Recommending Multimedia Web Services in a Multi-Device Environment

D. Rosaci and G.M.L. Sarné

DIMET, Università "Mediterranea" di Reggio Calabria
Via Graziella Loc. Feo di Vito, 89060 Reggio Calabria (Italy)
E-mail: {domenico.rosaci,sarne}@unirc.it

NOTICE: this is the postprint version of a work that was accepted for publication in Information Systems. The editorial version was subsequently published in Information Systems 38 (2) , pp. 198-212
DOI: 10.1016/j.is.2012.08.002

Abstract

In the last years, the Web community has shown a broad interest in Web services that handle multimedia contents. To improve the usability of these services different tools have been proposed in the literature, and in this context agent-based recommender systems appear to be a promising solution. However, the recommender systems presented in the past do not take into account, in their recommendation algorithms, the effect of the device exploited by the user, while it is clear that the same user shows a different behaviour in presence of different devices. This paper tries to give a contribution in this setting, in order to match more accurately user preferences and interests. In particular, a new agent-based system is proposed, whose architecture allows to compute recommendations of multimedia Web services, considering the effect of the currently exploited device. Some experimental results confirm the high quality of the recommendations generated by the proposed approach.

1 Introduction

For the W3C [57] a Web service (WS) is "a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A WS supports direct interactions with other software agents by using XML-based messages exchanged via Internet-based protocols". In other words, WSs realize a service-oriented architecture designed

to support interoperable machine-to-machine interactions in order to: *(i)* provide some functionalities over a network; *(ii)* accept one or more requests from different clients and send them one or more responses by using well defined interfaces described in a machine-processable format; *(iii)* assure the WS interoperability, without specifying how WSs are combined or implemented.

1.1 The problem: Automatically discovering multimedia Web services

In this context, nowadays we are assisting to a broad interest in Web services that handle multimedia resources and more specifically in multimedia Web services (MWSs), i.e. WSs that involve transportation of multimedia contents over the Web and management of composite devices [65]. In the recent past, researches on MWSs were mainly focused on compressing, caching and streaming multimedia data; currently, the trend is to integrate multimedia services with the most recent WS technologies in order to provide an easier discovery of WSs. In fact, the effectiveness of all WS-based systems strictly depends on the facility for the user to discovery the most interesting services.

Traditional discovery techniques range from manual (email, Web browsing, phone calls, etc.) to automatic (WSIL, UDDI, UWSD) [23]. However, these tools generally suffer in performance and automation level both for the intrinsic difficulty to find the most suitable WS for a user starting by very small text fragments, and for the necessity for the user to manually specify his request. Therefore, the automation introduced by the above tools only consists in discovering the services that satisfy the user request. Instead, it is emerged as a key issue in the Web community the need to provide the user with opportune recommendations in a really automatic fashion; this is even more important if WSs involve multimedia contents. In this context, Web recommender systems appear as a promising solution to satisfy such a necessity. Web Recommender Systems support Web activities of a user providing him with useful suggestions about objects, products, or services which he might be interested in [19, 21, 20]. Usually, recommender systems are intelligent applications which help the user to identify the Web contents which could meet his interests and preferences. A large number of different recommender systems [47, 48] have been proposed in the last years to support users' Web navigation. Generally, recommender systems are partitioned in three main categories [3], namely: (*i*) *Content-based* (CB), that suggest to a user the items which appear the most similar to those he has already accessed in the past; (*ii*) *Collaborative Filtering* (CF), that suggest to a user items which have been also considered by similar users; (*iii*) *Hybrid*, that exploit both content-based and collaborative filtering techniques to generate recommendations (e.g., a Web site can generate suggestions by considering both user personal interests and user commonalities among other known users). In these situations, hybrid recommender systems have been usually recognized as an effective solution [3].

1.2 A promising solution: Using agent-based recommender systems

A possibility to implement recommender systems that deal with different Web sites is represented by the Multi-Agent Systems (MASs). The main component of a MAS is the *software information agent*, which is an application capable to autonomously and proactively perform some tasks on the behalf of its human user. For example, a user can exploit during his navigation an agent as a client which observes his behaviour and in this way implicitly builds a model to represent his interests and preferences with respect to all the visited Web sites. For this purpose, agent-based systems exploit in their recommendation algorithms an internal representation (profile) of the user built by the associated software agent which monitors his Web activities [27, 58, 44, 8, 2, 7, 39, 41]. If the user accesses a Web site, his agent can exploit the profile in the interaction with the site in order to provide both content-based and collaborative recommendations to the user agent by adapting the site presentation. Now, is it possible to conceive agent-based recommender systems in the context of MWSs? In our opinion, an important problem arises when we try to answer such a question. In fact, nowadays users access MWSs by using different devices as desktop PCs, cellular phones, palmtops, etc. Each of these devices presents: (*i*) its own interface characteristics (e.g., display capability), (*ii*) a different Internet connection cost, (*iii*) different storage space and computational capability. These differences can influence both the user behaviour and his preferences; for example, when he accesses an MWS with a cellular phone, he could desire to download a light content. Consequently, a different profile should be built for each device exploited by the same user. This leads us to argue that a user should be provided with a different profile for each device he exploits and the recommendations generated by the system should consider also the device currently exploited by the user when he handles MWSs.

1.3 Our contribution: MWSuggest

In such a context, in order to show the effectiveness for a recommender system to consider the device in the generation of its suggestions, we have implemented a multi-agent recommender system called *Multimedia Web Services-Suggest* (MWSuggest).

An important idea underlying our approach is described in [43], where we proposed an architectural framework, called MUADDIB, to generate efficient recommendations in a multi-device context, based on the pre-computation of the recommendations. The theoretical analysis provided in that work has shown that a multi-agent architecture can introduce significant advantages in terms of time computational complexity, with respect to traditional, centralized recommender architectures. Some practical applications of this multi-agent approach have been proposed in e-Commerce context [40, 42], showing a quantification of the introduced advantage in presence of relevant number of users. However, while the possibility to apply MUADDIB to an e-Commerce scenario does not

imply significative modifications of the original framework proposed in [43], the application of MUADDIB principles to generate multimedia Web services recommendations require much more effort.

In MWSuggest each user is monitored by a software agent, provided with a personal profile, that supports his activity about MWSs. A user that accesses MWSs by using different devices should be characterized by a global profile that takes into account his behaviour when using all these devices. However, it is not suitable to construct such a global profile based on a software running on the exploited device, since this device (e.g., a cellular phone or a palmtop) may have limited resources. Therefore, we choose to provide each device with a device profile that autonomously collects information about the user behaviour associated with just that device. On the other hand, in our system the information are collected in a global user profile, stored on a server machine that is assumed to have more relevant resources than the client. These profiles are exploited in a multi-agent platform to support a user in the selection of the most suitable MWS by generating personalized suggestions that consider also the device currently exploited by him. Three types of agents are defined in the MWSuggest platform and below we describe in detail each of these types. As shown in Figure 1, each user device is associated with a *device agent* that monitors the user in order to build a device profile containing information about the services selected by the user only exploiting just that device. Then, a *profile agent*, unique for each user and running on a server machine, collects the local profiles provided by all the different user device agents in order to construct a complete user profile. Finally, a *recommender agent*, that runs on a server machine, collects the complete profiles built by the profile agents to provide users with useful suggestions about the most suitable MWSs.

The MWSuggest approach is significantly different from MUADDIB. While in MUADDIB the recommendations are generated via an interaction between an agent associated to the Web site visited by the user and the recommender agent, in MWSuggest the recommendation activity is executed during an interaction between the device agent and the recommender, and it is not present a site agent. This modification of the original MUADDIB schema is essential for generating effective multimedia Web services recommendations, since in this context there is not the necessity of managing the user exploration of a Web site by using a site agent, differently from the situations for which MUADDIB has been conceived. This feature allows MWSuggest to generate recommendations for a user by a direct interaction between the device agent and the recommender agent, avoiding the intermediation of the site agent present in MUADDIB. A second, important difference between MUADDIB and MWSuggest is represented by the presence, in MWSuggest, of a unique recommender agent that generates the recommendation, while in MUADDIB there are several recommenders, each of them associated with a different *cluster* of users. The presence of multiple recommenders in MUADDIB aimed at strongly decentralizing the collaborative filtering recommendation activity; however, all the experimental tests we have performed in the last years evidenced the small advantage, in terms of efficiency, introduced by the multi-recommender level, while the cost of the clustering has a

considerable impact on the system performance. Moreover, since in MWSuggest we have introduced a direct interaction between the device agent and the recommender, a solution with multiple recommenders is not practicable, since it would imply a large communication overhead. A further difference between MWSuggest and MUADDIB is in the recommendation algorithm, that in MUADDIB is based on the content of the Web sites, while in MWSuggest is only based on the user profiles. With respect to the e-Commerce applications of MUADDIB, presented in [40, 42], besides the architectural differences discussed above, there exists also a difference in the structure of the device profile and the user profile. While in [40, 42] these profiles are mainly based on information related to the products present in the e-stores, in MWSuggest we include the notions of *service* and *service evaluation* as key concepts to recommend multimedia Web service having high quality. The concept of *service rate*, that we introduce in MWSuggest and that was not present neither in MUADDIB or in its e-Commerce applications, has been taken into account in the recommendation algorithm we propose in this paper.

It is important to highlight that our architecture, although theoretically applicable to whatever type of Web service content, has been appositely designed for Web services that provide multimedia content. In fact, the proposed solution aims at minimizing the impact of the recommendation activity on the Web server, to avoid additional costs for a component that has to execute the onerous task of providing multimedia content. For an analogous reason, we have also designed a recommender level having a unique recommender, to reduce the communication cost of the recommendation activity on the device agent, already engaged in managing a multimedia streaming. The general architecture of MUADDIB or those proposed in [40, 42] are not suitable to this particular situation. We have performed some experiments on real users to evaluate the performance of the proposed system, and the results show a significant improvement of the quality of the produced recommendations with respect to the case of applying the traditional approach that does not take into account the effect of the exploited device.

The rest of the paper is organized as follows. Section 2 presents some related work. Section 3 describes the MWSuggest framework, while Section 4 deals with the architecture and the implementation details. Section 5 provides an experimental evaluation of the approach and, finally, in Section 6 we draw our conclusions.

2 Related Work

In the last years the opportunities provided by the Internet in terms of objects and services offered is exponentially increased together with the number of their potential users. In this context, an emerging issue is represented to get over the intrinsic limits of the existing tools (e.g., WSDL [62] and UDDI [55]) to provide users with an effective and personalized discovery service. For this purpose, several systems and tools based on different methodologies have been developed

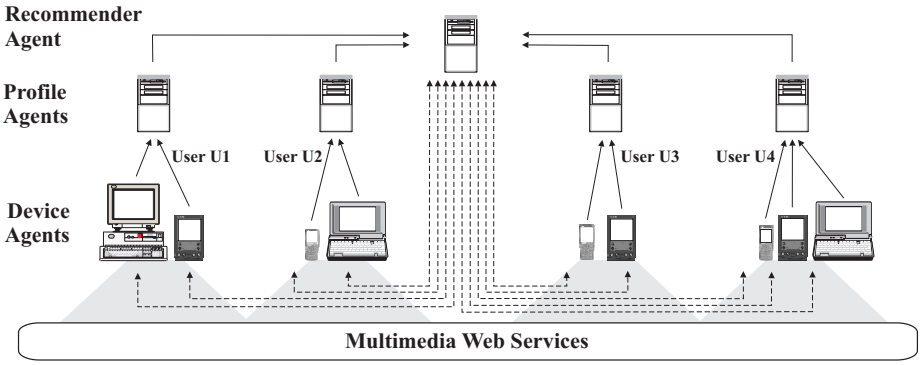

Figure 1: The MWSuggest Architecture

[12]. However, to the best of our knowledge, all the past proposals do not take into account the capabilities of the device currently exploited from the user. In our opinion, device capabilities can heavily influence both the behaviour of the user when he chooses an MWS and the consequent advantage obtainable from the MWS. In this section we analyze some work that can be considered, in our opinion, as effective approaches to support users in the choice of the best available MWSs.

2.1 A rough system classification

Most part of the existing approaches try to improve the performance of the crucial WS discovery phase with the aim to increase the automation level and/or the user satisfaction level. To achieve this goal, different strategies can be followed but they can be synthetically partitioned in two classes namely: *Ontology-based approaches* that enrich the expressiveness and avoid the ambiguity of the usual WSs textual descriptions. These approaches need manual users' interactions and the improvement of the performance is only due to more precise key-searches. In this case, it is required that all the actors share the same informative bases (i.e., ontologies, dictionaries, etc.), usually relative to a limited number of specific topics in order to reduce their size and complexity. *Recommendation-based approaches*, that provide users with suggestions (e.g. content-based and collaborative filtering- based) that could meet their interests and preferences based on suitable profiles built implicitly (by monitoring users' behavious) and/or explicitly (by rating WSs on the basis of different parameters and criteria). In this case, the focus is mainly on the capability of the system to realize an higher user satisfaction, but the price to pay is of having systems considerably more complex than those of the previous class. Below we provide, for each of the aforementioned classes, some short descriptions of well-known approaches proposed in the past.

2.2 Ontology-based approaches

Without any doubt, in order to improve the discovery process, a diffuse approach trend in the MWS world is that of providing MWSs with rich formal descriptions of their competencies by using *semantic languages* as DAML [5], DAML+OIL [6], DAML-S [4], ebXML [9], OWL-S [33], WSMF [10], XML etc. In such a way, the MWSs discovery can be realized as a matching between declarative descriptions of the MWSs sought and the semantic description of the MWSs offered. Moreover, the adding of semantic descriptions to MWSs simplifies also to composite services. Semantic MWSs are dealt within the literature in a really greatest number of work [17, 28, 11, 35, 24]. In some interesting cases [30, 34, 31] semantic annotations of WSs are exploited in combination with agent technology, where agents are delegated to realize the publishing and discovery services. Extending UDDI functionalities represent another significative approach to obtain more performing discovery services. For example, UDDI+ [36] is one of such extensions and in particular it deals with semantic and context features (types, capabilities and models) in MWS discovery in order to support also inexact service matching. A further UDDI improvement is E-UDDI [32]. It has been conceived to solve impediments in service discovery represented by different heterogeneities as platform, data formats, domain, conceptions in handling processes and tasks and so on. For this aim E-UDDI introduces the blue pages for including semantic description of WSs within the UDDI service. When a service discovery has to act in a peer-to-peer ($P2P$) environment, some well known elements absent in traditional scenarios are introduced. A simple translation of semantic enriching of MWSs in a P2P context is realized by [50] that exploits an ontology in a simple way to publish and realize an easier MWSs search in a generic P2P network. Another P2P network that adds semantic description to MWSs is Speed-R [52]. Speed-R assumes that some P2P nodes of the network, organized for domain of interest, act as registers and service discovery for all the other peers. Note that for each domain of interest can be used a different and specific ontology. The solution proposed by the authors of [22] combines a traditional key-word matching with a distributed P2P storage network in which MWSs are mapped by using XML and exploiting a hashing mechanism. Based on the Chord P2P architecture [54], which finds an increasing favor with P2P networks, it is described the P2P-based Web Service Discovery ($PWSD$) [22] that is formed by Service Peers, that run on logic machines (note that more service peers could run on the same hardware). The Service Peers explore the P2P network to search MWSs based on a service description. MWS descriptions and queries are directly hashed and routed in the Chord network.

2.3 Recommender-based approaches

Recommender Systems (RSs) are tools conceived to process large amounts of data in a semi-automated way and to represent in a suitable manner the interests and the preferences of a user [47, 48, 60, 3, 29, 26, 61]. RSs are useful to integrate the UDDI service with their suggestions.

In the last years, the Netflix contest has boosted the research on recommender systems [1], mainly due to the one million dollars prize and the challenging size of the dataset. The Netflix Prize is a collaborative filtering problem whose dataset is much larger than the previously known benchmark sets and thus traditional methods are stressed to their limits.

An introduction of Recommender system is provided in [19]. A complete review of the key advances in collaborative filtering recommender systems is proposed in [21], focusing on the evolution from research concentrated purely on algorithms to research involving questions around the user experience with the recommender. The authors argue that evaluating the user experience of a recommender requires a broader set of measures than have been commonly used, and suggest additional measures that have proven effective. They also identify the most important open research problems, and outline key challenges slowing the advance of the state of the art.

[20] shows how recommenders can be extended to more effectively address information-seeking tasks by expanding the focus from accurate prediction of user preferences to identifying a useful set of items to recommend in response to the user's specific information need.

The authors of [66] proposes to use taxonomic background knowledge for the computation of personalized recommendations, exploiting relationships between super-concepts and sub-concepts during profile generation. Besides addressing the sparsity issue, they use parts of the proposed taxonomy-based recommender framework for balancing and diversifying personalized recommendation lists in order to reflect the user's complete spectrum of interests.

A hybrid recommendation technique, implementing both CB and CF approaches (this latter compliant with [53]), is proposed in WSMX [16] by exploiting the WSMO ontology [38]. The generated suggestions are available for users with both an automated and a manual mechanism of MWS selection. Another hybrid recommender system is that presented in [45], where the authors propose a new multi-agent system, called ARSEC, where each device exploited by a customer is associated with a device agent that autonomously monitors his behaviour. Each customer is also associated with a customer agent that collects in a global profile the information provided by his device agents, while each e-Commerce Web site is associated with a seller agent. Based on the similarity existing among the global profiles, the customers are partitioned in clusters, each one managed by a counsellor agent. Recommendations are generated in ARSEC due to the collaboration between the seller agent and some counsellor agents associated with the customer. DUINE [56] is a prediction framework adopting a hybrid approach. More precisely, it implements a switching method to choose the prediction technique, among those available, able to provide more accurate recommendations. Unlike other hybrid recommender systems developed for specific domain, DUINE is domain-independent. Tests referred to different applications (a TV program guide and a movie recommender system) demonstrate in these cases the effectiveness of the multi-prediction approach adopted by DUINE with respect to a single-prediction strategy.

An *implicit culture* approach is adopted in [18], where a RS facilities the

discovery of MWSs that match user needs by using the history of user-system interactions and client-service communication logs. In [25], it is described a dynamic WS selection system that combines a RS with a semantic matching to compose software services. The system uses a CF approach for the RS part together with users' feedbacks; in such a way a user is helped to select the best service matching his request (based on a semantic service profile and Quality of Service (QoS) parameters) from a set of similar services. The W3C QoS Requirements for WSs [37] gives an overview of the QoS requirements. To estimate the QoS of an MWS, some systems use a reputation criterion for weighing QoS evaluations provided by users. [27] proposes an agent-based architecture based on a reputation model to choose the MWS having the best QoS valuation (considering attributes such as price, on-time delivery and so on). This is realized by the Agency that for each MWS selected by an agent has to aggregate its rate. Each agent rate will be suitably weighted by means of the agent reputation. Another multi-agent framework for QoS-based MWS selection is proposed in [58], where a new distributed reputation-based assessment algorithm supporting QoS is exploited. In [51], a fuzzy inference process is presented, which detects and eliminates invalid rating provided by users and identifies users' preferences for preventing collusion and deception. Furthermore, combining users' reputation and inferring their preferences, it simplifies the task to obtain personalized evaluation in the MWS choice.

Information retrieval is used in [13] to discovery and propose MWS operations. In particular, it uses a new schema matching algorithm based on the tree edit distance and some other specific algorithms for measuring and grouping similar WSs. A two-level method based on information retrieval and structure matching is described in [59]. The first level starts by a partial specification of the desiderata service and searches all the textual elements of the services having a similar descriptions. After that correspondences have been found, the services are ordered based on their corresponding level with the user requirement. Then, the second level provides with a structure-matching method to refine the first step. However, we remark the difficulties in an MWS context to produce significant performance for the common Web search engines by using, as usual, a limited number of keywords [63]. A different strategy is exploited in [46] to tackle the inadequacy of keyword-base MWS discovery by means of a vectorial representation. Indeed, the MWS description is considered as a vector in the vector space spanned by all the terms used in all WSs descriptions.

2.4 Similarity and Differences with MWSuggest

The system MWSuggest that we describe in this paper can be considered, with respect to the classification introduced above, as a hybrid approach. Indeed, on the one hand, similarly to the ontology-based systems MWSuggest exploits a rich knowledge representation model in order to characterize the users' profiles as well as to accurately describe the multimedia resource. However, differently from the ontology-based system, MWSuggest exploits recommendation techniques to provide suggestions for the users that are searching for Web services,

and this feature makes MWSuggest more effective in supporting users than the ontology-based approaches. On the other hand, MWSuggest differs from the recommender-based systems described above since it takes into account, in the generation of recommendations for a given user, the characteristics of the particular device exploited by that user. This feature improves the quality of the generated recommendations with respect to the recommender-based systems described above.

3 The MWSuggest Framework

In this section we describe the MWSuggest framework. We highlight that the technical contribution of our proposal is composed of three issues, namely (*i*) a new notion of user profile, that is differentiated into device user profile and global user profile, and that also introduces the important notions of *concept rate* and *service rate*. We clarify that this innovation allows to take into account the particular device the user is exploiting when he requests a recommendation; (*ii*) a new recommender system architecture, appositely designed to make efficient the generation of multimedia Web services recommendation; (*iii*) a new recommendation algorithm, that provides effective recommendations, based on the usage of the user profiles introduced in (*i*) For each of the issues above, we have introduced below a dedicated sub-section.

3.1 Knowledge Representation: Dictionary and User Profile

In order to generate suitable recommendations taking into account also the device currently used, we have conceived a recommender system, called MWSuggest, to exploit personal agent profiles associated with the different devices of each user, where each profile stores information based on the Multimedia Web Services (MWSs) selected by a user exploiting just that device. Furthermore, to univocally identify in the MWSuggest platform both service categories and users' interests, a publicly available *common dictionary* of categories is shared among all the agents that consequently share the same concepts (in our application a concept can simply be a term that abstractly describes a collection of multimedia objects, as a *map* or a personalized audio *message*, and in its turn a concept instance is an MWS belonging to a specific concept). An MWSuggest *user profile* synthetically describes the interests of a given user. It consists of two different components, called *Catalogue of Services* and *Catalogue of Concepts*, respectively. The *Catalogue of Services* (CS) describes the user interest, for the different MWSs selected by him in the past taking into account for each service accessed, how much the user is interested in it and how "old" is the user interest. Formally, CS is a set of l tuples $\langle s, c, sr, ssd \rangle$, where l is the number of selected services, s is an MWS, c is the concept of the common dictionary which s belongs to, sr measures the interest of the user in s and ssd is the date of the last selection of s. Analogously, the *Catalogue of Concepts* (CC) describes

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="conceptType">
      <xs:simpleType>
        <xs:restrictionBase="string">
          <xs:enumeration value= "c1"/ >
          <xs:enumeration value= "c2"/ >
          ...
          <xs:enumeration value= "cn"/ >
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:schema>
```

Figure 2: An XML Schema implementation of a concept type

the user interest in the different concepts belonging to the common dictionary. Obviously, the interest in a concept c derives from the interest shown by the user in those MWSs belonging to c, therefore CC represents, for each concept accessed in the past, how much the user is interested in that service, how "old" is the user interest and the list of all the MWSs belonging to c accessed by the user. More formally, CC is a set of m tuples $\langle c, cr, csd, Ls_c \rangle$, where m is the number of concepts contained in the dictionary, c is a concept of the dictionary, cr measures the user interest in c, csd is the last date on which the user selected c and Ls_c is the list of the selected MWSs associated with c. In the current implementation of MWSuggest, the common dictionary is realized as an XML (eXtensible Markup Language) Schema [64] document, where each XML element [57] represents a *concept* and each MWS is an *instance* of a concept. When an instance of a concept is selected by the user, we assume that the associated concept is selected. This implementation of an MWS allows the client agent to automatically detect the concept to which the service points, and therefore understand the interests of the agent. This is obtained in the XML schema by using a *restriction* construct. If the XML dictionary contains n different concepts, say $c1$, $c2$,..,cn, then the element *conceptType* is expressed as in Figure 2, stating that an instance of *conceptType* is an instance of $c1$, or an instance of $c2$,.., or an instance of cn.

3.1.1 Device Agent and Device User Profile

In the MWSuggest framework, each device d of each user u is provided with a software agent, called *device agent*, that manages and updates a device user profile DUP^d having the structure described above. The device agent updates its catalogue of service CS^d of the user profile DUP^d as follows. When the user u selects a service s associated with a concept c, then the u's device agent requests u to provide an evaluation of s, expressing a rate r belonging to the interval [0,1]. Then, if the service s has been selected for the first time, a new element CS_s^d is inserted in the catalogue of services CS^d of UP^d, such that the

service rate sr of CS_s^d is set to r and the parameter ssd of CS_s^d is set to the current date. Otherwise, if the service s is already present in CS^d, the device agent updates the service rate as follows:

$$sr = (sr + \alpha r)/2 \tag{1}$$

that is the new value sr computed as a weighted mean between the old value of sr and the current rate r provided by the user, where the weight of r with respect to the old sr is represented by the parameter α, ranging in $[0, 1]$, arbitrarily set by u. In particular, if α is equal to 1, the new sr is simply the average between the old sr and r, while if $\alpha < 1$ then r weights less that the old sr in the computation of the new sr. The parameter ssd is set to the current date. Also in this case the parameter ssd of CS_s^d is set to the current date. Note that different and more complex formulation can be used to compute sr and in the same way the chosen of the parameters on which is computed r (see Section 5) is orthogonal to the issue dealt in this work. Then the device agent updates the concept catalogue CC^d of the user profile as follows. For each concept c, associated with a service s selected by u, the element CC_c^d is computed by assigning to the concept rate cr the average of the sr values of all the services associated with c. Formally, if k is the number of the services associated with c, we have that:

$$cr = \frac{\sum_{i=1}^{k} sr_i}{k} \tag{2}$$

while the last date csd is set to the most recent current date among the k services associated with c. Note that sr (resp. cr) is considered a measure of the user satisfaction about the service (resp. the concept), and is strictly related to the characteristics of the exploited device. These two parameters will be used by the system in order to select the more interesting services and concepts for u. Obviously, when a service or, consequently, the associated concept are never accessed by u they are set to 0. The device agent also stores for each service and concept the last date of their selection. These values are used to periodically decrease the rate values of services and concepts no longer selected by u, based on their temporal distance from the last selection date.

3.1.2 Profile Agent and Global User Profile

The device agent is not the only agent in the proposed MWSuggest framework that constructs and maintains a profile of its user. Indeed, each user u is also provided with a personal agent, called *profile agent*, that runs on a server machine and stores a user profile called *global user profile* GUP. Such a global user profile contains all the concepts and MWSs selected by u, independently from the exploited device. The structure of the profile agent, described in Section 3.2, is composed by a *Catalogue of Services* (CS^p) and a *Catalogue of Concepts* (CC^p). In particular, for each service s of CS^p (resp., each concept c of CC^p) the profile agent computes the *global service rate* gsr (resp., *global concept rate*

gcr) and the associated *global service selection date gssd* (resp. *global concept selection date gcsd*) based on the corresponding values computed for the same service (resp. concept) by all the device agents of u. In particular, if ND is the number of device agents exploited by u, then his profile agent computes, for each service s (resp. concept c) accessed by u, the value gsr (resp. gcr) equal to the weighted sum of all the service rates sr (resp. concept rates cr) computed by the user device agents. In order to weight each service rate sr_d (resp. concept rate cr_d) coming from a device d, it is exploited the *cost* for using the device d, that we assumed to be measured by the *price per kilobyte* relative to the Internet connection of the device. More formally:

$$gsr = \frac{\sum_{d=1}^{ND} \mu_d \times sr_d}{\sum_{d=1}^{ND} sr_d} \tag{3}$$

$$gcr = \frac{\sum_{d=1}^{ND} \mu_d \times cr_d}{\sum_{d=1}^{ND} cr_d} \tag{4}$$

where sr_d (resp. cr_d) is the interest rate computed for the given service s (resp. concept c) by the d-th device, $d = 1, ..ND$, and μ_d is the device cost of the d-th device. In its turn, the global last date of the service s (resp. concept c) $gssd$ (resp. $gcsd$) is computed as the maximum value among the csd_d values associated with s (resp. c) and stored in the different d-th device profile. Device and profile agents will be detailed described in subsections 3.2.1 and 3.2.2.

3.2 The Multi-Agent Architecture

Figure 3 graphically describes how the proposed recommender system works. In particular, we observe that the device, profile and recommender agents collaborate in two different and parallel path. The first path, on the top of Figure 3, is relative to the construction of the user profile and its exploitation in computing similarities between users. In particular, the device agent sends its profile to the profile agent. The profile agent collects all user profiles coming from the device agents in order to construct the global user profile GUP and sends it to the recommender agent. The second path, on the bottom of Figure 3, is relative to the recommendation activities. When the user searches an MWS belonging to a concept c, his device agent interacts with the recommender agent and sends to it some information about the preferences of its user. These preferences are relative to the presentation format desired by the user when exploiting the device d and are stored in the corresponding device profile DP_d, whose structure will be described in Section 3.2.1. The recommender agent pre-computed the services that best match with the device profile of the user, to support content-based recommendations, and those selected by other similar users that have exploited the same device, to support collaborative filtering recommendations. Then, these information are transmitted to the device agent of the user for selecting the most suitable MWSs. In the next subsections we deal in details with each of the three agents that compose the MWSuggest agent architecture.

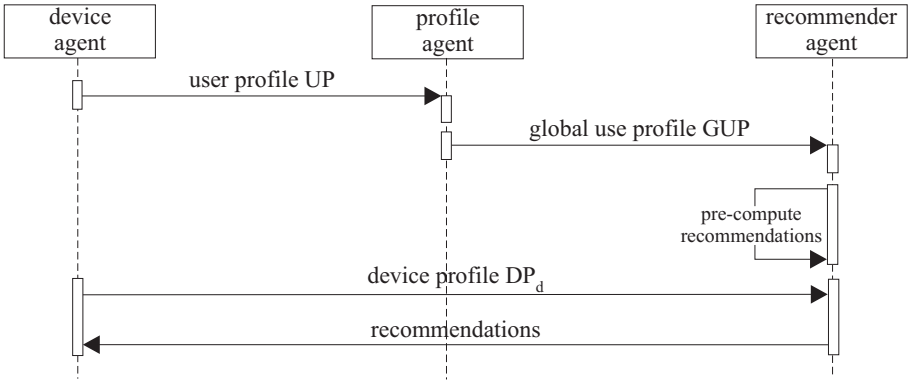

Figure 3: The UML sequence diagram of MWSuggest

3.2.1 The Device Agent, Data Structures and Behaviour

With each device exploited by a user u we associate a device agent that monitors u when he selects an MWS in order to locally update the associated user profile. We describe below both the data structure and the behaviour of the device agent. The device agent associated with a given device d, internally stores two data structures, namely the *Device Profile* (DP_d) and the *Device User Profile* (DUP_d). DP_d contains some parameters that characterize the user that is exploiting the device. These parameters are set by the user and allow him to specify how weighting his different actions in order to compute the interest rate and his preferences about the visualization of multimedia contents, the periodical pruning of his profile and the recommendations to be generated. A complete list of the parameters contained in the device user profile is reported in Section 4. The Device User Profile (DUP_d) of the device agent is conform to the description of the user profile provided in Section 3.1.1.

The device agent, associated with the device d, supports u as follows: (i) In order to construct the device user profile DUP_d, it monitors u, recording those MWSs selected by him. When the device user profile DUP_d is changed and it is older that t_d minutes (a parameter chosen by the user) the device agent sends the DUP_d to its profile agent. Note that if t_d is too large then the generated suggestions could be based on obsolete date; differently if it is too small then the user could bear high cost due to the frequent connections. (ii) When u searches a service belonging to a specific concept of the dictionary, the device agent sends the parameters of the exploited device, contained in the device profile DP_d, to the recommender agent in order to generate personalized suggestions for u. (iii) For taking in account the "age" of the interest values, each P (a parameter fixed by the user) days the device agent updates the sr and cr values (i.e., $sr = \phi(sr, ssd)$ and $cr = \phi(cr, csd)$) associated with each service and concept.

3.2.2 The Profile Agent, Data Structure and Behaviour

Each user u is associated with a user profile agent p that runs on a server machine. The profile agent collects by each device agent of u the information about the MWSs and the associated concepts selected by him in order to build his global user profile. This feature of MWSuggest allows us to leave possible limited computation and storage capability to the devices associated with the device agents. The profile agent provides the user with an off-line collector of all the information obtained by his different device agents which have monitored user activities. Then these information are sent to the recommender agent. Both the data structure and the behaviour of the profile agent are described below. The profile agent internally stores the *Profile Setting* (PS) and the *Global User Profile* (GUP). The parameters contained in the Profile Setting PS are used to compute the global rate of the interest relative to a concept (see below). Such parameters are ND, that is the number of device agents associated with u, and μL, that is a list containing ND elements, where each element μ_i is the cost of the Internet connection of the i-th device, measured in price per Kbyte. The Global User Profile (GUP) of the profile agent is conform to the description of the global user profile provided in Section 3.1.1.

The behaviour of the profile agent consists in updating the global user profile GUP by exploiting the data periodically provided by each user device agent. The profile agent, for each concept in its GUP provides to compute both *gcr* and *gcsd* (see Section 3.1.2).

3.2.3 The Recommender Agent, Data Structure and Behaviour

The last type of MWSuggest agent is the *recommender agent*, able to generate recommendations for each user. We describe in this section both the data structure and the behaviour of such an agent. Note that the behaviour of the recommender represents our recommendation algorithm, described in form of pseudo code in Figures 6 and 7.

The recommender agent contains two data structures called *Global User Profile Set* ($GUPS$) and *Device Profile Set* (DPS). Each element of the Global User Profile Set $GUPS$ is associated with each user of the MWSuggest platform and contains a copy of his GUP, periodically updated by his profile agent. The Device Profile Set DPS contains the device user profiles of all the MWSuggest users. Each element of DPS, denoted by $DPS[u, d]$, represents the device user profile of a given user u associated with his device d. The data structures described above are exploited by the recommender agent to compute, for each user u, two lists, denoted by SCB and SCF, containing content-based recommendations and collaborative-filtering recommendations, respectively, considered to be the most interesting for u and taking into account the device currently exploited by u.

3.3 The Approach to generate recommendations

Now, we explain the approach used to generate recommendations. Suppose that the user u exploits a given device d, when he searches an MWS able to satisfy his need. Then, the device agent associated with d contacts the recommender agent sending a request of recommendations to it. In its turn, the recommender agent returns to the user device agent two lists of MWSs, denoted by SCB and SCF, which store the content-based and the collaborative filtering recommendations, respectively, computed to support u in presence of the device d. For generating content-based recommendations, the recommender agent periodically pre-computes a matrix CB, relative to all its users, by exploiting the information about both users' interests in concepts and users' device profiles, stored in users' data structures ($GUPS$ and DPS). In particular, each element $CB[u, d]$ of the matrix CB is a list of concepts, ordered in a decreasing fashion based on the coefficient gcr, and compatible with the user preferences stored in the device profile set $DPS[u, d]$. When the device agent informs the recommender agent that its user u needs of recommendations in presence of a given device d, the recommender agent builds and returns a list SCB of MWSs. In particular, for each one of the first mCB concepts contained in $CB[u, d]$, the recommender agent selects the first $mICB$ services, based on their sr, chosen in the past by the user exploiting d. Remember that mCB and $mICB$ are two parameters internally stored into the device profile. Analogously, in order to generate collaborative-filtering recommendations, the recommender agent periodically updates the matrix CF. In particular, each element $CF[u, d]$ is associated with a user u and a device d and contains those concepts that the recommender agent suggests to the user u when exploits the device d. These concepts are the result of a collaborative-filtering phase performed by the recommender agent. Indeed, the recommender agent compares the device profile $DPS[u, d]$ of the device profile set, with the device profile $DPS[q, d]$ associated with each other user q, that has exploited the same device d. The result of this comparison is the value $ID(u, q, d)$, a measure of the global *interest difference* (ID) of the two device agents u and q with respect to concepts of their interest (see subsection 4.2 and formula (5) for the details about the computation of $ID(u, q, d)$). The recommender agent orders in a decreasing order both the users, based on the ID. Then the recommender agent inserts in $CF[u, d]$ at most the first mCF concepts, based on the gcr value, for each one of the first mU users most similar, based on the ID values. When the device agent contacts the recommender agent in order to obtain recommendations for a user u that is exploiting a device d, the recommender agent builds and returns the list SCF of MWSs. In particular, for each one of the concept contained in $CF[u, d]$, the recommender agent selects the first $mICF$ services, based on their sr, that in the past have been selected exploiting d. Remember that mU, mCF and $mICF$ are parameters internally stored into the device profile. Finally, both the list SCB and SCF are sent to the user u in a format according with the preferences set in the device profile DP.

An implementation of the recommendation algorithm, as well as the proce-

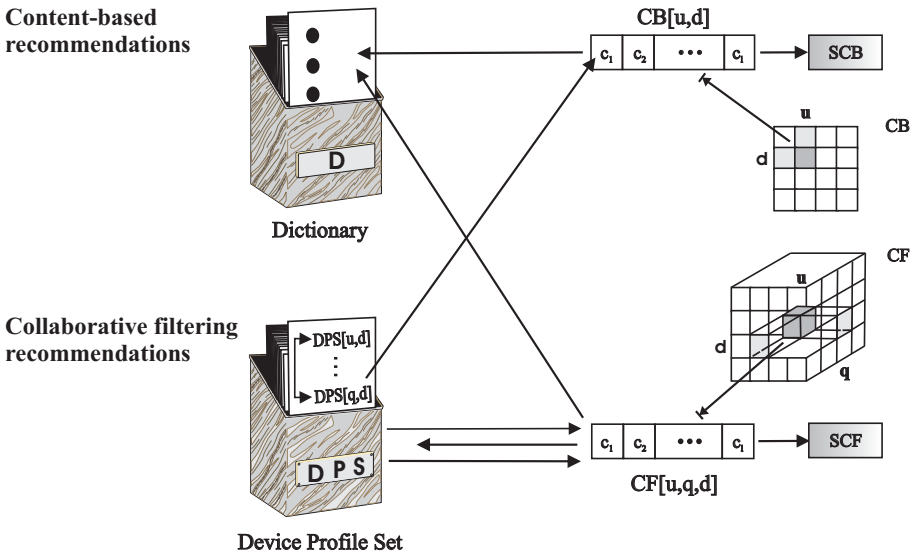

Figure 4: The behaviour of the MWSuggest recommender agent

dure to update CB and CF matrices is reported in Section 4.

4 System Implementation

In this section, we describe both the architecture and the implementation details of MWSuggest. The system has been realized as a Java package of classes, called MWSuggest, containing the three main classes DeviceAgent, ProfileAgent and Recommender Agent, and some auxiliary classes. In Figure 5, the diagram of the main classes is represented. In particular, the class DeviceAgent exploits the classes UserProfile and DeviceProfile to implement the two data structures DUP and DP described in Section 3.1. Also the class ProfileAgent exploits the class UserProfile in order to implement the structure GUP. The class RecommenderAgent, in order to implement the structures UPS and DPS, uses a vector of UserProfile objects and a vector of DeviceProfile objects, respectively, while the two structures SCB and SCF are implemented as a set of Recommendation objects. The class UserProfile uses a vector of Ctuple (resp. Stuple) to implement the data structures CC (resp. SC), where Ctuple (resp. Stuple) represents all the parameters contained in a tuple of the *Catalogue of Concepts* (resp. *Catalogue of Services*). A detailed description of these parameters is reported in the next subsection. The two methods `pre-comp` and `recommend`, characterizing the behaviour of the recommender agent, and performing the pre-computation of the recommendations and the generation of requested recommendations, respectively, are described in Subsection 4.2.

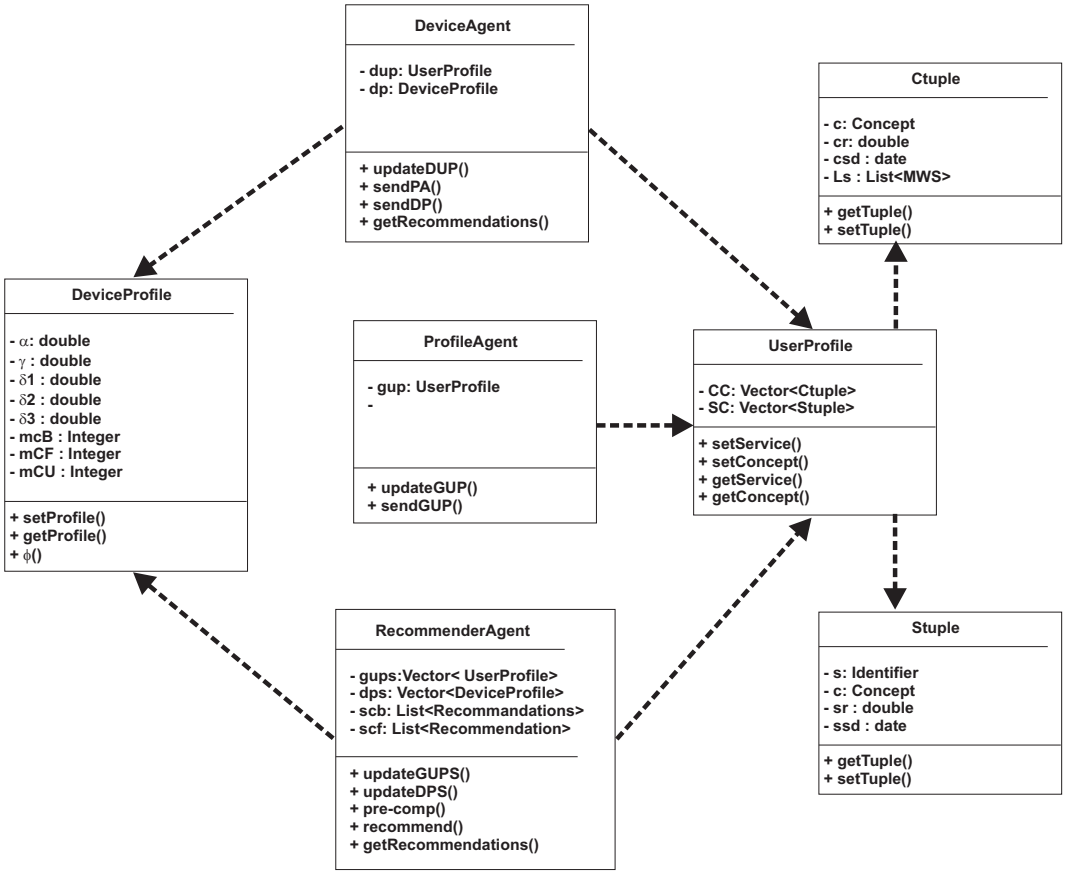

Figure 5: The class diagram of the MWSuggest package

4.1 The Device Agent Parameters

Below we report a complete list of the parameters contained in DP_d, together with their meanings.

- α: it is a parameter (ranging in [0;1]) used in formula 1 to weight the rate provided by u about a selected service;
- γ: it is the *attenuation period* expressed by the number of days between two consecutive selection of a service after which the interest for the service decreases;
- δ_1, δ_2, δ_3: represent the maximum sizes (in Kbyte) of text, audio and video contents that u desires to receive from the selected service when he uses this device;
- ϕ: it is a function used to decrease each γ days the cr and sr values relative to services and concepts no longer selected;
- mCB, mCF, mU, $mICB$ and $mICF$: they are parameters exploited by the device agent in its interaction with the recommender agent (see Section 3.2.3). In particular, mCB, mCF, mU, $mICB$ and $mICF$ represent respectively: (i) the maximum number of concepts that u desires to be considered as content-based recommendations; (ii) the maximum number of concepts that u desires to be considered as collaborative filtering recommendations coming from each other user; (iii) the maximum number of users that u desires to be considered for determining collaborative filtering recommendations; (iv) the number of maximum services associated with the mCB concepts that u desires to be visualized as content-based recommendations; (v) the number of maximum services associated with the mCF concepts that u desires to be visualized as collaborative filtering recommendations.

4.2 An implementation of the recommendation algorithm

Figure 6 shows an implementation, in the form of pseudo-code, of the recommendation algorithm described above that allows the recommendations to be pre-computed (off-line). The implementation is represented by the function `pre-comp`, that is called by the recommender agent to pre-compute and return the two matrixes of concepts CB and CF for each one of its users. This function receives as input the device profile set DPS and the global user profile set $GUPS$. The function `pre-comp` calls for each user the function `CBinsert` that provides to populate the element $CB[u, d]$ with the first mCB concepts, based on the gcr value. Then the interest differences among the users are computed by the function `InterestDiff` and ordered calling the function `sortID` with respect to each user. Finally, the function `CFinsert` provides to populate the element $CF[u, d]$ with the first mCB concepts, as it has been described in the previous subsection. When a device agent requires the help of the recommender

agent, the recommender calls the function `recommend` to build two lists of services (represented in the code by the type SList), namely SCB and SCF. The function `recommend` receives as input the user u, the device profile set DPS, the global user profile set $GUPS$ and the pre-computed matrixes CB and CF and then it calls the functions `SCBinsert` and `SCFinsert` to select the $mICB$ and the $mICF$ services compatible with the device currently exploited by u based on the method described in the previous subsection.

pre-comp(**device profile set** DPS, **global user profile set** $GUPS$, **CMatrix** CB, CF) {
IntD IDmatrix;
for each u
    $CB[u,d]$=`CBinsert`($CB[u,d]$, $DPS[u,d]$, mCB);
    for each $q \neq u$
        $ID[u,q,d]$=`interestDiff`($GUPS$[u], $DPS[u,d]$, $GUPS$[q], $DPS[q,d]$);
`sortID`($IDMatrix$);
for each u
    $CF[u,d]$=`CFinsert`($CF[u,d]$, $DPS[u,d]$, $DPS[q,d]$, mU, mCF);
return;

Figure 6: The Pre-computation Algorithm

recommend(**user** u, **device profile set** DPS, **global user profile set** $GUPS$, **CMatrix** CB, CF **SList** SCB, SCF) {
SCB=`SCBinsert`(CB[u,d], $DPS[u,d]$, $GUPS$[u], $mICB$);
for each $q \in mU$
    SCF=SCFinsert(CF[u,d], $DPS[u,d]$, $DPS[q,d]$, $GUPS$[u], $GUPS$[q], $mICF$);
return;

Figure 7: The Recommendation Algorithm

Computation of the Interest Difference ID The difference between a user u and another user q which uses the same device d is computed as follows. Let c be a concept that belongs both to the device user profile $DPS[u,d]$ of u and the device user profile $DPS[q,d]$ of q, and let $cr_u(c,d)$ be the interest rate assigned to the concept in $DPS[u,d]$ and $cr_q(c,d)$ be the corresponding interest rate in $DPS[q,d]$. The value $diff(c,d) = |cr_u(c,d) - cr_q(c,d)|$ is assumed to be a reasonably measure of the difference between the two users u and q in the evaluation of the concept c. We measure the global interest difference between the two device agents u and q using the device d, denoted by $ID(u,q,d)$, as the average of all the contributions $diff(c,d)$ relative to all the concepts c that the profile of u and q share. More formally:

$$ID(u,q,d) = \frac{\sum_{c \in DPS[u,d] \bigcap DPS[q,d]} |cr_u(c,d) - cr_q(c,d)|}{|DPS[u,d] \bigcap DPS[q,d]|} \quad (5)$$

Updating CB and CF matrices The updating of the CB (resp., CF) matrix is automatically performed when both the two following conditions are verified: *(i)* At least each t_{CB} (resp., t_{CF}) minutes an element of the CB matrix (resp., CF matrix) is updated, where t_{CB} (resp., t_{CF}) is an integer value fixed by the system administrator. *(ii)* When some modification has been applied to either the dictionary or the global user profile of u (resp., the device user profile of a user belonging to the community). Note that a value t_{CB} (resp., t_{CF}) to small could imply for the recommender frequent re-computation that could require a more powerful machine. On the contrary, a t_{CB} (resp., t_{CF}) too large could imply a generation of the recommendations based on obsolete data. However, the administrator has to carefully fix such values taking into account the number and the computational capabilities of the available machine that host the recommender and the number of user belonging to the community. Besides, the update of the matrix CF is a more onerous task than that of the matrix CB, involving all the users of the community. For this reason, it is necessary to fix a value t_{CF} more large than t_{CB}, to avoid too onerous computations.

5 Evaluation

This section presents some experiments devoted to evaluate, in the generation of recommendations, the effectiveness of considering the device currently exploited to support a user in the choice of the most suitable MWSs. We point out that it is possible, in MWSuggest, to disable the feature that takes into account the exploited device. In this case, our system becomes similar to traditional recommender-based approaches, using the traditional content-based and collaborative filtering approaches. This consideration gives the possibility to evaluate the improvements introduced by our mechanism of considering the exploited device, by comparing the performance of MWSuggest in the two cases in which this mechanism is disabled or enabled, respectively.

The experiments have been performed by using a common dictionary, implemented as an XML-Schema, containing 94 different concepts, and 388 different MWSs associated with the concepts belonging to the common dictionary. Note that the MWSs have been built by using multimedia contents publicly available on radio-television, photography, travel, map, literature and similar Web sites.

The experiments involved 60 users and each user has been provided with a set of three device agents (associated with a desktop PC, a palmtop and a cellular, built following the descriptions presented in Section 3.2.1) and a profile agent. In particular, the profile agent has built two Global User Profiles, the first following the description provided in Section 3.2.2 and the other in which the values gsr and gcr has been computed as two simple sums of the different contributes sr and cr provided by the device agents. Consequently, the recommender agent

Table 1: The setting of the MWSuggest device agents

device agent	α	γ	δ_1	δ_2	δ_3	ϕ	t_d
desktop PC	0.75	3	0.6	0.8	0.9	0.90	1 h.
palmtop	0.75	3	0.6	0.9	1.0	0.95	1 day
cellular phone	0,75	3	0.5	0.9	1.0	0.95	1 day

has computed two sets of suggestions, the first considering the devices currently exploited and the second without considering the devices. The implementation of all the agents have been realized under the JADE framework [15] and the setting of the agents parameters are shown in Table 1.

The experiments involved a set A composed by $numA$ device agents (since we have considered 3 agents for each of the users, i.e. $numA$ is variable from 30 to 180), where each agent $a \in A$ contains an initial user profile, populated with a set of concepts (with cardinality ranging from 5 to 10 elements) selected by the associated user from the common dictionary. Each agent navigated through a set of the MWSs registered in an UDDI platform. In particular, this navigation has been divided into two phases. In the first phase, called *learning phase*, each agent a selected 100 MWSs in sequence, and the user profiles of both device and profile agents are updated as previously specified. In the second phase, called *test phase*, two cases have been evaluated in which the device currently exploited has been (identified by "on") or has not been considered (identified by "off"), respectively. In each one of these two parallel sub-phases each agent a performed $numC$ choices in sequence, where $numC$ is variable from 50 to 100. Each choice c consists in selecting a set L_a of three MWSs, and before each choice a set of recommendations (R_a^{on} and R_a^{off}, respectively), containing three recommendations, has been provided by an interaction with the recommender agent, following the recommendation algorithm described in 4.2. Obviously, we desire that the recommendations coincides with L_a. However, only a part of the provided recommendations are *relevant* for u, where a recommendation is relevant if it is actually selected by u and thus belongs to L_a.

To evaluate the quality of the recommendation sets R_a^{on} and R_a^{off}, generated by MWSuggest, can be used *accuracy*, *classification accuracy* and *rank accuracy* metrics (see [14]) to measure how close the predicted ratings are to the true user ratings. Accuracy measure adopts the mean absolute error (MAE) metric, defined as the average absolute difference between predicted ratings (p) and actual ratings (r). In our experiments it is computed for each user, and then averaged over all the users with respect to the total numbers of recommendations (N) generated for all the users. Formally:

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \tag{6}$$

The Receiver Operating Characteristic (ROC) sensitivity measures the frequency with which a recommender system makes correct (relevance) or incorrect

(noise) decisions about the quality of an item. The ROC curve plots recall and fallout (percentage of good and bad recommendations returned, respectively). A recommendation will be considered good if the user will rate it with 4 or above, otherwise it will be considered bad; this ROC sensitivity with threshold 3 will be referred in the following as ROC-4. Note that ROC sensitivity ranges from 0 to 1, where 1 is ideal and 0.5 is random. To simplify the comparison of multiple systems by using ROC curves, we summarize in a single performance number, known as Swets A measure, the area underneath a ROC curve. This measure is able to discriminate between good and bad recommendations. Our analysis is completed by the CROC curve ([49]) and we use the area under the CROC curve as synthetic evaluation parameter. Moreover, as Rank accuracy metrics to measure the correspondence between the items orders provided by recommender system and user, we adopt the Normalized Distance-based Performance Measure (NDPM), ranging in $[0.0, 1.0]$ (where 0.0/1.0 means best/worst recommendations, respectively), computed as:

$$NDPM = \frac{2 \cdot C^{-} + C^{u}}{2 \cdot C^{i}} \tag{7}$$

where C^{-}, C^{u} and C^{i} are the numbers of contradictory preference, compatible preference and "preferred" relations between system and user rankings, respectively. More in detail: a preference relation is contradictory when the assigned preference order of two items is reversed between system and user; a compatible preference relations happens when the user preference levels of two items is different, while the system assigns them an equal preference level; a "preferred" relation in the user ranking is referred to the pairs of items rated by the user for which one is rated higher than the other. In our experiments, we have computed the average of NPDM on all the users.

The results obtained in the generation of suggestions by the system MW-Suggest in the cases previously identified by *on* and off are show in Figure 8. In particular, Figure 8-(A) shows that the MAE measure in the *on* case is always smaller of the 16 percent (for a size of 30 agents) to 23 percent than in the off case. The good quality of the recommendations so generated is confirmed by the analysis of the Swet's A measure relative to the ROC-4 curve (see Figure 8-(B)), where the advantage to consider the device range from the 17 to 21 percent with respect to the off case. Analogous considerations can be done by considering the area under the CROC curve in Figure 8-(C). The combined analysis of MAE and ROC shows that the *on* case performs significantly better than the off case in predicting the rates of the users and in providing recommendations judged as good by the users. The analysis of the NPDM measure (Figure 8-(D)) presents performances close enough in providing a recommended ordering of items matching how the user would have ordered the same items. However, also in this case, the inclusion of the exploited device in the construction of the user profile presents the best performance, with an advantage of about 20 percent with respect to the other case.

To better analyze the behaviour of MWSuggest in the *on* configuration with respect to the off case, we have reported, for each dimension of the agent set,

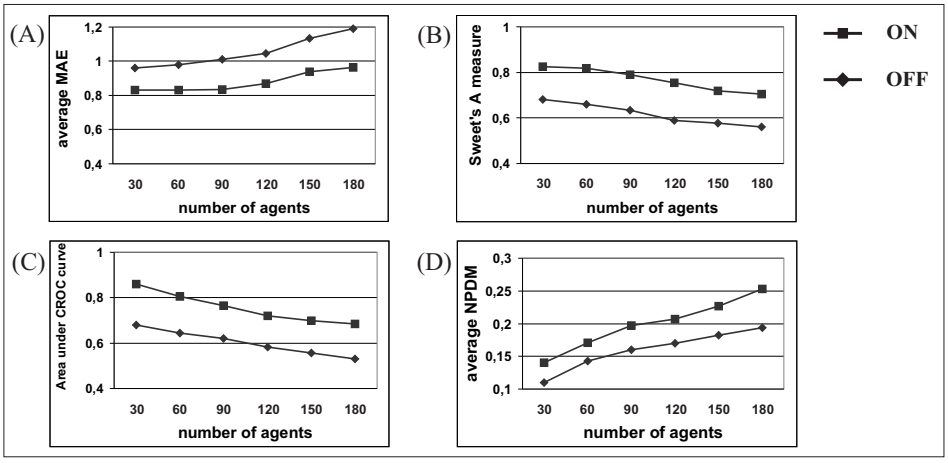

Figure 8: (A) The average MAE, (B) Swet's A measure relative to the ROC-4, (C) area under the CROC curve and (D) average NPDM relative to the two test hypothesis for different sizes of the agent-set

the average percentage of improvement introduced by MWSuggest for MAE (Figure 9) and Swet's A measure (Figure 10) and for the three cases of users exploiting a PC desktop, users exploiting a palmtop and users exploiting a cellular phone.

The results clearly show that the improvement introduced by MWSuggest is very significant for the users exploiting a cellular phone, achieving 34 percent for MAE (for a dimension of 180 agents). The analogous improvement associated to palmtop devices achieves a maximum of 22 percent, while the advantage introduced for the PC desktop users is limited to 15 percent. Also the improvement in the Swet's A measure confirms this trend, clearly showing that MWSuggest performs very well in presence of devices having limited resources, as in the case of palmtops and cellular phones, while it presents minor advantages in presence of PC desktop devices.

We have performed another experiment to better clarify this important issue. In this experiment, we have monitored three sets of users, called S1, S2 and S3, having a different composition of exploited devices. Each set contains 60 users, that can exploit a PC desktop, a palmtop and a cellular phone. The users of the set S1 use prevalently the cellular phone (in the 60 percent of the cases) while they exploit a palmtop in the 25 percent of the cases and the desktop PC only in the 15 percent of cases. The users of the set S2 use prevalently both cellular phones and palmtop (35 percent of cases for both the situations) and the PC desktop in the 30 percent of cases. Finally, the users of S3 exploit in the 70 percent of cases the desktop PC, while they use the palmtop in the 15 percent of cases and the cellular phone in the 15 percent of cases. Also in this experiment, we have used the values of Table 1 for the devices' parameters.

The results of the experiments, in terms of average improvement (computed

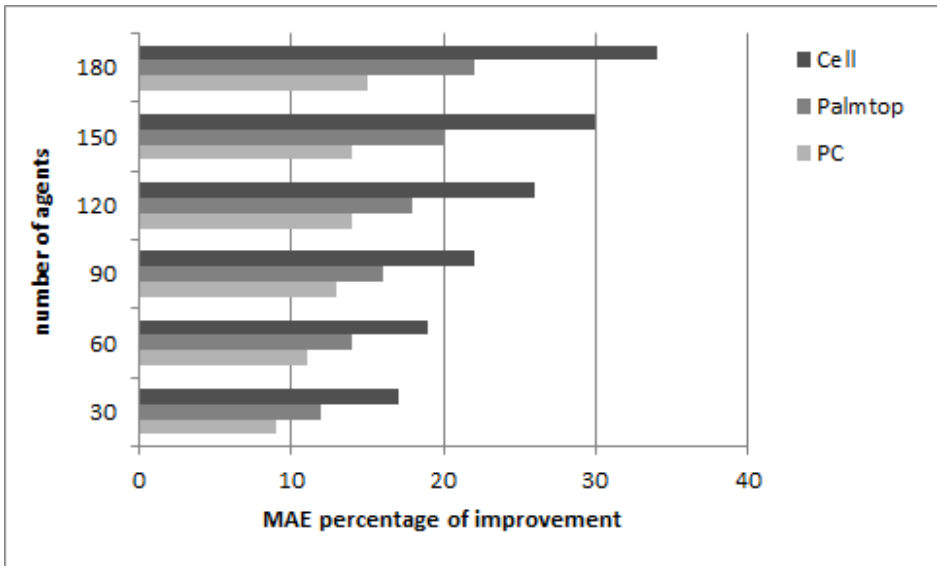

Figure 9: The average percentage of improvement for the MAE measure for different devices and different sizes of the agent-set

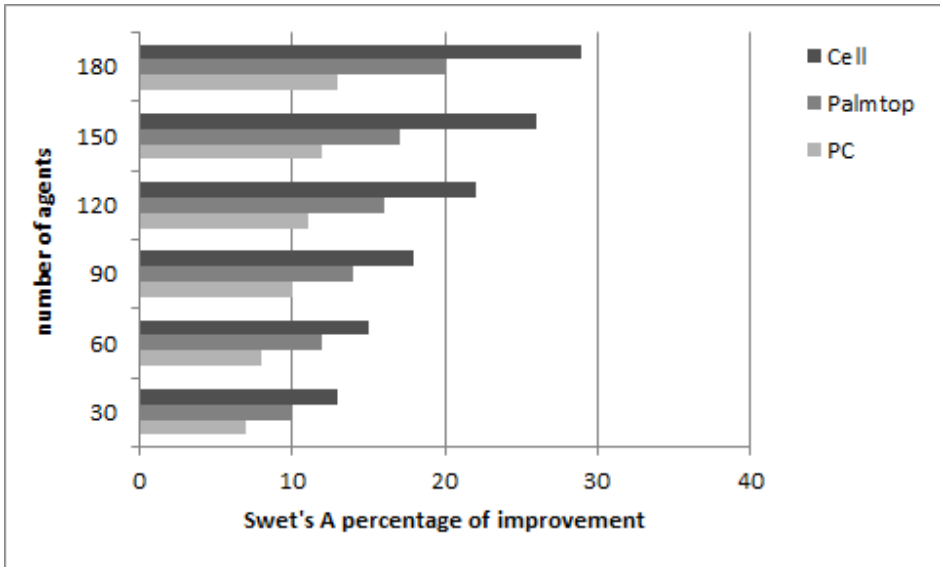

Figure 10: The average percentage of improvement for the Swet's A measure for different devices and different sizes of the agent-set

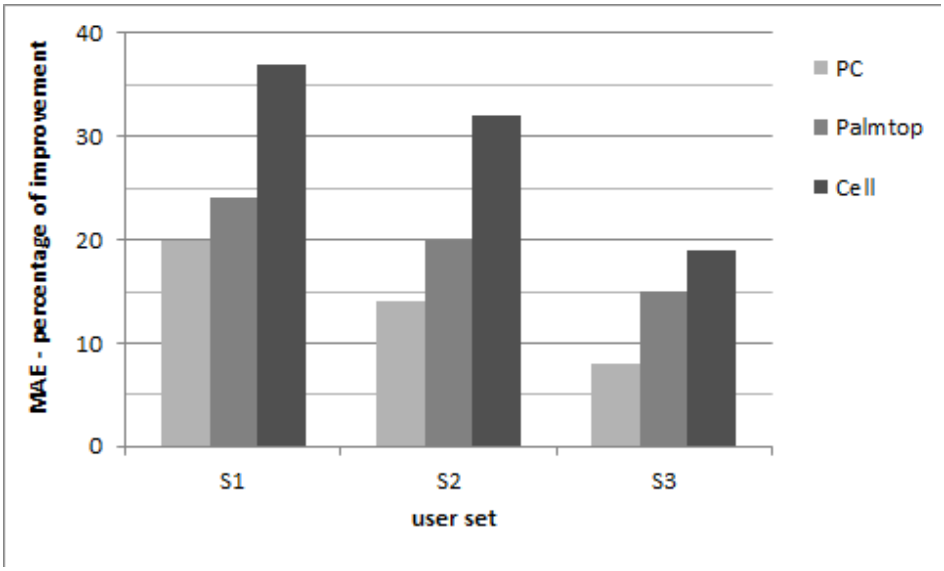

Figure 11: The average percentage of improvement for the MAE measure for the sets S1, S2 and S3

on all the 60 users belonging each set) produced using MWSuggest in configuration *on* with respect to the off case, are reported in Figure 11 for MAE and in Figure 12 for the Swet's A measure. The results have been plotted for each of the three device types.

As expected, the better performance is produced for the set S1, where the usage of cellular phones is largest. However, it is interesting to highlight that for this set we have the best performance also for the recommendations to PC users, that seem to be more effective in this situation. Evidently, the use of MWSuggest in the *on* configuration introduces significant advantages when devices with limited resources are used, and also the PC desktop users take advantage of this situation since the collaborative filtering recommendations become more effective.

This analysis also shows the limitations of MWSuggest approach, that produces small advantages in the case of a high usage of desktop PCs. In particular, for the users of the set S3, the improvement for the PC desktop users is only 8 percent for MAE and 7 percent for Swet's A measure.

6 Conclusions

In this paper we have proposed to consider the effect of the exploited device to improve the quality of the recommendations generated by a recommender system of multimedia Web services. To evaluate our approach we have presented

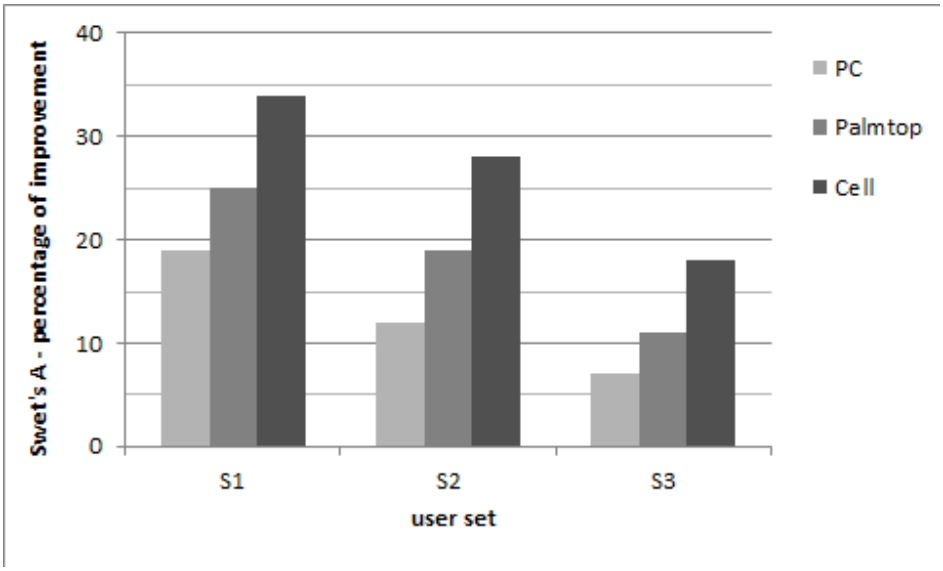

Figure 12: The average percentage of improvement for the Swet's A measure for the set S1, S2 and S3

and implemented a recommender system architecture called MWSuggest. This recommender system has been specifically designed to generate recommendations based on both user profile and exploited device in order to match more accurately user preferences and interests. In the proposed recommender system a device agent monitors a user that is exploiting a fixed device in order to build a light profile just for that device, while a profile agent constructs off-line a complete user profile. This choice leads to make very light the task of the device agent, that often has limited resources and, on the other hand, to take into account the different exploited devices in constructing the user profile. Furthermore, a recommender agent computes off line the similarity between the users and stores the users' behaviour in order to support both content-based and collaborative filtering recommendations. The experimental results, obtained by means of the proposed recommender system architecture, confirm the high quality of the recommendations generated taking into account the device currently exploited. We point out that the adoption of more rich knowledge descriptions for both users' profiles and contents of Web services, with respect to the simple XML representation that we have adopted in MWSuggest could probably further improve the quality of the recommendations. We are planning, in our ongoing research, to use ontological formalisms as, for instance, OWL, to address this important issue.

References

[1] J. Bennett and S. Lanning. The Netflix Prize. In *Proc. of KDD Cup and Workshop*, pages 3–6, 2007.

[2] F. Buccafurri, D. Rosaci, G.M.L. Sarné, L. Palopoli. Modeling cooperation in multi-agent communities. *Cognitive Systems Research*, 5(3):171–190, 2004.

[3] R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[4] M.H. Burstein, J.R. Hobbs, O. Lassila, D. Martin, D.V. McDermott, S.A. McIlraith, S. Narayanan, M. Paolucci, T.R. Payne, and K.P. Sycara. DAML-S: Web Service Description for the Semantic Web. In *Proc. 1st Int. Conf. on The Semantic Web (ISWC '02)*, volume 2342 of *LNCS*, pages 348–363, Berlin, Ger., 2002. Springer-Verlag.

[5] DAML+OIL URL. http://www.daml.org. 2011.

[6] DAML+OIL URL. http://www.w3.org/tr/daml+oil-reference. 2011.

[7] P. De Meo, A. Nocera, D. Rosaci, D.Ursino. Recommendation of reliable users, social networks and high-quality resources in a social internetworking system. *AI Communications*, 24(1):31–50, 2011.

[8] P. De Meo, D. Rosaci, G.M.L. Sarne, D. Ursino, G. Terracina. EC-XAMAS: Supporting e-commerce activities by an xml-based adaptive multi-agent system. *Applied Artificial Intelligence*, 21(6):529–562, 2007.

[9] ebXML URL. http://www.ebxml.org. 2011.

[10] D. Fensel and C. Bussler. The Web Service Modelling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[11] D. Fensel, C. Bussler, and A. Maedche. Semantic Web Enabled Web Services. In *Proc. 1st Int. Semantic Web Conf. on The Semantic Web (ISWC '02)*, volume 2342 of *LNCS*, pages 1–2, Berlin, Ger., 2002. Springer-Verlag.

[12] John D. Garofalakis, Yannis Panagis, Evangelos Sakkopoulos, and Athanasios K. Tsakalidis. Contemporary Web Service Discovery Mechanisms. *J. Web Engineering*, 5(3):265–290, 2006.

[13] Y. Hao and Y. Zhang. Web Services Discovery Based on Schema Matching. In *Proc. 13th Australasian Conf. on Comp. Sc. (ACSC '07)*, pages 107–113, Darlinghurst, Australia, 2007. Australian Computer Society, Inc.

[14] Herlocker J.L., Konstan J.A., Terveen L.G. and Riedl J.T. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[15] JADE URL. http://jade.tilab.com/. 2011.

[16] M. Kerrigan. Web Service Selection Mechanisms in the Web Service Execution Environment (WSMX). In *Proc. 2006 ACM Symp. on Applied Computing (SAC)*, pages 1664–1668, New York, USA, 2006. ACM.

[17] M. Klein and A. Bernstein. Serching for Services on the Semantic Web Using Process Ontologies. In *Proc. 1st Semantic Web Working Symp. (SWWS'01)*, pages 431–446, Stanford, USA, 2001. Stanford Univ. Press.

[18] N. Kokash, A. Biukou, and V. D'Andrea. Web Service Discovery Based on Past User Experience. In *Business Information Systems*, volume 4439 of *LNCS*, pages 95–107, Berlin, Ger., 2007. Springer.

[19] Joseph A. Konstan. Introduction to recommender systems. In *SIGMOD Conference*, pages 1373–1374, 2008.

[20] Joseph A. Konstan, Sean M. McNee, Cai-Nicolas Ziegler, Roberto Torres, Nishikant Kapoor, and John Riedl. Lessons on applying automated recommender systems to information-seeking tasks. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, pages 1630–1633, 2006.

[21] Joseph A. Konstan and John Riedl. Recommender systems: from algorithms to user experience. *User Model. User-Adapt. Interact.*, 22(1-2):101–123, 2012.

[22] Y. Li, F. Zu, Z. Wu, and F. Ma. A Scalable Web Service Discovery Architecture Based on Peer-to-Peer Overlay Network. In *Proc. of the Advanced Web Tech. and Appl. (APWeb04)*, volume 3007 of *LNCS*, pages 291–300, Berlin, Ger., 2004. Springer.

[23] Q. Liang, S.Y.W. Su, H. Li, and J.Y. Chung. A United Approach to Discover Multimedia Web Services. In *Proc. of the IEEE 5th Int. Symp. on Multimedia Software Eng. (MSE)*, pages 62–69, Washington, USA, 2003. IEEE.

[24] D. Mandell and S. McIllraith. A Bottom-up Approach to Automating Web Service Discovery, Customization, and Semantic Translation. In *Proc. 12th WWW Conf. Work. on E-Services and the Semantic Web (WWW2003); http://www.ksl.stanford.edu/sds/www-ESSW03-workshop.pdf*, New York, USA, 2003. ACM.

[25] U.S. Manikrao and T.V. Prabhakar. Dynamic Selection of Web Services with Recommendation System. In *Proc. Int. Conf. on Next Generation Web Services Practices (NWESP '05)*, page 117, Washington, USA, 2005. IEEE.

[26] N. Manouselis and C. Costopoulou. Analysis and Classification of Multi-Criteria Recommender Systems. *World Wide Web*, 10(4):415–441, 2007.

[27] E. Michael Maximilien and Munindar P. Singh. Conceptual Model of Web Service Reputation. *SIGMOD Rec.*, 31(4):36–41, 2002.

[28] S. A. McIlraith, T.C. Son, and H.i Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[29] M. Montaner, B. Lopez, and J.L. de la Rosa. A Taxonomy of Recommender Agents on the Internet. *Journal on Web Semantics (JWS)*, 19(4):285–330, 2004.

[30] M. Montebello and C. Abela. DAML Enabled Web Services and Agents in the Semantic Web. In *Proc. Web and Database-Related Workshops (NODe 2002)*, volume 2593 of *LNCS*, pages 46–58, Berlin, Ger., 2002. Springer.

[31] E. Motta, J. Domingue, L. Cabral, and M. Gaspari. IRS-II: A Framework and Infrastructure for Semantic Web Services. In *Proc. 2nd Int. Semantic Web Conf. on The Semantic Web (ISWC '03)*, volume 2870 of *LNCS*, pages 20–23, Berlin, Ger., 2003. Springer-Verlag.

[32] S. Overhage. On Specifying Web Services Using UDDI Improvements. In *Proc. of the Web and Database-Related Workshops (NODe 2002)*, volume 2593 of *LNCS*, pages 100–119, Berlin, Ger., 2002. Springer-Verlag.

[33] OWL-S URL. http://www.daml.org/services/owl-s. 2011.

[34] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *Proc. 1st Int. Semantic Web Conf. (ISWC '02)*, volume 2342 of *LNCS*, pages 333–347, Berlin, Ger., 2002. Springer-Verlag.

[35] J. Peer. Bringing Together Semantic Web and Web Services. In *Proc. 1st Int. Semantic Web Conf. on The Semantic Web (ISWC '02)*, volume 2342 of *LNCS*, pages 279–291, Berlin, Ger., 2002. Springer-Verlag.

[36] S. Pokraev, J. Koolwaaij, and M. Wibbels. Extending UDDI with Context-Aware Features Based on Semantic Service Descriptions. In *Proc. 2nd Int. Conf. on The Semantic Web (ISWC '03)*, volume 2870 of *LNCS*, pages 184–190, Berlin, Ger., 2003. Springer-Verlag.

[37] QoS URL. http://www.w3c.or.kr/kr-office/tr/2003/ws-qos/. 2011.

[38] D. Roman, U. Keller, H. Lausen, J. de Bruijn, Lara R, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.

[39] D. Rosaci. Trust measures for competitive agents. *Knowledge-based Systems*, 28:38–46, 2012.

[40] D. Rosaci and G. M. L. Sarnè. A Multi-agent Recommender System for Supporting Device Adaptivity in E-Commerce. *Journal of Intelligent Information Systems*, 38(2):393–418, 2012.

[41] D. Rosaci, G.M.L Sarné. Efficient personalization of e-learning activities using a multi-device decentralized recommender system. *Computational Intelligence*, 26(2):121–141, 2010.

[42] D. Rosaci and G. M. L. Sarnè. TRES: A Decentralized Agent-Based Recommender System to Support B2C Activities. *Lecture Notes in Computer Science*, volume 5559, pages 183–192, 2009. Springer.

[43] D. Rosaci, G. M. L. Sarnè, and S. Garruzzo. MUADDIB: A Distributed Recommender System Supporting Device Adaptivity. *ACM Transactions on Information Systems*, 27(4), 2009.

[44] D. Rosaci. Cilios: Connectionist inductive learning and inter-ontology similarities for recommending information agents. information systems. *Information Systems*, 32(6):793–825, 2007.

[45] D. Rosaci and G.M.L. Sarnè. A multi-agent recommender system for supporting device adaptivity in e-commerce. *J. Intell. Inf. Syst.*, 38(2):393–418, 2012.

[46] N A. Sajjanhar, J. Hou, and Y. Zhang. Algorithm for Web Service Matching. In *Business Information Systems, Proc. of the Advanced Web Tech. and Appl. (APWeb04)*, volume 3007 of *LNCS*, pages 665–670, Berlin, Ger., 2004. Springer.

[47] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of Recommendation Algorithms for E-Commerce. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC '00)*, pages 158–167, New York, USA, 2000. ACM.

[48] J.B. Schafer, J.A. Konstan, and J. Riedl. E-Commerce Recommendation Applications. *Data Mining Knowledge Discovory*, 5(1-2):115–153, 2001.

[49] Schein A.I., Popescul A., Ungar L.H. and Pennock D.M. CROC: A New Evaluation Criterion for Recommender Systems. *Electronic Commerce Research*, 5(1):51–74, 2005.

[50] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. In *Proc. 2nd Int. Conf. on P2P Computing (P2P '02)*, pages 104–111, Washington, USA, 2002. IEEE.

[51] W. Sherchan, S.W. Loke, and S. Krishnaswamy. A Fuzzy Model for Reasoning about Reputation in Web Services. In *Proc. of the 2006 ACM Symp. on Applied Computing (SAC '06)*, pages 1886–1892, New York, USA, 2006. ACM.

[52] K. Sivashanmugam, K. Verma, R. Mulye, and Z. Zhong. Speed-R: Semantic P2P Environment for Diverse Web Services Registries. Final Presentation, CSCI: 8350, Enterprise Integration, Dep. of Comp. Sc., Univ. Georgia, 2002.

[53] B. Smyth, E. Balfe, P. Briggs, M. Coyle, and J. Freyne. Collaborative Web Search. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 1417–1419, San Francisco, USA, 2003. Morgan Kaufmann.

[54] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[55] UDDI URL. http://uddi.xml.org. 2011.

[56] Mark Setten van. *Supporting people in finding information : hybrid recommender systems and goal-based structuring.* University of Twente, Enschede, 2005.

[57] W3C URL. http://www.w3c.org. 2009.

[58] H. Wang, D. Yang, Y. Zhao, and Y. Gao. Multiagent System for Reputation–based Web Services Selection. In *Proc. 6th Int. Conf. on Quality Software (QSIC '06)*, pages 429–434, Washington, USA, 2006. IEEE.

[59] Y. Wang and E. Stroulia. Flexible Interface Matching for Web-Service Discovery. In *Proc. 4th Int. Conf. on Web Information Systems Engineering (WISE 2003)*, pages 147–156, Washington, USA, 2003. IEEE.

[60] C.P. Wei, M.J. Shaw, and R.F. Easley. *E-Service: New directions in Theory and Practice*, chapter A Survey of Recommendation Systems in Electronic Commerce. ME Sharpe, Armonk, USA, 2002.

[61] K. Wei, J. Huang, and S. Fu. A Survey of E-Commerce Recommender Systems. In *Proc. 13th Int. Conf. on Service Systems and Service Management*, pages 1–5, Washington, USA, 2007. IEEE.

[62] WSDL URL. http://www.w3.org/tr/wsdl. 2011.

[63] WSL URL. http://www.webservicelist.com. 2011.

[64] XML URL. http://www.w3.org/xml. 2011.

[65] J. Zhang and J.Y. Chung. A SOAP-Oriented Component-Based Framework Supporting Device-Independent Multimedia Web Services. In *Proc. of the IEEE 4th Int. Symp. on Multimedia Software Engineering (MSE)*, pages 40–47, Washington, USA, 2002. IEEE.

[66] Cai-Nicolas Ziegler, Georg Lausen, and Joseph A. Konstan. On exploiting classification taxonomies in recommender systems. *AI Commun.*, 21(2-3):97–125, 2008.