
3.4. Output	12
3.5. Rating of models	13
4. Semantic-based feedback from the set of relevant models	14
4.1. Input	15
4.2. Grounding-based alignment	16
4.3. Ontology matching	16
4.4. Discrepancies between terminologies	16
4.5. Taxonomic inconsistencies	17
4.6. Structural QR discrepancies	17
4.6.1. Extraction of basic units	17
4.6.2. Integration of basic units of the same type	18
4.6.3. Comparison of equivalent integrated basic units	18
4.6.4. Matching of basic units of the same type	19
4.6.5. Comparison of equivalent basic units	19
4.7. Semantic-based feedback from the pool of models	20

5. Communication of results	21
5.1. Graphic User Interface	21
6. Evaluations	22
6.1. Evaluation of recommendation	22
6.2. Evaluation of semantic-based feedback	22
6.3. Evaluation of the entire system	23
7. Conclusion	24
8. Discussion	25
References	26
Appendix A: User Management System	28

1. Introduction

In DynaLearn¹, users create Qualitative Reasoning (QR) conceptual models that can be stored in a common repository. To support the modeling tasks, semantic technologies were developed to facilitate the reuse of relevant knowledge from existing models in the repository. The selection of relevant knowledge is done by means of recommendation techniques that find the most relevant models for a particular learner model.

In Deliverable D4.2 [12] we introduced the techniques developed to generate semantic feedback based on a reference model. In this document we describe how to select this reference model automatically, and how we adapted the initial feedback techniques to generate the feedback not only from the reference model but from the set of relevant models selected through recommendation. The goal is to allow learners to create good-quality QR models by reusing the shared knowledge contained in previously created models. Such shared knowledge is offered to the modeler as suggestions that steer learners in enhancing their models by themselves. The intention is not to replicate reference models but to extract from them valuable knowledge that allows learners to develop better knowledge and understanding of a domain system by producing better models. To select relevant models from the repository the system needs to apply recommendation techniques that automatically identify those models that might be relevant to the user.

Collaborative Filtering is a type of recommendation technique that aims to provide users with personalized recommendations based on information obtained from similar like-minded users [3]. The data typically used in these types of systems are based on preferences from other users expressed through ratings (typically subjective scores) of the available items, but not on the features that define the recommended items [2, 5, 17]. In DynaLearn the aim was to use Collaborative Filtering to filter the recommendations provided to the learner based on the context of their work, i.e., the models that they have created in the system. In addition to the usual sources of information used for Collaborative Filtering, such as numerical or discrete ratings, tags or keywords, DynaLearn also uses the knowledge characteristics of the models created by users, based on their semantic description (e.g., we explore their grounding into background knowledge sources).

According to the Description of Work [4], the task described in this document is aimed to “extend Collaborative Filtering algorithms to take into account object description models that go beyond simple ratings or values and are based on the knowledge models created either by experts or by other peers”. The result is a method that generates personalized recommendations ranked in a list, such that highly ranked models are most beneficial for the learning process. This set of relevant models constitutes the input for the generation of semantic feedback for a given learner model. The common discrepancies between the learner model and the set of relevant models are analyzed from different perspectives, and generate suggestions from these differences that are communicated to the learner as feedback aimed to improve their model. The particular Collaborative Filtering techniques applied in DynaLearn are both memory-based filtering (based on other users of the system) and model-based filtering (based on the characteristics of the models).

The rest of this document is organized as follows. Section 1 defines the different levels in the development of conceptual models. Section 2 provides an overview of all the semantic technologies developed so far and how they are related. Section 3 describes the process of recommending relevant models. Section 4 gives an update of the generation of semantic feedback using the relevant models as result of the recommendation. Section 5 shows how the results are communicated to the learner.

¹ <http://hcs.science.uva.nl/projects/DynaLearn/>

The evaluations run to test these techniques are described in Section 6. Finally, the conclusions and discussion are provided in Sections 7 and 8 respectively.

2. Overview of the semantic technologies

As it has been described in previous deliverables [10, 12], DynaLearn uses semantic techniques to link user² models with background ontologies and models, and to analyze and determine differences between learner and reference models. These differences are communicated to users as feedback aimed to stimulate learners to evaluate their models and to progress towards more advanced, descriptive, and useful models.

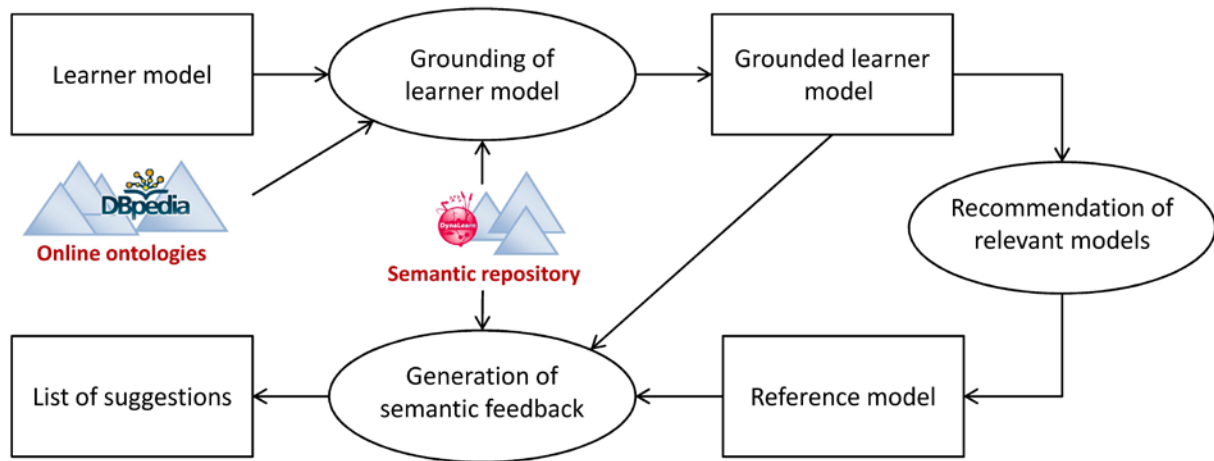


Figure 1: Overview of the semantic technologies

The models created by users are stored in a semantic repository where they remain accessible for later reuse. Besides this, the system executes the following processes (see Figure 1):

- **Grounding of learner model.** Determining links between the unrestricted terminology used by learners and well defined external vocabularies [9].
- **Recommendation of relevant models.** Recommendation of the most relevant model from the repository to be used as reference.
- **Generation of semantic feedback.** Analysis of the differences between the learner model and the reference model and generation of suggestions provided as feedback for the learner.

The DynaLearn workbench is used for creating the model. During this process, the learner grounds the terms to online knowledge sources principally using DBpedia (as described in [9]) and, at a certain point, the learner asks for semantic feedback. Then, relevant models contained in the semantic repository are obtained. The most relevant model is used as a reference model and compared with the user model for the generation of feedback. As a result, a list of differences is obtained which are communicated back to the learner as suggestions to improve the model. Then, the learner can decide whether to follow such suggestions or not, thus changing or maintaining the model accordingly. They can also decide to get more suggestions from other models from the set of relevant ones. Notice that the learner model is not stored in the repository until the user decides so, but feedback can be still gained without the model being saved in the repository.

The details of the relevant model selection and how they are use to get relevant feedback can be found in the following sections.

² In the rest of the document we use “user” to refer either to learner or to expert.

3. Recommendation of Relevant Models

The generation of feedback is based on the comparison between the user model and a set of relevant models used as reference. To select this set from the semantic repository, we run a recommendation algorithm based on the relevance of models. Our hypothesis is that the more relevant the reference model is to the user model, the better the feedback generated.

As we mentioned in previous sections, instead of using a classic Collaborative Filtering recommendation based only in ratings, we aim to take into account model-related characteristics as well. The result is a recommender system following both memory-based and model-based approaches.

In Section 1 we saw that there are different types of recommenders depending on the technique used for filtering and selecting the relevant models. In DynaLearn, we implement a multi-criteria utility based recommender system [1, 11]. These recommenders make suggestions based on the computation of the utility of each object for the user. In our approach, the objects are the models in the repository (candidates), and the utility function measures the relevance of a candidate reference model with respect to a user model. This relevance is based on the similarity between the models (model-based recommendation) and the user ratings of the reference model (memory-based recommendation). After obtaining the feedback, users can rate the usefulness of the generated feedback. These ratings give, indirectly, an idea of the utility of the model from which the feedback was generated.

The selection follows a heuristic that promotes getting reference models with a significant overlap on the user one. That way, the learner can progress more easily through models of the same domain of increasing level of complexity and completeness.

3.1. Input

When the learner wants to get feedback while building the model, some information is requested that is sent to the Semantic Technologies (ST) component of DynaLearn as input data. This information is the following:

- **Course** (if any): In DynaLearn the models can be organized in courses. These courses are created by teachers for a particular class or subject. Through the User Management System, teachers of that course can associate to it any model from the repository (see Appendix A for more detailed information about courses). When asking for feedback, the learner selects in which course they want to work, hence only the models of that course will be considered for recommendation (they are the *candidate models*). However, if the learner wants to work without a course context, all the available models of the repository will be considered as candidates.
- **Minimum relevance value:** During the recommendation process, we compute the relevance of each available model for the given learner model and we rank the models accordingly. When asking for feedback, the learner selects a minimum relevance value.
 - $0 < \text{Relevance threshold} \leq 1$: the models with relevance higher than the threshold are selected as reference models and taken into account for the feedback. If there are no models with relevance over the threshold, an empty response is sent to the CM (no possible recommendations given the user's preferences).
 - $\text{Relevance threshold} = 0$: all the models in the repository that are minimally related to the learner model (models that have one common grounding at least) are considered as relevant. No filtering is done.

- **Learner model** under construction. This model may be or not already stored in the repository, but it should be a grounded model in order to get relevant recommendations and, therefore, relevant feedback.

3.2. Criteria and computation of relevance

As mentioned before, we select the set of relevant models from the repository according to their relevance with respect to the learner model. This selection has to take into account not only the ratings of users but also the characteristics of the model. However, how can we compute this relevance value for each model following these indications? Since we want to use a multi-criteria utility-based recommendation, two steps are necessary: selecting the criteria for the recommendation and defining the utility function that applies to our case.

3.2.1. Multi-criteria attributes for recommendation

The goal is to find good quality, similar models to the learner model; hence they can presumably provide useful suggestions to the learner. These are the attributes taken into account for each model-model comparison:

- **Number of common groundings (c_i)** between a candidate reference model and the learner model. This measures the overlap of the two models, thus the similarity between them.
- **New concepts in the candidate M_i (d_i)**: To select a model within the user's zone of proximal development, we want to control the size of the candidate M_i and select a model that does not introduce too many changes or new concepts, thus being consistent with our incremental approach when providing feedback to the user.
- **Rating of the model (r_i)**: overall rating of the candidate M_i
- **Author's expertise (e_i)**, that depends on the role of the author in the system (learner, teacher or expert). We assume that expert authors will create better models than beginners. However, we do not want to limit the recommendation to the expert models. Therefore, we include the author's expertise as a new attribute to take into account.
- **Average rating of the author (r_a)**, which is the average rating of all the models created by the same author. This measures the popularity of an author, independently of the expertise degree.

3.2.2. Utility function (or how to get the relevance value)

Once we know the attributes for the recommendation, we need to define the utility function of our recommender. Typically [1], the utility function measures the appropriateness of the recommending item $i \in Items$ to user $u \in Users$ and is defined as $R : Users \times Items \rightarrow R_0$, where R_0 represents a non-negative integer or real number within a certain range [4]. The selected item is then the one that maximizes that function.

In our case, the utility of a candidate reference model is given by its relevance to the learner model, according to the Multi-Attribute Utility Theory [11], the utility function can be generally represented as:

$$R(u_1, \dots, u_n) = \sum_{j=1}^n w_j \bullet u_j$$

Equation 1

Where u_j is a single-attribute utility function over attribute l and w_j is the weight for attribute j and $\sum w_j = 1$ ($0 \leq w_j \leq 1$) .

Given the features listed above, our formula to calculate the relevance of a candidate reference model is the following:

$$Rel_i = w_1 \frac{c_i}{n_m} + w_2 (1 - \frac{d_i}{n_i}) + w_3 e_a + w_4 r_i + w_5 r_a$$

Equation 2

Where n_m is the number of terms in the learner model, n_i is the number of terms in the candidate reference model. We initially assign the same value to all weights. However, in the future these weights should be learnt by using some machine learning technique that automatically calculates the most appropriate values in order to get more accurate relevance values.

3.3. Recommendation process

Now that we have defined the attributes that we want to measure and how to compute the relevance of each candidate M_r , we can describe the process we follow to find the set of relevant models. The algorithm we follow is the next one:

Algorithm 1 *selectRefModel(learnerModel)*, selects the set of relevant models and picks the most relevant model as reference model

Require: Initial set of models (candidates) obtained from the selected course or the repository, *listCandidates*

Require: Minimum relevance selected by the user, *relevanceThreshold*

```

1: refModel  $\leftarrow$  null
2: listCandidates  $\leftarrow$  filterByCommonGroundings ()
3: if (listCandidates is not empty) then
4:   For each candidate  $\in$  listCandidates do
5:     calculateRelevance(candidate)
6:   end for
7:   rankByRelevance(listCandidates)
8:   refModel  $\leftarrow$  getMostRelevantModel(listCandidates)
9:   relevantModels  $\leftarrow$  filterByRelevanceThreshold(relevanceThreshold, listCandidates)
10:  If relevantModels is empty then
11:    refModel  $\leftarrow$  null
12:  end if
13: end if
14: return refModel

```

3.3.1. Filtering of unrelated models by number of common groundings

From the pool of models available in the repository, we first need to filter out those models that are unrelated to the learner model. We consider the existence of common semantic groundings as indicative of the overlap between models (at least partially) in the domain they describe, thus being potentially related semantically. We perform an initial filtering by leaving aside those models with no common groundings with the learner model. The models remaining after the filtering form the set of relevant models.

If no models have groundings in common with the learner model, the recommendation process (and thus the generation of feedback) finishes and an empty response is sent back to the DynaLearn framework (see Algorithm 1).

Algorithm 2 *filterByCommonGroundings()*, from the list of candidates, it filters out those models with no common groundings with the learner model

Require: Initial set of models (candidates) obtained from the selected course or the repository, *listCandidates*

Require: Learner model, *learnerModel*

```

1: relevantModels  $\leftarrow []$ 
2: For each candidate  $\in$  listCandidates do
3:   n  $\leftarrow$  getCommonGroundings(candidate, learnerModel)
4:   If n > 0 then
5:     relevantModels  $\leftarrow$  addModel(candidate)
6:   end if
7: end for
8: return relevantModels

```

3.3.2. Ranking of relevant models

Once we have the set of relevant models, we rank them according to their relevance to the learner model. For each candidate reference model, we calculate its relevance value using the function described in Equation 2. Then, all candidates are ranked based on their relevance.

3.3.3. Filtering of models based on relevance threshold

The models whose relevance value is under the minimum relevance selected by the learner are filtered out, thus removed from the list of relevant models. The resulting set of models after this filtering constitutes the set of relevant models to be used as reference during the generation of feedback.

If there are no models which relevance value is over the threshold, then no models can be recommended and not suggestions generated and sent back to the learner. In that case, the learner should choose a lower value as minimum relevance and try again.

3.4. Output

As result of the recommendation process, we obtain different output elements necessary for the generation of feedback, and thus taken as input for that process. These elements are the following:

- **Set of relevant models**, which corresponds to the source of knowledge used for generating the feedback for the learner
- **Reference model**, the model with the highest relevance from the set of relevant models. This model will be compared to the learner model by means of the ontology matching tool in order to get equivalences between both models.

After the recommendation process a ranking of relevant models is obtained, where the top one corresponds to the model with maximum overlap with the user model and minimum differences. This first model is selected to be used as reference during the generation of feedback.

3.5. Rating of models

We have seen that the relevance formula uses the ratings to calculate the relevance value. Although the relevance depends on other attributes as well, we could say that the highest rating a candidate has, the higher its relevance value. Since only the models with relevance over the threshold are selected, ratings have a direct influence on the selection process. Hence the question here is: how do the users rate the models?

Once the feedback is communicated to the learner, they receive a list of suggestions that they may decide to follow or not, depending on whether they agree or not with it. In the interface, together with the description of each suggestion, there is an option to indicate if the user agrees or not with that suggestion. After reviewing all the suggestions the ratings are sent back to the ST.

Each suggestion has been obtained from a particular model. If the learner agrees with a suggestion, this is translated into an extra positive point to the rating value of the corresponding model, value that is stored in the repository as part of its meta-information (see [10] for details of how the models are stored in the repository).

4. Semantic-based feedback from the set of relevant models

In [12] we introduced our techniques to provide semantic-based feedback for a given learner model. These techniques used a reference model to be compared with the learner model in order to analyze the differences between them and to provide suggestions as feedback. In this early implementation, the reference model had to be provided by the learner, who chose it among the list of models in the repository when asking for feedback. However, this list of models is continuously increasing as users create and store models in the repository. Therefore, the selection of an appropriate model for the feedback becomes a very difficult task for the learner who, furthermore, has no access to the full content of the models.

In this document we presented our recommendation technique aimed to automatically select the relevant models for the feedback. As a result of the recommendation we get not only a model automatically selected to be used as reference model, but also we get a set of relevant models from which we can obtain profitable information. Once this new scenario was possible, we updated our semantic feedback techniques accordingly, introducing also several improvements in the process.

In the rest of this section we describe the new feedback generation in this new context, in which not only one but several models are available as reference.

We perform the comparison of the models considering these two different levels of abstraction:

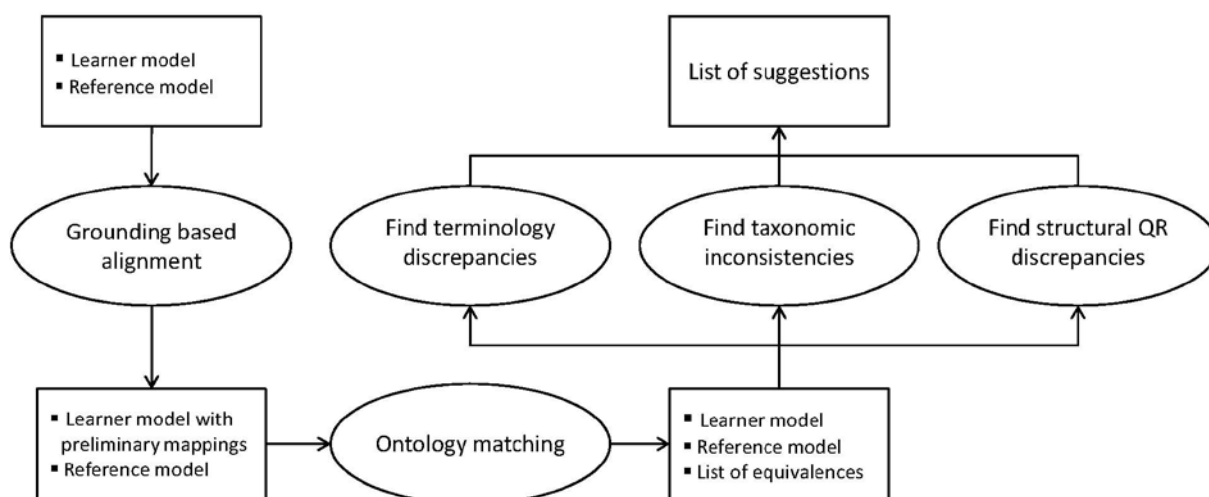


Figure 2: Generation of semantic feedback

Figure 2 shows the different processes that compose the generation of semantic feedback. Firstly, the learner and reference models are compared using a grounding-based alignment, which identifies the common groundings between the two models. These common groundings are used to generate a preliminary set of mappings that are added to the learner model. Then, ontology matching is performed between the enriched version of the learner model and the reference model to provide the final set of mappings. Finally, we process this list of equivalent terms to detect the semantic differences between the models. This detection is carried out through three different processes, according to the type of difference we want to find: terminological discrepancies, taxonomic inconsistencies or structural QR discrepancies. The result is a list of suggestions aimed to solve the detected problems in the learner model.

To illustrate the generation of feedback process we use Figure 3 and Figure 4 as examples of reference model and learner model respectively.

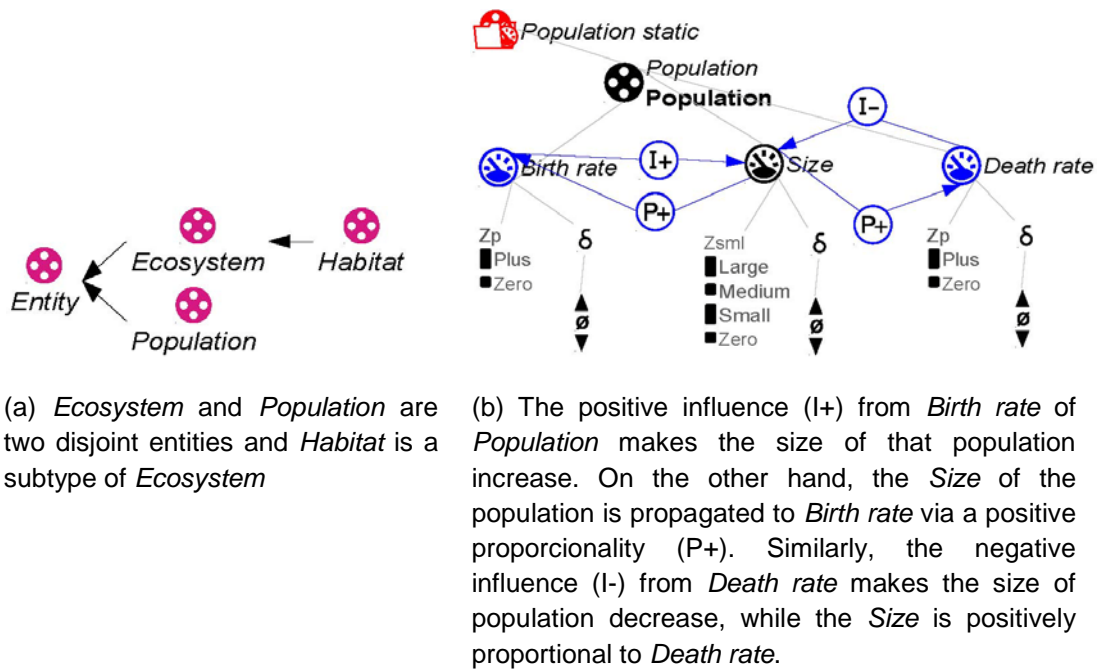


Figure 3: Reference model used in the examples, from *model of plant growth based on exploitation of resources* [15]

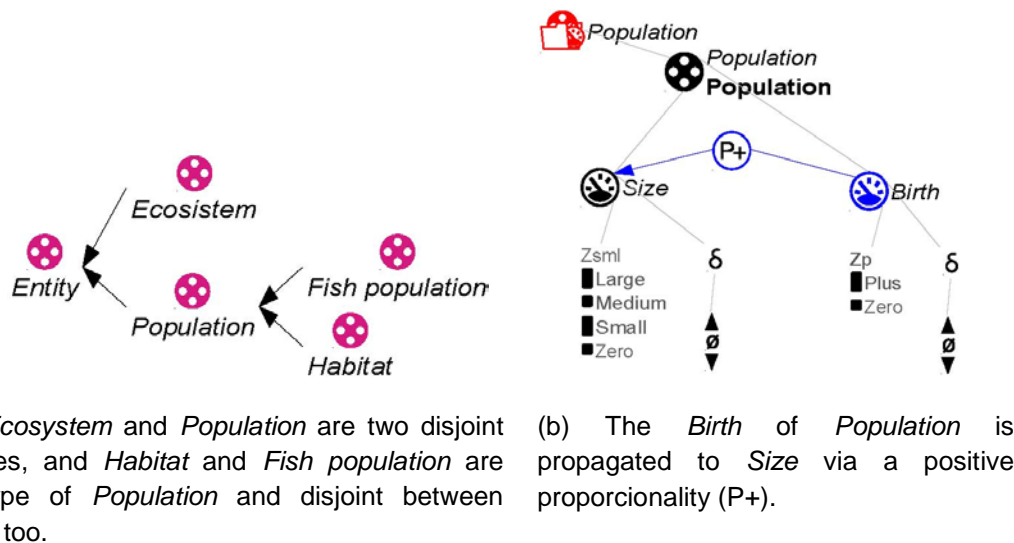


Figure 4: Learner model used in the examples

4.1. Input

As result of the recommendation process we obtain the reference model and the set of relevant models according to the relevance threshold specified by the learner. The input of the feedback component is then the following:

- **Reference model** automatically selected in the recommendation process.

- **Set of relevant models** which relevance value is higher than the minimum relevance indicated by the learner.
- **Learner model** under construction.

4.2. Grounding-based alignment

In order to facilitate the ontology matching of the models, the software first derives preliminary mappings by doing a grounding-based alignment. If two terms from different models are grounded to a common resource, both terms are assumed to represent equivalent concepts. These relations are used to generate a preliminary set of mappings. In our example, the learner model has a term labeled *Birth* that is grounded to the DBpedia term *Birth rate*³, and the reference model has a term labeled *Birth rate* that is also grounded to the same DBpedia resource *Birth rate*. In order to allow later inference, we determine that *Birth* and *Birth rate* are equivalent terms (expressed using *owl:EquivalentClass*). As result, we obtain pairs of terms considered semantically equivalent, that constitute the preliminary set of mappings.

4.3. Ontology matching

The set of preliminary mappings obtained in the grounding-based alignment is enhanced by applying ontology matching techniques [7] between the reference and the learner models. This generates the final list of mappings or equivalent terms. These mappings are added to the learner model as *owl:EquivalentClass* statements (more details can be found in [10, 12]).

4.4. Discrepancies between terminologies

The next step is to analyze the differences between each pair of equivalent terms to find the possible discrepancies in their terminologies. Depending on the type of differences, we find:

- **Discrepancies between labels or groundings:** The system compares the label and grounding of each pair of equivalent terms. In case any difference is found, the information of the reference model is suggested to modify the learner terminology. In our example, we would detect that the quantity *Birth* in the learner model could be renamed as *Birth_rate*.
- **Missing and extra ontological elements:** Those terms from the reference model with no equivalence in the learner model are seen as missing elements and suggested to be added to the learner model. We also indicate their relation with the existing elements in the learner model when possible. In the same way, those terms from the learner model with no equivalence in the reference model could be seen as extra elements to be removed from the model. In our example we have a case of extra ontological element, since the entity *Fish population* does not appear in the reference model and could be removed from the learner model.

³ http://dbpedia.org/resource/Birth_rate

4.5. Taxonomic inconsistencies

After the ontology matching process, we use semantic reasoning techniques [19] to detect inconsistencies between the hierarchies of the two models. Two equivalent entities should share the same terminology but also have the same equivalent position in their respective hierarchies. For instance, the entity *Habitat* cannot be subtype of *Population* in the learner model and subtype of *Ecosystem* in the reference model given that *Population* and *Ecosystem* are disjoint classes. The system informs the user of this inconsistent situation, so they can revise the hierarchy and change it accordingly.

A more detailed description of how we detect taxonomic inconsistencies can be found in the deliverable [8].

4.6. Structural QR discrepancies

Once we have identified the equivalences at the schema level we find out differences in how these concepts are used in the models. We make use of the particular semantics of QR models, as well as some QR modeling rules, to identify common patterns in the models and detect undesired situations.

4.6.1. Extraction of basic units

In a QR model, each entity instance can be associated with different quantity instances. This structure constitutes what we call a **basic unit**, and the type of a basic unit is the type of its entity instance. Quantities in a basic unit can be connected by different relations, as entities can be connected by configurations (still a different type of relation). When the origin and target of a relation belongs to the same basic unit we have an **internal relation** and an **external relation** if the target belongs to a different basic unit (see Figure 5). Notice that configurations are external relations by definition, since they connect two instances of entity, and each instance constitutes a different basic unit.

The first step is to extract all the basic units from both the reference model and the learner model.

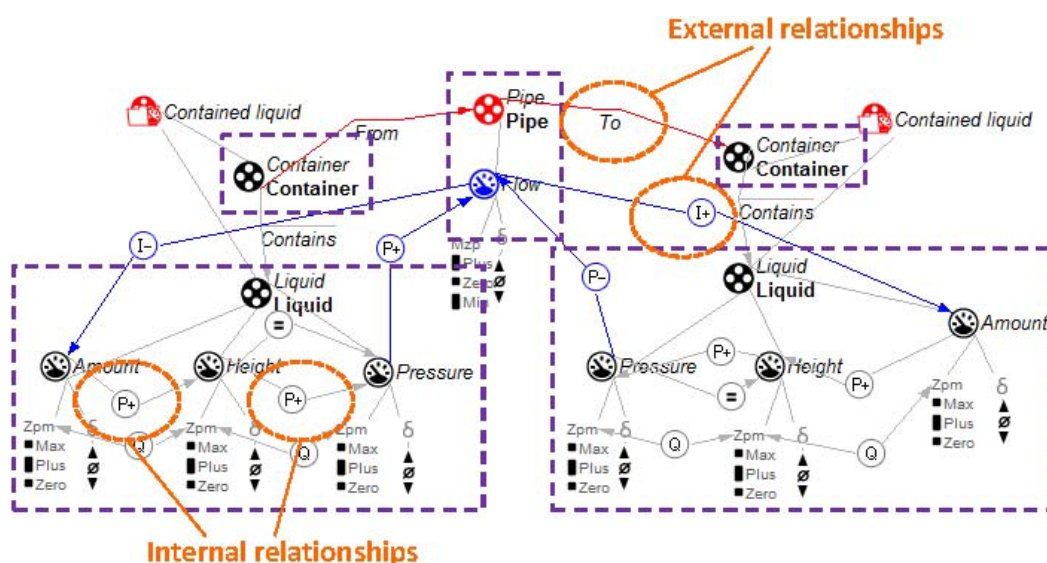


Figure 5: Example of basic units in a model

4.6.2. Integration of basic units of the same type

Now we have the two models represented by their basic units. The next step would be to find equivalent basic units and compare them to get the discrepancies between them. However, for models in learning space 6 (LS6), we need to do an additional step to handle the problem of the modularization of models into model fragments.

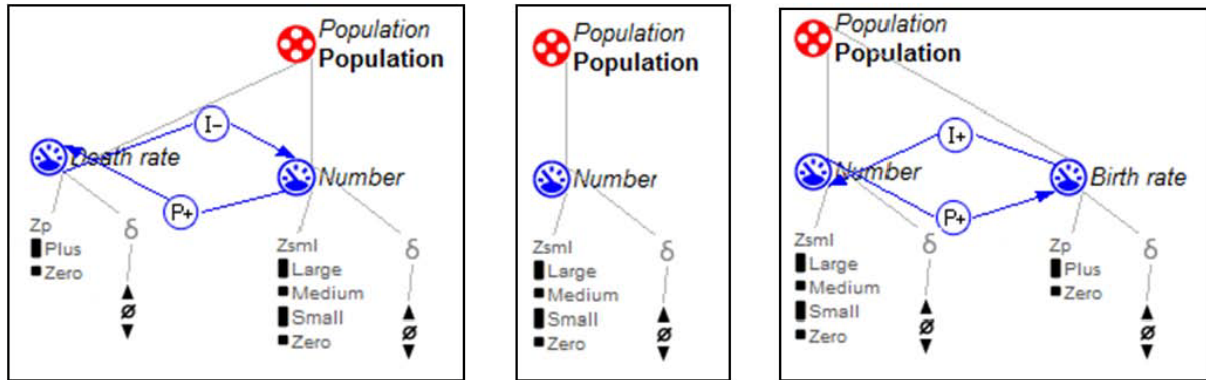


Figure 6: Example of different MFs of a reference model

Models in LS6 are usually represented using multiple modules or *model fragments* (MF). An entity can be used in different parts of the model, generating multiple instances and, therefore, multiple basic units of this entity. Let us assume that we have a reference model that contains the MFs in Figure 6. On the other hand, the learner creates a model about the same topic but representing all in the same MF (see Figure 7). We see that both models represent exactly the same knowledge. However, after extracting the basic units from each model we get three basic units of the type *Population* from the reference model and only one basic unit of the type *Population* from the learner model. If we try to compare their basic units, we will find that the learner model is missing two basic units. How can we detect this situation?

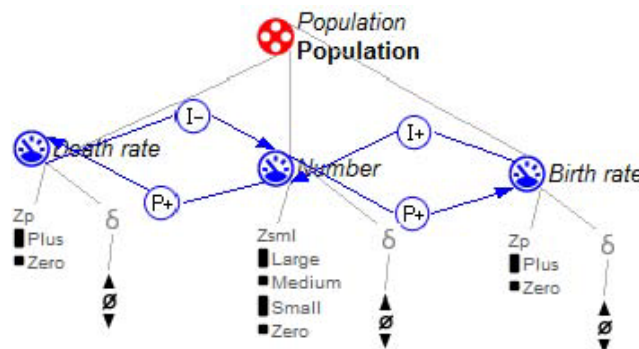


Figure 7: Learner model represented in one MF

To solve this we first integrate all the basic units with the same entity type in one called *integrated basic unit*. For each basic unit of the same type, we get the quantities related to the entity instance and the internal dependencies and add them to the integrated basic units (only if the same elements were not added before). In our example, the integrated basic unit of the type *Population* in the learner model would be the same that the original basic unit. However, in the reference model the three initial basic units would generate a new one that, in this case, would be the same we have in Figure 7.

4.6.3. Comparison of equivalent integrated basic units

Once we have the set of integrated basic units, we have to compare those integrated basic units that are equivalent in the two models. Two integrated basic units are equivalent if and only if their types (their entity descriptions) are equivalent (we found a matching between them).

For each integrated basic unit of the reference model, we get the equivalent integrated basic unit in the learner model. Then, we compare the list of related quantities of both integrated basic units, as well as their internal relations.

4.6.4. Matching of basic units of the same type

Once we have compared the quantities and the internal relations, we need to compare the external relations of equivalent basic units. However, this cannot be done with the integrated basic units, since an external relation occurs between two different basic units that could or could not be of the same type. Through the analysis and comparison of the external relations we can detect those basic units that are missing in the learner model. For this, we first have to match the basic units from the reference model with the learner model ones. Notice that we detect here missing basic units within the same model fragment, as missing origin or target of an external relation. Missing basic units related to missing model fragments are out of the scope of this version.

For each basic unit of the learner model, we compute the differences with each basic unit of the equivalent entity in the reference model. We count as differences the presence of the same quantities in the basic unit, the number of internal relations and the number of external relations. The reference unit with minimum differences is matched to the current user unit and cannot be matched in the following comparisons.

As result of the matching process, we detect missing entity instances in the user model (unmatched basic units in the reference model) and extra entity instances (unmatched basic units in the user model).

4.6.5. Comparison of equivalent basic units

The next step is to compare the equivalent basic units found in the previous step. First, we compare the elements within the basic units and their internal relations. At this point we detect missing and extra quantities in the user's basic units, as well as differences in their internal relations. Two particular quantities should be connected in the same way in all the basic units of the same entity. Thus, we check if this modeling rule is followed or not in the learner model, taking as correct the relations in the reference basic units. Finally, we check if the external relations of the basic units are the same in both models, reporting the differences otherwise.

As result, after the comparison of the models at the instance level we are able to detect the following type of discrepancies:

- **Missing instances of existing terms in the learner model:** A missing instance in the learner model could indicate that the model is incomplete. Let us assume that the learner in our example has already defined the quantity *Death* in their model, though it is not being used. Then, we suggest that, according to the reference model, an instance of *Death* should be added to the entity *Population* in the learner model.
- **Differences in the causal dependencies between quantities:** A difficult task in QR modeling is to correctly add the causal dependencies between quantities (influences and proportionalities). By comparing the causal dependencies between equivalent quantities in the reference and the learner model we detect those relations that are missing, extra or different in the learner model. In our example, we would detect that the quantities *Size* and *Birth* in the

learner model are causally related by a positive proportionality. However, according to the reference model this should be a positive influence and a positive proportionality in the opposite direction.

4.7. Semantic-based feedback from the pool of models

We have seen the new types of feedback and the new approach in its implementation based on a two level comparison of two models (the learner model and the reference model). However, as explained in the previous version what we obtain from the recommendation process is not just a reference model, but a set of models relevant to the learner model. Then, how can we obtain semantic-based feedback from a set of models instead of one?

Once the set of relevant models is selected, the feedback process is run for each relevant model. As a result a list of suggestions is obtained from the comparison between the learner model and each relevant model. Nevertheless, the results can be not only very diverse but also some results could be contradictory. The hypothesis here is, the more common a suggestion is (the same suggestion is generated from different models), the higher confidence we can have on its efficacy. But how is this measured?

We introduce here the idea of **agreement**. Given a particular suggestion, the agreement on it is computed by counting the number of models that generate the same suggestion (the number of models that *agree* on that). For this step, only those relevant models that contain the terms involved in the suggestion are taken into account.

When communicating the feedback, each suggestion is sent together with its agreement value. This information is useful for the learner when considering following or not a particular suggestion.

5. Communication of results

The result of the generation of semantic feedback is a list of suggestions intended to complete or improve the learner model. These suggestions are communicated back to the modeling tool, which shows the results to the user through a particular graphic interface (see Figure 8).

The list of suggestions appears on the left, and can be filtered by the type of ingredient involved. This allows the users to see only the suggestions about entities, or only those about quantities, etc. For each suggestion, the interface shows the matching elements between the user and the reference model that originated the suggestion, and a brief description of the existing discrepancy and how to solve it. In addition, some basic statistics about the entire process are provided, like the number of terms in the model and the percentage that matching terms represents in each model.

After evaluating this feedback the learner can rate the quality of the received suggestions, indirectly rating the model used as reference during the process.

5.1. Graphic User Interface

In the DynaLearn framework, the recommendations on the current model can be obtained under the menu option: Support→Get recommendations on current model. There is also the possibility of running this process together with the virtual characters, which provide additional support by guiding the learner through the graphic interface (see [22] for details about this).

Before running the recommendation process, the user can select the course in which they want to work (if any) and the minimum relevance value to be used as threshold for the recommendation. The results are shown in a window like the one in Figure 8. Through this interface the user can filter the results to be shown by the type of element involved in the suggestion, or the agreement value of the suggestions.

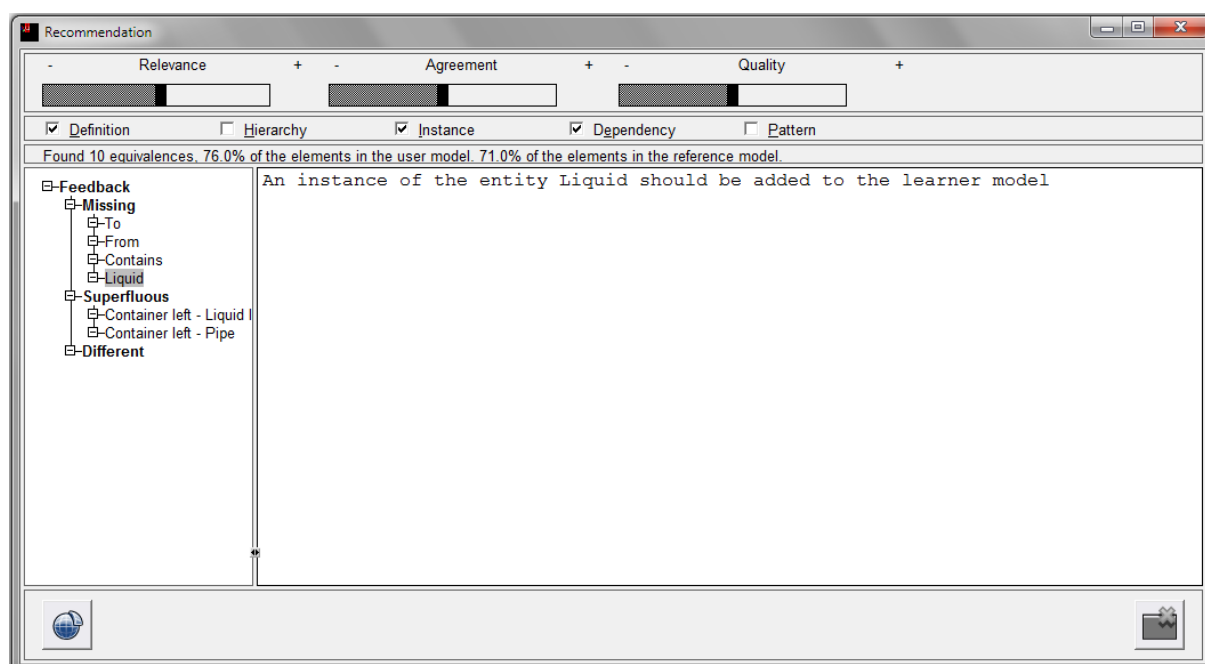


Figure 8: Interface for semantic recommendations

6. Evaluations

To evaluate our approach, we performed two different experiments⁴ to both validate each technique independently, and to make a first attempt to evaluate the entire process. However, these experiments represent a preliminary evaluation, and more experimentation must be done to fully validate the adequacy of our techniques and the usability of the system in a human-computer interaction manner. This will be done in the context of the WP7 evaluations to be performed in the reminder of the project.

6.1. Evaluation of recommendation

We evaluated the recommendation of relevant models based on the current heuristic of maximum overlap and minimum differences (no user ratings were used here). Although more complex heuristics can be used, we wanted to validate a simple approach first that can be used as basis for more elaborated solutions.

We selected a total of 13 learner models and a total of 7 expert models from two different domains. The expert models were stored in our repository, together with unrelated control models. Each learner model was processed through the recommendation component to obtain the ranked list of relevant models and the most relevant as the reference model. Six experts participated in this experiment evaluating all the cases. The expert models were provided to the evaluators, together with the learner models and their corresponding recommendation results. For each case, the evaluators had to indicate whether they agreed with the selected reference model, and whether they agreed with the whole ranking of relevant models.

The results of this evaluation showed that, for the selection of reference models, the experts considered correct the 91% of the cases, with a level of agreement among evaluators (Kappa coefficient) of 0.81, which indicates adequate inter-rater agreement. However, for the evaluation of the entire rankings, only the 61% of cases were assessed as correct, and with a Kappa coefficient of -0.1, which corresponds to almost an equal agreement to chance.

Discussion

The goal of the recommendation of relevant models is to select the most appropriate one as the reference model during the feedback. In that sense, the results of this experiment are very promising since in almost all the cases the first option was validated as the most relevant by the experts. However, results were not as good for the entire rankings. Though the first and last positions were normally accepted, the intermediate positions were difficult to assess, presenting a low agreement among the evaluators. In addition, not all the evaluators had the same criteria to select the most relevant model: the most complete model, the closest model to the learner case, the model with the best structure based on modeling best-practices, etc.

6.2. Evaluation of semantic-based feedback

To evaluate the adequacy of the generation of feedback, we took 28 pairs of expert and learner models in the domain of Natural Sciences⁵. Each pair was used as input for the generation of

⁴ All the test cases and evaluation results of the experiments are available at <http://estherlozano.es/dynalearn/experiments/feedback.htm>

⁵ Obtained as result of real modeling exercises conducted at University of Hull and in Tel Aviv University.

feedback. A total of 510 test cases (suggestions) were obtained and verbalized in natural language (e.g., "The configuration *Have* should be added to the learner model"). A total of 10 experts participated in the evaluation⁶. The experiment was based on an online survey, where the pair of models was automatically assigned; assuring that any test case was at least double evaluated. For each case, the images of the pair of learner and reference model were shown, together with the list of suggestions generated as feedback. Then, the evaluators assessed whether the proposed suggestions were correct or not.

As result, an average of 163 test cases was evaluated by each expert. A 74% of the generated suggestions were considered correct by the experts, which we consider a promising result. We also computed the Kappa coefficient to measure the level of agreement, based on a subset of 427 cases that were at least triple evaluated. The result was a 0.44, which indicates moderate agreement.

Discussion

This evaluation was run with learner models from real modeling exercises [13, 14, 16], while the reference model was created by their teachers (and of course unknown for the learners). This particular group of learners was formed by beginners in the conceptual modeling task in general, and the Qualitative Modeling in particular. The main problems identified in the learner models were:

- Representation of multiple concepts in the same term. For instance, to represent *Water concentration in mucus* as quantity, instead of representing the entities *Water* and *Mucus* and *Concentration* as a quantity of *Water*.
- Representation of concepts using different type of ingredients. For example, *Osmosis* was a quantity in the reference model, though it appeared as entity in some learner models.
- Learner models were not fully grounded. By grounding the terms some of the above problems can be detected, since not satisfactory groundings can be found for the term and learners can then reformulate the labels. In addition, the grounding facilitates the later alignment with the reference model.

Therefore, the learner models presented strong differences with the reference model and very small overlap. This makes the task of ontology matching difficult, thus obtaining few mappings between the models if any. The generated feedback was mainly about extra terms and missing terms with no relation to the existing ones in the learner model.

These results seem to confirm that recommending reference models with a significant overlap (e.g., common groundings) lead to more valuable and understandable feedback to the learner.

6.3. Evaluation of the entire system

Finally, we did a first evaluation of the full ST component, combining the recommendation and the generation of feedback processes. For this experiment we took the same dataset used in the recommendation evaluation (same expert models and same learner models), and the same evaluators. Then, for each learner model, we generated the feedback using as reference the model selected in the recommendation of models. We obtained a total of 55 suggestions. The evaluators assessed whether such suggestions were correct or not. As result, 81% of the suggestions were validated by the experts, what represents a slight improvement regarding the previous feedback experiment. The inter-rater agreement, however, was 0.48, which again indicates a moderate agreement.

⁶ All of them are experts in conceptual modeling, and five of them with a strong background in the modeled domain.

7. Conclusion

We present a novel approach to automatically select relevant models from the pool of available models, for a given learner model. This automatic selection is done through a recommender system that follows a Collaborative Filtering approach. However, instead of taking into account just the collaborative information provided by users (memory-based recommendation), we also take into account the content of the models (content-based recommendation). The result is a hybrid recommendation system that implements a multi-attribute utility function to calculate the relevance of each model in the repository for the given learner model.

In addition, we use the set of relevant model found during the recommendation to generate semantic-based feedback to assist the learner in the conceptual modeling task. We compare the models from two different levels: comparing the models at the schema level to find discrepancies between the terminologies and taxonomic inconsistencies; and comparing the models at the instance level analyzing the particular semantics of the QR models to find differences in their common structures. We finally communicate the feedback as suggestions aimed to improve the model. In order to select the most relevant models for the generation of feedback, we use a utility-based recommendation technique based on the relevance of models to the user model. Our experiment to test the correctness of the generated feedback led to a 74% accuracy, which we consider a promising result.

8. Discussion

As future work, we could apply machine learning techniques to automatically adjust the weights used in the relevance formula for the recommendation of models, to substitute their current manual tuning.

Regarding the type of feedback we generate, there are other model features that could be also compared to report differences (quantity spaces, modularization of models and reuse of model fragments, etc) and that are out of the scope of task T4.3. We would like to explore these options in the future and evaluate the impact that they have in the learning process.

Finally, we will run more experiments to evaluate not just the correctness of the feedback but also its effect on the learning process. The validity of our model selection technique has to be also evaluated with additional tests.

References

1. Adomavicius, G., Manouselis, N. and Kwon, Y. "Multi-Criteria Recommender Systems [book chapter]. In Recommender Systems Handbook: A Complete Guide for Research Scientists and Practitioners." Springer, 2010
2. Balabanovic, M., & Shoham, Y. "Fab: Content-based, collaborative recommendation." 40 (3). 1997
3. Billsus, D., Brunk, C., Evans, C., Gladish, C., & Pazzani, M. "Adaptive interfaces for ubiquitous web access." 45 (5). 2002
4. Bredeweg, B. et al. "DynaLearn – Engaging and informed tools for learning conceptual system knowledge." DynaLearn, EC FP7, STREP Project no. 231526, Description of Work (DoW). 2009
5. Breese, J.S., Heckerman, D. and Kadie, C. "Empirical analysis of predictive algorithms for collaborative filtering." Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pages 43-52. 1998
6. Burke, R. D. "Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-adapted Interaction." pages 331-370. 2001
7. Gracia, J., Bernad, J., and Mena, E. "Ontology matching with CIDER: Evaluation report for OAEI 2011", in Proc. of 6rd Ontology Matching Workshop (OM'11), at ISWC'11, Bonn, Germany, CEUR-WS, October 2011.
8. Gracia, J. et al. "Taxonomy-based collaborative filtering algorithms." DynaLearn, EC FP7 STREP project 231526, Deliverable D4.4. 2011
9. Gracia, J., Liem, J., Lozano, E., Corcho, O., Trna, M., Gómez-Pérez, A., et al. "Semantic Techniques for Enabling Knowledge Reuse in Conceptual Modelling." 9th International Semantic Web Conference (ISWC2010) (pages 82-97). Shanghai, China: Springer. 2010
10. Gracia, J., Trna, M., Lozano, E., Nguyen, T.T., Gómez-Pérez, A., Montaña, C., and Liem, J. "Semantic repository and ontology mapping." DynaLearn, EC FP7 STREP project 231526, Deliverable D4.1. 2010.
11. Huang, S. "Comparision of Utility-Based Recommendation Methods." Proceedings of the Pacific Asia Conference on Information Systems (PACIS 2008). 2008
12. Lozano, E., Gracia, J., Gómez-Pérez, A., Liem, J., van Weelden, C., and Bredeweg, B. "Ontology-based feedback on model quality." DynaLearn, EC FP7 STREP project 231526, Deliverable D4.2. 2010
13. Mioduser, D. et al. "TAU Evaluation of DynaLearn Prototype." DynaLearn, EC FP7 STREP project 231526, Deliverable D7.2.4. 2010
14. Noble, R. "University of Hull Evaluation of DynaLearn Prototype." DynaLearn, EC FP7 STREP project 231526, Deliverable D7.2.2. 2010
15. Nuttle, T., Salles, P., Bredeweg, B., and Neumann, M. "Representing and managing uncertainty in qualitative ecological models." Ecological informatics, 4(5-6):358–366, November/December 2009

16. Salles, P. et al. "FUB Evaluation of DynaLearn Prototype." DynaLearn, EC FP7 STREP project 231526, Deliverable D7.2.1. 2011
17. Sarwar, B., Karypis, G., Konsan J. and Riedl, J. "Item-based collaborative filtering recommendation algorithms." Proceedings of the WWW Conference. 2001.
18. Shih, Y. and Liu, D. "Hybrid recommendation approaches: Collaborative filtering via valuable content information." Proceedings of the 38th Annual Hawaii international Conference on System Sciences, 8(3): 217.2. 2005
19. Sirin, E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., & Katz, Y. "Pellet: A practical OWL-DL reasoner. Journal of Web Semantics: Science, Services and Agents on the World Wide Web", 51-53. 2007
20. Tran, T., Cimiano, P., Rudolph, S., & Studer, R.. "Ontology-Based Interpretation of Keywords for Semantic Search." Pages 523-536. 2008
21. Vygotsky, L. S. "Mind in society: The development of higher psychological processes." Cambridge, Massachusetts: Harvard University Press.1978.
22. Wißner, M., Bühling R., Linnebank F., Beek W., and André E. "Integrated characters with dialogue and tutorial strategies." DynaLearn, EC FP7 STREP project 231526, Deliverable D5.4. 2011

Appendix A: User Management System

This part of the document contains a complete description of the User Management System (UMS), a part of the ST module in the DynaLearn software, as well as the specification of the software requirements.

A.1. Definitions, acronyms and abbreviations

DEF1. **User** – Person that will access the Semantic Repository as well as the UMS system

DEF2. **Role** – A position that a User assumes and which reflects personal skills, abilities, and rights. In DynaLearn we have the following roles:

- a. Guest
- b. Learner
- c. Teacher
- d. Domain Expert
- e. Administrator

DEF3. **Model** – A Qualitative Reasoning (QR) model.

DEF4. **Course** – A unit of teaching which is led by one or more instructors (teachers or professors)

DEF5. **University** – Institution/organization of higher education and research, which grants academic degrees in a variety of subjects.

DEF6. **Anchor Term** – External resources for grounding may not cover all the necessary terms in the models. When this happens users can create their own grounding resources, called anchor terms. These anchor terms are stored in the repository and remain available for future groundings.

UMS – User Management System

ST – Semantic Technology

SR – Semantic Repository

DEF – Concept definition

REQ – Software Requirement

A.2. Overall description

This section gives a high-level description of UMS, focusing on the functions that this performs, as well as its general constraints and the domain model used in this application.

Product perspective

Users are required to authenticate in order to access the Semantic Repository (as well as any other functionality in the Semantic Technology component). The User Management System (UMS) was developed as a part of the Semantic Technology in order to facilitate the task of creation and management of the user accounts. This application takes the form of a web application; it is written in Java and stores the user's profile information in a relational database.

Product functions

The following list summarizes the key features of the UMS. The list is in decreasing order of significance.

- Handling user information by allowing create new users, upgrade and downgrade their roles and change its information
- It is possible also to visualize and delete, with certain restrictions, the models uploaded to the Semantic Repository
- Handling courses information by allowing the user to create, update, or delete a course
- Handling universities information by allowing the user to create, update, or delete a university
- It is also possible to create, update and delete a new anchor term for the Semantic Repository as well as view the complete list of anchor terms
- Finally it is possible to load a new language for the Semantic Technology component

User characteristics

The different user characteristics in this application are closely related to the roles they play, so we have:

Administrator: An administrator has the right to manipulate all the attributes of other accounts. He has explicitly forbidden the access to the Semantic Technology from the workbench. Thus, no models can be created under an administrator's account, and its sphere of access is entirely inside of the UMS.

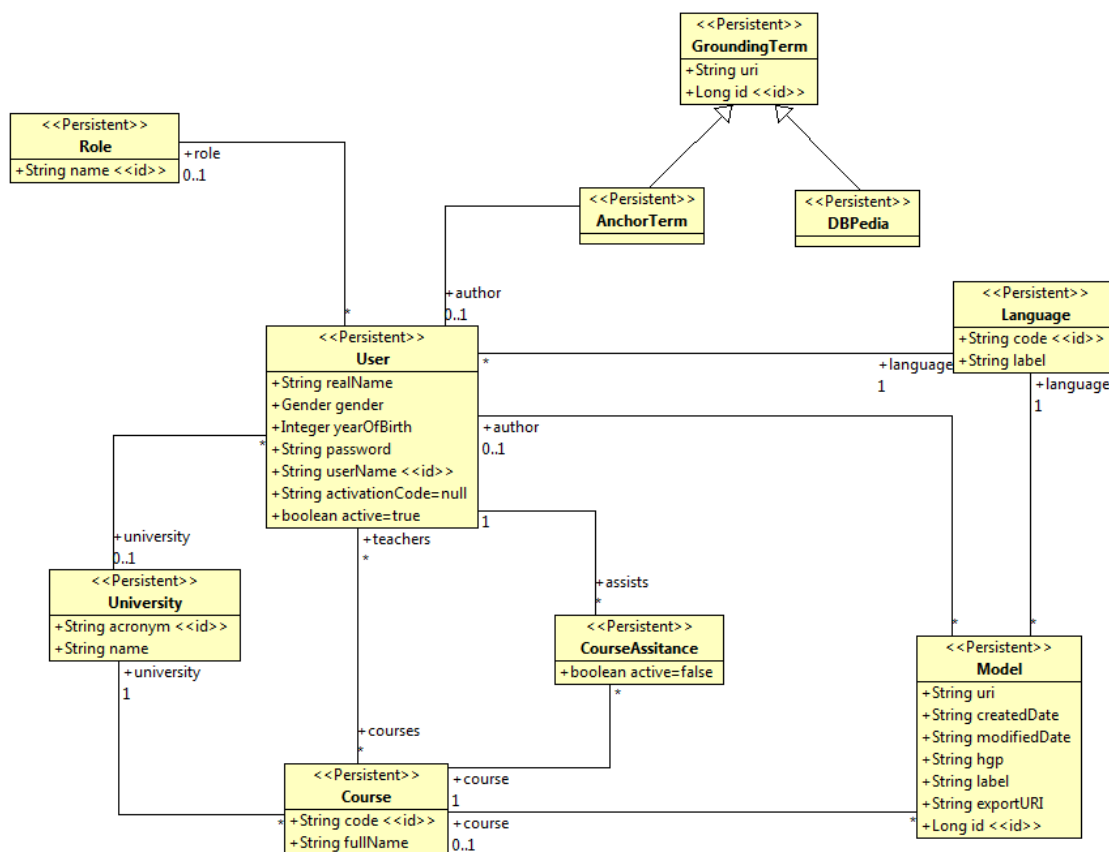
Teacher: A teacher can access the UMS and can create new accounts for learners. A teacher is allowed to use features provided by the Semantic Technology component from the workbench. A teacher can choose to assign a TA (Teachable Agent) flag to a model and thus to allow learners to use the specified model when operating with the teachable agent in the workbench.

Domain expert: A domain expert has similar rights as a teacher. Domain experts, however, cannot create new accounts for learners.

Learner: A learner is not allowed to access the UMS. A learner can upload new models and can load all the models that were previously stored under his/her account. Under this role it is also possible to load any model with a TA flag. Learners are not allowed to see the content of these models though.

Domain model

The controlled objects for this application are those defined before. The data model diagram for UMS is:



Constraints

There is a strict authorization policy in this system, that is, not all users are allowed to do any functionality. Each role has a precisely specified set of rights. This is better explained in the Security Requirements section.

A.3. Specific requirements

This section gives more details about the functional aspects of the system.

A.3.1. Functional requirements

Functional requirements are categorized by the controlled object.

User management

REQ-1: The system shall allow anyone to create a Guest user.

- Only an e-mail, full name, gender and language are required to create a guest account. Optionally he can also apply for a course.
- An e-mail to activate the guest account is sent.

REQ-2: The system shall activate a Guest user once the email confirmation has been set.

- In this moment the new guest user define his password

REQ-3: A user can Login the UMS application by setting his "login name" = e-mail and password

REQ-4: The system shall allow a user to reset his/her password without authenticating into the system, if the user missed the password.

REQ-5: The system shall allow a user to modify his information.

REQ-6: The system shall display a complete list of users.

REQ-7: The system shall allow updating user information, as well as down/upgrade the role.

REQ-8: The system shall allow deleting a user.

Model management

REQ-9: The system shall display the list of models.

REQ-10: The system shall allow assigning models to courses.

REQ-11: The system shall allow deleting models form the Sematic Repository.

Course management

REQ-12: The system shall allow creating new courses in a university. Any teacher can create courses in his/her university.

REQ-13: The system shall allow enabling/disabling a course.

REQ-14: Teachers belonging to a university can teach on different courses.

REQ-15: The system shall display the list of courses

REQ-16: The system shall allow editing the course information

REQ-17: The system shall allow deleting a course

REQ-18: The teacher can assign a learner to a course or validate the request of a learner to be assigned to a course.

University management

REQ-19: The system shall allow creating a new university. The administrator is the only one that can create a university.

REQ-20: The system shall display the list of universities.

REQ-21: The system shall allow selecting the teacher and courses that belongs to that university.

REQ-22: The system shall allow editing university information.

REQ-23: The system shall allow deleting a university.

Anchor terms management

REQ-24: The system shall display the list of anchor terms.

REQ-25: The system shall allow editing the anchor term value.

REQ-26: The system shall allow deleting an anchor term

Languages management

REQ-27: The system shall allow loading a new language for the Semantic Technology component.

A.3.2. Authorization requirements

This section describes the actions that can be performed by a user with certain objects: users, models and anchor terms.

User management authorization

	Owned				Others			
	R	C	U	D	R	U	D	RL
guest	✓	✓	✓					
learner	✓	✓	✓					
teacher	✓	✓	✓		g, l, t	g, l, t		RL(g,l,t)

domex	✓	✓	✓		g, d	g, d		RL(g,d)
admin	✓	✓	✓	✓	all	all	all	RL(all)

Model management authorization

	Owned				Course			Others		
	R	C	U	D	R	U	D	R	U	D
guest	✓	✓	✓							
learner	✓	✓	✓							
teacher	✓	✓	✓		✓					
domex	✓	✓	✓					✓		
admin	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Anchor term management authorization

	Owned				Others		
	R	C	U	D	R	U	D
guest	✓	✓	✓	✓*	✓	✓	
learner	✓	✓	✓	✓*	✓	✓	
teacher	✓	✓	✓	✓*	✓	✓	
domex	✓	✓	✓	✓*	✓	✓	
admin.	✓	✓	✓	✓	✓	✓	✓

*: The anchor terms can only be deleted only if the anchor is used only in one model and this model is owned by the user.

A.4. Use cases

In this section you will find the most representative use cases in UMS application.

A.4.1. User creation use case

Any person (see use case in Figure 9) can access a registration form. This form asks the email of the person, the full name and if he/she likes to apply for a course (see Figure 10).

At that moment, a guest user is created but not activated yet. Then, the UMS sends an email to the user's email address to verify the email (see Figure 11) and ask to the user to follow a link to activate the user and ask for a password (see Figure 12).

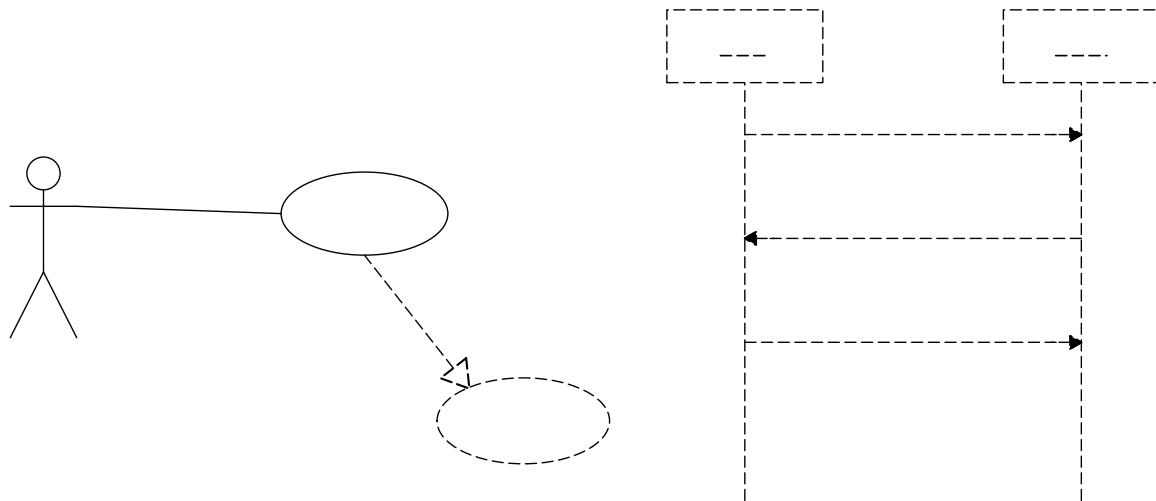


Figure 9: User creation use case

Full name	<input type="text"/>
Email:	<input type="text"/>
Country:	<input type="text" value="Spain"/>
Apply to the course:	<input type="text"/>
<input type="button" value="Submit"/>	

Figure 10: Registration form

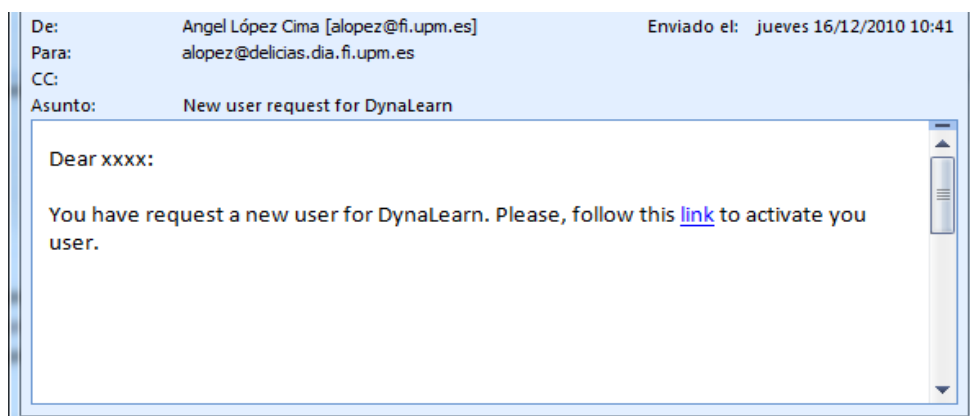
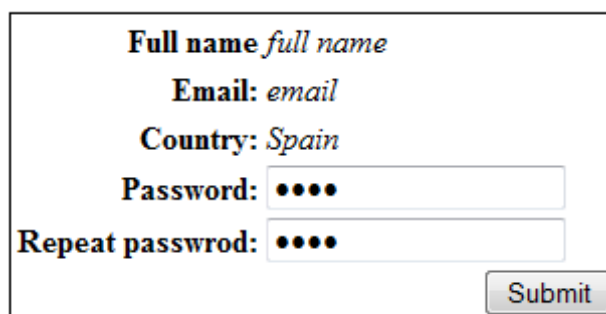


Figure 11: Verification email for a new user



Full name *full name*
 Email: *email*
 Country: *Spain*
 Password:
 Repeat password:
 Submit

Figure 12: User password form

A.4.2. Application of a user to a course use case

In this use case (see Figure 13) the user applies to use DynaLearn in a course, the system stores the application. In this way, a teacher can access to a course management section in UMS (only in the courses that he/she teaches) and verify or reject the applications to that course (see Figure 14).

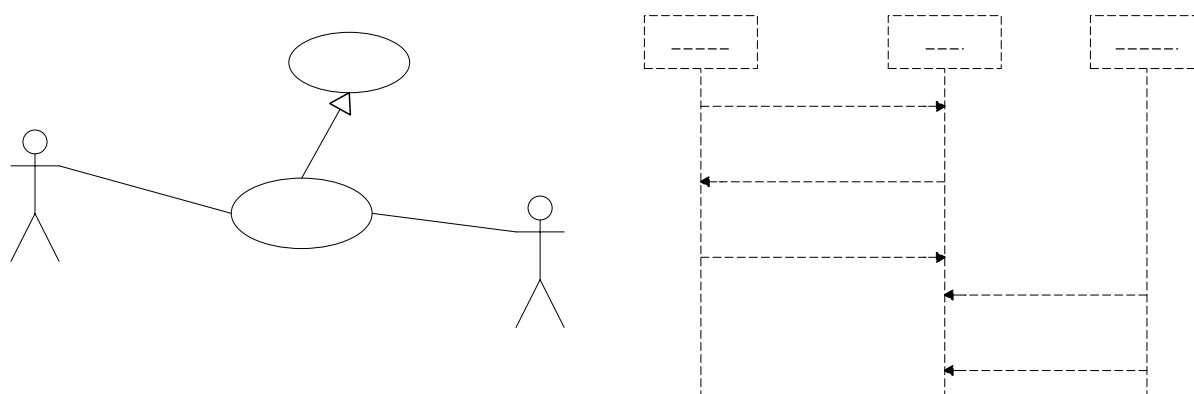


Figure 13: Learner user creation and application to a course use case

User name	Email	Creation date	Deactivation date	Action
User 2	user2@email.com	13/12/2010 10:08	--	Activate
User 5	user5@email.com	13/12/2010 10:07	--	Activate
User 4	user4@email.com	13/12/2010 9:19	--	Reset password Deactivate
User 1	user1@email.com	13/12/2010 10:07	--	Reset password Deactivate
User 3	user3@email.com	03/12/2010 10:02	07/12/2010	Activate
User 6	user6@email.com	23/07/2009 18:35	30/09/2010 08:35	Activate

Figure 14: List of learners in a course and links to verify the application (activate) or reject (deactivate)

A.4.3. Up/downgrading a user use case

A user (see Figure 15) with rights to upgrade or downgrade a user role can do this action by selecting the user from a list and updating his/her role (see Figure 16).

In the case of up/downgrading a learner, the system removes the links to a course of this user.

In the case that the user is set to a teacher, UMS asks in which university is working.

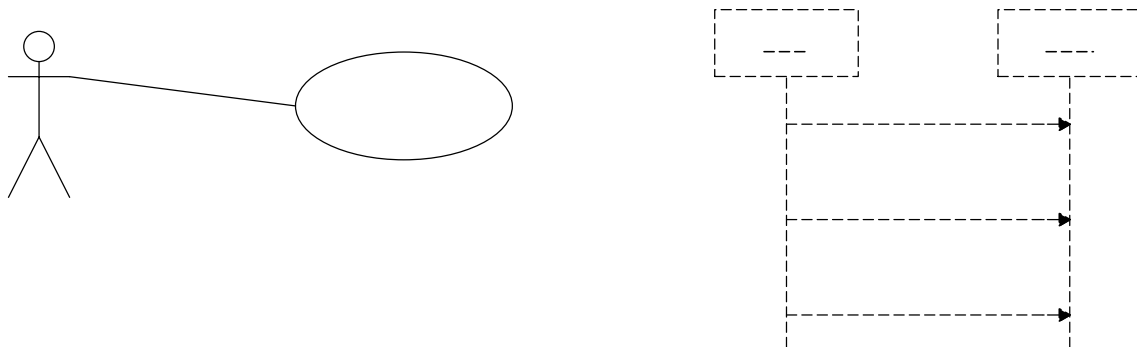


Figure 15: User upgrading and downgrading use case

Full name	<i>full name</i>
Email:	<i>email</i>
Country:	<i>United Kingdom</i>
Role:	From <i>Guest</i> to <input type="text" value="Student"/>
<input type="button" value="Submit"/>	

Figure 16: Upgrade and downgrade user form

A.4.4. Request a new password use case

For a user who misses his/her password, UMS shows a link in the login form to request a new password. Given the email of the user, UMS sends to that user an email with a link to set a new password, similar to user activation in A.4.1 (see Figure 11 and Figure 12).

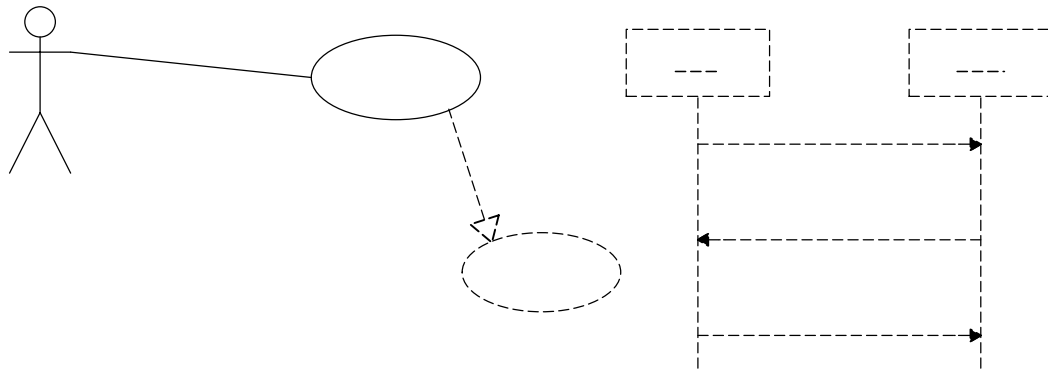


Figure 17: Request for a new password use case

A.4.5. Course management use cases

There are two use cases to manage courses in the UMS (see Figure 18). In the use case "Create Course", the teacher creates a course in a university (see user interface in Figure 19), and the other use case, "Remove Course", the teacher removes the course (see user interface in Figure 20). The consequence of removing a course is that all learners are removed from this course and all models miss the reference to this course.

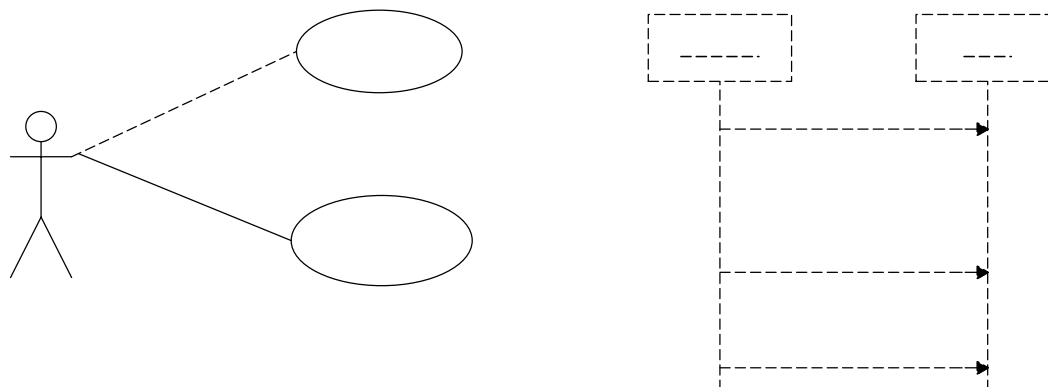


Figure 18: Course management use cases

Create a course in the university xxxx

Course Name:

Course Code:

Figure 19: Create a new course form

Courses in the university xxx

Course name	Creation date	Code	Teachers	Students	Action
Course 1	01/09/2009 08:00	BR7YKL	Teacher 1 , Teacher 2	Student 1 , Student 2 , Student 3	Delete
Course 2	01/09/2009 08:01	COUR02	Teacher 2 , Teacher 3 , Add teacher	Student 4 , Student 5	Delete Manage students

Figure 20: List of courses in a university with links to request to remove them

A.4.6. University management use cases

There are two use cases to manage universities in UMS (see Figure 21). In the use case "Create University", the administrator creates a university (see user interface in Figure 22), and the other use case, "Remove University", the administrator removes the university (see user interface in Figure 23). The consequence of removing a university is that all courses in that university are removed (with the consequences described in A.4.5) and all learners and teachers are move to guest users.

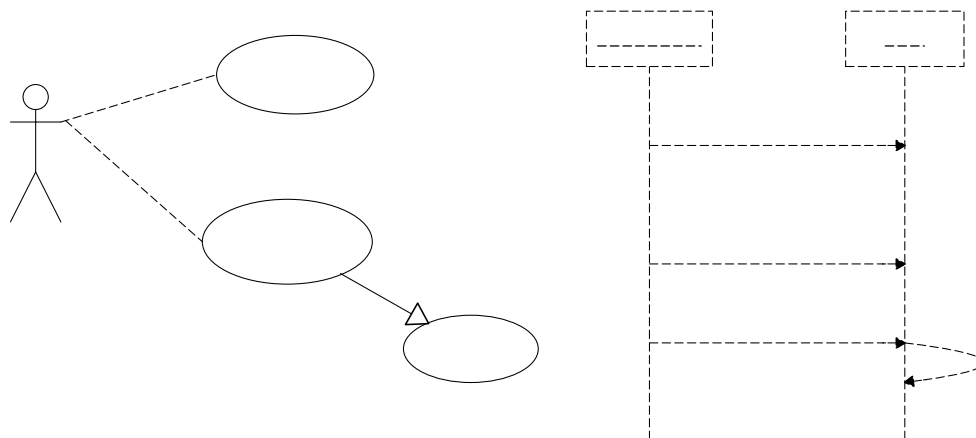


Figure 21: University management use cases



Create a university

Name:	<input type="text"/>
Acronym:	<input type="text"/>
Country:	<input type="text"/>

Figure 22: Create a new university form

Courses in the university xxx

Univ. name	Acronym	Teachers	Action
Univ 1	Univ1	Teacher 1 , Teacher 2	Delete Add teacher
Univ 2	Univ2	Teacher 3 , Teacher 4	Delete Add teacher

Figure 23: List of universities with links to request to remove them

A.4.7. Anchor term use cases

Delete and update of an anchor term can be performed within the UMS (see use cases and sequence diagram in Figure 24 and user interface in Figure 25).

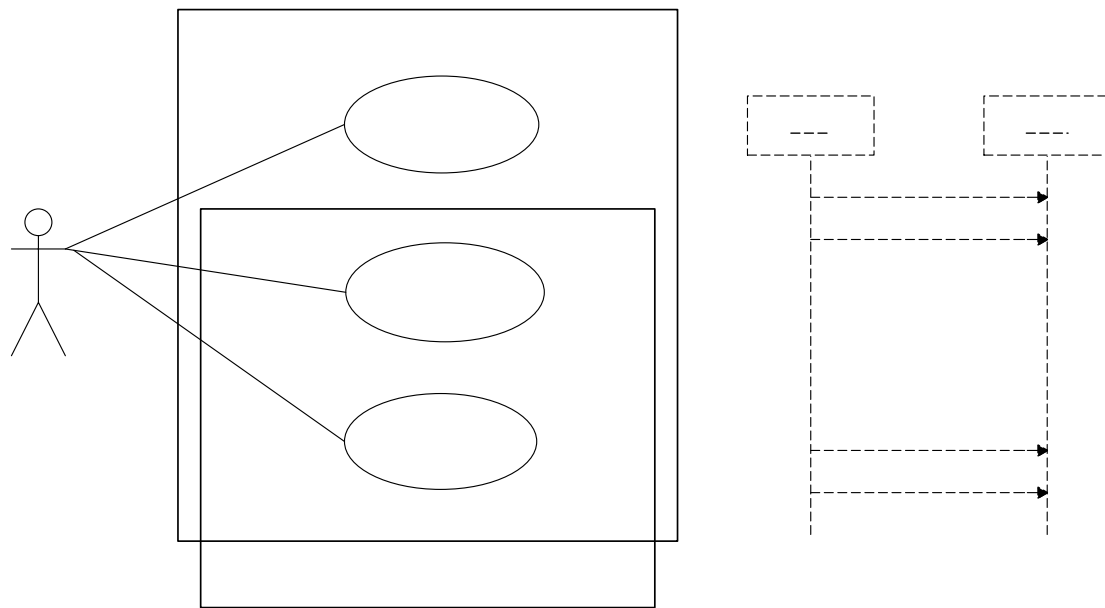


Figure 24: Anchor term use cases

Anchor Terms

Anchor term			Owner	N° of models	Action
Label	URI	Description			
Label1@en	http://dynalearn.eu/anchorTerm#Label1	Description1, description1, description1	User 1	1	Update
Label2@en	http://dynalearn.eu/anchorTerm#Label2	Description2	You	3	Update
Label3@es	http://dynalearn.eu/anchorTerm#Label3	Description3, description3, description3, description3	User 2	2	Update
Label4@pt	http://dynalearn.eu/anchorTerm#Label4	Description4, description4	You	1	Update , Delete

Figure 25: List and action of anchor terms

e-mail:
website:

Info@DynaLearn.eu
www.DynaLearn.eu

