A SITUATION-AWARE RESOURCE RECOMMENDER BASED ON FUZZY AND SEMANTIC WEB RULES

ALESSANDRO CIARAMELLA
IMT Lucca Institute for Advanced Studies,
Piazza San Ponziano 6, 55100 Lucca, Italy
a.ciaramella@imtlucca.it

MARIO G. C. A. CIMINO*, BEATRICE LAZZERINI† AND FRANCESCO MARCELLONI‡
Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni,
University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy
**m.cimino@iet.unipi.it*
†*b.lazzerini@iet.unipi.it*
‡*f.marcelloni@iet.unipi.it*

Received 30 January 2010
Revised 15 May 2010
Accepted 5 June 2010

Nowadays, a huge quantity of resources for mobile users are made available on the most important marketplaces. Further, handheld devices can accommodate plenty of these resources, such as applications, documents and web pages, locally. Thus, to search for resources suitable for specific circumstances often requires a considerable effort and rarely brings to a completely satisfactory result. A tool able to recommend suitable resources at the right time in each situation would be of great help for the mobile users and would make the use of the handheld devices less boring and more attractive. To this aim, new levels of granularity, together with some degree of self-awareness, are needed to assist mobile users in managing and using resources. In this paper, we propose an efficient situation-aware resource recommender (SARR), which helps mobile users to timely locate resources proactively. Situations are determined by a semantic reasoner that exploits domain knowledge expressed in terms of ontologies and semantic rules. This reasoner works in synergy with a fuzzy engine, which is in charge of handling the vagueness of some conditions in the semantic rules, computing a certainty degree for each inferred situation. These degrees are used to rank the situations and consequently to assign a priority to the resources associated with the specific situations. The application of SARR to two real business cases is also shown and discussed.

Keywords: Mobile service recommender; context-awareness; fuzzy inference system; web ontology; semantic rules.

1. Introduction

In the last years an enormous quantity of resources for mobile users have been made available on the most important marketplaces.[1–5] Such resources range from entertainment (games, songs, etc.) to information (weather forecast, traffic maps, news, etc.), from transactions (money transfer, airline reservation, etc.) to productivity (notepad, voice reader, etc.). The number and diversity of these resources make practically

impossible for the average user to identify the most suitable service or application for a specific situation. Indeed, the users probably do not know the specific features of all the resources and therefore rarely can associate these resources with specific situations. Even if the users were conscious of which service is suitable for each specific situation, to manually retrieve them while the situation occurs might be very hard. Indeed, the standard categorization of resources, e.g. based on their functions, is often ineffective. Actually, mobile resources may belong to several categories depending on the use case and the user's individual preferences.[6]

Searching and browsing are the most used mechanisms to locate resources in repositories. Typically, due to the large number and variety of resources, a vast amount of time is required to find the most suitable one. Furthermore, users in mobility are strongly limited in their search, using a hardware with reduced information presentation and interaction capabilities. Indeed, mobile devices feature small screens and miniaturized keypads in order to be portable and really handheld. Eventually, once installed, these resources have to be configured and launched with a set of proper parameters, which often vary in dependence of specific user circumstances.[7] Thus, a significant cognitive effort is required to users in mobility to find and configure the most appropriate resources among the many available.[8,9] These users would considerably benefit from a system able to automatically recommend resources at the right time and for the specific context.

In this paper, we propose a situation-aware resource recommender (SARR) for mobile users, which allows locating resources while the users need them, by taking the current situation into account.[8,9] In the proposed approach, recommendations are delivered in a proactive way, considering possibly uncertain contextual information. This is achieved by the integration of a fuzzy logic engine and a web semantic engine. More specifically, the semantic engine infers one or more current situations by exploiting domain knowledge expressed by ontologies and semantic rules. If multiple possible situations are inferred, the fuzzy engine computes a certainty degree for each situation, taking the intrinsic vagueness of some conditions of the semantic rules into account. Thus, the system can associate a rank with the recognized situations depending on such certainty degrees. Each situation is therefore associated with specific tasks, on the basis of domain knowledge expressed in terms of a task ontology.[10,11] Finally, the specific current task together with contextual information is used to recommend a set of resources, identified by means of a Label (or Tag)-based file system.[12]

We applied SARR to two real business cases, namely a pharmaceutical consultant in typical business situations and an off-site university student, who performs a daily travel to go to university and return. We show that SARR is able to provide an efficient and domain-independent resource recommender, thanks to the use of runtime engines and configuration procedures based on international standards, respectively.

In particular, the paper is organized as follows. In Sec. 2, a background of the concepts that are relevant to the development of a recommender system is presented and discussed. Section 3 is devoted to the characterization of the overall system architecture and of some minor modules. Sections 4 and 5 describe in detail the most important

modules of the SARR architecture, i.e., the semantic and the fuzzy engines, respectively. Section 6 shows the application of SARR to two business cases with some qualitative comparison to similar systems previously proposed in the literature. Finally, Sec. 7 draws some conclusion.

2. Background

To organize personal resources independently of their location, a bookmark management system is usually employed. The function of bookmarks is to offer an associative memory for personal usage. Conventional bookmarks contain a URL (or local path) and a title of the resource.[13] Bookmarks reduce the cognitive and physical loads of managing URL addresses, and facilitate the return to groups of related resources. Users collect bookmarks to create their own personal information space and share it with others.[14] However, organizing bookmarks is labor-intensive, requires a lot of time, and is difficult to do. In fact, typically users do not organize bookmarks.[13] Further, web usability studies show that bookmark lists are far from representing an effective personal information space.[14]

A number of researches have been performed to enhance bookmark functionality. In Ref. 15, context-dependent bookmarks have been discussed, with a method based on the automatic extraction of representative keywords of resources. The automatic extraction of representative keywords is applicable only to documents. In the case of applications, descriptors should convey the user intention, which is difficult to extract without semantically representing high-level concepts.

Recently, the *tagging* paradigm for information organization has become popular, especially in the context of collaborative tagging systems for managing shared bookmarks or public digital images. Resources are tagged by annotating them with simple descriptors. Conjunctions of tags can be used to narrow down the search space and, at the limit, to identify a limited set of resources such as a folder path. Information organization based on tags is capable of overcoming many problems inherent in hierarchical file systems.[16,17] Information about tags can also be represented in an ontology, with the advantage that extensions of the data model and integration with other semantic-aware applications are easy to realize.

The number of tags is likely to grow with the increase of the collection of resources. Hence, information represented by tags cannot be efficiently exploited without a proper user interface (using a laptop or desktop device), or without a further level of semantics, which helps the system take the current intention of the users into account. Key to support mobile users with an efficient access to resources is an intelligent platform that mediates between services and users by observing the user activity.

The research on personalized services has been carried out by many researchers in recommendation systems.[18] In the literature of mobile computing, the use of context information is introduced in terms of implicit input from changes in the environment.[19] This model is usually referred to as context-awareness, because the output of the system depends on who is using the application, where, when, and in which situation. Designing

context-aware applications involves two main steps: (i) designing a set of rules to infer high-level situations and (ii) designing proper input drivers to gather context information from the surrounding environment. Several projects consider the use of ontology as a key technology for context-awareness. In the framework of semantic web, an ontology is a knowledge model that describes a domain of interest using semantic aspects and structure. An ontology consists of: (i) facts representing explicit knowledge, consisting of concepts and their properties, and instances that represent entities described by concepts; (ii) axioms and predicates representing implicit knowledge, by means of rules used to add semantics and to derive knowledge from facts.[20] In Ref. 21, a comparative analysis shows that the most promising assets for context modeling in ubiquitous computing environments can be found in the use of ontology.

To reflect the varying nature of context and to ensure a universal applicability of context-aware systems, context is typically represented at different levels of abstractions.[8] At the lowest level, which takes the raw context sources into account, there are contextual data coming from sensor devices and/or user applications. These contextual data are generally imprecise and vague. For instance, a typical smart phone GPS receiver provides a device position with dynamic accuracy ranging from some meters to hundreds of meters, depending on many environmental variables. Also, the time and location provided by the user's calendar are in practice ideal references only, because real events usually happen approximately at the referred time and place. Nevertheless, logic embodied in semantic languages does not allow managing uncertainty[22] and forces the resolution of uncertainty before the inference process. On the contrary, a situation recognizer should permit to express situations in terms of vague characterizations. Fuzzy logic has proved to be a very effective tool to manage uncertainty by using a very intuitive language.[23]

3. Overall Architecture

SARR runs on the mobile device as an advanced menu, whose elements are dynamically updated, according to the different situations in which the user is involved. The overall system architecture is shown in Fig. 1. In the server side, there are two main modules, i.e., the *fuzzy engine* and the *semantic engine*. The fuzzy engine module is in charge of assessing conditions that are inherently vague, such as mobility and proximity state of users. The domain model and the behavior of the system are instead handled in the semantic engine module, which infers the current situation of the users and suggests the most useful resources for that situation. The *observer* module is in charge of controlling the state of the fuzzy and the semantic engines, allowing the interoperability between these modules.

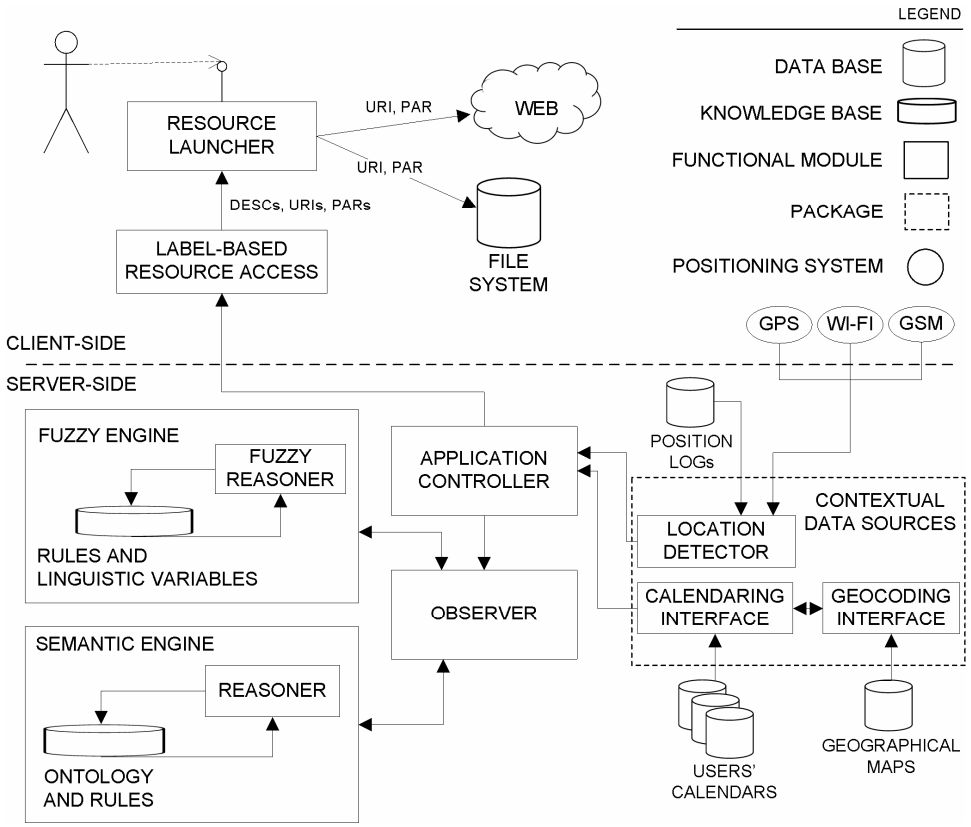

Fig. 1. Overall architecture of SARR.

The control flow of the server-side application is steered by the *application controller* module, which acquires the data collected by the *contextual data sources* block. Whenever a new value is acquired, it is transmitted to the observer, which triggers the fuzzy engine module. This module verifies whether the value belongs with certainty degree higher than 0 to a fuzzy set in the partition corresponding to the linguistic variable which the value refers to. If the certainty degree is higher than 0, the observer inserts the corresponding property into the ontology and triggers the semantic engine. If the semantic rules infer more than one situation, the observer asks the fuzzy engine to assess the final certainty degree for the recognized situations. The certainty degree of a situation is important for considering the order with which resources are recommended. If more than one situation is recognized, all the related resources are recommended, with an order depending on the certainty degrees.

The *contextual data sources* package comprises a set of interfacing modules for different data sources, such as geographical maps, users' calendars and positions.

In particular, numerical data concerning user's movements, i.e., position, time and speed, are fed by the *location detector* module. This module provides outdoor/indoor location estimation, also on the basis of several possible technologies, such as GPS,

GSM, WiFi.[24] Regardless of the available technologies, the *location detector* provides a generalized interface in terms of user movement data and its accuracy. To this aim, the GPX (GPx eXchange format) standard abstraction is used.[25] GPX is a lightweight XML data format, which allows describing waypoints, tracks and routes. More specifically, as regards the location detector, in GPX a collection of time-spatial points is considered as a track. A piece of a simplified GPX track is shown in Fig. 2.

```
<trkpt lat="43.92765" lon="10.915965">
  <time>2009-10-19T7:47:11Z</time>
  <speed>4.671609504842717</speed>
</trkpt>
```

Fig. 2. An example of GPX document representing the user movement.

The *geocoding interface* module is a basic service that provides associated geographic coordinates (expressed as latitude and longitude) from other intelligible textual location data, such as street addresses, or zip codes (postal codes). Intelligible location data comes from the user's calendar, where meetings or other events are recorded by the user. The data format used in this module is imported from the Google Maps API,[26] a web mapping service application. This allows a great interoperability with the client-side simulator, i.e., an auxiliary web application that has been used in the experiments.

The *calendaring interface* module offers time-management services. This module allows users to insert, via mobile application, the events or appointments for each day, which are used as a reference by the system. In particular, the application controller uses the user daily timetable to schedule the specific events to monitor. The calendaring interface module is based on the Google Calendar API,[27] a web application that can be synchronized with the most common mobile devices.

On the client side, the *label-based resource access*[17] module is supplied by the *application controller* module with a set of labels and contextual parameters. This information is used to locate and adapt recommended resources. More specifically, the label-based resource-access module provides an abstraction of the file system with tag semantics.[17] In a traditional file system, a resource is only located within its exact (most specific) path, but not implicitly contained in higher-level directories. For instance, considering pictures, which can be organized by author, genre or date, it allows only one access path, such as "year/author/album" but not "author/album/year". On the contrary, the label-based file system allows for large flexibility, since it allows treating information objects, such as bookmarks, addresses, e-mails and applications, uniformly with respect to metadata.[17] Hence, the specification of a resource becomes a set of labels rather than a URI.

Finally, the selected resource is identified in terms of description, URI and parameters, and can be started by the *resource launcher* module, which is directly connected to the local or web resources.

In the following, we consider the design of the server-side application, focusing on the semantic and fuzzy engines.

4. The Semantic Engine Module

To recommend resources inherent in the current user task, the system takes the current user situation into account. According to Ref. 20, the term "situation" is a business level concept that allows targeting precisely and at different levels of granularity the demand of the user at a certain time. In our system, each situation is devoted to identify a collection of user tasks. In a task-navigation paradigm,[8] the user is supported to find appropriate resources by relying on a task ontology, which represents common sense knowledge about her/his usual activities. In order to suggest in a proactive manner tasks and resources actively, i.e., without the need for initial input from the user, the context is a fundamental vehicle. Context refers to any relevant information that can be used to characterize a user.[28] Therefore, a situation can be modeled as a collection of context information that is invariant as long as the situation occurs.[20] For instance, the situation "meeting" can be inferred from a set of contextual information such as "user is stationary", "user is located in the scheduled place at the scheduled time", "user is close to the meeting organizer", and so on.

Another important advantage of using contextual information is the possibility of deriving contextual parameters to adapt the identified resource to the current demand of the user. Hence, the full goal of the ontology is to identify a set of resource descriptors together with a set of contextual parameters. Furthermore, to make the ontology independent of the specific applications and related path installations, and of the number, type and sequence of parameters, two abstraction mechanisms have been introduced in the system, by means of the following respective modules: the *label-based resource access,* which allows the exact localization of an application or a document, described more generically as a resource in the ontology, and the *resource launcher,* which enables the forwarding of the gathered parameters, and the launching of the selected application.

The semantic engine module exploits two ontologies: the first ontology (*situation ontology*) allows connecting contextual information to situations, and the second one (*task ontology*) allows connecting situations to tasks, and then to specific resources. The ontologies have been developed by using the Web Ontology Language (OWL, see Ref. 10), a W3C standard well supported in most semantic engines.

To develop the ontologies we adopted the following iterative and incremental process.[29] First, we interviewed some domain experts to model some user scenarios and to understand basic domain concepts and relationships among these concepts. Each interview allows producing the narration of a story. After the interview, the narration is formalized, producing a register of sentences, as in the excerpt of Fig. 3 referred to the situation ontology. This register is then processed, with a textual analysis approach. Textual analysis is a process of analysis of a domain which helps to identify the fundamental ontology elements: classes, relations, properties and values. In Fig. 3, nouns, verbs and attributes are highlighted with a different underlining to identify classes,

relations, properties and values. To identify an upper ontology, which is valid for many application scenarios, in the first interviews a bottom-up development process has been employed, starting with the definition of the most specific classes, with subsequent generalization of these classes into more general concepts. Figure 4 shows basic concepts (e.g., *User, Calendar, etc.*) and basic relationships (e.g. *owns, contains, etc.*) identified for the situation ontology. Here, concepts and relationships are represented by oval shapes and directed edges, respectively. In particular, general concepts such as *Time* and *Place* are inherited from publicly available ontologies[30, 31] according to the best practices of reusing domain ontologies. In the figure, external ontologies are enclosed in dashed rectangular shapes.

s_1 : User owns a Calendar;

s_2 : Calendar contains Events;

s_3 : Event can be of two types, whose value can be business or private;

s_4 : Event is attended by a User;

s_5 : Event is organized by (hence, is attended by) User, is located in a Place;

s_6 : Event is scheduled at a Time;

s_7 : User has a name;

s_8 : User owns a Device;

s_9 : User is located in a Place, has a mobility, is close to a User;

LEGEND: class, relation, property, value;

Fig. 3. Excerpt of the register of sentences.

Similarly, we developed also the task ontology. Figure 5 shows the upper task ontology. Here, the dashed edge named "required" represents a property that is not implemented in the ontology, but is conceived only for a better understanding.

In addition to the ontologies, a set of rules is employed to infer the situation. Rules are expressed in the Semantic Web Rule Language (SWRL, see Ref. 11), an emerging standard that extends OWL with additional rule-based knowledge representation. In terms of expressiveness, this reasoning standard corresponds to description logics, a particular decidable fragment of first order logic, and it is named OWL DL.[10]

Figure 6 shows an example of rule in human readable syntax (a), commonly used in the literature, and in natural language (b). We point out that there are two types of antecedent conditions, i.e., crisp (based on two-valued logic) and fuzzy, represented in Fig. 6 in bold and italic bold, respectively. The condition "is a participant" is derived from the user's calendar, and is inherently crisp, whereas the other conditions can be assessed only with vagueness. This implies that also the conclusion inferred from the rule

is characterized by vagueness. Although web ontology is the most promising assets for context modeling for ubiquitous computing,[21] the classical semantic web formalisms do not allow the representation of uncertainty.[32]

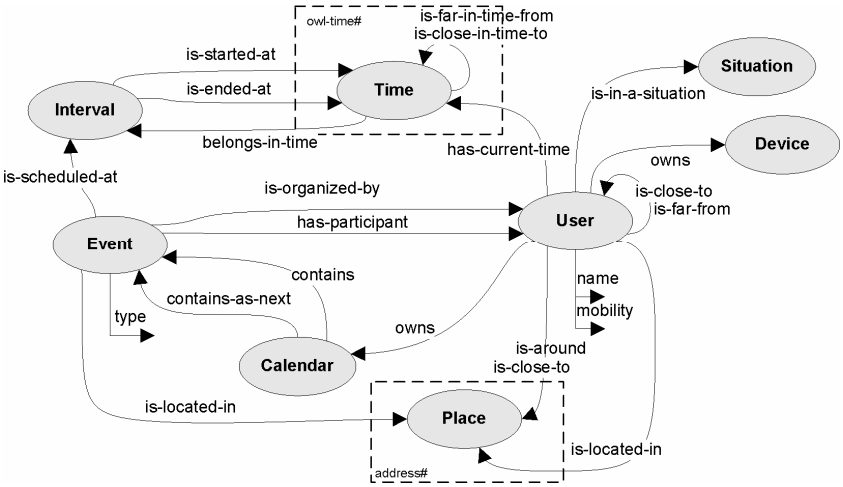

Fig. 4. The situation ontology.

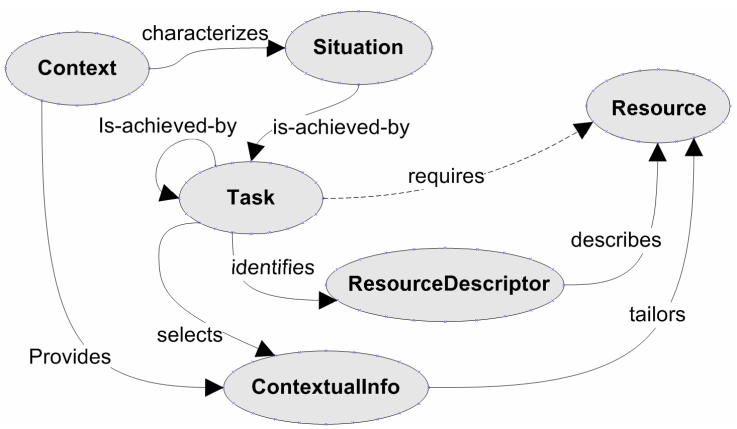

Fig. 5. The task ontology.

As regards fuzzy logic, there has been a significant theoretical work in extending Description Logics with fuzzy set theory.[33,34] Considering also the semantic web perspective, an OWL ontology to represent fuzzy extensions of the OWL language has recently been proposed.[35] With this approach, some reasoning can be performed by using standard OWL reasoners. The ontology can be extended to other fuzzy statements. Ongoing works concern the development of a plug-in for a well-known visual editor,[36] and the implementation of some optimization techniques to reduce the running time.

In our approach, to deal with uncertainty still continuing to use classical semantic web formalisms, we have coupled the semantic engine with a fuzzy engine. Thus, the semantic engine does not handle directly the uncertainty. This approach allows achieving several advantages (see Ref. 37 for more details): (i) there is no need to agree with a non-standard fuzzy ontology to use; (ii) a number of resources is available for standard ontologies, such as ontology editors[36] and public ontologies[38] to reuse; (iii) existing, well-known, widely-used crisp reasoners,[39] and APIs[40] can be used.

```
  owns(?user1, ?aCalendar)
^ contains(?aCalendar, ?anEvent)
^ is-located-in(?anEvent, ?aPlace)
^ is-started-at(?anEvent, ?anEventTime)
^ has-current-time(?user1, ?user1Time)
^ has-participant(?anEvent, ?user2)
^ is-close-in-time-to(?user1Time, ?anEventTime)
^ is-close-to(?user1, ?aPlace)
^ is-far-from(?user1, ?user2)
^ PreMeeting(?aSituation)
⟹ is-in-a-situation(?user1, ?aSituation)
```

(a)

IF *user2* **IS A PARTICIPANT** *to the scheduled event*
AND *user1Time* ***IS CLOSE IN TIME TO*** *the scheduled time*
AND *user1* ***IS CLOSE TO*** *the scheduled place*
AND *user1* ***IS FAR FROM*** *user2*
THEN *user1* ***IS IN A SITUATION*** *of pre-meeting*

(b)

Fig. 6. An example of SWRL rule: (a) human readable syntax and (b) natural language.

The observer module is responsible for integrating the semantic engine and the fuzzy engine. More specifically, when the semantic rules are characterized by fuzzy conditions, the observer asks the fuzzy engine for their evaluation. The fuzzy engine returns a certainty value in [0, 1] for each uncertain condition. If the certainty value is larger than zero, the condition is considered to be true in the semantic inference. Otherwise the condition is considered to be false. When the semantic engine infers a situation, the fuzzy engine, based on these fuzzy conditions, computes a certainty degree for this situation. Thus, the semantic engine using a two-valued logic determines which situation occurs, whereas the fuzzy engine establishes, once a situation has occurred, its certainty degree.

5. The Fuzzy Engine Module

In the system, the fuzzy model is described using the Fuzzy Control Language (FCL) specification.[41] FCL is a standard for Fuzzy Control Programming published by the

International Electrotechnical Commission (IEC). Figure 7 shows an example of linguistic variable defined using FCL. It is worth noting how each term is defined in terms of vertices of a trapezoidal fuzzy set.

```
// Define linguistic variable
FUNCTION_BLOCK distance

  // Define base variable
  VAR_INPUT
     distance : REAL; // meters
  END_VAR

  // Define linguistic values
  FUZZIFY distance
     TERM veryLow := (0,1) (0,1) (200,1) (400,0);
     TERM low := (0,1) (0,1) (400,1) (1400,0);
     TERM high := (20,0) (700,1) (1500,1) (1500,1);
  END_FUZZIFY

END_FUNCTION_BLOCK
```

Fig. 7. An example of linguistic variable expressed in FCL.

Each fuzzy condition is expressed by using linguistic variables declared in FCL. In the system, we have defined a set of linguistic variables to express a series of common contextual conditions. For instance, the condition *"user1* is close *to the scheduled place*" depends on the linguistic variable *distance*. More specifically, the linguistic variable *distance* can assume the linguistic values *veryLow*, *low*, and *high,* as defined in Fig. 7. The state of each fuzzy variable is monitored by the observer module, which, for each variation of the values of the linguistic variables, updates the corresponding properties in the semantic model.

The values of the variables are collected from contextual data sources. To design the linguistic variables, a representative set of contextual data is used. Figure 8 shows an example of GPS track, provided by a smart phone. A user moves from *Q* to *P* to participate to a meeting event. In the fuzzy engine, spatial and temporal proximities are expressed as linguistic variables, let us say Δs and Δt, respectively. The number and meaning of the possible linguistic values for these variables are application-dependent. In our case study, we partitioned the universe of definition of these variables with trapezoidal membership functions, appropriately extracted from experimental data. The use of trapezoidal membership functions helps constrain the number of activated conditions, thus limiting the number of concurrently inferred situations. We adopted the linguistic values [*veryLow*, *low*, *high*] and [*low*, *medium*, *high*] for Δs and Δt, respectively.

In particular, let $(\mathbf{s},t)$ be the reference location and time for the event scheduled in the user's calendar. Let $(\mathbf{s}_1,t_1)$ and $(\mathbf{s}_2,t_2)$ be the current location and time of *user1* and *user2*, respectively, provided by their mobile devices. Let $\Delta t_1 = |t_1 - t|$, $\Delta s_1 = \|\mathbf{s}_1 - \mathbf{s}\|$ and

$\Delta s_{12} = \|\mathbf{s}_1 - \mathbf{s}_2\|$ be the current user temporal proximity and distances. Hence, the fuzzy rule corresponding to the semantic rules in Fig. 6 is

```
IF  Δt1  is LOW AND
    Δs1  is LOW AND
    Δs12  is HIGH
THEN situation1 is Pre-Meeting
```

In the fuzzy engine, we implemented the logical AND and the implication operators as minimum. To allow an efficient integration with the semantic engine, fuzzy rules are processed in the fuzzy engine in a two-stage way. In the first stage, the observer module periodically synchronizes the properties of the semantic model, considering the certainty degrees of the fuzzy conditions in the antecedent part of the fuzzy rules. For each condition with a certainty degree larger than zero, the observer inserts the corresponding property in the ontology and triggers the semantic engine. Hence, at this stage each fuzzy condition is monitored separately in the fuzzy engine. In the semantic engine, the corresponding crisp property is processed in the overall semantic model. Hence, the semantic engine can infer one or more situations. Once the semantic rules have inferred the current situations, in the second stage, the observer asks the fuzzy engine to assess the final certainty degree for the recognized situations. The certainty degree of a situation is important for considering the order with which services are recommended. If more than one situation is recognized, all the related services are recommended, with an order depending on the certainty degrees.

Fig. 8. An example of GPS track, provided by a user smart phone.

6. Case Studies and System Evaluation

We applied SARR to two real business cases, assessing the effectiveness of the semantic engine and fuzzy engine modules.

The first evaluation case study concerns a pharmaceutical consultant in typical business situations. Since the first interviews, we realized that a pharmaceutical

consultant would better benefit from a new generation smart phone, to gain a more effective way of managing messages, of accessing contact data, of opening documents, and of communicating with clients and colleagues during meetings or while traveling. The most common use case is inherent to meetings with medical specialists. Initially, a specific domain ontology has been added to the ontology shown in Fig. 4, by means of a series of interviews. In particular, the situations of interest are: (i) *Meeting-Planning*, when the user is planning the calendar of business appointments; (ii) *Pre-Meeting On Movement*, when the user is going to have a meeting; (iii) *Ongoing-Meeting*, when the user is involved in a meeting; (iv) *Post-Meeting*, when the user has just finished a meeting; (v) *Hospital-Conference*, when the user is giving a scientific talk in a hospital; (vi) *Call-for-Tenders,* when the user is attending a public auction; (vii) *Meal,* when the user is having a meal during the lunch break. For each situation, a set of possible tasks has been defined. For each task, a set of related resources have been characterized in terms of labels and parameters and a domain-specific task ontology has been developed. Figure 9 shows a simplified excerpt of this ontology. The task-driven hierarchy guarantees that the resulting domain model is highly reusable.

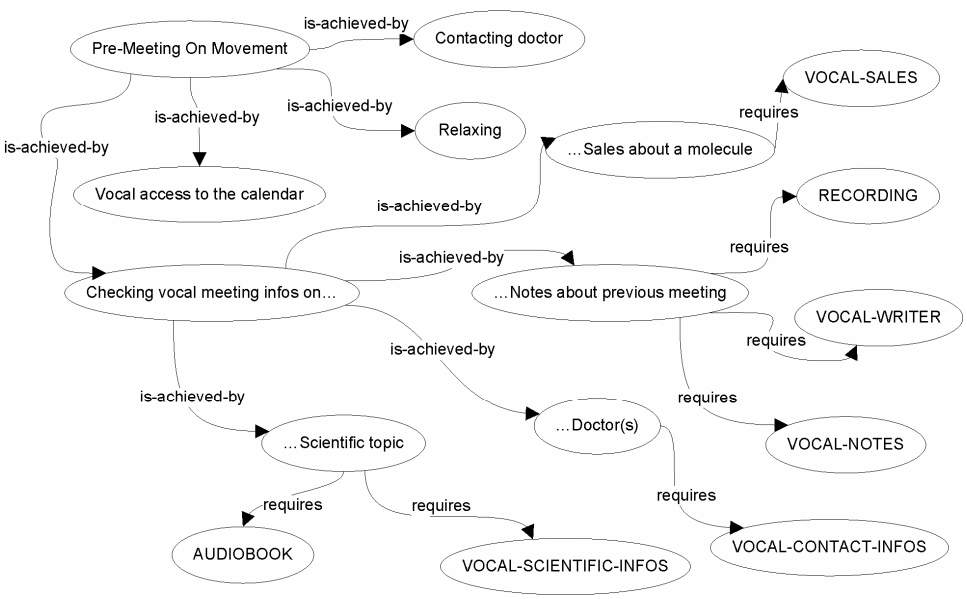

Fig. 9. An excerpt of the task ontology defined for the situation “pre-meeting on movement”.

The explicit specifications of domain knowledge about tasks and resources are useful for new users, who learn what terms in the domain mean, achieving a better understanding about available resources. In fact, we experimentally realized that, once the user has completed the interviews, his demand of mobile services has appreciably increased as shown in Table 1.

Table 1. Number of demanded services per situation by the pharmaceutical consultant, before and after the development of the task ontology.

Situation id	Before interviews	After interviews
i	6	16
ii	5	15
iii	2	8
iv	2	7
v	2	9
vi	2	8
vii	0	3

The second case study concerns an off-site student, who performs a daily travel to go to university and return. The student can be in the following situations: (i) *Pre-University-Day,* when he is leaving his apartment and he is going to take the train; (ii) *Preparing-for-Transportations,* when he is going to the train station or he is waiting for the train at the station; (iii) *Traveling,* when he is heading to some place of interest; (iv) *Studying*, when he is waiting the beginning of the lectures; (v) *Attending-Courses,* when he is attending lectures; and finally (vi) *Meal,* when he is having a meal during the lunch break in the student canteen.

In the use cases, the linguistic variables of the fuzzy engine module have been also tuned. To his aim, for each case study, starting from five real tracks, 30 different training tracks have been generated. To produce real tracks, we used an *Apple iPhone 2G* smart phone, permanently connected to the Internet and to the GPS signal. Training tracks have been generated by means of a client-side simulator, i.e., an auxiliary web application, based on Google-Maps API.[26] The SARR simulator provides new tracks considering calendars provided by real users, and some real tracks used as a reference. This allows producing a number of contextual data sources, in order to test the system in different conditions. Furthermore, the simulator provides an online simulation interface that is used by the user himself, in order to qualitatively assess the effectiveness of the system response. More specifically, the trace simulator can also produce tracks with different circumstances, such as different means (by foot, bicycle and car), traffic conditions (without/with traffic jam), environments (indoor and outdoor), event types (formal and informal, periodic and unique events), etc. Noise is also introduced to make contextual sources very close to real world signals. Figure 10 shows the user interface of the SARR simulator during a batch simulation. In particular, some conditions can be noted in the configuration area, such as *transport: walk*, and *traffic: none*.

The SARR client has been implemented and tested for the *iPhone OS*. In the following, we first illustrate the client-side user interface, and then we show an example of interaction of a user with the SARR client by using a simple scenario. The client-side application has been thought of as an intelligent resource launcher, with the possibility of performing proactive and parameterized launches. Figure 11 shows the user interface for the off-site student business case in a specific situation of the simulation scenario of

Fig. 10. On the top of the interface, the first header contains the name of the user who is currently logged in through the device. The second header shows the current situation, e.g. *Traveling*. The main menu is structured as a series of contextualized items, representing tasks and resources, ordered according to the ranking determined by the fuzzy engine. The proposed items are dynamically changing, according to the recognized situations and to the certainty degree calculated for each situation. Finally, the items are implicitly parameterized/personalized, thanks to the information available from the user context, as detailed in Sec. 3.

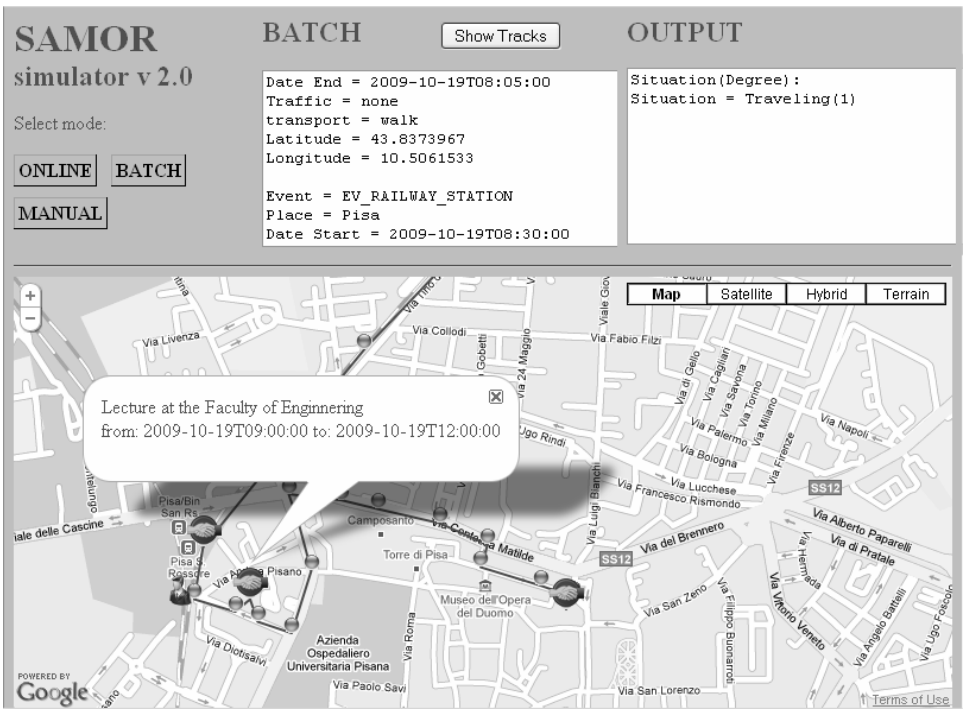

Fig. 10. The SARR simulator: the off-site student case study.

More specifically, to catch the proactivity of the launcher, let us consider the following excerpt of an off-site student scenario. *Alan, a university student, leaves his home in Lucca in the early morning, and catches the train to go to Pisa. While the train is arriving at the Lucca railway station, the student accesses the latest news. During the travel, Alan would like to revise the slides of the next lesson. While the train is arriving at the Pisa railway station, he checks the course timetable for knowing the classroom of the next lesson. When he is close to the classroom, he checks for classmate messages.* According to this scenario, the SARR server is made available with proper situation and task ontologies, together with specific instance-related information for the user and the preferred resources. For instance, in the example of Fig. 11, the system recognizes the situation *Traveling* because, as shown in Fig. 10, the user is moving towards the faculty

and is close in time to the scheduled lesson. Thus, the client application proposes specific resources (such as Alan's course timetables and his lesson slides) that are also implicitly parameterized in terms of the context (e.g., next appointment and current location).

Similar to the off-site student business case, SARR can be configured for the pharmaceutical consultant business case, employing the task ontology of Fig. 9, and related situation rules. In this case, there are different situations and resources that will be managed by the user interface. However, from the user's perspective, the interaction in terms of resource recommendation and selection is very similar to the off-site student scenario.

Fig. 11. An example of the SARR client interface, for the student case study.

After the tuning process, the system has been tested by a user for each case study. In particular, for each user, a week timetable has been considered, consisting of 51 and 49 events for the pharmaceutical consultant and the off-site student, respectively.

During the experimentation, for each event, we simulated different conditions, by considering events delayed or anticipated with respect to the timetable in the user's calendar. We assessed that SARR is able to provide a reliable and timely recommendation, during the period in which the situation effectively occurs. Let us consider a generic situation S_i, i = 1,.., N, the starting time t_i at which that situation occurs, and the time t_i' at which the system recognizes the situation. Let us define the *responsiveness* of the system as the mean of the absolute deviation between t_i and t_i', i.e., $Resp = \sum |t_i' - t_i| / N$. Table 2 shows the responsiveness of the system for each situation occurred during the testing.

Table 2. Responsiveness of the system.

case study	Situation (Event)	Resp (minutes)
Pharmaceutical consultant	Ongoing-Meeting (begin)	1.08
	Ongoing-Meeting (end)	4.98
	Meal (begin)	0.62
	Meal (end)	0.37
	Hospital Conference (begin)	6.83
	Hospital Conference (end)	0.63
Off-site student	Preparing-for-Transportations (begin)	2.28
	Traveling (begin)	5.77
	Traveling (end)	7.75
	Attending-Courses (begin)	3.91
	Attending-Courses (end)	0.77

We deduce from the table how SARR allows proposing resources within a range of few minutes with respect to the time at which the related events happen. We experienced that, once tuned, fuzzy variables can be reused for a number of different purposes. We are also experiencing the use of linguistic hedges to adapt this fuzzy inference system to different user needs. Finally, we would like to observe that the responsiveness of SARR can be improved considering more specific fuzzy conditions, which, however, would be less reusable.

Recently, other context-aware recommenders have been proposed in the literature. Luther *et al.*[8] have integrated a situational reasoning engine into a mobile service recommendation, using an approach based on the standard representation language OWL. Weißenberg *et al.*[20] have proposed a demand-driven personalized service recommender, based on user profiles, semantic service, context- and situation-awareness. Goix *et al.*[9] have introduced a rule-based approach for inferring situations of mobile users, considering context data collected from heterogeneous and distributed sources. Unlike SARR, all the three approaches do not consider the inescapable uncertainty that affects contextual data in order to infer the correct user situation. It follows that these approaches cannot adequately manage concurrent situations. Further, they cannot handle the gradual recognition of a situation. Since the papers which introduce the three recommenders propose no evaluation of them in terms of responsiveness, no comparison is possible with respect to this dimension. However, when we added the fuzziness to SARR, we verified that the gradual recognition of situations had considerably increased the capability of SARR to react proactively. Thus, we expect that SARR may outperform the other recommenders in terms of responsiveness.

Also, in the situation inference process, the four recommenders use different approaches. In Luther *et al.*'s recommender, the situation inference is performed by applying dynamic assertional classification of contextual entities such as the location, the time and the neighbour people. Classification is carried out directly into OWL DL by

subsumption. In Weißenberg *et al.*'s recommender, situation inference is implemented by means of F-Logic.[42] a formalism that allows complex rules with high-level of expressiveness. However, F-Logic is generally undecidable. In Goix *et al.*'s recommender, context is modelled by ContextML, a proprietary XML-based Context Markup Language, whereas situation inference is performed by RuleML,[43] an XML-based standard language to tackle the much broader problem of rule interchange. In SARR, as we have already pointed out, situation inference is carried out by means of SWRL, a OWL-specific standard language that provides a formally sound way of inferring information in OWL ontologies, offering an officially standardized rule formalism for the Semantic Web.

Finally, in SARR, the context model is separated in upper and domain-specific ontologies. This approach enhances reusability in different domains, enabling a truly general-purpose recommender, easy to adapt to different use scenarios. Among the compared systems, only the Weißenberg *et al.*'s recommender employs a similar approach to context modelling.

7. Conclusions

In this paper, a situation-aware mobile resource recommender has been proposed. The study comprises an analysis of the service recommendation problem, considering how it can be improved using the context-aware paradigm. An overall architecture is presented, in which fuzzy logic and web ontology can be efficiently integrated thanks to an intermediate module based on the observer pattern. The paper focuses on two important modules, i.e., the fuzzy engine, which analyzes real-world inaccurate information, and the semantic engine, which contains the resource recommendation ontology and the related semantic rules. A methodology for designing domain ontologies for resource recommenders is also presented. Finally, real evaluation case studies are provided, together with a comparison with other systems proposed in the literature, to give a concrete and comparative view of the system, and to assess its reliability and responsiveness.

Acknowledgements

The presented work was supported by the MOVAS Lab, a joint project at the University of Pisa between academy and industry. The authors would like to thank the company Softec S.p.a. for financial and technical support.

References

1. Apple Inc., App Store, www.apple.com/iphone/ appstore, accessed Jan 2010.
2. Google Inc., Android Market, www.android.com/ market, accessed Jan 2010.
3. Microsoft Corp., Windows Mobile Catalog, www.microsoft.com/windowsmobile /catalog, accessed Jan 2010.
4. Nokia Corp., Ovi Store, store.ovi.com, accessed Jan 2010.

5. Research In Motion Limited, BlackBerry App World, www.blackberry.com/appworld, accessed Jan 2010.
6. K. Heinonen and M. Pura, Classifying mobile services, in *Proc. Helsinki Mobility Roundtable, Working Papers on Information Systems,* Sprouts, Vol. 6, Art. 42, 2006.
7. S. Figge, Situation-dependent services — a challenge for mobile network operators, *Journal of Business Research*, Elsevier, Vol. 57, Issue 12, 2004, pp. 1416–1422.
8. M. Luther, Y. Fukazawa, M. Wagner and A. Kurakake, Situational reasoning for task-oriented mobile service recommendation, *The Knowledge Engineering Review,* Cambridge University Press, Vol. 23, 2008, pp. 7–19.
9. L. W. Goix, M. Valla, L. Cerami and P. Falcarin, Situation inference for mobile users: a rule based approach, in *Proc. Int. Conf. on Mobile Data Management* (*MDM '07*), IEEE Computer Society, 2007, pp. 299–303.
10. W3C, OWL Web Ontology Language Reference, W3C Recommendation, 10 February 2004, electronically available at http://www.w3.org/TR/owl-ref/.
11. W3C, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission 21 May 2004, electronically available at http://www.w3.org/Submission/SWRL/.
12. Y. L. Tai, S. R. Tsai, G. H. Huang, C. M. Lee, L. J. Tsai, K. F. Ssu and S. J. Wey, A label-based file system, *Journal of Computers,* Vol. 19, No. 4, January 2009.
13. D. Abrams, R. Baecker, and M. Chignell, Information archiving with bookmarks: personal web space construction and organization, in *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems* (*CHI '98*), ACM Press, 1998, pp. 41–48.
14. R. Kanawati and M. Malek, Cowing: a collaborative bookmark management system, in *Proc. Int. Workshop on Cooperative Information Agents* (*CIA '02*), Springer-Verlag, Lecture Notes in Artificial Intelligence 2828, 2001, pp. 34–39.
15. S. Nakajima, S. Oyama, K. Sumiya and K. Tanaka, Context-dependent web bookmarks and their usage as queries, in *Proc. Int. Conf. on Web Information Systems Engineering* (*WISE '02*), Systems Engineering, 2002, pp. 333–341.
16. P. K. Vatturi, W. Geyer, C. Dugan, M. Muller and B. Brownholtz, Tag-based filtering for personalized bookmark recommendations, *Proc. 17th ACM Conference on information and Knowledge Management* (*CIKM '08*), ACM, 2008, pp. 1395–1396.
17. S. Bloehdorn, O. Görlitz, S. Schenk and M. Völkel, TagFS, tag semantics for hierarchical file systems, in *Proc. 6th Int. Conf. on Knowledge Management,* (*IKNOW' 06*), 2006.
18. J. Hong, E.-H. Suh, J. Kim and S. Y. Kim, Context-aware system for proactive personalized service based on context history, *Expert Systems with Applications,* Elsevier, Vol. 36, No. 4, 2009, pp. 7448–7457.
19. F. A. Hansen, Context-aware mobile hypermedia: concepts, framework, and applications, PhD Thesis, University of Aarhus, 2006.
20. N. Weißenberg, An ontology-based approach to personalized situation-aware mobile service supply, *GeoInformatica*, Vol. 10, No. 1, Springer, 2006, pp. 55–90.
21. T. Strang and C. Linnhoff-Popien, A context modeling survey, in *Proc. First Int. Workshop on Advanced Context Modelling, Reasoning and Management* (UbiComp '04), 2004.
22. E. Sanchez and T. Yamanoi, Fuzzy ontologies for the semantic web, H. L. Larsen *et al*. (eds.), FQAS 2006, LNCS, Vol. 4027, pp. 691–699, Springer, Heidelberg, 2006.
23. L. A. Zadeh, Is there a need for fuzzy logic? *Information Sciences*, No. 178, 2008, pp. 2751–2779.
24. G. Sun, J. Chen, W. Guo and K. J. R. Liu, Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs, *IEEE Signal Processing Magazine,* 2005, Vol. 22, No. 4, pp. 12–23.
25. Topografix, GPS Exchange Format, official website, http://www.topografix.com/gpx.asp, accessed Jan 2010.

26. Google Inc., Google Maps API, http://code.google.com/apis/maps/, accessed Jan 2010.
27. Google Inc., Google Calendar API, http://code.google.com/apis/calendar/, accessed Jan 2010.
28. A. K. Dey and G. D. Abowd, Towards a better understanding of context and context-awareness, in *Proc. Workshop on the What, Who, Where, When and How of Context-Awareness,* ACM Press, 2000.
29. M. Fernández López, Overview of methodologies for building ontologies, in *Proc. Workshop on Ontologies and Problem-Solving Methods* (KRR5), Stockholm, Sweden, August 2, 1999.
30. W3C, Public Time Ontology, date, http://www.w3.org/TR/owl-time, accessed June 2009.
31. L. Ding, H. Chen, L. Kagal and T. Finin, Public Address Ontology, http://daml.umbc.edu/ontologies/ittalks/ address, accessed June 2009.
32. F. Van Harmelen, Fuzzy logic and the semantic web, E. Sanchez (eds.), *Capturing Intelligence* (Elsevier, 2006).
33. T. Lukasiewicz and U. Straccia, An overview of uncertainty and vagueness in description logics for the semantic web, Technical Report INFSYS RR-1843-06-07, Institut für Informationssysteme, Technische Universität, Wien (2006).
34. F. Bobillo and U. Straccia, FuzzyDL: An expressive fuzzy description logic reasoner, in *Proc. 2008 International Conference on Fuzzy Systems* (FUZZ-IEEE 2008), Hong Kong, June 2008.
35. F. Bobillo and U. Straccia, An OWL Ontology for fuzzy OWL 2, J. Rauch *et al*. (Eds.): ISMIS 2009, LNAI 5722, pp. 151–160 (Springer-Verlag, Berlin, Heidelberg, 2009).
36. Stanford Center for Biomedical Informatics Research, Stanford University School of Medicine, Protégé OWL and SWRL editor, http://protege.stanford.edu, accessed Jan 2010.
37. F. Bobillo, M. Delgado and J. Gomez-Romero, Crisp representations and reasoning for fuzzy ontologies, in *Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 17, No. 4, 2009.
38. University of Maryland, Baltimore County (UMBC), Swoogle, semantic web search, http://swoogle.umbc.edu/, accessed Jan 2010.
39. Clark&Parsia, Pellet: OWL 2 Reasoner for Java, http://clarkparsia.com/pellet/, accessed Jan 2010.
40. HP Labs Semantic Web Programme, Jena – A Semantic Web Framework for Java, http://jena.sourceforge.net/, accessed Jan 2010.
41. P. Cingolani, JFuzzyLogic, a Java package that implement Fuzzy Control Language (FCL) specification (IEC 1131p7), http://jfuzzylogic.sourceforge.net, accessed Jan 2010.
42. M. Kifer, G. Lausen and J. Wu, Logical foundations of object-oriented and frame-based languages, *J. ACM*, Vol. 42, No. 4, 1995.
43. The Rule Markup Initiative, RuleML, http://ruleml.org/, accessed April 2010.