DIGITAL LIBRARIES AND TECHNOLOGY-ENHANCED LEARNING

TARGET

Transformative, Adaptive, Responsive and enGaging Environment

European Commission Seventh Framework Project (IST 231717)

Deliverable D3.3:

TARGET Navigator Services

Document ID: 3.3

Workpackage: 3

Version: V 3.2

Author(s): Ioana Hulpus (Cyntelix), Stefano Bocconi (Cyntelix)

Main Contributor(s): Manuel Fradinho (Cyntelix)

Date: 30.05.11

Status: Final

Dissemination Level: CO

REVISION HISTORY

Date	Version	Author/Contributor	Comments
18.06.09	V1.0	Cyntelix	Initial version of the document.
25.06.10	V2.0	Cyntelix/All	Added baseline and revision of the document
25.05.11	V3.0	Cyntelix	Full rewrite of chapter 3 adding details on CBR, collaborative filtering and user profiling
28.05.11	V3.1	Cyntelix	Rework based on comments from internal reviewers
30.05.11	V3.2	Cyntelix	Explicitly describe the changes in the executive summary. Add a conclusions section. Minor corrections

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The aim of Task 3.5 is to develop the TARGET Navigator Services which will provide three levels of recommendations of TARGET stories (simple, personalized and adapted). The nature of the resulting deliverable D3.3 is a software prototype, consequently the purpose of this document is to support the software with some additional information that expounds on the motivation, background and technical overview of the TARGET Navigator Services. The outlined technical overview will include the architecture, API and installation instructions.

The current version of this deliverable is a resubmission of the version submitted at M18, as required by the reviewers. Their comment was that the deliverable needed to "improve the description of how case based similarity and collaborative filtering is carried out in TARGET". Therefore chapter 3 has been rewritten to include details and formulas on how Case Based Reasoning and Collaborative Filtering techniques are implemented. For each of the recommendation type we provide formulas that are either based on collaborative filtering or Case Based Recommendation principles or both. For the similarity measure required by Case Based approached we introduce a distance metrics for competences based on the CBKST theory as used in TARGET and described for example in D4.4. Furthermore we included and explanation on how Case Based Reasoning supports the construction of a user profile to support recommendations.

1 INTRODUCTION

The TARGET Navigator Services form a component in the TARGET platform used by the Knowledge Ecosystem Navigator (KEN). As shown in the conceptual diagram of Figure 1, the KEN (D7.3) engages with the TARGET Knowledge Ecosystem, which consists of the Target Navigator Services, interacting with the Knowledge Ecosystem Sharing Services (D3.2) for storing and retrieving knowledge assets from the TARGET Knowledge Ecosystem Repository (D3.4).

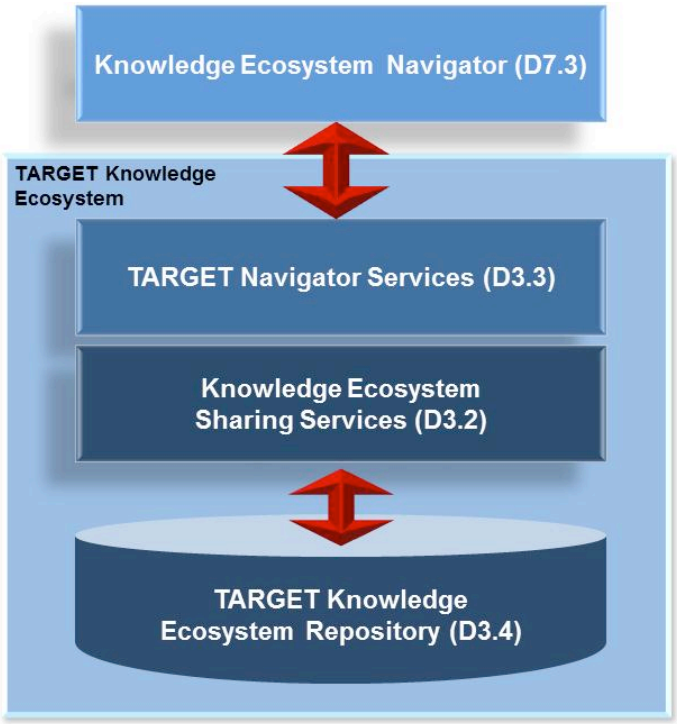

Figure 1 Positioning of the TARGET Navigator Services in relation to other components of TARGET platform

The TARGET Navigator Services (in short TARGET Navigator or TN) provide services which the KEN can use to support navigation by searching and navigation by recommendation.

1.1 Navigation by Searching

This type of navigation is initiated by the learner when using the KEN, by submitting competence-based queries. The user can query for stories that have (or have not) been played by a particular user, that match any, all or none of the competences in a particular set. In this type of navigation, the TN combines low-level search functionalities provided by Knowledge Ecosystem in order to offer higher level search capabilities. An example is filtering the stories the user has already played from the results of the *retrieveStoriesByCompetences* functionality of the KE.

1.2 Navigation by Recommendation

In this context, the TARGET Navigator behaves like a recommender system, capable of making personalized recommendations to the current user. In this process the TARGET Navigator is responsible for making relevant queries to the KE, analyzing and ranking the results and returning to the KEN a personalized list of recommendations for the particular user. For the implementation of this feature, we adopt a hybrid methodology, inheriting from both collaborative filtering and case based reasoning techniques, which are described in the following section in more detail.

2 BACKGROUND ON RECOMMENDER SYSTEMS

Recommender systems have become very popular in the current context of the World Wide Web (WWW). The TARGET knowledge ecosystem is meant to grow continuously with the interaction of the TARGET community of learners and mentors, and this raises a significant challenge concerning the visualization of the data. Therefore, to facilitate the navigation of the knowledge contained in the ecosystem, the TARGET Navigator will provide different recommendation services. The most common recommender systems can be grouped based on two main approaches:

The first approach is the **content-based filtering** where the system matches the user's requirements to the content it has and recommends the content which matches best these requirements.

The second approach is based on **collaborative filtering** which constructs groups of like-minded users with whom the target user shares similar interests, and makes recommendations by analyzing these interests and the selection behavior and history of users (Tang & McCalla, 2009).

In the following subsections we shortly summarize these types of recommender systems, pointing out the similarities and differences between approaches and we justify our approach for the TARGET Navigator.

2.1 Content-Based Recommendation Systems

As stated earlier, content based recommendation systems analyze item descriptions to identify items that are of particular interest to the user (Pazzani & Billsus, 2007). Therefore, content-based recommendation systems rely on the data representation of the items, and various recommendation algorithms have been developed for each representation. For example, text documents can be represented by keywords that describe the content. Recommendations for such documents can be made by matching the keywords with the user's interests. A common representation for text documents is the vector space representation where stemming is used to retrieve the base of each word, and for each such stem a weight is computed. This weight is a value which suggests how relevant the term is for the document, and it is computed based on term-frequency and inverse document frequency (Pazzani & Billsus, 2007). Another type of representation might be based on item databases, where the detailed features of items are stored in form of table rows, and the recommender tries to match the values for the features with the user's preferences.

Therefore, "content-based recommendation" is an umbrella term covering many types of systems, each one suitable for specific domains and recommendation items. The more specific recommender type we are interested in for the purpose of TARGET Navigator, is the "case based recommender", (Smyth B., 2007), (Bridge, Goeker, McGinty, & Smyth, 2006) which is a particular style of content-based recommendation systems. It relies heavily on similarity knowledge, machine learning and reasoning, being inspired by the case based reasoning (CBR) (Aamodt & Plaza, 1994) artificial intelligence paradigm.

2.1.1 Case Based Recommendation Systems

Case based recommenders have their origins in Case based Reasoning (CBR) techniques. CBR systems rely on a case base of past problem solving experiences. The typical representation of cases has two parts, the problem description and the solution part, which describes the solution used to solve the problem. The main hypothesis these systems rely on is that similar problems tend to have similar solutions. A new problem is solved by retrieving a case whose specification is similar to the current problem, and then by adapting its solution. Figure 2 describes the typical CBR process, consisting of 4 steps:

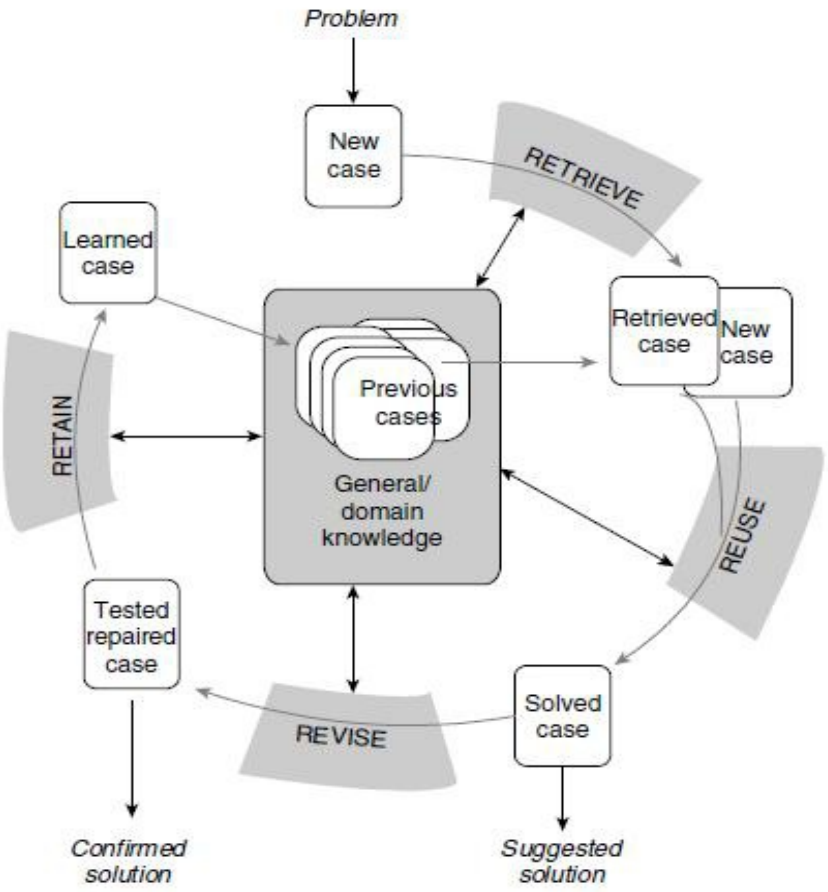

Figure 2 Overview of Case Based Reasoning process (Aamodt & Plaza, 1994)

RETRIEVE: at this step, the system retrieves from the case base all the past cases similar to the current problem description.

REUSE: during this stage, the system decides what of the retrieved solutions will be reused;

REVISE: at this step, the system adapts the reused solutions to perfectly match the current problem;

RETAIN: after the new solution is executed, the whole case (problem-solution) is saved in the case base for future retrieval;

Among the content-based filtering techniques, the case based recommenders hold the advantage of being a lazy-learning technique. This means that reasoning is delayed to run-time, bringing benefits in dynamic environments where data is changing continuously. The alternative eager approaches have the disadvantage that models (e.g. d-trees or neural nets) can quickly go out of date. Also the lazy approach of CBR has the advantage that it can model local phenomena better than eager learning techniques that tend to focus on more global models (Hayes, Cunningham, & Smyth, 2001).

2.1.2 Case Based Recommenders and Personalization

The capability of case based recommendation systems to learn puts them above other content-based recommender systems which aim to give personalized recommendations. Smyth identifies in (Smyth B., 2007) two different personalization approaches, which he names “weak personalization” and “strong personalization”. Weak personalization is the approach of the systems which reacts to the personal needs of the user, but the information is discarded as soon as the session ends, due to the absence of a persistent user profile. On the other hand, the so-called

"strong personalization" approaches have persistent user profiles, which evolve as the user uses the system.

A case based recommender system provides the facility of storing user profiles. Such a user profile can be made up of a set of cases (experiences of the user with the system), plus some indication whether the solution has worked (user liked or disliked or more specifically, in TARGET, the learning outcomes). These profiles can then be used in different ways to influence subsequent recommendations for the specific user, but for other users as well. This type of profiling has similarities with collaborative filtering profiling, described in the next section, but a major difference is that pure collaborative filtering systems are content-free, in the sense that they are agnostic of the item description. They only care about the item's ratings. On contrast, the case based profiles rely heavily on information about the items themselves (Smyth B., 2007).

One case based recommender system which uses case based profiling of users is CASPER described in (Smyth, Bradley, & Rafter, 2002) and (Bradley, Rafter, & Smyth, 2000). It is an online recruitment system that uses recommendation to suggest jobs to users.

2.1.3 Similarity and Diversity in Case based Recommenders

CBR systems retrieve cases with similar description and then adapt the solutions to the new particular problem. Still, a distinction has to be made between the ways case based recommenders approach similarity. The system needs to retrieve similar cases in order to suggest suitable recommendations, but the recommendations themselves need not to be similar, on the contrary, they have to be diverse. In the context of the TARGET Navigator, this means that although reasoning on how story recommendations are made is based on learner similarity, one learner should not be recommended stories with very little differences. The recommended stories have to be diverse so that the learner has a whole spectrum of possibilities. Therefore a system needs to have techniques for achieving both, similarity and diversity, respectively.

For similarity assessment in CBR systems, various algorithms have been used. One of the simplest forms of similarity is the computation of the Euclidean distance, with the formula shown in Equation 1. This measure is suitable for numerical values where normalization is not needed: the smaller the distance between the two items, the most similar they are. Other popular similarity metrics are the cosine similarity (Equation 2) and the Pearson correlation (Equation 3).

The cosine similarity is a normalized distance as the values can be in the [-1, 1] intervals, where -1 means completely opposite, 0 indicates independence 1 indicates full similarity, and all the other values show the intermediate similarity or dissimilarity. This measure is frequently used to compare documents, where the two vectors p and d are usually the term frequency vectors of the documents.

The Pearson correlation coefficient is also a normalized measure whose values range between -1, which means perfect negative linear relationship, and 1 which means perfect positive linear relationship. This measure shows how two sampled sets correlate, being used frequently in determination whether two users have similar behavior and preferences, for example, if their ratings correlate. For example, two users having the ratings 1,2,3 and 2,5,6 respectively for the same 3 products, will have a Pearson correlation coefficient computed with the formula in Eq3, as 0.96. This is a very high correlation, even if the numbers themselves are not similar, but because of the linear relationship between them (both increasing at an almost equal rate).

The previous measures are suitable mainly with numerical data, but depending on the domain, the features of items might be impossible to represent numerically. One example from literature (Smyth B., 2007) is the tourism domain, where vacations' destinations cannot be represented numerically. In this case, similarity trees based on ontologies are an option for features, as shown in Figure 3.

$$d(p,q) = \sqrt{(p1-q1)^2 + (p2-q2)^2 + \ldots + (pn-qn)^2}$$

Equation 1 Euclidean Distance between points p (p1,p2...,pn) and q (q1, q2...,qn)

$$\text{Sim}(p,d) = \frac{p1*p2+q1*q2}{\sqrt{p1^2+q1^2}*\sqrt{p2^2+q2^2}}$$

Equation 2 Cosine similarity between points p (p1, p2) and q (q1,q2)

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{ns_x s_y}$$

Equation 3 Pearson correlation for n measurements of X and Y written as xi and yi

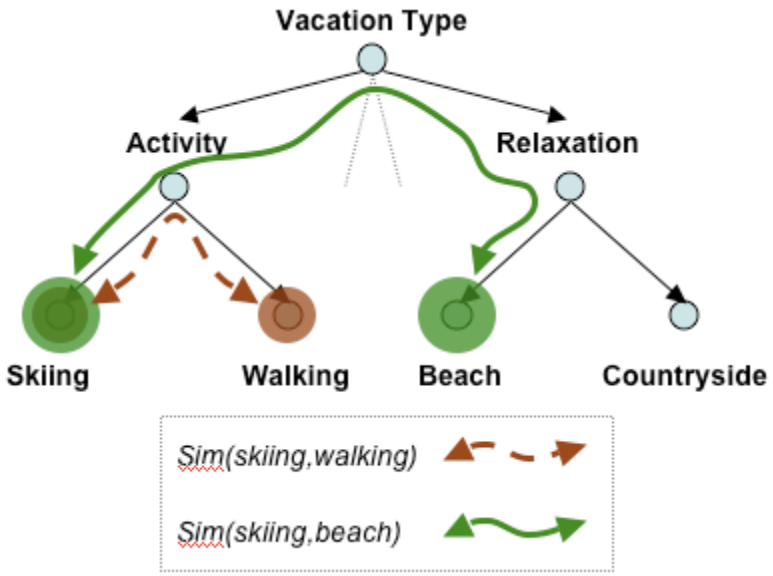

Figure 3 Example of similarity tree based on a partial ontology of vacation types, taken from (Smyth B., 2007)

Because similarity-based retrieval strategies have the danger of producing recommendation sets that lack diversity and therefore limit the user's options, there is research being carried on, which explores alternatives that improve the recommendation diversity while maintaining query similarity. One such option is to retrieve the top ***bk*** most similar cases, and then randomly select *k* cases out of them. This is called "bounded random selection" strategy (Smyth & McClave, 2001). Another strategy is the "bounded greedy selection", introduced as well in (Smyth & McClave, 2001). The core of this strategy is the computation of a *quality* metric, which combines diversity with similarity. It is described in detail in (Smyth B., 2007). Another approach is proposed in (McSherry, 2003). This approach is "compromise-driven" as it relies on the hypothesis that the compromises the user is prepared to accept are not always relevant in similar cases. Therefore the retrieval in such systems is based on the assumption that a case is more acceptable than another if it is more similar to the user's query, and it involves a subset of the compromises of the other case.

In TARGET we base recommendations on similarity. If the user evaluation of the TARGET Navigator performances will show too little variation in the recommended results, we will consider implementing a diversity-preserving algorithm.

2.2 Collaborative Filtering Recommender Systems

In (Hayes, Cunningham, & Smyth, 2001) the authors summarize the idea of collaborative filtering using Figure 4.

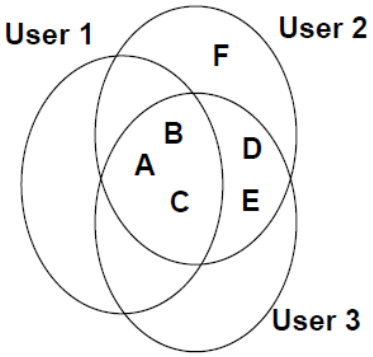

Figure 4 Three users with their interests in assets ABCDEF

The figure illustrates 3 users, which have all shown interest in assets A, B and C (for example they all bought books A, B and C). Having this high overlap, the system can decide that the users have similar tastes/interests. Therefore, it can then recommend D and E to user 1, since these assets were popular among both similar users 2 and 3.

More formally, collaborative filtering is the process of filtering or evaluating items using the opinions of other people (Schafer, Frankowski, Herlocker, & Sen, 2007). At the core of collaborative filtering systems stands the user feedback, usually in form of a *rating*. These ratings can be taken through *explicit* means*, implicit* means, or both. Explicit ratings are those where the user is asked to provide an opinion (stars, like/dislike, etc.) The implicit ratings are those inferred from a user's actions (buy/not buy). There are also systems which use both methods of getting ratings for items (ex: Amazon).

The personalization within these systems is achieved by keeping track of users' ratings, match similar users and recommend items for which the similar users have had positive feedback. Pearson correlation summarized in the previous section is used in many such systems. Once the collaborative filtering systems became popular and used in systems with very large numbers of users, performance for matching users on the run turned out to be a problem. Therefore, most systems now use eager learning techniques applied on the database, like subsampling and clustering, algorithms which help easily locate the neighbors of a user.

Many researchers have tried to identify the main similarities and differences between collaborative filtering and case based recommender techniques. The main difference remains that CBR focuses on the properties of the content while collaborative filtering focuses on the public opinion about the content. This difference does not lead to choosing between either one or the other technique, but rather leads to hybrid systems, as the two techniques can be seen as complementing each other towards the extraction of more appropriate results (Aguzzoli, Avesani, & Massa, 2002), (Schafer, Frankowski, Herlocker, & Sen, 2007).

Nevertheless, one advantage CBR holds over collaborative filtering in the context of learning with TARGET, is its capability of using domain knowledge in order to create new solutions, rather than only reuse past solutions. In the context of the TARGET Navigator, the recommended content is stored and extracted from a repository, but it can also be *automatically generated*. On the other side, considering a recommending system which recommends papers or books, the system has to recommend available products and cannot customize them to meet the user's needs. In TARGET the system can adapt existing stories, therefore the recommendations can be personalized. While collaborative filtering techniques only filter already existing content, CBR techniques have in this aspect the advantage of being able to generate content, by adapting past solutions to new problems, using domain knowledge, whereas collaborative filtering techniques are domain agnostic.

Still, one challenge that both types of systems have to face is the so-called *cold-start* problem. For case based recommendation systems this is the situation when the system has not yet enough cases to make relevant similarity retrieval. For collaborative filtering the problem occurs when an item is

new and there are no ratings yet. In TARGET this means that at the beginning, when there are no cases yet in the case base, recommendations will be purely content-based, trying to match the competences a story develops with the competences requested by the user. Later on the experiences of the users will be taken into account. Therefore, the TARGET Navigator will be a *hybrid* recommender, using features from both recommending approaches described in this section.

2.3 Recommender Systems in e-Learning

Recommendations within eLearning are used in order to help learners make choices even when they do not have sufficient personal experience to choose among the alternatives. What sets making recommendations to learners aside from recommending in commercial domains is that it is not enough to know that the learner liked the recommended material, but it is also necessary to know if the learning outcomes have been achieved. As well, the background knowledge of the learner has to be considered, therefore the final ranking needs to consider more significant parameters rather than only preferences. Tang and McCalla researched multi-dimensional recommenders in (Tang & McCalla, 2009) and (Tang & McCalla, 2003), where the recommendations are scientific papers with the purpose of learning. In (Tan, Guo, & Li, 2008), the authors present how collaborative filtering techniques can be used to recommend learning courses on the web.

In (Yang, Wang, & Chen, 2007), the authors use case based reasoning inspired techniques (experience based reasoning and capability based reasoning) to recommend adequate IT certification exams to learners. In (Khribi, Mohamed, & Nasraoui, 2008) the authors present a two stage (offline and online) recommender. The system uses eager learning techniques which work offline to cluster and model learners and learning material. At the online recommendation phase, the system uses hybrid techniques which combines content based filtering with collaborative filtering.

3 TARGET NAVIGATOR

In TARGET the recommended objects are stories. The TARGET Navigator (TN) provides retrieval strategies for learners and mentors. Hereby the difference consists in how the competences are identified. For a learner, his target competence profile and the current competence profile are used. For a mentor, the target competence profile and the current competence profile of his mentees. Besides these sources, the TN uses the history of the user's choices in the form of cases according to the CBR paradigm.

The recommendations of the TARGET Navigator can be represented on three levels, in increasing degree of complexity and capabilities from top to bottom as shown in Figure 5.

Figure 5 Overview of the TARGET Navigator services

The first type of recommendations is the 'Simple Recommendations' and it is used when the user does not provide his competence profile to personalize the recommendations, either because he does not have one yet or he does not want a personalized recommendation. An example of such recommendation is "The highest ranked stories". Since the recommendations are independent on the user, each user gets the same non-personalized recommendations.

The second type of recommendations is the "Personalized Recommendations". It is useful for recurrent users of the system, where the navigator can reason with the knowledge it has about the user in order to give more targeted recommendations. The user provides his current competence profile to be used in the recommendation. An example of such a personalized recommendation is "Users with similar competences ranked the following stories positively". These recommendations can be classified as collaborative filtering, since the user competence profile is a proxy to assess user similarity.

On the third layer of personalization are the "Adapted Recommendations", which are an enhancement to the "Personalized Recommendations". The users targeted by this type of recommendations are the learners which have specified their targeted competences in the system. Recommended stories match the user target competence profile. Moreover, recommended stories are not only retrieved from the existing set of stories, but they can also be created by adapting existing stories, so that they better match the user query. Again, the history of the learner is important at this layer since past choices are saved as cases that help assessing the most important features for the user.

In this section we discuss the methods used for these types of recommendations and the data these methods operate on. The API interface is described in section 5.

3.1.1 Simple recommendation

This algorithm retrieves the complete set of stories and orders them based on the rankings given by users. As mentioned in section 2.3, in a learning environment it is not enough to know that the user liked a learning object (in our case a story), he should also have achieved a positive learning

outcome. Therefore we introduce a ranking measure which is based on the explicit ranking given by the user and his learning outcome:

$$total\_ranking^{U}_{C} = (w_{ranking} * ranking^{U}_{C} + w_{learning} * learning\_outcome^{U}_{C}) / (w_{ranking} + w_{learning})$$

Equation 4 Total ranking for user U and story C as a weighted sum of the ranking given by the user U on story C and his learning outcome

The ranking $ranking^{U}_{C}$ is normalized to the (0,1] interval, i.e. the values given by the users are divided by $MAX_{ranking}$.

The learning outcome is measured according to the probabilistic competence evaluation framework described in D4.4. According to this framework, after each story has been played the probabilities of a user's competences are updated based on the evidence gathered by playing the game (see for example section 3.2 and 3.3 in D4.4). According to this framework, we can define the learning outcome based on the difference between the probabilities of the user having a particular competence before and after the story has been played, for each competence the story is meant to train:

$$learning\_outcome^{U}_{C} = \sum_{i=1,..,n} (p^{U(}comp_i, t+1) - p^{U(}comp_i, t)) / n$$

Equation 5 Learning outcome after playing story C, calculated with the probability p^U for user U before (t) and after (t+1) playing C, for each $comp_i$ trained by story C

The outcome can theoretically be negative, since after playing a story there might be less evidence that the user has a particular competence. Therefore this metric has a range equal to [-1,1].

Finally, the total ranking for a story C is the sum of the total rankings of all users that have played the story:

$$total\_ranking_C = \sum_{i=1,..,m} total\_ranking^{Ui}_{C} / m$$

Equation 6 Total ranking for story C based on all users U_i that have played it

As normal for rankings, this metric suffers from the "cold-start" problem described in section 2.2 . New stories have no rankings and no learning outcomes associated, and therefore a total ranking equals to zero. In the case of recommendations in TARGET, as discussed in section 2.2, this issue has a more limited impact since also content-based recommendations are supported (as explained in section 3.1.3).

For the initial value of $w_{ranking}$ and $w_{learning}$ we use the arbitrary values of 0.33 and 0.66 respectively. An intuitive explanation is that we give more importance to the learning outcome calculated by the system than to the users' judgment. In section 3.1.4 we show a method to revise these values[1].

In the following unless otherwise specified we will use ranking and total ranking interchangeably.

3.1.2 Personalized recommendation

In this case the recommendation is based on ranking from similar users, according to the principles of collaborative filtering. As discussed above, we use the current competence profile of the user to assess similarity between users[2]. In order to do so, we use the Case Based Recommendation framework described in (Smyth B.,2007), where similarity is assessed as a weighted sum of similarity functions, each of them measuring the similarity of a different dimension of the item to recommend.

$$sim\ (T,C)=(\sum_{i=1..n} w_i * sim_i(T_i,C_i))/(\sum_{i=1..n} w_i)$$

[1] Although the weights are indicated with the same names for the three different types of recommendations, each weight is specific to each recommendation. This means that their values differ across recommendation types.

[2] At a later stage we envision the possibility to expand the user profile with social data gathered by the social tools described in D5.2

Equation 7 Similarity between the target item t and the recommended item c, where i=1,..,n are the dimensions of the items. The closest sim(T,C) is to zero, the more similar T and C are.

This measure takes advantage of the fact that items to be recommended are represented as structured data, in the form of a set of dimensions, i.e. predefined (attribute, value) couple such as (price, €100), (colour, black). For each dimension of the item a fine-grained similarity function sim_i can be defined, since a global similarity function would not capture the specificity of each dimension. In the following we calculate similarity using the distance between two items. Therefore lower values of similarity imply more similar items.

In the case of Personalized Recommendations there are two dimensions, total ranking and current user profile. We want our similarity function to have values close to zero for stories that have been ranked highly by similar users. In other words, the similarity is to a target story that is ranked the maximum by an identical user. Therefore the similarity of a story C ranked by a user userC is the weighted sum of the total ranking similarity (in this case how close is the story to be top ranked) and how similar userC is to the current user U.

$$simPR^{userC}(C) = (w_{totalranking} * (MAX_{totalranking} - total\_ranking^{userC}{}_C) + w_{currcomp} * sim_{currcomp} (currcomp_U, currcomp_{userC})) / (w_{totalranking} + w_{currcomp})$$

Equation 8 Similarity between the ideal target story and the candidate story C in the Personalized Recommendation. userC is the user that ranked C, and U is the current user

The metric $total\_ranking^{userC}{}_C$ was introduced in the previous section.

Calculating similarity between competence profiles is more complex. In doing so we adopt the pedagogical framework provided by Competence-based Knowledge Space Theory (CbKST), described in D2.2. In this approach, competences are organized in a directed acyclic graph where the edges are prerequisite relations. Such a graph represents a structure similar to the one illustrated in Figure 3, and analogously to what is discussed in (Smyth B., 2007) we can define a metric based on such a structure.

The basic idea is as follows: competences that are connected by a path in the graph have a distance proportional to the number of links between them. This metric is asymmetric, since if competence $comp_A$ is prerequisite of competence $comp_B$, dist($comp_A$,$comp_B$) is not the same as dist($comp_B$,$comp_A$). This stems from the fact that the prerequisite relation implies that having $comp_B$ entails having $comp_A$ in the previous example, while the opposite is false. Therefore, a user that possesses $comp_B$ is more similar (for the scope of a recommendation) to a user that possesses $comp_A$, than the other way around.

Competences are stored in the Knowledge Ecosystem together with pair-wise prerequisite links. From these relations a directed acyclic graph of competences can be constructed. Using this graph standard graph algorithms can calculate the distance between two nodes. Such distance is defined as the number of directed edged between two nodes. Strictly speaking, in a directed acyclic graph if there is a path from $comp_A$ to $comp_B$ there cannot be a path from $comp_B$ to $comp_A$, since the two paths would form a cycle. Therefore dist($comp_B$, $comp_A$) = ∞. In order to define a metric, we adopt the approach that in such cases the distance has a high but finite value.

Therefore given a user T with competence profile CT = {$comp^T_1$,…,$comp^T_n$} and a "candidate" user C with competence profile CC = {$comp^T_1$,…,$comp^T_m$}, we define the similarity as the sum of the minimum distances from each competence in CT to each competence in CC:

$$sim_{currcomp} (CT, CC) = \sum_{i=1,..,n} \sum_{j=1,..,m} min(dist(comp^C_j, comp^T_i)) / (n * MAX_{dist})$$

Equation 9 Similarity function for competence profiles

If two competence profiles are equal, the similarity value is zero. To have a metric range equal to [0,1], dist($comp^C_j$, $comp^T_i$) is equal to the number of directed edges (prerequisite relations) from $comp^C_j$ to $comp^T_i$, if such path exists, or MAX_{dist} otherwise.

To calculate Equation 8 we still need to define $w_{ranking}$ and $w_{currcomp}$. We initially assume an equal influence of the two factors and assign the value .5 to both. In section 3.1.4 we provide a method to refine these values.

Finally, the similarity of story C to the ideal target story across all users that ranked it is:

$$simPR(C) = \sum_{i=1,..,m} simPR^{Ui}(C) / m$$

Equation 10 Total similarity of story C to the target story for user U, calculated across all users U_i i=1,..,m that have ranked C

3.1.3 Adapted recommendation

An adapted recommendation takes into account also the user's target competence profile. This means that the metric also takes into consideration the similarity between the target competence profile of the user and the competences trained in the story. Reusing what defined in the previous section, the metric for the adapted recommendation is:

$$simAR^{userC}(C) = (simPR^{userC}(C) * (w_{totalranking} + w_{currcomp}) + w_{targetcomp} * sim_{targetcomp} (targetcomp_U , storycomp_c)) / (w_{totalranking} + w_{currcomp} + w_{targetcomp})$$

Equation 11 Similarity between the target story and the story C in the Adapted Recommendation. UserC is the user that ranked story C, and U is the current user. Story C trains competence $storycomp_c$

The similarity function $sim_{targetcomp}$ is defined as in Equation 9 Similarity function for competence profilesEquation 9. In this case the asymmetry of the distance between two competences corresponds to the fact that a story that trains a competence the user already has should be preferred to a story that trains a competence not included in the user target competence profile. This is because we assume that the user is not interested in competences not in the target competence profile.

A significant different of Adapted Recommendations is that existing stories can be modified according to a set of desired competence using the Didactical Modeler (described in D4.5). The DM can modify an existing story to tailor it to a particular set of competences to train. This results in the generation of new stories that can be recommended to the user.

The way this is used by the Adapted Recommendation is the following: the best story candidate calculated according to Equation 11 is examined. If the set of competences the story trains is equal to the user target competence profile, no action is taken. If this is not the case, the DM is invoked with the candidate story and the user target competence profile. The DM generates a story that trains all the competences in the target competence profile. The generated story has never been played by any learner, and therefore has no rankings or user similarity. Therefore, $simPR^{userC}(C)$ is equal to zero, and the new story is ranked purely based on $sim_{targetcomp}$.

This approach is inspired by CBR where the most promising case retrieved is adapted to provide a new solution. The newly created story is also evaluated according to the following equation.

The total similarity for story C considering all users that ranked it is:

$$simAR^{userC}(C) = (simPR(C) * (w_{totalranking} + w_{currcomp}) + w_{targetcomp} * sim_{targetcomp} (targetcomp_U , storycomp_c)) / (w_{totalranking} + w_{currcomp} + w_{targetcomp})$$

Equation 12 Total similarity to the ideal target story for story C and user U

As it can be seen, it is composed by a part dependent on ranking (*simPR(C)*) and one dependent exclusively on the competence in the user target competence profile and in the story. Only the first part suffers from the "cold-start" problem, while the second part does not since it is "content-based".

Again, the weights are initially set to 0.33 and refined with the use according to the CBR paradigm. The next section provides a way to fine-tuning them.

3.1.4 Fine-tuning metric weights

As discussed in section Case Based Recommenders and Personalization2.1.2, CBR systems have the capability to learn from previous cases, by retaining the cases that have been solved in the past. In TARGET cases are stories and their outcomes in terms of game performances. A possibility the system has to learn is by fine-tuning the weights used in the similarity functions introduced in the previous sections.

The basic idea is that since TARGET stores stories played by users and their outcome in terms of performances, the TARGET Navigator can modify the weights a-posteriori trying to adjust them so that stories played by a particular user with a positive learning outcome would have been ranked highly by the recommender.

The problem can be considered as a non - linear regression problem, where the similarity function is used to predict the learning outcome of the story, and the non - linear regression is used to identify the weights that give the best approximation. We want the story with the best learning outcome to be ranked first, and so on. This can be achieved by scaling the learning outcomes to the similarity functions range and solving a non - linear regression problem. In this case the learning outcomes are a proxy for the recommendation values provided by the similarity functions.

If LO = $\{lo_1, \ldots, lo_n\}$ is the set of learning outcomes with lo_i = learning outcome of story s_i[3], we can scale these value in the range [0,1] by defining the transformation $rv_i = (\max(LO) - lo_i) / (\max(LO) - \min(LO))$. The set RV = $\{rv_1,..,rv_n\}$ is the set we aim to predict by tuning the weights of the similarity functions.

For example, for the Adapted recommendation we have:

$$RV = ((1 - w_{targetcomp} /(w_{totalranking} + w_{currcomp} + w_{targetcomp})) simPR(S) + w_{targetcomp} / (w_{totalranking} + w_{currcomp} + w_{targetcomp}) * sim_{targetcomp} (targetcomp_U, storycomp_c) In)$$

where S is the set of stories related to the learning outcomes, S = $\{s_1,..,s_n\}$, simPR(S) is a vector of dim(n), simPR(S) = { simPR(s_1), ..., simPR(s_n)} and In is the identity vector of dim(n).

Such a problem can be solved using non–linear Least Square fitting (Kelley C., 1999) for example.

It is worth noting here that the weights are user specific, and therefore represent the user interest for a particular factor influencing the recommendation, such as rankings by similar users or competences addressed by a story.

The weight refinement can be performed at regular intervals, and as discussed in section 2.1.2 the cases stored by TARGET can be considered as user profile that is continuously refined by the user interacting with the system. An improved user profile has in turn a positive consequence on the precision of the recommender system.

3.1.5 Structure of the case base

According to the recommendations of Ian Watson in (Watson, 1997), cases should be saved in the problem – solution - solution_outcome - solution_explanation form. For the TARGET Navigator is important to save cases in a form that supports the recommendation types, in particular the fine-tuning described in section 3.1.4. Since the user competences change, it is important to record them in a case. The same is valid for stories, which contains rankings[4] that might change as a consequence of user interactions. The competence profile of the users that rated the story must also be saved. The case represents a snapshot at the moment the recommendation was required. Each case has a time-stamp and it is stored in the case base. The data captured in the case would not otherwise be possible to reconstruct *a posteriori*.

---

[3] To simplify the explanation we assume here that the user plays a story only once

[4] Strictly speaking, rankings are associated to the experiences of a story. A story can contain more rankings since it can have more experiences associated the user have ranked.

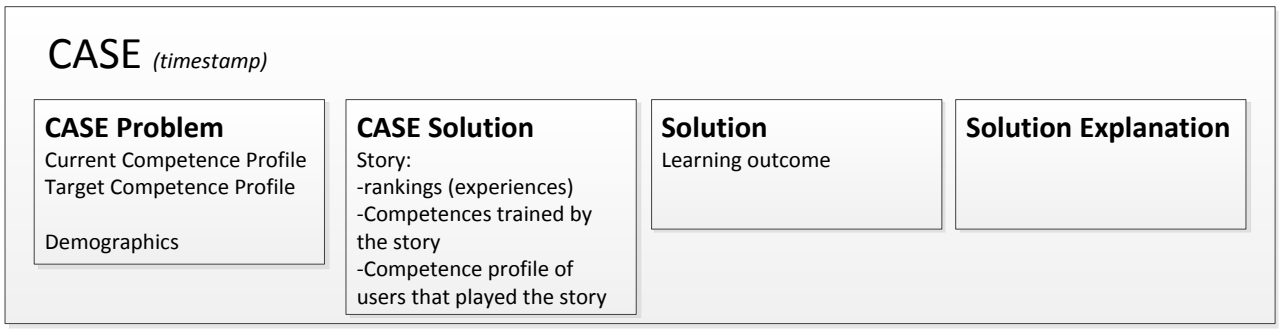

Figure 6 Components of a case in the case base

A case is saved for each played story by the user, containing the competence profile, the story and the learning outcome at the moment of playing.

We also save demographics since it might be useful in calculating user similarity, although at the moment we only use competence profiles. Demographics are data contained in the user profile, such as age, gender, interests, profession, role, etc. Extending our approach to consider also demographics would imply to define a similarity function between user profiles. The same Case Base approach as described in section 3.1.2 would be used, as it has the advantage of using different similarity functions for different dimension of the user profile.

At the moment we make no use of the solution explanation category.

4 ARCHITECTURE

Figure 7 illustrates the main components of the TARGET Navigator architecture. It is a three layered architecture. The Data Layer contains the components responsible for the organization of data. The main such components are the CaseBase and the IndexManager. The case base is a cache memory where the data needed for the recommender is stored in form of CBR style cases. The IndexManager component is responsible for keeping track and accessing the indexes on the CaseBase. The Data Layer contains also the functionality to interface with the Knowledge Ecosystem, the KE Reader.

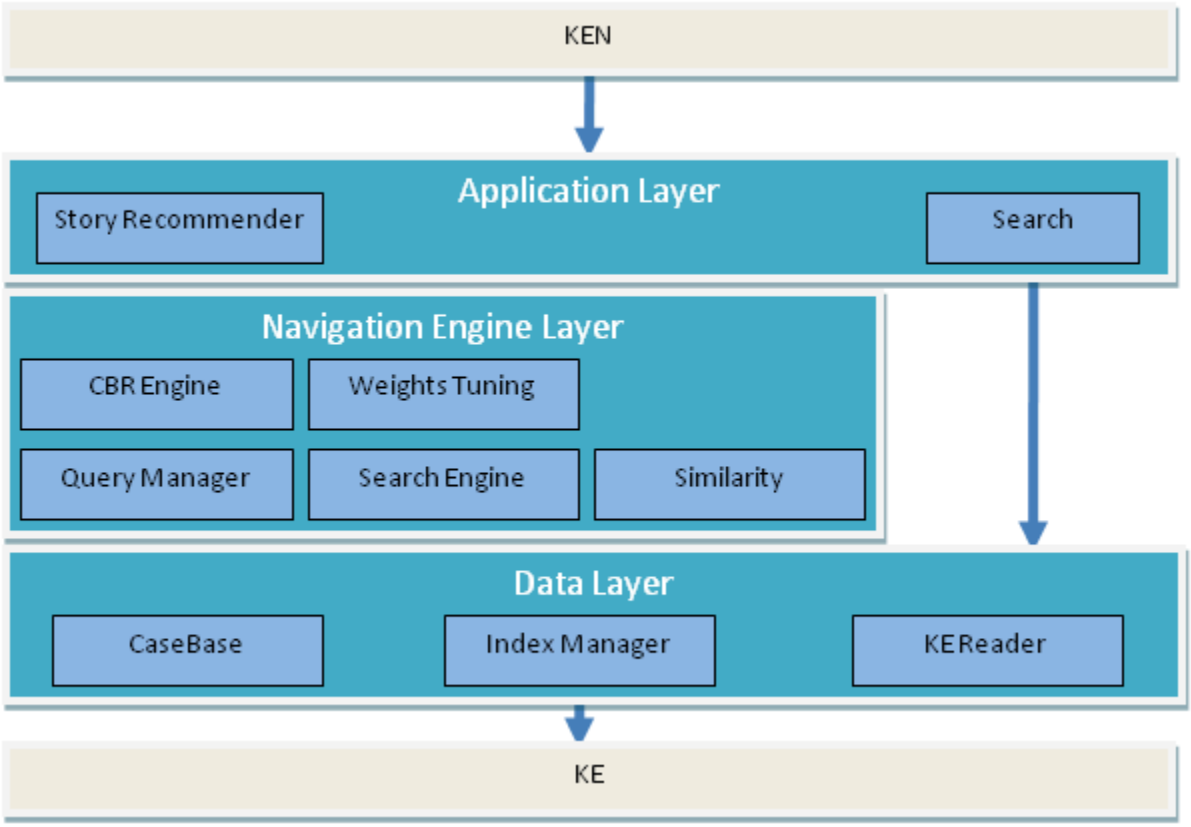

Figure 7 TARGET Navigator Architecture

On top of the Data Layer there are the components responsible for analyzing and searching this data. The Similarity component ranks the results obtained by the Query Manager, which in turn uses the services of the Search Engine. The latter is responsible for making the searches on the case base using the needed indexes. The Weight Tuning is responsible for the eager learning techniques applied on the case base, i.e. to recalculate the weights used to provide recommendations, as described in the previous section. The CBR Engine is the component responsible for the case based reasoning functions. Not shown in the picture, the interface with the Didactical Modeller obtains stories generated to match the user targeted competences.

The top layer of the architecture is the application layer, where the recommender systems are implemented, and also simple search functionality is offered to the KEN. The three types of recommenders belong to this level and make use of the services provided by the Navigation Engine Layer components. The simple search functionality interacts directly with the KE via the KE Reader. The applications on this layer receive their input data from the KEN.

4.1 Development Perspective

The most important functionality of the TARGET Navigator is the recommendation of stories for the learners. The flowchart diagram for all supported types of recommendations is presented in Figure 8.

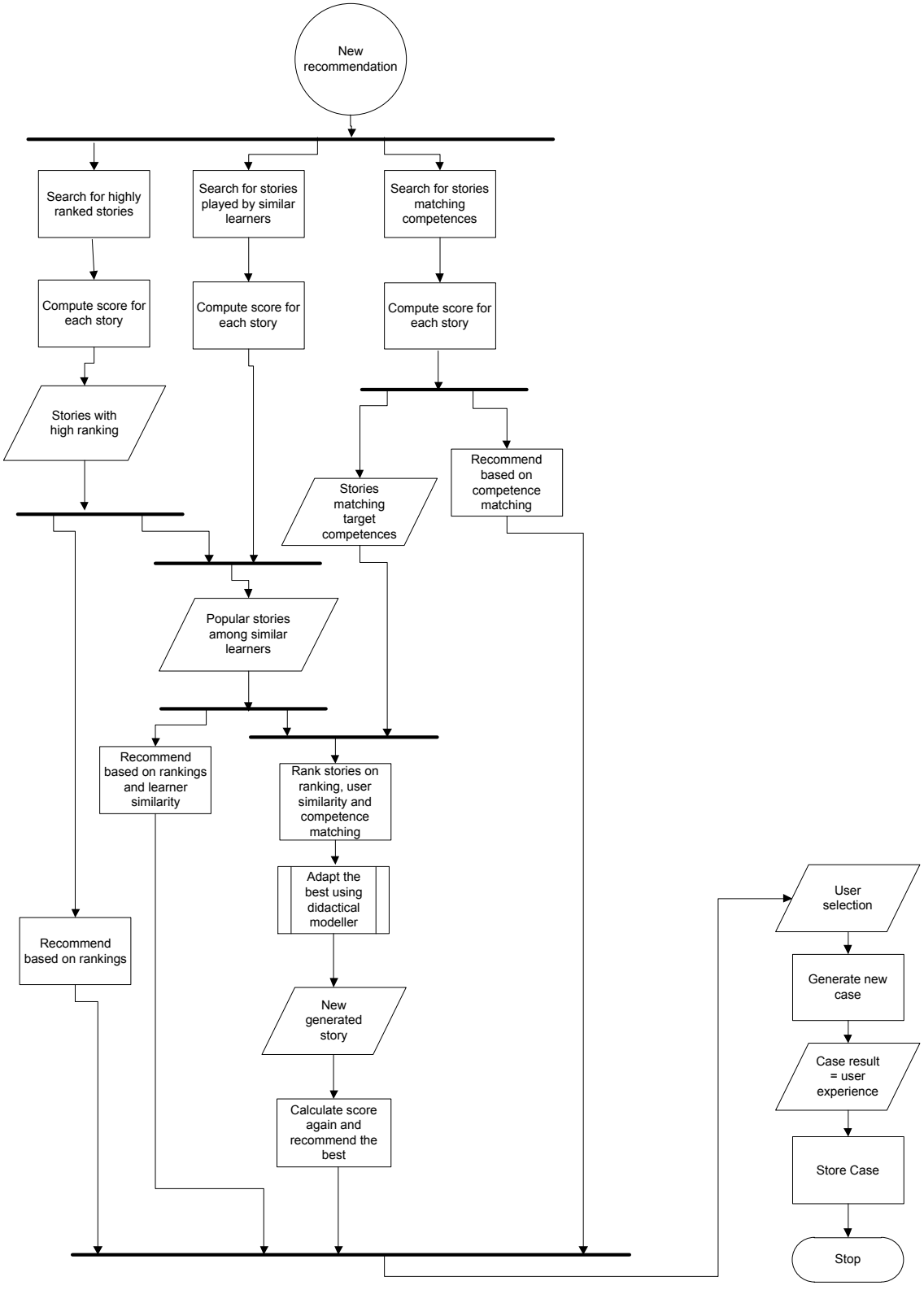

Figure 8 Target Navigator process flow

The simple recommendation is the leftmost flow, and it makes use only of the rankings of the stories. The second flow, the personalized recommendation, calculates recommendations based on the similarity with the users that played the story to be recommended, and it reuses the ranking functionality of the simple recommendation to provide recommendations based on rankings and similarity. The third flow, the adapted recommendation, calculates the similarity between the user target competences and the story competences, and it reuses the functionality provided by the previous two to recommend stories based on rankings, similarity and competence match. The picture also shows the possibility of recommendations that are only based on the match between the competences targeted by the user and the competences trained by the stories. This type of recommendation is a subclass of the adapted recommendations when the user profile part is disregarded. In the flow of the adapted recommendation the DM invocation is not shown as a separate flow, but as a process that generates an adapted story based on a story to modify and a target competence profile.

The implementation of the TARGET Navigator is based on the CBR Framework jColibri[5], adapted to fit our API definition, the data structures we use and the interface with the Knowledge Ecosystem. In Figure 9 we describe a simplified diagram of the TN, corresponding to the architecture described in Figure 7.

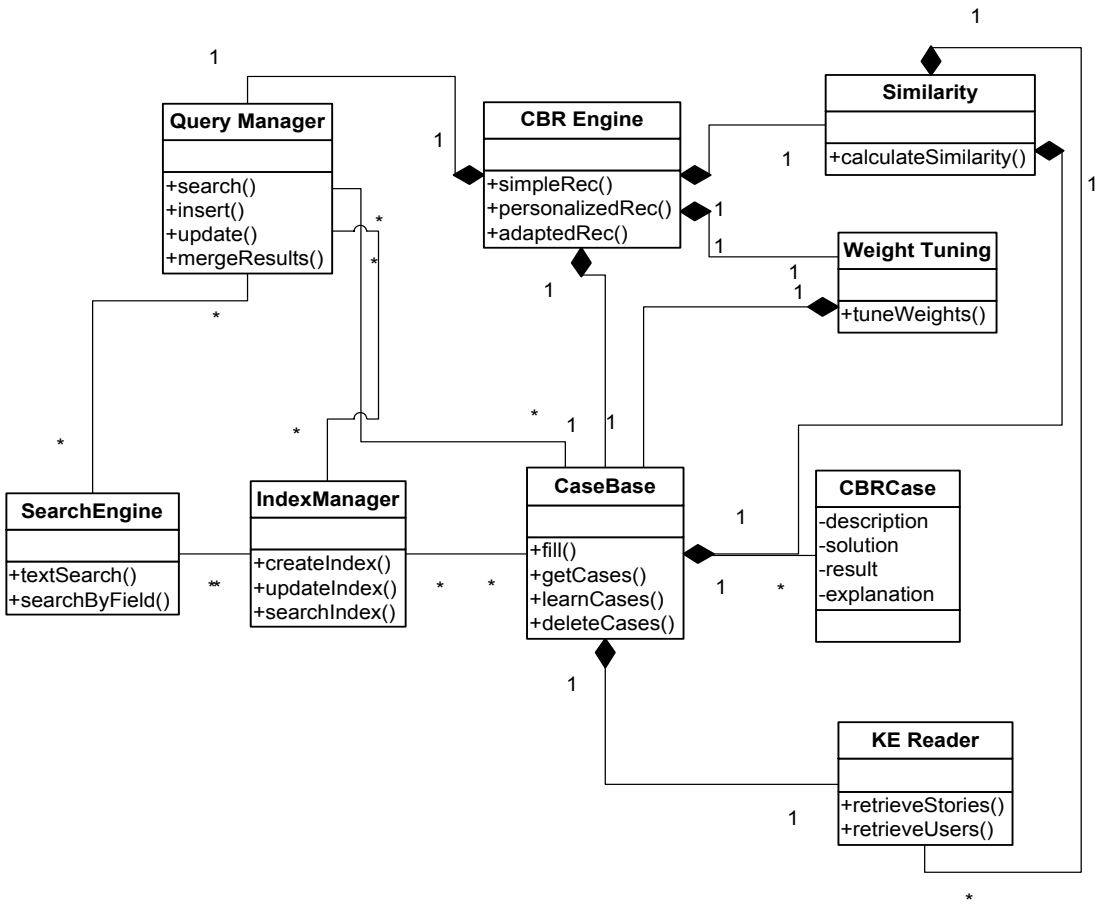

Figure 9 Class diagram of the TARGET Navigator Services

[5] http://gaia.fdi.ucm.es/projects/jcolibri/

In the current status of the Navigator, the Story Recommender has been developed. The StoryRecommender uses two indices, one which indexes stories based on the competences they train, and one which indexes cases based on the learner competences. This enables searching for stories played by similar learners, or retrieval of stories which target a set of competences.

5 SUMMARY AND CONCLUSIONS

In this document we presented the TARGET Navigator Services (or shortly TN). The TN provides search and recommendation services on top of the Knowledge Ecosystem Repository. These services are used by the Knowledge Ecosystem Navigator.

The major focus on this document is on explaining the type of recommendation the TN implements. We base our approach on two techniques: Case Based Recommendation and Collaborative Filtering. The first technique is content-based and has the advantage of being able to define similarity between items to recommend in a very flexible way. This is particularly true in the case of structured data, which is the type of data stories and competence profiles in TARGET are. In Case Based Recommendation for each dimension of a data item a similarity function can be defined, instead of a global similarity function that would not capture the semantics of the single dimension.

The TN implements three types of recommendations: simple (only based on user rankings), personalized (based on rankings and user similarity) and adapted (based on rankings, user similarity and similarity between the story and what the user wishes to learn).

In our approach we define a similarity function for user rankings. Since our focus is a learning environment, we expand the term ranking to mean a combination of the actual user rankings and the learning outcome of the played story. In order to implement a collaborative filtering approach we also define user similarity based on competence similarity. Two users are similar if their competence profiles (the competences they possess) are similar. In order to define similarity between competence profiles, we define a distance metric on the graph of competence dependencies. This graph is defined in TARGET according to the CBKST theory as described for example in D4.4.

The distance metric defined on competences supports both personalized as a well as adapted recommendation. In the former it is used to establish similarity between users, and in the latter between a user's target competence profile and the competences trained by a particular story.

Since each similarity function is a weighted sum of different factors, we define a weight tuning mechanism based on Case Based Reasoning. We save each story the user plays, together with the user's learning outcome and the data that is used to calculate a recommendation. We then try to adapt the weights so to match the user learning outcome, so that we have high recommendation values for high learning outcome, and vice versa. The implication is that the recommender favour stories that have a good learning outcome for the user. Mathematically this is a non-linear regression problem that can be solved with standard techniques.

This approach needs to be validated on real data. There are two type of analysis that can be performed on the current approach. The first one aims at establishing the best parameters to get good performances, and can be considered as an optimization problem, just as the weight tuning problem. The second one is more of a qualitative nature, in that it tries to assess whether the proposed approach is capable of modelling the dynamics under study. A concrete example of the outcome of such analysis would be to determine that user similarity needs to take into account also the user profile, as discussed in section 3.1.5.

6 API

STORY SEARCH AND STORY RECOMMENDATION

Method	Parameters
findStories(enum{played,not_played,all), String username, int[] Competenceids, boolean Matching, enum{any,all}): int[] Storyids Simple search interface: finds stories (all or played or not played), by a particular user (null = all users), that match or not match (Matching) any or all of Competence ids (null = do not consider competences)	*enum{played,not_played,all)* whether the stories should be new to the user, not new or all *String username* the particular user that played or not the stories *int[] Competenceids* The competences to match (or not) *boolean Matching* Whether there must be a match or not with the competences *enum{any,all}* whether all of any competence must be matched
recommendStories(int[] TargetCompetences, String username, boolean Contained, int[] Storyids): int[] recommendedStoryids Provides the three types of Recommender services: Recommend stories based on Competences (if not null) and positively rated by users similar to username (if not null) and contained/not contained (boolean Contained) in Storyids (if not null). By setting a parameter to null the recommendation does not keep it into account. For example, leaving the user to null and only specifying competences to be matched, the recommendation is not based on user similarity, but only on competences and rankings.	*int[] TargetCompetences* Target competences *String username* The reference user for collaborative filtering *boolean Contained* Used to filter already played stories *int[] Storyids* The array to stories to filter *int[] recommendedStoryids* The stories to be recommended

7 INSTALLATION

The TARGET Navigator exposes its services as WebServices. In order to deploy them on a web server, you need the following applications:

- Java Runtime Environment (JRE)[6];
- A J2EE 6 Web Profile (JSR-316) compliant Application Server. TARGET uses Glassfish[7]

After the two applications are installed, the TARGET Navigator needs to be deployed on the selected web server. The software can be found at the project repository on Sourceforge:

https://reachyourtarget.svn.sourceforge.net/svnroot/reachyourtarget/TN/TARGETNavigatorWebService/build/.

Installation on GlassFish:

Copy `TargetNavigator.WAR` file to the `/autodeploy` directory for the domain to which you want to deploy. If you are using the default domain, `domain1`, which is set up by the GlassFish installation process, the appropriate directory path would be `<AS_HOME>/domains/domain1/autodeploy`.

[6] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[7] http://glassfish.java.net/

8 REFERENCES

Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. *AI Communications* , 39-52.

Aguzzoli, S., Avesani, P., & Massa, P. (2002). Collaborative Case-Based Recommender Systems. *ECCBR '02: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning* (pp. 460-474). Springer-Verlag.

Bradley, K., Rafter, R., & Smyth, B. (2000). Case-Based User Profiling for Content Personalization. *Proceeding of the 1st International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH2000)* (pp. 62-72). Springer-Verlag.

Bridge, D., Goeker, M. H., McGinty, L., & Smyth, B. (2006). Case-based recommender systems. *The Knowledge Engineering Review* , 315-320.

Hayes, C., Cunningham, P., & Smyth, B. (2001). A Case-Based Reasoning View of Automated Collaborative Filtering. *ICCBR '01: Proceedings of the 4th International Conference on Case-Based Reasoning* (pp. 234-248). London: Springer-Verlag.

Kelley C. T. (1999). Iterative Methods for Optimization. SIAM Frontiers in Applied Mathematics 18. Available at: http://www.siam.org/books/textbooks/fr18_book.pdf.

Khribi, M. K., Mohamed, J., & Nasraoui, O. (2008). Automatic Recommendations for E-Learning Personalisation based on Web Usage Mining Techniques and Information Retrieval. *ICALT'08 Eighth IEEE International Conference on Advanced Learning Techniques* (pp. 241-245). IEEE Computer Society.

McSherry, D. (2003). Similarity and Compromise. *Proceedings of the 5th International Conference on Case-Based Reasoning* (pp. 291-305). Springer-Verlag.

Pazzani, M. J., & Billsus, D. (2007). Content-Based Recommendation Systems. In P. Brusilovski, A. Kobsa, & W. Nejdl, *The Adaptive Web* (pp. 325-341). Springer-Verlag.

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative Filtering Recommender Systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl, *The Adaptive Web* (pp. 291-324). Springer-Verlag.

Smyth, B. (2007). Case-Based Recommendation. In A. K. P. Brusilovsky, *The Adaptive Web* (pp. 342-376). Springer-Verlag.

Smyth, B., & McClave, P. (2001). Similarity vs Diversity. *Proceedings of the third International Conference on Case Based Reasoning* (pp. 347-361). Springer Verlag.

Smyth, B., Bradley, K., & Rafter, R. (2002). Personalisation Techniques for online recruitment services. *Commun ACM* , 39-40.

Tan, H., Guo, J., & Li, Y. (2008). E-Learning Recommendation System. *Internation Conference on Computer Science and Software Engineering* (pp. 430-433). IEEE Computer Science.

Tang, T. T., & McCalla, G. (2009, July/August). A Multidimensional Paper Recommender. *IEEE Internet Computing* , pp. 34-41.

Tang, T. Y., & McCalla, G. (2003). Smart Recommendation for an Evolving E-Learning System. *International Conference on Artificial Intelligence in Education (AIED).*

Yang, H.-L., Wang, C.-S., & Chen, M.-Y. (2007). A Personalisation Recommendation Framework of IT Certification eLearning System. *Proceedings of KES*, (pp. 50-57).