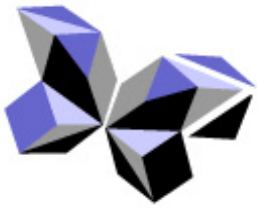


Digital Business Ecosystem

Contract n° 507953

WP 7: User Profiling

Del 7.3: Prototype Implementation of Profiling Mechanism

Information Society Technologies

Project funded by the European Community under the “Information Society Technology” Programme

Contract Number: 507953

Project Acronym: DBE

Title: Digital Business Ecosystem

Deliverable N°: 7.3

Due date: 31/07/2006

Delivery Date: 29/09/2006

Short Description:

The deliverable provides the current implementation status for the User Profile software provided to the DBE and a detailed theoretical analysis of a possible User Profiling algorithm which could be used for gathering and aggregating further information about a user and their behaviour taking several state-of-the-art algorithms and ontologies into account.

Partners owning: FZI (Christian Bartsch)

Partners contributed: UCE, INTEL, TUC, TCD, T6

Made available to: DBE Partners and European Commission

VERSIONING

VERSION	DATE	AUTHOR, ORGANISATION
0.1	**01/06/2006**	**CHRISTIAN BARTSCH (FZI)**
0.2	**15/06/2006**	**CHRISTIAN BARTSCH (FZI), NIKOS PAPADAKIS (FZI)**
0.3	**14/07/2006**	**CHRISTIAN BARTSCH (FZI), MANUEL ODENDAHL (FZI)**
0.4	**01/09/2006**	**VICTOR BAYON (UCE) – 1ST INTERNAL REVIEWER** **JOHN KENNEDY (INTEL) – 2ND INTERNAL REVIEWER**
1.0	**29/09/2006**	**CHRISTIAN BARTSCH (FZI) – FINAL VERSION**

Quality check

1st Internal Reviewer: Victor Bayon (UCE)

2nd Internal Reviewer: John Kennedy (INTEL)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In the DBE Deliverable D7.2 [Bart05b] the basic functionality of storing and editing a User Profile within the DBE environment was discussed. The User Profile Model was described and two Model Packages (UserInformation and UserPreferences) were introduced. Moreover, a high-level algorithm for the possible construction of implicit user information for the DBE users was introduced and discussed briefly. In addition to this, some User Profiling Scenarios were presented and analysed thoroughly. A very interesting aspect of the User Profiling is the way that the system will be able to operationalise user preferences in order to produce adequate recommendations. The Deliverable D7.3 at hand basically consists of two main sections as illustrated in Figure 1. The first section describes the prototype implementation of the "Assemblage as Mechanism" approach which has been decided to be the User Profile mechanism for the project in the previous Deliverable D7.2.

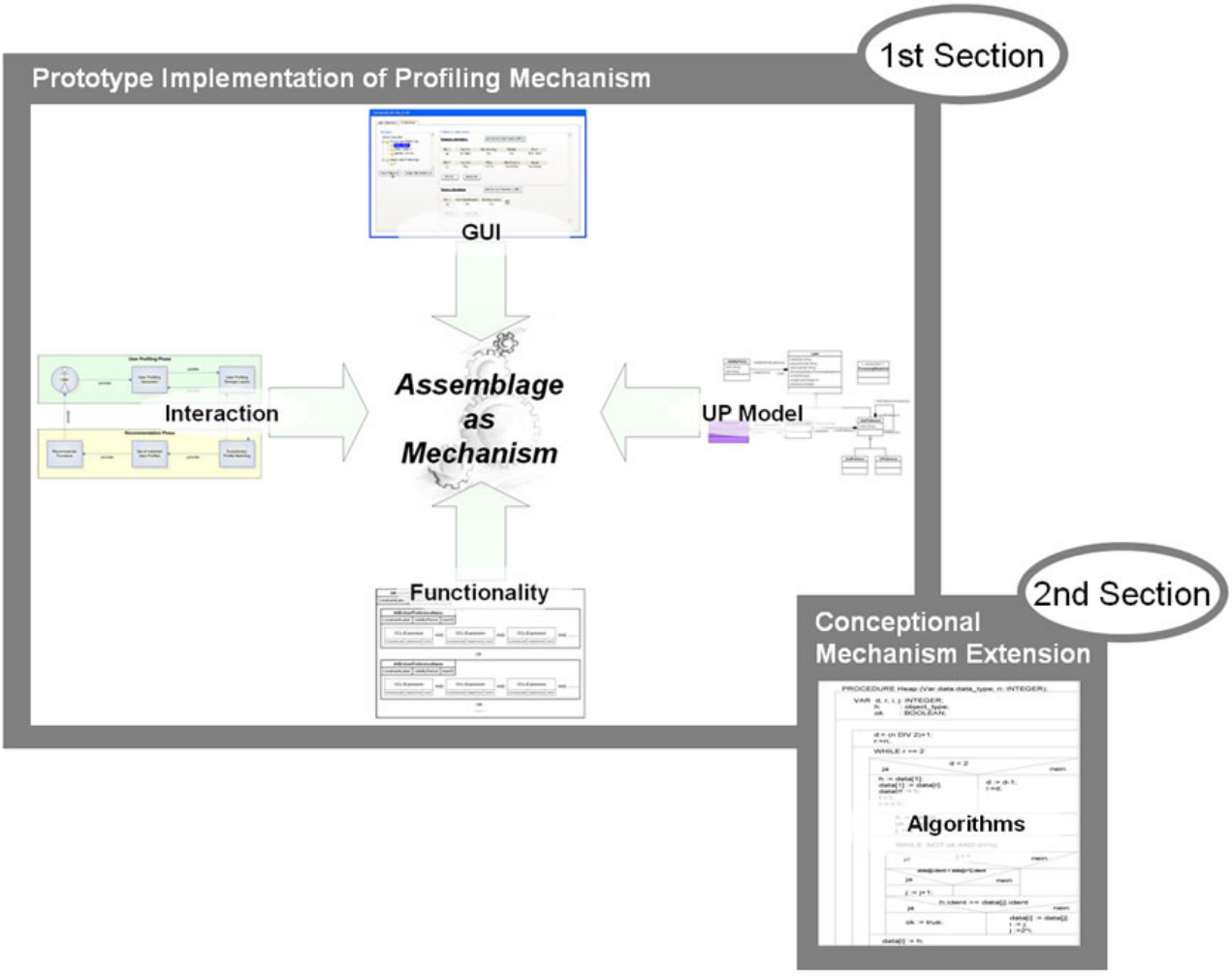


Figure 1: Basic Structure of the Deliverable D7.3

The second section deals with conceptional approaches for extending the functionality of the current User Profiling mechanism using additional algorithms. This procedure merges the two definitions given in D7.2 (“Assemblage as Mechanism” and “Algorithm as Mechanism”) to provide a more sophisticated and holistic view. An implementation of these algorithms in addition to the current Mechanism will be out of the scope of this Work Package.

The second section “Conceptional Mechanism Extension” discusses various approaches that might provide aforementioned algorithm based functionality to the DBE. As proposed in the last Deliverable D7.2, Collaborative Filtering seems to be a suitable and appropriate approach. Various Collaborative Filtering Algorithms are evaluated in this Deliverable by analysing experimental results from different scientific papers and studies taking each algorithm’s strengths and weaknesses into account. Within the scope of the particular requirements that the nature of DBE platform poses, a suitable recommendation algorithm for User Profiling purposes is proposed. Basic assumptions are made regarding the changes that this algorithm should undergo in order to be successfully incorporated into the DBE environment. The use of Ontologies within the p2p network is also investigated followed by a final discussion about the possibility of combining ontologies with Collaborative Filtering in order to produce better results.

The next chapter 2 shows the current implementation status of the User Profile software. The features and functions of both User Profile applications, based on OpenLaszlo[1] for the DBE Portal and on a Java Eclipse Plug-In for the DBE Studio, are illustrated by screenshots. Chapter 3 introduces the outline of the DBE environment as a basis for the elaboration of the User Profiling’s role in chapter 4. It is also referred briefly to some well known algorithms that are used in order to create a User Profile. What follows, is the discussion of the algorithms that generate item and user recommendation. That is covered in chapter 5 where Standard Collaborative Filtering, Distributed Collaborative Filtering and Ontologies are discussed. Finally, in chapter 6 a more general overview of the proposed mechanisms is presented and discussed and a possible algorithm for extending the current User Profile Functionality and Interaction with the Digital Business Ecosystem is proposed. Chapter 7 concludes this Deliverable.

[1] http://www.openlaszlo.org/

SECTION ONE:
PROTOTYPE IMPLEMENTATION OF USER PROFILING MECHANISM

2 IMPLEMENTATION STATUS

2.1 IMPLEMENTATION OVERVIEW

The following Figure 2 illustrates the framework in which the User Profile applications are integrated in. The DBE Users have two possibilities to access their User Profile: They can either use the web-based log-in via the DBE Portal (the User Profile is represented as an OpenLaszlo Application) or via the DBE Studio (the User Profile is represented as an Eclipse Multi-Page Editor Plug-In). Both applications use the same UserProfileLib.jar-File to communicate with the required Services in the Servent to perform the User Profile specific actions. All User Profiles (User Profile Models, UPM) are represented as MOF-compliant XMI-Files in the Knowledge Base. Other components of the DBE such as, for example, the Evolutionary Environment can use the User Profile XMI-files as input for further activities. The User Profile Applications utilise several core services of the DBE: The Recommender Service (RC Service), for example, is primarily used to store and retrieve User Profile Models from the Knowledge Base as theses methods are integrated in the Recommender Service. The Semantic Registry Service (SR Service) allows the look-up and retrieval of BML model IDs from the Semantic Registry whereas the Knowledge Base Service (KB Service) is used for retrieving BML models (represented as XMI-Files) from the Knowledge Base as a basis for the creation of explicit User Preferences.

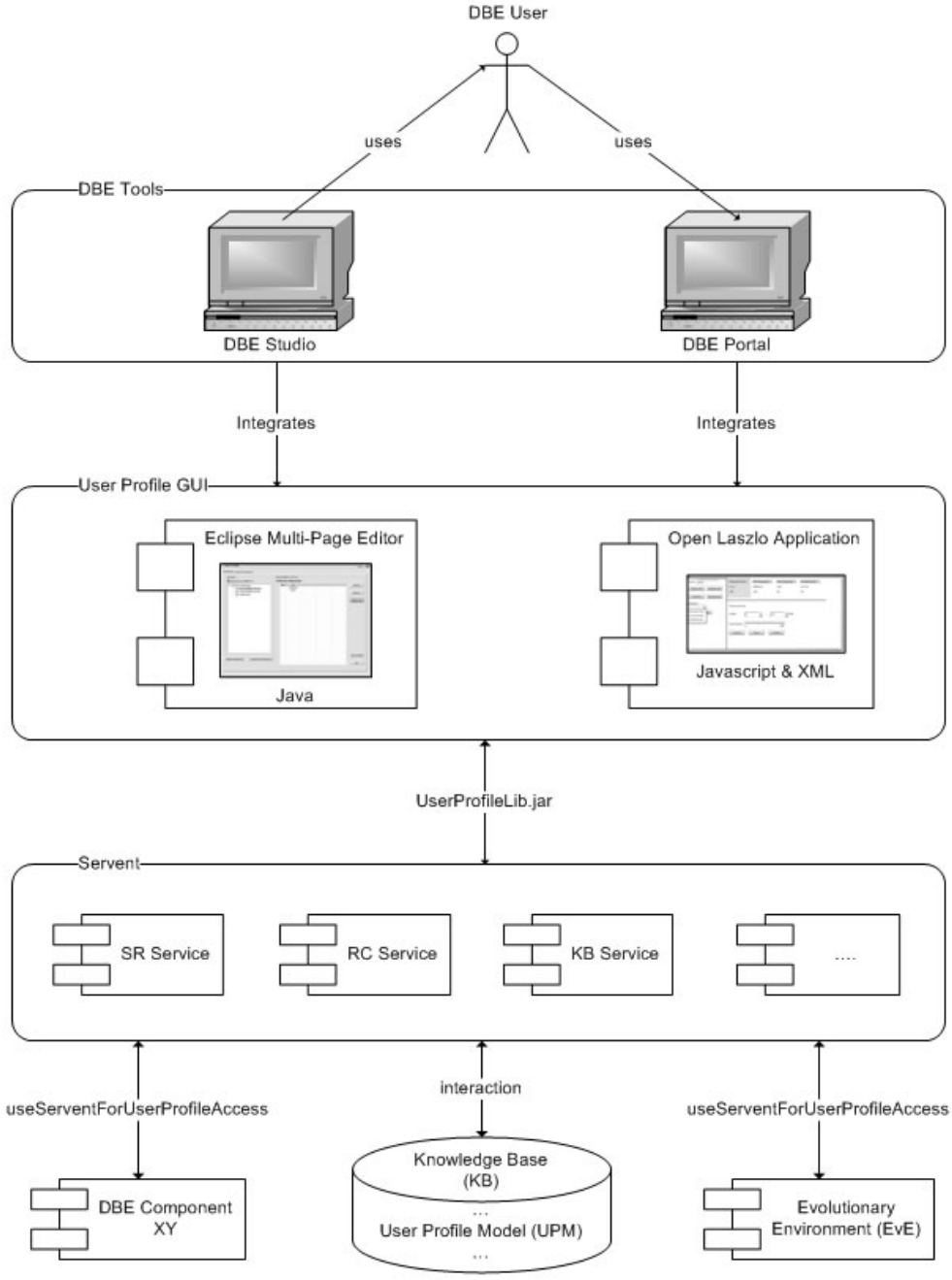


Figure 2: User Profile Implementation

The following sections clarify aforementioned functionality.

2.2 USER PROFILE OPENLASZLO DEVELOPMENT

2.2.1 GUI Functionality

The User Profile application based on OpenLaszlo is an implementation of the User Profile meta-model (see D7.2) to define preferences and queries. It parses BML models from the Knowledge Base and generates a GUI to build requests against the Service Registry. Requests are formulated as OCL constraints and grouped into User Preference documents. These grouped preferences build the User Profile Model (UPM). User Preferences can be retrieved and stored into the KB.

The constraints are built using the attribute descriptions provided by BML models. Each attribute description is parsed and, if necessary, additional ODM (Ontology Definition Models) ontologies are retrieved from the KB. A customised User Interface is generated according to the respective type of each attribute (type: string, type: integer, …).

Figure 3: OpenLaszlo - Model selection form

On screenshot in Figure 3, BML models in the Knowledge Base are browsed by using a simple keyword search algorithm (using the SR service). Furthermore, models related to stored User Preferences are also displayed to show the user the source of the BML model the attribute were chosen from.

On the right side, the available search profiles are shown, with the search attributes the user entered. Clicking on an attribute set opens the set editing screen. Adding a new set is achieved by clicking on the “Add new set...” button. Searching for a query set is achieved by using the “Search KB...” button. Adding a search agent for the set is done by using the “Autosearch” button. This “pull”-functionality will later be used to retrieve recommendations from the recommender without specifying a certain BML model manually. These buttons are available in most screens dealing with User Profiles.

2.2.2 Set editing screen

The window in Figure 4 is the main window of the application, and is used to edit a User Profile search query.

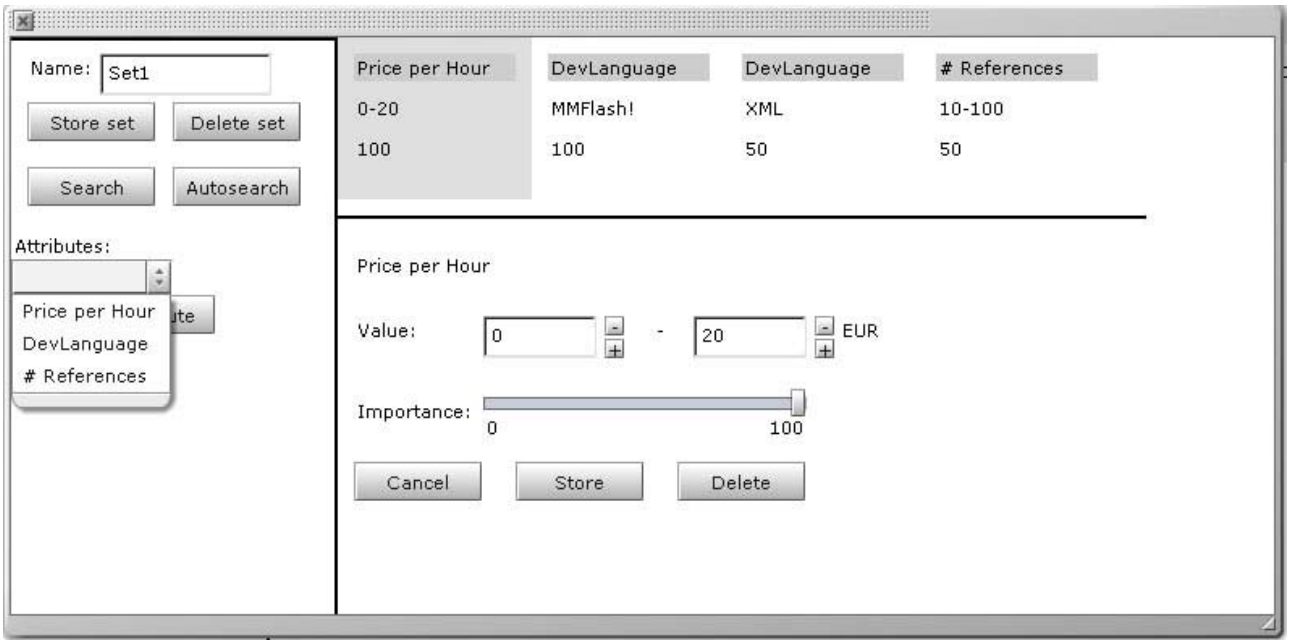


Figure 4: OpenLaszlo - Set editing screen

The available attributes of a model can be selected on the left side. Three different categories of attributes are available, pertaining either to BML Services, BML Assets or BML Entities. The category can be selected using the topmost drop-down menu. The category instances available in the selected model can be selected using the second drop-down menu, and finally the attribute to be edited can be selected using the drop-down menu at the bottom. Existing attributes can be edited or deleted by clicking on them in the right screen. Storing or deleting attributes or complete sets is done by using the “Store” and “Delete” buttons. The attribute description provided by the model description is parsed by the application and a custom GUI for the attribute is generated as shown in Figure 5.

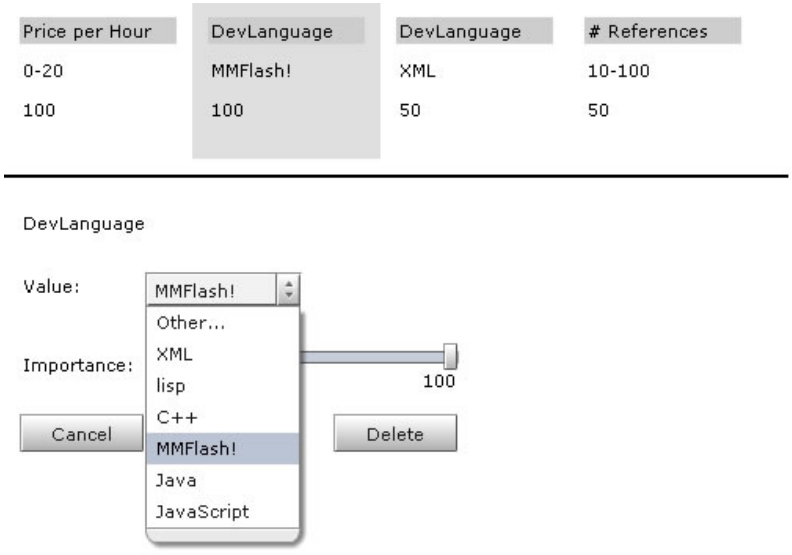


Figure 5: OpenLaszlo - Attribute editing screen

It supports all the basic attribute types used in model descriptions and ontologies. However, most of the BML models in the KB use only primitive types (either integer or string, most of the time).

2.2.3 Technical Overview

The application uses two different types of data: a dataset containing model descriptions and a dataset containing the User Profiles corresponding to the currently active model. The data itself is obtained using a Java-RPC bridge to a JAR file containing all the actual functionality. This library contains wrapper code for UPM, BML and ODM models, functionality to generate and parse OCL queries, and a wrapper to the different services. The XMI data retrieved from the KB (be it UPM, BML or ODML data) is converted into a Java data structure consisting of Arrays and Hashtables which can be passed to the OpenLaszlo code. The OpenLaszlo code then populates the datasets using the data received from the Java code. This is because OpenLaszlo is based on using datasets for data storage, but cannot easily parse XMI data.

The model data is parsed so that each attribute is mapped to a data type. The datasets can be switched by selecting different models or User Profile preferences in the application (selecting a new model shows the corresponding User Profile preferences, for example).

The data types can be: *integer* (with upper and lower boundaries, increment size), *float*, a slider (for example for percents), *boolean*, a selection list, a *selection list* with the possibility to create new entries, *strings* (or keywords). These data types can easily be extended by adding a class to the values.lzx file.

The Java wrapper is also responsible for converting data received from OpenLaszlo into XMI data (UPM preferences) and stores them into the Knowledge Base. Furthermore, the search functionality to browse BML models is also implemented in the Java code.

The KB Service is used to store and retrieve BML and ODM models, the RC Service is used to store and retrieve UPM files, and to start background queries of UPM requests, and the SR service is used to browse the BML models and retrieve Service Manifests.

2.3 USER PROFILE ECLIPSE DEVELOPMENT

The functionality of the Java-based development is very similar to the OpenLaszlo based User Profile and includes similar features. The following small tutorial will illustrate the process how to create a User Preference and how to edit it. In this example it is assumed that a user wants to create a “Hotel” preference to build the basis for a better quality on search and recommendation results provided by the Recommender. This preference could appear as following user statement: “I prefer 3-5 star Hilton Hotels in Germany”. The formal representation of this statement in the User Profile allows the Recommender to find individual results by matching the preference with all available services in the DBE. In Figure 6 the user creates a preference container called “Hotels” which will contain all Hotel relevant preferences.

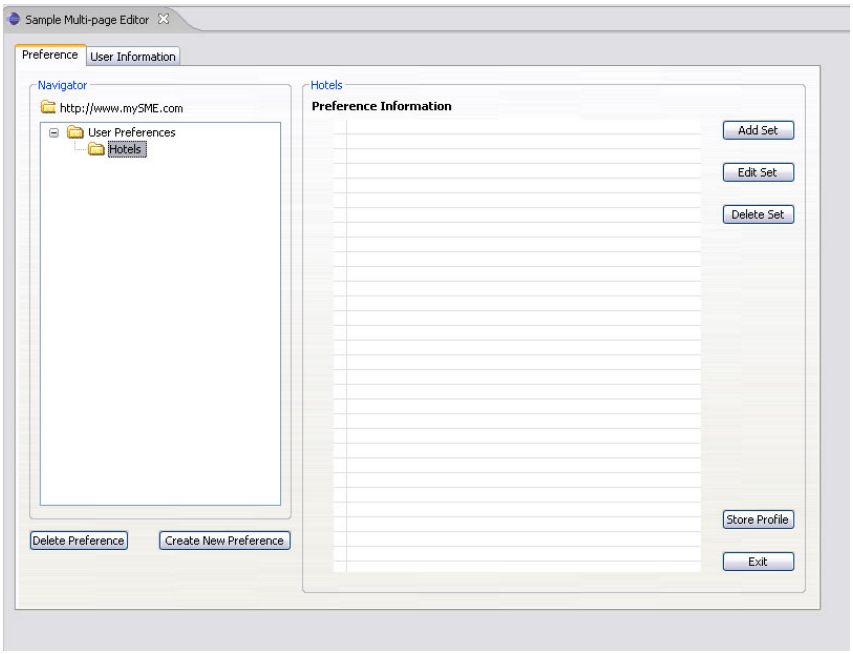


Figure 6: Create Preference Container

After pressing the “Add Set” button a window, as shown in Figure 7, appears including all BML Models available in the Knowledge Base. The user is now able to browse the models for attributes they would like to add to their preference set.

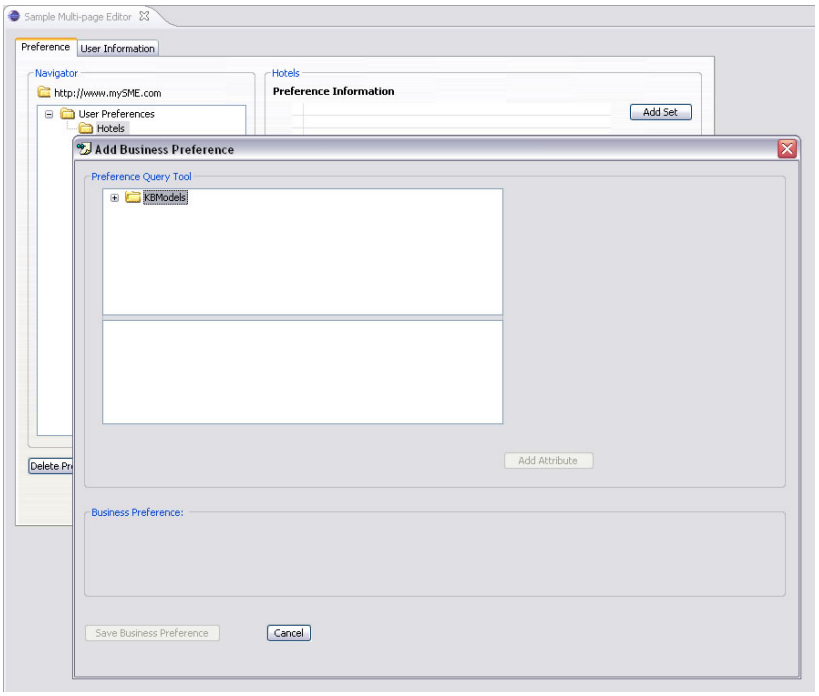


Figure 7: Add Set by retrieving BML Models from the Knowledge Base

After the user has selected a certain BML model, the referring attributes – available within the Business Entity folder – appear and can be chosen by the user as shown in Figure 8.

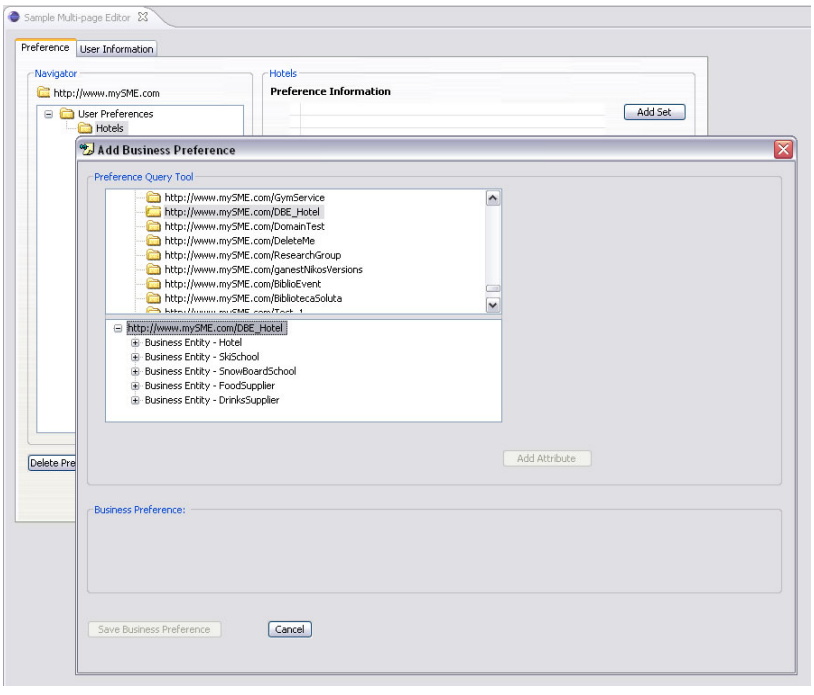


Figure 8: Select BML Model

Depending on the selected attribute type (integer, string, …) a “value window” appears and the user can enter their preferred real world information, in this case “Value = Hilton”.

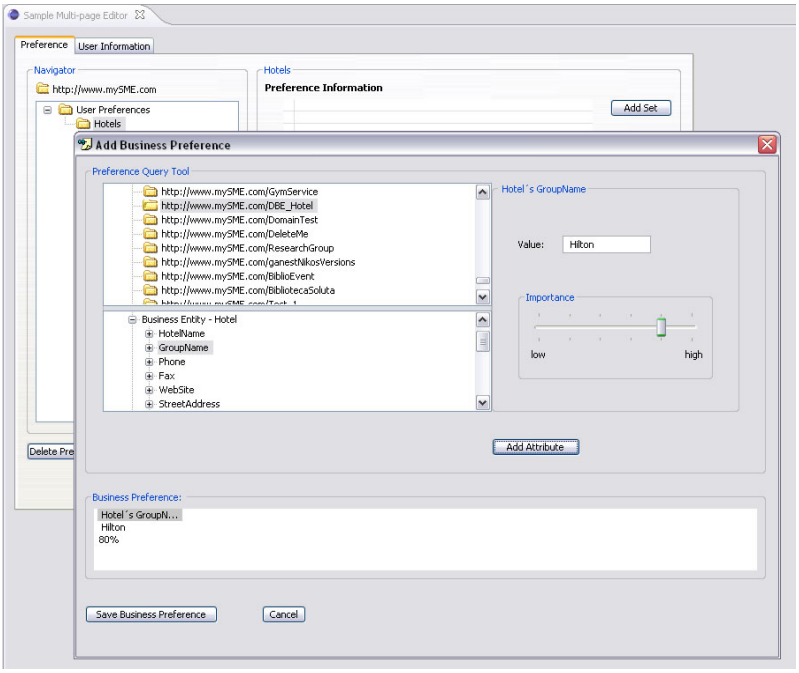


Figure 9: Browse BML Model’s Entities and select Attributes

The users repeat this procedure as often as they want to refine their preference set. Every attribute can be weighted with an importance value to express the user’s personnel rating as seen in Figure 10.

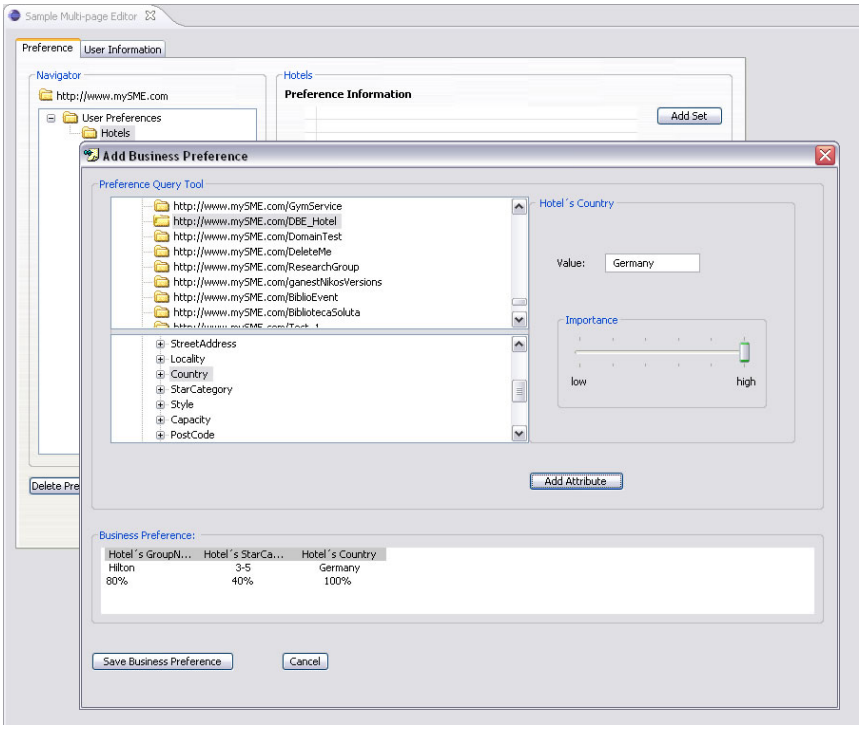


Figure 10: Add value to selected Attribute

The final preference set is presented in Figure 11.

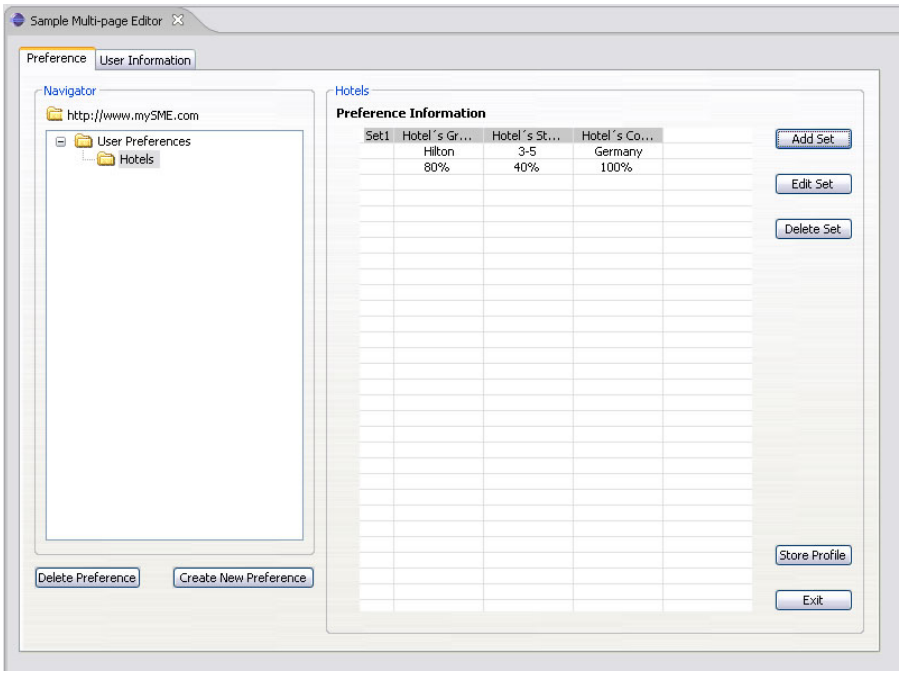


Figure 11: Preference Set for a "3-5 star Hilton Hotel in Germany"

If the user decides to change a preference set (see Figure 12) they can easily do so by selecting the respective set and edit it by pressing the "Edit Set" button.

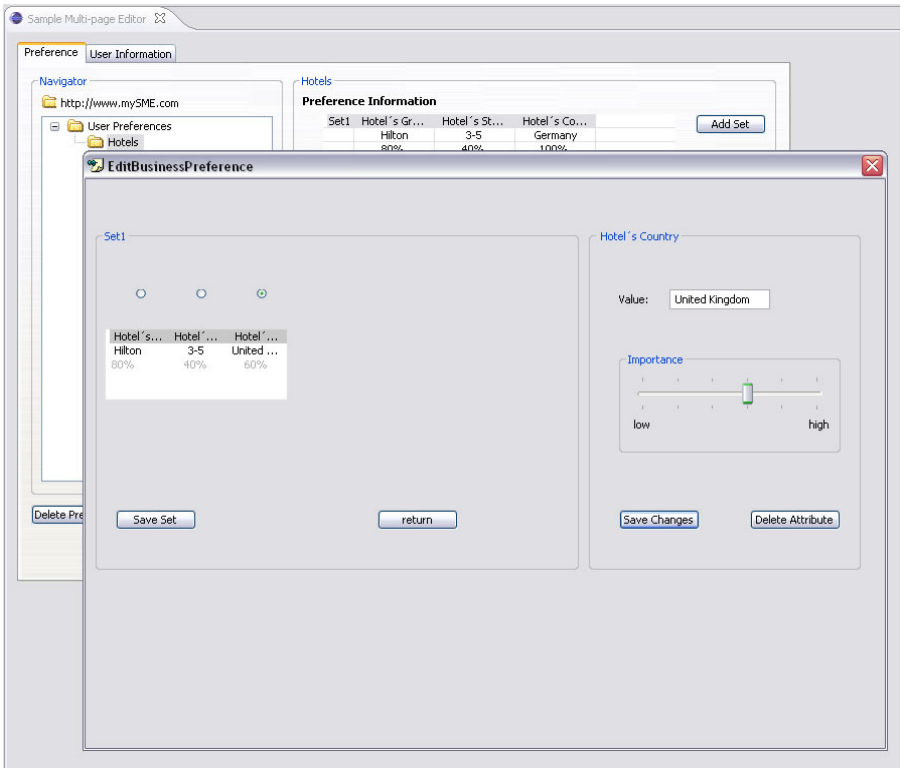


Figure 12: Edit a Preference Set's attribute and change its value

The previous screenshots exemplarily described how to create user preferences by operationalising business knowledge represented in BML models. The next chapter 3 shortly introduces the outline of the DBE environment before investigating possible User Profiling algorithms.

SECTION TWO:
CONCEPTIONAL MECHANISM EXTENSION

3 OUTLINE OF THE DBE ENVIRONMENT

As stated in Deliverable 24.3 [SBMC05, p.49], the DBE architecture is proposed to be implemented as a p2p network that consists of two overlay network topologies optimised for different types of services. The unstructured overlay, called Gradient Topology is designed for the Semantic Registry (SR), Knowledge Base (KB) and SR/KB Name Service. The p2p structure is based on the Distributed Hash Table (DHT) paradigm and is used for the Service Lookup (SL), Identity Service (IS) and Distributed Storage System (DSS).

This section briefly describes the architectural outline of the DBE p2p network. The attributes of the network are analysed and it is referred to the challenges this architecture poses to a User Profiling algorithm are highlighted. Some possible solutions to these challenges are also discussed.

3.1 ATTRIBUTES OF THE P2P NETWORK

As in most p2p networks, DBE is a decentralised network. This means that every interaction with the system requires distributed resources. Although the nodes connected to the p2p network are not identical in terms of performance and available resources, there is no central server that coordinates the functions of the network.

Since each node can connect to and disconnect from the network arbitrarily, it is not guaranteed that the communication between the nodes of DBE will be reliable per se. Referring to the Deliverable 24.3 the quality of the services offered in the DBE can be expected to be stable in general.

Scalability is an important attribute of p2p networks. A p2p network should be able to provide good performance and quality of services regardless to the number of users connecting to it.

In reality it cannot be guaranteed that no errors are likely to occur in a software system, so in the DBE. The system should therefore be designed in a way that enables it to recover from errors without external intervention (e.g. from a system administrator). Tolerance to network errors is also required. Regarding the User Profiling algorithms, the system should be able to recover from errors occurred during the calculations of the filtering algorithms. Such malfunctions could result from errors found in the User Profiles (invalid data types) but the system should be able to deal with them.

Connections over the internet are insecure by default. DBE is no exception to this rule although some (basic) security measures are being deployed. It is necessary to built mechanisms that ensure the genuine identity of each SME as well as the validity of the offered services. The respective approaches provided to the DBE (Identity Management, FADA etc.) will probably ensure these requirements. The DBE, as suggested in the previous Deliverable 24.3 [SBMC05] is a fully connected network. Within the DBE, each SME is able to locate and interact with any other available SME.

As analysed in previous research papers, the DBE is implemented as an autonomous p2p network consisting of two overlays. One overlay is used for data management and query support (structured network) and a second overlay is used for exact-match identifier routing (unstructured network). In the Figure 13 below, the architecture of DBE's p2p network is illustrated.

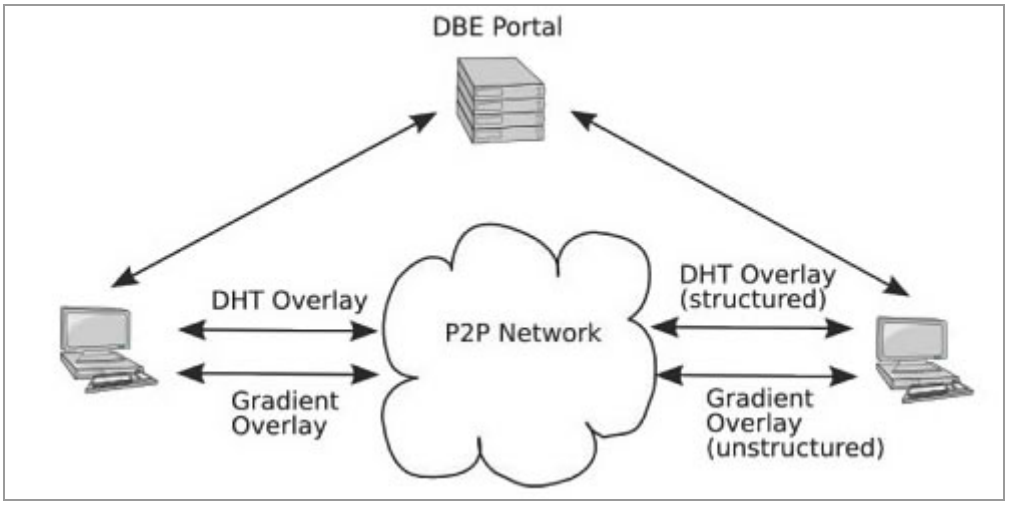


Figure 13: The basic DBE P2P architecture.

Source: [SBMC05, p.16]

The first, unstructured overlay is shared by the KB, SR and SR/KB Name Service and the second, structured overlay is used by the SL, IS and DSS. It is important to emphasise the use of Distributed Hash Tables (DHT) within the structured DBE P2P overlay topology. Each peer is assigned a unique key and has a responsibility for a certain key range. As

soon as a request is made, it is routed from one peer to another (towards the final destination). Every lookup is usually completed in $O\left(\log\left(N\right)\right)$ complexity [SBMC05, p.16].

4 THE ROLE OF USER PROFILING

4.1 WHAT IS USER PROFILING?

In recent years, due to the enormous growth of the World Wide Web, the increasing emergence of e-commerce and the tremendous growth of the available data in information systems, User Profiling algorithms have become more and more necessary. In the DBE-Project the term User Profiling refers to the “creation of a user related knowledge-base that is adapted from the user’s explicit and implicit activities within an Information System” [Bart05a].

The collection of this user-related information can be later processed and used for accurate recommendations based on the personal needs and/or habits of the user. User Profiling algorithms have been applied to many different information systems during recent years. Apart from traditional information retrieval applications, e-commerce web sites have gained the trust of their clients due to the use of User Profiling mechanisms operationalising respective algorithms. Instead of looking through hundreds of different items, users nowadays rely more and more on an automated mechanism that recommends a number of items that match their personal profile. In this way, little effort is needed from the user’s point of view and a significant amount of time is saved. Moreover, when the recommended items matches the user’s expectations, they may feel more satisfied and are likely to use the online service again.

Apart from e-commerce online applications, User Profiling algorithms have also been successfully used in media content recommendation applications (web TV/ web video) as well as in traditional information retrieval tasks. Especially within the e-commerce business, recommendation systems have become one of the most useful and powerful tools nowadays. Web sites like ‘Amazon.com’ profit by recommender systems. Since there are thousands of available items for sale, users are not able to browse them all. As a result, a single user might not be able to find a product that they will possibly need. Also, in the case that one user tries to locate a specific item, traditional search methods prove to be inaccurate. User disappointment due to the poor quality of the search results is unavoidable and therefore couterproductive for the e-commerce industry. Recommender systems provide automatic product recommendation that makes the “User-System-Interaction” a pleasurable procedure.

Examples for recommender system applications are the Middleton’s Foxtrot recommender system which provides recommendations for online research papers [MaSR03] as well as

the Elena Project[2] (a distributed e-learning environment that takes advantage of semantic Web technologies and personalization capabilities) [DHNS04].

4.2 USABILITY OF USER PROFILES

The main advantage of User Profiling algorithms is that they minimise the required effort from the user's side for discovering useful information. Instead of having to search within hundreds of documents, users only receive the results that match to their personal profile and might be relevant to their individual interests. Irrelevant information is filtered out from the system. Users always have the possibility to refine and tune the system-provided results by appraising the accuracy of these. The longer and intensive each user interacts with the system, the better and more accurate the provided results are. The user interaction might be the strongest restriction to a running algorithm as if the user doesn't interact with the system, the system cannot collect information for later recommendations.

4.2.1 Traditional mechanisms for creating User Profiles

There are many mechanisms that create User Profiles. The choice of the appropriate mechanism is indicated by the nature of the application in which the User Profile will be used. User Profiling Mechanisms can be divided into many categories. One first classification would be the division of profiling mechanisms into two classes: Mechanisms that require user input for the creation of the profile and mechanisms that work independently and create User Profiles automatically by using profiling algorithms. The first category, explicit user interaction and user input, is developed by Work Package 7. The term "assemblage-based mechanism" is used as a synonym describing this approach in this paper.

The "assemblage-based mechanism" approach is the natural and more intuitive one. Each user defines his interests and theses declared by the user preferences are later used for the information filtering process.

The second algorithm-based approach is much more complex and has many different variations. Techniques like Keyword Analysis, Social Filtering, Machine learning techniques, Artificial Neural Networks and Rule Based Filtering can be used. These techniques are introduced briefly.

[2] http://www.elena-project.org

Keyword Analysis: This is a very common method which relies on information retrieval techniques and is based on the analysis of keywords. It locates words in a given text and compares their frequency with a reference distribution from a larger corpus of text [ScWL06]. Despite its simplicity, the profiles created are not that accurate.

Social Filtering: This technique is based on the mass processing of a community of users for the creation of user profiles. Social software agents can support this procedure [GuKV97].

Machine Learning Techniques: This method uses machine learning algorithms to derive User Profiles. Software agents are mostly used for this purpose. Agents are persistent computations that have the ability to perceive, reason, act and communicate [SiYV01]. Agents take into consideration a user's past behaviour and previous experiences in order to identify the particular items that interest them. Agents can also communicate with each other in order to exchange information. Finally, to achieve better results, agents organise themselves into communities.
Two further examples of the machine learning mechanisms category are Bayesian Networks (BN) and Case-Based Reasoning (CBR). According to [ScAm00, p.2], a BN is a compact, expressive representation of uncertain relationships among parameters in a domain. A Bayesian Network is able to model the relationships between the attributes that are expressed in user queries. Case-Based Reasoning is a problem-solving paradigm that is able to utilise the knowledge acquired from previous problems in order to provide the solution of new, previously unknown problems.

Artificial Neural Networks (ANN): Neural networks can be created if a sample set of the input data items is available. Using this data set, the training of the neural network takes place. After the training procedure, a neural network is able to extract information about the user and plays the role of a User Profile that can be further used for filtering.

Rule-based Filtering: All previous methods focus on creating a content-based User Profile. Alternatively it is possible to create a rule-based profile. This can be done by questioning each user about his personal interests and his filtering behaviour. This process can generate more general rules which are the main components of a User Profile.

4.2.2 Static vs. Dynamic Profiling algorithms

Automated User Profiling algorithms are considered to be a valuable tool in computer science in general. Moreover, they concentrate many important advantages over traditional techniques and can offer solutions to many problems. Users benefit from automated algorithm-based profiling mechanisms since they do not have to spend time

describing themselves. Moreover, automated mechanisms often provide more accurate information about each user's attributes compared to the information provided by the users themselves. That is achieved exemplarily by monitoring a user's interaction with the system in real time. Nevertheless, traditional user profiling mechanisms still have advantages to offer. A form of interaction between the system and the user is still necessary for the construction of trustworthy User Profiles. It is a matter of further research to locate the ideal balance between automated algorithm-based and traditional mechanisms for User Profile creation. Nevertheless, this matter is outside of the paper's scope and will not be further investigated at this point.

5 POSSIBLE USEFUL ALGORITHMS FOR USER PROFILING IN THE DBE

5.1 INTRODUCTION AND ALGORITHM BASED MECHANISM REQUIREMENTS

As previously mentioned, the DBE is implemented as a decentralised p2p network in which no central server exists. This lack of central server coordination means that every profiling algorithm should have the ability to run in each node autonomously. In other words, User Profiling algorithms for the DBE should also be decentralised.
In the previous user profiling deliverable (D7.2) log file analysis was mentioned as a possible way to record user actions in the DBE. In this scenario, log analysers are developed and allow the retrieval of user data from the system. This data can contain user information like

- Frequency of user visits to the DBE
- Time and dates of the visits
- File Types used
- Browser/operating system of the user's PC
- Services offered and looked for

The more extensive the log analysis is performed, the more details can be acquired for each user. Of course, an excessive amount of information could result to huge computational cost and delays during operation. As a result, there will be a maximum critical mass of information that is able to provide useable results. In addition to the above, log file analysis might also be implemented as an implicit or explicit procedure. Users can provide ratings which express their opinion towards the recommendations that have been suggested to them as direct user feedback. Later on, the system automatically collects user's habits and behaviour and processes them in order to suggest new services and items.

In the case of the DBE, each node within the p2p network can be interpreted as a Small Medium Enterprise (SME) as an SME maybe has multiple nodes, or multiple SMEs share one node. Each SME offers a variety of services and is able to look for all other services on the DBE. Due to the infrastructure of the DBE, log file analysis might not be efficiently

used as a User Profiling algorithm as described in the previous Deliverable D7.2. The decentralised nature of the p2p network poses difficulties in the implementation of log file analysis. Moreover, it is assumed that the services offered and requested from the SMEs participating in the DBE are equivalent to the 'items' being part of traditional p2p networks. Recording the user's behaviour inside a p2p network might also prove inadequate for providing trustworthy recommendations. These reflections lead to the following assumptions and requirements for a potentially usable and efficient (to be proved) User Profiling algorithm.

Requirements for a potentially adequate User Profiling algorithm:

- works in a decentralised environment
- copes with the nature of a p2p network
  (non reliable connections, frequent connections/disconnections)
- provides accurate results without the need of extreme computational resources
- is able to scale when the size of the network increases
- provides the basis for trustworthy recommendations
  (matching the user's needs and interests)

Several possible solutions are discussed in the following sections. The proposed algorithms have strengths and limitations. Most of them have already been used successfully in various other applications. The experimental results from these applications are used and interpreted in order to draw conclusions regarding the value of each algorithm contributing to the DBE. Of course, the distinctiveness of the DBE's architecture is taken into consideration to allow the identification of the User Profiling algorithm with the most potential.

5.1.1 Standard Collaborative Filtering (CF)

Collaborative Filtering (CF) is one of the earliest and most successful technologies for building recommendation mechanisms. It has been widely used in many commercial (Personalised Web Pages, Recommender Systems for e-commerce, alternative methods for information retrieval) and research applications. Standard CF algorithms can be divided into two main categories: *User-Based Recommendation* and *Item-Based Recommendation* [Kary01].

In the case of the User-Based Recommendation, each user's historical information is analysed in a way that users with similar interests and history can be discovered. The identified users that have had similar habits in common with the target user in the past construct a *neighbourhood*. Once this neighbourhood is built, it is used to produce several

recommendations for the target-user. User-Based Recommendation is also known as the “nearest neighbour algorithm”. It is indubitable that the formation of the neighbourhood is a critical procedure. The quality of the future recommendations primarily relies on the correct selection of the users that build and join the same neighbourhood.

User-Based Recommendation suffers from a major drawback. As more users join the recommendation system, the computations required for the formation of the neighbourhood increase linearly. As a result, in a system with potentially millions of users, the algorithm requires very large (and possibly costly) computational resources in order to execute and these resources may not always be available when needed. This problem, which is also known as ’scalability’, is one of the most serious drawbacks of the standard CF algorithm.

There are some ways to decrease the complexity of the mentioned algorithm. One of the most common solutions is to cluster the group of the users who participate in the recommender system, and limit the neighbourhood formation procedure only to the users that belong to the same cluster. Although this approach is relatively simple to implement and accelerates the algorithm, it usually leads to recommendations of lesser quality. In order to solve the scalability problem, the Item-Based Recommendation algorithm is proposed as a more efficient alternative.

In this case, the system gathers historical information regarding specific items. It is assumed that if a user chooses a certain item, their choice can indicate a set of similar items that they will find interesting. The analysis of the connections between different items can be pre-computed. In this case, the algorithm can be used in real time applications since it is able to produce results and recommendations without the need of enormous computational resources even if the set of the available items inside the system is large. Moreover, experimental results have indicated that the accuracy of the recommended items is similar to that of the User-Based Recommendation algorithm. In the following section the entire process of item recommendation based on a Collaborative Filtering algorithm is discussed in detail.

5.1.2 How the Collaborative Filtering Algorithm works

The recommendation process that is based on a CF algorithm can be divided into three main parts introduced in the following sections:

- Representation
- Neighbourhood formation
- Recommendation generation

During the representation phase, the collected information is gathered from the user's profile and/or their activity history inside the system. The relevant information is modelled in a way that it is easily to be processed by the algorithm. After this phase, the users who might be potentially similar to the target user are looked for. If the system locates them, a neighbourhood around the target user is created. This is the second step and is called "Neighbourhood Formation". Finally, during the third and last phase, the top-N relevant users from the neighbourhood are chosen. In the following Figure 14 the three parts of the CF algorithm are illustrated.

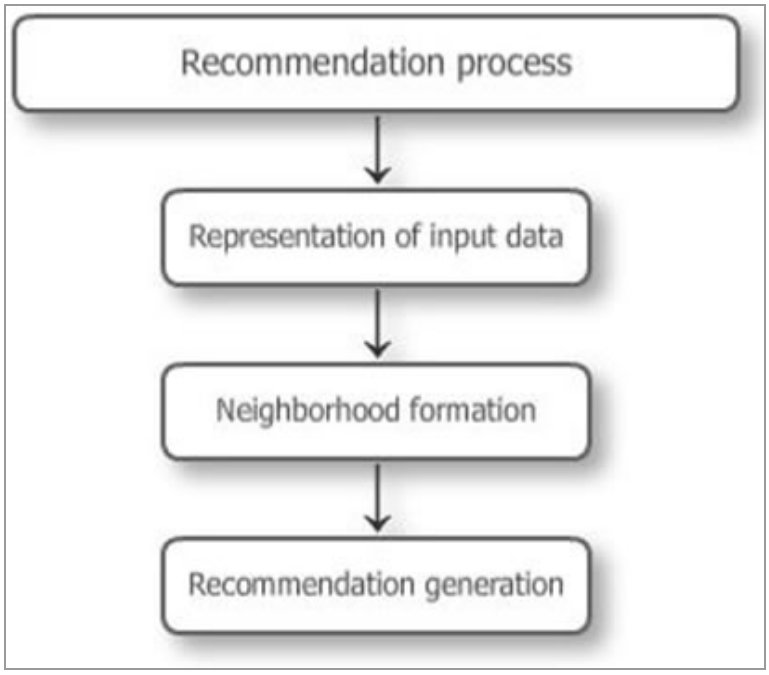


Figure 14: The 3 steps of the recommendation process.

Source: [SKKR00, p.161]

The following sections discuss possible variations in each step of the CF based recommendation procedure. Previously existing experimental results for every variation are discussed in order to identify the advantages and disadvantages of each method with respect to the DBE p2p environment. The experiments have been carried out in two different data sets: a movie recommender system called Movie-Lens[3] containing thousands of user reviews about certain movies, and an e-commerce web site containing sales information of 6.500 customers and almost 100.000 items. Although the nature of the tested recommender systems is somewhat different from the DBE, the conclusions drawn are relevant for the DBE, too. In the case of the DBE, each SME can be perceived as a user and each service it offers or requests as an item. Further reading regarding this experiment as well as the experimental results can be found in [SKKR00].

[3] www.movielens.com

5.1.3 Representation Phase

As previously mentioned, the first step of the CF algorithm is to model the input data. Almost always, the input data is represented as a $m \times n$ matrix such that $r_{i,j}$ is 1 if the ith customer has purchased in the past item j and equals to 0 otherwise. The basic advantage of this representation is its simplicity. Nevertheless, 2 significant problems arise: *Scalability* and *Sparsity*.

In a typical web based application, thousands of users and millions of items exist. As a result, the matrix created for the purposes of the CF algorithm is very large. This might lead to scalability problems. Large matrices have considerable requirements in terms of processing power and memory/storage resources. Nevertheless, it is vital that any system running a profiling algorithm is able to bear large number of individual users. Scalability is an issue of great importance and various suggestions have been made in order to deal with it.

The second and more serious challenge that arises from the above representation is called sparsity. In a typical web based application, each user interacts with a very small proportion of the total number of available items. In many cases this ratio is well under 1%. For example, a web based music e-shop contains millions of different songs (items). In this case, 1% of the items equals to thousands of items. A really small amount of users (if any) will interact with so many different items. As a result, the matrix created by the algorithm is sparse and the algorithm fails to make product recommendations. In the best case scenario, the algorithm is able to make some recommendations but their quality is low and they are useless in practice.

Possible Solutions

As described before, the original representation of the input data is a $m \times n$ customer-product matrix. The original, sparse, matrix can be transformed into a denser one using a linear-shift invariant transformation (LSI). In particular, using truncated singular value decomposition [Mood01], a new matrix is obtained which is a rank-k approximation of the original one. Since $k << n$, the new matrix is considerable smaller than the original one and contains the same amount of information. Consequently, the processing costs are smaller and the algorithm is much more scalable. Regarding the sparsity problem, it is obvious that the newly created matrix with the size $n \times k$ contains only non zero entries. This means that all the users of the system have opinions on the k meta-products [SKKR00].

5.1.4 Neighbourhood Formation phase

As mentioned earlier the Neighbourhood Formation phase is considered to be the most important step of the standard Collaborative Filtering Algorithm. In this phase a group of users with similar declared interests is created. In other words, during this phase, the algorithm tries to find for each user u_i a list of other users $N = \{N_1, N_2, ..., N_x\}$ that are similar to user u_i. This phase can further be divided into two sub-steps: the *Proximity Measure* and the *Final Neighbourhood Formation*.

Proximity Measure

During the Proximity Measure, the proximity between two users is actually measured. This can be done with two different methods

- **Correlation**
  Using the Pearson Correlation[4] the similarity between two users a and b is:

$$corr_{ab} = \frac{\sum_i \left(r_{ai} - \overline{r_a}\right)\left(r_{bi} - \overline{r}_b\right)}{\sqrt{\sum_i \left(r_{ai} - \overline{r_a}\right)^2 \sum_i \left(r_{bi} - \overline{r_b}\right)^2}} \quad (1)$$

- **Cosine**
  Proximity can also be computed by using the cosine equation. In this case the users 'User A' and 'User B' are treated as vectors of a m dimensional product-space. The angle between the two vectors indicates the proximity of the two users they represent. The cosine of this angle can be found by:

$$\cos\left(\vec{a},\vec{b}\right) = \frac{\vec{a} \cdot \vec{b}}{\left\|\vec{a}\right\|_2 * \left\|\vec{b}\right\|_2}, \quad (2)$$

Final Neighbourhood Formation

During this phase, the actual creation of the neighbourhood takes place. There are some variations regarding the formation of the neighbourhood. The two most popular schemes are mentioned here.

- The *Aggregate Neighbourhood* scheme forms a neighbourhood N of size j around user u in the following manner: First, the closest neighbour to user u is chosen. The rest $j-1$ neighbours are selected as follows. Let a certain point where t neighbours with $t < j$ can be found. Then the centroid $\vec{C}$ of

[4] Further details regarding the Correlation Coefficient r can be found under http://mathworld.wolfram.com/CorrelationCoefficient.html (Last call on 01.09.2006)

neighbourhood N is computed using the equation $\vec{C} = \frac{1}{j}\sum_{\vec{V}\in N}\vec{V}$. A user w who does not belong to the neighbourhood N $(w \notin N)$ is selected as the $t+1$-st neighbour only if w is closest to the centroid C. Then the centroid is recomputed for all $t+1$ neighbours. This routine ends when the whole neighbourhood is formatted.

- During the *Center Based* scheme, the neighbourhood around user u is formatted only by selecting the j nearest users.

The basic advantage of the Aggregate Neighbourhood variation is that each neighbour of the target user u has an active contribution in the formation of the neighbourhood. This scheme can provide significantly improved results in systems that have sparse input data sets.

5.1.5 Recommendation Generation

The generation of adequate recommendations is the final step of the standard CF algorithm. In this step, items that are relevant to the target user u are proposed by the system. Two variations of this step are examined: *Most-Frequent Item Recommendation* and *Association Rule-Based Recommendation.*

- During the first technique, a list is made for each user who is in the neighbourhood. The list contains their top-N accessed items. Afterwards, all lists from all users are compared with each other and the items that have the highest frequency counter and have not been used by the target user u are the ones that are recommended.
- In the second case, association rules are used in order to make the necessary recommendations. These rules are generated only for those users that are inside the neighbourhood. In this scenario, a minimal number of users are required for the creation of the association rules. But, fewer users will produce weak association rules which will result in low quality item recommendation.

The following Figure 15 depicts the 3 main steps of the Standard Collaborative Filtering algorithm:

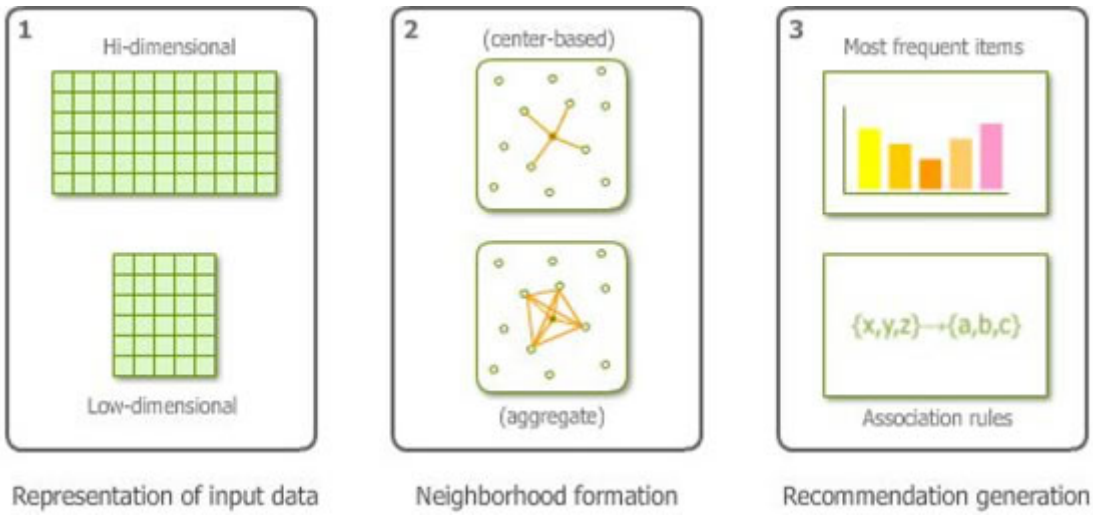


Figure 15: The 3 steps of the Standard CF Algorithm.

Source: [SKKR00, p.161]

5.1.5.1 Experimental Results

In this section experimental results conducted by [SKKR00] are interpreted and analysed in order to evaluate the impact of the CF algorithm on the quality of the recommendations considering the DBE specific requirements. As stated by the authors of that paper, "the purpose of these experiments is to explore the possibilities of combining different variations of the 3 subtasks of the CF algorithm in order to produce the most efficient recommendation mechanism" [SKKR00]. The experiments are conducted on the two recommender systems mentioned in 5.1.2.

For the purposes of this experiment the F1 metric [YaLi99] is used. The F1 metric measures the quality of the produced recommendations. In particular,

$$F1 = \frac{2 * recall * precision}{recall + precision}$$ where $$recall = \frac{size\ of\ \ hit\ set}{size\ of\ \ test\ set}$$ and

$$precision = \frac{size\ of\ \ hit\ set}{size\ of\ \ top - N\ \ set}.$$

5.1.5.2 Input data matrix dimensions

The first variable examined is the *dimensions of the input data matrix*. As described before, a large matrix leads to serious scalability problems whereas a small one leads to loss of valuable information regarding the relations between users and items. In the experiment, the ideal size of the input data matrix is sought. For the two different input data-sets the F1 metric is calculated for different size of the input data matrix. The plots of the experiments are shown below in Figure 16.

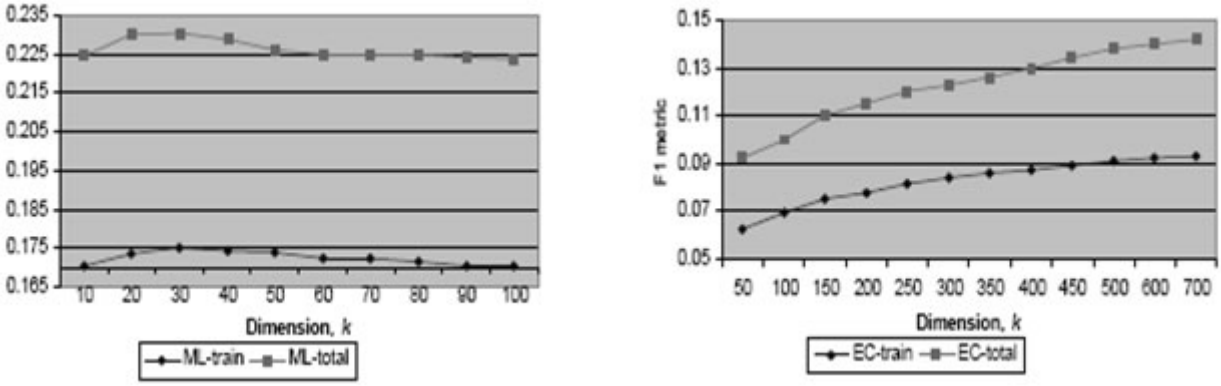


Figure 16: F1 metric in relation with the size of the input data matrix for the two different data sets. Source: [SKKR00]

In the first plot, the MovieLens data set is tested. It is observed that despite the initial increase in the F1 metric, the quality of the recommendation quality soon reaches a peak and is steady after that. On the other hand, the second data set (e-commerce web site) shows a totally different behaviour. The F1 metric continues to improve all over the way to the value 800. This contradiction can be attributed to the different nature of the two input data sets (MovieLens has significantly less users and items compared to the e-commerce web site) and to the fact that the e-commerce input matrix is much sparser.

To conclude, in sparse input data sets, the size of the final matrix plays an important role in the quality of the recommendations. In denser data sets, small matrix dimensions can be very beneficial in terms of scalability without sacrificing much precision. However, it should be noted that in both data sets the oscillation of the F1 metric is not significant. As a result, a relatively small dimension can be chosen (20 for MovieLens and 300 for the e-commerce web site).

5.1.5.3 Neighbourhood Formation

As discussed before, during the neighbourhood formation phase, two possible methods exist, the *center-based* and the *aggregated* one. For the two different data sets, an experiment that measured the F1 metric was conducted. The results of this experiment are shown in the next Table 1.

Experimental data set	Representation	Most frequent item Center-based nbrhood (F1 metric)	Most frequent item Aggregate nbrhood (F1 metric)
MovieLens data	High dimensional	0.21393	0.18928
	Low dimensional ($k = 20$)	0.22009	0.20211
E-commerce data	High dimensional	0.16654	0.11726
	Low dimensional ($k = 300$)	0.12158	0.08579

Table 1: Impact of recommendation algorithm on recommendation quality.

Source: [SKKR00, p.165]

Contrary to the previous estimations, the center-based technique seems to provide better results compared to the aggregate neighbourhood.

5.1.5.4 Recommendation Generation

To investigate which of the 2 variations (*most frequent items* and *association rules*) of the recommendation generation performs better, they have been tested using the two input data sets. The results are illustrated in the following Table 2.

Experimental data set	Representation	Most frequent item Center-based nbrhood (F1 metric)	Association rule based Center-based nbrhood (F1 metric)
MovieLens data	High dimensional	0.21393	0.20711
	Low dimensional ($k = 20$)	0.22009	0.21479
E-commerce data	High dimensional	0.16654	0.16654
	Low dimensional ($k = 300$)	0.12158	0.13209

Table 2: Quality of recommendations with respect to the different neighbourhood formation techniques.

Source [SKKR00, p.164]

For both *high and low Representation* of the input data, the two methods perform similarly in terms of quality recommendation. Nevertheless, *most frequent items* method is much simpler and has smaller computational requirements. As a result, this method is deemed

the preferable option for the recommendation generation process of the Standard Collaborative Filtering algorithm.

5.1.5.5 Conclusion

In this chapter the Standard Collaborative Filtering Algorithm was introduced. After describing its general attributes and its known limitations the three basic phases of the algorithm were examined in some detail. Possible variations of each phase were also analysed. These variations aim to further improve the performance of the algorithm. Finally, using experimental data from previous scientific work, the value of those variations were investigated in terms of speed and quality. The main deduction from those experiments is that low-dimensional input-data matrices offer similar quality of recommendations and significant improvement in speed and that they should be preferred to the original, high-dimensional matrices. Moreover, during the recommendation generation, it is advisable to use the *most frequent items* method since it is simple and has similar performance to the alternative methods.

As stated in the beginning of this chapter, Standard Collaborative Filtering is regarded as the most successful recommendation algorithm to date. It has been used efficiently in many applications. Nevertheless, in the case of the DBE, a fundamental problem exists. Collaborative Filtering is by nature a fully centralised algorithm. This means that the computations required for the creation of the user's neighbourhood should be made in a central server. Nevertheless the DBE is based on a P2P architecture where no central node exists. As a result, all necessary computations should be made in the peers of the network. This is the major drawback Standard Collaborative Filtering Algorithm suffers from. One possible solution will be discussed extensively in the following section.

5.2 DISTRIBUTED COLLABORATIVE FILTERING

5.2.1 Introduction

As described in the previous section, distributed collaborative filtering seems to be the most promising solution to challenges that arise from the nature of the p2p platform on which DBE is implemented. Until now, various efforts have been made for the organisation of information within p2p networks. Most of them focus on making meta-data based content search possible by means of a keyword search [WRLP05, p.1].

Although many of these methods offer significant improvements, none of them is equivalent in terms of performance with standard Collaborative Filtering. Distributed CF on

the other hand is a fully decentralised variation of standard CF that is able to organise the available information within a p2p network and to produce item recommendations for each user with the same quality as the standard CF algorithm. The following aggregated definition from [WRLP05] describes the distributed CF algorithm best: Distributed Collaborative Filtering Algorithm is a self organizing, distributed, binary collaborative filtering approach that can be used in a p2p system and offers statistically equivalent results to the traditional CF algorithm. In the case of the DBE, the design and implementation of a recommendation mechanism that can operate on a P2P environment is needed. The major problem that has to be solved is that in the P2P network no central server exists. As a result, User Profiles can't be stored in a central database to run the CF algorithm in a way that relevancies between users can be created. Therefore, a fully decentralized collaborative filtering algorithm has to be implemented.

The scientific publication of Wang et al "Distributed Collaborative Filtering for Peer-to-Peer File Sharing Systems" [WRLP05] seems to be a very promising approach which is introduced and analysed in the following pages. In this approach the use of so-called *"Item Buddy Tables"* is suggested. In general, buddy tables store information regarding the similarity between the available items in a P2P network. These similarities can be used later on to build item recommendations for each user who is logged on the P2P network. Although buddy tables are among the most important features of the Distributed CF algorithm, many other additions have been made to the standard CF model.

In the next section the architecture of the algorithm is introduced and a closer look is taken at how recommendations are made.

5.2.2 Algorithm Description

In this section, the outline of the distributed CF algorithm is described. Although this algorithm has already been introduced and implemented within a file sharing p2p network, some thoughts for necessary alterations are made in order to conceptionally adjust the algorithm to the DBE specifications. As mentioned in the beginning of this deliverable an assemblage-based mechanism implementation has been preferred to an algorithm-based mechanism implementation. Therefore the proposed alterations are subjected to further discussion. Their main purpose is to suggest a work frame in which the algorithm-based Profiling Mechanism will be implemented in future.

5.2.2.1 Definitions and Representations

The elements of the DBE p2p network are represented as follows:

Peer: A peer is every SME that is connected on the network and is represented by $P_i, i = \{1,...M\}$ where M is the total number of SME's connected to the DBE.

Item: In DBE, each SME offers or needs one or more services. For example a small software development company might offer services like "credit card verification" or "search engine web service" and another SME like a Hotel might request services like "catering service" or "cleaning service". These services are available items within the p2p network and are represented as:

$$R_t = \left\{ I_i^k \ is\ available \mid i \in \{1,...,M\}; k \in \{1,...,K_i\} \right\}$$

For instance, item I_2^3 stands for the 3rd item (service) of user (SME) number 2.

Shelf: Shelf is a set of services (items) each peer is willing to share within the network.

Transaction: If a peer A accesses some service offered by peer B a transaction takes place.

Cart: Each peer has a cart that contains all the services that it has accessed in the past. A cart is represented with:

$$C_i = \left\{ I_j \mid P_i \ has\ once\ accessed\ item\ I_j \right\}$$

The role of the cart is very important. It represents each user's past interaction with the other peers of the network. As a result, it implicitly defines each user's personal preferences. It might therefore play a major role in the recommendation process.

5.2.2.2 Bayes Rule

At this point the Bayes Rule should be briefly explained as this will be used frequently in this section. Mathematically, the Bayes Rule states that

$$posterior = \frac{\text{likelihood} * \text{prior}}{\text{marginal likelihood}}.$$

In symbols, the rule can be expressed as:

$$P(R=r \mid e) = \frac{P(e \mid R=r) * P(R=r)}{P(e)} \ .$$

Where $P(R=r \mid e)$ denotes the probability that random variable R has value r given evidence e.

The denominator is just a normalizing constant that ensures the posterior adds up to 1; it can be computed by summing up the numerator over all possible values of R,

$$P(e) = P(R=0,e) + P(R=1,e) + ... = \sum P(e \mid R=r) * P(R=r)$$

This is called the marginal likelihood (since we marginalize out over R), and gives the prior probability of the evidence.

5.2.2.3 Relevance Model

In the distributed CF algorithm, a relevance model is used in order to formulate the queries made by the peers with respect to the available items (services). For this model, a binary random variable R is introduced. Since this variable is binary, it takes on two values: r for relevant and $\bar{r}$ for not relevant. This variable represents the relevance of an item towards a certain peer. As mentioned before, the peer's preferences are expressed by their individual User Profiles.

In order to perform the required information filtering, the relevance between a target item I_t and a user's profile $P_{i,t}$ needs to be calculated. After these probabilities are calculated for every single item, they can be used in order to rank items that the user doesn't posses. The relevance rank of a target item I_t for a certain peer P_i can be calculated from the following equation:

$$R_{I_T,P_i} = \log \frac{P(r \mid I_T, P_i)}{P(\bar{r} \mid I_T, P_i)} \quad (3)$$

In equation (3) $P(r \mid I_T, P_i)$ is the probability that item I_T is relevant to peer P_i. In the same fashion, $P(\bar{r} \mid I_T, P_i)$ is the probability that item I_T is not relevant to peer P_i.

After various mathematical transformations and the application of the Bayes Rule (described in detail in [WRLP05]), the following equation describes the relevance rank:

$$R_{I_t,P_i} \propto \sum_{q=1}^{Q_i} \log \frac{P\left(r \mid I_T, c_i^q\right)}{P\left(r \mid I_T\right)} + \log P\left(r \mid I_T\right) \quad (4)$$

It is obvious that the prior relevance probability of the target item as well as the new relevance probability of the target item needs to be calculated if the target item is combined with another one.

If equation (4) is executed for every single peer, the outcome includes all the relevancies between the peers and the items of that are available on the network. In the case of a centralised algorithm that would be sufficient. All data would be collected on the central server and with the appropriate process, recommendations would occur. Nevertheless, in the case of a p2p network these relevancies need to be broadcasted throughout the network (flooding technique). Although this technique is quite simple it is not very efficient. As a result, a dynamic way of building the relevance links is proposed.

5.2.2.4 Item-Buddy Tables

Buddy tables are without doubt one of the most important aspects of the Distributed CF algorithm. The quality of the algorithm is mainly based on the item-buddy tables. These tables contain information regarding the relevancies between the services that are offered within the DBE. This information is derived from a user's profile and is stored in a distributed way in each item. The functionality of buddy tables is described in the following section.

How Item-Buddy tables "work"

If one peer expresses a preference for two services (let them be I_a and I_b) then the relevance between those two services increases. This means that since one user is interested in these two services, they should match the user's profile. Moreover, the fact that they are both selected from a single user should be taken into account. This can be done by updating the relevance list of the two services in the following manner:

At a given time, the relevance between two items is updated according to the following equation:

$$\Delta P_t\left(r \mid I_a, I_b\right) = \sum_{\forall T_k : t-\Delta t<k<t} \Delta P_k\left(r, I_a = item\left(T_k\right), I_b \in C_{peer(T_k)} \mid I_a, I_b\right) + \sum_{\forall T_k : t-\Delta t<k<t} \Delta P_k\left(r, I_b = item\left(T_k\right), I_a \in C_{peer(T_k)} \mid I_a, I_b\right) \quad (5)$$

It is obvious that the relevancies between services I_a and I_b are updated only if those services are accessed by a peer. In other words, the relevance between two services can be updated using only the information about the service that is being accessed and the cart of the peer that is accessing this service. If one peer declares their interest in a service offered within the DBE, this preference should be depicted in their profile.

Equation (5) shows that the relevance probabilities between two services can be calculated incrementally. These probabilities need to be stored in each peer of the network. In order to achieve this task *buddy tables* are used. They contain valuable information (location and relevance probability) about the top-N relevant services that are available on the network. Each time a service is accessed by a peer, its buddy table updates with the new relevancies. As described before, this update is performed according to equation (5).

5.2.2.5 Final Step: Recommendation

With item-buddy tables, the problem of knowing the top-N relevant neighbours of each peer is solved. Consequently, the algorithm can form a neighbourhood with similar items (services) that will be used for the production of item recommendation. The final step of the algorithm will therefore be the recommendation generation. Using equation (4) the algorithm generates a recommendation in the following way [WRLP05, p.5]:

$$R_{I_T, P_i} \propto \sum_{q \in \{1,\ldots Q_i\} \cap I_T \in c_i^q.BT} \log \frac{R_{I_T, c_i^q}}{R_{I_T}} + \sum_{p \in \{1,\ldots Q_i\} \cap p \neq q} \log \frac{\partial}{R_{I_T}} + \log R_{I_T} \quad (6),$$

where ∂ is the default ranking for the target item that does not present in the buddy tables and $c_i^q.BT$ denotes the buddy table of item c_i^q. R_{I_T} represents the overall relevance of item $I_T\left(P\left(r \mid I_T\right)\right)$. Finally, the ranking recommendation equation becomes:

$$Rank_{I_T, P_i} = \sum \log R_{I_T, c_i^q} - \left(\left|q\right| - 1\right) \log R_{I_T} \quad (7),$$

where $q \in \left\{1,..,Q_i\right\} \cap I_T \in c_i^q$. BT and $\left|q\right|$ is the number of elements belonging to q.

5.2.2.6 Experimental Results of the Distributed CF algorithms

After describing the architecture of the algorithm and the way it manages to produce item recommendations, experimental data [WRLP05] of the algorithm running on an actual p2p system is presented. The purpose of those experiments was to prove both the validity of the algorithm and to compare the quality of the recommendations produced to those of the traditional, centralised Collaborative Filtering Algorithm. An adaptive implementation of the distributed CF algorithm into the current User Profile components of the DBE won't be possible due to a lack of resources in WP7.

At this point the theoretical background and requirements for a possible User Profiling algorithm-based mechanism has been presented and it builds a strong basis for further implementation activities. Therefore a first evaluation is presented for implicit user knowledge basing on experimental results available in [WRLP05]. In this scientific research, a distributed CF algorithm has been implemented on a music exchange p2p network simulation. The music data was generated from a community called Audioscrobbler[5] . At the time the experiment took place, 6.359 peers and 857.020 items (music files) were available on the p2p network. Testing data was filtered (inactive users were removed, wrong artist and song titles were corrected, users with less than 2 items were removed) in order to minimize error in the recommendations produced.

Finally, after the filtering process, 3854 users and 10869 items were left. This data set was afterwards randomly divided into two parts. The training set (80% of the users) and the test set (20% of the users). The training test was used to build the buddy tables, the relevance links and the recommendations. The test set was used to evaluate the accuracy of the produced recommendations.

The test was divided into two major parts: the observation of the relevance links and the measurement of the recommendation performance and quality. These two parts are described in the following sections.

5.2.2.7 Self Organising Relevance Links

As mentioned in the previous sections, a transaction takes place if a peer accesses one or more items in the network. In the case of the music exchange p2p network, each time one user downloads a song, a transaction takes place. During this transaction the relevance links in the buddy table of the item being downloaded are updated. Moreover, a new link can be created if this user has never downloaded a similar song. In the Figure 17 below, the relevance links created between the song artists are shown. For each item links to the top-5 relevant items are shown as directed arrows.

[5] www.audioscrobbler.com

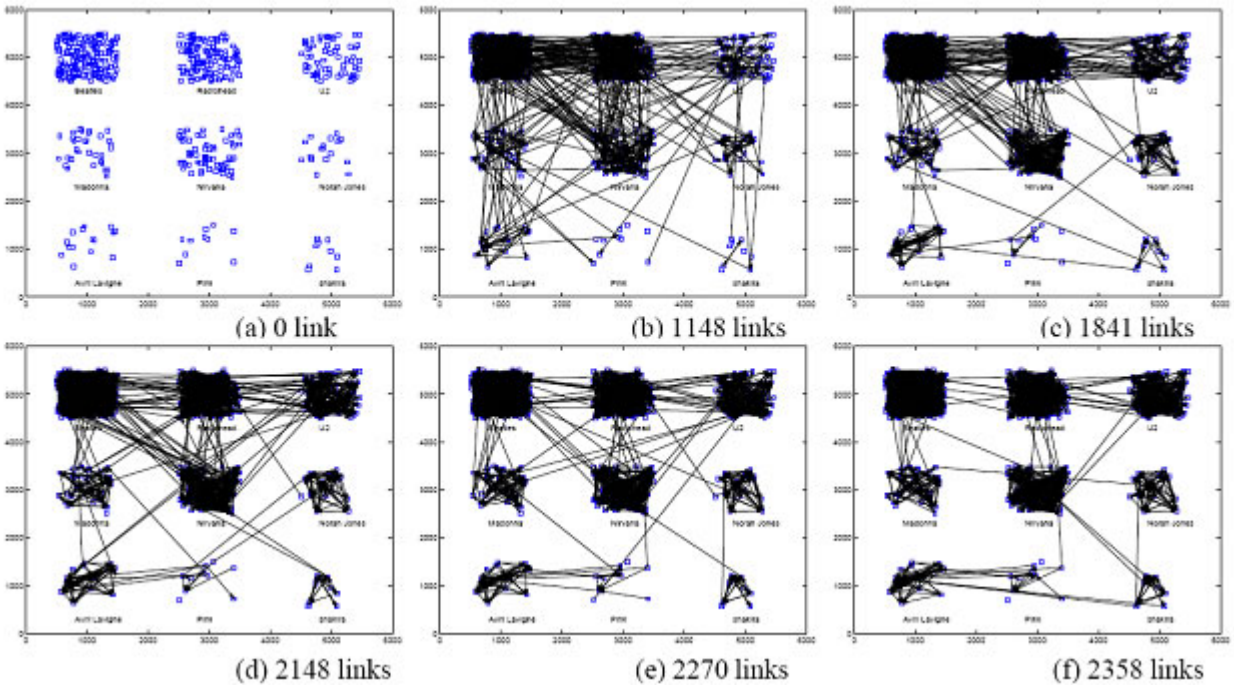


Figure 17: Relevance links created between the songs of nine different artists.

Source: [*WRLP05, p.7*]

As expected, the more transactions that take place within the network, the more links are created. The largest number of links is created within songs of the same artist. This is of course expected. Songs of the same artist are more relevant to each other compared to songs from other artists. Another important observation is the fact that songs that belong to the same music genre have many more relevance links. For instance many relevance links have been created between songs written by U2 and Radiohead. These songs belong to the rock genre and it is natural that will be chosen by users that have a preference for rock music.

5.2.2.8 Recommendation Generation Performance

Two metrics are used for the evaluation of the recommendation performance: *Coverage and Precision*. In order to measure these two metrics for each user 50% of their items are put into their cart. The remaining 50% are called “ground truth items”. A recommendation process is performed afterwards for the “ground truth items” set and the two metrics are calculated. Coverage measures the proportion of the “ground truth items” that are recommended by the system. Precision measures the percentage of the recommended items that are also “ground truth items” The Figures a and b in the Appendix show the recommendation results. These results indicate that recommendations tend to improve if the number of transactions increases. Figure c contains valuable information regarding the performance of the Distributed CF algorithm compared to traditional, centralized algorithms. It depicts the normalised error of the recommendation produced with:

1. Distributed CF algorithm (with the prior term of equation 4)
2. Distributed CF algorithm (without the prior term)
3. Centralized top-N item based recommendation
4. Centralized user-based top-N recommendation
5. Recommendation based on average ranking

It is clear that Distributed CF algorithm produces equal or even better recommendations than the traditional centralised methods. Recommendation accuracy results compared to other CF algorithms are shown in detail in the Appendix.

5.3 ONTOLOGIES

5.3.1 Introduction

During the last two decades, ontologies have proven to be a very important and promising tool in computer the science field. Although ontologies were first treated as *esoteric fields in philosophy that study being* [Pret04], nowadays ontologies constitute a critical part of the semantic web with numerous practical applications in various fields of science. But what is an ontology? According to Thomas R. Gruber [Grub93], an ontology is *"an explicit specification of a shared conceptualisation"*. With the term "conceptualisation" we describe an abstract and simplified view of the world or domain of interest, which is being represented. In this world (or in that domain of interest), objects or entities exist. These entities are connected to each other with specific relationships. According to Halpin [Halp99], these relationships can be interpreted as roles that the entities play with respect to each other. The term "*explicit specification"* means that the concepts and relationships of the abstract model are modelled in explicit terms and definitions.

To sum up, an ontology is a designed model that uses specific vocabulary in order to describe specific entities that belong to a domain of interest. Moreover, a set of assumptions exists which links the entities with each other and describes the meaning of each term in the vocabulary [Guar98].

Ontologies have many practical applications. These applications can be summarised in the following categories [RuLe0204]

- Communication
- Computational Inference
- Reuse and organization of knowledge

As far as ‘Communication’ is concerned, ontologies provide a common vocabulary that supports the communication between implemented computational systems, between humans or between humans and computational systems.

If used for ‘Computational Inference’, ontologies support the internal representation and manipulation of plans and planning information or they are used in order to analyse the internal structures, algorithms, inputs and outputs of previously implemented systems.

Finally, ontologies prove to be very useful for the structure of libraries and the planning of information.

To further analyse the concept of ontologies, the following practical example is proposed. In an implemented information system, two software agents communicate using a shared vocabulary. A term of this vocabulary is the term “mouse”. This term can either refer to the input device known as “mouse” or to the identically named rodent. If no ontology exists, there is no way for the agents to discern between the two meanings. Nevertheless, if an ontology is established, the world in which the two agents operate is clearly described. As a result, it is clarified that each reference to the term “mouse” refers to the input device. In other words, the concept “mouse” has a certain role and the ontology is a supporting element for the communication of the two software agents by providing a background knowledge that rules out irrelevant references.

5.3.2 Ontology: Design and Construction

The process of designing and implementing an ontology can be complicated. Moreover it is time consuming and often requires a great deal of resources in order to be successfully carried out. Another crucial factor that determines the effectiveness of a designed ontology is the previous experience of the designers. Various examples exist, where ontologies with correct formal properties were created but couldn’t capture the intended semantics of the user’s terminology correctly. In this case the ontology has little practical use and has to be replaced with a better one.

Another challenge the designers of ontologies have to deal with is the fact that “every ontology requires consensus across a community whose members may have radically different vision of the domain that is under consideration” [RuLe02]. In order to satisfy a general consensus, two major methods are followed. Either small ontologies are designed from a large number of people and are then merged into one, or Standardisation Organisations suggest formal ontologies which everyone has to follow. Each of the two extreme suggestions has its own strengths and weaknesses. The final decision on the

type of ontology that is used usually depends on the nature of the developed information system.

Gruber [Grub95] suggests an approach considering five design criteria. According to this paper, these criteria can be used for objective evaluation of any ontology. Moreover, if these suggestions are followed during the design phase, an efficient, working ontology can be created. These criteria are:

- Clarity
- Coherence
- Extendibility
- Minimal encoding bias
- Minimal ontological commitment

Every definition inside ontologies should be clear. That means that definitions must be objective, complete and independent of any social or computational context. In that way, it is ensured that each concept has the minimal amount of possible representations. This results in efficient communication between software agents. The coherence “rule” suggests that only inferences consistent with existing definitions should be allowed. Any ontology should be able to be extended without having to alter the initial definitions. This property is called extendibility and is one of the most important attributes of any given ontology. It ensures the effectiveness of the ontology in the course of time. The minimal encoding bias criterion suggests that every conceptualisation should be specified at a knowledge-level. Finally, in every ontology the minimum ontological commitment sufficient to support the intended knowledge sharing activities should be allowed [Pret04]. Also, the weakest theory necessary to facilitate communication consistent with the conceptualisation should be specified. In this way, software agents have the ability to extend the ontology according to their individual needs and purposes.

At this point, the difference between ontologies and a knowledge-base should be clarified. Although they have many features in common, they also share different objectives. As stated in [Grub95], *ontologies aim to capture the conceptual structures of a domain while a knowledge-base aims to specify a concrete state of the domain*. In another words, ontologies consist of a set of characteristics that is used to distinguish various concepts whereas a knowledge-base comprises of entity instances.

These sections introduced the basic qualities and characteristics of ontologies. Basic design and construction principles for ontologies have been described. Finally the differences between ontologies and a knowledge-base were briefly discussed. The following section tries to analyse how ontologies might improve recommendations in a

recommender system that relies on Collaborative Filtering techniques for the generation of item and user recommendations.

5.3.3 Ontologies and Collaborative Filtering

As described in the beginning of chapter 4, Collaborative Filtering algorithms suffer mainly from the following three limitations

- Scalability problems if the number of users becomes very large
- Less reliable recommendations if the input data set is sparse
- Inability to produce recommendations for newly added items or users that do not have recorded history.

For the first two limitations, item-based Collaborative Filtering was introduced as an improvement over the Standard user-based Collaborative Filtering algorithm. With the item-based CF algorithm, similarities between the available items are calculated offline. As a result recommendations can be produced quickly without consuming many system resources. It has also been demonstrated that with this technique the prediction accuracy that is achieved is comparable to that of the standard user-based CF algorithm

Even the most efficient item-based recommendation algorithms still suffer from problems associated with data sparsity. Moreover, all present algorithms lack the ability to provide the users with trustworthy recommendations for a newly added item for which no recorded history exists. This problem is described by Middleton in [MiAY02] as the *Cold Start Problem*. In fact, the problem in [MiAY02] is divided into two categories: The *new system cold-start problem* which appears in new systems where no User Profile or initial rating from users exists and the *new-user cold-start problem* where no information is available for a new user who recently logged on the system.

Ontologies could be the solution to the above described limitations of the Collaborative Filtering algorithms. In [BaXY04] a possible combination of ontologies and CF is proposed. There, CF is semantically enhanced. This means that knowledge regarding items is extracted in an automatic manner. This knowledge is based on a previously defined ontology and serves the purpose of supporting the user's profiles. In this way, User Profiles contain more useful information which leads to improved item recommendations. The "enriched" User Profiles contain additional knowledge taking the user's preferences and dislikes into account. This fact increases accuracy and coverage of the recommendations made by the system. In the case where no previous information regarding a new user exists, the system is able to use the semantic similarities in order to provide recommendations. These recommendations might not be 100% accurate but experimental results (see section 4.3.4) appear to be very satisfying.

In order to take advantage of the semantic information regarding the items of the recommender, items according to a previously described reference domain ontology need to be extracted and classified. This ontology not only defines the items but also clarifies their semantic relationships. In this way, instances of the ontology classes are extracted from relevant web pages. With the completion of this extraction, a semantic attribute matrix is created. This matrix is processed in order for the noise to be reduced and to collapse highly correlated attributes. This is done by using Latent Semantic Analysis (LSI), a procedure which reduces the dimensions of the initial matrix by removing the noise and therefore reducing the complexity of the algorithm. LSI is often implemented with help of the Singular Value Decomposition (SVD) technique. After the completion of this step, the item similarities are computed. For this computation both the user-item matrix and the semantic attribute matrix are used. In order to produce the final similarity and the item recommendation, a linear combination of the two previously described similarities is used.

In the following section the experiment conducted in [BaXY04] is described. In order to prove the theoretical assumptions regarding the improvements in recommendation performance and accuracy if ontologies are combined with CF, the authors have implemented a recommender system which provides recommendations for movies. Also, an ontology regarding movies is created and combined with the CF algorithm. The implementation of this semantically enhanced recommender is briefly described first and the experimental results are discussed afterwards to classify the relevance of this approach in the context of the DBE.

5.3.4 The experiment

For the implementation of this experiment, the data set was obtained from the MovieLens[6] recommendation system which has been introduced in the beginning of chapter 4. This system consists of 1682 movies that have been rated by 943 unique users. 100.000 ratings exist in the system in total. Based on a custom-made ontology, a web agent was implemented. The role of this agent was to extract movie instances from the Internet Movie Database[7] which is a web site that offers information regarding movies. The structure of the ontology used can be seen in the next Figure 18.

[6] http://www.movielens.org
[7] http://www.imdb.com

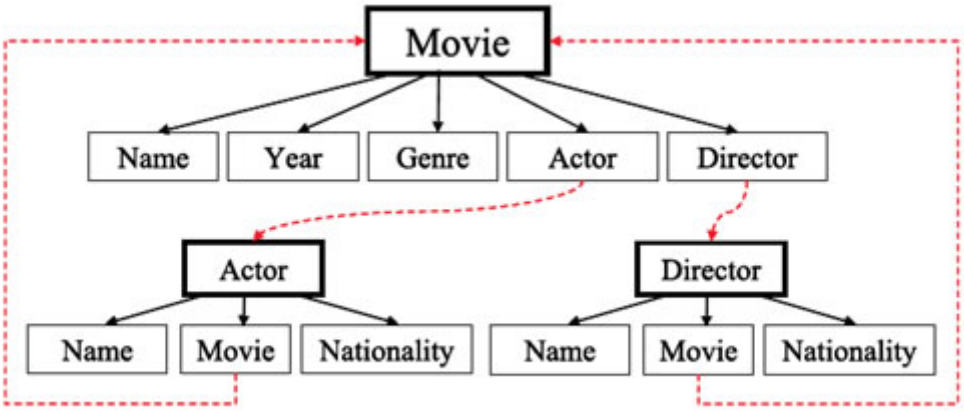


Figure 18: Representation of the ontology used to capture instances from the imdb.com movie web site.

Source: [BaXY, p.8]

The extracted instances from www.imdb.com web site were converted into a binary table where each row represents a movie and each column a unique attribute of this particular movie. Finally SVD transformation was performed on this table resulting in the final semantic similarity matrices. These matrices were combined with the rating similarities matrices that were produced by the original ratings data. Finally, to evaluate the accuracy of the produced recommendations the *Mean Absolute Error (MAE)* metric was used (see equation 8)

$$MAE = \frac{\sum_{i=1}^{n} |a_i - p_i|}{n}, \quad (8)$$

where a_i is the actual rating, p_i the predicted rating and n the total number of movies. Finally, a parameter α was added in the equation. This parameter represents the degree to which the semantic and rating similarities are used in the neighbourhood formation phase. In other words, with parameter α the impact of the ontology on the outcome of the recommendation process can be adjusted. If $\alpha = 0$ only the semantic similarity is used (100% ontology dependant) whereas if $a = 1$ only similarity rating is used (the ontology plays no role).

In the following Figure 19, the accuracy of the recommendations for both the traditional CF algorithm and the hybrid CF-ontology algorithm is depicted.

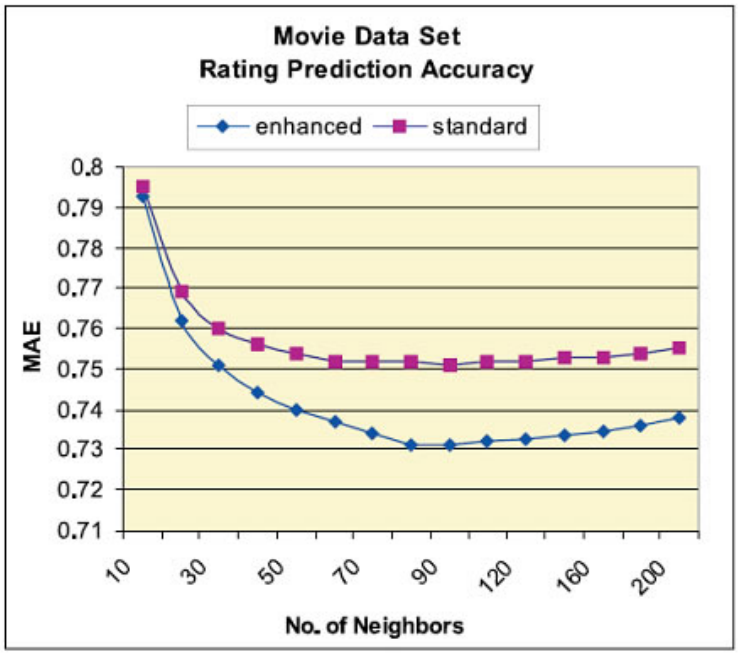


Figure 19: Recommendation Accuracy (MAE) for standard CF vs. semantically enhanced algorithm.

Source: [BaXY, p.13]

It is obvious that the semantically enhanced algorithm produces recommendations that are more accurate (MAE is lower). Moreover, due to the fact that the semantic matrix is pre-processed, the computational performance of this method is much better.

Another aspect is the impact of the ontology/similarity rating ratio to the accuracy of the semantic enhanced algorithm. Therefore an experiment was carried out where the MAE was calculated for different values of the α parameter. As it is clearly shown in the next Figure 20, the best results were achieved when using a combination parameter $\alpha = 0.4$. Moreover, the results indicate that for sparser data sets the semantic approach achieves larger improvements in the overall accuracy.

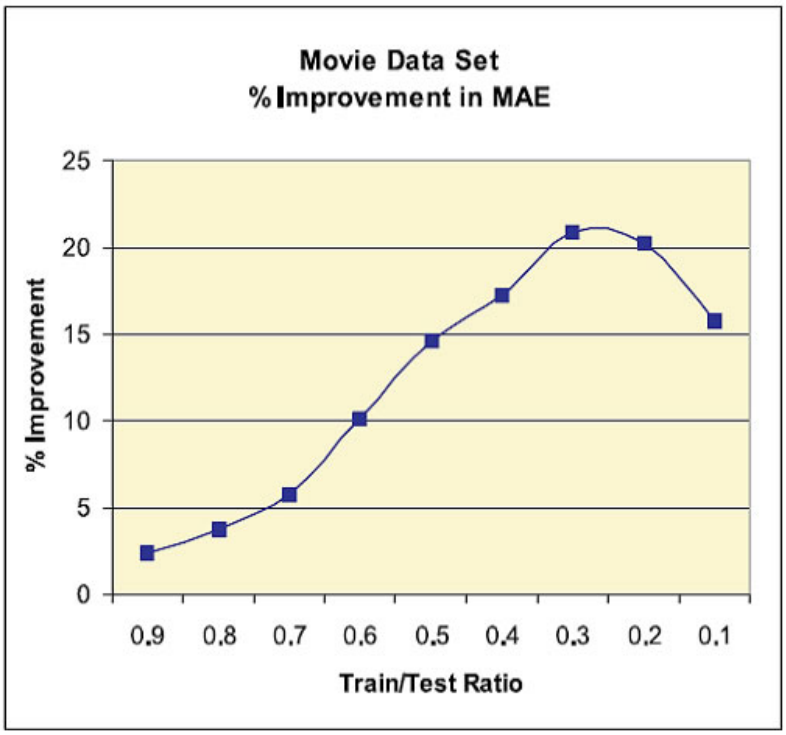


Figure 20: Improvement in MAE for different test/train ratios.

Source: [BaXY, p15]

The final goal of the experiment was to clarify the impact of ontologies on the so called *cold start problem*. As described before, this problem appears if recommendations are requested for new items for which no valid profile or previous user rating exists. To conduct this experiment, only new movies that had no user reviews were selected. Both algorithms were executed and the results obtained are shown in the following Figure 21.

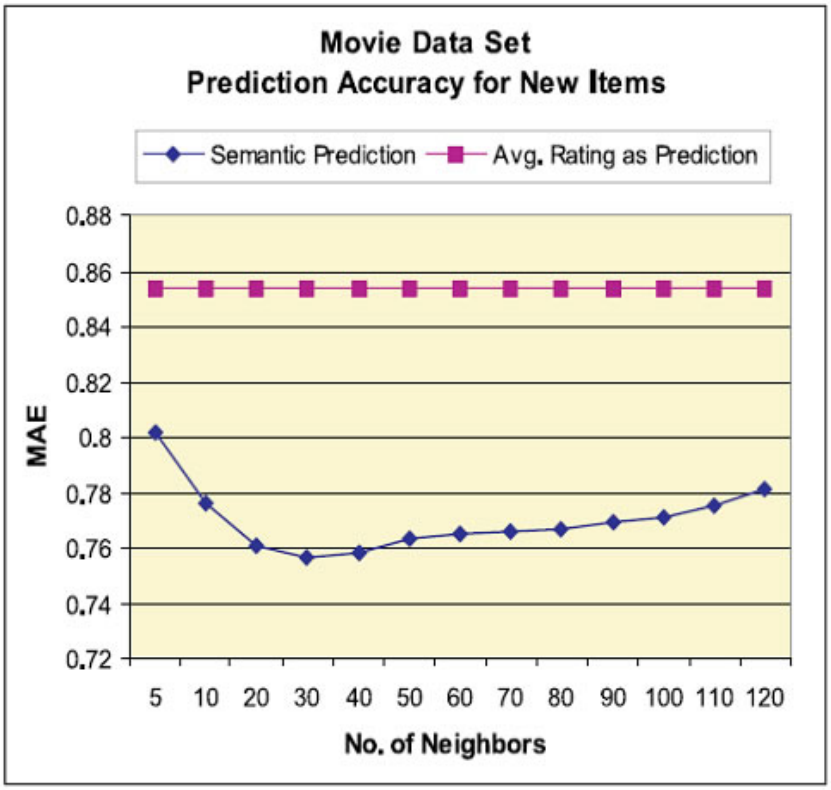


Figure 21: Evaluation of the semantic enhanced algorithm for new items (no previous rating).

Source: [BaXY, p.16]

It is clear that the semantically enhanced algorithm performs much better than the traditional CF algorithm. Regardless of the size of the data set the enhanced algorithm produces more accurate recommendations, partially solving the *cold-start problem*.

5.3.5 Conclusion

The aim of this section was to research whether the addition of Ontologies in a recommender system could be beneficial. The results indicate that this goal has been reached. After the provision of an ontology overview the fundamental rules that have to be followed during the design of an ontology where introduced. Then the main limitations of standard Collaborative techniques finally present experimental evidence that ontologies can indeed contribute to the production of more accurate recommendations that require fewer computational resources. Moreover, it was shown that a semantically enhanced recommendation algorithm might overcome the *cold-start problem*, a situation with which standard techniques fail to cope. This is an important aspect as probably every new user joining the DBE network will have to cope with the cold-start problem as long as no history exists. Finally, it was observed that the application of semantic analysis reduces the noise of the input data set and further improves the performance of the recommendations. The hybrid approach performs better compared to either the traditional CF algorithm or the semantic-only recommendation.

5.4 SEMANTICALLY RELATED NODES

5.4.1 Introduction

This section describes and discusses the architecture of Semantic Overlay Networks (SONs). SONs “are a flexible network organisation that improves query performance while maintaining a high degree of node autonomy” [HeAr03, p.1]. Although SONs cannot be treated as another User Profiling algorithm, they are used in order to perform a very efficient search within a p2p network topology. In the following a model is briefly described where nodes are organised into groups (SONs) according to their content. Afterwards a working example of this mode is lighted up by referring to the experimental results described in [HeAr03]. These results show a significant improvement regarding search performance within a p2p network.

5.4.2 Semantic Overlay Networks (SON)

As mentioned before, a Semantic Overlay Network is a network organisation where nodes that have similar content are grouped together. Each node can belong to one or more SONs depending on its content. After a query is made, it is analysed to identify the SON which best matches the requested query. Afterwards, the query is sent to this particular SON only. In this manner, the communication costs decrease significantly compared to traditional flooding techniques where ever single node / neighbour receives every query. Moreover, more system resources are free and the overall performance of the network will probably increase.

One of the main challenges to cope with is to choose the most efficient size for each SON. If the granularity of the SON is not high enough, it won’t generate enough representative localities whereas if the granularity is too large it would increase the maintenance costs [HeAr03]. This represents the classical problem of identifying the trade-off between network performance and accuracy of the query results.

A possible approach for User Profiling might be the construction of a node classification algorithm which, as soon as a request appears, selects a small number of overlay networks that have nodes with a “high” number of hits. In that way the query will be answered quickly and the nodes that are not relevant won’t be occupied freeing system resources. [HeAr03] suggests the main steps of this algorithm to be as follows:

- Using the data contained in each node, a node hierarchy is created
- The hierarchy is stored in every node of the network and is used to define the SONs

- If a new node joins the system it requests information regarding the SONs that currently exist (can be done by using a flooding technique)
- According to the nature of the data it contains, the node joins one or more SONs with relevant context

In order to study the impact of hierarchies the following experiment was conducted by [HeAr03]. Music files were firstly divided according to their style (Rock, Jazz, Classic, etc.) and then to their substyle (Soft Rock, Alternative Rock, Hard Rock, etc.). 26 categories were defined for a music style whereas 255 categories were identified to describe the substyle of each style. The second hierarchy of music files was based on the decade they were published and finally a third hierarchy divided music files based on their "tone" (energetic, soft, party, etc.).

The experiments conducted proved that a combined style and substyle hierarchy performs better than the other alternatives. With this method each node has to join a small number of different SONs and each SON contains a small number of nodes. The "decades" hierarchy was rejected since more than half of the SONs were found to have more than 600 nodes. Finally it was found that using a hierarchy based on the tone of each song results in a small number of SONs but the nodes maintain large number of connections. It is therefore clear that the style/substyle hierarchy produces the best results and is the one that was chosen for the rest of the experiments.

Further experiments on hierarchies were carried out. Layered SONs, a non-conservative assignment strategy where nodes enter only some of all possible SONs have also been introduced. It is not the intention of this paper to further expand on the specifics of this experiment. Nevertheless, it is obvious that SONs offer significant improvements in the query performance area as well as in the maximum achievable recall level when compared with random overlay networks. SONs are therefore suitable for current and future P2P systems where, by nature, the available data is clustered. For further reading as well as for the extensive experimental results regarding SONs the research paper [HeAr03] is recommended.

6 DISCUSSION OF THE PROPOSED ALGORITHM-BASED MECHANISMS

In this section, the previously described algorithm-based User Profiling mechanisms will be discussed. Having always in mind the DBE's special features and requirements each algorithm's strengths and weaknesses are highlighted. Log File Analysis, Collaborative Filtering, Distributed Collaborative Filtering and Ontologies will be examined under a more general viewpoint. Due to limited resources a fundamental research and theoretical evaluation of an adequate algorithm was performed to build the basis for a future implementation of this specific algorithm. The extensive User Profile development for the different applications and integration effort with other DBE components is still in progress and provides functionalities for the user to explicitly define their preferences. The implicit information retrieval which could be realised by the proposed UP algorithm will be out of scope of this work package but are a recommended issue for further research in other projects dealing with the ideas of the DBE. Results of the theoretical evaluation provide objective advantages and disadvantages of each algorithm by drawing conclusions from experiments carried out by other scientists in different projects taking the complex environmental network of the DBE into account. At this point, the Distributed Collaborative Filtering algorithm seems to be the best solution for Recommendation purposes. The fact that this is the only fully distributed algorithm is a precious advantage. Moreover, as experimental data suggest, distributed CF shows the same level of performance as traditional CF algorithm.

6.1 STANDARD COLLABORATIVE FILTERING ALGORITHMS

One of the key features of the DBE is that despite its decentralised architecture, some peers undertake greater responsibilities than normal peers. These peers are called super-peers and they are the fastest, most reliable peers that offer the highest bandwidth and are known to be trusted. Due to the fact that these peers will probably have greater resources than other peers, they are responsible for running the most critical infrastructural services like the Semantic Registry Name Service and Knowledge Base and Semantic Registry Services [SBMC05].

One possible assumption would be that super-peers could also undertake the responsibility to execute User Profiling algorithms and therefore to produce item and user-based recommendations. In that way, one can make use of super-peers and choose a

standard, non distributed, CF algorithm as the User Profiling algorithm. In this case, each super-peer could store the user-item matrices for a certain number of DBE peers. Accordingly, all the necessary calculations required for the user proximity estimations can be carried out in each super-peer. Since super-peers are known to have plenty of available resources, recommendations could be produced without major system slowdowns. For the second step of the Collaborative Filtering algorithm which is the neighbourhood formation, communication between the super-peers is required. In order to form a neighbourhood of users who share similar interests, each super-peer has to exchange information with the other super-peers regarding user's relevancies. In that way, before forming the neighbourhood, each super-peer will know the relevancies between every peer who is logged on the network. Unfortunately, this information exchange breeds significant communication costs. For a p2p network with small number of participating nodes, these costs can be handled. Nevertheless, as the size of the network increases, slowdowns might be observed due to these communications. This has to be tested in the future and is out of the scope of WP7. As a result, scalability problems could occur. It can therefore be concluded that although Standard Collaborative techniques can be used in DBE with the help of super-peers, this method is not the ideal one since it burdens the system with unnecessary load. Even super-peers do not have limitless capacities.

6.2 DISTRIBUTED COLLABORATIVE FILTERING

As analysed in the previous sections, distributed CF is a variant of the standard CF algorithm that is completely decentralised. No central server is necessary for the production of recommendations. Instead, all the necessary calculations are performed in each peer itself. Each item contains a list called item-buddy table, where the relevancies with all the available items of the network are stored. Based on this list, if one peer requests access for a certain file, it can calculate all the relevant items that match its profile. After that, recommendations based on those relevancies are built. Apart from being a fully distributed algorithm that shows compatibility attempts to the nature of DBE's p2p network, distributed CF algorithm has another important advantage: It produces recommendations of equal quality to the standard CF algorithm. As experimental results have shown, distributed CF approximates performance of the standard CF and in some cases even outperforms it. The distributed CF algorithm as an adopted CF algorithm seems to be a very good choice for the implementation of an algorithm-based User Profiling mechanism in the DBE in the future.

6.3 ONTOLOGIES

In the previous sections the performance of a semantically enhanced Collaborative Filtering Algorithm has been analysed with the conclusion that ontologies could both improve the precision of the produced recommendations and reduce the computational costs of the algorithm significantly. Moreover, as other studies suggest [PrGa99], ontologies can prove to be helpful for the design of accurate and noise-free User Profiles. Fully automatic User Profile creation mechanisms have been proposed that rely on ontologies for the creation of User Profiles. Furthermore, searching within an information system can become more efficient and personalised with the use of ontologies.

The design and implementation of an ontology for the purposes of the DBE project seems to be useful even though the exact nature of the recommendation algorithm that will be chosen for the DBE is not yet determined. A possible combination of this algorithm with ontologies is considered to prove to be beneficial. Improved search accuracy, concise User Profiles that are free from irrelevant information, trustworthy recommendations and improved performance and scalability could be some of the benefits of combining of traditional Collaborative Algorithms with ontologies. Of course, further research is needed in order to define the exact specifications of the ontology that will be used for the purposes of the DBE project and cannot be realised by WP7. The fact that the DBE is implemented on an extended P2P network and the existence of super-peers should also both taken into account.

7 CONCLUSION

The objective of this paper was to present the current implementation status of the assemblage-based User Profiling mechanism and to set the scientific foundations for the selection of the most suitable algorithm-based User Profiling mechanism for the DBE.[8].

The first section described the development results for the two User Profiling Tools which will be integrated into the DBE Studio and the DBE Portal. The User Profile allows the users to express their preferences using attributes of BML models in order to generate complete preference sets. They build a basic input for the Recommender Tool to perform user recommendations. Additionally the profiles contain basic user information about the user themselves like ‘name’, ‘country’, ’contact’ and so on. The user profiles stored as XMI-file representations in the KB can also be used by other components like for example the Evolutionary Environment.

The second section investigated the Distributed Collaborative Filtering algorithm to be the potentially most appropriate mechanism for automated data retrieval for a user profile in the DBE. It will probably cope with the distributed nature of the p2p network and it produces similar results to the most common algorithms that are based on the standard Collaborative Filtering mechanism. Ontologies play an important role within recommendation mechanisms. The fact that they improve the quality of the recommendations and that they are a possible solution to the “cold-start” problem is very important. For the future, it is suggested that the following two areas of interest should be researched further.

- *Ontologies:*
  These appear to be a very useful tool that can prove really helpful for our cause. Nevertheless, further research should be made on the design details of the ontology that could be implemented for the DBE. Moreover, the decentralised nature of DBE should also be taken into consideration in the design phase of the ontology

- *Combining the distributed collaborative filtering and ontologies:*
  The experimental results presented were obtained from the combination of standard, centralised CF algorithm and ontologies. The Distributed variation

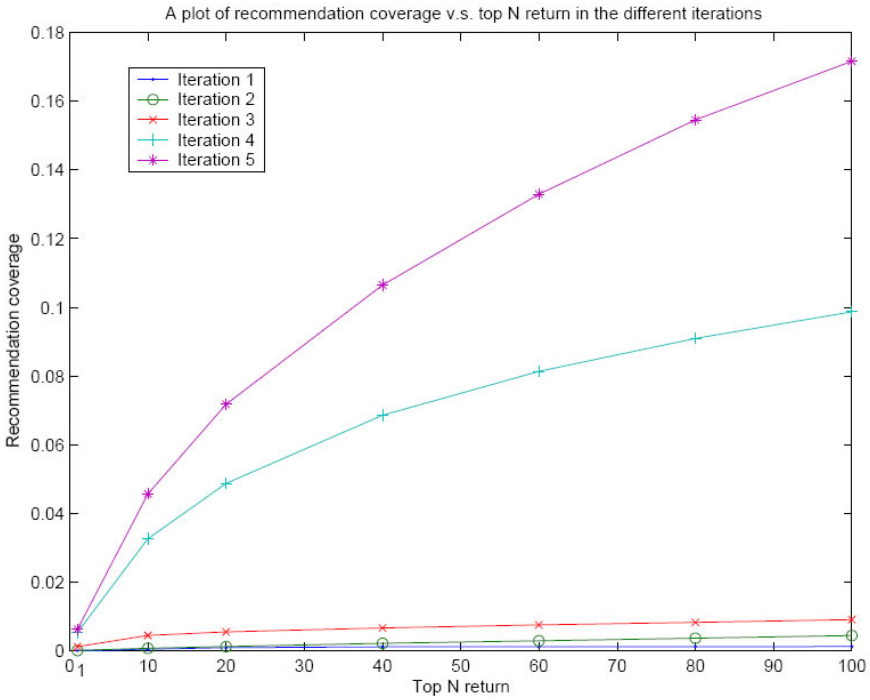
[8] The implementation of the algorithm-based approach to extend the existing of the assemblage-based mechanisms is out of the scope of this Work Package.

of CF has certain requirements that should be further studied for a successful combination with ontologies.

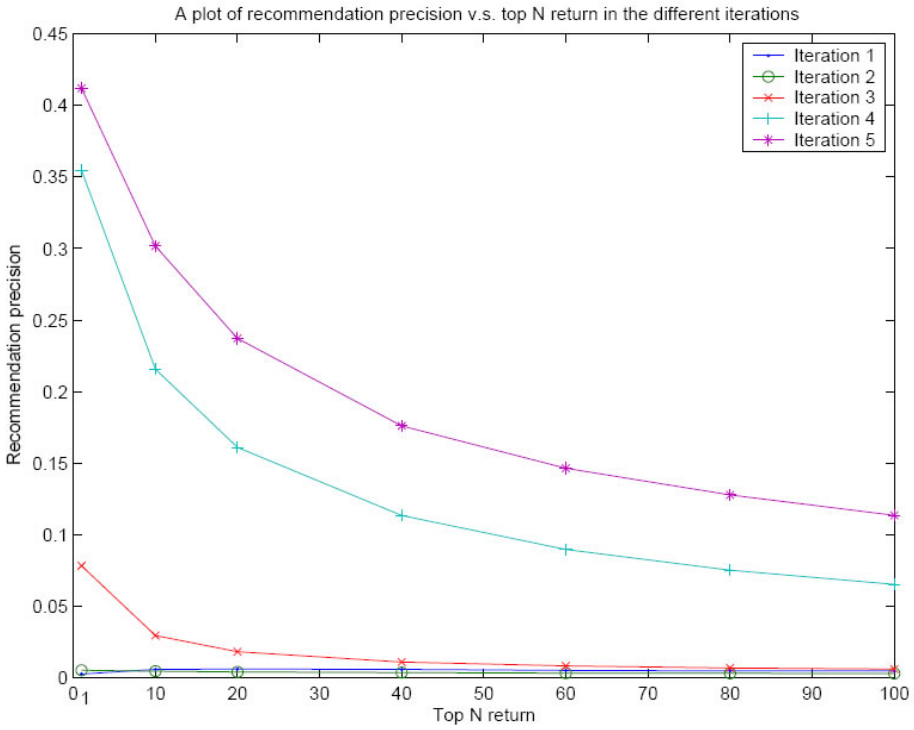
This deliverable described the way the distributed CF algorithm works. A simple model has been proposed according to which the algorithm could be implemented on the DBE. Nevertheless, further research (especially implementation and testing) is needed in this field especially to see how the proposed algorithm can be adapted to the DBE's particular requirements and constraints. The solution could be a combination of item and user-based recommendation techniques to create a hybrid recommendation system that encapsulates the advantages of both CF techniques. The adapted hybrid algorithm might match the DBE's p2p network requirements more efficiently but is out of the implementation scope of this Work Package WP7. In any case, all approaches need sufficient informational input to work properly. The user has to interact with the system as often as possible to generate useful information for the recommendation process.

8 APPENDIX

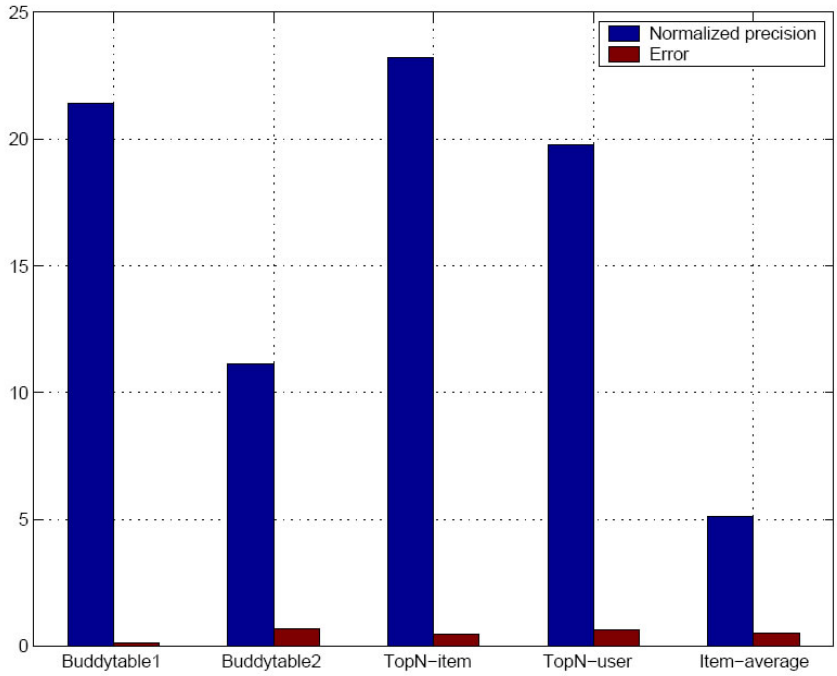
Recommendation accuracy results compared to other CF algorithms
Source: [WRLP05, p.8]



(a) Coverage as a function of the N (top-N) most relevant items after training using the five different iterations.



(b) Similarly for the precision



(c) The normalised precision for:

(1) the proposed distributed collaborative filtering approach with the prior term of equation 15;

(2) without the prior term;

(3) the centralized item-based top-N suggest method;

(4) the centralized user-based top-N suggest method;

(5) a reference recommendation method based on a average ranking

9 REFERENCES

[Bart05a] Bartsch, C.: Description of necessary information about DBE customer, to support a long term evolutionary business relationship. DBE-Project, Deliverable D7.1, 2005.

[Bart05b] Bartsch, C.: Initial Description of Profiling mechanism design and rationale with respect to one or two use cases. DBE-Project, Deliverable D7.2, 2005.

[BaXY04] Bamshad, M.; Xin, J.; Yanzan, Z.: Semantically Enhanced Collaborative Filtering on the Web. Telecommunication and Information Systems, DePaul University, Chicago, Illinois, USA, 2004.

[DaMe06] Dahlem, D.; Meier, R.: Decentralised Identity System. DBE-Project, Report on Task C18, 2006.

[DHNS04] Dolog, P.; Henze, N.; Nejdl, W.; Sintek, M.: Student tracking and personalization: Personalization in distributed e-learning environments. In: Proceedings of the 13th international Worlds Wide Web conference on Alternate track papers & posters, 2004.

[Grub93] Gruber, T.: A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition, 5(2), pp. 199-220, 1993.

[Grub95] Gruber, T.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: International Journal of Human and Computer Studies, 43(5/6), pp. 907-928, 1995.

[Guar98] Guarino, N.: Formal Ontology in Information Systems, In: Proceedings of Formal Ontology in Information Systems (FOIS'98), Trento, pp.3-15, 1998.

[GuKV97] Guo, H.; Kreifelts, T.; Voss, A.: SOaP: Social Filtering through Social Agents. In: Proceedings of the 5th Delos Workshop on Filtering and Collaborative Filtering, Budapest, 1997.

[Halp99] Halpin, T.A.: Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design. San Francisco, California, 1999.

[HeAr03] Hector, G.-C.; Arturo, C.: Semantic Overlay Networks for P2P Systems. Technical Report, Stanford University, 2003.

[Kary01] Karypis, G.: Evaluation of Item-Based Top-N Recommendation Algorithms. In: Proceedings of the tenth international conference on Information and knowledge management, pp. 247-254, 2001.

[KuSh00] Kuflik, T.; Shoval, P.: Generation of User Profiles for Information Filtering-Research Agenda. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 313-315, 2000.

[MASR02] Middleton, S.E.; Alani, H.; Shadbolt, N.R.; Roure, D.C.D.: Exploiting synergy between ontologies and recommender systems. In: The 11th International World Wide Web Conference (WWW2002), Semantic Web Workshop, Hawaii, USA, 2002.

[MaSR03] Maddleton, S.; Shadbolt, N.; De Roure, D.: Capturing interest through inference and visualization: Ontological user profiling in recommender systems. In: Proceedings of the Third International Conference on Knoweledge Capture (Sanibel Island, FL, USA), ACM Press, pp. 62-69, 2003.

[Maed03] Maedche, A.D.: Ontology Learning for the Semantic Web. Norwell, Massachusetts, Kluwer Academic Publishers, 2003.

[Mood01] Moody, T.C.: On the statistical meaning of truncted singular value decomposition. 2001. www4.ncsu.edu/~mtchu/Research/Papers/tsvd.pdf, Recall: 30.06.2006.

[Pret04] Pretorius, A.J.: Ontologies – Introduction and Overview. 2004. http://www.starlab.vub.ac.be/teaching/Ontologies_Intr_Overv.pdf, Recall: 01.07.2006.

[PrGa99] Pretschner, A.; Gauch, S.: Ontology Based Personalized Search. In: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence, pp. 391-398, 1999.

[RBMO04] Rousseau, B.; Browne, P.; Malone, P.; ÒFoghlù, M.: User Profiling for Content Personalisation in Information Retrieval. 19e Congres ACM en Informatique Appliquee, ACM SAC 2004, Nicosia, Cyprus, 2004.

[RuLe02] Runinger, M.; Lee, J.: Ontology applications and design. In: Communications of the ACM, Vol. 45(2), pp. 39-41, 2002.

[SBMC05] Sacha, J.; Biskupski, B.; Meier, R.; Cunningham, R.: DBE Peer-to-Peer Architecture Design. DBE-Project, Deliverable D24.3, 2005.

[ScAm00] Schiaffino, S.N.; Amandi, A.: User profiling with Case-Based reasoning and Bayesian Networks. In: Open discussion track proceedings – International Joint Conference IBEREMIA-SBIA 2000, Atibaia, Brazil, pp. 12-21, 2000.

[ScWL06] Scharl, A.; Weichselnbraun, A.; Liu, W.: An Ontology-based Architecture for Tracking Information across Interactive Electronic Environments. In: Proceedings of the 39th Hawaii International Conference on System Sciences (HICSS-39), 2006.

[SiYV01] Singh, M.P.; Yu, B.; Venkatraman, M.: Community-Based Service Location. In: Communications of the ACM, Vol. 44(4), pp. 49-54, 2001.

[SKKR00] Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J.: Analysis of Recommendation Algorithms for E-Commerce. In: Proceedings of the 2nd ACM conference on Electronic commerce, pp. 158-167, 2000.

[WRLP05] Wang, J.; Reinders, M.J.T.; Lagendijk, R.L.; Pouwelse, J.: Distributed collaborative filtering for peer-to-peer file sharing systems. In: Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC06), 2006.

[YaLi99] Yang, Y.; Liu, X.: A Re-examination of text Categorization Methods. In: Proceedings of ACM SIGIR'99 conference, pp. 42-49, 1999.