Adapted Information Retrieval in Web Information Systems Using PUMAS

Angela Carrillo-Ramos, Jérôme Gensel, Marlène Villanova-Oliver, and Hervé Martin

Laboratory LSR-IMAG. B.P. 72
38402 Saint Martin D'Hères Cedex, France
{carrillo, gensel, villanov, martin}@imag.fr

Abstract. In this paper, we describe how *PUMAS*, a framework based on Ubiquitous Agents for accessing *Web Information Systems* (*WIS*) through *Mobile Devices* (*MD*), can help to provide nomadic users with adapted information. Using *PUMAS*, the information delivered to a nomadic user is adapted according to, on the one hand, her/his preferences, intentions and history in the system and, on the other hand, the limited capacities of her/his *MD*. The adaptation performed by *PUMAS* relies on *pieces of knowledge* (we call "*facts*"), which are stored in *Knowledge Bases* managed by *PUMAS* agents. We focus here on the facts exploited to achieve adaptation by two of the four *Multi-Agent Systems* (*MAS*) that constitute the architecture of *PUMAS* (the *Information* and the *Adaptation MAS*). We also present an example which illustrates how *PUMAS* works and considers these facts when processing a query.

Keywords: PUMAS, Adaptation, Web Information System, Mobile Devices, Information Retrieval, Agent, Knowledge, Fact.

1 Introduction

Web-based Information Systems (*WIS*) are systems that permit collection, structuring, storage, management and diffusion of information, like traditional *Information Systems* (*IS*) do, but over a Web infrastructure. A *WIS* provides users with complex functionalities that are activated through a Web browser in a hypermedium interface. Nowadays, *Mobile Devices* (*MD*) can be used as devices for accessing a distant *WIS* but also as storage devices for (simple) *WIS* or applications. Thus, a *WIS* which executes on *MD* allows access, search and storage of resources (files) located on these *MD*.

However, having to cope with the limited capacities of *MD* (e.g. size of screen, memory, hard disk), *WIS* designers must use mechanisms and architectures in order to efficiently store, retrieve and deliver data using these devices. The underlying challenge is to provide *WIS* users with useful information based on an *intelligent search* and a *suitable display* of delivered information. In order to reach this goal, a *Multi-Agent System* (*MAS*) constitutes an interesting approach. The *W3C* [1] defines an *agent* as "*a concrete piece of software or hardware that sends and receives messages*". These messages can be used to access a *WIS* and to exchange information. A *MAS* can be a useful tool for modelling a *WIS* due to the inherent properties of

M. Kolp et al. (Eds.): AOIS 2005, LNAI 3529, pp. 243 – 258, 2006.

agents like the defined, owned and acquired knowledge they manage, their ability to communicate with users or other agents, etc. Carabelea *et al.* [2] have defined a *MAS* as "*a federation of software agents interacting in a shared environment that cooperate and coordinate their actions given their own goals and plans*". Moreover, agents can be executed on the *MD* and/or migrate through the net, searching for information on different servers (or *MD*) in order to satisfy user's queries. This is the underlying idea of the *Mobile Agent* concept [3].

Rahwan *et al.* [4] recommend the use of agent technology in *MD* applications because agents that execute on the user's *MD* can inform the systems accessed by the user about her/his contextual information. However, in the case of a mobile user, the agent must consider the fact that the changing location could produce changes in user tasks and information needs. Then, the agent also has to be proactive, and has to reason about user goals and the way they can be achieved.

Applications running on the *MD* (and their agents) must allow users to consult data at any time from any place. This is the underlying idea of *Ubiquitous Computing* (*UC*) [1]. Shizuka *et al.* [5] have stressed the fact that *Peer to Peer* (*P2P*) computing is one of the potential communicative architectures and technologies for supporting *ubiquitous/pervasive* computing. We can consider a *MAS* as a *P2P System*, since an agent is an inherent peer, because it can perform its tasks independently from the server and other agents. *P2P* systems [6] are characterized by i) a direct communication between peers with no communication needed through a specific server, and ii) the autonomy a peer gets for accomplishing some assigned tasks.

Concerning adaptation, special attention is paid to user's location in her/his profile. In order to provide the nomadic user only with *relevant information* (i.e. "*the right information in the right place at the right time*"), Thilliez *et al.* [7] have proposed "*location dependent*" *queries,* which are evaluated according to the user's current physical location (e.g. *"which are the restaurants located in the street where the user is?"*). Our work focusses also on this kind of *queries.*

Regarding adaptation to the reduced capacities of the *MD*, one objective is to anticipate the fact that some retrieved information cannot eventually be properly displayed (e.g. *MD* may not support a cumbersome format file). It is necessary to anticipate such situations at design time in order to decide which solution to implement. For instance, considering a query whose result contains video data, the corresponding result may not be delivered if the user accesses the *WIS* through a mobile phone that cannot display videos. In that case, the Negotiation vocabulary proposed by Lemlouma [8] can be used for adaptation purposes. It permits description of the user's *MD*, considering constraints in terms of network, software and hardware.

Many technical and functional aspects have to be considered when designing a *WIS* accessed through *MD*, especially when addressing the issue of adaptation of delivered information to the nomadic user [6, 7]. The goal of our work is to provide nomadic users who access a *WIS* through a *MD* with the most relevant information according to their preferences, but also according to their contextual characteristics and to the features of their *MD*. In [9], we have defined *PUMAS*, a framework for retrieving information distributed among several *WIS* and/or accessed through different types of *MD*. The architecture of *PUMAS* is composed of four *MASs* (a *Connection MAS*, a *Communication MAS,* an *Information MAS* and an *Adaptation MAS*), each one encompassing several *ubiquitous agents* which cooperate in order to achieve the

different tasks handled by *PUMAS* (*MD* connection/disconnection, information storage and retrieval etc.). In *PUMAS*, data representation, agent roles and data exchange are ultimately based on *XML* files (using *OWL*[1]). Through *PUMAS*, our final objective is to propose and build a framework which is, beyond the management of accesses to *WIS* through *MD*, also in charge of performing some adaptation processing over information. Users equipped with *MD* can use the *PUMAS* central platform in order to communicate together by means of agents that execute on their *MD,* or in order to exchange information (user's contextual information). In our case, users communicate through an *Hybrid P2P system*.

This paper is structured as follows. We first describe in Section 2 the architecture of *PUMAS*. The main contributions of this paper are, on the one hand, the definition of *pieces of knowledge* (that we call *facts*) used for adaptation purposes by *PUMAS* agents, especially those belonging to the *Information* and to the *Adaptation MAS,* and, on the other hand, the data representation based on *XML* files. In Section 3, we present a scenario that shows how *PUMAS* processes a query submitted to the system. An example that illustrates our proposition is given in Section 4. We discuss works related to *PUMAS* in Section 5 before we conclude in Section 6.

2 The PUMAS Framework

In this section, we present the architecture of *PUMAS*, its four *MASs*, their relations and, the data exchange and communications they perform in order to achieve adaptation of information for the user.

2.1 An Overview of the PUMAS Architecture

The architecture of *PUMAS* is composed of four *MASs* (see Fig. 1 for the logical structure of *PUMAS*). Firstly, the *Connection MAS* provides mechanisms for facilitating connection from different types of *MD* to the system. Secondly, the *Communication MAS* ensures a transparent communication between *MD* and the system, and applies a *Display Filter* to display the information in an adapted way according to technical constraints of the user's *MD*. To achieve this, it is helped by agents of the *Adaptation MAS*. Thirdly, the *Information MAS* receives users' queries, redirects them to the "*right*" *WIS* (e.g. the nearest *WIS*, the more consulted one), applies a *Content Filter* (with the help of the *Adaptation MAS* agents) according to the user's profile in the system and returns results to the *Communication MAS*. Finally, the *Adaptation MAS* communicates with agents of the three other *MAS* in order to provide them with information about the user, connection and communication features, *MD* characteristics etc. The services and tasks of its agents essentially consist of managing specific *XML* files that contain information about the user and the device. These agents also have some knowledge, which allows them to select and to filter information for users. This knowledge comes from analysis of the user's history in the system (e.g. last connections, queries, preferences).

---

[1] *OWL*: *Ontology* Web *Language* builds on *RDF* and *RDF Schema* and adds more vocabulary for describing properties and classes (relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes). *http://www.w3.org/2004/OWL/*

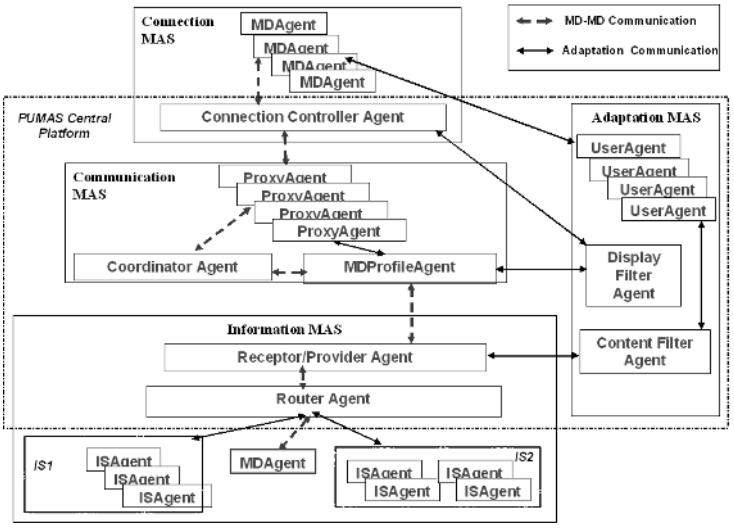

Fig. 1. The *PUMAS* Architecture

The inherent mobility of nomadic users is supported by *ubiquitous agents*: the *Mobile Device Agents* executed on the user's *MD* and the *ISAgents* executed on the same device than the *WIS* to which they belong. Such ubiquitous agents retrieve some needed information and can communicate with other agents to perform tasks. The *Hybrid P2P Architecture* of *PUMAS* copes with the following issues: security in applications (security problems inherent to agent mobility), communication between agents in a point to point or in a broadcast way, management of the status of the agent (e.g. connected, disconnected and killed) and its services. In the following subsections, we describe the tasks achieved by each *MAS* of *PUMAS*.

2.2 The Connection MAS

This *MAS* includes several *Mobile Device Agents* (*MDA*) and one *Connection Controller Agent* (*CCA*).

The ***Mobile Device Agent*** is executed on the user's *MD*. Its knowledge is composed of general rules of behaviour and characteristics related to the type of *MD* used (e.g. *PDA*) as well as some specific roles defined according to the application (e.g. this agent is used for transmitting a file). The *Mobile Device Agent* manages a *XML* file (*Device Profile XML* file, located on the user's *MD*), which describes *MD* features and shares this information with the *Display Filter Agent* (belonging to the *Adaptation MAS*) through the *Connection Controller Agent* (the *Mobile Device Agent* sends this file to the *Connection Controller Agent* – executing on the central platform of *PUMA* – and the latter exchanges this information with the *Display Filter Agent*). This file contains some information about the requirements of the application, network status, hypermedia files supported by the *MD*, conditions for disconnecting: inactive session for more than *X* minutes, disconnection type (e.g. willingly, automatic), etc. One *Mobile Device Agent* also manages another *XML* file, which describes characteristics of the user's session (using *OWL*, see Fig. 2): who is the user

connected (user *ID*), when the session began and what is the *MD* connected (beginning time, *CurrentMD*). This file will be sent to the *UserAgent* (belonging to the *Adaptation MAS*):

```
<?xml version="1.0"?>
   <rdf:RDF…   …
         <owl:Ontology rdf:about=""/>
               <owl:Class rdf:ID="SessionProfile"/>
               <owl:Class rdf:ID="CurrentUser">
                     <rdfs:subClassOf rdf:resource="#SessionProfile"/></owl:Class>
               <owl:Class rdf:ID="BeginningTime">
                     <rdfs:subClassOf rdf:resource="#SessionProfile"/></owl:Class>
               <owl:Class rdf:ID="CurrentDevice">
                     <rdfs:subClassOf rdf:resource="#SessionProfile"/></owl:Class>
   </rdf:RDF>
```

Fig. 2. Code excerpt of the *User's Session XML* file

The ***Connection Controller Agent*** executes on the central platform of *PUMAS* and gets the user's location and *MD* type (e.g. *PDA*) from the *User Location XML file* (which contains the physical and logical user's location features; this file is also defined using *OWL*) and from the *Device Profile XML file* (which describes the features of the *MD*), respectively. Both files are provided by the *Mobile Device Agent* and locally managed by the *Connection Controller Agent*. The latter serves as an intermediary between the *Connection MAS* and the *Communication MAS*. It also checks connections established by users and the status of agents (e.g. connected, disconnected, killed), and links each *Mobile Device Agent* to its corresponding *Proxy Agent* in the *Communication MAS* (see next section).

The *XML files* (*User Location, Session* and *Device Profile XML* files) managed by the *Mobile Device Agent* and the *Connection Controller Agent* have been defined using extensions introduced by Indulska *et al.* [10] to *CC/PP* [1]. These extensions include some user's characteristics like her/his location, application requirements, session features (e.g. user, device, application) and the profile of the *MD* in order to provide a complete description of the user and her/his *MD*.

2.3 The Communication MAS

This *MAS* has an interface that makes communication between users transparent and activates the mechanism for displaying the information according to the features of the *MD*. It is composed by several *Proxy Agents* (*PA*), one *MDProfile Agent* (*MDPA*) and one *Coordinator Agent* (*CA*). These agents execute on the central platform of *PUMAS*.

There is one ***Proxy Agent*** for the connection of each *Mobile Device Agent.* Two different users can connect themselves to the system through the same *MD,* which leads to two different *Proxy Agents* and two different sessions. The main task of a *Proxy Agent* is to represent a *Mobile Device Agent* within the system. In this case, there are two agents: one *Mobile Device Agent* in the *MD* and one *Proxy Agent* in the central platform of *PUMAS*.

The ***MDProfile Agent*** has to check the user's profile (according to her/his *MD*) and her/his information needs. In addition, this agent together with the *Coordinator Agent* defines and checks the mechanism that sends, for example, hypermedia data to the user. If the user's request has, as a result, several images, these agents define the order and number of images to be shown by the screen according to the capabilities of the user's *MD*. The *MDProfile Agent* also shares information about specific *MD* features for the user's session with the *Display Filter Agent* (belonging to the *Adaptation MAS*).

The ***Coordinator Agent*** is in permanent communication with the *Connection Controller Agent* in order to verify the connection status of the agent that searches for information. The *Coordinator Agent* knows all the agents connected in the system thanks to *XML* files managed by the *Mobile Device Agent* (through its *Proxy Agent*). If there are some problems with the *Connection Controller Agent* (e.g. if the *Connection Controller Agent* fails or if there is a lot of connections), the *Coordinator Agent* can play the role of the *Connection Controller Agent* until the problems are fixed. At that moment, the *Connection Controller Agent* and the *Coordinator Agent* must synchronize the information about connected agents and check current connections.

A more detailed description of the *Connection* and the *Communication MAS* can be found in [9]. The main contribution of this paper, described in the next section, deals with the description of the knowledge managed by the *Information* and the *Adaptation MAS* agents in order to support the adaptation capabilities of *PUMAS*.

2.4 The Information MAS

The *Information MAS* is composed of one or several *Receptor/Provider Agents* (*R/PA*), one or several *Router Agents* (*RA*) and one or several *ISAgents* (*ISA*).

A ***Receptor/Provider Agent*** that is located in the central platform of *PUMAS* owns a general view of the whole system. It knows agents of both the *Communication* and the *Information MAS*. The *Receptor/Provider Agent* receives all requests that are transmitted from the *Communication MAS* and redirects them to the *Router Agent,* which is in charge of finding the "*right*" *WIS* in order to execute the query. Once a query has been processed by the *ISAgents*, the *Receptor/Provider Agent* checks whether query results consider the user's profile (i.e. preferences, user's history) by means of the *Content Filter Agent* (belonging to the *Adaptation MAS*).

In order to redirect a query to the "*right*" *WIS*, a ***Router Agent*** (which executes on the central platform of *PUMAS*) applies a strategy that depends on one or several criteria: user's location, peer similarity, time constraints, user's preferences etc. The strategy can lead to sending the query to a specific *WIS*, to sending the query using broadcast and/or to the division of the query in sub-queries, each being sent to one or several *WIS*. A *Router Agent* is also in charge of compiling results returned by the *WIS* and of analyzing them (according to the defined criteria) to decide whether the whole set of results or only a part has to be sent to a *Receptor/Provider Agent*.

The *Router Agent* stores in its *Knowledge Base pieces of knowledge* (that we call *facts* and describe below using *JESS*[2]) for each *WIS*. One *fact* is made up of the

[2] *JESS* is a rule engine and scripting environment that enables building Java applications that have the capacity of "reasoning" using knowledge supplied in the form of declarative rules. *http://herzberg.ca.sandia.gov/jess/*

characteristics of the *WIS,* like its name, its managed information, the type of device on which it is executed (e.g. server, *MD*) and the agent (*ISAgent*) associated with this *WIS* and which can be asked for information. When the *Router Agent* has to redirect a user's query, it exploits these facts in order to select the *WIS*, especially, the *ISAgents* (which execute on the same device that the *WIS*) to which sub-queries have to be redirected. The following fact defines a *WIS* and is represented by a *JESS* template[3]:

(*deftemplate WIS* (*slot name*)
(*slot agentID*) (*slot device*)
(*multislot information_items*))

For instance, the following fact defines the *Pharmacy WIS* of a hospital. The *WIS* is called *PharmacyWIS* and it executes on a *server*. *PharmacyISA* is the *ISAgent* which executes on this *WIS*. The *PharmacyWIS* contains information about *medicines* and *patient prescriptions*:

(*assert* (*WIS* (*name PharmacyWIS*)
(*agentID PharmacyISA*) (*device server*)
(*information_items "medicines" "patient's_prescription"*)))

The location of the *WIS* could change, especially if this *WIS* runs on a *MD*. The *Router Agent* can be informed about the changes in the location of the *WIS* by means of the *ISAgents* that execute on these *WIS*.

In order to send (sub-) queries and analyse their results, the *Router Agent* must check the user's preferences (information provided by the *Content Filter Agent* via the *Receptor/Provider Agent*). The user's preferences are represented as facts defined as follows:

(*deftemplate User_Preference* (*slot userID*)
(*slot required_info*)
(*multislot complementary_info*)
(*multislot actionD*) *; actions for doing*
(*slot problem*)
(*multislot actionR*)) *; actions for recovering*

An ***ISAgent*** associated with a *WIS* (and which executes on the same device that the *WIS*) receives the user's query from the *Router Agent* and is in charge of searching for information. Once a result for the query is obtained, the *ISAgent* returns it to the *Router Agent*. An *ISAgent* can execute a query by itself or delegate this task to the adequate *WIS* component. This depends notably on the nature of the *WIS*. Our approach addresses complex and possibly a distributed *WIS* located on server(s) but also a very simple *WIS* that only relies on some files located on a *MD*. In this last case, one *ISAgent* may be sufficient to ensure the right functioning of the *Information MAS*. It is worth noting that, in this case, what we call an *"ISAgent"* is in fact the *Mobile Device Agent* of a *MD* that can play the role of an *ISAgent* since it has the knowledge required for executing a query on files stored in the *MD*. In a complex *WIS*, the *ISAgent* can collaborate with other *ISAgents* (if the *WIS* has been developed

[3] We define our pieces of knowledge using the syntax of the *JESS* unordered facts. We declare each unordered fact by means of the primitive "*deftemplate*". To define an instance of an unordered fact in *JESS* and store it into the *JESS Knowledge Base*, we use the primitive "*assert*".

following the *MAS* paradigm) or with any other *WIS* component to perform a query. In the case of a non *MAS* based *WIS*, our approach only requires that an *ISAgent* is developed in order to ensure the communication between *PUMAS* and the *WIS*.

2.5 The PUMAS Adaptation MAS

The adaptation capabilities of *PUMAS* rely on a two step filter process that aims at providing a user with adapted information according to both the user and her/his *MD*. First, the *Content Filter* allows selection of the most relevant information according to the user's profile defined. Second, the *Display Filter* is applied on the results of the first filter and considers characteristics and technical constraints of the user's *MD*.

The *Adaptation MAS* is composed of several *UserAgents* (*UA*), one *Display Filter Agent* (*DFA*) and one *Content Filter Agent* (*CFA*). These agents execute on the central platform of *PUMAS*.

```
<rdf:RDF ...
<owl:Ontology rdf:about=""/>
        <owl:Class rdf:ID="UserProfile"/> <owl:Class rdf:ID="Beliefs">
                <rdfs:subClassOf rdf:resource="#UserProfile"/></owl:Class>
        <owl:Class rdf:ID="Intentions">
                <rdfs:subClassOf rdf:resource="#UserProfile"/></owl:Class>
         <owl:Class rdf:ID="User">
                <rdfs:subClassOf rdf:resource="#UserProfile"/></owl:Class>
        <owl:Class rdf:ID="Preferences">
                <rdfs:subClassOf rdf:resource="#UserProfile"/></owl:Class>
</rdf:RDF>
```

Fig. 3. Code excerpt of the *User Profile XML* file

Each ***UserAgent*** manages a *XML* file (*User Profile XML* file, see Fig. 3) that contains personal characteristics of the user (e.g. user ID, location) and her/his preferences (e.g. the user wants only video files). This file is obtained by means of the *Mobile Device Agent* (this file is managed by the *UserAgent* and updated by the *Mobile Device Agent*). There is only one *UserAgent* that represents a user at the same time (even though the user has two sessions at the same time through the same or different *MD*). Since a user can access the system through several *MDs,* the *UserAgent* communicates with the *Mobile Device Agents* and the *Proxy Agents* (which respectively belong to the *Connection* and the *Communication MAS*) to analyse and centralize all the characteristics of the same user. The *UserAgent* communicates with the *Content Filter Agent* to send the *User Profile XML* file. When the *Content Filter Agent* receives this file, it stores this information as facts in its *Knowledge Base* (this agent manages a registry of user's preferences). When the *Receptor/Provider Agent* (belonging to the *Information MAS*) asks the *Content Filter Agent* for the user's preferences, the latter sends it the latest *XML* file received from the *UserAgent*. If the *UserAgent* does not send this file (e.g. there are no user preferences for the current session), the *Content Filter Agent* considers the preferences from previous sessions.

We can establish that queries depend on one or several criteria for adaptation purposes: the user's location, her/his history in the system, activities developed during a time period, movement orientation, privacy preferences, etc. An *Adaptation_ Criterion* could be defined as:

(*deftemplate Adaptation_Criterion*
(*slot userID*) (*multislot criteria*) (*multislot attributes*))

An example of *Adaptation_Criterion* that expresses that all of *Doctor Smith's* queries depend on his location, especially when he is at the *North Hospital* could be:

(*assert* (*Adaptation_Criterion*
(*userID "Doctor Smith"*) (*criteria location*) (*attributes "North_Hospital"*)))

The ***Display Filter Agent*** manages a *Knowledge Base* that contains general information about features of different types of *MD* (e.g. format files supported) and acquired knowledge from previous connections (e.g. problems and capabilities of networks according to data transmissions). Each *MDFeature* is defined as a fact and represented as follows:

(*deftemplate MDFeature* (*slot MDtype*) (*multislot feature*))

where each feature is represented as a fact as follows:

(*deftemplate feature* (*slot type*) (*multislot causes*))

An example of a fact for a *MDFeature* which corresponds to file formats that are supported by a *Pocket PC hp IPAQ h5550* in different network types is shown as follows. We assume that it cannot support video sent on a *Wi-Fi Network* but it does support several images using either *Bluetooth* or a *Classical Network*:

(*deffacts MDFeature* (*MDType "PocketPC hpIPAQ h5550"*)
(*feature* (*type "video_not_supported"*) (*causes "Wi-Fi Network"*))
(*feature* (*type "several_images"*) (*causes "Bluetooth" "Classical Network"*)))

The ***Content Filter Agent*** manages a *Knowledge Base* that contains preferences, intentions and characteristics of users. The *User_Preference* fact is composed of a *userID* (which identifies the owner of this preference), required information (*required_info*) and complementary information (*complementary_info*). The last is added to the *User_Preference* definition by the *Content Filter Agent,* which analyses queries of previous sessions (e.g. information frequently asked). This fact is also composed of information describing what and how user would like answers from the system (to be presented to her/him) and in case there are any problems, what and how the system must answer (*list of actions for recovering*). In order to do this, each *action* is defined as a fact and represented as follows:

(*deftemplate action* (*slot name*) (*multislot attribute*))

In this definition, *name* refers to an action chosen between a defined list (e.g. "*show"*, "*save*", "*transfer file*", "*cancel*") and each action has a list of *attributes*. For instance, the fact which represents the action "*show*" has for its properties the *order, format* and *type of the file,* is:

(*assert* (*action* (*name show*) (*attributes "order" " format" "file_type"*)))

Since an *attribute* can be complex, we define it as a fact:

(*deftemplate attribute* (*slot name*) (*multislot list*))

An example of *attribute* that defines the order in which information is displayed, could be:

(*assert* (*attribute* (*name order*)
(*list "patient's_tests" "patient's_diet" "patient's_prescribed_medicines"*)))

We define a *problem* as an event that is not desirable during the execution of an action or that is the cause of a failure (e.g. the *MD* cannot *show* an image). Each problem is defined as a fact and represented as follows:

(*deftemplate problem* (*slot name*) (*slot type*) (*multislot causes*))

where *name* corresponds to a description of the problem, the *type* can be chosen from a defined list (e.g. *incompatibility, unable IS, unable agent*) and the *causes* correspond to a list of causes of this problem (e.g. *MD* cannot support a specific format file, network problems). A fact, which defines the *problem* related to a specific user's location that is out of range of a wireless network and disables her/him from accessing the Internet, is:

(*assert* (*problem* (*name "out_range_connection"*) (*type "lack_of_access"*)
(*causes "user_located_out_of_range" " network_out_of_service"*)))

3 PUMAS Scenario

In this section, we present a scenario in order to show the interactions that take place between *PUMAS* agents when a query is submitted to the system.

When a user sends an information query *Q* (see Fig. 4), the *Mobile Device Agent* sends it to the *Connection Controller Agent.* Whenever this query is location and time dependent, the *Connection Controller Agent* introduces the *time of connection*, the *user's location* and the characteristics of the *user's MD* connection (these latter characteristics are exchanged with the *Display Filter Agent*) in query *Q* which leads to the production of a new query *Q'* (in Fig. 4, *Q'=Q + user's ST*) that is then sent to the *Proxy Agent*. The query passes by the *Coordinator Agent* and then by the *MDProfile Agent*. The latter adds to query *Q'* some features related to the *MD;* these features are provided by the *Display Filter Agent* which has previously learned them from previous queries or retrieved them from its *Knowledge Base*. The new query *Q"* (in the Fig. 4, *Q"= Q' + MD features*) is sent by the *MDProfile Agent* to the *Receptor/Provider Agent.* The *Receptor/Provider Agent* complements the query *Q"* with specific characteristics of the user in the system by requesting the *Content Filter Agent* (in Fig. 4, *Q'''=Q" + user's preferences, intentions, history*). The *Receptor/Provider Agent* sends the query *Q'''* to the *Router Agent,* which decides (according to the query, the system rules and the facts in its *Knowledge Base*) which are the *ISAgents* able to answer. It can send the query to a specific *ISAgent* or to several *ISAgents* (e.g. waiting for the first to answer) or, it can divide the query into sub-queries, which are sent to one or several *ISAgents*. The scenario in Fig. 4, shows for instance that query *Q'''* is divided into $Q^{'''-1.1}$, $Q^{'''-1.2}$, $Q^{'''-1.3}$ and $Q^{'''-1.4}$, which are sent to the *ISAgents* executed on a server and different *MD*.

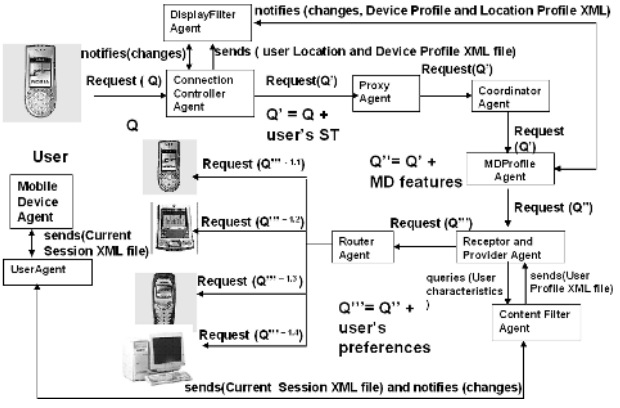

Fig. 4. Scenario of sending a query

When a user *U1* has an information query for another user *U2*, both equipped with *MD,* the query is propagated from the *Mobile Device Agent* executed on the *U1*'s *MD* towards the *Router Agent,* which redirects it to the *Mobile Device Agent* executed on the *U2*'s *MD.* This *U2*'s *Mobile Device Agent* changes its role to become an *ISAgent,* i.e. the agent in charge of answering information queries. This change of role is possible because a *Mobile Device Agent* has knowledge for managing information stored in the *MD* on which it executes and it has the capability of answering information queries.

4 Example

In this section, we illustrate processes performed by *PUMAS* agents using the example of a hospital *WIS*.

Let us suppose that doctors equipped with *MD* (e.g. *PDA*) access the information system of a hospital that is distributed between several *MD* and/or one or several *WIS* (see Fig. 5). Doctors can also receive information according to their location, preferences, technical characteristics of their *MD* and considerations about their connection time. For instance, when visiting a patient, doctors with *MD* can consult information about her/his clinical history, medical tests, prescriptions etc. By indicating the location of the patient (e.g. room, bed) and the current date (extracted from the system), the doctor can identify the patient and get her/his personal information. To do this, the application on her/his *MD* must consult different *WIS* of the hospital (e.g. pharmacy, doctors). Doctors could also communicate with other doctors (peers), through their *MD*, in order to get some advice or help (e.g. questions which can only be answered by the specialist doctor who has previously examined this patient).

When a doctor comes into the patient's room, she/he enters information about the location of the patient while the application gets the date of the system (information about the time). The *Mobile Device Agent* that executes on the doctor's *MD* sends the

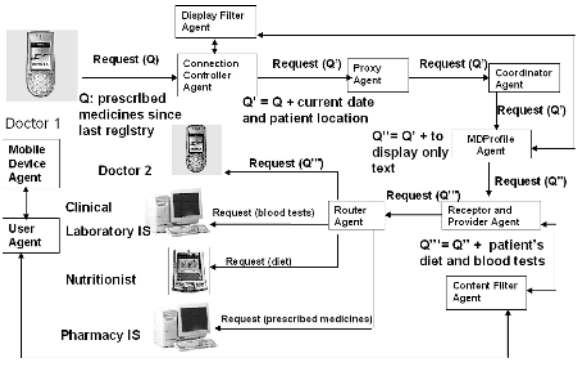

Fig. 5. Sending a query in the hospital *WIS*

query (*who is the patient?*). The query is propagated through the *PUMAS* core: it is first transmitted through the *Connection Controller Agent*, then to the *Communication MAS* agents (*Proxy Agent, Coordinator Agent* and *MDProfile Agent*). The *MDProfile Agent* can include, in the query, information according to the *MD* (e.g. this kind of *MD* cannot support graphical format but only text files. Then, if the doctor asks for the results of the test, she/he only could get them in a text format). For example, if the doctor has been connected through a *Palm Tungsten C,* the *MDProfile Agent* asks the *Display Filter Agent* for information about this *MD.* The *MDProfile Agent* could receive from *Display Filter Agent* facts defined as follows:

(*deffacts MDFeature* (*MDType "Palm Tungsten C"*)
(*feature* (*type "video_not_supported"*)(*causes "Wi-Fi_Network"*))
(*feature* (*type "several_images"*)(*causes "Wi-Fi_Network"*)
(*feature* (*type "text"*)(*causes "Wi-Fi_Network" "Classical_Network Bluetooth"*))) *;*

Then, the *MDProfile Agent* sends the query to the *Receptor/Provider Agent,* which can include in the query preferences previously expressed by the doctor. Those preferences are expressed in the *User Profile XML* file (see Section 2.5) and are translated as facts by the *UserAgent* and the *Content Filter Agent*. The following example corresponds to a doctor's preference: when a doctor says "*when asking for a blood tests, the system must also provide me with the patient's diet and the prescribed medicines. I do prefer graphical results but if my MD cannot support this format, I would like receive the results in text format*", this can be translated into the following fact of the *UserAgent*:

(*deffacts User_Preference*
(*userID "Doctor Smith"*)(*required_info "blood tests"*)
(*complementary_info "patient's_diet" "prescribed_medicines"*)
(*action show*) (*attribute* (*name order*)
(*list "patient's_tests" "patient's_diet" "patient's_prescribed_medicines"*))
(*attribute* (*name graphical_format*) (*list "JPEG"*))
(*problem* (*name "HyperMediaNotSupportedByMD"*)
(*type "incompatibility"*) (*causes "OnlyTextFileSupported"*)) (*attribute* (*name order*)
(*list "patient's_tests" "patient's_diet" "patient's_prescribed_medicines"*))
(*attribute* (*name text_format*) (*list "XML" "txt"*)))

The *UserAgent* transfers this information to the *Content Filter Agent,* which stores this fact and sends it to the *Receptor/Provider Agent.* The *Receptor/Provider Agent* adds this preference to the query and sends it to the *Router Agent*. The *Router Agent* receives the complete query and, with the information about the *WIS*, the *Router Agent* can split the query in sub-queries and redirect each one towards the appropriated *WIS*. The following facts are exploited in this example by the *Router Agent* in order redirect the queries to the *ISAgents* of the hospital's *WIS*:

(*assert* (*WIS* (*name LaboratoryWIS*)
(*agentID LaboratoryISA*) (*device server*)
(*information_items " test" " patient's_ test" "reactive"*)))

(*assert* (*WIS* (*name PatientDietWIS*)
(*agentID DietISA*) (*device MD*)
(*information_items "patient's_diet" "nutritionist's_appointments"*)))

The *Router Agent* redirects the query to the *ISAgent* located in the *WIS,* which manages information about patients in the hospital. All queries follow the same path from the *Mobile Device Agent* towards the *Router Agent*. If the doctor wants to know the *last medicines prescribed* to this patient, the *Router Agent* redirects the query to the *ISAgent* located in the *PharmacyWIS*. If the query concerns another *doctor* (*peer*), the *Router Agent* redirects the query to the *ISAgent* located in the peer's *MD*. A doctor can also ask for information about a specific patient to several of her/his peers. In this case, the *Router Agent* could send the query using broadcast or it could split the query according to the receiver peer (e.g. queries related to heart conditions for the cardiologist) or according to the defined criteria in the *User Profile XML* file (e.g. if the criterion of adaptation of the query is the *location*, queries must only be redirected to doctors at the *same* or *closed location* of the sender). Retrieved information is organized by the *Router Agent* (e.g. the last prescribed medicines, peer answers about this patient) and is returned to the doctor who has sent the query following the inverse path. The different agents have to check results because, for instance, the doctor may have been disconnected from the system (due to some network problems), and recovered her/his session in a new connection whose characteristics are different from the previous ones: it could be that she/he can now consult the system using another kind of *MD* that supports some graphical format (which constitutes a doctor's preference that can now be satisfied).

Through this example, we can observe the behaviour of the *Hybrid P2P Architecture* of *PUMAS*. The core of *PUMAS* centralizes queries: i) it is in charge of obtaining the most relevant information and, ii) it is in charge of applying the *Content* and *Display Filters* to adapt answers. The main peer characteristics of *PUMAS* agents are illustrated by the fact that, firstly, agents have the autonomy of connecting to and disconnecting from the system. Secondly, a *MD* can ask for a communication with a specific *WIS* (located on a server or on a *MD*) passing this information as a parameter of the query; the *Router Agent* transmits the query to this specific *WIS* which exemplifies an agent to agent communication (e.g. when doctors exchange information about a patient using their *MD*).

Another advantage offered by *PUMAS* is that it helps a user who does not know which specific *WIS* to ask for information to find the most appropriate one(s). The *Router Agent* redirects a query by means of an intelligent analysis of the query and the

help of the *ISAgents* that achieve an intelligent search inside the different *WIS* (pharmacy, laboratory, patients etc. in our example).

5 Related Works

In this section, we present some agent-based architectures or frameworks for adapting information to users.

Berhe *et al.* [11] proposes an architectural framework that exploits four profiles for adapting information content to a user: *content or media* (type, format, size, location where media is stored), *user* (preferences), *device* (hardware and software capabilities), *network* and *service* (supported media formats, network connection, bandwidth, latency performance). However, unlike *PUMAS*, this proposal does not consider information retrieval from different types of devices (servers and *MD*).

Sashima *et al.* [6] proposes an agent-based coordination framework for ubiquitous computing. It coordinates services and devices to assist a particular user in receiving a particular service in order to maximize her/his satisfaction. This framework assists users in accessing resources in ubiquitous environments. These authors consider contextual features of nomadic user, especially location. Unlike *PUMAS*, this framework does not consider the adaptation of information according to the access devices or the possible distribution of data among different devices.

The work of Gandon *et al.* [12] proposes a Semantic Web architecture for context-awareness and privacy. This architecture supports automated discovery and access of a user's personal resources subject to user-specified privacy preferences. Service invocation rules along with services ontologies and services profiles allow identification of the most relevant resources available to answer a query. However, it does not consider the information that can answer a query can be distributed between different sources.

CONSORTS Architecture [13] is based on ubiquitous agents and designed for a massive support of *MD*. It detects user's location and defines the user's profile to adapt the information to her/him. The *CONSORTS* architecture proposes a mechanism for defining relations that hold between agents (e.g. communication, hierarchy, role definition), with the purpose of satisfying user's requests. However, it does not consider the distribution of information between *MD* (which could improve response time) nor user's preferences.

PIA-System [14] is an agent-based personal information system for collecting, filtering and integrating information at a common point, offering access to information by *WWW*, e-mail, *SMS*, *MMS* and *J2ME* clients. It allows the user on the one hand, to search explicitly for specific information and, on the other hand, to be informed automatically about relevant information divided into slots (user specifies her/his working time and this divided the day in pre, work and recreation). A personal agent manages the individual information provisioning, tailored to user's needs according to her/his profile and current situation. However, a *PIA-System* only searches information in text format (e.g. documents). It does not take into account either the adaptation of different kinds of media to different *MD* or the user's location.

6 Conclusion

In this paper, we have presented *PUMAS*, a framework based on agents and the *P2P* approach. Peer characteristics of *PUMAS* appear in the cooperation developed by agents in order to store and retrieve information and in the possibility that two users, equipped with *MD*, communicate through the central platform offered by *PUMAS*. Its architecture relies on four *Multi-Agents Systems* (*MAS*) for the *Connection,* the *Communication,* the *Information* and the *Adaptation MAS*. *PUMAS* also benefits from the *P2P* characteristics of an *Hybrid P2P architecture*. *PUMAS* provides each agent with a mechanism for identifying, authenticating and recognizing its peers. This paper has focussed on the representation of the *pieces of knowledge* (called *facts*), stored in *Knowledge Bases* and used by *PUMAS* agents in order to perform their assigned tasks. We can highlight the *intelligent* and *adaptive information search* achieved by means of *PUMAS* agents. The search is *intelligent* because it is based on the knowledge of an agent and its capability of reasoning. It is also *adaptive* because it considers nomadic user's profiles, characteristics of her/his *MD* and features of the ubiquitous context.

Our future work concerns the implementation of each component (*MAS*) of *PUMAS.* We also need to define an extension of the current *ACL* that considers spatial-temporal (contextual) features and a strategy description language, as well as *Query Routing* mechanisms and algorithms [15] for the *Router Agent* in order to propagate queries towards the "*right*" *WIS* and to compile answers. Moreover, the mechanisms and strategies applied by the *PUMAS* agents (especially those belonging to the *Adaptation MAS*) in order to achieve the *Content* and the *Display Filters* also have to be precisely defined.

Acknowledgments. The author Angela Carrillo-Ramos is partially supported by Universidad de los Andes (Bogotá, Colombia).

References

1. *http://www.w3.org/TR/webont-req/* (L.A.: August 2006).
2. Carabelea, C., Boissier, O., Ramparany, F.: Benefits and Requirements of Using Multi-agent Systems on Smart Devices. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.): Proc. of the European Conference on Parallel Processing. Lecture Notes in Computer Science, Vol. 2790, Springer-Verlag, Berlin Heidelberg New York (2003) 1091-1098.
3. Lin, F.C., Liu, H.H.: MASPF: Searching the Shortest Communication Path with the Guarantee of the Message Delivery between Manager and Mobile Agent. In: Yang, L.T., Guo, M., Gao, G.R., Jha, N.K. (eds.): Proc. of the Conference on Embedded and Ubiquitous Computing. Lecture Notes in Computer Science, Vol. 3207, Springer-Verlag, Berlin Heidelberg New York (2004) 755-764.
4. Rahwan, T., Rahwan, T., Rahwan, I., Ashri, R.: Agent-Based Support for Mobile Users Using AgentSpeak(L). In: Giorgini, P., Henderson-Sellers, B., Winikoff, M. (eds.): Proc. of the 5th International Bi-Conference Workshop on Agent-Oriented Information Systems. Lecture Notes in Artificial Intelligence, Vol. 3030, Springer-Verlag, Berlin Heidelberg New York (2004) 45-60.

5. Shizuka, M., Ma, J., Lee, J., Miyoshi, Y., Takata, K.: A P2P Ubiquitous System for Testing Network Programs. In: Yang, L.T., Guo, M., Gao, G.R., Jha, N.K. (eds.): Proc. of the Conference on Embedded and Ubiquitous Computing. Lecture Notes in Computer Science, Vol. 3207, Springer-Verlag, Berlin Heidelberg New York (2004) 1004-1013.
6. Sashima, A., Izumi, N., Kurumatani, K.: Bridging Coordination Gaps between Devices, Services, and Humans in Ubiquitous computing. In: Proc. of the Workshop on Agents for Ubiquitous Computing. *http://www.ift.ulaval.ca/~mellouli/ubiagents04/.* (L.A.: July 2006)
7. Thilliez M., Delot T.: Evaluating Location Dependent Queries Using ISLANDS. In: Ramos, F.F., Unger, H., Larios, V. (eds.): Proc. of the Symposium on Advanced Distributed Systems. Lecture Notes in Computer Science, Vol. 3061, Springer-Verlag, Berlin Heidelberg New York (2004) 126-136.
8. Lemlouma, T.: Architecture de Négociation et d'Adaptation de Services Multimédia dans des Environnements Hétérogènes. PhD Thesis, Institut National Polytechnique de Grenoble, Grenoble, June 2004 (in French).
9. Carrillo-Ramos, A., Gensel, J., Villanova-Oliver, M., Martin, H.: PUMAS: a Framework based on Ubiquitous Agents for Accessing Web Information Systems through Mobile Devices. In: Haddad, H., Liebrock, L.M., Omicini, A., Wainwright, R.L. (eds.): Proc. of the 20th ACM Symposium on Applied Computing. ACM Press, New York (2005) 1003-1008.
10. Indulska, J., Robinson, R., Rakotonirainy, A., Henricksen, K.: Experiences in Using CC/PP in Context-Aware Systems. In: Chen, M.S., Chrysanthis, P.K., Sloman, M., Zaslavsky, A.B. (eds.): Proc. of the 4th International Conference on Mobile Data Management. Lecture Notes in Computer Science, Vol. 2574, Springer-Verlag, Berlin Heidelberg N.Y. (2003) 247-261.
11. Berhe, G., Brunie, L., Pierson, J.M.: Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing. In: Vassiliadis, S., Gaudiot, J., Piuri, V. (eds.): Proc. of the 1st Conference on Computing Frontiers. ACM Press, New York (2004) 60-69.
12. Gandon, F., Sadeh, N.: Semantic Web Technologies to Reconcile Privacy and Context Awareness. In: Journal of Web Semantics 1(3) (2004). *http://www.websemanticsjournal.org/ps/pub/2004-17* (L.A.: August 2006).
13. Kurumatani, K.: Mass User Support by Social Coordination among Citizen in a Real Environment. In: Kurumatani, K., Chen, S., Ohuchi, A. (eds.): Proc. of the International Workshop on Multi-Agent for Mass User Support. Lecture Notes in Artificial Intelligence, Vol. 3012, Springer-Verlag, Berlin Heidelberg New York (2004) 1-16.
14. Albayrak, S., Wollny, S., Varone, N., Lommatzsch, A., Milosevic, D.: Agent Technology for Personalized Information Filtering: The PIA-System. In: Haddad, H., Liebrock, L.M., Omicini, A., Wainwright, R.L. (eds.): Proc. of the 20th ACM Symposium on Applied. ACM Press, New York (2005) 54-59.
15. Xu, J., Lim, E., Ng, W.K.: Cluster-Based Database Selection Techniques for Routing Bibliographic Queries. In: Bench-Capon, T.J.M., Soda, G., Tjoa, A.M. (eds.): Proc. of the Workshop on Database and Expert Systems Applications. Lecture Notes in Computer Science, Vol. 1677, Springer-Verlag, Berlin Heidelberg New York (1999) 100-109.