Web Intelligence and Agent Systems: An international journal 5 (2007) 273–310
IOS Press

Utilization of intelligent agents for supporting citizens in their access to e-government services[1]

Pasquale De Meo[a], Giovanni Quattrone[a], Giorgio Terracina[b] and Domenico Ursino[a,*]
[a]*Università Mediterranea di Reggio Calabria, Via Graziella, Località Feo di Vito, 89060 Reggio Calabria, Italy*
[b]*Dipartimento di Matematica, Università della Calabria, Via Pietro Bucci, 87036 Rende* (*CS*)*, Italy*

Abstract. This paper aims at studying the utilization of Intelligent Agents for supporting citizens to access e-government services. For this purpose, it proposes a multi-agent system capable of suggesting to the citizens the most interesting services for them; these suggestions are determined by considering both their needs/preferences and the capabilities of the devices used by them. The paper first describes the proposed system and, then, reports various experimental results. Finally, it presents a comparison between the proposed system and other related ones already presented in the literature.

Keywords: Multi-agent systems, e-government, user profiling, device adaptivity, recommender systems

1. Introduction

The term "e-government" is generally used to indicate the utilization of Information and Communication Technologies to support both Public Administration offices, in delivering services, and citizens, in accessing them [30]. In the last few years, the number of citizens utilizing e-government services has been constantly growing; as an example, a study published in [30] shows that, in the year 2002, about 70 million US citizens accessed an e-government service at least once. Simultaneously, many Public Administration offices were showing interest in providing their services also on the Internet; as an example, in the year 2001, the Public Administration of Singapore was able to provide 92% of its services online [24].

The adoption of e-government services provides both citizens and Public Administration offices with several advantages. As an example, citizens can access online services without time and space limitations, thus avoiding the obvious problems rising when Public Administration offices must be physically reached. As for Public Administration offices, the adoption of Information and Communication Technologies allows huge reductions of management costs; as an example, a recent study conducted by Accenture in the year 2002 indicates that the development of an e-government portal in Singapore allowed the corresponding Public Administration to save 14.5 millions of US Dollars [8].

These considerations motivate the enormous, both technological and scientific, efforts made in the last few years to improve the range and the quality of online services delivered by Public Administration offices.

From a technological standpoint, these efforts have concentrated on various directions; two of the most significant ones are: (*i*) the design and the implementation of software architectures (*middleware*) supporting the cooperation of information sources associated with various Public Administration offices [31]; (*ii*) the design and implementation of telecommunication infras-

[1] A preliminary version of part of the material presented in this paper appears under the title: *A Multi-Agent System for the management of E-Government Services* in the "Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2005)", Compiegne University of Technology, France, 2005.

* Corresponding author. E-mail: ursino@unirc.it.

tructures to simplify the information exchange between Public Administration offices and citizens; these infrastructures consist of multiple information channels, such as computer networks, mobile phones, and so on (*multichannel approaches*) [23].

From a scientific standpoint, research efforts have concentrated on various directions [30]; among them we cite: (*i*) *Transaction Service Management*, mainly concerning privacy and security issues; (*ii*) *Citizen Participation*, regarding the development of tools for involving citizens in the decisional processes of Public Administration offices; (*iii*) *Information/Service Access*, aiming at simplifying both the access and the utilization of the data/services provided by Public Administration offices.

This latter research line is particularly interesting because the amount of data yearly produced by Public Administration offices is extremely large and its efficient management is a key feature for the success of an e-government portal. An Information/Service Access system can support both citizens and Public Administration offices; in fact, on one hand, it can select a set of services interesting for citizens by filtering out a (generally wide) set of services supposed to be not relevant to them [16,17]; on the other hand, it can analyze data/services of Public Administration offices for simplifying citizens' access to them [32]; as an example, it can verify if it is possible to divide offered services into simpler sub-services and the outsourcing of these last ones in such a way as to optimize the costs for their management.

This paper aims at providing a contribution in this setting; indeed, it presents a system for supporting citizens in their access to the services delivered by Public Administration offices. The proposed system provides citizens with a *personalized* and *adaptive* access to services, since it considers their profile as well as the profile of the devices they are currently utilizing in their activities.

The reference context considered in this paper is intrinsically distributed; moreover, the ultimate goals of the system and the features it should present make it particularly suited to be developed by means of the Intelligent Agent technology [44]. This technology has been extensively applied in the past for handling the distributed access to a wide variety of e-services (e.g., e-commerce, e-learning, e-recruitment, and so on). Its adoption in the context of e-government, instead, received less attention. The goal of this paper is to show, by presenting a system, that the Intelligent Agent technology not only can be applied but also can provide important benefits in this context.

In our system five types of agents operate, namely:

- *User-Device Interface Agent*; it is an interface agent that makes the communication between a user and the corresponding User-Device Agent easier; it is specialized for the device on which it must operate.
- *User-Device Agent*; it supports a user in the search of services of his interest. Each *User-Device Agent* utilizes both a *user profile*, storing the personal data, the preferences and the past behaviour of a user, and a *device profile*, registering the capabilities of the device he is currently utilizing.
- *User Profile Agent*; it manages the profiles of the users registered in our system. The presence of this agent is necessary because each user can access our system by means of several devices; as a consequence, it is necessary to maintain a unique copy of his profile, independently of the device he is currently utilizing.
- *Service Recommender Agent*; it evaluates user queries and suggests those services appearing to be the most interesting ones, according to their characteristics as well as user needs, preferences and past behaviour.
- *Public Administration Agent*; it supports Public Administration officers to add, remove or modify available services.

As previously pointed out, the main features of our system are the utilization of the Intelligent Agent technology, personalization and device adaptivity.

As far as the first aspect is concerned, it is worth pointing out that Intelligent Agents are characterized by the following properties, that are particularly interesting in our reference context [44]:

- *Reactivity*: agents are capable of detecting modifications of the environment they are operating in, and can rapidly react to these modifications by suitably adapting their behaviour.
- *Autonomy*: agents can carry out their own activities without a direct and continuous control of human users.
- *Proactivity*: agents can "foresee" user needs and, without external inputs, can plan or perform activities providing benefits to their users.
- *Social Ability*: agents can interact with other agents (or humans) to exchange information or to cooperate in performing activities.
- *Learning*: agents can apply suitable techniques (e.g., machine learning techniques) to automatically construct and maintain a user profile and can adapt their behaviour to it.

As for personalization, it is worth pointing out that our system adopts *user modelling* techniques [25] to derive and update user profiles; these play a key role in the algorithms for selecting services interesting for users and for adapting provided suggestions to user needs. In this respect, we point out that, in the e-government context, it is possible to handle particularly rich and detailed user profiles since Public Administration offices are entitled to access a wide variety of information about citizens. Clearly, the utilization of this information is regulated by laws on privacy that vary from country to country. If necessary, a user can be required to authorize the utilization of his personal data. Alternatively, it is possible to adopt an approach, analogous to that described in [32], that allows a user to formally define the rules for the utilization of his personal data that must be followed by the system.

Finally, device adaptivity appears to be particularly important in the present technological scenario where users can utilize various kinds of devices (e.g., personal computers, notebooks, PDAs, and so on) to access telecommunication networks. In order to understand this concept better, assume that a user visits a Web page related to an e-government service twice and that each visit takes n seconds. Suppose, also, that during the first access he utilizes a PDA having a low processor clock and supporting a connection characterized by a low bandwidth and a high cost. During the second access, he uses a personal computer having a high processor clock and supporting a connection characterized by a high bandwidth and a low cost. Since, during the two accesses, connection time is identical, it is reasonable to argue that the interest the user exhibited for the service during the former access is greater than that he exhibited during the latter one. The importance of device adaptivity is confirmed by the existence of several approaches, already proposed in other contexts (such as e-health, e-banking and e-learning), in which the knowledge of the devices currently utilized by users plays a relevant role (see [19,29,39]).

The plan of this paper is as follows. Section 2 presents a detailed description of the proposed system. Experiments carried out to test its performances are illustrated in Section 3. A detailed comparison of our system with other ones supporting e-government activities, and previously proposed in the literature, is presented in Section 4. An overview of some possible future enhancements of our system is presented in Section 5. Finally, in Section 6, we draw our conclusions.

2. Description of the proposed system

The general architecture of our system is shown in Fig. 1. From this figure it is possible to observe that our system is characterized by five types of agents, namely: (*i*) User-Device Interface Agent, (*ii*) User-Device Agent, (*iii*) User Profile Agent, (*iv*) Service Recommender Agent, and (*v*) Public Administration Agent.

Information about provided services is stored in a Service Database, since this information is directly handled by more than one agent. On the contrary, information about citizens accessing our system is stored in a Support Data Structure, internal to the User Profile Agent, since this is the only one that directly manages it.

In the following we describe the various components of our system in detail.

2.1. User-Device Interface Agent

The User-Device Interface Agent (hereafter $UDIA_{ij}$) is associated with a user U_j who wants to access our system by means of a device D_i.

$UDIA_{ij}$ is an interface agent, which is activated each time U_j connects to our system.

It works on D_i and is specialized in such a way as to take the characteristics of this device into account.

It supports U_j to supply our system with services of his interest; in addition, it visualizes our system's answers to U_j in a friendly fashion.

2.2. User-Device Agent

A User-Device Agent (hereafter UDA_{ij}) is associated with a user U_j utilizing a device D_i to access our system. It works on D_i and is activated each time U_j wants to access our system for performing some activity.

2.2.1. Support data structure

The Support Data Structure of UDA_{ij} consists of a triplet $\langle DP_i, UP_j, PPB_{ij}\rangle$, where DP_i represents the profile of D_i, UP_j denotes the profile of U_j and PPB_{ij} indicates the Price per Byte of D_i for U_j.

DP_i consists of a pair $\langle DevId_i, B_i\rangle$, where $DevId_i$ is the *Device Identifier* (i.e., a code identifying D_i), whereas B_i is the maximum *Bandwidth* that D_i can provide.

UP_j is represented by a tuple:

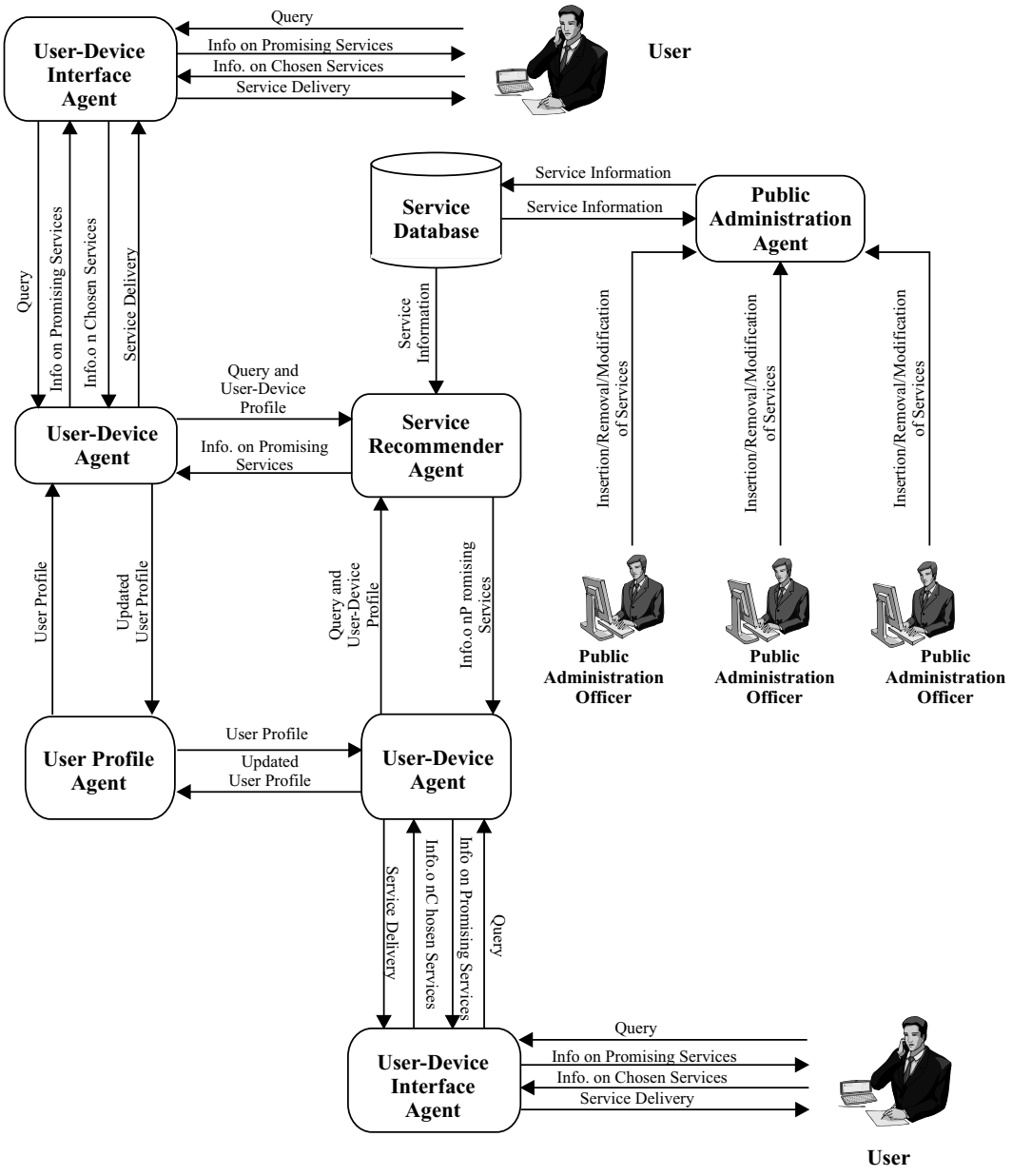

Fig. 1. Architecture of the proposed system.

$$\langle UId_j, PEDataSet_j, InterestSet_j, Audacity_j, Satisfaction_j \rangle$$

where:

- UId_j is a code identifying U_j (e.g., his Social Security Number).
- $PEDataSet_j$ stores personal and economic data of U_j. Each element $PEData_j^f \in PEDataSet_j$ consists of a pair $\langle PEName_j^f, PEValue_j^f \rangle$, where $PEName_j^f$ represents the name of a personal or economic information, whereas $PEValue_j^f$ denotes the corresponding value.
- $InterestSet_j$ represents the set of keywords that U_j has specified during his previous queries. An interest $Interest_j^s \in InterestSet_j$ is represented by a tuple:

$$\langle IntId_j^s, IntName_j^s, IntFVTS_j^s, IntLVTS_j^s, IntAvgNAT_j^s,$$

$IntAccNum_j^s\rangle$

where:

* $IntId_j^s$ is a code identifying $Interest_j^s$;
* $IntName_j^s$ is a string representing the name of $Interest_j^s$;
* $IntFVTS_j^s$ is the *First Visit Time Stamp* of $Interest_j^s$; it stores the exact time in which U_j visited a service associated with $Interest_j^s$ for the first time;
* $IntLVTS_j^s$ is the *Last Visit Time Stamp* of $Interest_j^s$; it stores the exact time of the latest access of U_j to a service associated with $Interest_j^s$;
* $IntAvgNAT_j^s$ is the *Average Normalized Access Time* of U_j to services associated with $Interest_j^s$ (see below);
* $IntAccNum_j^s$ is the number of accesses of U_j to services associated with $Interest_j^s$.

$IntAvgNAT_j^s$ normalizes the Access Time of U_j to services associated with $Interest_j^s$ against the characteristics of the devices utilized by him. In fact, this coefficient is computed by means of the formula:

$$IntAvgNAT_j^s = \frac{\sum_{k=1}^{IntAccNum_j^s} IntNAT_j^{sk}}{IntAccNum_j^s}$$

Here:

* $IntNAT_j^{sk}$ normalizes the time spent by U_j during his k^{th} access to services associated with $Interest_j^s$. It is computed as:

$$IntNAT_j^{sk} = \left(\sum_{p=1}^{Visited_j^{sk}} \left(IntT_j^{skp} - \frac{PageSize^p}{B^k} \right) \right) \times \left(PPB^k \times \left(\sum_{p=1}^{Visited_j^{sk}} PageSize^p \right) \right)$$

  where: (*i*) $Visited_j^{sk}$ denotes the number of pages visited by U_j during his k^{th} access to services associated with $Interest_j^s$; (*ii*) $IntT_j^{skp}$ indicates the time spent by U_j to access and consult the p^{th} page visited by him during his k^{th} access to services associated with $Interest_j^s$; (*iii*) $PageSize^p$ is the size, in bytes, of the p^{th} page visited by U_j during his k^{th} access to services associated with $Interest_j^s$; (*iv*) PPB^k and B^k represent the Price per Byte and the bandwidth associated with the device utilized by U_j during his k^{th} access to services associated with $Interest_j^s$.
  This formula is justified by observing that the importance given by U_j to $Interest_j^s$ can be assumed to be directly proportional to the time spent by him consulting pages related to $Interest_j^s$ (in this computation it is necessary to disregard the time needed for page download, that can significantly vary with the bandwidth of the utilized device), as well as to the price that he must pay for accessing these pages.
* $IntAccNum_j^s$ is necessary for normalizing (and, therefore, for correctly comparing) access times, by removing the dependency of the term $\sum_{k=1}^{IntAccNum_j^s} IntNAT_j^{sk}$ from the number of accesses to services associated with $Interest_j^s$ performed by U_j.

– $Audacity_j$ represents our *system's audacity* in the selection of answers to queries submitted by U_j; it belongs to the real interval $[0, 1]$ and denotes how much our system must be permissive in selecting services for U_j. As will be clear in the following, our system re-computes this coefficient after each query submitted by U_j, on the basis of his feedbacks to its previous recommendations. In the following we shall use the symbol α_j for indicating this coefficient.
– $Satisfaction_j$ represents the *satisfaction of* U_j; it belongs to the real interval $[0, 1]$ and indicates the fraction of services recommended by our system that have been really accessed by U_j. In the following we shall use the symbol σ_j for indicating this coefficient. σ_j can be computed as:

$$\sigma_j = \frac{NAccessed_j}{NRecomm_j}$$

where $NRecomm_j$ indicates the number of services our system suggested to U_j when he submitted his last query, whereas $NAccessed_j$ represents the number of services suggested by our system that U_j has really accessed.

Finally, PPB_{ij} represents the Price per Byte of D_i for U_j; it indicates the price payed by U_j for downloading a byte of data by means of D_i. PPB_{ij} is handled as a subjective measure because different prices might be associated with different users for the utilization of the same device.

PPB_{ij} is stored in our system when personal data related to U_j are inserted in it. U_j can always modify PPB_{ij}, as well as all his other personal data, by means of the User-Device Interface Agent.

2.2.2. Behaviour

The behaviour of a User-Device Agent UDA_{ij} can be described as follows:

1. It retrieves the Device Profile DP_i from D_i.
2. It requires the User Profile UP_j to the User Profile Agent.
3. It receives, from the corresponding User-Device Interface Agent, a query Q_{ij} submitted by U_j by means of D_i. Q_{ij} can be represented as a tuple $\langle SelDegree_{ij}, QKeywordSet_{ij}\rangle$. $SelDegree_{ij}$ represents our system's selectivity degree and indicates how much it must be selective in service filtering. $QKeywordSet_{ij}$ consists of a set of keywords describing desired services.
   Once UDA_{ij} receives Q_{ij}, it first updates the profile of U_j. In this activity, for each keyword $QKey^z_{ij} \in QKeywordSet_{ij}$, one of the following situations might happen:
   - there does not exist an interest $Interest^s_j$, corresponding to $QKey^z_{ij}$, in $InterestSet_j$; in this case, an interest corresponding to $QKey^z_{ij}$ is inserted in $InterestSet_j$;
   - an interest $Interest^s_j$, corresponding to $QKey^z_{ij}$, already exists in $InterestSet_j$; in this case the corresponding coefficients are suitably updated.
4. It sends Q_{ij} and its Support Data Structure to the Service Recommender Agent, which processes Q_{ij} and returns a list of services satisfying it.
5. It sends these services to U_j via the corresponding User-Device Interface Agent. At this point, U_j can choose those ones best satisfying his interests; after this choice, it computes the new value of σ_j and stores it in its Support Data Structure.
6. When the User-Device Interface Agent informs it that U_j has decided to end the current session, it sends the updated UP_j to the User Profile Agent.

In order to maintain UP_j always up-to-date, yet avoiding an excessive growth of its dimension, UDA_{ij} performs a pruning activity and removes from UP_j those keywords that no longer represent current interests for U_j. Pruning activity can be performed periodically or, if the number of interests stored in UP_j becomes excessively large, asynchronously. This task is carried out as follows: first, for each interest $Interest^s_j \in InterestSet_j$, a corresponding relevance coefficient ρ^s_j is computed; after this, all interests having a relevance coefficient less than a certain threshold $\overline{\rho}$ are removed from $InterestSet_j$.

The relevance coefficient ρ^s_j depends on:

- The number $IntAccNum^s_j$ of accesses of U_j to services associated with $Interest^s_j$; specifically, the higher the value of $IntAccNum^s_j$ is, the higher the relevance of $Interest^s_j$ for U_j will be.
- The interval $(IntLVTS^s_j - IntFVTS^s_j)$, where $IntLVTS^s_j$ (resp., $IntFVTS^s_j$) represents the Last Visit Time Stamp (resp., the First Visit Time Stamp) associated with $Interest^s_j$; specifically, the larger this interval is, the higher the relevance of $Interest^s_j$ for U_j will be.
- The exact time TS when the evaluation of ρ^s_j has been carried out; specifically, the greater the interval between TS and $IntLVTS^s_j$ is, the lower the relevance of $Interest^s_j$ for U_j will be.
- The Average Access Time $IntAvgNAT^s_j$ that U_j spent to access services associated with $Interest^s_j$, normalized w.r.t. the characteristics of the devices utilized by him; specifically, the higher the value of $IntAvgNAT^s_j$ is, the higher the relevance of $Interest^s_j$ for U_j will be.

As a consequence of the previous reasoning, ρ^s_j can be defined as:

$$\rho^s_j = IntAccNum^s_j \times \frac{IntLVTS^s_j - IntFVTS^s_j}{TS - IntFVTS^s_j} \times IntAvgNAT^s_j$$

2.3. User Profile Agent

The User Profile Agent (hereafter, UPA) has been specifically conceived for guaranteeing a device-independent management of user profiles. In fact, each user can access our system by means of several devices; as a consequence, it is necessary to maintain a unique copy of his profile, independently of the device he is utilizing during a session.

UPA stores the various user profiles in an internal support data structure. It can be activated by UDA_{ij} each time this last needs the profile of U_j. In this case, it retrieves this profile from its support data structure and sends it to UDA_{ij}.

UPA can be activated by UDA_{ij} also at the end of a session. In this case it receives the updated user profile and stores it in its support data structure.

2.4. Service Recommender Agent

The Service Recommender Agent (hereafter SRA) is the core of our system; it makes its recommendations to a user by taking available services, as well as user needs, preferences and past behaviour into account.

2.4.1. Support Data Structure

The Support Data Structure of SRA consists of a collection of Service Profiles; specifically, a Service Profile SP_l is associated with a service S_l and is represented by the tuple:

$$\langle SId_l, SName_l, SURL_l, SDescr_l, SCharSet_l, SReqSet_l \rangle$$

where:

- SId_l represents the identifier of S_l;
- $SName_l$ denotes the name of S_l;
- $SURL_l$ stores the URL where S_l can be accessed;
- $SDescr_l$ stores a brief description of S_l which will be utilized by the graphical interface for helping a user to select those services of his interest, more easily.
- $SCharSet_l$ represents the set of characteristics describing S_l; each element $SChar_l^n \in SCharSet_l$ is a keyword representing one of the characteristics of S_l;
- $SReqSet_l$ represents the set of requisites a user must have for accessing S_l; a requisite $SReq_l^m \in SReqSet_l$ is represented by a triplet $\langle ReqName_l^m, ReqOp_l^m, ReqValue_l^m \rangle$, where: (*i*) $ReqName_l^m$ represents the name of $SReq_l^m$; (*ii*) $ReqOp_l^m$ indicates an operator belonging to the set $\{>, \geqslant, <, \leqslant, =, \neq\}$; (*iii*) $ReqValue_l^m$ denotes a value for $ReqName_l^m$. $ReqOp_l^m$ and $ReqValue_l^m$, together, specify the condition associated with $SReq_l^m$ that must be satisfied by a user if he wants to access S_l. As an example, an Italian citizen can apply for a driving licence only if he is at least 18 years old; if the previous formalism is adopted, this requisite would be specified by the triplet $\langle Age, \geqslant, 18 \rangle$.

As an example, the Service Profile associated with the free online health-care support system "e-care CUP 2000" [2] is $\langle Id_1$, "Health care e-care CUP 2000", www.cup2000.it/cup2000/eng/cup2000.asp, {Health, Telemedicine, Online doctor, Consulting, Booking}, $\{\}\rangle$.

2.4.2. Behaviour

SRA is activated by a User-Device Agent UDA_{ij} when a user U_j, utilizing the device D_i, submits a query Q_{ij}. It receives Q_{ij}, DP_i and UP_j and returns the list of services answering Q_{ij}, best matching the past needs of U_j and presumably satisfying his future interests. In order to perform its task, SRA carries out the following steps:

1. It utilizes Information Retrieval techniques to extract, from the Service Database, all services satisfying user desires and constraints, as specified in Q_{ij} and UP_j. Such a task is performed by applying keyword-based matching and characteristic-based matching techniques [22].
2. It associates a numeric coefficient with each selected service; this coefficient is computed as follows. Let S_l be a service and let $MatchIntSet_{jl}$ be the set of interests of U_j satisfied by S_l; the interest degree associated with S_l is computed as:
$$\iota_{jl} = \sum_{Interest_j^s \in MatchIntSet_{jl}} \rho_j^s$$
where ρ_j^s represents the relevance of $Interest_j^s$ for U_j (see Section 2.2.2).
At the end of this phase SRA constructs a temporary list $STempList_{ij}$ of services obtained by ordering those services selected in the previous step on the basis of their interest degree ι_{jl}. $STempList_{ij}$ is already a good solution for U_j; however, two further improvements can be performed on it, making it more adequate to user needs. First, we observe that it assigns an interest degree to each service on the basis of its relevance for U_j; this is computed by taking only his past preferences into account. As a consequence, it does not consider services that U_j disregarded in the past (for example because he did not know of their existence) but that might be interesting for him in the future. Second, it might contain an excessive number of services. The next steps performed by SRA aim at implementing these improvements.
3. SRA constructs a set $SeedServices_{ij}$, obtained by selecting the first $\lceil SelDegree_{ij} \times |STempList_{ij}| \rceil$ services of $STempList_{ij}$. Services of $SeedServices_{ij}$ are utilized by SRA as seeds for selecting other services not considered by U_j in the past but that might be of his interest in the future.

4. SRA constructs the final list of services $SList_{ij}$ as:

$$SList_{ij} = SeedServices_{ij} \cup \{S_y | S_y \text{ is a service registered in the Service, Database } SD \delta(S_x, S_y) < \alpha_j \text{ for some } S_x \in SeedServices_{ij}\}$$

In this formula, $\delta(S_x, S_y)$ represents the "dissimilarity degree" between S_x and S_y. It is defined as:

$$\delta(S_x, S_y) = 1 - \frac{2|SCharSet_x \cap SCharSet_y|}{|SCharSet_x| + |SCharSet_y|}$$

where $\frac{2|SCharSet_x \cap SCharSet_y|}{|SCharSet_x| + |SCharSet_y|}$ is the well known Dice's coefficient.
δ belongs to the real interval $[0, 1]$; it is minimum when S_x and S_y coincide whereas it is maximum when they do not share any characteristic.
α_j is the audacity coefficient introduced in Section 2.2.1 and is dynamically updated by SRA on the basis of the feedbacks of U_j for the previous system recommendations. In the following we shall examine it in detail.
$SList_{ij}$ contains at least those services used as "seeds"; moreover, it could contain also some services that U_j disregarded in the past but that could be relevant for him in the future. In the selection of these additional services δ plays a key role; in fact, it measures the dissimilarity degree of two services on the basis of their semantics, without considering the relevance that U_j assigned to them in the past.
5. SRA sends to UDA_{ij} the names, the URLs and the descriptions of the services present in $SList_{ij}$, along with the new value of α_j.

The audacity α_j is incrementally updated after each query performed by U_j, on the basis of his feedbacks to the corresponding system proposals. Such a task is carried out as follows:

– If the value $\sigma_j^{previous}$ of the satisfaction coefficient after the execution of the previous query is greater than a constant value $\overline{\sigma}$ (acting as a discriminating factor), then it is possible to conclude that U_j has appreciated proposed services and, therefore, the system can try to suggest a greater number of services; as a consequence, its audacity coefficient can increase. In this reasoning, it is necessary to determine a correct value for $\overline{\sigma}$; we have carried out this task by taking users' feedback into account; at the end of this analysis we have set $\overline{\sigma} = \frac{1}{2}$.
– If $\sigma_j^{previous} < \overline{\sigma}$, then it is possible to conclude that U_j has not appreciated proposed services and desires the system to be more selective; as a consequence, the system audacity should decrease.
– If $\sigma_j^{previous} = \overline{\sigma}$, then it is possible to conclude that the system audacity should be kept constant.

The previous reasoning allows us to conclude that the higher user satisfaction is the higher the increase of α_j (w.r.t. its previous value $\alpha_j^{previous}$) should be, whereas the higher user dissatisfaction is the higher the decrease of α_j should be. In order to quantitatively specify this reasoning we introduce a function ε representing the increase (or the decrease) of the audacity coefficient. ε is defined as:

$$\varepsilon_j = | \overline{\sigma} - \sigma_j^{previous} |$$

We are now able to formally specify how α_j can be computed from $\alpha_j^{previous}$; specifically:

$$\alpha_j = \begin{cases} \min\{1, \alpha_j^{previous} + \varepsilon_j\} \\ \quad \text{if } \sigma_j^{previous} > \overline{\sigma} \\ \alpha_j^{previous} \text{ if } \sigma_j^{previous} = \overline{\sigma} \\ \max\{0, \alpha_j^{previous} - \varepsilon_j\} \\ \quad \text{if } \sigma_j^{previous} < \overline{\sigma} \end{cases}$$

2.5. Public Administration Agent

The Public Administration Agent (hereafter, PAA) is an interface agent, analogous to that described in [7]. It is utilized by a Public Administration officer (or by an employee authorized by him) for adding, modifying or removing information about the services supplied by the corresponding office. PAA plays an important role in our system since it allows our system to provide a uniform interface for managing services that might be highly heterogeneous.

2.6. Service Database

As previously pointed out, the Service Database stores the profiles of services handled by our system. The structure of a Service Profile has been illustrated in Section 2.4.1.

In order to efficiently deal with possible failures of the Service Database, we adopt the *Database Replica-*

tion strategy [34]. It uses two or more servers; one of them acts as a primary (*active*) server; the other ones act as secondary (*mirror*) servers and run on different machines.

There is a communication link between the primary server and the mirror ones; each time a service information is added, removed or updated in the primary server, the communication link is utilized for performing the same operation in the mirror ones. As a consequence, at each time instant, both primary and mirror servers store the same data. Such a synchronous data modification policy is possible because addition, removal or update of service information is quite infrequent, if compared with the other activities performed by our system.

In addition to the improvement of our system's robustness, this form of replication is useful for spreading the network/computational load across more than one server, for managing failures of individual servers and for increasing our system's processing capability.

2.7. Summary of the characteristics of our system

From the previous description it is possible to observe that our system is characterized by the following, interesting, properties:

- *It is proactive*. In fact, it not only proposes to the user those services appearing to be in line with his past behaviour but also tries to detect and suggest further services that he disregarded in the past but that appear to be presumably of interest for him in the future.
- *It is autonomous*. In fact, it constructs and handles a user profile in an unobtrusive fashion. In addition, it automatically utilizes information stored in that profile, along with data derived from user monitoring, for computing relevance and audacity coefficients. Finally, when it receives the description of new services provided by Public Administration offices, it automatically computes their corresponding dissimilarity degrees with the previously inserted services. As specified in Section 2.4.2, all these data are essential for performing recommendation activity. In general terms, it is possible to observe that our system, with all its activities, does not require the presence of external entities performing supervision and coordination tasks.
- *It shows a good learning capability*. In fact, it is capable of continuously monitoring a user for constructing and, then, updating his profile. In this activity it is capable of identifying the new interests of a user and/or removing interests appearing to be no longer relevant to him.
- *It considers the characteristics of utilized devices* when it measures the relevance of an interest, and consequently of a service, for a user. As a consequence, device profiles play a key role in the system recommendation activity.
- *It is flexible*, i.e., it is capable of operating on a wide variety of devices.
- *It is reactive*. In fact, it is capable of reacting to external stimuli and to adapting its behaviour to variations produced by them. Specifically, it is capable of reacting to: (*i*) *variations on user preferences*, since, in their presence, it modifies the corresponding user profile in such a way as to update the relevance coefficient of the various user interests; (*ii*) *variations on the utilized devices*, since, in presence of variations of the device utilized by a user, it updates the $IntAvgNAT_j^s$ coefficient and, therefore, modifies the relevance coefficient of the various user interests; (*iii*) *variations on user needs*, since, in their presence, it re-computes the audacity coefficient and, therefore, enlarges or reduces the set of proposed services.
- *It is XML-based*; specifically, (*i*) agents support data structures are represented as XML documents; (*ii*) agents communication language is ACML [18], a language obtained from the combination of XML and KQML; (*iii*) information extraction from support data structures is carried out by means of *XQuery* [4], which is becoming the standard query language for XML; (*iv*) the manipulation of agents support data structures is carried out by means of the Document Object Model (DOM) [1].
  Due to space limitations, we cannot provide here examples of the utilization of XML, ACML, XQuery and DOM. However, the interested reader can find them in [10], although they relate to a different application context.
  The usage of XML provides various benefits to our system. First, XML is rapidly spreading and is becoming the reference language for data exchange. Moreover, XML documents storing profiles and agent support data structures are textual, and, therefore, light; as a consequence, they can be handled by means of devices characterized by

limited capabilities, such as mobile phones. This would be much more difficult in case of the same information would have been stored in relational databases, whose management is much heavier.
- *It is scalable*; in fact, it is possible to extend its functionalities by simply defining new agents and integrating them with the pre-existing ones. Moreover, it is capable of handling a large number of users because User-Device Agents operate on user devices and, consequently, their overhead for the system is marginal.
- *It is robust*; specifically, it might be possible to handle more instances of the core agents, namely User Profile Agent and Service Recommender Agent, as well as more instances of the Service Database. This allows User-Device Agents to communicate with alternative agents in case of faults.
- *It is capable of uniformly handling heterogeneous services*, since the presence of the Public Administration Agent allows the adoption of a common interface for handling possibly heterogeneous services; as a consequence, highly heterogeneous services can be handled in a uniform fashion because their representation in the Service Database follows a precise common format.

In the e-government application context, a multi-agent architecture guarantees higher usage simplicity, efficiency and robustness w.r.t. a classical client-server architecture. To better clarify this concept consider a client-server e-government system in which a user (*client*) must query more databases (*servers*) that are semantically related to each other (e.g., a welfare institution and an employment agency). In this case, he must manually contact and query the various databases one by one; as a consequence, he must know of the existence, the address and the content of each of them. When the number of databases involved is large and/or when the user is a simple citizen, this task might become prohibitive. On the contrary, in our system, each user must only submit his query to a User-Device Agent by means of a friendly interface handled by a User-Device Interface Agent; the User-Device Agent, then, interacts with the Service Recommender Agent, which automatically determines those service providers appearing to be the most adequate for answering user's query, and presents their links to the user by means of a friendly interface handled, again, by the User-Device Interface Agent. This behaviour allows our system to dynamically create a personalized answer page for each user query. Moreover, numerous experimental studies point out that a multi-agent architecture guarantees a more efficient utilization of available bandwidth, a reduction of latency time and, finally, a higher fault tolerance w.r.t. a client-server architecture [6].

The usage of the Intelligent Agent technology provides various benefits even w.r.t. a classical distributed system based on cooperating processes. In fact, all properties typical of Intelligent Agents are profitably exploited: the role of autonomy and proactivity has been previously examined; adaptivity and learning capability are fundamental for allowing our system to construct and handle user profiles and to adapt its behaviour to both the profile of a user and that of the device he is currently utilizing. Collaborative behaviour also plays a key role; in fact, agents continuously exchange messages and strictly cooperate to pursue the common goal of services recommendation.

Finally, as previously pointed out, a multi-agent architecture, if compared with a classical architecture based on Web services, provides our system with important properties, such as proactivity, autonomy, learning capability, and so on.

Clearly, although a multi-agent architecture provides several benefits, it implies a higher system complexity and higher costs. Specifically, it requires the definition of quite complex policies for handling coordination, messaging and concurrency.

2.8. An overview of the system prototype

The prototype of our system has been realized in JADE (Java Agent DEvelopment Framework), a FIPA (Foundation for Intelligent Physical Agents [3]) compliant platform for developing Intelligent Agents. However, a pure JADE-based implementation requires a significant amount of memory and CPU clock, which, usually, are not available on small devices, like PDAs or mobile phones; therefore, we have adopted JADE/LEAP (*JADE Lightweight Extensible Agent Platform*), a version of JADE capable of operating also on small devices.

The choice of JADE (i.e., the choice of an open interface specifically conceived for handling multi-agent systems) allows many facilities w.r.t. generic infrastructures/architectures conceived for managing communication among objects in distributed systems [9]. Specifically:

- JADE offers a lot of built-in (and, often, off-the-shelf) functions for efficiently and effectively handling agent communication.

Fig. 2. The Web interface that our system presents to a citizen.

– JADE can choose the proper message transportation mechanism by taking the agent location into account. This lightens the programmer's work because he has no need to know of the physical addresses of agents, if they live or not on the same platform, and so on.
– JADE provides specific primitives for handling message passing in wireless environments.
– JADE offers a good scalability.

However, it is worth pointing out that, in spite of these facilities, our framework does not require a citizen to install JADE or to perform other complex software installations on his device. In fact, a citizen can access our system by means of a suitable Web interface (see Fig. 2). If he accesses it for the first time, he must register himself. On the contrary, if he is already registered, an authentication process starts; in case this process is successful, our system sends a Java applet which implements the SHA-1 algorithm for assuring the integrity of data exchanged between the e-government Web site and the citizen's browser. When this applet is run by the citizen's browser, the User-Device Agent is active on the citizen's device and he can see his personalized home page (see Fig. 3).

Each time he submits a query, our system processes it, according to the algorithm described in Section 2.4.2; after this, it generates a set of recommendations, in the form of links to external services, and displays them to him in such a way as that he can select those of his interest (see Fig. 4).

When he accesses one of these services, our system unobtrusively monitors him in order to update his profile.

In order to avoid misinterpretations of the behaviour of a user, that might be generated by his casual browsing, our system allows him to suspend the current session at any time (see the link "Suspend session" in Figs 3 and 4).

In order to understand this aspect better, consider a user U who submits a query Q and receives a list of recommended services satisfying Q. Assume that U, after having started to consult these services, has to leave our system for some reason. If our system interpreted this behaviour as the end of the current session, it would conclude that U disliked its recommendations; as a consequence, the next time U accesses it and asks for Q again, he would receive fewer recommendations

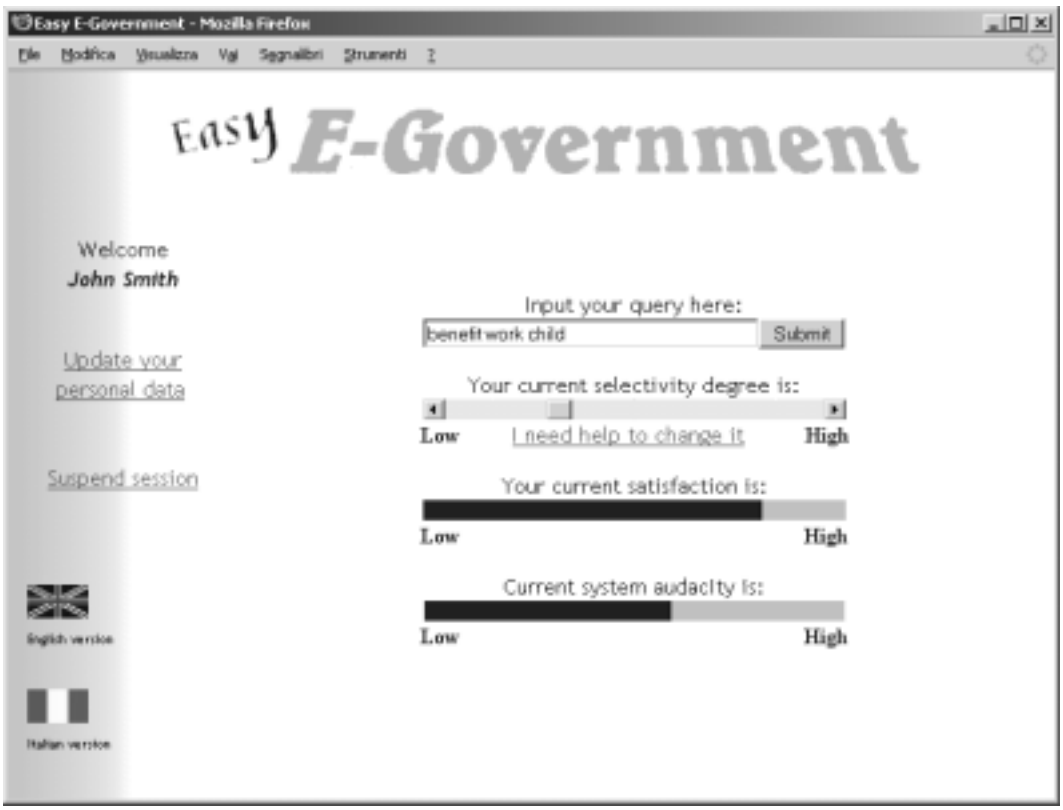

Fig. 3. The personalized page that our system shows to a citizen.

w.r.t. his previous visit; this is, probably, an undesired result for him.

The possibility to suspend the current session allows this problem to be solved. In fact, if U must leave our system, but he wants to continue the analysis of its current recommendations, he can suspend his current session, by clicking the link "Suspend session"; in this way, the next time he accesses our system, he will be presented with exactly the same configuration which he left. This has also further implications; in fact, our system considers the two accesses of U as a single session and, consequently, the User-Device Agent updates the profile of U only at the end of his second access (if he did not require any further suspension).

A manager of a Public Administration office can access our system by means of a Web interface. When a Public Administration office wants to join our system, a registration task must be performed. If a manager of a registered Public Administration office accesses our system, an authentication process is activated; in case if it is successful, our system sends a Java applet to the manager's browser. When this applet is run, the Public Administration Agent is active and the manager can insert, remove or update information about services offered by the corresponding office (see Fig. 5).

Observe that a service might have several characteristics and several requisites; in order to allow a manager to specify all of them, the Web interface presents the buttons "Add Characteristic" and "Add Requisite". When a manager clicks on one of them, our system presents him with a further page allowing the insertion of the corresponding information.

3. Experiments

This section illustrates the experiments we have carried out to evaluate our system. Specifically, in Section 3.1 we describe the characteristics of users and devices involved in the various tests. Section 3.2 analyzes the variation of accuracy measures for different values of the initial system audacity as well as the variation of system audacity against the number of submitted queries. Section 3.3 studies the role played by the device profile in the accuracy of our system. An analysis of the role of the selectivity degree is presented in

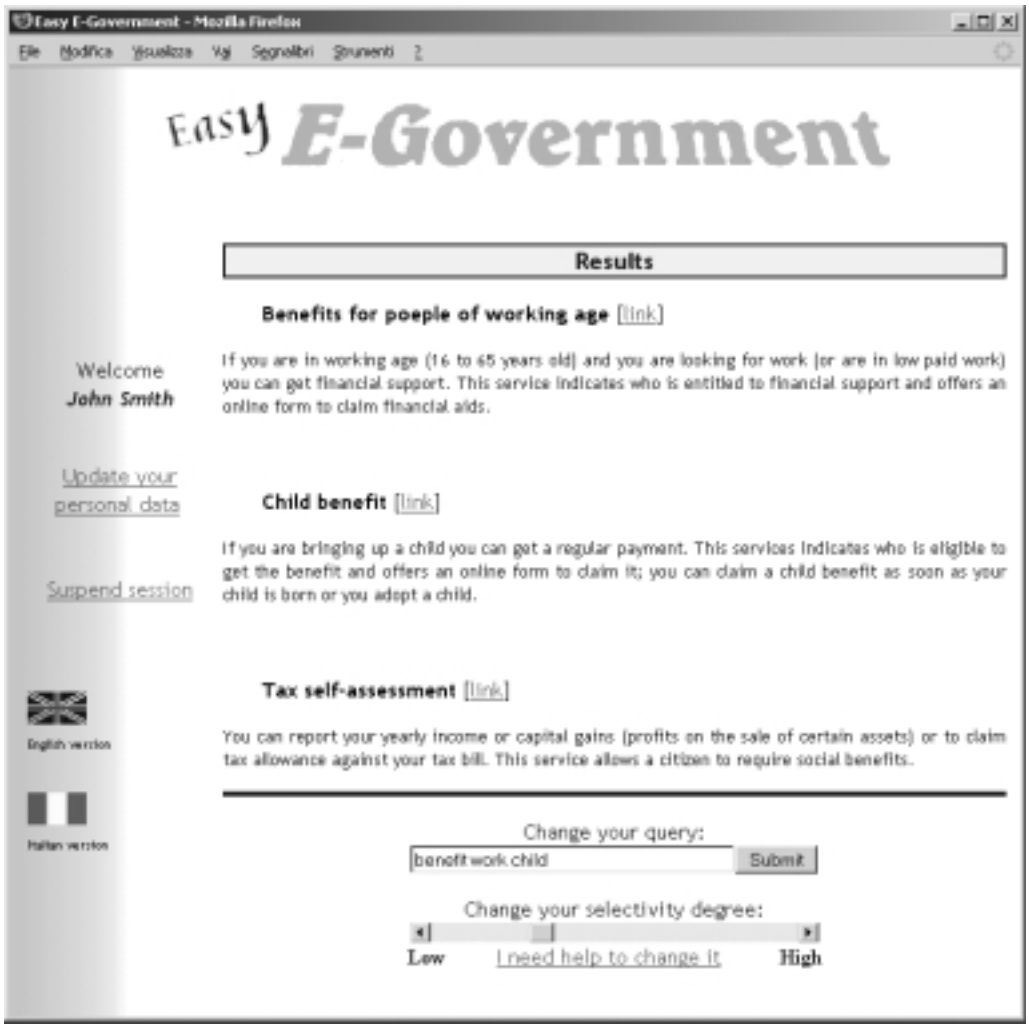

Fig. 4. The page containing the personalized recommendations to a citizen.

Section 3.4. In Section 3.5 we present an analysis of the performance of our system against an expert human administrator. In Section 3.6 we analyze the computational costs necessary for making our system adaptive w.r.t. user's needs. In Section 3.7 we analyze the answer delay of our system in overload conditions. Finally, in Section 3.8, we present an experimental comparison between our system and other e-government systems currently available on the Internet.

3.1. Characteristics of users and devices

In our system, the User Profile Agent, the Public Administration Agent and the Service Recommender Agent operate on Personal Computers equipped with a 3.4 GHz CPU and 512 Mb of RAM; some of the User-Device Agents and of the User-Device Interface Agents have been installed on computers equipped with a 2.6 GHz CPU and 256 Mb of RAM, whereas others ran on QTEK 2020 PDAs equipped with a 400 MHz CPU and 128 Mb of RAM.

In our tests we have considered a set $USet = \{U_1, U_2, \ldots, U_{30}\}$ of 30 users and a set of 90 services derived from the Italian Government Web site http://www.italia.gov.it. Selected services were associated with different application domains, such as Health, Welfare, Education, Public Transports, and so on.

3.2. Evaluation of the impact of audacity on the accuracy of our system

As pointed out in Section 2.4, audacity coefficient plays a key role in our system since it allows user

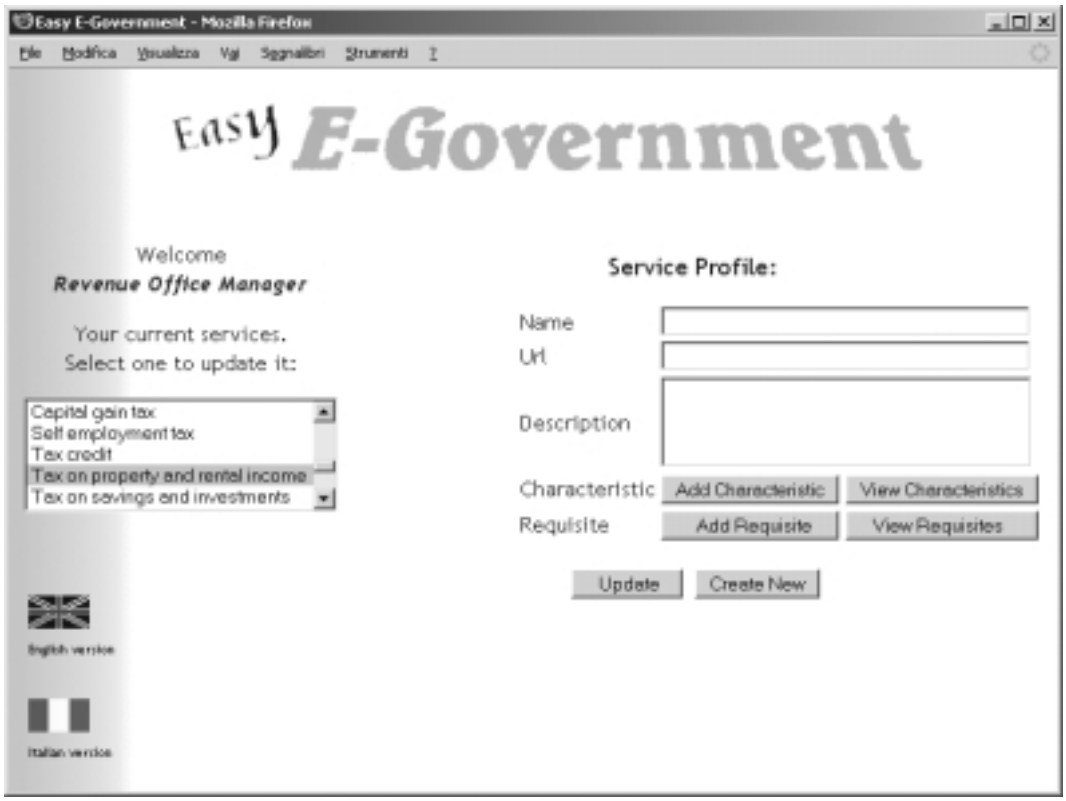

Fig. 5. The Web interface that our system presents to a manager of a Public Administration office.

satisfaction on past proposals to be taken into account.

Our approach requires to state an initial value for it[2]; as a consequence, the tuning of this initial value and its role in the system accuracy must be carefully studied. For this purpose, we have adopted some classical measures for evaluating system accuracy, namely *Precision*, *Recall*, *F-Measure* and *Overall*; these measures have been initially defined in the field of Information Retrieval [5,41] and, then, they have been applied to measure the accuracy of Recommender Systems [37].

Precision is defined as the proportion of retrieved and relevant services to all the services retrieved; analogously, Recall is defined as the proportion of relevant services that are retrieved, out of all relevant services available. Both Precision and Recall vary in the real interval $[0, 1]$; high values of them indicate a good accuracy.

The adoption of Precision and Recall is explained as follows: an e-government system should maximize the number of relevant services retrieved by it and should minimize the number of irrelevant ones. The system is optimum if all relevant services are retrieved (we call this property "completeness" in the following) and no irrelevant service is suggested to the citizen (we call this property "soundness" in the following). Precision is a measure of the soundness of a system whereas Recall is a measure of its completeness. A system is, then, optimum (i.e., it is sound and complete) if both Precision and Recall are equal to 1. Actually, this is an ideal case; in real cases Precision and Recall are generally lower than 1 and, often, are conflicting measures (i.e., for obtaining a good Precision, often, Recall must be sacrificed, and vice versa).

Precision and Recall have been computed as follows:

- each user $U_j \in USet$ submitted a query; this has been processed by means of classical Information Retrieval techniques (keyword-based and characteristic-based matchings, see Section 2.4) and a set $SSet_j$ of services possibly satisfying it has been generated;
- our system has been applied to identify a set $SS_j \subseteq SSet_j$ of services considered particularly interesting for U_j;

[2]Recall that the value of audacity coefficient belongs to the real interval $[0, 1]$.

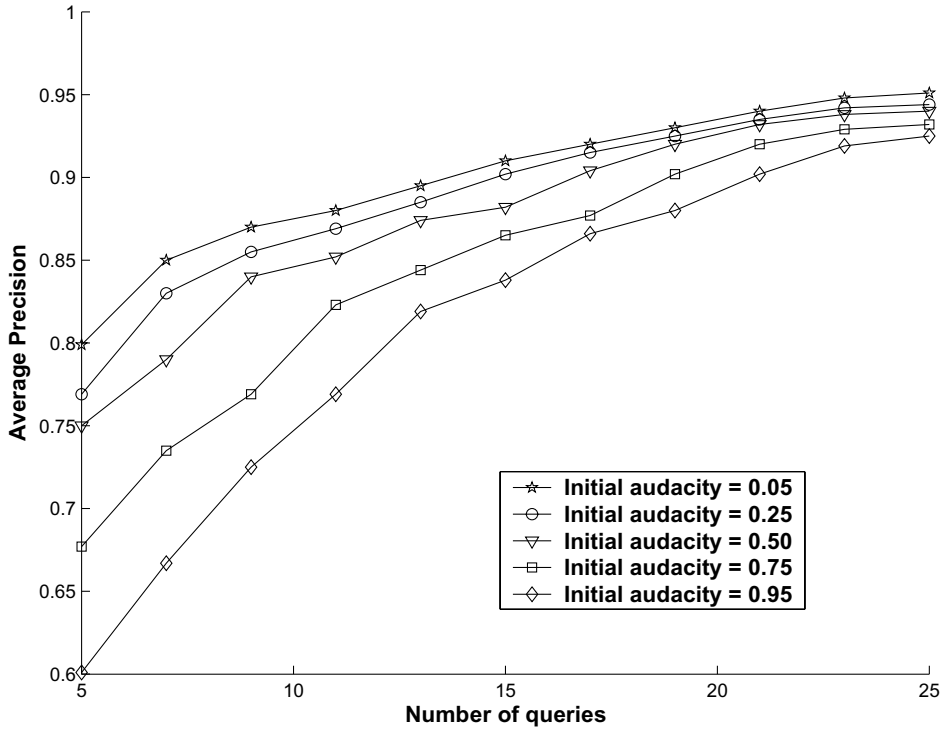

Fig. 6. Impact of the initial value of audacity coefficient on the Precision of our system.

– each user U_j has been asked to specify a set $SU_j \subseteq SSet_j$ of services that he considered interesting. The Precision P_j, associated with U_j, has been defined as:

$$P_j = \frac{|SU_j \cap SS_j|}{|SS_j|}$$

whereas the Recall R_j has been computed as:

$$R_j = \frac{|SU_j \cap SS_j|}{|SU_j|}$$

However, neither Precision nor Recall *alone* are good indicators of the accuracy of a system. In fact, a system might achieve a high Precision at the expense of a poor Recall by returning few (almost surely sound) services; in this case users might be dissatisfied because many relevant services are filtered out. On the other side, Recall can be easily maximized at the expense of a poor Precision by returning all services that might be, even vaguely, interesting for users. Also in this case users might be dissatisfied because they would be overwhelmed by many irrelevant proposals. This last reasoning motivates the definition of joint measures between Precision and Recall; two very popular joint measures are F-Measure and Overall.

F-Measure [41] represents the harmonic mean between Precision and Recall; it is defined as:

$$F_j = 2 \times \frac{P_j \times R_j}{P_j + R_j}$$

it varies in the real interval $[0, 1]$; the higher F_j is, the more accurate the system will be. F-Measure gives the same relevance to Precision and Recall in assessing the accuracy of a system; in fact, the formula defining this measure is symmetrical w.r.t. P_j and R_j (i.e., P_j and R_j are interchangeable).

Overall [33] measures the effort needed for adding false negatives and removing false positives from the set of services returned by a system; it is defined as:

$$O_j = R_j \times (2 - \frac{1}{P_j})$$

O_j is a real value between $-\infty$ and 1; high values of O_j indicate a good accuracy.

Figures 6, 7, 8 and 9 show the values of Precision, Recall, F-Measure and Overall, averaged on all involved users, against the number of submitted queries[3]. From

[3] Note that when the number of queries submitted by a user grows, the system refines the corresponding User Profile (see Section 2.2).

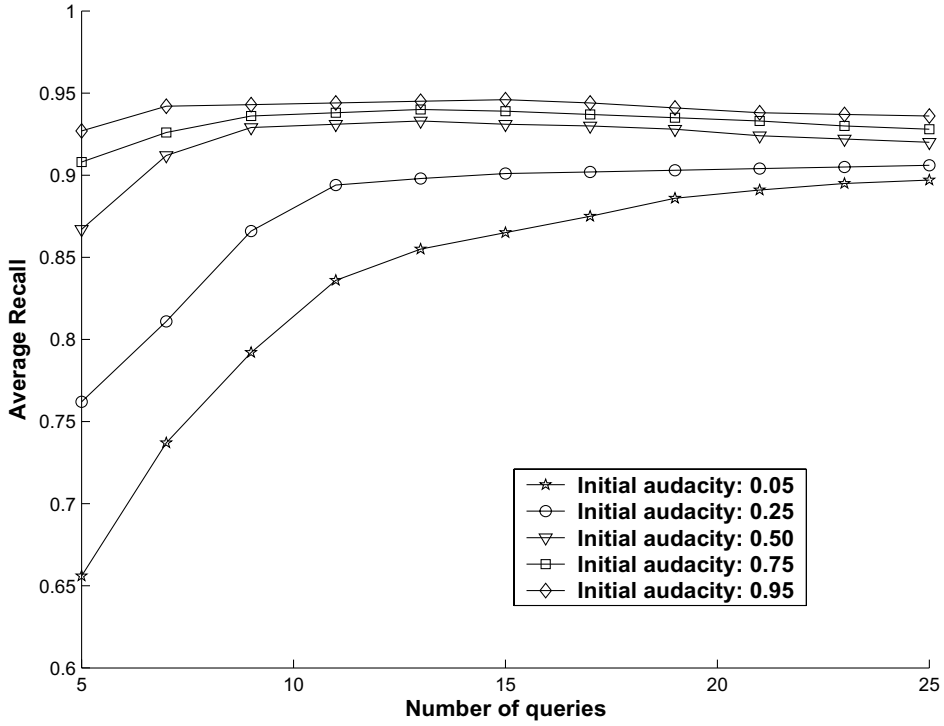

Fig. 7. Impact of the initial value of audacity coefficient on the Recall of our system.

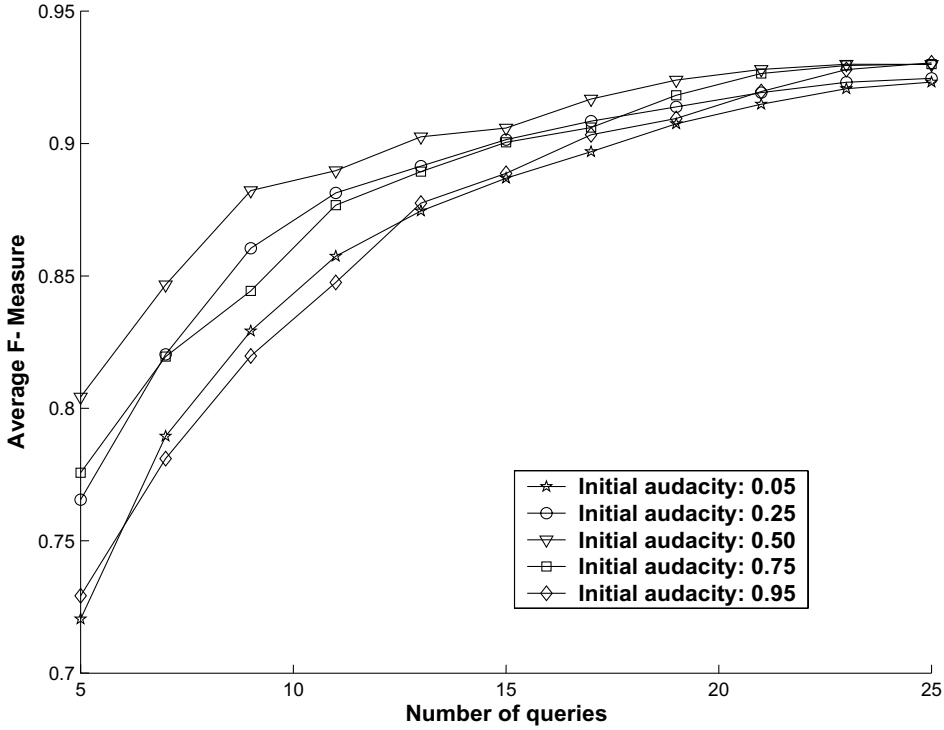

Fig. 8. Impact of the initial value of audacity coefficient on the F-Measure of our system.

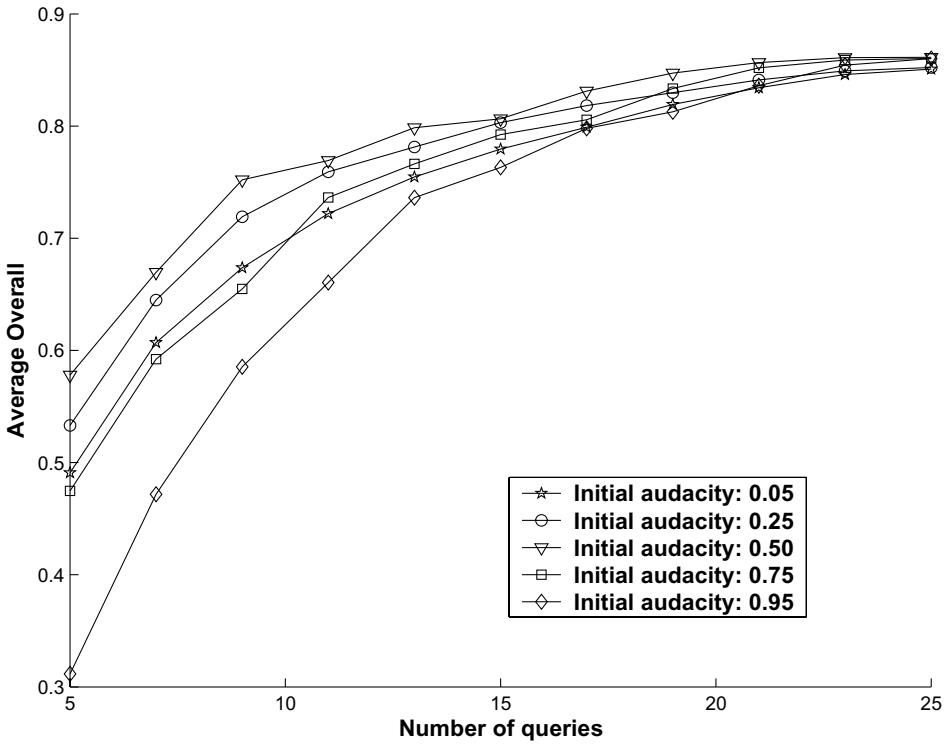

Fig. 9. Impact of the initial value of audacity coefficient on the Overall of our system.

the analysis of these figures, it is possible to observe that the best results are obtained if the initial audacity is 0.50. In fact, "low" values of the initial audacity (e.g., lower than 0.25) allow high values of Precision, but not particularly high values of Recall, to be obtained. On the contrary, "high" values of the initial audacity (e.g., higher than 0.75) allow extremely satisfying values of Recall, but not particularly high values of Precision, to be obtained.

This behaviour can be explained by the following reasoning: if the initial audacity is "low", our system is extremely cautious and suggests only those services it considers interesting for the user with a high degree of confidence; as a consequence, almost all its suggestions are sound, and this justifies the high values of Precision obtained already during the initial queries. However, since a small number of services is selected, it may be that those services appearing only partially interesting for users are discarded; this negatively influences Recall. When the number of user queries increases, the audacity value grows; this allows our system to suggest a higher number of services and causes an increase of Recall. Moreover, information stored in user profiles grows both in "quantity" (i.e., a profile is enriched with new data) and in "quality" (i.e., a profile stores information better describing user needs/preferences). As a consequence, service selection is more accurate and this produces an increment of Precision.

An opposite behaviour is registered if the initial audacity value is "high"; in this case, our system filters out only few services and suggests also services that might not be interesting for the user. This behaviour allows high values of Recall to be obtained already in the initial queries; however, the system could erroneously classify as interesting some services discarded later by the user; this negatively influences its Precision. When the number of user queries grows, audacity value is progressively reduced, our system is more "selective" and filters out a higher number of services; this causes a sensible improvement of Precision and a (limited) reduction of Recall.

An initial audacity value of 0.50 is capable of suitably balancing the two opposite behaviours described above; values of F-Measure and Overall (i.e., the two accuracy measures taking both Precision and Recall into account) confirm that this initial audacity value guarantees the best results (see Figs 8 and 9).

In order to provide an in-depth analysis of the role of the audacity coefficient in our system, in Fig. 10 we show its values, averaged on all involved users, against the number of submitted queries.

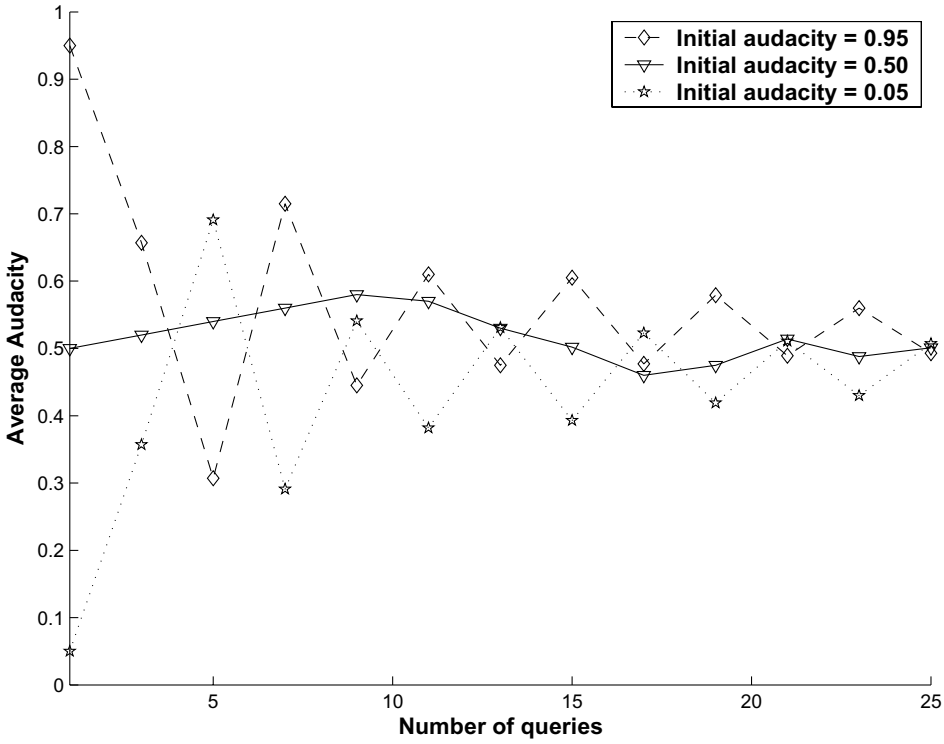

Fig. 10. Variation of the audacity of our system against the number of submitted queries.

From the analysis of this figure we can see that, independently of the initial value, average audacity tends to converge to 0.50. Specifically:

- If the initial audacity is very high (e.g., about 0.95), then it quickly decreases to a low value (i.e., about 0.30). After this, it presents various, quite high, fluctuations; these decrease when the number of submitted queries increases.
- An opposite behaviour can be observed when the initial audacity is very low (e.g., about 0.05); in this case it is possible to see that audacity quickly increases to a high value (i.e., about 0.70). After this, it presents some large fluctuations that, however, decrease with the increase of the number of submitted queries.
- A more regular trend can be observed if the initial audacity is set to 0.50 (i.e., the initial audacity value detected to be optimal in the previous experiment). In this case fluctuations are more limited and regular and the convergence to 0.50 is much more rapid than in the previous cases.

Finally, observe the trend of Average Recall when system audacity is set to its optimal value (i.e., 0.50). In Fig. 7 we can see that Average Recall initially increases and, then, after about 10 queries, slightly decreases to an asymptotic value. This trend can be understood by observing the variation of audacity against the number of submitted queries when its initial value is 0.50. In Fig. 10 we can see that, initially, audacity increases and reaches its maximum value when the number of submitted queries is about 10; a high audacity value implies that our system proposes to citizens a high number of services; this behaviour tends to increase its Recall and justifies that the maximum value of Recall is obtained when audacity is maximum. After 10 queries audacity decreases; this justifies the slight reduction of Recall. After some other queries, audacity tends to converge to 0.50; moreover, citizen profiles become quite rich and, consequently, the interest degree associated with a service can be more accurately computed and services of interest can be more correctly detected. For this reason Average Recall tends to an asymptotic value.

As a further interesting issue, observe that all curves depicted in Fig. 7 tend to the same asymptotic value, independently of the initial audacity value, although this last value influences the “convergence speed”. This trend can be explained by the fact that all audacity curves tend to 0.50, although the corresponding “convergence speed” is different.

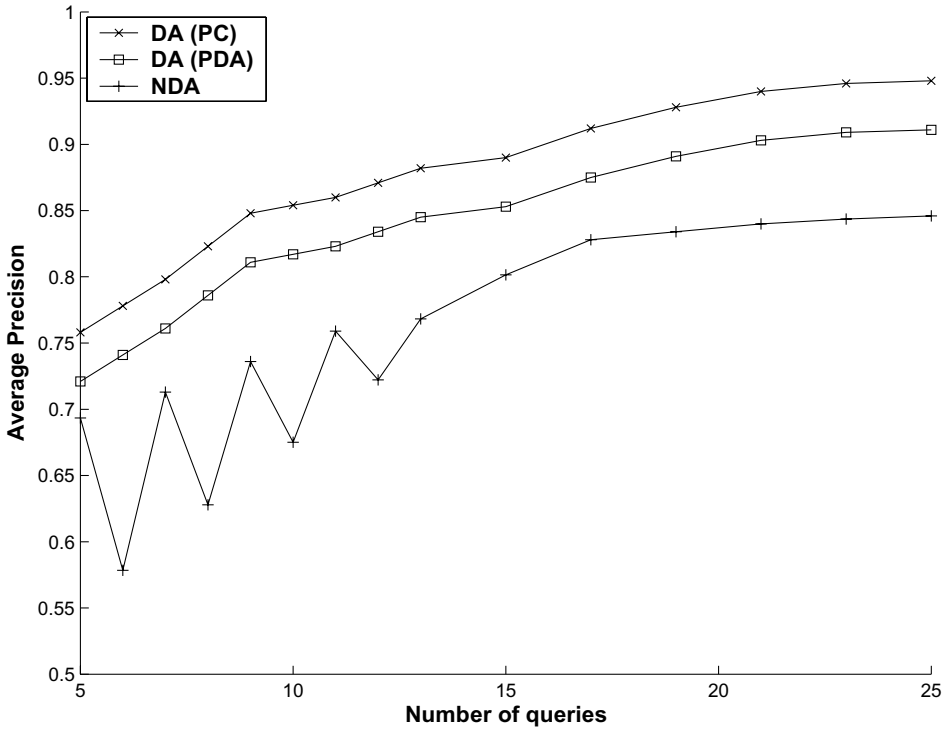

Fig. 11. Precision of our system against the number of queries for NDA, $DA(PC)$ and $DA(PDA)$ configurations.

3.3. *Analysis of the role of the device profile in the accuracy of our system*

A second series of experiments has been performed for evaluating the improvements of the accuracy of our system obtained from the utilization of device profiles. Specifically, we have considered two kinds of devices, namely a Personal Computer and a PDA, whose characteristics have been described in Section 3.1; moreover, we have asked each user to choose, for each query, the preferred device. From these choices, it emerged that 72% of times users preferred to access our system by means of a Personal Computer; 28% of times, instead, their preference was for PDAs.

In a first phase (that we call *non device-aware*, *NDA* for short) system suggestions have been computed without taking device profiles into account. In a second phase (that we call *device-aware*, *DA* for short), these profiles have been taken into consideration; in order to improve the significance of our experiment we have partitioned *DA* configuration in two sub-configurations, namely $DA(PC)$, when utilized device was the Personal Computer, and $DA(PDA)$, when utilized device was the PDA. Finally, we have asked each user to validate system results for both *NDA*, $DA(PC)$ and $DA(PDA)$ configurations; this allowed us to compute the value of Precision, Recall, F-Measure and Overall, averaged on all users, against the number of queries submitted by them; the initial system audacity has been set to 0.50. Figures 11, 12, 13 and 14 show the variation of Average Precision, Average Recall, Average F-Measure and Average Overall against the number of submitted queries for NDA, $DA(PC)$ and $DA(PDA)$ configurations.

From the analysis of these figures it is possible to observe that, in the *DA* configurations, accuracy is significantly better w.r.t. the *NDA* configuration. Such an improvement derives from the fact that the knowledge of the characteristics of the device utilized by a user for accessing recommended services in the past allows his interest for them to be quantified in a more precise fashion and, consequently, provides a better knowledge of his preferences and needs.

In addition, we observe that, in the *DA* scenarios, Precision, Recall, F-Measure and Overall present a more "stable" trend w.r.t. the *NDA* scenario, where an oscillatory trend can be observed. In order to understand the reasons underlying this behaviour, consider a user U_j and an interest $Interest_j^s$. As previously pointed out, the Access Time of U_j for services associated with

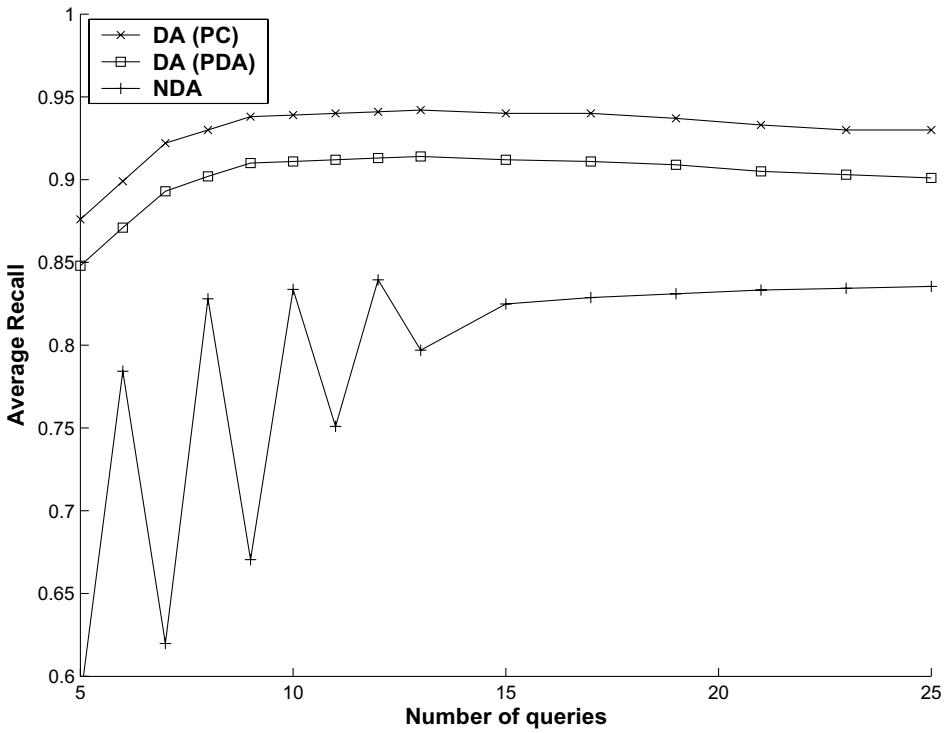

Fig. 12. Recall of our system against the number of queries for NDA, $DA(PC)$ and $DA(PDA)$ configurations.

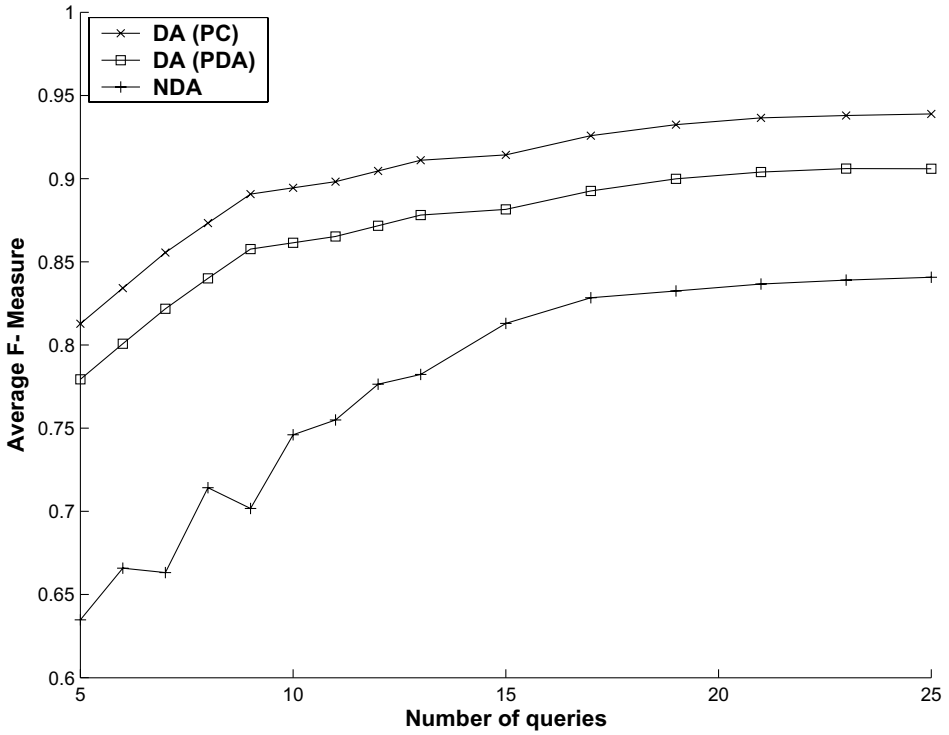

Fig. 13. F-Measure of our system against the number of queries for NDA, $DA(PC)$ and $DA(PDA)$ configurations.

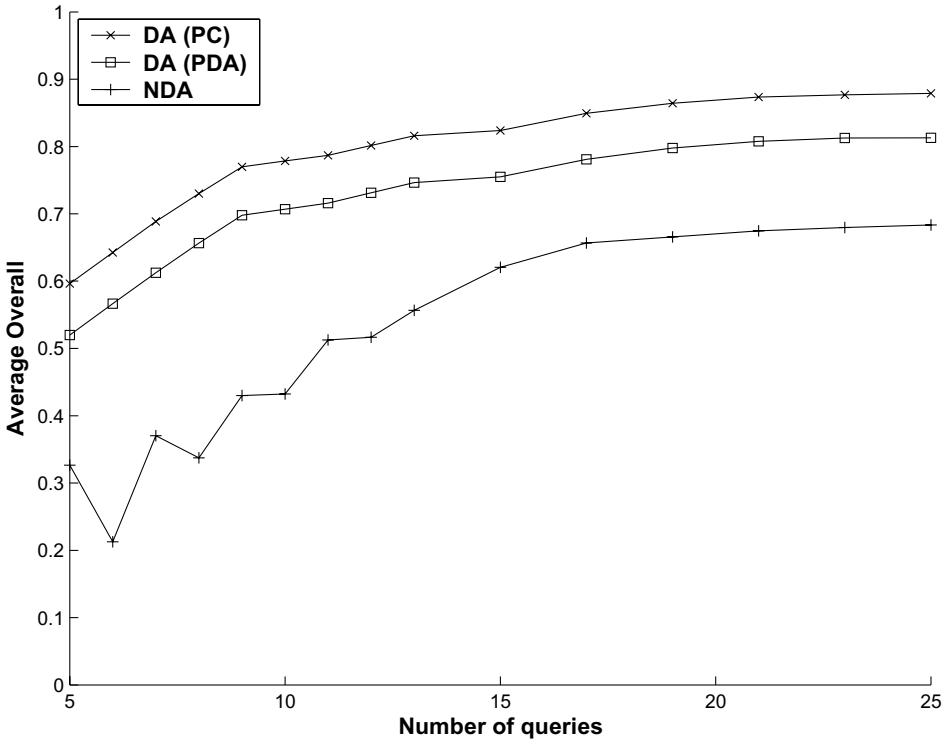

Fig. 14. Overall of our system against the number of queries for NDA, $DA(PC)$ and $DA(PDA)$ configurations.

$Interest_j^s$ plays a key role in the behaviour of our system. This Access Time is analyzed in very different ways in the DA and in the NDA configurations. In fact, in the NDA configuration, it is simply set to the time (expressed in seconds) spent by U_j for accessing services associated with $Interest_j^s$ and suggested by our system. On the contrary, in the DA configurations, the influence of this time is mitigated by the factors: (*i*) $\frac{PageSize^p}{B^k}$, which subtracts the time necessary for downloading the corresponding pages (in such a way as to disregard the time in which the user is inactive because he is forced to wait for page downloading); (*ii*) PPB^k, which takes the connection cost of the device the user is utilizing into account.

In order to understand the positive consequences caused by the more "stable" trend observed in the DA configurations, in the following, we analyze the impact of underestimation and overestimation errors in accuracy measures.

In case of an overestimation, $Interest_j^s$ is classified as relevant for U_j, even though it would be uninteresting for him; as a consequence, our system suggests to U_j many services specifying $Interest_j^s$ among their characteristics; in this case U_j filters out almost all these services and our system reacts to this action by notably reducing its audacity.

In case of an underestimation, $Interest_j^s$ is classified as irrelevant for U_j even though it would be interesting for him; as a consequence, our system suggests to U_j a very limited number of services specifying $Interest_j^s$ among their characteristics; in this case U_j selects all of them and our system reacts to this action by suitably increasing its audacity.

As a consequence, until our system is able to construct a reliable profile of U_j, audacity fluctuates from high values to low ones, and vice versa; this implies that the number of services suggested for two comparable queries in two subsequent iterations might be very different. Therefore, it may be that, for a specific query, our system generates a small set of sound answers (i.e., it achieves a high Precision but a low Recall) but, for the next query, it might return a large amount of (possibly irrelevant) results (i.e., it achieves a high Recall but a low Precision), and vice versa.

The previous reasoning explains the quick initial fluctuations of Precision, Recall, F-Measure and Overall for the NDA scenario (where estimation errors might play an important role), and the more "stable" trend observed for the $DA(PC)$ and $DA(PDA)$ scenarios.

Another important conclusion can be drawn from Figs 11, 12, 13 and 14 by comparing the accuracy mea-

sures obtained for $DA(PC)$ and $DA(PDA)$ configurations.

Specifically, the Precision obtained for $DA(PC)$ configuration is rather higher than that obtained for $DA(PDA)$ configuration. A similar trend can be observed for Recall although, in this case, the corresponding gap is mitigated.

This result can be explained by the fact that, in spite of the enormous advances made in the last few years, generally, the access to a Web system performed by means of a PDA is still more difficult and expensive than the access to the same Web system performed by means of a Personal Computer. Although our system is device-adaptive and, consequently, takes this fact into account and tends to propose only particularly relevant services to a user accessing it by means of a PDA, we have observed that it is still optimistic, if compared with the behaviour of real users that, in these cases, generally, tend to consult only very few services for each submitted query. This reasoning explains the trend observed for Precision in the $DA(PC)$ and $DA(PDA)$ configurations.

In addition, the tendency of citizens to consult only few proposed services, if they are accessing our system by means of a PDA, induces our system to reduce its audacity: as a consequence, for each submitted query, it tends to propose quite a limited number of services. This fact explains the trend observed for Recall in the $DA(PC)$ and $DA(PDA)$ configurations.

The experiment illustrated in this section makes clear the importance of device adaptivity for our system. Clearly, a system supporting this feature is more complex to manage; however, this complexity is largely balanced by the improvements of the result accuracy obtained by it.

3.4. Analysis of the role of the Selectivity Degree on the performances of our system

In Section 2.2 we have shown that, in our system, each submitted query Q_{ij} can be represented as a pair $\langle SelDegree_{ij}, QKeywordSet_{ij}\rangle$, where: (*i*) $SelDegree_{ij}$ represents the system selectivity degree, indicating how much it must be selective in service filtering; (*ii*) $QKeywordSet_{ij}$ represents the set of keywords associated with Q_{ij}. In this section we illustrate the experiments that we have performed for studying the role of $SelDegree_{ij}$ in the behaviour of our system.

In Figs 15 and 16 we plot the average accuracy measures obtained by our system when the selectivity degree varies between 0 and 1. In these experiments the initial value of audacity coefficient has been set to that guaranteeing the best performances, i.e., 0.50 (see Section 3.2). Furthermore, in order to make our tests independent of the number of submitted queries, we have started to compute all accuracy measures after the initial training phase (i.e., after 15 queries), in such a way as that user profiles are rich enough.

From the analysis of Fig. 15 it is possible to observe that, for low values of the selectivity degree, Precision is prioritised over Recall; on the contrary, for high values of this coefficient, Recall is prioritised over Precision. This behaviour can be explained by a reasoning analogous to that we have illustrated for audacity coefficient (see Section 3.2).

Figure 16 shows that, when the values of the selectivity degree range between 0.3 and 0.6, our system presents the best trade-off between Precision and Recall and is capable of guaranteeing the best global accuracy. In fact, when the selectivity degree is within this range, both F-Measure and Overall reach their best values.

As a further remark about this experiment, we observe that our system provides a user with a flexible mechanism for tuning the various aspects of accuracy. In fact, if he wants to prioritise Precision over Recall (without a strong detriment to it), he can choose a low value of selectivity degree (i.e., near 0.3), whereas, if he desires to prioritise Recall over Precision (without a strong detriment to it), he can choose a higher value of selectivity degree (i.e., near 0.6).

In Fig. 17 we illustrate the variation of the Response Time of our system when selectivity degree ranges between 0 and 1. From an analysis of this figure it is possible to observe that the best Response Time is obtained for low values of the selectivity degree (i.e., for values less than 0.25). This result is explained by the fact that, in presence of a low selectivity degree, the number of seed services considered by SRA is limited (see Section 2.4.2); as a consequence, it is necessary to compute few dissimilarity degrees and, therefore, Step 4 of Section 2.4.2 can be quickly executed. However, Fig. 17 shows that the increase of the Response Time caused by an increase of the selectivity degree is limited, if compared with the overall Response Time of our system.

3.5. Analysis of the performance of our system against an expert human administrator

In order to further analyze the performance of our system, we have carried out another experiment, de-

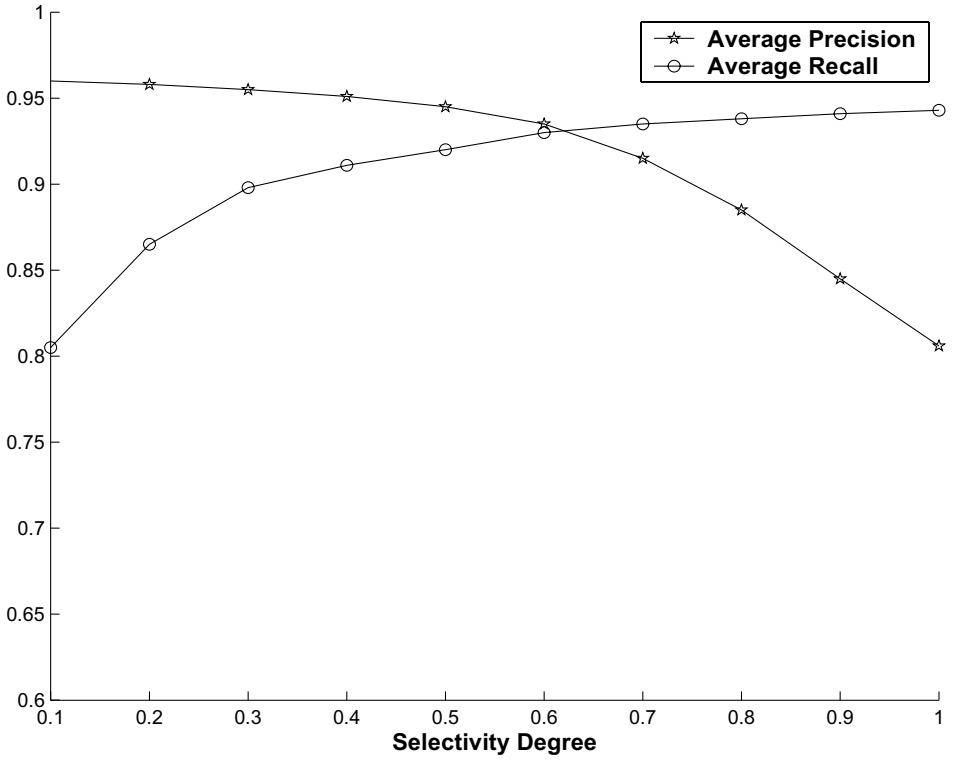

Fig. 15. Variation of Precision and Recall against variation of the selectivity degree.

voted to verifying to what extent the recommendations of our system and those provided by an expert human administrator were similar.

In this experiment we have once again used Precision and Recall as performance measures. The experiment has been performed as follows:

- each user $U_j \subseteq USet$ submitted a query; this has been processed by means of classical Information Retrieval techniques and a set $SSet_j$ of services possibly satisfying it has been generated;
- the expert human administrator has been asked to identify a set $SE_j \subseteq SSet_j$ of services that he considered particularly interesting for U_j;
- our system has been applied to identify a set $SS_j \subseteq SSet_j$ of services considered particularly interesting for U_j;
- U_j has been asked to specify a set $SU_j \subseteq SSet_j$ of services that he considered interesting for him.

The Precision of our system (P_j) and that of the expert human administrator (P'_j) have been computed as:

$$P_j = \frac{\mid SU_j \cap SS_j \mid}{\mid SS_j \mid} \qquad P'_j = \frac{\mid SU_j \cap SE_j \mid}{\mid SE_j \mid}$$

Analogously, the Recall of our system (R_j) and that of the expert human administrator (R'_j) have been computed as:

$$R_j = \frac{\mid SU_j \cap SS_j \mid}{\mid SU_j \mid} \qquad R'_j = \frac{\mid SU_j \cap SE_j \mid}{\mid SU_j \mid}$$

In these tests the initial audacity and the initial selectivity degree of our system have been set to 0.50 and 0.45, respectively, since previous experiments showed that these values allow the best system performance to be obtained.

Figure 18 (resp., Fig. 19) plots the values of P and P' (resp., R and R'), averaged on all involved users, against the number of submitted queries.

From the analysis of these figures we can observe that:

- Both P' and R' are almost constant against the number of submitted queries; on the contrary, P and R increase when the number of submitted queries increases. This behaviour is explained by the fact that the expert human administrator bases his recommendations much more on his experience than on user past behaviour (that he, generally, does not register in a systematic way). On

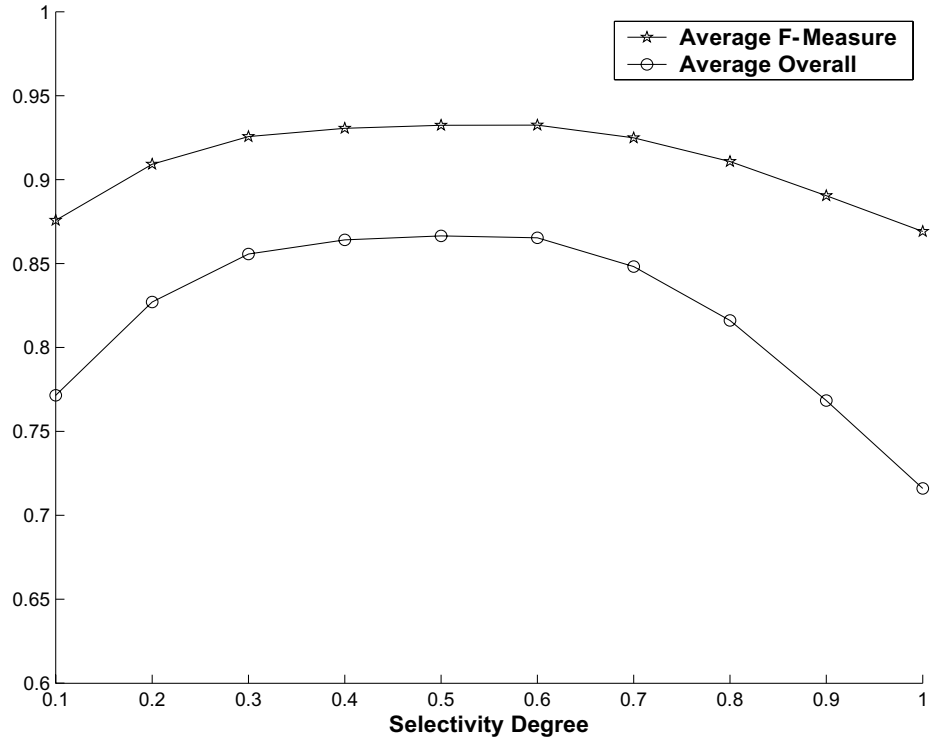

Fig. 16. Variation of F-Measure and Overall against variation of the selectivity degree.

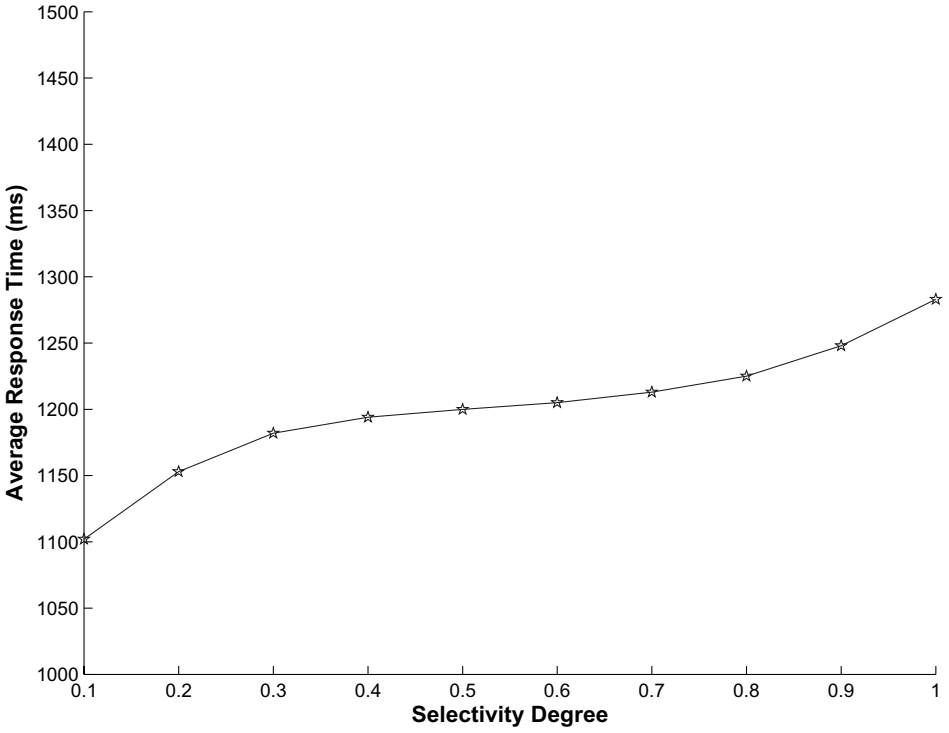

Fig. 17. Variation of Response Time against variation of the selectivity degree.

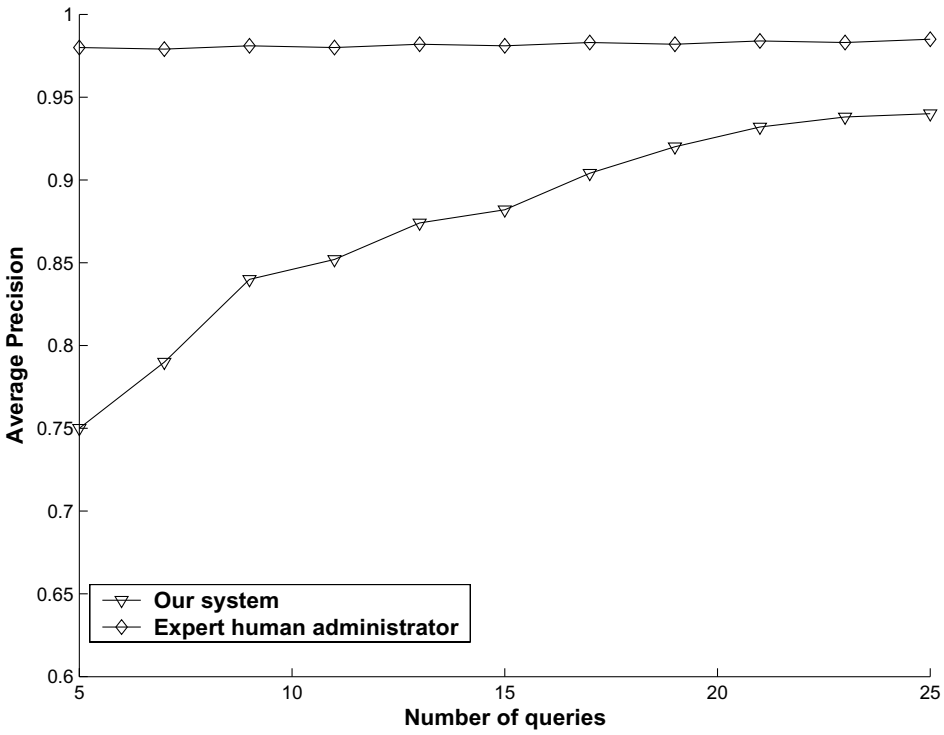

Fig. 18. Variation of Average P and P' against the number of submitted queries.

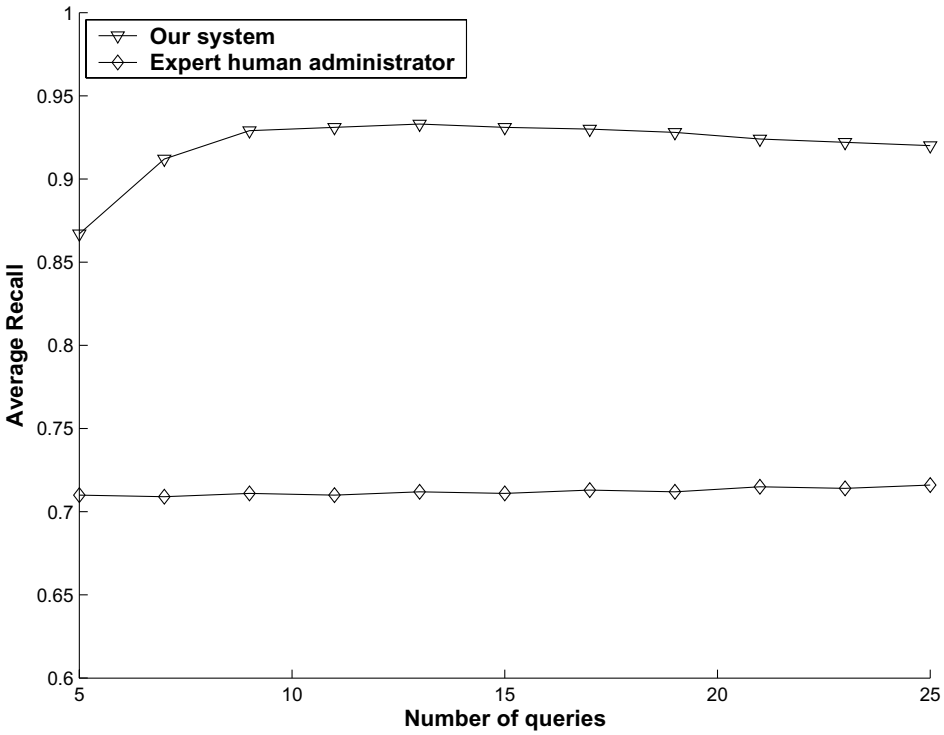

Fig. 19. Variation of Average R and R' against the number of submitted queries.

the contrary, in our system, user past behaviour plays a key role; in fact, our system refines the profile of each user each time he submits a query and is capable of adapting its recommendations to the user's profile, which stores, among other information, his past behaviour.

– The Precision obtained by the expert is always higher than that reached by our system, although the difference between them significantly decreases when the number of submitted queries becomes high. This result can be explained by the fact that a human expert, due to his experience, can interpret more precisely what is the intended meaning of a submitted query; as a consequence, he can provide a more precise selection of the services corresponding to it. Our system, instead, utilizes automated techniques that are intrinsically less precise; however, when the number of submitted queries increases, he refines user profile and, consequently, acquires "experience" on user needs and preferences.
– The Recall obtained by our system is always higher than that obtained by the human expert. This trend has a precise explanation. In fact, it is well known that one of the main shortcomings of manual approaches is that they are error prone when the amount of information to be handled becomes large. In our context, since, generally, the number of services to be examined is high, it is plausible that a human (even if expert) may fail to propose some relevant services. This reasoning explains the lower values of Recall obtained by the expert compared to those returned by our system. As seen in Section 3.2, the initial increase of our system's Recall is due to the refinement of user profile and to the dynamic adaptation of system audacity.

Summarizing, expert human administrators can obtain more precise results, although, when the number of involved services is large, they may miss some relevant answers. On the contrary, our system can be slightly less precise than a human expert, although this gap: (*i*) is counterbalanced by a higher completeness and less response time; (*ii*) decreases as long as our system refines user profiles.

3.6. *Analysis of the computational costs necessary for making our system user-adaptive*

This experiment is devoted to analyze the computational costs necessary for making our system user-adaptive.

Recall that, for a query Q_{ij}, our system first applies classical, well assessed, Information Retrieval techniques for extracting, from the Service Database, those services satisfying Q_{ij} (in the following we call t_1 this task; it corresponds to Step 1 described in Section 2.4.2). After this, it constructs an ordered list of services satisfying a user's query, taking his past behaviour into account and presumably matching his future interests (in the following we call t_2 this task; it corresponds to Steps 2, 3 and 4 described in Section 2.4.2).

We have run several queries for measuring the cost of t_2 compared to that of t_1 and we have found that the average time necessary for executing t_2 is equal to 17.95% of the average time necessary for executing t_1. In other words, it emerges that the important characteristic of our system to adapt the results returned by classical Information Retrieval techniques to user's needs, past behaviour and possible future interests does not require a significant computational effort.

In order to carry out a more detailed analysis, we have studied the variation of the execution time of t_1 and t_2 when the number of keywords present in a user query increases. Figures 20 and 21 show obtained results. From their analysis we can observe that:

– Differently from task t_1, the execution time of task t_2 slightly decreases when the number of keywords present in a query increases. This trend can be explained by the fact that, when the number of keywords in a query increases, t_1 is more selective (i.e., it returns a smaller set of services); as a consequence, t_2 works on a smaller set of data and its execution time decreases.
– For each value of the number of keywords present in a query, the execution time of t_2 is significantly lower than that associated with t_1.

Another important factor that could negatively influence the performance of our system is the correlation possibly existing among the keywords of a query. Specifically, [20] states that a low correlation among the keywords of a query can lead to higher Response Times. We have investigated if our system follows this general trend. Specifically, Figs 22 and 23 plot the average execution time associated with t_1 and t_2, in presence of a low or a high keyword correlation, for various numbers of keywords present in submitted queries[4]. From the analysis of this figure we can see that keyword

[4] In this experiment the correlation degree of the various keywords of a query has been directly specified by users.

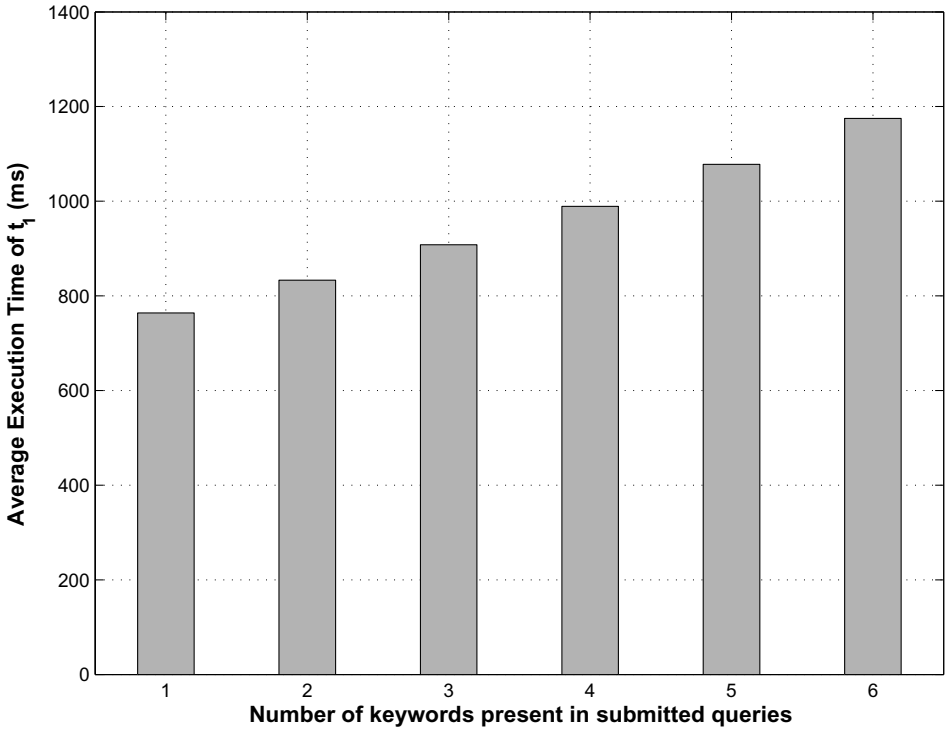

Fig. 20. Response Time of task t_1 against the number of keywords present in a query.

correlation actually influences the execution time of t_1; specifically, the higher the keyword correlation is, the lower the execution time will be; this trend confirms the observations of [20]. As for t_2, our experiments did not find keyword correlation affected its execution time.

3.7. Answer delay in overload conditions

Our system might be simultaneously accessed by several users and each of them might submit a different query. For each query, the corresponding User-Device Agent contacts the Service Recommender Agent, which provides it with the list of services answering the query and best matching user profile.

If the number of users simultaneously accessing our system is huge, the Service Recommender Agent might be overwhelmed by an enormous number of requests; this would lead to the congestion of both the Service Database and the network; in this case the time necessary for answering user queries might become excessive.

In order to verify the behaviour of our system in the case of several simultaneous requests, we have conducted an experimental study to measure the query answer delay when the number of users concurrently accessing our system increases. We define "answer delay" as the extra time necessary for answering a query Q_{ij}, when it is submitted simultaneously with other queries, as opposed to the case in which it is the only query to be processed by the system.

In order to carry out this test, a huge number of users should have simultaneously submitted queries using our system; however, since testing our system with thousands of human users would have been extremely difficult, we have implemented a software module simulating the querying activity of a user. Specifically, each instance of this module simulates a user submitting a query to our system and measures the time elapsed between the query submission and the corresponding answer reception.

We have run several benchmark tests; for each of them we have activated an increasing number of concurrent simulated users and we have measured the average answer delay of our system; obtained results are shown in Fig. 24. From the analysis of this figure we can observe that:

– For a low number of users (i.e., less than 1000), the average answer delay is almost null and the

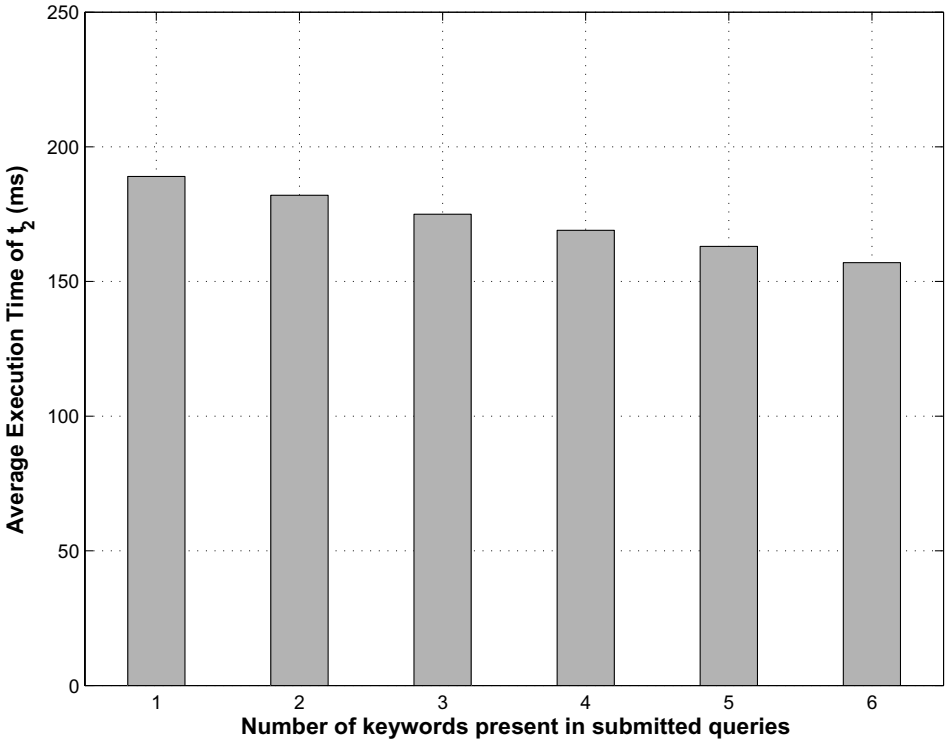

Fig. 21. Response Time of task t_2 against the number of keywords present in a query.

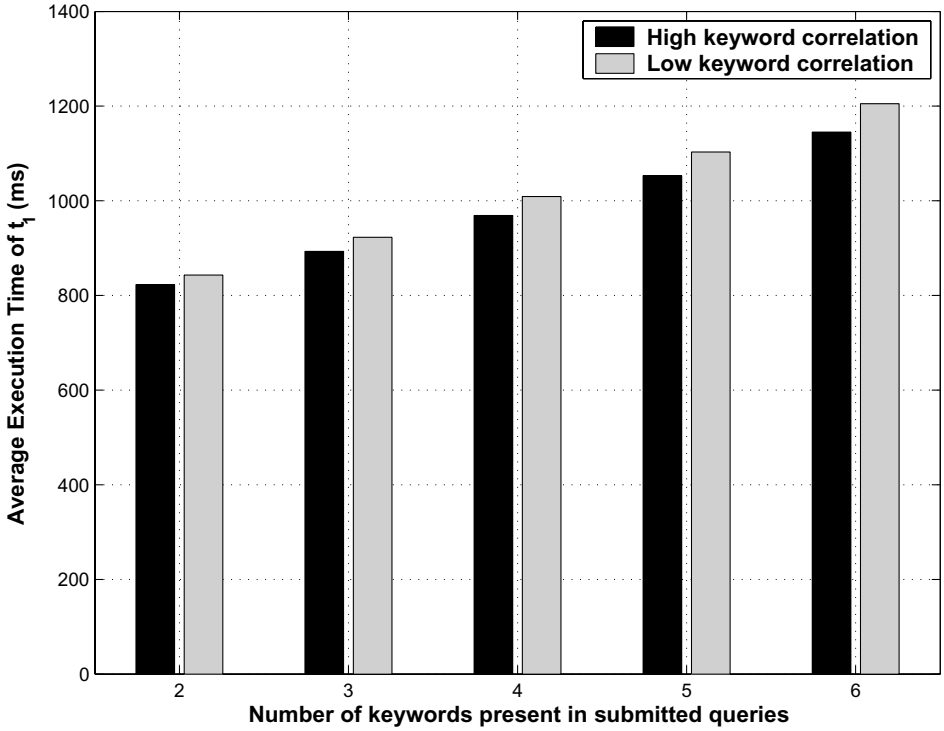

Fig. 22. Role of keyword correlation in the execution time of task t_1.

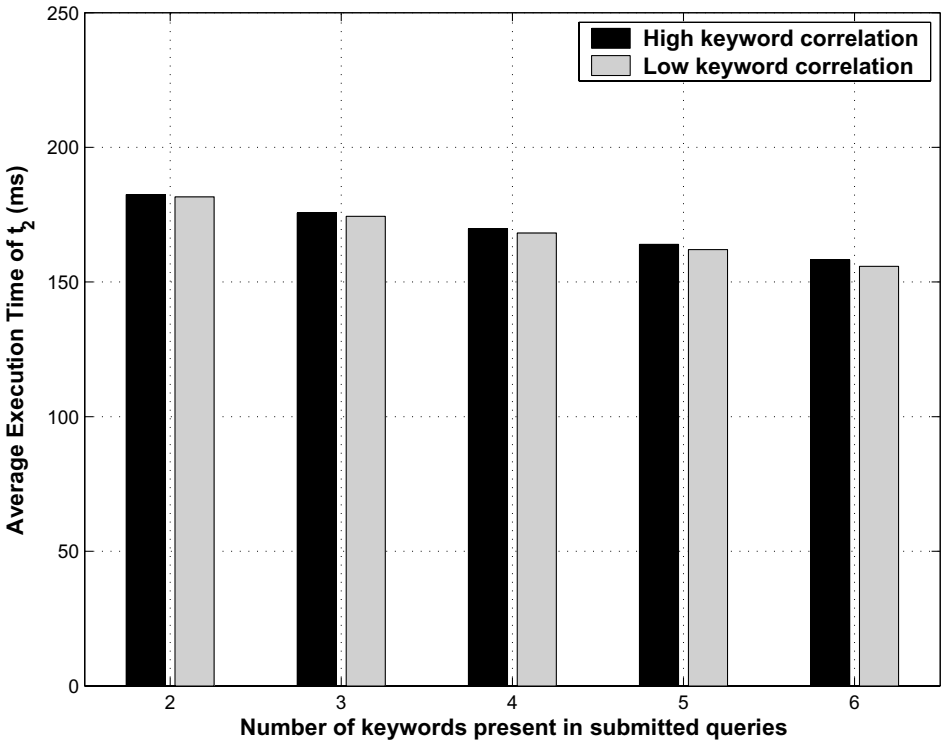

Fig. 23. Role of keyword correlation in the execution time of task t_2.

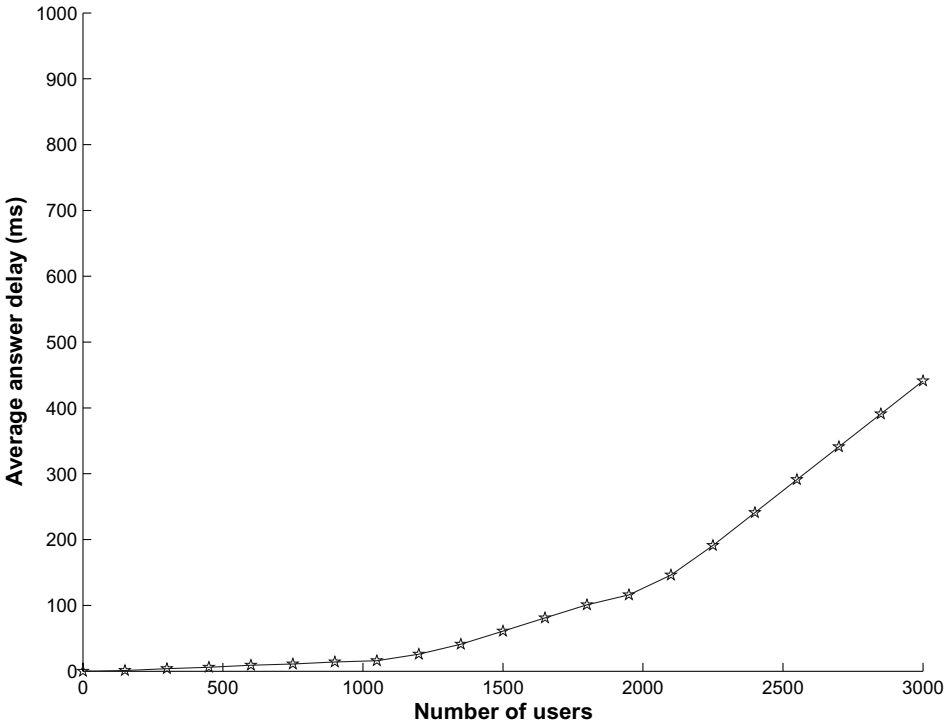

Fig. 24. Average answer delay against the number of simulated users accessing our system.

Table 1
Average Precision and Average Recall obtained by analyzed systems after 5, 15 and 25 queries.

System	After 5 queries		After 15 queries*		After 25 queries	
	Average Precision	Average Recall	Average Precision	Average Recall	Average Precision	Average Recall
Our system	0.70	0.85	0.87	0.91	0.94	0.90
Italy	0.50	0.68	0.73	0.60	0.83	0.79
United Kingdom	0.52	0.56	0.77	0.74	0.81	0.72
France	0.54	0.64	0.75	0.73	0.82	0.77
Germany	0.53	0.61	0.72	0.71	0.82	0.76
United States	0.54	0.58	0.71	0.66	0.79	0.72

Response Time of our system is independent of the number of users accessing it.
- For an average number of users (i.e., between 1000 and 2000), an increase of the number of users accessing our system causes a slight increase in the corresponding average answer delay.
- For a high number of users (i.e., more than 2000), the average answer delay of our system increases linearly with the increase of the number of users accessing it.

These results can be explained by the fact that:

- The various agents involved in our system exchange a very small number of, generally simple, messages. As a consequence, the network traffic caused by message exchange is very limited. This allows a significant reduction of the risk of network congestion.
- As pointed out in Section 2.6, our system adopts a Database Replication strategy [34] to efficiently handle possible failures of the Service Database. This also allows computational load to be spread across more than one server and the processing capability of our system to be increased.

3.8. Experimental comparison of our system with other e-government systems

In this experiment we have compared the accuracy of our system with that of other, already existing, e-government systems. Specifically, we have examined the official e-government systems of: (*i*) *Italy* – www.italia.gov.it; (*ii*) *United Kingdom* – www.direct.gov.uk/Homepage/fs/en; (*iii*) *France* – www.service-public.fr; (*iv*) *Germany* – www.bund.de; (*v*) *United States* – www.firstgov.gov.

In order to carry out our comparisons we have asked users of $USet$ to submit some queries; submitted queries were very heterogeneous, ranging from immediate and simple queries to complex and sophisticated ones. Queries have been processed by each system under examination[5]; obtained results have been utilized for computing Precision and Recall.

The values of Precision and Recall, averaged on all users of $USet$, are reported in Table 1. From the analysis of this table, it is possible to deduce that our system obtains extremely satisfying results; these are explained by the following reasoning:

- With regard to the identification of the most relevant services to recommend, our system utilizes information concerning both user and device profiles; on the contrary, the other systems consider only the keywords specified by users. Profile utilization makes our system more sensitive to user needs and preferences; this feature justifies the improvement of Precision.
- Our system recommends to a user not only those services exactly corresponding to keywords specified in his query but also ones that he disregarded in the past (for example because he did not know of their existence) but that might be of interest to him in the future. As a consequence, it is capable of identifying some services potentially interesting for the user that the other systems generally filter out; this feature explains the improvement of Recall.

4. Related work

In the literature, several systems supporting users in their access to e-government services have been proposed. In this section we compare some of these systems with that presented in this paper.

In [17] the system PASSPORT, aiming at improving the interaction between citizens/companies and Public Administration offices, is proposed. PASSPORT

[5]Taking the analysis presented in Section 3.2 into account, the initial value of the audacity coefficient of our system has been set to 0.50.

consists of three types of software components, namely: (*i*) *service providers*, each associated with a Public Administration office; (*ii*) *end users*, each associated with a citizen/company; (*iii*) the *intermediary hub*, that collects user requests, performs some evaluations and selects those service providers capable of satisfying them.

The main similarities existing between our system and PASSPORT are the following: (*i*) both of them allow services to be added/removed in a very simple way; (*ii*) both of them are *transparent*, i.e., provide mechanisms to hide the complexity of the interactions between Public Administration offices and users. Specifically, PASSPORT achieves this simplification with the support of the intermediary hub, whereas our system realizes it by automatically handling the cooperation among involved agents.

The main differences between our system and PASSPORT are the following: (*i*) PASSPORT is characterized by a centralized architecture, not based on agents, whereas our system is based on a distributed multi-agent platform; (*ii*) in our system both user and device profiles play a key role in service selection; this does not happen in PASSPORT.

In [23] the Authors propose a software architecture for delivering e-government services. The main elements of this architecture are: (*i*) a component devoted to collect and evaluate user requests; (*ii*) a component handling the information systems supporting Public Administration offices; (*iii*) a component identifying those services best satisfying user requirements and scheduling the activities necessary to deliver them.

It is possible to find some similarities between our system and that described in [23]. Specifically: (*i*) both of them define mechanisms for service selection, aiming at satisfying the present needs of a user as well as at identifying services potentially of interest to him in the future; (*ii*) both of them provide a user with the capability of utilizing heterogeneous kinds of devices for accessing e-government services.

The main differences existing between our system and that described in [23] are the following: (*i*) in the system proposed in [23] services are selected by means of a *workflow analysis*, whereas our system utilizes user profiles and past user feedbacks; (*ii*) the software architecture presented in [23] is centralized; it does not utilize the Intelligent Agent technology and is multi-layered; on the contrary, our system is multi-agent and, consequently, it is distributed and utilizes most of the features characterizing the Intelligent Agent technology.

In [32] the WebDG (*Web Digital Government*) system, conceived to simplify citizens access to services of local Public Administration, yet preserving their privacy, is presented. The architecture of WebDG consists of three software components, namely: (*i*) *providers*, each associated with a Public Administration office; (*ii*) *consumers*, each conceived for supporting citizens; (*iii*) *registry*, that retrieves, from providers, the most interesting services for consumers.

The main similarities existing between our system and WebDG are the following: (*i*) both of them utilize user profiles; however, in WebDG, these are utilized only for guaranteeing user privacy, whereas, in our system, they allow service selection to be personalized; (*ii*) both of them utilize a conceptual model for service representation; specifically, our system adopts an XML based model, whereas WebDG utilizes the Web service paradigm.

The main differences between our system and WebDG are the following: (*i*) WebDG is characterized by a centralized architecture, not based on the Intelligent Agent technology, whereas our system is multi-agent; (*ii*) WebDG provides mechanisms allowing both the composition and the outsourcing of available services; such a feature is not provided by our system; (*iii*) for service suggestion, our system considers the characteristics of the device currently utilized by a user; such a feature is not included in WebDG.

In [28] the Authors propose a system for simplifying user access to an e-government portal. This system consists of the following software elements: (*i*) a *database*, storing services provided by a Public Administration office; these are represented by means of the Web services paradigm; (*ii*) a component, called *CLIPS* (C Language Integrated Production System), which is in charge of identifying the most interesting services for users.

Some similarities exist between our system and that described in [28]. In fact: (*i*) both of them provide mechanisms to represent available services; specifically, the system proposed in [28] utilizes a suitable ontology, whereas our system adopts a suitable XML Schema; (*ii*) both of them provide a software component supporting Public Administration offices in the management of services delivered by them.

The main differences existing between our system and that proposed in [28] are the following: (*i*) our system is multi-agent and completely distributed, whereas the system presented in [28] has a centralized architecture consisting of several interacting software "layers"; (*ii*) the approach described in [28] handles service

modularity, i.e., it is capable of segmenting a complex service in several simple sub-services and of assigning each simple sub-service to a public/private administration for its execution; this feature, not included in our system, can be profitably utilized for service outsourcing.

In [16] the Authors describe a system helping users to access services provided by Japanese railways. This system assumes that each user is provided with a device storing his characteristics (e.g., his personal data, his possible handicaps, and so on); this device can be utilized by him for contacting a server to construct an itinerary between two cities. The server elaborates received requests and generates a set of routes best satisfying user needs. The user can choose one of these routes and, possibly, can buy tickets.

Some similarities exist between our system and that presented in [16]; specifically, both of them: (*i*) utilize a user profile; (*ii*) are based on a distributed architecture capable of interacting with heterogeneous devices.

The main difference existing between our system and that described in [16] is that the latter has been conceived for operating with public railway systems; as a consequence, some of the design/implementation choices have been specifically made for this application context and are difficult to extend to other domains. As for another, important, difference between the two systems, we observe that the approach of [16] does not utilize the Intelligent Agent technology.

In [36] an approach for assisting citizens to access e-government services is proposed. It relies on *dynamic taxonomies*. These are taxonomies designed by domain experts; their components are called *concepts*. A concept is a label that identifies a set of instances (*items*). Concepts are linked together through semantic connections stating *subsumptions* relationships. When a user submits a query the system identifies a subset of concepts (along with their descendants) satisfying it; this set of concepts is called *focus*. After this, it presents to the user the set of services corresponding to the concepts of the focus; the user may accept or reject this set. In the former case the recommending activity successfully ends; in the latter one the system refines the focus (*zooming activity*) by discarding some of the corresponding concepts. Zooming activity continues until a suitable set of services of interest to the user has been identified. This service retrieval activity can be, therefore, depicted as an iterative thinning of a dynamic taxonomy.

We can highlight some similarities between our approach and that presented in [36]. Specifically: (*i*) both of them are capable of finding correlations existing among e-government services; for this purpose our approach utilizes a distance function whereas the approach of [36] breaks a compound service into simple elements; (*ii*) in both of them service retrieval requires intelligently inspecting a database of services; for this purpose, our approach utilizes audacity coefficient; conversely, the approach of [36] carries out several zooming operations on a dynamic taxonomy.

As for the main differences between the two approaches, we observe that: (*i*) the approach of [36] does not take device profiles into account; (*ii*) in order to find relevant services for citizens, our approach utilizes user profiles; conversely, the approach of [36] requires citizen collaboration; for this purpose, it adopts an interactive tool.

In [43] the *eip.at* project, aiming at supporting heterogeneous users accessing e-government services, is presented; involved users might be citizens, Public Administration offices, businesses, and so on. In *eip.at*, services delivered by a Public Administration office are described by means of a suitable ontology called *knowledge map*. This ontology comprises various *classes*, each representing a real life aspect (e.g., asylum/immigration, cultural heritage, and so on). Each class contains various *instances*. Classes can be related to each other by means of several kinds of relationships (e.g., *is-a* relationships). Each time a user submits a query, *eip.at* splits it into its keywords; after this, it locates, on the knowledge map, those classes matching keywords. If a class exactly matches one (or more) keywords, its instances are included in the result. If no class exactly matches at least one keyword, the original query is replaced by an *extension* of it, obtained by relaxing or dropping some constraints.

As for the main similarities between our system and *eip.at*, we can observe that: (*i*) both of them provide a formalism for describing services; in *eip.at* it relies on ontologies, whereas in our system it consists of XML-based profiles; (*ii*) both of them consider different types of users accessing e-government services; specifically, our system handles both citizens and Public Administration offices whereas *eip.at* can support citizens, Public Administration offices, private businesses, and so forth.

As for the main differences between the two systems, we can observe that: (*i*) in *eip.at* service recommendation is carried out by utilizing ontologies and ontology-based query languages (e.g., RQL); on the contrary, for performing the same activity, our system utilizes both user and device profiles; (*ii*) *eip.at* does not consider the

features of the devices utilized by users for accessing e-government services.

In [21] the system *STEF* (Smart Trade Exhibition Finder) is described. *STEF* has been conceived for managing trade exhibitions, i.e., international trade fairs in which businesses can advertise their products or services. *STEF* supports companies to find the right trade exhibitions for them (and, consequently, it improves their product export while reducing the time, costs and risks associated with their entry in an international market). It collects information about business preferences, behaviours and activities, as well as information about past trade exhibitions. This information is utilized by a Recommendation Engine that separately applies a collaborative filtering algorithm and a content-based one and, then, merges obtained results for generating recommendations.

We can recognize some similarities between our system and *STEF*. Specifically, both of them: (*i*) learn and manage rich user profiles for producing their recommendations; (*ii*) are provided with a methodology for computing the similarity degree of two services.

As for the main differences between the two systems, we can observe that: (*i*) *STEF* has been conceived for managing a specific class of e-government users (i.e., private businesses) whereas our system is devoted to handle citizens and Public Administration offices; (*ii*) *STEF* does not manage device profiles.

In [14] *ServiceFinder*, a system conceived for supporting citizens in their selection of relevant e-government services, is presented. In order to carry out its activity, *ServiceFinder* uses Web Mining techniques; specifically, given an e-government portal, it examines Web logs recording users' behaviour on accessing it and mines the corresponding data to discover the N services best matching users' needs in the past. After this, it modifies the portal home page in such a way as to specify N hyperlinks, one for each service classified as interesting for users.

ServiceFinder and our system share some similarities. Specifically, both of them: (*i*) solve the *e-service selection* problem, i.e., select a small number of relevant services for citizens and, at the same time, discard a great number of irrelevant ones; (*ii*) unobtrusively monitor user behaviour for producing their recommendations; specifically, our approach analyzes user queries whereas *ServiceFinder* focuses on patterns extracted from Web logs.

As for the main differences between the two systems we observe that: (*i*) *ServiceFinder* does not associate a profile with a user; in fact, it has been conceived for analyzing the *joint behaviour* of a group of users accessing an e-government portal; (*ii*) *ServiceFinder* does not consider the characteristics of the devices adopted by users for accessing the e-government portal associated with it.

In [13] the Authors propose a system for disseminating news coming from Web sites associated with newspapers. In this system a profile is associated with each user; it consists of two main sections describing long-term and short-term interests, respectively. This information is used to rank available news. The news having the highest ranks are, then, presented to users.

Even though the system of [13] does not focus on e-government, it shares some similarities with our system. Specifically, both of them take user feedbacks into account for ranking "information items" (i.e., services in our approach and news in the approach of [13]); in addition, both of them provide a mechanism for representing the main features of an information item.

As a final remark about related literature, we point out that, in the past, we considered the problem of defining multi-agent systems for personalizing information content delivery in various application contexts, such as e-commerce [12] and telecommunications networks [11].

While, in these application scenarios, service delivery is based on quite standard protocols, well accepted worldwide (think, for example, of e-commerce protocols), the management and the distribution of e-government services are still characterized by a high heterogeneity, since the corresponding rules and protocols may be very different in the various countries.

This implies an intrinsic difficulty to develop e-government applications sufficiently precise to fully satisfy citizen needs and, at the same time, sufficiently generic to be applied in different countries. As a proof of these difficulties, consider the very limited number of multi-agent systems proposed for the e-government application context in the past.

One of the main contributions of this paper consists of the development of a multi-agent system capable of both addressing the heterogeneity of e-government applications and guaranteeing a sufficiently high accuracy of results (see Section 3). This important feature has been obtained by means of a careful definition and a careful utilization of both service and citizen profiles.

In addition, the system we are presenting in this paper appears to be more flexible than our previous ones in the management of user preferences and behaviour. Such an important feature has been obtained by introducing the novel concept of system audacity; this prop-

erty is directly related to citizen satisfaction for system answers and allows our system to dynamically adapt its behaviour to citizen needs.

Last, but not least, in this paper we have introduced a clear formalization of service requisites; in this way it is possible to formally and unambiguously specify which users can access which services. In addition, as it will be clear in Section 5.3, service requisite formalization allows our system not only to clearly and precisely control service access but also to adapt itself to past citizen behaviour.

5. Future work

In this section we have a look at the future of the approach proposed in this paper; specifically, we first present an overview of three possible enhancements of our system and, then, we glance over some further ideas that we plan to analyze and develop in the future.

5.1. Extension towards semantics

The mechanism of deriving the most appropriate e-government services, described in this paper, might be extended in such a way as to include semantics-driven tools, like ontologies.

An ontology can be considered as a taxonomy containing the concepts interesting for an application domain; the concepts stored on the top of an ontology have a generic meaning, whereas those stored at the bottom are more specific. If a concept A is more generic (resp., specific) than a concept B, we say that A is a *super-concept* (resp., a *sub-concept*) of B. Some ontologies defined in the e-government application domain are presented in [15,38].

We argue that the utilization of ontologies might be highly beneficial for both Public Administration officers and citizens.

From the Public Administration officer standpoint, ontologies could support the process of planning, realizing, delivering and, possibly, modifying services. Specifically, when an officer must plan the realization of a new service or the modification of an existing one, he must have a high-level view of the services delivered online, of the relationships existing among them, as well as of the needs that they aim at solving and the relationships existing among these needs. An ontology can provide the officer with this high-level view because it can represent all information mentioned above in a clear, precise and systematic fashion.

The integration of our system with ontologies can be beneficial also for citizens. In fact, an ontology can effectively help a citizen to submit queries that precisely specify his needs and, consequently, can improve our system's capability of proposing the most relevant services for him.

To understand this aspect better, consider that Public Administration offices might use different terminologies for naming and describing the services delivered by them. This heterogeneity makes the detection of desired services, as well as their comparison, quite a difficult task. As a consequence, queries formulated by citizens might return too few or too many results.

If a query returns too few results, it is necessary to relax it. This task can be performed by replacing some of its terms with more general ones; for this purpose, the ontology can be traversed in such a way as to detect one or more super-concepts for each term to be substituted. If a query returns too many results, it is necessary to restrict it. This task can be carried out by replacing some of its terms with more specific ones; for this purpose the ontology can be traversed for detecting the most appropriate sub-concepts.

Clearly, ontology utilization for deriving super-concepts and sub-concepts should be performed by the user by means of a friendly graphical interface, handled by the User-Device Interface Agent.

5.2. Integration with collaborative filtering techniques

From the Recommender Systems point of view, our system can be considered as a content-based one; a future research effort might be devoted to study the possibility to include also collaborative filtering algorithms in it.

In this context, observations presented in [42], where the Authors show how collaborative filtering algorithms can positively benefit from user personal data (like age, gender and educational background), appear to be extremely interesting.

In this section we sketch a collaborative filtering algorithm that, in the future, could be carefully analyzed and, possibly, integrated in our system; it extends the core ideas of the user-based collaborative filtering algorithm described in [35] in such a way as to consider also personal data of involved citizens.

The proposed algorithm considers m citizens and n services; it receives a pair $\langle C_a, S_b \rangle$, where C_a is a citizen and S_b is a service, and predicts the rating that C_a will assign to S_b; the higher this rating is, the more relevant S_b will be for C_a.

The algorithm consists of the following steps:

– *Step 1*: *construction of the Rating Matrix* R. R is a $m \times n$ matrix whose generic element R_{ij} belongs to the real interval $[0, 1]$ and specifies the rating that the citizen C_i assigned to the service S_j in the past. It is worth pointing out that some elements of R could be missing.
– *Step 2*: *construction of the Past Behaviour Matrix* PB. PB is a $m \times m$ matrix whose element PB_{hk} belongs to the real interval $[0, 1]$ and specifies the similarity degree of the past behaviours of the citizens C_h and C_k. The computation of PB is based on the knowledge of the past ratings specified by C_h and C_k and stored in R.
– *Step 3*: *construction of the Personal Data Matrix* PD. PD is a $m \times m$ matrix whose generic element PD_{hk} belongs to the real interval $[0, 1]$ and specifies the similarity degree of the personal data of the citizens C_h and C_k.
– *Step 4*: *construction of the Citizen Affinity Matrix* CA. CA is a $m \times m$ matrix whose generic element CA_{hk} belongs to the real interval $[0, 1]$ and specifies the overall similarity degree between C_h and C_k. CA_{hk} can be computed as:

$$CA_{hk} = PB_{hk} \times PD_{hk}$$

– *Step 5*: *citizen partitioning*. A clustering algorithm is applied to CA in such a way as to partition involved citizens into homogeneous clusters Cl_1, Cl_2, ..., Cl_p. At the end of this step each citizen will belong to only one cluster.
– *Step 6*: *rating prediction*. The algorithm is, now, able to predict ratings assigned to services by citizens. For rating computation it is possible to adapt, to our application context, the formula proposed in [35]. Specifically, let C_a be a citizen and let S_b be a service; let Cl_a be the cluster of C_a and let $\overline{r}_b$ be the average rating assigned by the citizens of Cl_a to S_b; finally, let $\overline{r}_a$ be the current average rating for C_a. The predicted rating R_{ab}, assigned by C_a to S_b, can be computed as:

$$R_{ab} = \begin{cases} \max\left(0, \overline{r}_a + \dfrac{\sum_{C_i \in Cl_a} (R_{ib} - \overline{r}_b) \cdot CA_{ai}}{\sum_{C_i \in Cl_a} CA_{ai}}\right) \\ \quad \text{if } \overline{r}_a + \dfrac{\sum_{C_i \in Cl_a} (R_{ib} - \overline{r}_b) \cdot CA_{ai}}{\sum_{C_i \in Cl_a} CA_{ai}} \leqslant 0 \\ \min\left(1, \overline{r}_a + \dfrac{\sum_{C_i \in Cl_a} (R_{ib} - \overline{r}_b) \cdot CA_{ai}}{\sum_{C_i \in Cl_a} CA_{ai}}\right) \\ \quad \text{if } \overline{r}_a + \dfrac{\sum_{C_i \in Cl_a} (R_{ib} - \overline{r}_b) \cdot CA_{ai}}{\sum_{C_i \in Cl_a} CA_{ai}} > 0 \end{cases}$$

5.3. Enhancement of requisite management policy

In our system each service can be associated with a set of requisites that a citizen must satisfy for gaining access to it. These requisites are represented as a set of conditions; a user can access a service only if he can satisfy *all* conditions associated with it.

In the current version of our system the set of conditions can be seen as a conjunction of comparisons. A possible enhancement for requisite management might consist in allowing more complex and refined formulas for composing the various comparisons; as an example, the various comparisons might be connected by AND, OR and NOT operators.

However, we point out that the requisite management policy currently adopted by our system, if coupled with a careful and complete definition of the requisites associated with each service, already allows quite a fruitful utilization of the rich information stored in citizen profiles; in its turn, this allows very interesting situations to be faced.

As an example of these situations, consider a citizen interested to know how to get a new passport. Assume that he accesses the service "passport issue" (hereafter denoted by S_p), allowing him to apply for a new passport; assume, also, that he eventually obtained his passport.

After this, assume that the same citizen continues to access our system for asking information about "passport usage". Now, without an accurate definition of service requisites, our system would propose to him, among others, also the service S_p, although he will be no longer interested in it for, at least, the next five years.

On the contrary, if S_p is associated with the requisite $\langle$ passport expiry date, $\leqslant$, \$today $\rangle$ (here "\$today" stands for a dynamic parameter returning the current date), when the citizen asks for "passport usage", S_p is filtered out until his passport needs to be issued again.

As a final remark about this argument, note that the personal and economic data of a citizen are stored in his profile (see Section 2.2.1); in our application context (i.e., an e-government scenario) much of this data might be updated by our system without citizen intervention, since Public Administration offices are entitled to access a wide variety of information about citizens, including that concerning passport existence and validity.

5.4. A glance to some further ideas

In the previous sections we have provided quite a detailed overview of three possible future developments

of our system. In this section we glance over some further ideas that, presently, we have only touched on but that we plan to analyze carefully in the future. Specifically:

- It would be interesting to add new functionalities to our system in such a way as to make it capable of helping not only citizens but also Public Administration offices with their decision making. As an example, our system could help the manager of a Public Administration office to decide, on the basis of citizens preferences, needs and past behaviour, if it is necessary to suspend the delivery of some existing services and/or to propose new ones.
- Another, presumably interesting, research direction could consider the possibility to define more sophisticated profiles of citizens, storing not only their preferences/needs but also the quality of a recommended service as it has been perceived by them during their access. Such an information could play a key role in the evaluation of the overall quality of the set of services delivered by a Public Administration office.
- In order to speed up query processing activities, it would be interesting to adopt, in our system architecture, a pool of SRAs, that cooperatively handle user queries, instead of only one SRA. These SRAs would share some computational resources, like CPU or storage devices; as a consequence, *ad-hoc* mechanisms, relying on *concurrency management*, should be used for handling common resources. For this purpose, it would be interesting to define an approach capable of merging the approach for coordinating resource management in a multi-agent system, proposed in [27], with the approach for planning/replanning, outlined in [40].
- It would be interesting to study how to provide our system with a high level of security and prevention from privacy attacks. This requires some specific software modules to be implemented in our User-Device Agent. These modules might inherit, for instance, some of the ideas outlined in [26], where an encryption scheme conceived for mobile settings and based on homomorphic functions is proposed.
- It would be interesting to verify the possibility of defining a more refined formalism for representing user interests. In the current version of our system, these interests are represented by means of keywords in the corresponding User Profile. In the future we plan to verify the possibility of applying, in our system, some of the ideas outlined in [45], where a theoretical framework for representing and identifying the information needs of a team of agents is proposed. As shown in [45], this framework is particularly suited for gathering information in a multi-agent setting, as well as for supporting decision making activities.

6. Conclusions

In this paper we have presented a multi-agent system supporting citizens in their selection of services delivered by Public Administration offices. The proposed system identifies and suggests the most interesting services for a citizen; in order to achieve this, it considers both his profile and the characteristics of the devices utilized by him.

In our opinion, our system represents an interesting attempt to apply the Intelligent Agent technology in the e-government context. We argue that this technology, which has already proved to be successful in other application domains, such as e-commerce and e-learning, can significantly improve the quality of the interaction between citizens and Public Administration offices.

Acknowledgments

The authors thank Danilo De Benedetto for his contribution to the implementation of the proposed approach, and Diane Selvanayagam for helping them with the improvement of their English in the paper.

References

[1] Document Object Model. http://www.w3.org/DOM/.
[2] e-care cup 2000. http://www.cup2000.it/cup2000/eng/cup2000.asp.
[3] Foundation for Intelligent Physical Agents (FIPA) Specifications. http://www.fipa.org.
[4] World Wide Web Consortium – XML query. http://www.w3.org/XML/Query.
[5] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley Longman, 1999.
[6] A. Bivens, R. Gupta, I. McLean, B. Szymanski and J. White, Scalability and performance of an agent-based network management middleware, *International Journal of Network Management* **14**(2) (2004), 131–146.
[7] A. Cesta and D. D'Aloisi, Building interfaces as personal agents: a case study, *ACM SIGCHI Bullettin* **28**(3) (1996), 108–113.

[8] Accenture Consulting. e-Government Leadership Rhetoric vs. Reality"Closing the Gap. Technical report, Available at www.accenture.com/xdoc/en/industries/government/final.pdf, 2001.

[9] E. Cortese, F. Quarta and G. Vitaglione, Scalability and performance of JADE message transport system. Technical report, Available at jade.tilab.com/papers/Final-ScalPerfMessJADE.pdf, 2002.

[10] P. De Meo, A. Garro, G. Terracina and D. Ursino, *X-Learn: an XML-based, multi-agent system for supporting "user-device" adaptive e-learning,* in Proc. of the International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE 2003), pages 739–756. Taormina, Italy, 2003. Lecture Notes in Computer Science, Springer.

[11] P. De Meo, J. Mbale, G. Terracina and D. Ursino, XICOMASQ: an XML-based Information Content Oriented Multi-Agent System for QoS management in telecommunications networks, *Web Intelligence and Agent Systems Journal* **2**(1) (2004), 55–70.

[12] P. De Meo, D. Rosaci, G.M.L. Sarnè, G. Terracina and D. Ursino, *EC-XAMAS: supporting e-commerce activities by an XML-based adaptive multi-agent system*, Intelligenza Artificiale, Forthcoming.

[13] A. Diaz and P. Gervas, Personalisation in News Delivery Systems: Item Summarization and Multi-Tier Item Selection Using Relevance Feedback, *Web Intelligence and Agent Systems: An International Journal* **3**(3) (2005), 135–154.

[14] X. Fang and O.R. Liu Sheng, *Designing a better Web portal for digital government: a Web-mining based approach,* in Proc. of the National Conference on Digital Government Research (DG.O 2005), pages 277–278, Atlanta, Georgia, USA, 2005. Digital Government Research Center.

[15] A. Gomez-Perez, F. Ortiz-Rodriguez and B. Villazon-Terrazas, *Ontology-based legal information retrieval to improve the information access in e-government,* in Proc. of the International Conference on World Wide Web (WWW '06), pages 1007–1008, Edinburgh, Scotland, UK, 2006. ACM Press.

[16] K. Goto and Y. Kambayashi, *A new passenger support system for public transport using mobile database access,* in Proc. of the International Conference on Very Large Databases (VLDB 2002), pages 908–919, Hong Kong, China, 2002. VLDB Endowement.

[17] D. Gouscos, G. Mentzas and P. Georgiadis, *PASSPORT, a novel architectural model for the provision of seamless cross-border e-government services,* in Proc. of the International Workshop on Database and Expert Systems Applications (DEXA 2001), pages 318–322, Munich, Germany, 2001. IEEE Computer Society.

[18] B. Grosof and Y. Labrou, *An approach to using XML and a rule-based content language with an agent communication language,* In Proc. of the IJCAI-99 Workshop on Agent Communication Language, pages 96–117, Stockol, Sweden, 1999.

[19] N.F. Guler and E.D. Ubeyli, Theory and applications of telemedicine, *Journal of Medical Systems* **26**(3) (2002), 199–220.

[20] L. Guo, F. Shao, C. Botev and J. Shanmugasundaram, *XRANK: Ranked keyword search over XML documents,* in Proc. of the ACM International Conference on Management of Data (SIGMOD 2003), pages 16–27, San Diego, California, USA, 2003. ACM Press.

[21] X. Guo and J. Lu, *Recommending trade exhibitions by integrating semantic information with collaborative filtering,* in Proc. of the International Conference on Web Intelligence (WI 2005), pages 747–750, Compiegne, France, 2005. IEEE Computer Society.

[22] V. Hristidis, L. Gravano and Y. Papakonstantinou, *Efficient IR-style keyword search over relational databases,* in Proc. of the International Conference on very large Databases (VLDB 2003), pages 850–861, Berlin, Germany, 2003.

[23] M. Janssen, R.W. Wagenaar and J. Beerens, *Towards a exible ICT-architecture for multi-channel e-government service provisioning,* in Proc. of the Hawaii International Conference on System Sciences (HICSS 2003), page 148, Big Island, Hawaii, USA, 2003. IEEE Computer Society Press.

[24] W. Ke and K.K. Wei, Successful e-government in Singapore, *Communications of the ACM* **47**(6) (2004), 95–99.

[25] A. Kobsa, Generic user modeling systems, *User Modeling and User-Adapted Interaction* **11** (2001), 49–63.

[26] H. Lee, J. Alves-Foss and S. Harrison, The construction of secure mobile agents via evaluating encrypted functions, *Web Intelligence and Agent Systems: An International Journal* **2**(1) (2004), 1–20.

[27] X. Li and L. Soh, Hybrid negotiation for resource coordination in multiagent systems, *Web Intelligence and Agent Systems: An International Journal* **3**(4) (2005), 231–259.

[28] L. Lu, G. Zhu and J. Chen, *An infrastructure for e-government based on semantic Web Services,* in Proc. of the IEEE International Conference on Services Computing (SCC'04), pages 483–486, Shangai, China, 2004. IEEE Computer Society Press.

[29] N. Mallat, M. Rossi and V.K. Tuunainen, Mobile banking services, *Communications of the ACM* **47**(5) (2004), 42–46.

[30] G. Marchionini, H. Samet and L. Brandt, Introduction to the special issue on Digital Government, *Communications of the ACM* **46**(1) (2003), 24–27.

[31] M. Mecella and C. Batini, Enabling italian e-government through a cooperative architecture, *IEEE Computer* **34**(2) (2001), 40–45.

[32] B. Medjahed, A. Rezgui, A. Bouguettaya and M. Ouzzani, Infrastructure for e-government Web Services, *IEEE Internet Computing* **7**(1) (2003), 58–65.

[33] S. Melnik, H. Garcia-Molina and E. Rahm, *Similarity Flooding: A versatile graph matching algorithm and its application to schema matching,* in Proc. of the IEEE International Conference on Data Engineering (ICDE 2002), pages 117–128, San Jose, California, USA, 2002. IEEE Computer Society Press.

[34] R. Ramakrishnan and J. Gehrke, *Database Management Systems,* McGraw-Hill, 2003.

[35] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl, *Grouplens: An open architecture for collaborative filtering of netnews,* in Proc. of the Conference on Computer Supported Cooperative Work (CSCW'94), pages 175–186, Chapel Hill, North Carolina, USA, 1994. ACM Press.

[36] G. Sacco, *User-centric access to e-government information: e-citizen discovery of e-services*, in Proc. of the International Simposium The Semantic Web meets eGovernment (SWEG'06), pages 114–116, Stanford, California, USA, 2006. AAAI Spring Symposium Series.

[37] B.M. Sarwar, G. Karypis, J.A. Konstan and J. Riedl, *Analysis of recommendation algorithms for e-commerce,* in Proc. of the ACM Conference on Electronic Commerce (EC-00), pages 158–167, Minneapolis, Minnesota, USA, 2000. ACM Press.

[38] L. Stojanovic, A. Abecker, N. Stojanovic and R. Studer, *On managing changes in the ontology-based e-government,* in Proc. of the International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE 2004), pages 1080–1097, Agia Napa, Cyprus, 2004. Springer.

[39] A. Triantis and P.E. Pintelas, *A Mobile Multi Agent Architecture for Web Based Learning*, in Proc. of the IEEE International Conference on Advanced Learning Technologies (ICALT 2004), pages 51–55, Joensuu, Finland, 2004. IEEE Computer Society.

[40] R. van der Krogt, M. de Weerdt and C. Witteveen, A resource-based framework for planning and replanning, *Web Intelligence and Agent Systems: An International Journal* **1**(3) (2003), 173–186.

[41] C.J. Van Rijsbergen, *Information Retrieval*, Butterworth, 1979.

[42] M. Vozalis and K.G. Margaritis, On the enhancement of collaborative filtering by demographic data, *Web Intelligence and Agent Systems: An International Journal* **4**(2) (2006), 117–138.

[43] M. Wimmer, *Implementing a knowledge portal for e-government based on semantic modeling: The e-government intelligent portal* (*eip.at*), in Proc. of the Hawaii International International Conference on Systems Science (HICSS 2006), Kauai, Hawaii, USA, 2006. IEEE Computer Society.

[44] M. Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons, 2002.

[45] J. Yen, X. Fan and R. Volz, Information needs in agent teamwork, *Web Intelligence and Agent Systems: An International Journal* **2**(3) (2004), 231–248.