An adaptive rule-based approach for managing situation-awareness

Mario G.C.A. Cimino [a,*], Beatrice Lazzerini [a], Francesco Marcelloni [a], Alessandro Ciaramella [b]

[a] Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni, University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy
[b] IMT Lucca Institute for Advanced Studies, Piazza San Ponziano 6, 55100 Lucca, Italy

ARTICLE INFO

Keywords:
Situation awareness
Context awareness
Fuzzy rules
Semantic Web
Rule-based system

ABSTRACT

Situation awareness is a powerful paradigm that can efficiently exploit the increasing capabilities of handheld devices, such as smart phones and PDAs. Indeed, accurate understanding of the current situation can allow the device to proactively provide information and propose services to users in mobility. Of course, to recognize the situation is a challenging task, due to such factors as the variety of possible situations, uncertain and imprecise data, and different user's preferences and behavior.

In this framework, we propose a robust and general rule-based approach to manage situation awareness. We adopt Semantic Web reasoning, fuzzy logic modeling, and genetic algorithms to handle, respectively, situational/contextual inference, uncertain input processing, and adaptation to the user's behavior. We exploit an agent-oriented architecture so as to provide both functional and structural interoperability in an open environment. The system is evaluated by means of a real-world case study concerning resource recommendation. Experimental results show the effectiveness of the proposed approach.

1. Introduction

Situation-awareness (SAW) is a computing paradigm by which applications can sense and comprehend the user's *situation* in order to anticipate or predict her/his demand (Weißenberg, Gartmann, & Voisard, 2006).

The fundamental means to achieve situation awareness is the *context*, i.e., all relevant data and information (e.g., the user's position in space and time, the surrounding things and events, etc.) that help the decision maker to understand what is happening in the environment and then take more informed decisions (Anagnostopoulos & Hadjiefthymiades, 2008; Coutaz, Crowley, Dobson, & Garlan, 2005; Dey, 2001).

The various approaches available in the literature on context and situation awareness have produced a plethora of domain-specific frameworks, each characterized by proper sensors and databases (Gu & Pung, 2005). Usually, contextual data sources are provided via Service-Oriented Architecture (*SOA*) interfaces, and enriched with a domain *context ontology*, which is a powerful formalism to model context information, enabling context reasoning and context knowledge sharing among several, different applications (Bettini et al., 2010; Gu & Pung, 2005).

Of course, gathering appropriate and easily understandable information is particularly challenging in a mobile context where the workspace dynamically and rapidly changes, and data from multiple and heterogeneous sources may be uncertain and partially true (Korpipää, Mantyjarvi, Kela, Keranen, & Malm, 2003).

To this aim, Korpipää et al. (2003) have proposed a framework for managing uncertainty in raw data and inferring higher-level context abstractions with a related probability. The framework uses the shared blackboard metaphor to enable communications among entities in the system. All context sources publish their data in the blackboard, which acts as a centralized module to process contextual data and deliver high-level information, i.e., the user's situation, to the application. Fuzzy sets are employed to convert unstructured raw data into a representation defined in a context ontology by means of predefined fuzzy labels. A confidence value is associated with contextual data to describe the context uncertainty. Situations are recognized by means of a basic Bayes classifier, which learns conditional probabilities from training data for each situation. Mäntyjärvia and Seppänen (2003) have proposed a system to represent context information by applying fuzzy membership functions. In particular, raw data from sensors are converted into context information by means of fuzzy quantization. Such information is then employed as input to fuzzy rule-based controllers to adapt applications according to the context. However, no semantic description of context is considered. Ranganathan, Al-Muhtadi, and Campbell (2004) have modeled uncertainty in situation awareness by associating a confidence value with all pieces of contextual information. The authors adopt three methods to infer the user's situation, (i) probabilistic logic, (ii) fuzzy logic, and (iii) Bayesian networks. In the probabilistic and fuzzy approaches, developers have to write down their own rules to infer situations, whereas in

* Corresponding author. Tel.: +39 050 2217455; fax: +39 050 2217600.
E-mail addresses: m.cimino@iet.unipi.it (M.G.C.A. Cimino), b.lazzerini@iet.unipi.it (B. Lazzerini), f.marcelloni@iet.unipi.it (F. Marcelloni), a.ciaramella@imtlucca.it (A. Ciaramella).

the Bayesian approach developers have to define the network specifying the relations among contextual information. Gu, Pung, and Zhang (2004) have proposed a context-aware middleware to support context reasoning in order to derive the user's situation. Uncertainty is faced in two manners. First, the context ontology is extended to allow additional probabilistic markups. Second, Bayesian networks are adopted to support the inference process of the user's situation. In Cao, Xing, Chan, Yulin, and Jin (2005), the user's situation is assessed as a combination of context information which is expressed by fuzzy linguistic variables. More specifically, a situation is represented by a set of three-element tuples, where each tuple contains certain contextual information (e.g., the current network rate), a linguistic value that characterizes that situation (e.g., high), and finally the fuzzy membership degree of the contextual information to the linguistic value. Thus, the recognized situations contain a list of fuzzy degrees referred to several linguistic values and it is difficult to compare situations with each other and to rank them. Haghighi, Krishnaswamy, Zaslavsky, and Gaber (2008) have proposed an approach for situation modeling and reasoning under uncertainty based on fuzzy theory. Situations are expressed by multiple contextual conditions joined in a fuzzy rule, where the consequent represents the degree of confidence in the occurrence of a situation. Moreover, developers can specify weights to represent the relative importance of each contextual condition for inferring a situation.

One of the main problems with the above approaches is that the relationship between contextual information and situations is static, and cannot be adapted to the changing user's behavior. Indeed, as formerly pointed out by Byun and Cheverst (2003), when context awareness is reached by means of predefined rules, users have to reconfigure the system when their behavior changes, resulting in a frustrating and annoying task. In order to automatically recognize the user's situation related to the user's behavior, the authors propose to exploit context history. Adaption is provided by fuzzy decision trees, which take uncertainty in the raw data into account. More recently, Hagras, Doctor, Callaghan, and Lopez (2007) have proposed a novel learning technique to adapt the system to the continuous changing in the user's behavior. The technique is an unsupervised data-driven one-pass approach for extracting type-2 fuzzy membership functions and rules from the context history of the user. However, the authors do not separate the *situation determination* phase from the *system response* phase, on the basis of the inferred situation. Indeed, the sensed contextual information is immediately employed to adapt the system to the user's needs, which are application-specific. For instance, a particular configuration of some sensor values such as internal light level, bed pressure, and internal temperature can lead to activate the window blinds. Thus, the concept of situation is lacking in the system. Anagnostopoulos and Hadjiefthymiades (2010) have introduced advanced semantics in the context representation, combining the fuzzy logic approach with the semantic one. In particular, advanced representation schemes concern specialization, mereonomy, mutual exclusion and compatibility. By means of a neuro-fuzzy classification engine, the system learns to map sets of contextual information to particular situations, and builds the corresponding fuzzy rules. However, the proposed system is capable of dealing with physical contextual information only, such as orientation of the mobile device, illumination level, humidity. It does not deal with other indirect contextual information, such as user calendars or geographical maps. Finally, explicit means of representing user's situations are also lacking. The logical mapping between contextual information and situations should be accessible at the knowledge level, to be easily inspected and edited by the user or by an analyst.

This paper tries to address the question of how to provide a robust and general approach to situation awareness, in which both system architecture and behavioral knowledge can be easily integrated in an open environment, by supporting a variety of contextual, possibly uncertain, inputs and providing situational knowledge to multiple applications. To guarantee such structural and functional interoperability, the proposed system has been designed in compliance with an *agent-oriented* approach (Greenwood, Lyell, Mallya, & Suguri, 2007), which operates at the knowledge level, shows flexible behavior, easy maintenance, reusability and platform independence. This is achieved thanks to the use of highly standardized technologies, such as Semantic Web and Approximate (Fuzzy) Reasoning (Wang, Ma, Yan, & Meng, 2008), as well as architectural patterns such as the Event-Control-Action (ECA, Costa, Pires, & Sinderen, 2005; Etter, Costa, & Broens, 2006).

More specifically, the basic behavioral model of the proposed situation agent is expressed in terms of condition rules. The antecedent part of each rule is made of a logical combination of contextual conditions, referred to as an event in ECA pattern. The corresponding consequent part models a reaction to the event, assessing the new user's current situation and delivering the related results to an external service. Hence, the transitions of the user's current situations are established by the rule base, which represents the *control* of the situation-aware system.

Of course, automated context reasoning has to deal with events that may occur gradually and conditions that are inherently vague and imprecise. To this aim, fuzzy logic has been adopted in order to associate a truth degree with each context condition. If more than one situation is inferred from a given condition, a certainty degree for each such situation is computed based on the truth degrees of the condition. In this way the inferred situations can be ranked on the basis of their certainty degrees. Finally, situations are associated with a set of relevant tasks that the user would perform in the specific situation (Luther, Fukazawa, Wagner, & Kurakake, 2008), by means of domain knowledge expressed through task ontologies.

In general, context awareness and situation awareness rely on a distributed system. Hence, knowledge portability, integration and extensibility are key features since context reasoning implies collaboration among software agents that manage their own contextual sources. For this purpose, our proposal employs web knowledge standards such as Semantic Web Rule Language (SWRL, Horrocks & Patel-Schneider, 2004) and Fuzzy Markup Language (FML, Acampora & Loia, 2005).

Finally, rule bases are usually created by domain experts that model the situations in which the average user can be involved, and her/his behavior. However, users have different habits that may affect the way in which situations arise. Thus, an appropriate tuning aimed at adapting the situation model to the specific user is desirable. In our approach, this can be automatically achieved by using the context history and exploiting genetic algorithms.

The paper is structured as follows. In Section 2 we present the adopted architectural pattern (called ECAA in the following); Section 3 provides a detailed description of the structure of the proposed general-purpose situation reasoner (called GEPSIR in the following). Section 4 concerns the design of the Fuzzy Context Ontology. Section 5 discusses the representation languages employed in the model. Sections 6–8 are devoted to describe in detail the modules of the proposed architectural pattern. Section 9 presents an evaluation case study and, finally, Section 10 draws some conclusions.

2. The extended architectural pattern ECAA

According to the ECA pattern (Costa et al., 2005), the basic situational model can be expressed in terms of condition rules, which have the form

if < *context conditions* > **then** < *situation* > .

The antecedent part is made by a logical combination of conditions expressed in terms of contextual variables which represent basic data or sensor samples. For instance, a sensor that detects whether a person enters a laboratory, a recognition card that the person carries with her/him. Such combined conditions model some happening of interest in the application domain or in its environment. This happening is referred to as an *event* of interest in the ECA pattern. For instance, an occurrence of the above mentioned context conditions can mean that "Tom enters the laboratory". It is worth nothing that the same event could be alternatively represented in terms of other context conditions, e.g. based on the output of a facial recognition system.

In our approach, a user's new situation can be possible only when new events occur, according to the condition rules. Hence, the transitions of the user's current situations are established by the rule base, representing the *control* of the situation-aware system.

Finally, the consequent part of each rule models a reaction to context information changes, assessing the user's new current situation and delivering the related results. In terms of the ECA pattern, this part is referred to as an *action*, because it is responsible for an outcome which affects an external module (e.g., a situation-aware actuator or service). In our system, the action is made of both the generation and the delivery of a notification. The generation of the notification is a knowledge-level task, modeled via the assessment of the user's situation, as established in the consequent part of the transition rule. The delivery of the notification is a transport-level task, modeled in a domain-independent manner according to service oriented architecture standards. Thus, the ECA pattern can reflect the inferential nature of context-aware processing, whose behavior can be entirely expressed through modular rules with a high interoperability.

In this paper we propose an extension of the ECA architectural pattern. The extended pattern, named ECAA (Event Control Action Adaptation), provides a high-level structure that helps in the design of situation-aware applications, solving recurring problems associated with the management of situation information upon context changes. As previously stated, in our approach an application model defines the behavior via condition rules. Fig. 1 shows the structure of the ECAA pattern, its top modules and the relationships among them. Here, solid and dashed boxes represent functional modules and information, respectively. The fundamental components are the *Context Processing*, the *Reasoning Engine*, the *Situation Assessment*, and the *Behavior Tuning* modules. The pattern reflects the proactive nature of context-aware applications as well as the necessary customization to fit user's characteristics. The *Context Processing* module detects events that occur in the context source and are significant to fire one or more rules; the *Reasoning Engine* module monitors rule conditions and triggers update actions when a condition is satisfied; this module is provided with high-level behavior descriptions (condition rules); the *Situation Assessment* module implements the actions for the situation ranking, identification, and notification.

Each rule in the *Behavior Description* knowledge base is expressed by using linguistic variables. For each linguistic variable, we define a set of linguistic values and associate a fuzzy set with each value. The fuzzy sets describe the meaning of the linguistic values. This meaning is generally fixed by considering a generic user in the domain ontology. Actually, different users may have different behaviors. Thus, in order to take the specificity of each user into account, we use the context history of the specific user and adopt a genetic algorithm for adapting the meaning of the linguistic values used in the rules. This process is managed by the *Behavior Tuning* module.

3. The software architecture of GEPSIR

Fig. 2 presents a generic software architecture for managing situation awareness. The application controller of a situation-aware service (represented with a gray box) accesses the user's current situation/context via the *Situation Reasoner* (SR) module, a domain-independent web service which steers the control flow of the reasoning, based on the ECAA pattern. More specifically, the SR (i) gathers data coming from contextual sources, available via a generic SOA interface, (ii) loads the knowledge base (domain ontologies, rules and linguistic variables), (iii) infers the situation once contextual conditions are assessed, (iv) updates the situation/context history for the tuning of linguistic variables.

Raw contextual sources are processed by specialized modules (represented on the right in the figure). This low-level processing is abstracted by the *Context Source* (CS) module. The activities provided by the specialized CS module include: (i) *sensing*: gathering source information from sensor devices, e.g., location information (latitude, longitude, speed) from a GPS device; (ii) *aggregating* (or fusion): observing, collecting and composing context information from various providers, e.g., combining location information from GPS and RFID devices, for outdoor and indoor sensing, respectively. The activities provided by the SR module include: (i) *inferring*: interpretation of context information in order to derive another type of context information, the situation information; (ii) *focusing/instantiation*: the projection of context information on given instances and on given situations, e.g., focusing on the temporal window represented by the next appointment in the user's agenda.

The above architecture can be used as a modular engine for open environments. We can chain multiple instances of it in a

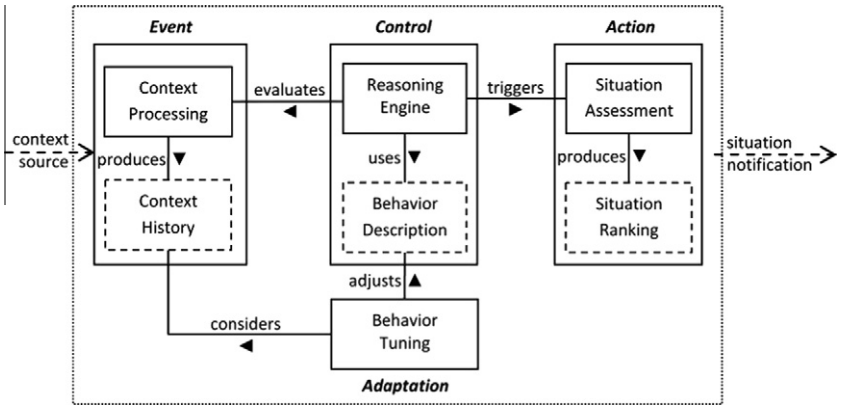


Fig. 1. ECAA, an extension of the Event-Control-Action pattern for situation awareness.

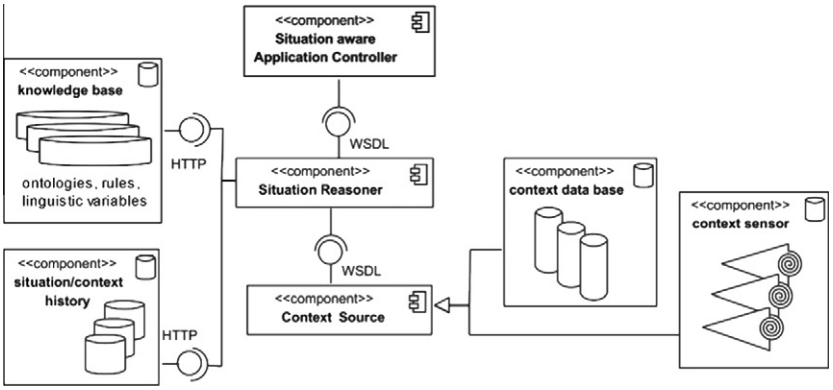


Fig. 2. Architectural support for a general purpose situation reasoner.

hierarchy of SRs. Indeed, there could be many inference levels in situation processing. Thus, one can envision many abstraction levels, each supported by an inference level. As shown in Fig. 3, in this chain of situation processing, the outcome of an SR unit becomes input to a higher level SR unit in the hierarchy, until a top (application) level is reached. The two types of components, CS and SR, follow a hierarchy pattern defined by Costa et al. (2005). CS components encapsulate single domain sensors, such as time, position, speed. Hierarchical chains of CS and SR can be represented as a directed acyclic graph, in which a CS can be only an initial vertex of the graph, whereas an SR can be only an intermediate or final vertex. In Fig. 3, the directed edges of the graph, concerning context and situation information flows between units, are represented with dashed and solid lines, respectively. Hence, multiple levels of inference can be realized by connecting reasoners with a single level of inference, according to a knowledge distributed (agent-based) approach.

Let us consider more specifically the components of the SR module, as depicted in Fig. 4. The *Situation Observer* is responsible for observing context changes sensed by CS components and, as a consequence of these changes, for triggering actions that produce the situation assessment, according to the behavior description. The behavior description consists of *Ontology*, *Linguistic Variables* and *Rules* that describe the situation in terms of context rules. A rule engine (or reasoner) is able to process the knowledge base and fire the rules. In Fig. 4, two different reasoners have been included, namely the *Semantic* reasoner and the *Fuzzy* reasoner, whose activities are coordinated by the *Situation Observer*. More specifically, upon the occurrence of context conditions sensed via the *Context Source*, the *Situation Observer* has the capability to check the knowledge base for rules which are fired by these changes, and to coordinate the execution of both engines.

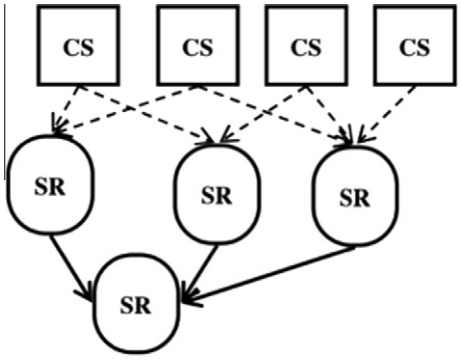


Fig. 3. A hierarchy of context sources (CSs) and situation reasoners (SRs).

The produced situations are then used by a situation-aware application to adapt its services, and recorded by the *Context History Sampler*. *Context History* is employed to adapt to the specific user the meaning associated with the linguistic terms used in the linguistic variables, via the *Linguistic Tuner*.

The above architectural view is introductory for a detailed view of the functional modules, discussed in the next sections and aimed at proposing a generic scheme for the solution of situation awareness problems. The scheme contains components, their responsibilities, their relationship, as well as exhibits some desirable properties, such as: (i) to provide a common vocabulary and understanding for design principles; (ii) to offer a means for specifying situation-aware architectures; (iii) to support the construction of software agents with defined properties; (iv) to build complex and heterogeneous situation-aware middlewares; (v) to help managing software complexity. The solution has been devised to decouple contextual management issues, such as context sensing and processing, from situational reasoning upon context changes.

4. The design of the knowledge base via the Fuzzy Context Ontology

Building a domain ontology is an expensive task, since it is difficult to extract relevant knowledge from the problem domain and to express it in a proper manner. Generally, an initial prototype is developed based on explanatory text and/or interviews with domain experts. Then the prototype is refined, by testing and modifying it in collaboration with the experts and the potential users of the system.

The context ontology is located at the core of the SR, allowing connecting contextual information to situations. In the design of the context ontology, we faced with two main issues: (i) to represent both general and specific information in the knowledge base of the system; (ii) to represent fuzzy information within the Web Ontology Language. The first issue has been tackled according to a design pattern suggested by Gu, Pung, and Zhang (2005). More specifically, the context ontology has been split into an *upper* ontology, valid for general concepts, and a set of *lower* (domain-specific) ontologies, which detail the upper ontology describing low-level concepts and their properties. In this way, the upper ontology is valid for many application scenarios and can be considered as a stable reference to be specialized for the different domains. Fig. 5 shows the upper context ontology employed in the system. Here, ovals and arrows represent classes and properties, respectively. The upper ontology has been designed in a bottom-up manner, starting from the basic domain concepts and their relationships as highlighted via interviews to domain experts. For a more detailed description of the process followed in the ontology

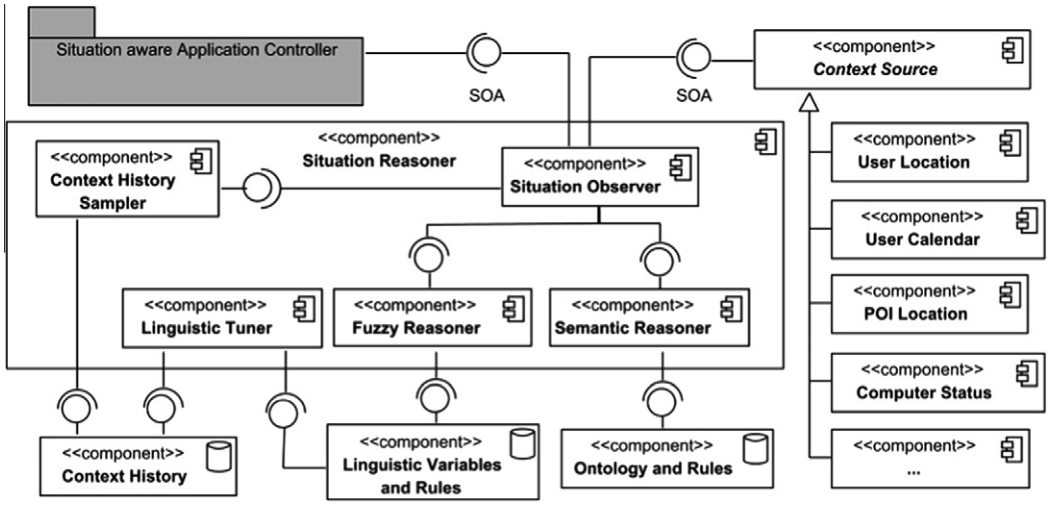


Fig. 4. Component diagram of the proposed general purpose situation reasoner.

design, refer to Ciaramella, Cimino, Lazzerini, and Marcelloni (2010a).

For the upper *context* ontology, the following entities have been identified:

- A *User* is a person who *owns* a mobile *Device* and a *Calendar*. Spatial relationships among other *Users* (such as *is-close-to* and *is-far-from*) and *Places* (such as *is-close-to*, *is-far-from*, and *is-located-in*) are defined.
- A *User* has a *name* and he/she can be *still* or *moving*. Users also *have a current time* and can be in one or more *situations*.
- A *Calendar* contains *Appointments*, which *are organized by Users* and count participants in (i.e., *has-participant*).
- An *Appointment is located in* a *Place* and *is scheduled at* an *Interval*, which has a start-time (i.e., *is-started-at*) and an end-time (i.e., *is-ended-at*).
- Finally, time relationships among *Time Instants* are defined, such as *is-after*, *is-before*, *is-close-to*, and *is-far-from*.

Very general concepts such as *Time* and *Place* are inherited from publicly available ontologies (Ding, Chen, Kagal, & Finin, 2011; Hobbs & Pan, 2006), according to the best practices of reusing domain ontologies. In Fig. 5, these concepts have been included in a dashed rectangle.

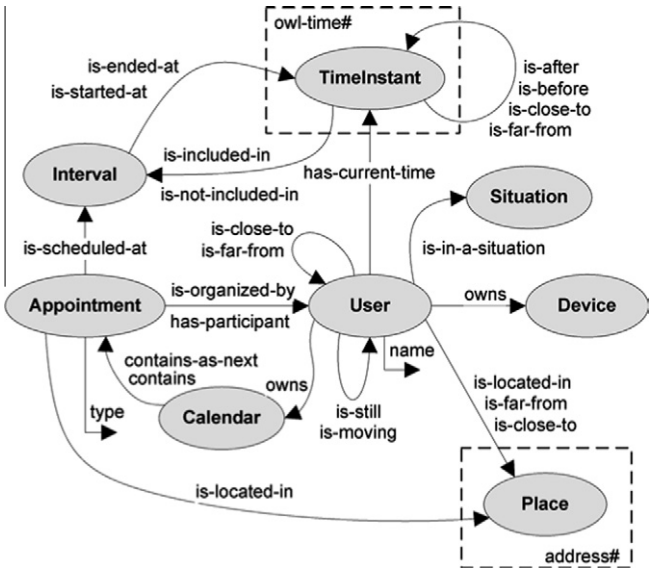


Fig. 5. The upper context ontology.

Domain-specific ontologies are bound to the upper ontology when necessary, narrowing down the scale of the semantic database (Gu & Pung, 2005) and enabling an easy transition of the system from an application domain to another.

In the Semantic Web domain, the Web Ontology Language (OWL, Bechhofer et al., 2004) is the basic language traditionally employed to author ontologies. OWL is a W3C standard, well-supported by semantic engines. In order to manage fuzzy information in an OWL compliant ontology, we used a representation pattern proposed in (Ciaramella, Cimino, Marcelloni, & Straccia, 2010c). The pattern, named Fuzzy Ontology Representation (FOR), considers a fuzzy property as a relation between two concepts, representing additional attributes to describe each relation instance. It is applicable to properties that are related to the same base variable and to the same pair of concepts. As an example, let us consider the base variable *distance*, and the concepts *User* and *Place*. Depending on the actual value of the distance, and considering a prefixed set of distance intervals, we can establish properties like *User is-located-in* a *Place*, *User is-close-to* a *Place*, and *User is-far-from* a *Place*, related to spatial proximity. The presence of each property depends on the membership of the distance value to a prefixed interval. Fig. 6a shows a representation of such example, with the three aforementioned properties and corresponding truth intervals. For instance, in a specific application domain it could be said that a *User is-close-to* a *Place* when the distance between the user and the place belongs to the interval *close-to* ([30,90] m), more formally $is\text{-}close\text{-}to|_{close\text{-}to}$. Fig. 6b shows an abstract representation of this mechanism, for a series of n properties and related n intervals. In order to capture vagueness in this representation, in Fig. 6c the FOR version is shown. Here, an OWL group of properties is transformed into a concept, which includes a specification of the degree for each property. In other words, we assert that there is a property with a certain degree. Each degree is the membership level of the base variable to a specific fuzzy set. It is worth noting that this scheme can be used also in case of a property related to a single concept. In such case, the concept property corresponds to the concept itself. As an example, Fig. 6d shows the representation of the aforementioned fuzzy properties concerning the spatial proximity.

The FOR pattern is RDF and OWL compliant. Indeed, it follows other solutions in the literature, which represent a relation as a class rather than a property.[1] As a result, it is possible to extend

[1] http://www.w3.org/TR/swbp-n-aryRelations/#pattern1.

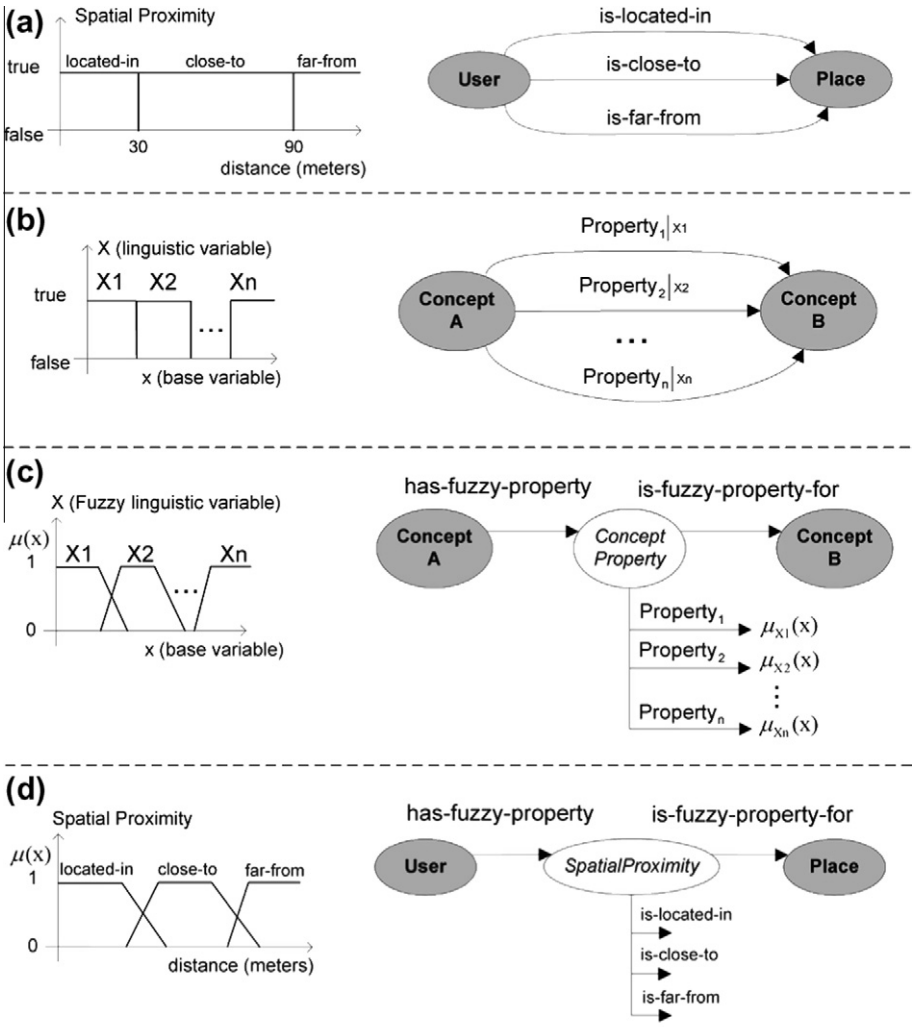


Fig. 6. (a) Concrete representation of three properties related to the spatial proximity, with corresponding truth intervals; (b) abstract representation of interval-related properties; (c) abstract representation of fuzzy set-related properties with the FOR pattern; (d) concrete representation of the fuzzy property *spatial proximity* with the FOR pattern.

any property with fuzzy characters using conventional RDF/OWL engines. Fig. 7 shows the upper context ontology expanded with fuzzy properties using the FOR pattern.

5. Representation languages for situation reasoning

However, it has been recognized in the literature that OWL has expressive limitations, particularly with respect to the representation of properties (Horrocks & Patel-Schneider, 2004). Thus, extensions of OWL based on rule languages have been proposed in the literature. Horrocks and Patel-Schneider (2004) have introduced the OWL Rules Language (ORL), adding rules as a new kind of axiom in OWL. However, using such rules can lead to undecidability. Motik, Sattler, and Studer (2004) have proposed a decidable extension of OWL with rules, where each variable in the rule is required to occur in a non-DL-atom in the rule body. Starting from 2004, a proposal (Horrocks et al., 2004) for an extension of OWL with rules has been submitted to the W3C. The proposed language has been named the Semantic Web Rule Language (SWRL) and it combines the OWL Language with the Rule Markup Language, a markup language to express rules in XML.

To deal with uncertainty in the Semantic Web, several extensions of OWL have been proposed, as reported in Stoilos, Simou, Stamou, and Kollias (2006). Actually, a mechanism to represent vague and imprecise knowledge and information is highly desirable. In the literature, rule languages that take uncertainty into account have been introduced. Pan, Stoilos, Stamou, Tzouvaras, and Horrocks (2006) have proposed f-SWRL that extends SWRL enabling fuzzy rules. More specifically, condition atoms in a rule can include a weight that represents the importance of the atom in the rule itself. Wang et al. (2008) have enhanced f-SWRL enabling the representation of the importance of membership degrees. Recently, a Rule Interchange Format (RIF, Boley et al., 2010) has become a W3C candidate recommendation to enable an interchange format among existing rule systems. The working group has designed a family of languages, called *dialects*, in order to cover the broad categories of rule systems: first-order logic, logic-programming, and action rules (Kifer & Boley, 2005). However, no dialects of RIF provide a support to manage fuzzy rules (Wang, Ma, Yan, & Zhao, 2010). Some non-standard extensions have been proposed, such as RIF Fuzzy Rule Dialect (FRD) based on fuzzy sets (Wang et al., 2010) or RIF Uncertainty Rule Dialect (URD) to represent directly uncertain knowledge (Zhao & Boley, 2008).

In our system we used standardized Semantic Web formalisms (Ding et al., 2005), i.e., the Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL), to express the semantic database and the semantic rule base, respectively. More specifically, the

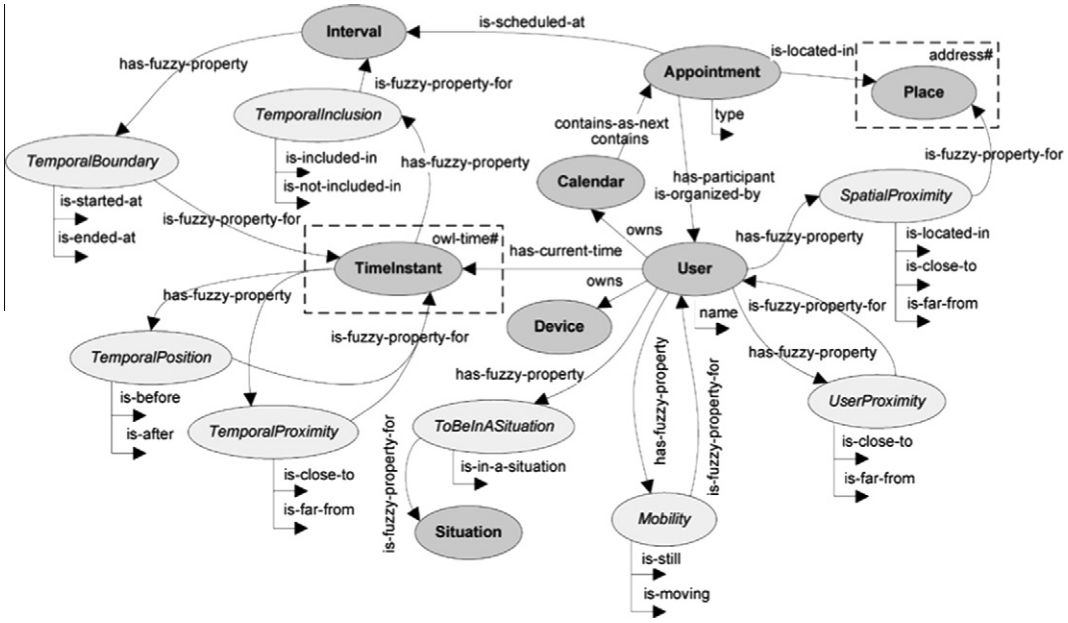


Fig. 7. The upper context ontology expanded with fuzzy properties via the FOR pattern.

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

(a)

IF *user1* **IS A PARTICIPANT** *to the scheduled appointment*
AND *user2* **IS A PARTICIPANT** *to the scheduled appointment*
AND *user1Time* ***IS CLOSE 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. 8. An example of a rule expressed in SWRL: (a) human readable syntax and (b) natural language.

semantic database is formed by the *context ontology* that allows connecting contextual information to situations. The upper context ontology has been shown in Fig. 5. The context ontology is mainly devoted to maintain contextual conditions that hold in the system, providing the antecedents to fire the semantic rules. The semantic rules are the core of the situation reasoning process, enabling the inference of current user's situations. Fig. 8a shows an example of a SWRL rule expressed in the human readable syntax, commonly employed in rule editor GUI.[2] In Fig. 8b the same rule is also expressed in natural language. We point out that there are two types of antecedent conditions, i.e., crisp (binary) and fuzzy, represented in Fig. 8b 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.

With respect to XML and RDF serializations, which are rather verbose and not particularly easy to read, the human readable form is very usable by designers (Lim, Dey, & Avrahami, 2009).

The fuzzy knowledge base has been represented by using the Fuzzy Markup Language (FML, Acampora & Loia, 2005). FML is an XML-based language used to model fuzzy controllers. It provides a platform-independent grammar over shared resources. FML is particularly suitable for: (i) distributing the fuzzy control flow, in order to minimize the global deduction time and to better exploit the natural distributed knowledge repositories; (ii) acquiring, online, the user's behavior and environment status, in order to apply context-aware adaptivity (FML, Acampora & Loia, 2005). A brief example of FML fuzzy knowledge base, with a unique linguistic variable (i.e., the *Spatial Proximity*) is shown in Fig. 9. Here, Fig. 9a shows the definition of the membership functions associated with the linguistic terms "located-in", "close-to", and "far-from", expressed in the usual graphical representation; Fig. 9b is the corresponding FML serialization.

In order to describe how the above representations are processed by the system, the main modules are illustrated in the following, starting from the context sources.

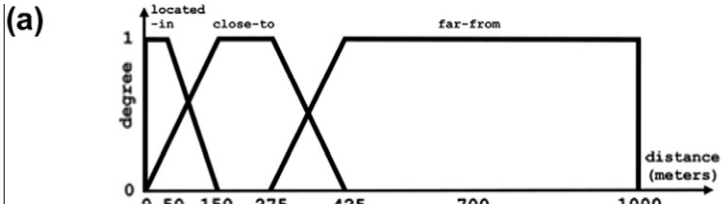
6. The context sources

In the system, raw contextual data are processed by specialized modules. This low-level processing is represented by the *Context Source* module. Fig. 10 shows how the contextual information is gathered in our application domain.

We would like to point out that the user's position is detected by a *Positioning System* at the client side. Location estimation can be based on a GPS signal reader, or can be computed by means of other technologies, such as GSM and Wi-Fi (Sun, Chen, Guo, & Liu, 2005). Whatever the positioning mechanism employed, the server is expected to receive the location of the user, in order to start the inference process. Such information is represented using an interchange standard, the GPX (GPS eXchange format),[3] an open and widely-used XML format that allows describing waypoints, tracks and routes. An example of a GPX waypoint sent to the server is shown in Fig. 11.

[2] In out setting, we used Pellet API to support SWRL rules, http://clarkparsia.com/pellet.

[3] http://www.topografix.com/gpx.asp, accessed on September 2011.



(b)

```
<!DOCTYPE FUZZYCONTROL SYSTEM "fml.dtd">
<FUZZYCONTROL ip = "localhost">
 <KNOWLEDGEBASE>
  <FUZZYVARIABLE domainleft = "0" domainright = "1000"
                 name = "distance" scale = "meter" type = "INPUT">
   <FUZZYTERM name = "located-in">
    <TRAPSHAPE param1 = "0" param2 = "0" param3 = "50" param4 = "150"></TRAPSHAPE>
   </FUZZYTERM>
   <FUZZYTERM name = "close-to">
    <TRAPSHAPE param1 = "0" param2 = "150" param3 = "275" param4 = "425"></TRAPSHAPE>
   </FUZZYTERM>
   <FUZZYTERM name = "far-from">
    <TRAPSHAPE param1 = "275" param2 = "425" param3 = "975" param4 = "975"></TRAPSHAPE>
   </FUZZYTERM>
  </FUZZYVARIABLE>
 </KNOWLEDGEBASE>
</FUZZYCONTROL>
```

Fig. 9. (a) Classical visual definition of fuzzy linguistic variable *Spatial Proximity* and (b) its FML serialization.

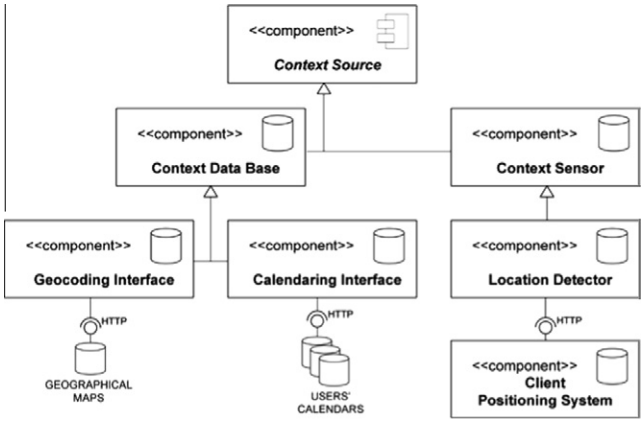


Fig. 10. The context sources.

```
<trkpt lat="43.35382" lon="10.45587">
     <time>2011-10-19T9:16:5Z</time>
</trkpt>
```

Fig. 11. A fragment of a GPX waypoint containing the user position.

Hence, the *Location Detector* module is fed with the user's position. By means of a local database, the *Location Detector* maintains the recent history of the user's positions in order to estimate the user's speed. Speed is computed considering the last three positions of the user. Indeed, experiments have shown that the accuracy of the GPS receiver of common mobile devices is very low in evaluating the user's speed, and then more reliable values could be derived via positions. In our system, the GPX waypoint sent by the client is enriched with the information about user's speed as shown in Fig. 12.

The *Calendaring Interface* module provides a generalized interface for accessing user's personal calendars. An XML-based exchange schema for representing the user's calendar and the related appointments has been designed, as shown in Fig. 13. The schema comes from an abstraction of the XML feed representation of Google Calendar.[4] In

```
<trkpt lat="43.35382" lon="10.45587">
     <time>2011-10-19T9:16:5Z</time>
     <speed>27.82</speed>
</trkpt>
```

Fig. 12. A fragment of a GPX waypoint enriched with the user speed.

this way, information about user events can be loaded into the system regardless the specific software for calendaring.

The *Geocoding Interface* module allows a mapping between geographic data (expressed in a mnemonic manner, such as street address or city name) and geographic coordinates (i.e., latitude and longitude). Moreover, it provides a list of the geographically nearest stores or service locations, such as schools, hospitals, bus stops, etc. The exchange format for this module has been directly derived from the Google Maps API,[5] a free web mapping service.

By exploiting this module, the system is able to locate exactly a user's appointment, to compare that location with the user's position, and to find some points of interest in the user's area. Such information is used to execute the situation assessment rules.

7. The architectural core of the Situation Reasoner

The Situation Reasoner encapsulates the knowledge processing, coordinating the communication of internal engines, via the *Observer* design pattern (Gamma, Helm, Johnson, & Vlissides, 1995). This allows using independently fully standard-compliant mechanisms to manage web semantic and fuzzy engines, i.e., OWL/SWRL and FML based, respectively. Fig. 14 shows a detailed view of the three modules involved in the situation reasoning process.

In the *SituationObserver* package, the *Observer* module steers the execution flow of the other modules. The *Observer* acts as a bridge between the fuzzy and semantic reasoners. It takes the context sources via the *ContextGathering* module, which is responsible for collecting online input data related to the context (e.g., the user's current position). Such data are used by the *ContextMaintainer* module to handle a collection of *ActualConditions*. An *ActualCondition* is an object model representing an instance of a possibly fuzzy ontologic relation, i.e., a quadruple

[4] www.google.com/calendar/, accessed on September 2011.

[5] code.google.com/apis/maps/index.html, accessed on September, 2011.

```
<?xml version="1.0"?>
<calendar id="http://cig.iet.unipi.it/cig/Task-oriented.owl#Calendar_of_Rossi">
      <entry>
            <id>EV_HOSPITAL_SHEART</id>
            <is-located-in>
              <description>Sacred Heart Hospital</description>
              <latitude>43.72220</latitude>
              <longitude>10.41056</longitude>
            </is-located-in>
            <is-started-at>2011-10-19T10:00:00Z</is-started-at>
            <is-ended-at>2011-10-19T14:00:00Z</is-ended-at>
            <type>business</type>
      </entry>
      <entry>
            ...
      </entry>
</calendar>
```

Fig. 13. An excerpt of the exchange format for representing calendar information.

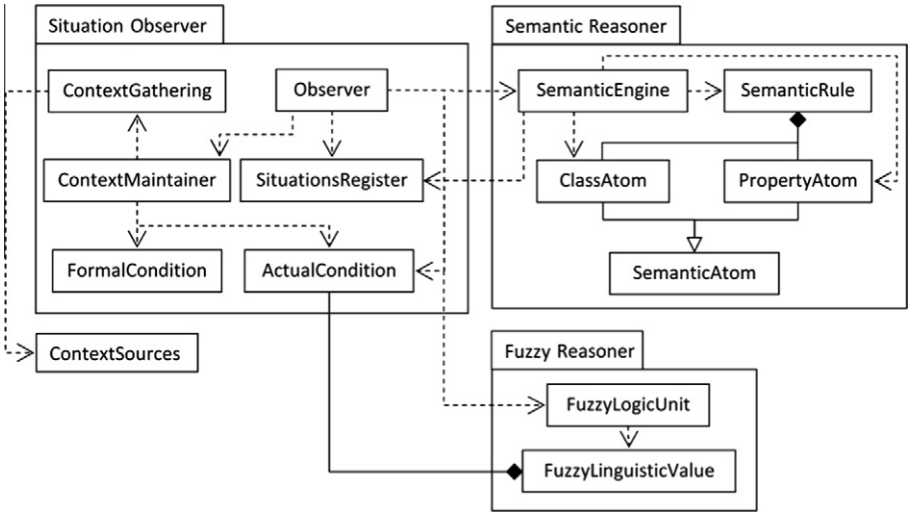


Fig. 14. Decomposition of the situation reasoning.

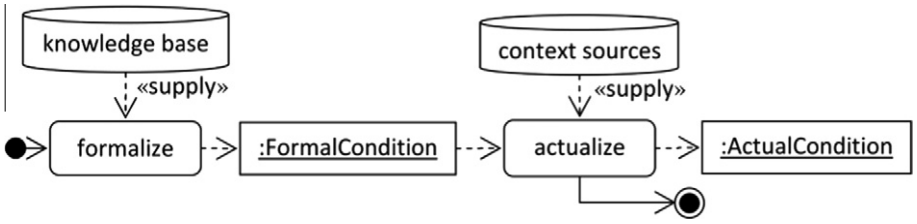


Fig. 15. The *formalization–actualization* process.

$\langle subject, predicate, object, fuzzyMembership \rangle$.

This model extends RDF canonical triples with fuzzy membership, providing an optional fuzzy membership value. The extension is compliant with the foundational specification of the Semantic Web technology stack.[6]

An *ActualCondition* instance is created/updated by the *ContextMaintainer* by means of both the formal ontology and the current context sources. More specifically, this is done in the two-step process shown in Fig. 15. First, an instance of *FormalCondition* is created via the formalization process, which takes as input the knowledge base. A formal condition can be used for multiple individuals (e.g., users). For example, let us consider the (formal) condition *A is-close-to B*. There can be a number of occurrences of this condition in the knowledge base. Hence, the corresponding *FormalCondition* instance encapsulates only the structural parameters of such condition, via the *formalization* process. For example, a *trapezoidal* fuzzy set defined by the vertexes 0, 200, 400 and 800, can represent the semantics of the fuzzy property *is-close-to*. Subsequently, many *ActualCondition*s can be derived, considering the contextual inputs. For example, the position of the user u_1 and the location l_1, on Friday, 20 October 2011, at 5.03 pm can represent a context input. We call *actualization* the process of connecting input sources to the parameters of the formal conditions. In the case of fuzzy formal condition the actualization process also determines the corresponding membership degree. The *formalization–actualization* process can be an important design pattern in any knowledge-based context-aware setting.

The *Observer* module uses the *ContextMaintainer*, the semantic and fuzzy reasoners to process the user's current situations. An in-

[6] See, for instance, *N-Quads: Extending N-Triples with Context*, http://sw.deri.org/2008/07/n-quads/.

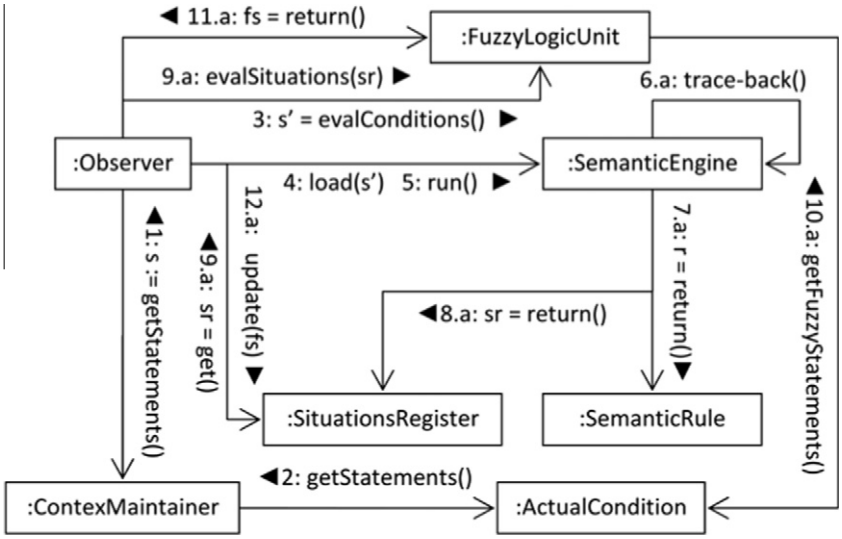


Fig. 16. Communication diagram for the situation reasoning process.

stance of *SituationsRegister* encapsulates the current situations related to a specific user, as processed by the reasoners.

Let us consider the right side of Fig. 14. In the *SemanticReasoner* package, the *SemanticEngine* module represents a wrapper for a standard OWL/SWRL compliant engine,[7] with augmented interfacing capabilities related to fuzzy semantic rules. In particular, the *SemanticRule* is an object model of a (possibly) fuzzy rule (as the one in Fig. 8), made of semantic classes (*ClassAtom*) and semantic properties (*PropertyAtom*). This object model extends the conventional model available in the *SemanticEngine*. In the *FuzzyReasoner* package, the fundamental modules are related to the fuzzy operators (e.g., logic connectives) and to the fuzzy membership processing,[8] implemented in the *FuzzyLogicUnit* and the *FuzzyLinguisticValue* modules, respectively. It is worth noting that a *FuzzyLinguisticValue* instance is made of *ActualCondition* instances, processed via context sources in the *SituationObserver* package.

In order to show a dynamic view of the system, Fig. 16 represents a scenario of communication among the most important modules, by using the UML communication diagram.

The interaction starts with the *Observer* (1–2), which reads the *ActualCondition* objects (by the *ContextMaintainer*) as canonical ontological statements, decoupling the (possibly) fuzzy extension. More specifically, if the *ActualCondition* object consists of fuzzy conditions, the *Observer* asks the *FuzzyLogicUnit* for their evaluation (3.a). The *FuzzyLogicUnit* returns a certainty value in [0,1] for each uncertain condition. If the certainty value is larger than zero, the condition will be considered to be true in the semantic inference. Otherwise the condition will be considered to be false. Once the fuzzy extension has been decoupled, the *Observer* loads the ontological statements into the ontological model (4), handled by the *SemanticEngine*. The *Observer* runs the *SemanticEngine* (5). The semantic rules are then executed, and the ontological properties internal to the *SemanticEngine* are updated. At that time, the resulting user's situations can be derived from such ontological properties. If more than one situation has become *true*, the *Observer* will ask the *FuzzyReasoner* for the corresponding membership degree. To this aim, the *SemanticEngine* performs a *tracing-back* (6.a) of the ontological predicates related to those situations, returning the *SemanticRule* objects (7.a) corresponding to the user's current situations. It is worth noting that, since the crisp (binary) conditions have been already evaluated as true for those situations, the membership degree of each recognized situation can be calculated considering only the fuzzy conditions. Such fuzzy conditions have been already calculated in step 3. Hence, first the *SemanticEngine* updates the *SituationsRegister* (8.a), and then the *Observer* triggers the *FuzzyLogicUnit* to deduce the fuzzy membership degrees for each of the user's current situations (9.a). The *FuzzyLogicUnit* gets the fuzzy statements (10.a), and returns the membership degree of each involved situation (11.a), in order to allow the *Observer* to update the *SituationsRegister* (12.a). Basically, a fuzzy situation is defined as a logical conjunction of contextual conditions; in our settings the certainty degree of a situation is computed as the *minimum* of the truth degrees of all the fuzzy contextual conditions.

8. The Behavior Tuning module

As already described, contextual conditions in each semantic rule are expressed via linguistic variables. In particular, each linguistic variable comprises a set of linguistic values, each defined by a fuzzy set, and describes one or more contextual conditions. In general, such fuzzy sets would be user-dependent, because different users have different behaviors. For instance, a latecomer user would assign a different meaning to the contextual condition *is-close-to*, with respect to a punctual user. In order to recognize properly the current situation, the system should take the user's behavior into account. This personalization allows timely rule firing. In order to show a method to derive the user's behavior, let us refer again to the latecomer user as an example: the situation *on-going-meeting* has to be recognized by taking into account some delay scenarios of the user, and adapting accordingly shape and position of the fuzzy sets. This process can be done employing the *context history*.

In the literature, the use of the context history has been analyzed and proved as extremely effective in enabling personalization and adaptation, by discovering recurrent patterns in the data (Byun & Cheverst, 2003). More specifically, the relevant data to collect about user context history concern the state of context sources and the related situations. To this aim, the user is supposed to expresses an explicit feedback by signaling the beginning/end of each occurred situation, holding this process up to collect a sufficient amount of training data.

Once collected, the context history is employed as training set for a genetic algorithm (GA). The GA aims to adapt fuzzy sets to the actual behavior and habit of the user, increasing the accuracy and responsiveness of the situation assessment. Indeed, GAs have

[7] In our setting we used the *Jena* API, http://jena.sourceforge.net.
[8] In our setting, we used the *JfuzzyLogic* API, http://jfuzzylogic.sourceforge.net.

been widely used to tune membership functions of linguistic values in fuzzy rule-based systems. For such methods a specific term, *genetic fuzzy systems*, has been coined in the literature (Herrera, 2008). Although in the last years different algorithms and procedures have been proposed to learn membership functions from data (Cordón, 2011; Kaya & Alhajj, 2005), in this paper we refer to the method used in Ciaramella, Cimino, Lazzerini, and Marcelloni (2010b).

Let us consider a generic linguistic variable X_j shown in Fig. 17. We assume that each linguistic value is represented by a trapezoidal membership function, $A_{j,t}$, whose support is $[a_{j,t},d_{j,t}]$ and whose core is $[b_{j,t},c_{j,t}]$. Further, for each fuzzy set $A_{j,t}$, $t = 1,\ldots,T_j - 1$, we suppose that $c_{j,t} = a_{j,t+1}$ and $d_{j,t} = b_{j,t+1}$. Finally, $a_{j,1} = b_{j,1}$ and $c_{j,T_j} = d_{j,T_j}$ coincide with the left and right extremes of the universe, respectively. Thus, the strong partitions made of these membership functions can be represented by $T_j - 1$ pairs $(a_{j,t},b_{j,t})$, with $t = 2,\ldots,T_j$. Let M be the number of linguistic variables that have to be tuned. The overall database can be defined by the chromosome shown in Fig. 18.

We aim to tune the membership functions so as to increase the capability of the system to recognize the desired situation. To this aim, we maximize the following fitness function f. Let $s_1,\ldots,s_S$ be the possible situations the system can recognize. Let s_t be the target situation. Then, f is defined as:

$$f = \sum_k (\mu_{s_t}(k) - \max_{r \neq t}(\mu_{s_r}(k))) \quad (1)$$

where μ_{s_t} and μ_{s_r} are the certainty degrees with which the fuzzy engine recognizes the target situation s_t and each situation s_r different from s_t for the sample k in the training set. The training set is built on the basis of samples of the context history. Each sample is made of the contextual variables that allow inferring the situation, together with the user's situation itself. To give a glimpse of the context history, let us consider again the semantic rule reported in Fig. 8.

Here, the context history is made of: (i) the temporal proximity of the *user*1 time to the *meeting* start time; (ii) the spatial proximity of the *user*1 position to the place of the *meeting*; (iii) the spatial user-proximity of the *user*1 position to the *user*2 position. These contextual values are periodically recorded and associated to the respective user's situations. Once the training set is large enough (a few hundreds of samples for each situation), the GA can be executed.

We would like to highlight some parameters of our setting. For each observed situation, we store approximately an average of 400–450 samples (about one per minute). The initial population of the GA is made of 50 chromosomes. Each individual of the population is randomly generated within the universe of the base variables. We adopt a BLX-α crossover operator with $\alpha = 0.5$ (Eshelman & Schaffer, 1993), an adaptive feasible mutation operator (Vasant & Barsoum, 2009) and stochastic uniform selection (Baker, 1987). The algorithm stops when the average fitness of the population, over 2000 generations, varies less than 10^{-6}. At the end of the GA execution, the membership function parameters are tuned by using the values of the chromosome with the highest fitness value.

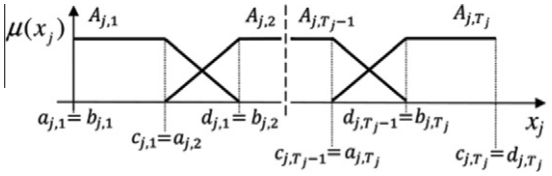


Fig. 17. A generic partition of a linguistic variable.

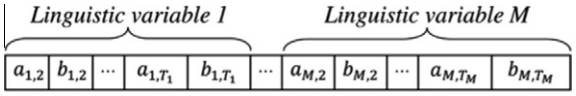


Fig. 18. The chromosome coding.

9. Evaluation case study: service recommendation

In order to show the effectiveness of the proposed approach for situation awareness, we implemented the overall method in the field of resource recommendation. In general, recommenders aim to suggest the most relevant items to the user, usually based on information about the item (*content-based approach*) or on the user's relationships with other users of the system (*collaborative filtering approach*). Indeed, the use of context in recommendation systems is a recent introduction. The vast majority of the recommenders in the literature do not take any additional contextual information into account (Adomavicius & Tuzhilin, 2011). In the following, we first introduce a brief literature review about context-aware recommenders, and after we describe our solution based on the proposed GEPSIR design approach. Finally, we show simulation results on a case study.

9.1. Context-aware recommenders in the literature

One of the first approaches that acknowledged the importance of context in recommendation comes from Herlocker and Konstan (2001). Here task-specific recommendations have been proposed. A task is identified by a set of sample items related to the task itself. For instance, if the user provides a hammer as example item in a shopping recommender, the system can recommend buying nails. Such associations can be easily identified automatically by the system, via association rules. Naganuma and Kurakake (2005) have proposed a task-oriented service navigation system that supports users in finding appropriate services by browsing a rich task ontology. Such ontology contains a variety of real-world structured tasks and related services. In Luther et al. (2008), the authors have extended this system by taking the user's situation into account, in order to suggest tasks and services actively, without the need for initial user input. However, this approach does not consider the uncertainty that affects contextual data in order to infer the user's correct situation. Indeed, situations are recognized by applying dynamic assertional classification of contextual entities such as the location, the time and the neighboring people.

Weißenberg et al. (2006) have proposed a system that exploits situation awareness to provide the user with the desired information and services. In this approach, a situation describes a user demand that occurs at a certain time and is formed by a sequence of contexts defined as a logical expression, such as *LocationOfTheUser(stadium)* or *TypeOfMovement(fast)*. Both situation inference and service selection are based on ontologies, to infer first a set of situations and then a set of services which may be relevant in such situations. However, no uncertainty issue is considered. Moreover, the user may be in none, one or many situations in parallel, but no ranking is given to help the user choose the best fitting situation, or to list the recommended services in a suitable order. Recently, Petry, Tedesco, Vieira, and Salgado (2008) proposed ICARE, a recommendation system that returns references to experts in a requested domain using contextual information. More specifically, the system improves its recommendations by using the user's and expert's context, privileging those experts who best fit the user's current needs. Examples of contextual information are expert's availability, approachability, and social distance. Contextual rules are defined to set appropriate weights in order to decide, given a context, which contextual information should be favored. Hence,

the recommendations are different from one user to another, according to her/his context. However, ICARE does not consider uncertainty aspects in the contextual information. Moreover, the system does not act proactively but waits for the user's requests in order to provide the desired recommendations.

Fuzzy logic has been proved to be a promising approach to manage the natural uncertainty that affects contextual data. Cena et al. (2006) have employed fuzzy logic in a context-aware tourism recommender. The system exploits personalization rules to suggest services (e.g., restaurants, places to visit, etc.) tailored to the user's profile and context. User's profile is a very important piece of the system, built by: (i) explicit user's data, such as age, gender, general interests, etc.; (ii) data inferred via fuzzy rules based on domain knowledge, such as propensity to spend, specific interests, etc; (iii) current user's needs and wishes, by observing the sequence of interactions of the user with the system, such as printed pages, on-line booking, etc. Based on the user's interests, maintained in the profile, and the user's position, the system computes an overall score for each service and recommends services in an order depending on the score. Thus, the context is limited mainly to the user's location, which acts as a filter to recommend services close to the user. Moreover, here proactivity of the recommendations is not provided, but only envisioned as future work. Park, Yoo, and Cho (2006) have proposed a context-aware music recommendation system that employs fuzzy Bayesian networks and utility theory. In particular, a fuzzy system is exploited to preprocess contextual data from various sensors and the Internet, in order to have quantized inputs for the Bayesian network. Based on these inputs, the network can infer the user context and assign a probability. Finally, recommendations are proposed depending on a final score, which is computed taking the inferred context and user's preferences into account. In this approach, no semantic aspects of the contextual information are considered. Moreover, the inference process is entirely based on the Bayesian networks, resulting in a not easily accessible mechanism for average users.

9.2. The GEPSIR approach for context-aware resource recommender

From the user perspective, a resource recommender shows the set of the most relevant resources depending on the user's current situation. To this aim, the situations inferred by GEPSIR have been connected by means of domain knowledge expressed by *task* ontologies to a set of relevant tasks that, given the situation, the user would do (Luther et al., 2008). Hence, the current tasks together with the contextual information are used to recommend resources, in an order that depends on the fuzzy membership degrees associated with the situation. Further, the recommender can automatically parameterize services by using the context. The upper *task* ontology employed in the recommender is shown in Fig. 19.

9.3. A real-world case study

The resource recommender has been applied to a real-world business case, in order to show the effectiveness of the proposed approach. The business case concerns a pharmaceutical consultant in typical business situations.

By means of a series of interviews with domain experts, a knowledge model for the business case has been developed. In particular, the upper *context* ontology has been extended with domain-specific ontologies, identifying the concepts and relations among concepts that better describe the business case. Fig. 20 shows the comprehensive *context* ontology defined for this case study. The domain-specific *context* ontology contains specific concepts such as *Hospital*, *Doctor*, *Meeting*, etc. Sub-concepts are represented by white oval shapes and white directed edges indicate inheritance.

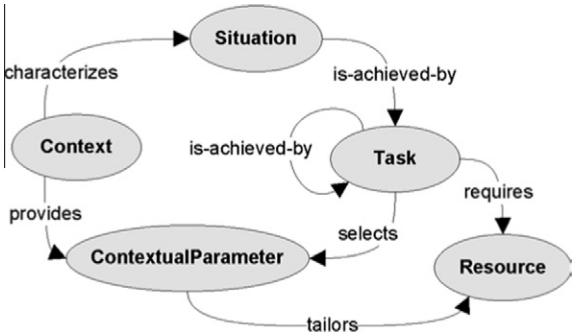


Fig. 19. The upper *task* ontology that connects situations with resources.

Moreover, different situations have been identified, and the related semantic rules have been defined. 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.

Afterwards, for each situation, a set of possible tasks and related resources have been identified, starting from the actual demands of interviewed experts. Thus, the upper *task* ontology has been extended with domain-specific concepts and relations. Fig. 21 illustrates a simplified excerpt of the comprehensive *task* ontology for the pharmaceutical consultant case study.

In order to tune the linguistic variables of each semantic rule, the GEPSIR genetic approach has been employed. Firstly, starting from five real tracks, 21 training tracks have been generated. Each track contains the user movements for a whole day and the related context history, as explained in the previous section.

To produce real tracks, we used an *Apple iPhone 2G* smart phone, permanently connected to the Internet and to the GPS signal, and equipped with InstaMapper,[9] a free service that enables to track a phone in real time. To assess the system properties (e.g., the robustness), we simulated other tracks by means of an auxiliary web application based on Google Maps API.[10] The simulator generates new tracks based on instructions provided by the user, such as the geographical coordinates of the starting and end points, the number of appointments during the day, the distance between each appointment, etc. Moreover, it can simulate user movements under different circumstances, such as different means of transportation (walking, by bicycle or by car), different traffic conditions (without/with traffic jam), or different weather conditions (sunny day, cloudy, rainy, etc). Noise has been also introduced to make contextual sources very close to real world signals. Fig. 22 shows the user interface of the simulator during a batch generation of the tracks. In particular, some conditions can be noted in the configuration area, such as weather: 'calm' and transportation: 'by car'.

Starting from the fuzzy linguistic variables in the upper context ontology, a domain expert defined the set of linguistic terms to be tuned by the GA. The linguistic variables involved in the tuning process are: (i) *spatial proximity*, which represents the distance of the user from a place, expressed linguistically as *close-to* and *far-from*; (ii) *temporal position,* which denotes the order between two

[9] InstaMapper, http://www.instamapper.com/, accessed January 2011.
[10] Google Maps API, http://code.google.com/apis/maps/, accessed January 2011.

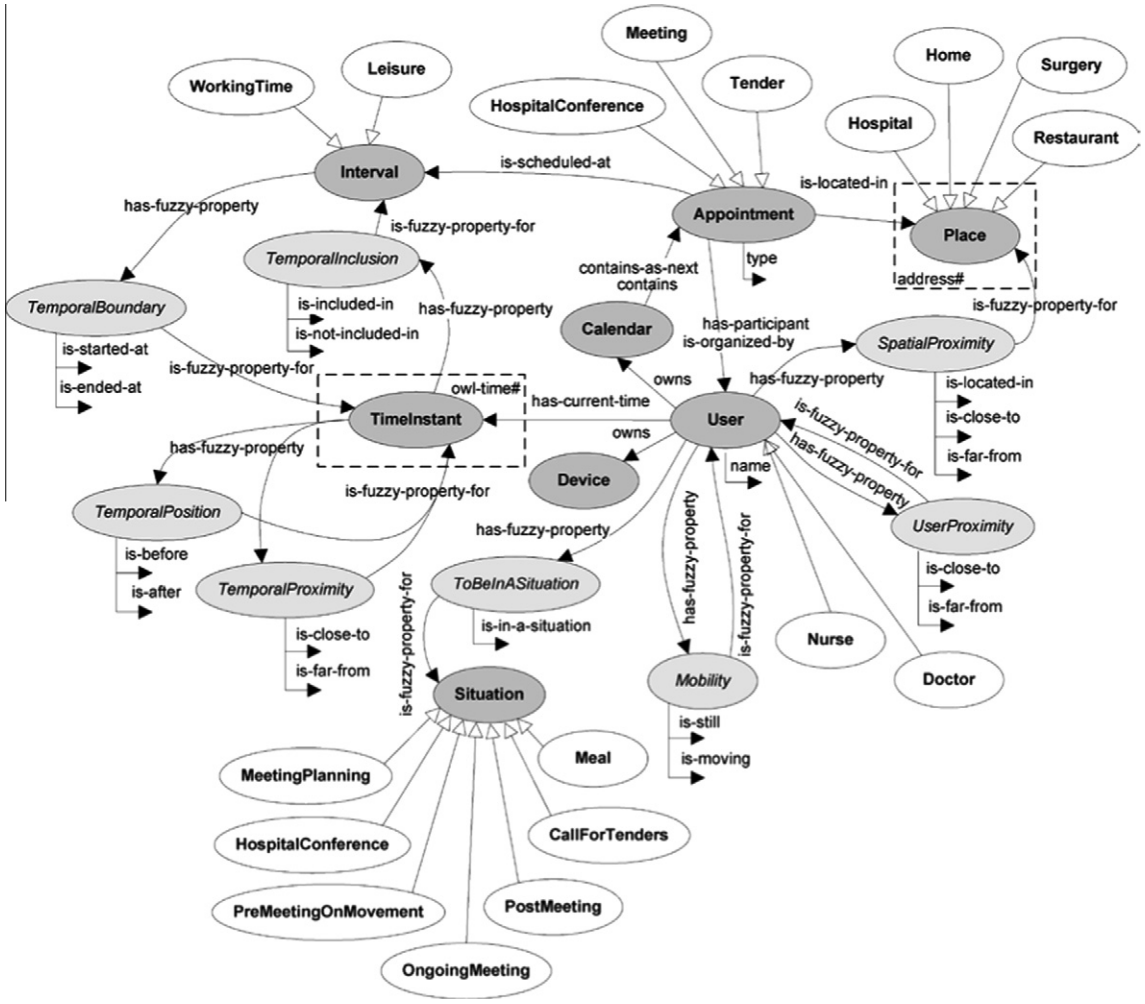


Fig. 20. The comprehensive *context* ontology for a pharmaceutical consultant.

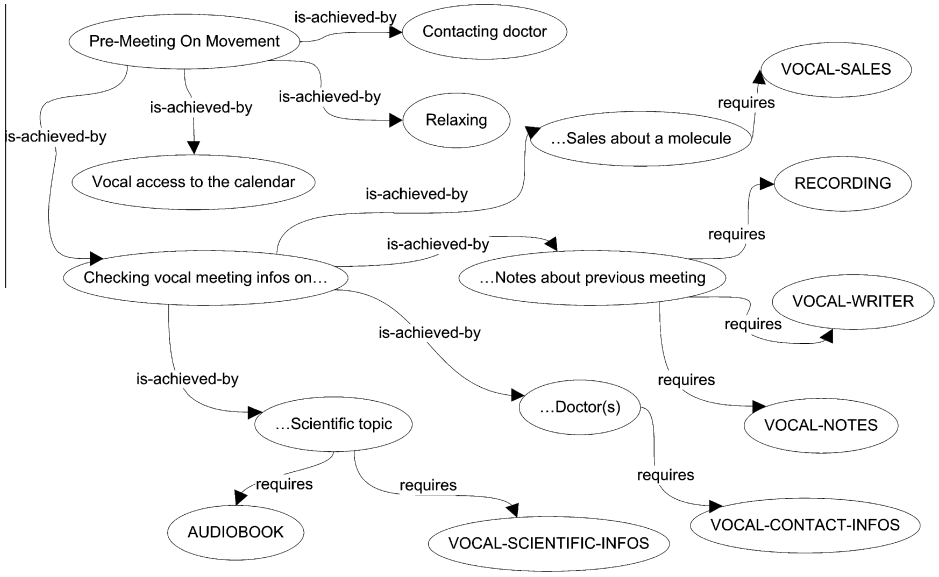


Fig. 21. An excerpt of the *task* ontology defined for the situation *Pre-Meeting On Movement*.

instants of time, and is expressed linguistically as *before* and *after*; (iii) *temporal inclusion*, which assesses whether an instant of time belongs to a temporal interval and is expressed linguistically as *included-in* and *not-included-in*; and (iv) *user mobility*, which repre-

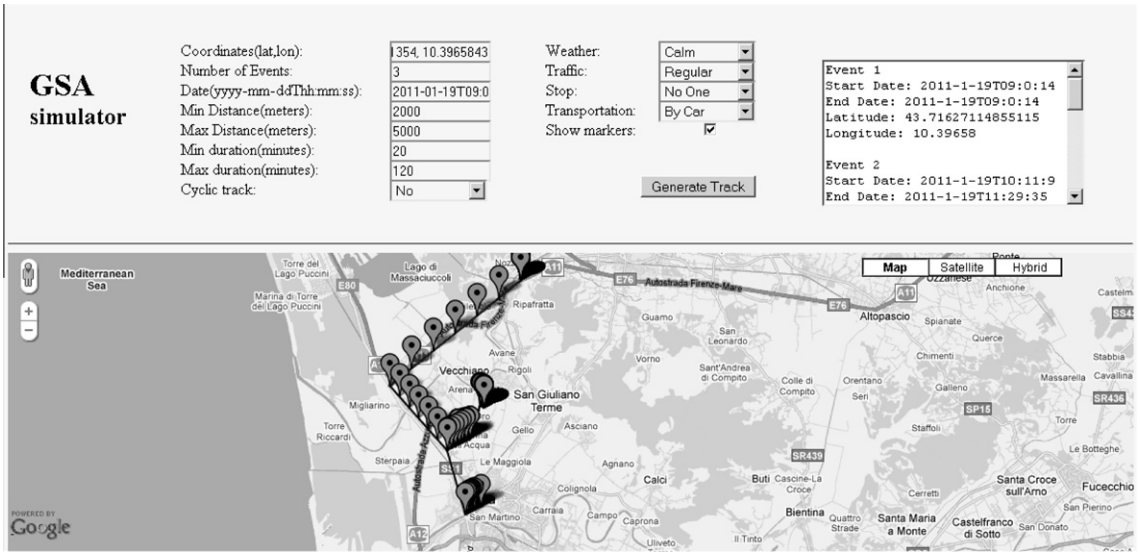


Fig. 22. The client-side simulator to generate tracks.

sents the speed of the user and is expressed linguistically as *still* and *moving*. Fig. 23a and b shows the linguistic variables defined by the domain expert and tuned by the GA, respectively.

After the tuning process, the system has been tested by a user, considering a weekly timetable consisting of 77 appointments. To assess the reliability and timeliness of the recommender, we employed the responsiveness as a performance index. Such index is defined as the average of the differences between the step in which a situation S_i starts/ends for a user and the step in which the system detects the start/end of the same situation for the same user. Formally, let N_i be the number of occurrences $o_{i,p}$ of start and end of a situation S_i. For the pth occurrence $o_{i,p}$, we record the instant of time $t_{i,p}$ at which that occurrence occurs, and the time $t'_{i,p}$ at which the recommender recognizes the occurrence. Let us define the responsiveness of the recommender to the situation S_i as:

$$Resp(S_i) = \frac{\sum_{p=1}^{N_i} |t'_{i,p} - t_{i,p}|}{N_i}. \quad (2)$$

Table 1 shows the responsiveness of the recommender for each of the situations occurred during the testing.

We can observe that, as expected, the recommenders that use partitions tuned by the GA on average outperform appreciably the recommender with partitions defined by the domain expert. This result proves the effectiveness of the tuning process.

Table 1
Responsiveness of the system for the pharmaceutical consultant case study.

Situation (start/end)	Responsiveness (s)	
	Recommender defined by the domain expert	Recommender tuned by GA
Pre-Meeting (start)	50.519	36.008
Pre-Meeting (end)	86.898	48.238
Ongoing-Meeting (start)	111.796	84.830
Ongoing-Meeting (end)	32.171	23.285
Post-Meeting (end)	27.533	23.148
Hospital-Conference (start)	117.450	68.857
Hospital-Conference (end)	41.610	35.786
Meal (start)	102.746	69.253
Meal (end)	59.298	49.294
Average	70.002	48.744

We have also tested the user acceptance of the recommender tuned by the GA. In particular, five pharmaceutical consultants have been asked to evaluate the recommender. To this aim, an auxiliary web application, based on Google Maps API, has been

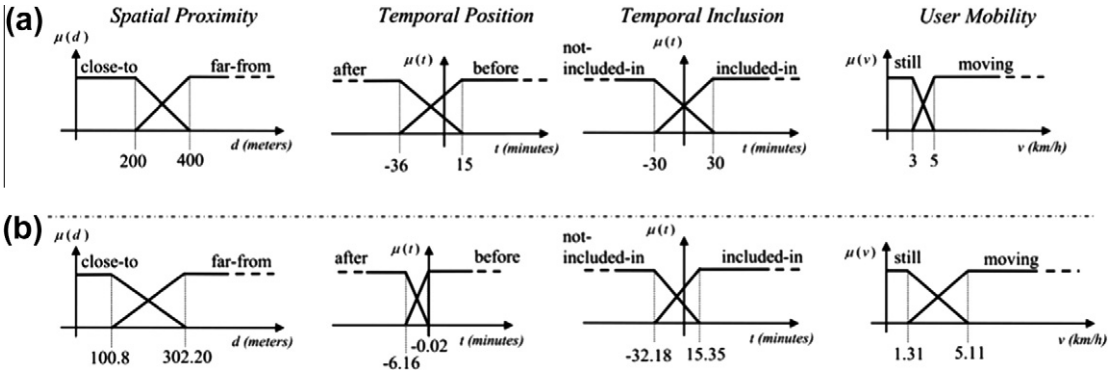


Fig. 23. Linguistic variable partitions for the pharmaceutical consultant case study: (a) defined by a domain expert and (b) tuned by the GA.

Fig. 24. The auxiliary web application to evaluate the recommender.

Table 2
Selection time for a resource for the pharmaceutical consultant case study.

User	Selection time without recommender (s)	Selection time with recommender (s)
#1	15.0	8.6
#2	13.7	7.1
#3	13.2	9.8
#4	20.1	12.4
#5	16.7	8.5
Average	15.74	9.28

developed and used. The application provides an online simulation interface that is used by the user himself, as shown in Fig. 24.

The user can choose her/his position in the map and provide information about her/his speed and the current date. The test is composed by two phases, which comprise a predefined number of recommendations, i.e., 10 iterations. First, after the user has input the data about her/his position, the application proposes a list of all resources in the smart phone, without a particular order and relation with the user's situation. Thus, the user is invited to choose the resource that she/he needs, given the figured situation. In the second phase, the recommender is enabled and, after the user has input the data about her/his position, the application exploits the inferred situation to filter the recommended resources. Hence, the user is invited to choose the desired resource, guided by the predefined task ontology of the case study. The times required for the selections of both phases are registered and compared. Results of the tests are reported in Table 2. It is worth noting that the selection time of a resource is sensibly reduced. Moreover, in the interviews, users have asserted that they selected resources that were not foreseen.

Finally, we have evaluated the response time of the system for each recommendation. In a system equipped with an Intel Core 2 Duo Processor 2.2 GHz, with 3 GB DDR2 of RAM, the average response time of the system is 0.932 s, which guarantees a soft real time response to the user needs.

10. Conclusions

To recognize a situation in which a user is involved leads to better identify her/his demand at a certain time. In this paper, a rule-based, robust and general approach for managing situation awareness is proposed. In the approach, situation is derived from a logical conjunction of contextual conditions. Domain knowledge is expressed by means of ontologies and semantic rules, in order to guarantee portability, integration and extensibility. In this way, software agents can administrate their own contextual sources, easily communicating with each other. The overall system can rely on a formal representation avoiding inconsistency of the knowledge base. Contextual conditions can be affected by uncertainty, e.g., due to inaccurate sensor measurements or imprecise human expression concepts. Fuzzy logic theory is employed to effectively manage the uncertainty, enabling a richer processing of the contextual conditions. Thus, a rule base approach with fuzzy and semantic technologies has been developed.

Using predefined rules to infer situations can lead to unsatisfactory results. Indeed, users have different habits that may affect the way in which situations arise. Further, the same user can change her/his behavior over time. In our approach, context history is considered as a powerful source of information on user's behavior. By means of a genetic algorithm, the rule base can be automatically tuned to fit the actual behavior of the user, increasing the accuracy and responsiveness of the situation assessment.

Finally, a real evaluation case study concerning resource recommendation has been provided. The study has been focused on a pharmaceutical consultant in typical business situations. Simulation results assess the reliability and effectiveness of the proposed approach. Moreover, the user acceptance of the system has been tested, thus confirming that the GEPSIR approach significantly improves the user interaction.

Acknowledgements

This work was partially 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. Prato (Italy) for financial and technical support.

References

Acampora, G., & Loia, V. (2005). Using FML and fuzzy technology in adaptive ambient intelligence environments. *International Journal of Computational Intelligence Research, 1*(2), 171–182.

Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (pp. 217–253). Boston, MA: Springer US.

Anagnostopoulos, C., & Hadjiefthymiades, S. (2008). Enhancing situation-aware systems through imprecise reasoning. *IEEE Transactions on Mobile Computing, 7*(10), 1153–1168.

Anagnostopoulos, C., & Hadjiefthymiades, S. (2010). Advanced fuzzy inference engines in situation aware computing. *Fuzzy Sets and Systems, 161*(4), 498–521.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In L. Erlbaum Assoc. Inc. (Ed.), *Proceedings of the second international conference on genetic algorithms and their application, Hillsdale, NJ, USA* (pp. 14–21).

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., et al. (2004). OWL Web Ontology Language Reference. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/owl-ref/>.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathanf, A., & Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing, 6*(2), 161–180.

Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., & Reynolds, D. (2010). RIF core dialect. W3C Recommendation 22 June 2010. <http://www.w3.org/TR/rif-core/>.

Byun, H. E., & Cheverst, K. (2003). Supporting proactive intelligent behaviour: The problem of uncertainty. In *Proceedings of the workshop on user modeling for ubiquitous computing (UM'03). Johnstown, PA, USA* (pp. 17–25).

Cao, J., Xing, N., Chan, A. T. S., Yulin, F., & Jin, B. (2005). Service adaptation using fuzzy theory in context-aware mobile computing middleware. In *Proceedings of the 11th IEEE international conference on embedded and real-time computing systems and applications (RTCSA'05), Hong Kong, China* (pp. 496–501).

Cena, F., Console, L., Gena, C., Goy, A., Levi, G., Modeo, S., & Torre, I. (2006). Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide. *AI Communications, 19*(4), 369–384.

Ciaramella, A., Cimino, M. G. C. A., Lazzerini, B., & Marcelloni, F. (2010a). A situation-aware resource recommender based on fuzzy and semantic web rules. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 18*(4), 411–430.

Ciaramella, A., Cimino, M. G. C. A., Lazzerini, B., & Marcelloni, F. (2010b). Using context history to personalize a resource recommender via a genetic algorithm. In *Proceedings of the IEEE international conference on intelligent systems design and applications (ISDA'10). Cairo, Egypt* (pp. 965–970).

Ciaramella, A., Cimino, M. G. C. A., Marcelloni, F., Straccia, U. (2010c). Combining fuzzy logic and semantic web to enable situation-awareness in service recommendation. In *Proceedings of the 21st LNCS international conference on database and expert systems applications (DEXA '10). Bilbao, Spain* (Vol. 6261, pp. 31–45).

Cordón, O. (2011). A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems. *International Journal of Approximate Reasoning, 52*(6), 894–913.

Costa, P. D., Pires, L. F., & Sinderen, M. V. (2005). Architectural patterns for context-aware services platforms. In *Proceedings of the second international workshop on ubiquitous computing (IWUC'05). Miami, FL, USA* (pp. 3–19).

Coutaz, J., Crowley, J. L., Dobson, S., & Garlan, D. (2005). Context is key. *Communications of the ACM, 38*(3), 49–53.

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing, 5*(1), 4–7.

Ding, L., Chen, H., Kagal, L., & Finin, T. (2011). Public address ontology. DARPA Agent Markup Language (DAML) Program, DAML Tools for supporting Intelligent Information Annotation, Sharing and Retrieval. <http://daml.umbc.edu/ontologies/ittalks/address>.

Ding, L., Kolari, P., Ding, Z., Avancha, S., Finin, T., & Joshi, A. (2005). Using ontologies in the Semantic Web: A survey. University of Maryland, Baltimore County (UMBC) Technical Reports, TR-CS-05-07.

Eshelman, L. J., & Schaffer, J. D. (1993). Real-coded genetic algorithms and interval-schemata. In L. Darrell Whitley (Ed.). *Foundations of genetic algorithms* (Vol. 2, pp. 187–202). San Mateo, CA, USA: Morgan Kaufmann Publishers, Inc..

Etter, R., Costa, P. D., & Broens, T. (2006). A rule-based approach towards context-aware user notification services. In *Proceedings of the ACS/IEEE international conference on pervasive services (PERSER '06)* (pp. 281–284). Washington, DC, USA: IEEE Computer Society.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Reading, MA, USA: Addison-Wesley.

Greenwood, D., Lyell, M., Mallya, A., & Suguri, H. (2007). The IEEE FIPA approach to integrating software agents and web services. In *Proceedings of the 6th ACM international joint conference on Autonomous Agents and Multiagent Systems (AAMAS '07). New York, NY, USA* (pp. 1412–1418).

Gu, T., Pung, H. K., & Zhang, D. Q. (2004). A Bayesian approach for dealing with uncertain contexts. In *Proceedings of the second international conference on pervasive computing (Pervasive'04). Pervasive in the book, Advances in pervasive computing*. Austrian Computer Society (OCG), 176. Vienna, Austria.

Gu, T., & Pung, H. K. (2005). A middleware for building context-aware mobile services. In *Proceedings of the IEEE 59th vehicular technology conference (VTC2004-Spring). Milan, Italy* (Vol. 5, 2656-2660).

Gu, T., Pung, H. K., & Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications, 28*, 1–18.

Haghighi, P. D., Krishnaswamy, S., Zaslavsky, A., & Gaber, M. M. (2008). Reasoning about context in uncertain pervasive computing environments. In *Proceedings of the third European conference on smart sensing and context (EuroSSC 2008). LNCS* (Vol. 5279, pp. 112–125). Berlin, Germany: Springer.

Hagras, H., Doctor, F., Callaghan, V., & Lopez, A. (2007). An incremental adaptive life long learning approach for type-2 fuzzy embedded agents in ambient intelligent environments. *IEEE Transactions on Fuzzy Systems, 15*(1), 41–55.

Herlocker, J. L., & Konstan, J. A. (2001). Content-independent task-focused recommendation. *IEEE Internet Computing*, 40–47.

Herrera, F. (2008). *Genetic fuzzy systems: Taxonomy, current research trends and prospects. Evolutionary Intelligence* (Vol. 1). Springer-Verlag.

Hobbs, J. R., & Pan, F. (2006). Time ontology in OWL. W3C Working Draft 27 September 2006. <http://www.w3.org/TR/owl-time>.

Horrocks, I., & Patel-Schneider, P. F. (2004). A proposal for an OWL rules language. In *Proceedings of the thirteenth international world wide web conference (WWW 2004). New York, NY, USA* (pp. 723–731).

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML.W3C Member Submission 21 May 2004. <http://www.w3.org/Submission/SWRL/>.

Kaya, M., & Alhajj, R. (2005). Genetic algorithm based framework for mining fuzzy association rules. *Fuzzy Sets and Systems, 152*(3), 587–601.

Kifer, M., & Boley, H. (2005). RIF overview. W3C Working Group Resources. <http://www.w3.org/2005/rules/wiki/Overview>.

Korpipää, P., Mantyjarvi, J., Kela, J., Keranen, H., & Malm, E. J. (2003). Managing context information in mobile devices. *IEEE Pervasive Computing, 2*(3), 42–51.

Lim, B. Y., Dey, A. K., & Avrahami, D. (2009). Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the 27th ACM international conference on human factors in computing systems (CHI'09). New York, NY, USA* (pp. 2119–2128).

Luther, M., Fukazawa, Y., Wagner, M., & Kurakake, A. (2008). Situational reasoning for task-oriented mobile service recommendation. *The Knowledge Engineering Review, 23*, 7–19.

Mäntyjärvia, J., & Seppänen, T. (2003). Adapting applications in handheld devices fuzzy context information. *Interacting with Computers, 15*, 521–538.

Motik, B., Sattler, U., & Studer, R. (2004). Query answering for OWL-DL with rules. In *Proceedings of the 2004 international semantic web conference (ISWC 2004). LNCS* (Vol. 3298, pp. 549–563). Berlin, Germany: Springer-Verlag.

Naganuma, T., & Kurakake, S. (2005). A task oriented approach to service retrieval in mobile computing environment. In M. H. Hamza (Ed.), *Proceedings of the IASTED international conference on artificial intelligence and applications (AIA'05), Innsbruck, Austria* (pp. 527–532).

Pan, J. Z., Stoilos, G., Stamou, G., Tzouvaras, V., & Horrocks, I. (2006). F-SWRL: A fuzzy extension of SWRL. *Journal on Data Semantics, Special Issue on Emergent Semantics, 4090*, 28–46.

Park, H. S., Yoo, J. O., & Cho, S. B. (2006). A context-aware music recommendation system using fuzzy Bayesian networks with utility theory. In L. Wang et al. (Eds.), *Proceedings of the international conference on fuzzy systems and knowledge discovery (FSKD'06). LNAI* (Vol. 4223, pp. 970–979). Berlin, Germany: Springer-Verlag.

Petry, H., Tedesco, P., Vieira, V., & Salgado, A. C. (2008). ICARE: A context-sensitive expert recommendation system. In *Proceedings of the workshop on recommender systems (ECAI'08). Patras, Greece* (pp. 53–58).

Ranganathan, A., Al-Muhtadi, J., & Campbell, R. H. (2004). Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing, 3*(2), 62–70.

Stoilos, G., Simou, N., Stamou, G., & Kollias, S. (2006). Uncertainty and the semantic web. *IEEE Intelligent Systems, 21*(5), 84–87.

Sun, G., Chen, J., Guo, W., & Liu, K. J. R. (2005). Signal processing techniques in network-aided positioning: A survey of state-of-the-art positioning designs. *IEEE Signal Processing Magazine, 22*(4), 12–23.

Vasant, P., & Barsoum, N. (2009). Hybrid genetic algorithms and line search method for industrial production planning with non-linear fitness function. *Engineering Applications of Artificial Intelligence, 22*(4–5), 767–777.

Wang, X., Ma, Z. M., Yan, L., & Meng, X. (2008). Vague SWRL: A fuzzy extension of SWRL. In D. Calvanese & G. Lausen (Eds.), *Proceedings of the 2nd international conference on web reasoning and rule systems (RR2008)* (pp. 232–233). Berlin, Germany: Springer-Verlag.

Wang, X., Ma, Z., Yan, L., & Zhao, R. (2010). RIF-FRD: A RIF dialect based on fuzzy sets. In *Proceedings of the seventh international conference on fuzzy systems and knowledge discovery (FSKD'10)* (pp. 1912–1916).

Weißenberg, N., Gartmann, R., & Voisard, A. (2006). An ontology-based approach to personalized situation-aware mobile service supply. *GeoInformatica, 10*(1), 55–90.

Zhao, J., & Boley, H. (2008). Uncertainty treatment in the rule interchange format: From encoding to extension. In *Proceedings of the 4th international workshop on uncertainty reasoning for the semantic web (URSW 2008)*. Germany (pp. 1–13).