This document is published in:

Knowledge-Based Systems (March 2012), vol. 27 , pp. 1–17.

DOI: 10.1016/j.knosys.2011.08.017

A knowledge-based system approach for a context-aware system ☆

Nayat Sánchez-Pi *, Javier Carbó, José Manuel Molina

Computer Science Department, Carlos III University of Madrid, Avda de la Universidad Carlos III, 22 Madrid, Spain

ARTICLE INFO

Keywords:
Knowledge-based systems
Context aware systems
Realworld applications
Methodologies
Airport domain
Evaluation

ABSTRACT

Knowledge-based systems (KBS) are advanced systems for representing complex problems. Their architecture and representation formalisms are the groundwork of today's systems. The knowledge is usually derived from expertise in specific areas and has to be validated according to a different methodology than is used in conventional systems because the knowledge is symbolic. This paper describes the design, definition and evaluation of a knowledge-based system using the CommonKADS (CKADS) methodology to formally represent contextual information for the Appear platform. We also evaluate the context-aware information system from the user's point of view using a U2E system and also validate it through a simulated example in a realistic environment: an airport domain, which is a significant step towards formally building KBS applications.

1. Introduction

A significant change is now afoot in how people conceive computing and how they look at their interaction with new technologies. More and more people are coming into contact with computational resources thanks to the development of devices such as personal digital assistants (PDAs) and mobile phones, which have now become very popular and are increasingly being networked (e.g., Bluetooth, IEEE 802.11). The use of this technology gave birth to a new concept: mobile computing. The mobile computing field is receiving increasing attention as many systems are designed for this purpose. Most of them aim to be context aware with the aim of optimizing and automating the distribution of their services at the right time and in the right place. Context-aware computing is a paradigm that was first defined by Schilit [34] some years ago. He claimed that the main components of context were: where you are, who you are with and what resources there are nearby. This paradigm studies methods for modeling and utilizing contextual information. This information can be user location, time, space, type of device, meteorological conditions, user activity, nearby people or devices, etc. A more widely used definition of what context is was given by Dey et al. [14]: ''any information that characterizes a situation related to the interaction between humans, applications, and the surrounding environment''. As can be observed from this definition, any information source can be considered context as long as it provides knowledge relevant to handle the communication between the user and the system. Chen and Kotz [11] classified context into more fine-grained categories: *physical*, *computing* and *user* context information types.

The physical context type is related to environmental factors which can be usually evaluated by specialized hardware mechanisms. The computing context refers to the information which describes the resources available by the computing infrastructure. Finally, the user context refers to the user's profile by focusing on the user needs, preferences, mood, etc.

So, the user is also considered to be part of the contextual information. Kang et al. [23] differentiate two types of context: *internal* and *external*. The former describes the user state (e.g., communication context and emotional state), whereas the latter refers to the environment state (e.g., location and temporal context). Most studies in the literature focus on the external context. However, although external information, such as location, can be a good indicator of the user intentions in some domains, in many applications it is necessary to take into account more complex information sources about the user state, such as emotional status [10] or social information [24]. External and internal context are intimately related, as can be seen in representative examples like service context and proactive systems.

Accordingly, there are several approaches to developing mobile systems such as platforms, frameworks and applications for offering context-aware services. The Context Toolkit proposed by Dey et al. [14], for instance, provides developers with abstractions enabling them to build context-aware applications. The Context

☆ This work was supported in part by Projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, CAM CONTEXTS (S2009/TIC-1485) and DPS2008-07029-C02-02.

* Corresponding author. Tel.: +34 918561327.

E-mail addresses: nayat.sanchez@uc3m.es (N. Sánchez-Pi), javier.carbo@uc3m.es (J. Carbó), josemanuel.molina@uc3m.es (J.M. Molina).

Fusion Network [12] was introduced by Chen et al. [12]. It allows context-aware applications to select distributed data sources and compose them with customized data fusion operators in an information fusion graph. The graph represents how an application computes high-level understandings of its execution context from low-level sensory data. The Context Fabric [21] is another toolkit that facilitates the development of privacy-sensitive, ubiquitous computing applications. Gaia [36] is another platform developed by Roman et al. [28] and also designed to support the development and execution of applications for ubiquitous computing spaces in which users interact with several devices and services at the same time. RCSM is middleware software proposed by Yau and Karim [43] designed to vest applications with the properties of context-awareness and ad hoc communications. It provides an object-based framework for supporting context-sensitive applications similar to CORBA [26] or TAO [17] for fixed networks. Earlier approaches like Entree [9], which uses a knowledge base and case-based reasoning to recommend a restaurant, or the Cyberguide [1] project, which provides users with context-aware information about the projects performed at the GVU Center in Atlanta, where TV remote controllers throughout the building detect user locations and provide them with a map highlighting the project demos available in the users' neighboring area. A recent development is Appear,[1] which is a context-aware platform designed to provide contextual information to users in particular and well-defined domains with a modular architecture.

After reviewing some of the context-aware solutions, it is clear that these applications must be able to adapt to context changes in order to provide users with quality of service. In other words, context-aware systems are expected to utilize contextual information to adapt their behavior based on a predefined set of rules. Such rules are mostly monitored by a system that dynamically adapts its operations based on this information. Appear is a good solution for context management, which is why we selected it to model and test our prototype.

As a result of the increasing development of these kinds of systems and applications, more knowledge-based systems (KBS) are also being developed for use in areas where context-aware system failures can be costly because of losses in services, property, etc. KBS are advanced systems for representing complex problems. Their architecture and representation formalisms are the groundwork for today's systems. The knowledge is usually derived from expertise in specific areas and is validated according to a different methodology than is used in conventional systems because the knowledge is symbolic. Unlike conventional systems, in KBS, domain knowledge is represented explicitly, it is separated from the general reasoning algorithm used for processing it and the internal operation is based on non-deterministic processing, that is why they are adequate for representing context-aware systems.

There are several methodologies that tackle the development of KBS. CommonKADS [37] is, for instance, one methodology that we will use to formally represent a use case. A model set is the main product resulting from the application of this methodology. CommonKADS offers six models to represent context: organization, tasks, agents, communication, knowledge and design. The knowledge model is a very good approach to knowledge representation [42]. It uses three categories (domain knowledge, task knowledge and inference knowledge) to describe what knowledge a specific agent has and what knowledge is relevant for the development of a particular task describing the structure based on its use. Model-based and Incremental Knowledge Engineering (MIKE) [2,3] is a methodology for developing KBS covering all the aspects of the process as of knowledge acquisition. It provides a structured lifecycle facilitating the maintenance, debugging and reuse of the resulting product. Protegé [27,16,40] is another methodology that generates knowledge acquisition tools using reusable libraries. VITAL [38,15] is a lesser known methodology but has similar points to CKADS and MIKE. For instance, it outputs four products, of which the conceptual model is like CKADS and the design model is similar to MIKE. We have chosen CKADS to model our problem because it covers the main points of the development of a KBS from the very start of analysis (for identifying the problem and establishing the feasibility of a KBS-based solution) to the implementation of the project.

The contribution in this paper is the design, definition of a KBS using CKADS in order to formally represent contextual information for the Appear platform. We also evaluate the context-aware information system from the user's point of view using a U2E system developed in previous work and also validate it in a realistic environment: an airport domain, which is a significant step towards formally building KBS applications. This constitutes a complex research problem due to the number of technologies involved (mobile devices, web services, pervasive environments, evaluation techniques, etc.), the heterogeneous nature of the information sources and the number of modules which must be carefully designed and developed to manage each one of the services and functionalities that are provided by the system.

The remaining of the paper is organized as follows. Section 2 describes the main characteristics of the proposed architecture using Appear for providing context-aware services. Section 3 presents the definition of the KBS using CommonKADS methodology. Sec-tion 4 describes the context representation process in Appear plat-form. Section 5 shows a simulated example used to validate our proposal. Section 6, presents an evaluation method for context aware systems from the user's point of view using a U2E system that can be applied to the case of use described in Section 5 and Fi-nally, our conclusions are presented.

2. The Appear platform

Appear is an application providing a solution for a wireless network. This solution enables the distribution of location-based applications to users in the proximity of predefined interest points. All Appear needs is a IP-based wireless network and a Java-enabled wireless device. Appear uses a platform-independent external positioning engine in order to locate devices and calculate their position.

2.1. Appear architecture

The Appear platform consists of two parts: the Appear Context Engine, which is the core of the system, and the Appear Client, which is installed in the device. Applications distributed by the Appear Context Engine are installed and executed locally in these wireless devices.

The architecture of the Appear Context Engine is modular and separates the system responsibilities into server, one or more proxies, and a client (Fig. 1). The Appear Context Server is part of the network management. It manages the applications distributed by the platform and the connections to one or more publishers, or proxies or positioning engines.

The Appear Context Proxy is typically installed inside but can also be located outside the network. Its main function is to eliminate unnecessary traffic between the local network and the main server since the bandwidth is usually limited. It keeps a cache of the active user sessions and the most accessed services. When a user requests to download a service, the proxy checks if it can be served from the cache instead of from the main server in order to carry out its main function.

When a wireless device enters the network, it immediately connects with a local proxy, which evaluates the position of the client

[1] www.Appearnetworks.com.

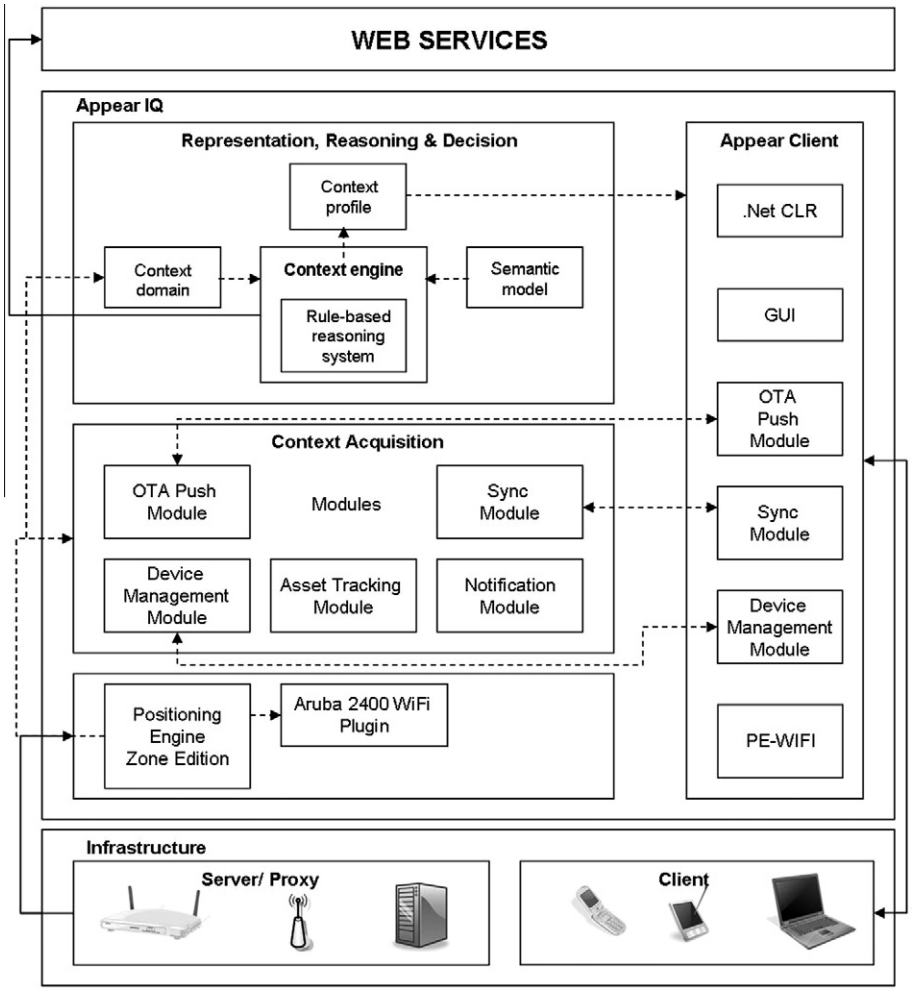

Fig. 1. Appear architecture.

device and initiates a remote connection with the server. Once the client is in contact with the server, they negotiate the set of applications that the user can access depending on his or her physical position.

The Appear solution consists then of the Appear Context Engine and its modules: Device Management Module, Push Module and the Synchronization Module. The three modules collaborate to implement a dynamic management system that allows the administrator to control the capability of each device connected to the wireless network.

The Push or Provisioning Module manages the automatic distribution of applications and content to handheld devices. It pushes services on these devices using client-side intelligence when it is necessary to install, configure and delete user services. There are six steps in Context-Aware Provisioning:

- *Device detection*: The network discovers the client and provides it with the required configuration.
- *Service discovery*: The Context Profile is made up by the Context Engine and it then compiles an offering for the client. The offering is shown as icons on the handheld device using a user interface. The client then decides which service to pull by clicking on the respective icon.
- *Download*: The download of the necessary resources is the first step after service discovery.
- *Install*: Once the resource is available on the device it will be installed.
- *Execute service*: Once the service has been launched, the application is executed.
- *Discard service*: A service will be discarded when a user leaves the network or if the context condition for a service changes.

The Device Management Module provides management tools to deploy control and maintain the set of mobile devices. The context-aware actions on the client side are:

- The configuration elements that describe the specific steps to be taken by the client. They are initially installed together with the client and then updated using the Synchronization Module.
- *Context conditions*: A condition associated with the current context is applied to determine if the action is applicable. This is done by the client rule engine.
- *Mirroring*: This is a mechanism used by the client to monitor file updates on the device. These updates are replicated to a secondary device such as a storage card or a remote host using FTP/HTTP.

The Synchronization Module manages file-based information between corporate systems and the mobile handheld devices. Device Management is continuously updated with up-to-date versions of the configuration files. There are three steps in the Synchronization Module:

- The Synchronization Module compiles contextual data to gain an understanding of the user's information needs.
- Available data is filtered against the user's context to determine which should be the most relevant information.
- The Synchronization Module automates synchronization, detecting and synchronizing files that have changed. This is a dynamic synchronization of the profile based on user and role, location, time, device status and connectivity.

All of these modules are made context aware using the Appear Context Engine.

2.2. Appear context representation

In Appear, the Appear Context Engine gathers the context of user data and builds a model based on the needs of the end user. It implements a rules engine, which determines which service is available to whom, and when and where it should be available. Services are filtered against a profile, and when it is decided that data are relevant, the information is proactively pushed to the device.

As mentioned, the Appear Context Engine gathers all the context information about the device and produces a context profile for that device. The main components of this model are Context Domain, Context Engine, Context Profile and Semantic Model.

The Context Domain is a set of context values that the system can monitor. All values of the context domain are given without any internal relationship. It is fed with context parameters that measure real-world attributes that are transformed into context values. Context parameters include physical location, device type, user role, date/time, temperature, available battery.

The Semantic Model is the administrator model of the relationship between different context parameters and how these should be organized using context predicates.

The Context Engine (Fig. 2) is the model that maps the Context Domain to the Semantic Model outputting the Context Profile.

There are several lowest level objects, called context parameters managed by the Context Engine and they should be taken into account when deciding what to do in the system. To get into a more abstract level Appear creates more complex predicates combining and constraining the values of these context parameters and other context predicates.

A predicate is a set of simple rules or conditions combined to create a more complex predicate. A predicate is constructed out of one or more conditions and an evaluation criterion that can be "match all" or "match any". *Match all* requires that all conditions are evaluated as true. *Match any* requires that at least one condition is evaluated as true.

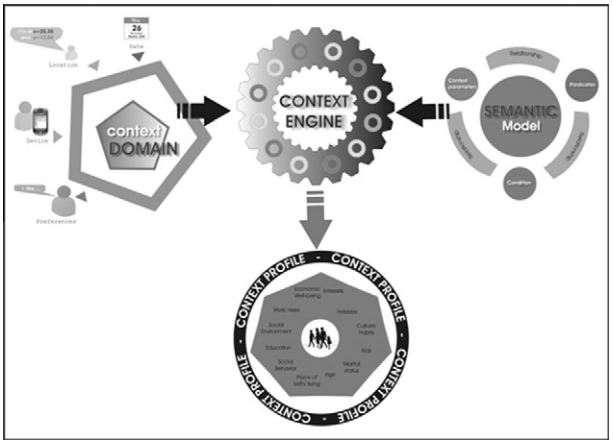

Fig. 2. Context engine representation.

A condition can be a context parameter condition or a condition on another predicate. The context parameter condition is a constraint on a specific parameter string value. The keyword can be evaluated against a condition ("is", "is not", "contains" and "does not contain") and the expected value (i.e. *zone_airport = zone_airport is true*). The predicate condition is a reference to another context predicate and the expected result is the evaluation of that predicate (i.e. *if zone_airport then news_service*).

Context Profile components are classified into different categories: behavior (social behavior), personality (habits, interests, etc.), demographic information (age, place of birth, marital status, kids, etc.), economic information (economic/well-being; work/area, etc.) and social indicators (social environment, hobbies, etc.). The Context Profile contains all context values and values of predicates based on the context values.

Context information in the system is used throughout the entire service lifecycle. The rules engine filters and determines which service should be pushed to each user, when and where.

2.3. Scalability issues

Context-aware systems gather contextual information from many sources and process it to create the final representation available for a variety of clients. The context-aware system developed has a hierarchical architecture with the following layers: Web Services/Application Layer, Context Reasoning Layer and Sensor Layer The hierarchical architecture reflects the complex functionality of context-aware systems as shown in the following brief description of the functionality of particular layers:

Sensor Layer: The lowest level of the location management architecture is the Sensor Layer which represents the variety of physical and logical location sensor agents producing sensor-specific location information.
Context Reasoning Layer: This layer takes sensor-specific location information and other contextual information related to the user and transforms it into a standard format. This is where the reasoning about context takes place.
Web services/Application Layer: This layer interacts with the variety of users of the context aware system and therefore needs to address several issues including access rights to location information (who can access the information and to what degree of accuracy), privacy of location information (how the location information can be used) and security of interactions between users and the context-aware system.

The issue of scalability of context-aware systems is important. For instance, there could be a great difference between the frequency of location updates from Aruba sensors and the frequency of location information requests from the users of the system. Moreover, once a location is determined at the Context Layer it is cached as a complete location description which will be valid until another update pertaining to that user. As a result, many requests for a single user's location can be served without significantly increasing the cost of generating the location from individual location fragments if location information updates are not very frequent.

3. KBS definition for an airport domain using CommonKADS (CKADS) methodology

The representation of knowledge in KBS is varied. For instance, the development of Semantic Web technology gives the information on the current Web a precise meaning and machine-interpretable form providing computers and people processing the same data with a common understanding of what the terms mean [6].

The Semantic Web is also used in KBS development through ontologies that enable the construction of KBSs through reusable components across domains and tasks [18]. Ontologies are used to represent domain knowledge in knowledge-based programs. This is achieved using formal declarative representations of the domain knowledge; that is, sets of objects and their describable relationships [20]. Researchers in the area of knowledge modeling have started to realize the importance of ontologies in developing domain models since the underlying principle of modeling is to achieve agreed representations in a unified manner for the domains in which they are investigating. Ontologies can be described using Semantic Web languages, such as DAML + OIL, SHOE and RDF, and these descriptions are used to create the knowledge base of the KBS [25]. This way, the KBS developer can focus on domain knowledge representation instead of markup tags and correct syntax to build KBS faster.

There are also several methodologies for the development of these systems. The best known is KADS, and this is the methodology that we will use in this section to formally represent our problem defined in Appear. KADS (Knowledge Acquisition and Document Structure) and its successor CommonKADS [37] (which is the de facto European standard) is a knowledge engineering methodology.

CKADS is a structured approach to the development of KBS and, as such, is to be seen in contrast to unstructured approaches such as rapid prototyping. CKADS does not require a commitment to any specific implementation paradigm. According to KADS, the development of a KBS is to be seen as a modeling process, during which acquired knowledge is modeled at different levels of abstraction.

KADS identifies three levels of models:

- The process level identifies the tasks involved in the domain, the nature of data flows and stores, and the assignment of ownership of tasks and data stores to agents.
- The system level, the co-operation model, details the interactions between the system and external agents, and how the internal agents interact. The co-operation model is used to separate the user task model from the system task model, and allows the knowledge wholly internal to the system to be readily identified.
- The expertise level corresponds to an expertise model. This divides the task of describing domain and expert knowledge and its use within the system into a number of supportive tasks. The layer-based framework for expertise consists of:
  - The domain layer is comprised of static or slowly changing knowledge describing concepts, relations, and structures in the domain.
  - The inference layer reformulates the domain layer in terms of the different types of inferences that can be made.
  - The task layer defines knowledge about how to apply the knowledge in these two layers to problem-solving activities in the domain.
  - The strategy layer concerns selecting, sequencing, planning and repairing tasks.

Within this framework knowledge engineering becomes a structured search for appropriate strategy, task, inference and domain models. The layers can be modeled in three steps: (a) determine the static features or domain ontology, (b) obtain the inferences that describe the dynamic side, and (c) group the inferences sequentially to form tasks.

3.1. Domain layer

This layer represents the representative knowledge of the domain. It is here where the information and the static knowledge are described.

Concepts in CKADS [7] are used to define object collections with similar characteristics. In the case of the airport domain, the concepts are identified as:

- *Airport*: This is the high-level concept representing the domain of discourse.
- *Location*: This includes the airport location, zones and the user location.
  - o (x,y): The coordinates of the airport location, zones and the user position.
  - o *Zone*: A segment of the airport area distinguished from adjacent zones by different characteristics and the coordinates represents the zones in the domain.
- *Airport_zone*: This is the general zone. It contains the other zones. Every service available in this zone can be seen in the other zones.
- *Customs_zone*: This is where the customs authority is located in order to control the flow of goods.
- *Commercial_zone*: This is where restaurants and stores are located.
- *Offices_zone*: This is where the airline offices are located.
- *Check-in_zone*: This is where the check-in desks are located.
- *Jetway_zone*: This is zone leading to the aircraft. It contains the plan of the current flight.
- *User*: Every person who has a role in the domain.
  - o *Role*: Role of each user in the domain.
- *Pilot*: Airline captain and pilots.
- *Passenger*: Passenger.
- *FTO*: Flight engineer
- *CBC*: Cabin crew.
- *ASBG*: Airline ground staff: boarding.
- *ASIC*: Airline ground staff: incidents.
- *ASSV*: Supervisor.
- *ASCH*: Airline ground staff: Check-In.
- *Offering*: A class containing a set of services grouped by categories.
  - o *Category*: A class containing a set of grouped services.
  - o *Services*: Applications or notifications.
- *Flight_Plan*: A web service that delivers the flight plan of the current flight. It is available to pilots and FTO.
- *Crew_License*: A web service containing the flight license of the crew assigned to the current flight. It is available to pilots only.
- *Safety Demo*: An informative document about safety available to pilots and passengers only.
- *Weather Map*: An informative document available to pilots, FTO and CBC.
- *News*: Access to the URL of major Spanish, English and French newspapers. It is available to pilots and passengers.
- *Passenger_Control_Report*: A web service that delivers the passenger list of the current flight. It is available to pilots and FTO.
- *ID*: A barcode used to improve the ID data capture. It is available to pilots, CBC and FTO. Code 39 is used to encode the one dimensional barcode.

These concepts can be represented in different ways. Fig. 3 shows the representation of the user concept using CML language, where the characteristics are represented by means of "attributes" unlike other approaches, where functional information is included as well as the characteristics.

Another way of representing these concepts is to use a domain ontology. The domain ontology is defined by means of three sets: a set of terms, a set that contains their definitions (typology) and a third set of relations between the terms (taxonomy). The ontology provides the explicit conceptualization of the domain terms to support the knowledge base implementation which is prepared for use

```
CONCEPT user ;
DESCRIPTION :
  "Every person who has
   a role in the domain"
ATTRIBUTES :
  Name: STRING;
  Address: STRING;
  Age: NATURAL;
  Role: value-role;
  …
AXIOMS
  …
END CONCEPT user;
```

```
VALUE-TYPE value-role ;
  TYPE: ORDINAL;
  VALUE-LIST: (pilot,
      passenger, FTO,
      CBC, ASBG, ASIC,
      ASSV, ASCH )
END VALUE-TYPE value-role;
```

Fig. 3. CML definition of the user concept.

by the applications to perform different tasks. Fig. 4 represents the specific conceptualizations for the particular domain of an airport.

CKADS has an extension of the initial phases for the development of ontologies that covers its entire lifecycle, from the feasibility study until the maintenance phase. Fig. 4 contains the representation of the domain knowledge for the airport. The definition of the context in the domain of an airport was partially presented in [29].

Once we the concepts have been defined, relationships can be established between them. We need to represent an *antecedent* and a *consequent* in order to define the different type of rules in our domain schema. It is also necessary to represent the *connection symbols* used to connect the antecedent to the consequent.

The input/output curve represents a global relationship or a group of partial I/O relationships. The Domain Knowledge is gathered from these relations between input ranges and output ranges. For instance, if we know a passenger user of our domain is in a check-in zone, he will be offered the check-in service. In this case, the rule can be assessed as:

Rule Inference: If a user with a certain role is in a certain zone at a certain time, then he or she will be made an offering.
Rule Inference: Given an offering and a user category, then some services are provided to the user and others are not.

3.2. Inference Layer

After defining the Domain Ontology, the domain dynamic component must be obtained. This dynamic aspect deals with the system input/output behavior that is stated as a set of production rules. In this layer we will describe how the static structures defined above will be used to develop the reasoning process.

Inferences are completely described through a declarative specification of their inputs and outputs (dynamic roles). Returning to the CML language representation, Fig. 5 shows the CML specification of an inference, where the input is a role "case" representing the knowledge elements of the domain and the output is the role "abstracted-case" representing the qualified description of the input.

Fig. 6 shows the "case" and "abstracted-case" roles representing the smoking-related characteristics in the airport domain.

3.3. Task layer

When both the Domain Ontology and Dynamics have been defined, we have to divide inferences into tasks, where tasks are similar to traditional functions but the data manipulated by the task are described domain-independently. They describe the input/output and how a task is performed through an ordered decomposition into subfunctions, like another task, inference and transfer function. They form a small program or algorithm that captures the expert's reasoning strategy.

The particularity of CKADS is the partial reuse of knowledge models in new applications, proposing a catalog of task templates comprised by a tentative inference structure, a typical control structure and a typical task-based domain schema, which are specific for a task type.

Of the task types suggested by CKADS, we consider our problem as a special case of an assessment task template, where an assessment problem consists of finding a decision category for a case based on domain-specific norms.

The typical control structure suggested by CKADS for assessment problems is shown in Fig. 7.

4. Appear context configuration

Section 2 explained how Appear represents the context and how the KBS for the airport domain was defined.

4.1. Domain knowledge configuration in Appear

The *offering mapping* is where offerings are mapped to zones and times. A mapping consists of a condition and an offering. The meaning of the mapping is that when the context of the user's de-

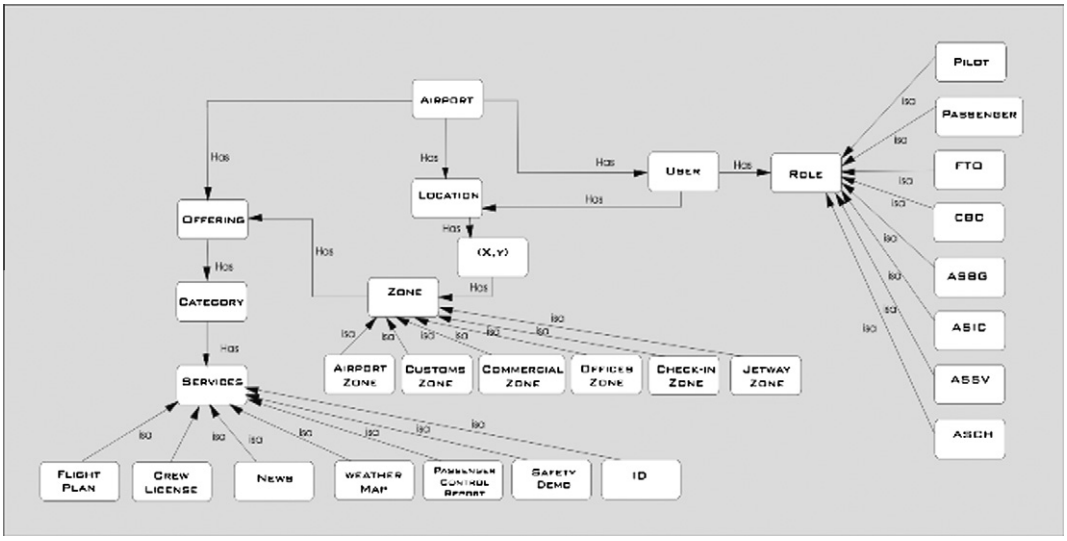

Fig. 4. Domain knowledge in the airport KBS.

```
INFERENCE abstract ;
  OPERATION-TYPE :ABSTRACT;
  ROLES:
    INPUT: case;
    OUTPUT: abstracted-case;
    STATIC: knowledge  -abstraction;
  SPECIFICATION:
  "Every time this inference is executed, it generates a transformation of
   the input data producing a more qualified description of the case"
END INFERENCE abstract;
```

Fig. 5. CML specification of an inference.

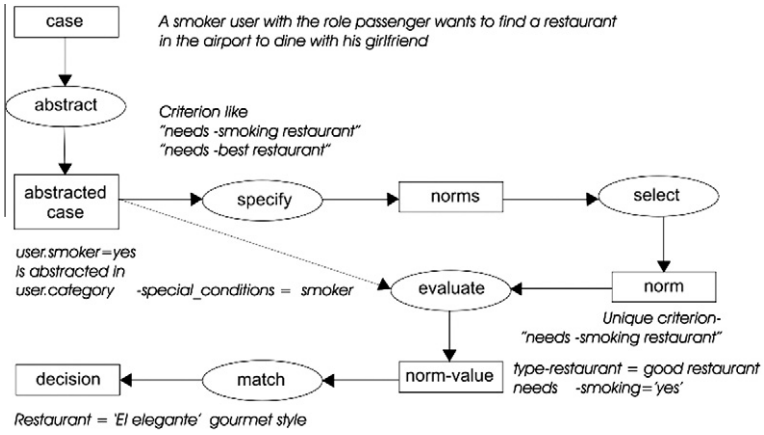

Fig. 6. Inference diagram for decision making on choice of an airport restaurant.

```
while new-solution abstract(case-description -> abstracted-
  case)
do
case-description := abstracted-case;
end

while specify(abstracted-case -> norms);
 repeat

 select(norms -> norm);
 evaluate(abstracted-case + norm -> norm-value);
 evaluation-results:=norm-value union evaluation-results;

 until has-solution match(evaluation-results -> decision);
```

Fig. 7. Control structure.

vice matches a condition, the client will be eligible for the offering During the context configuration procedure, we need to define every concept in our domain knowledge and represented in the ontology described in Fig. 4. These concepts are:

- *Service*: An application or data available to a client.
- *Category*: A set of logically grouped services.
- *Offering*: A set of available applications for a client.
- *Zone Mapping*: Where offerings are mapped to zones and times.
- *Roles*: Used to grant users access to specific categories. It is possible to specify one or more groups that have access to a category.
- *Users*: Users are only used if the system requires login.

There are several setting values for the context elements that form the condition. They are: *Zone name*, *Time period*, *IP pattern* and *Roles*. The *Zone name* is known in the positioning system. Once the positioning system gives the location coordinates of the user's device, the plug-in defined in Appear, which in our domain is divided into zones, translates and assigns a Zone name to these coordinates. The *Time period* assures that the mapping is valid only when the time period is valid. The *IP pattern* restricts access to users whose IP addresses comply with a particular pattern. The name of the offering that should be used when a condition matches is typed in the Offering field.

Icons, offerings, categories and *services* are the other concepts that need to be defined in order to complete the offering mapping. Offerings, categories and services can all use icons that will be displayed in the user's interface. *Icons* are grouped by their usage into icons for services, icons for categories and icons for offerings. A *category* is a group of services. A category restricts access to services (Fig. 8). *Services* in Appear are easily configured, and it is possible to create services of different types. There are templates for creat-ing document services, web services, native services (Windows services) and Java services.

For a document service the file must be uploaded; for a Web Service, the URL for the web page must be specified; for native services, the executable file must be specified, and for Java services, the jar file and the main class should be specified.

And, finally, the *offering* concept groups categories and services. One offering can contain several categories (Fig. 9), and each category can contain a number of services. The user's device can only display one offering at a time.

A time period, as its name suggests, consists of a name and a definition of a period of time. Fig. 10 shows the configuration of a period of time. Period name, start time and duration are concepts that need to be configured.

The first time a device connects to the system, it is assigned a unique key by the Appear Context Engine. After that, each device can be enabled or disabled individually, as shown in Fig. 11.

The services are automatically provisioned in the Appear Context Engine as each user's context is identified: role, zone, location, time period, etc. After defining the users (Fig. 12), we need to

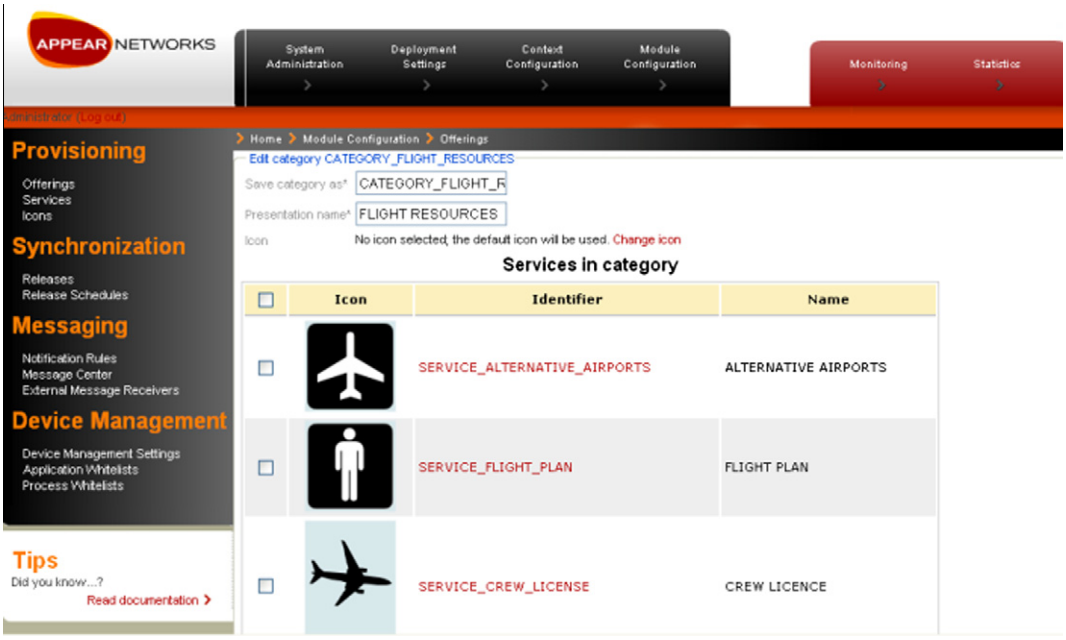

Fig. 8. Definition of a category with Appear.

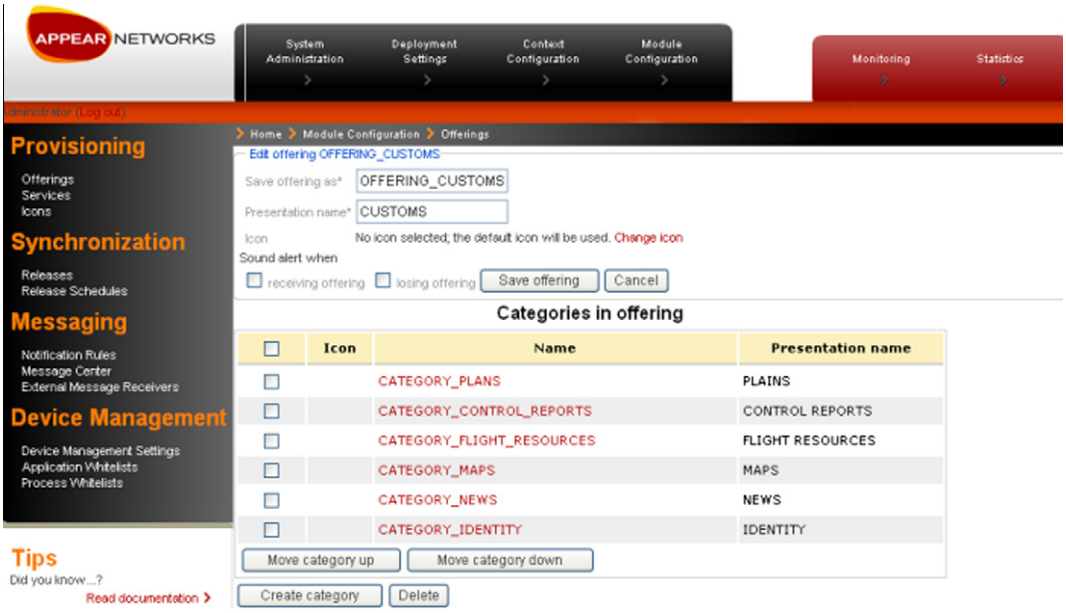

Fig. 9. Definition of an offering with Appear.

create the roles (Fig. 13) that are used to give users access to specific categories of services.

It is possible to specify role restrictions on categories. If a category has no roles, every user has access to the category. To restrict access, it is possible to specify one or more groups that have access. This means that only users that belong to the specified role can gain access to the category. Other users will not see these categories in the client.

4.2. Inference knowledge configuration in Appear

As explained abstractly, there are concepts in Appear, like predicates, context predicates and conditions (Fig. 14), that are combined to create rules which are part of the rules engine in the Appear Context Engine. This is what determines which service is available to whom, and where and when it should be pushed to the user's device.

A new predicate is created as shown in Fig. 15, adding a condition with a keyword in the key field, selecting the required condition ("is", "is not", "contains") and then typing an appropriate value to test against the value field.

Relations between corresponding input and output segments could be assessed as:

Rule Inference: *Given a zone and time of the client, then the client uses some offering.*

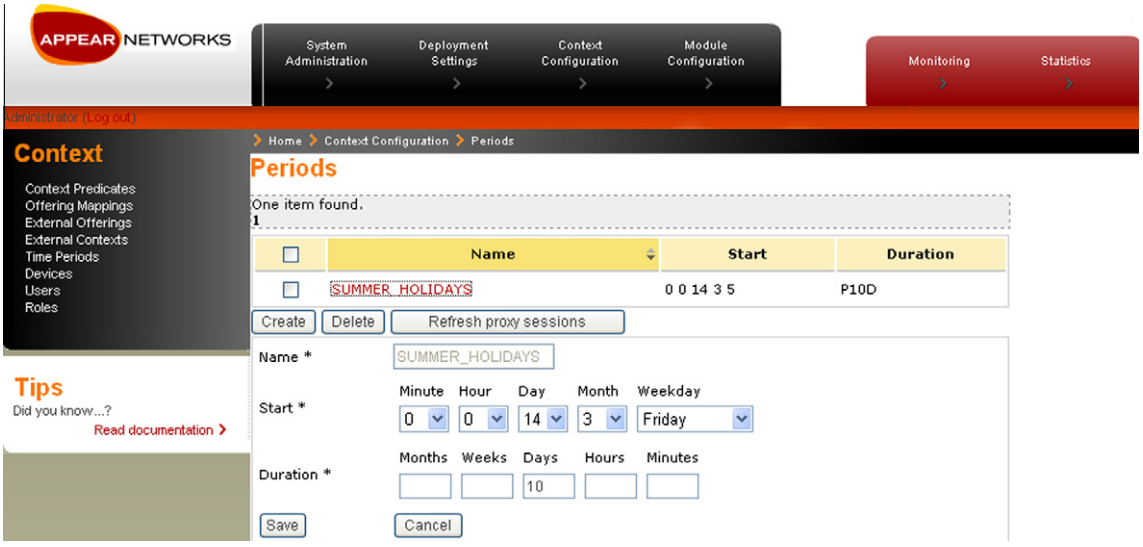

Fig. 10. Periods definition with Appear.

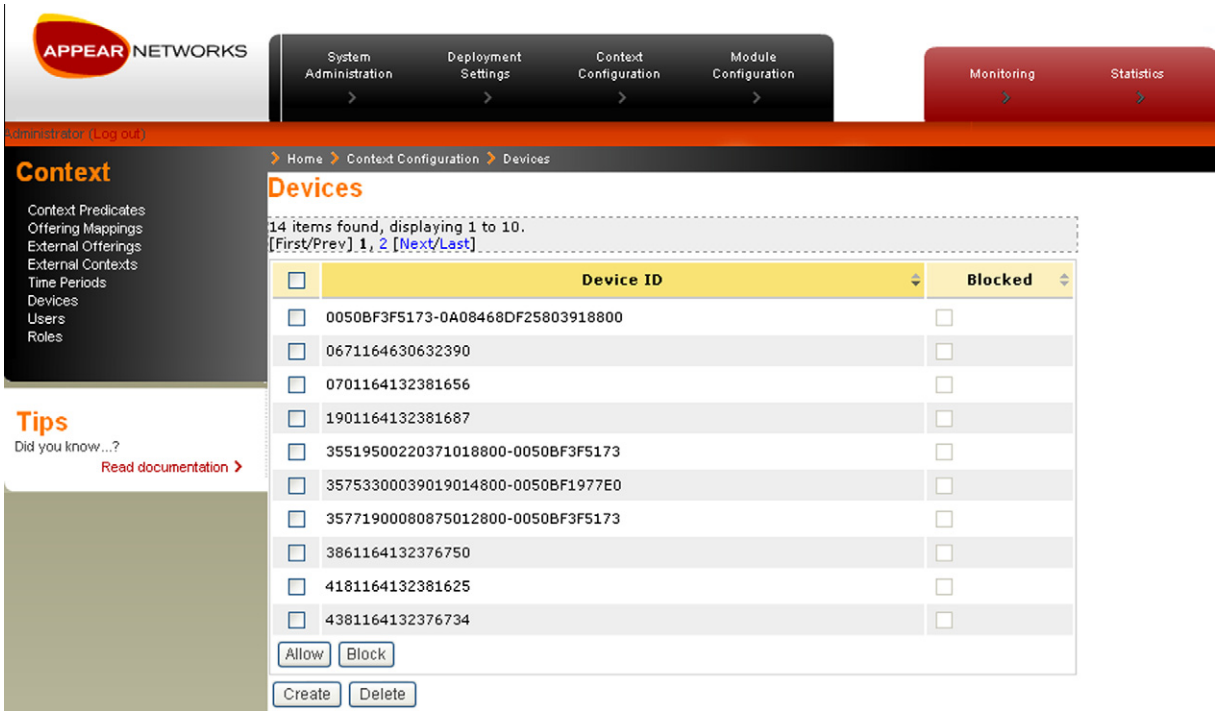

Fig. 11. Devices definition with Appear. Enable or block access.

There are several setting values for the attributes of the domain concepts that form the condition. They are: *Zone name, Time period, IP pattern* and *Roles*.

- The *Zone name* is known in the positioning system. Once the positioning system gives the location coordinates of the user's device, the plug-in defined in Appear, which in our domain is divided into zones, translates and assigns a Zone name to these coordinates.
- The *Time period* assures that the mapping is valid only when the time period is valid.
- The *IP pattern* restricts access to users whose IP addresses comply with a certain pattern. The name of the offering that should be used when a condition matches is typed in the Offering field.

Icons, offerings, categories and services are the other concepts that are need to be instantiated in order to complete the offering mapping, since an offering consists of several services, but a category restricts access to some of these services corresponding to the offering. For this reason, they form another type of inferences.

Rule Inference: Given an offering and client category, then some services are provided to the user and others are not.

5. Validation of functionalities of the KBS using a simulated example of the airport use case

In this section we will present the validation of the desired functionalities of our KBS using a realistic simulation of a particular

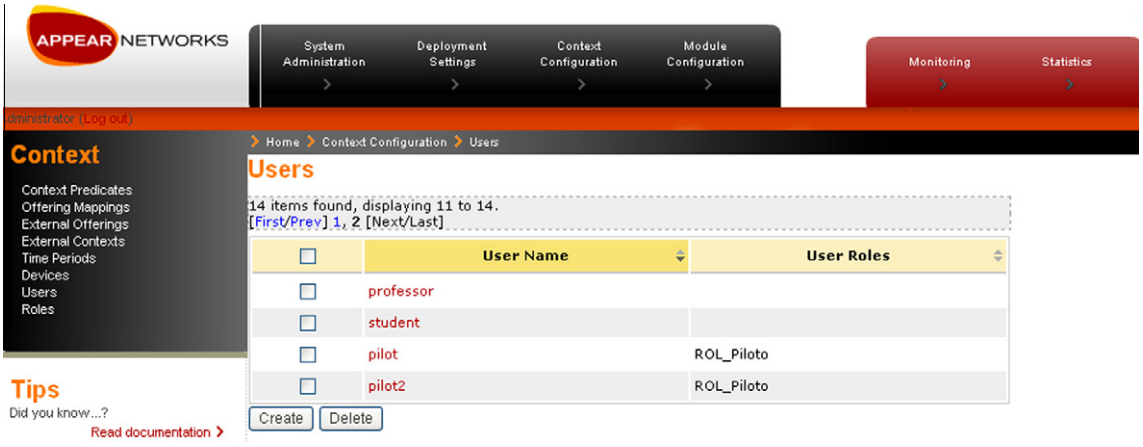

Fig. 12. Users definition with Appear.

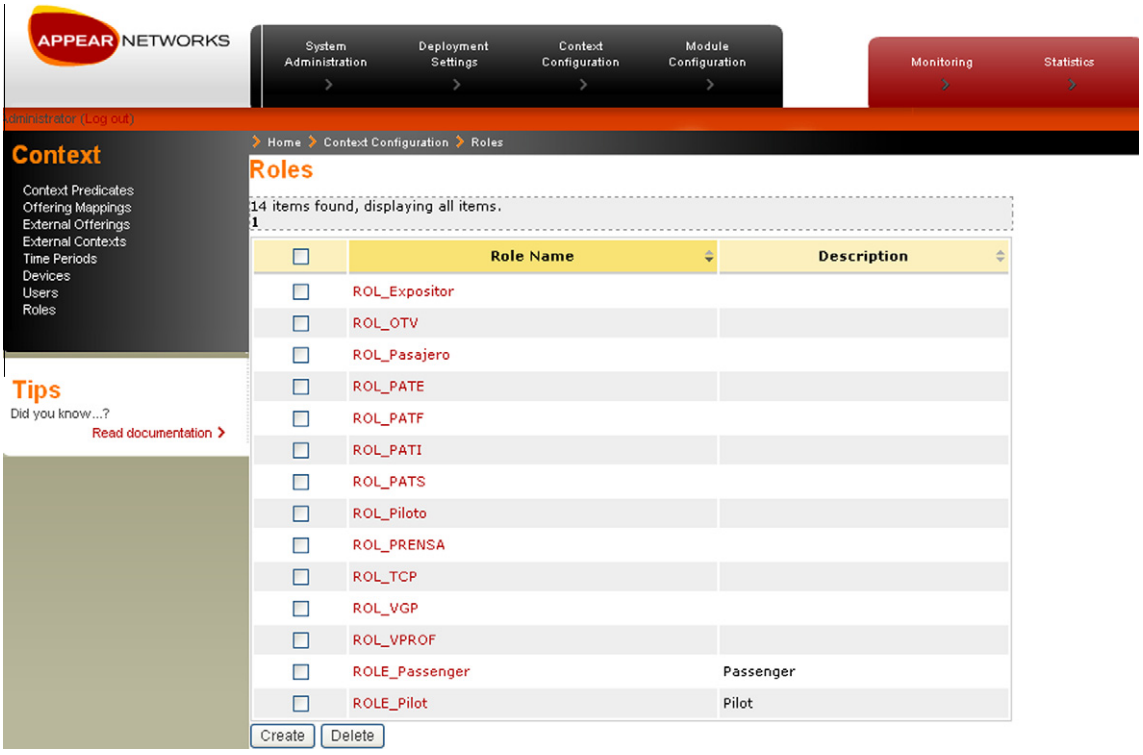

Fig. 13. Roles definition with Appear.

use case. This use case shows an approach to context-aware services provisioning in the context of a traditional airport and the services offered to the airline staff and passengers. We partially present this case of use in the International Symposium of Distributed Computing and Artificial Intelligence (DCAI 2009) [30].

Note that we tested our system on an HTC Touch terminal with a TI OMAP™ 850, 200 MHz processor and Windows Mobile 6. We also deployed a Wi-Fi positioning system in order to locate user terminals. Particularly, we used an Aruba[2] WLAN deployment with a 2400 mobility controller that supports up to 48 access points and 512 simultaneous users and several AP 60/AP61, which are single/dual radio wireless access points. Aruba Networks [17] has a location tracking solution that uses an enterprise-wide WLAN deployment to provide precise location tracking of any Wi-Fi device in the research facility. The RF Locate application can track and locate any Wi-Fi device within range of the Aruba mobility infrastructure. Using accurate deployment layouts and triangulation algorithms, devices can be easily located with an accuracy of about 8 m. Although there are many alternative indoor positioning systems, we decided to use Aruba because it has the useful capability of permanently configuring APs in 'listening' mode in order to avoid missing transmissions. This is a very valuable capability because the air monitors (AMs: dedicated RF monitors) contribute not only to location accuracy but also improve security coverage, detecting RF sources that may be security risks or interferers. The only drawback of using dedicated AMs is that they add to the capital costs of the network.

The use case will be explained with two users of the system: Don and Donna who are partners traveling to the airport. Donna is a passenger who is to take a flight to London, and Don is an

[2] www.arubanetworks.com.

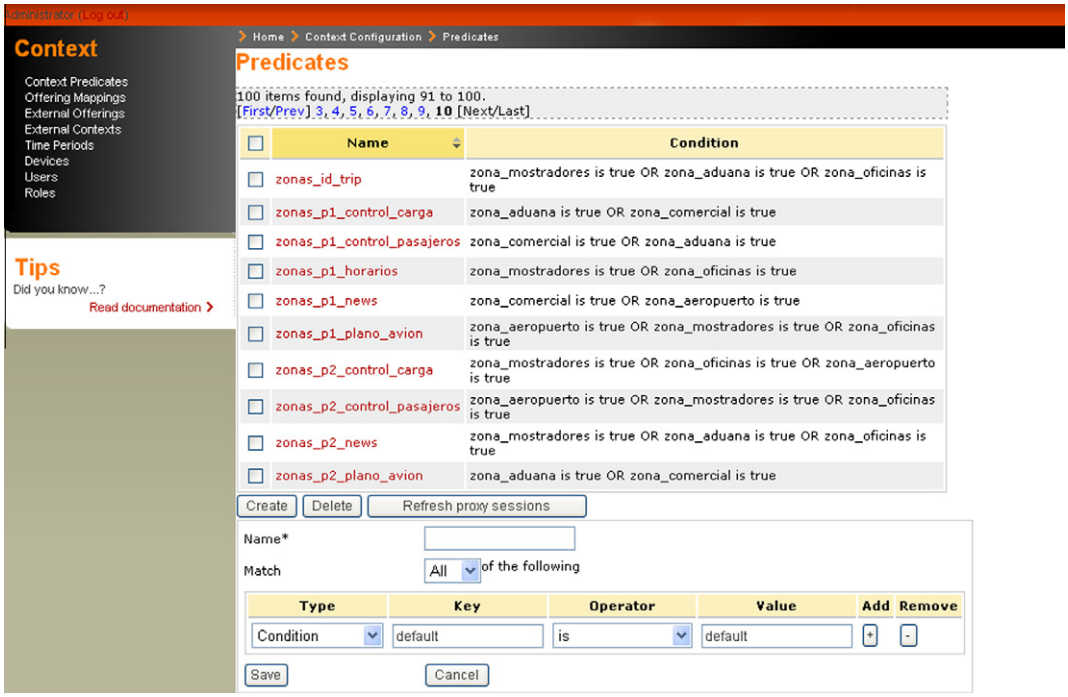

Fig. 14. Predicate definition with Appear.

Fig. 15. Predicate in Appear.

Air France pilot. When they enter the wireless network, the position of each user's device is evaluated, and it starts to negotiate which set of applications the users can access depending on their physical position and their role in the system. Our illustrative simulation will describe the provisioning solution for the users first in the commercial zone and later in the customs zone. It will illustrate the different categories of services to which users have access when they are positioned in the same zone depending on their role in the system.

It will also illustrate what impact the concepts, relationships and inferences described in the dynamic domain knowledge on how services are provisioned for each user and how this is represented in the UI.

5.1. Provisioning of services in the commercial zone

As they move around the airport, users with different roles in the system receive different offerings of services. Our lead actors, Don and Donna, logged into the system and received different offerings of services as they moved around (Fig. 16).

Context information in the system is used throughout the entire service lifecycle: selection based on the context profile, filtering of individual services, and enhancement of services at boot or runtime and the constant feed of context information to services during execution to allow service adaptation depending on the zone in which the user is positioned and his or her role in the system. Dona and Donna arrive at the commercial zone. Fig. 17 shows how Don, who is a pilot, receives offerings of categories of services related to his role in the system: maps, control report, load control, support control, news and fight resources. Then he checks the flight resources category, which has a group of services like alternative airports, flight over license, crew license and flight plan. All of these services were designed as Java services. If he decides then to check the flight plan service, for instance, the system uses his ID to check the information in the server and return the information request as a PDF file.

The other user, Donna, is positioned in the same zone as Don, but receives different offerings (Fig. 18). News, maps, stores and restaurants are the categories of services that she is offered in the commercial zone. If she accesses the stores category, she can then choose one of all the brands that have a store in the airport and get a location plan of the store.

5.2. Provisioning of services in the customs zone

When Don and Donna arrive at the customs zone, the related services are automatically provisioned as the right rules are evaluated. Fig. 19 shows the different categories of services that they will each receive while in the customs zone. The passenger has ac-cess to the maps and news categories only. Donna chose the news category, and this is when Spanish news, English news and French news Appear on Donna's UI. The pilot has different categories, like maps, control report, load control, support control, flight resources and a new category: ID. News services were designed as URL accesses.

As the positioning system can only track and locate the Wi-Fi device within range of the Aruba mobility infrastructure, this demo ends when the users leave the airport domain or when they log off the system (Fig. 20).

The concepts, attributes and relationships described and represented above are clearly mapped to the UIs, where services provisioning were illustrated in the particular case of two users with different roles in the system moving around the airport zones and getting different services.

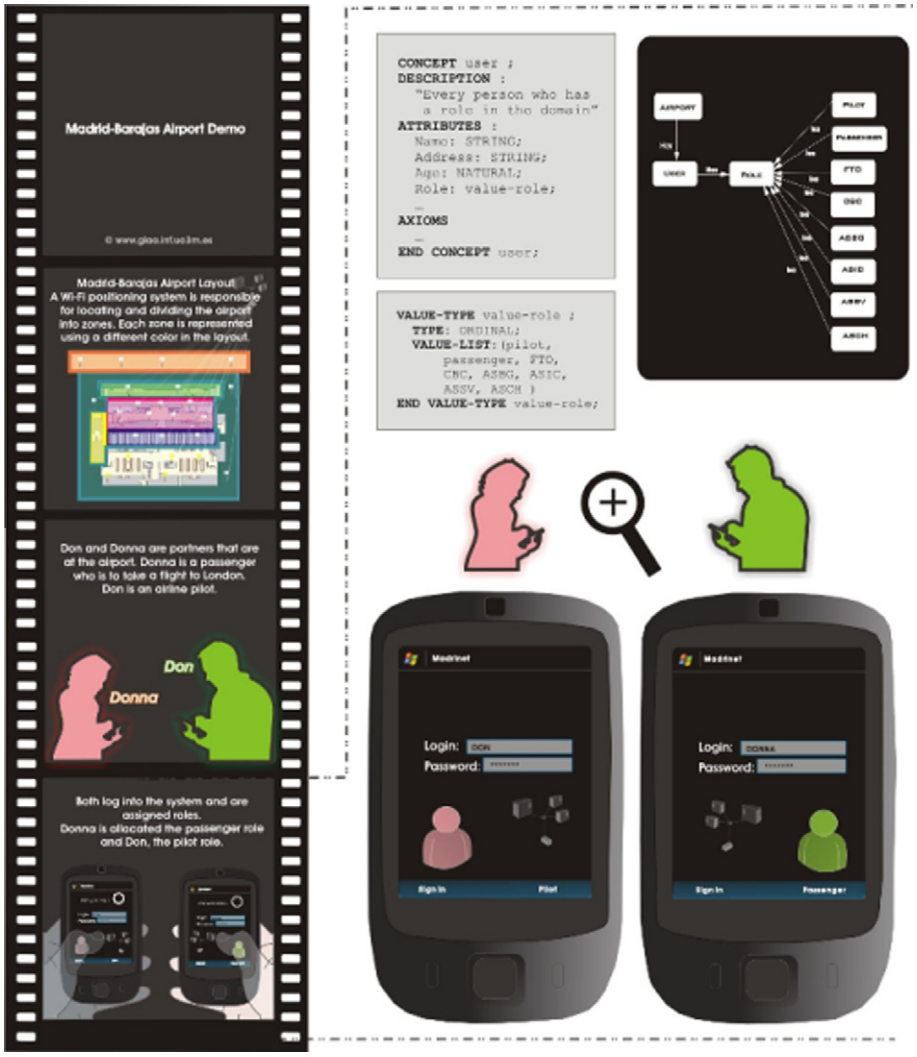

Fig. 16. KBS validation through a simulated example. Frames 1–4.

The execution of this simulated example has validated the desired functionalities of our KBS:

- The ability of positioning users through Aruba has been tracking users and it has been correctly integrated with Appear Platform.
- Roles and locations have determined the offered services in a dynamic way, showing the adaptation abilities of our system.

Therefore, we have validated the integration of complex technologies (Aruba, Appear) with a CKADS-created KBS, and we have validated its context-aware adaptive behavior. Although validation with real users in a real airport or even using a dataset from real users, would be much more desirable, academic context aware systems have been uptodate validated through simulated examples (for instance, [22] as we did or have validated the correctness of context data are required and not in the system itself [8].

6. User-centered evaluation of context-aware systems

When looking at context-aware there are still very basic problems relating to evaluation. It is important before evaluating a system to figure out what is the evaluation goal. In our case, such a goal is to evaluate enhanced user experience and beyond the evaluation goal it is also a central concern about what to evaluate. There are several methods Albrecht Schmidt [35] and approaches used to evaluate context-aware systems. But, as there are no established evaluation frameworks in literature, we used, first of all, a pre-implementation evaluation method as the "Wizard of Oz", Dahlback et al. [13]. In this method a human mimics the computer's behavior to save implementation time. Humans are used to mimic or simulate tasks in which they're better than computer, for instance, the prediction of behavior. And latter we used a meth-od called "revisiting the hypotheses" Albrecht Schmidt [35], which he divides it into four hypotheses to be investigated. First of all, some basic questions related to the context acquisition (that means that when assessing a situation not all information is taken into account but only the information that is discriminating), an-other two concerned with modeling of context (the first one: the domain context of an entity is more universal than of a complex system, suggesting building prototypes bottom-up rather than top-down; the second one: claiming that new contexts can be cre-ated when contextual knowledge is already available and that this leads to a more flexible use) and the last one is concerned with pro-totyping the context-aware system. But there is an important issue not addressed in these kinds of approaches: the user-centric evaluation paradigm and the self-adaptation problem. Especially with user-centric evaluation, these systems are expected to automatically and autonomously adapt to maximize user satisfaction.

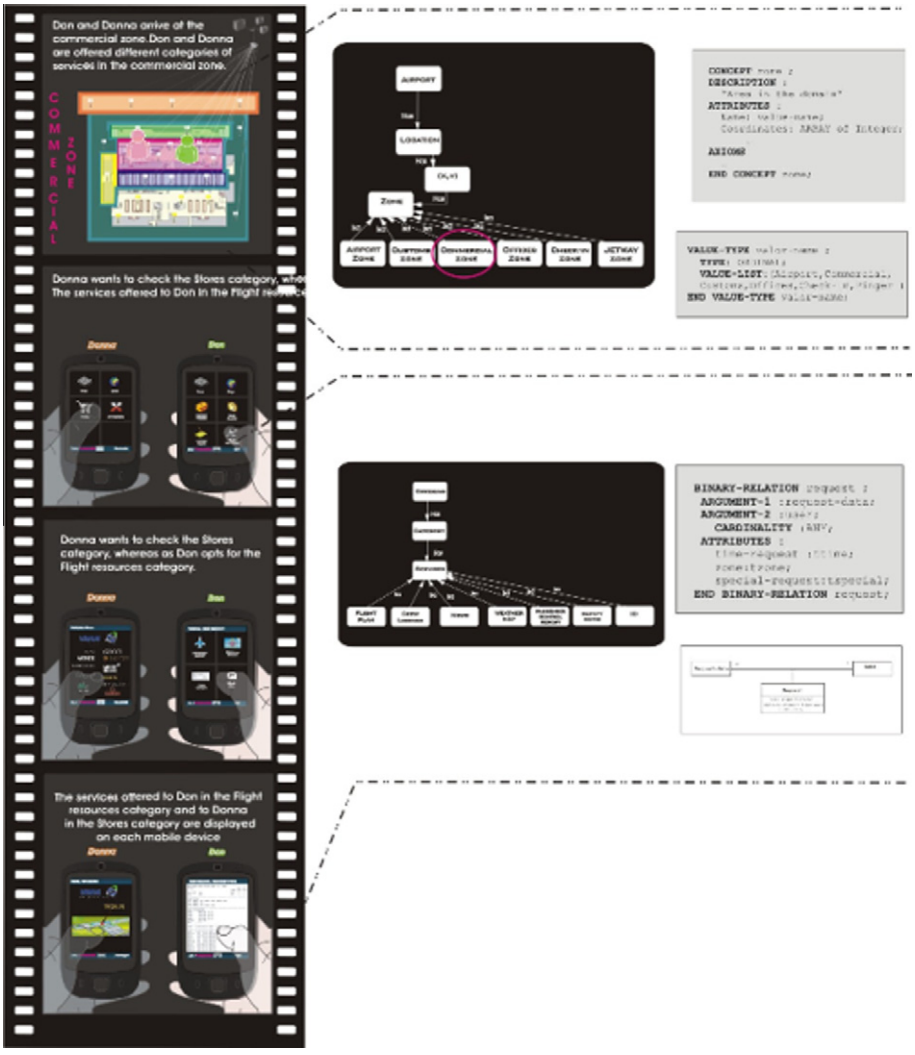

Fig. 17. KBS validation through a simulated example. Frames 5–8.

Fig. 18. KBS validation through a simulated example. Zoom of the UIs of the passenger and pilot user in the commercial zone.

Fig. 19. KBS validation through a simulated example. Zoom of the UIs of the passenger and pilot user in the customs zone.

Non-obtrusive user feedback collection is one of the challenges of new context aware applications in ubiquitous environments. Context-aware applications need to take advantage of user feedback to evaluate its contexts and adapt them where necessary. Inaccurate context data can lead to incorrect behavior that is why this kind of applications needs to adapt and to eliminate issues arising with

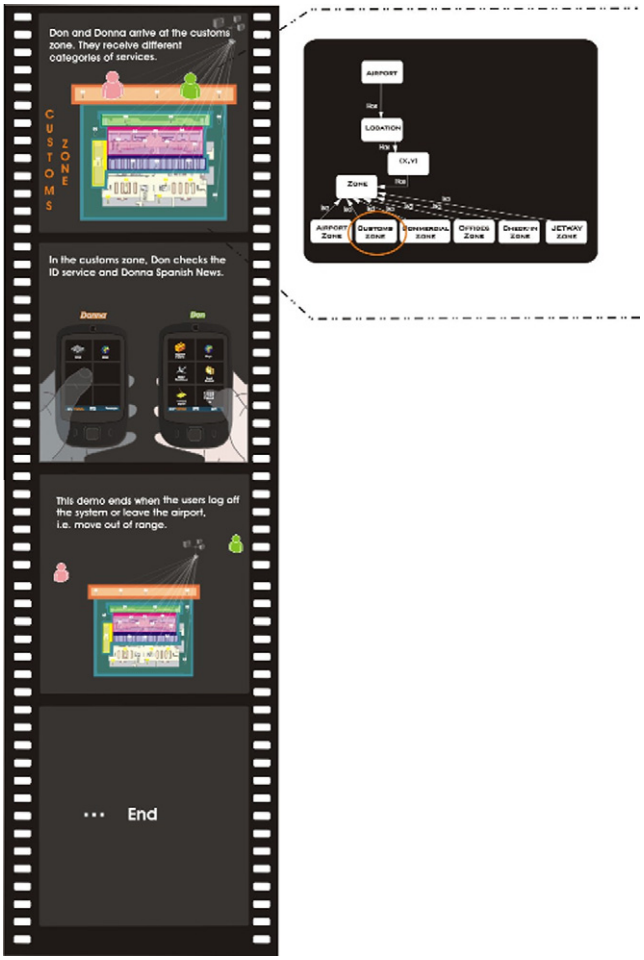

Fig. 20. KBS validation through a simulated example. Frames 9–*n*.

initial contexts definitions. For that reason we need a data structure to store user profiles.

The problem of modeling user profiles has been widely researched in other fields, for instance. User profiles are typically either knowledge-based or behavior-based. Knowledge based user profiles are static, and built using knowledge acquisition techniques (such as interviews and questionnaires) before they are applied. On the other hand, behavior-based user profiles use the user's behavior itself to build and improve the profile dynamically. The problem of the knowledge-based approach is that it is intrusive and time-consuming, while the behavior-based approach requires much behavior data to build an accurate enough user model.

So the usual trade-off_consists of a hybrid system, with a low number of questions applied when user join the system, combined with an unobtrusive monitoring the behavior of the user. The limitations of this approach are that initially the user profile is not very accurate since it is based on a few questions, and that often user's position alone are not enough to indicate interest in the closest neighborhood. First the system will ask some questions about preferences when the user registers in the system. Once this important information is collected, the answers form an initial model of the user transforming any kind of the preference expressions mentioned before to a numerical rating (relevance predicate).

Additionally, since the system monitors the physical movements of the user, it can induce from its decisions a more accurate user profile. The profiling algorithm performs correlation between time spent in particular stands and its interests. So, this is the algorithm for building the user profile.

Due to highly dynamic, uncertain and even noisy properties of the nowadays environments, the software running on these complex environments have to face problems such as: user mobility, service failure, resources or goal changes which may happen in any moment. That means that the initial states that prompt the decision making process in the first place may dynamically change while the decision making process is still going on. To cope with these problems, such systems should be able to acquire user's opinion and also self-adapt according to it. In order to achieve this objective, we develop a generic evaluation system (U2E system) already tested in several scenarios: u-commerce [32]; e-health [31]; in order to collect the user feedback for being used later in the adaptation process. See Fig. 21, where U2E system is been adapted to cope with the airport domain characteristics.

That is why there is a need of special kind of system that will combine ubiquity, context-awareness, intelligence, natural interaction, adaptation in an AmI environment. These make the decision making process of the system more complex than in a single environment.

Regarding to the issue of adaptation management [41] distinguishes three types of adaptation approaches: action based, goal based and utility function based. The first two are considered to be less advanced, so we concentrate our efforts on utility- based approach. Utility function-based approaches aim at delivering the best possible decision rather than just a feasible one as is the case of the goal-based adaptation. So, we have developed a user evaluation system capable of collect user feedback and also that performs self-adaptation.

Qualitative evaluation should consider the user as a central piece in the system, its satisfaction levels. On the other hand, quantitative methods should evaluate objectively the contribution of context to the application.

We based on two propositions an offline user's evaluation, Fig. 21 (once the user has finished using the system and that can be acces-sible via: www.giaa.inf.uc3m.es/u-shopping/myfeedback/). On the other hand we think it is also important to have an approach to Schmidt proposal but introducing the dynamically online user's evaluation allowing the user being capable of having dynamic access to specific context concepts in order to modify them and also of having the possibility to propose a modification on the reasoning algorithms and of course on the inferences rules executed and responsible of receiving one service or another. After that we can compare the inferences made about what a user needs, the evaluation must also take into account the correctness of our reasoning. If we draw incorrect conclusions the user must likely receive incorrect information.

Regarding to quantitative evaluation, users may be measured either quantitatively, typically by defining a utility function and mapping the "satisfaction state" to numeric value. If we know the basic task the user is doing and if we have some knowledge of the information needed for that task, we record the situational information the user goes about doing this particular task and take note about what actions the user does in a particular time. This is stored as part of the quality of services concept in our ontology as an historical file. And later analyze its behavior though the user's opinion that is crucial in these kinds of systems. The value of quality of services corresponds to the user's feedback and can have values like: (1) Correct, (2) Different order or (3) Incorrect for every attribute.

This opinion is given after he receives the ranking of the products the system recommends. We want to know whether the system satisfies, in some degree, a set of different users. As the process of information classification is generally a complex and personal task, and may differ among persons, we can measure the average system behavior over a population. We can then compare, after that, the inferences made by looking at the situational data at

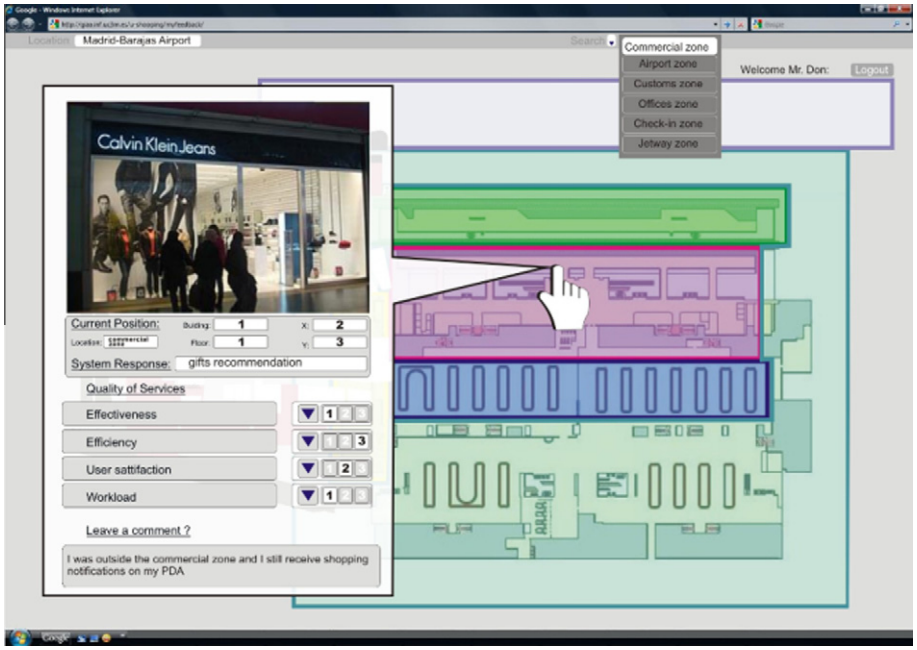

Fig. 21. Feedback collector system: U2E.

the actual user request to draw some conclusions. The preferences and restrictions introduced by users as input to the system, together with the system results and the user's feedbacks, constitute our *N*-cases set. Each case in the dataset will be composed by

- *User's Input*: a user ID and his graded preferences and restrictions.
- *System's Result*: the system returns a ranking of at most ten products and offers.
- *User's Feedback*: as explained previously, after analyzing the information of the recommended plans the user provides a feedback by evaluating the result as:

  (1) Correct, (2) Different order or (3) Incorrect.

In order to give a general measure of the system's results over the satisfactory cases we will evaluate how close is the system's ranking with regard to the user's own ranking. For this, we choose the Manhattan distance between the position of the first three products selected by the user and their position in the system ranking. This distance was adopted because it is appropriate for capturing positional differences.

If we assume the quality of user (which is the user's feedback) is $Q_i = (P_{i1}, P_{i2}, P_{i3})$ and the system ranking for this consult is $R_i = (R_1, R_2, \ldots, R_9)$. Then, if $P_{i1} = R_j$, $P_{i2} = R_k$, $P_{i3} = R_n$, the distance between the user's and the system rankings is defined by

$$\mathrm{Dist}(Q_i, R_i) = |1 - j| + |2 - k| + |3 - n|.$$

Regarding the quality of information, below the area each question a comment area could be utilized by a participant to further express the satisfaction or dissatisfaction experienced with any particular aspect or suggestion to improve the feedback. Both quantitative (e.g., effectiveness, efficiency, user satisfaction, and workload) and qualitative (e.g., user comments) data were obtained from the user.

So our system could then: (i) discriminate between contextual information; (ii) users can rewrite the concept "quality of user service" described in the ontology (which will be mapped to obtained a quantitative evaluation); (iii) users also can make personalized annotations and leave it as a note in the current location (qualitative); (iv) introduce a new concept in the ontology like the situational information concept described above.

Therefore in this section we have proposed an evaluation method which can be applied to our case of use, that is user-centered, and that allows a quantitative and qualitative evaluation of context-aware systems.

7. Conclusions

Interest in research into context-aware systems has increased lately, since they are becoming one of the main new challenges of computer science, and particularly of artificial intelligence. As applications are expected to provide a quality of service to users, they have to be able to adapt to context changes. The Appear Context Engine is a good context management solution as it provides for running a KBS on dynamically updated information about user location and statically defined information about the user profile.

In our context aware system, the different functionalities are distributed into specific modules to guarantee the most efficient and adapted service to the user. Two subsystems have been defined that carry out sensing and rendering functions, context-aware web services management, context information is captured, updated, managed and stored by means of the interaction between the different modules in the three layers. The Appear IQ Platform has been integrated for the web services management, dealing with the different processes that are required to implement this layer.

Our contribution in this paper is the definition of a KBS for a realistic scenario according to the CKADS methodology and conceptual modeling language (CML) in order to formally represent contextual information for the Appear platform. Linking a methodological approach (CKADS) to the development of the Knowledge base while this knowledge base is integrated in a commercial, extensively used platform (Appear) and proposing a user centered evaluation method (quantitative and qualitative) for context-aware systems is an innovative and relevant contribution to the state of the art.

First, we first defined the domain knowledge layer as a set of 22 concepts with their respective attributes and relationships. Additionally, we modeled the set of production rules as an inference structure that deals dynamically with the system input/output behavior. In this inference layer we described how the concepts and relationships are assigned to declarative specifications of dynamic roles. Finally, these inferences were fired in a sequential order defined by a control structure corresponding to one of the task templates suggested by CKADS. Particularly, we consider our problem to be a special case of an assessment task template, where an assessment problem consists of finding a decision category for a case based on domain-specific norms.

Next we showed step-by-step how Appear was effectively configured to implement the KBS designed according to CKADS principles. Furthermore, the KBS was validated running a simulated example of an airport use case. And finally, we propose an evaluation method of context-aware systems from a user-centered perspective, that can be applied to our case of use.

Therefore, we have design, implement and propose a way to evaluate a prototype of a realistic context-aware system that integrates a KBS (formally designed with CKADS methotodology) into a commercial, extensively-used location platform like Appear. This contribution is then a significant step towards the integration of soundly obtained KBS in context-aware systems to be applied in real-world environments.

We have also additional lines of research related to this work that includes a dialog system to our context-aware system to provide a natural and intelligent interaction of the user by means of an advanced speech-based interface [19]. And a further issue of interest for us is the inclusion of agent technology to distribute the contextual information instead of using Appear platform and finally compare results of both approachs [33].

Additionally since obtaining experimental results from a comparison of context-aware systems is not yet possible (current context-aware systems are too adhoc different to be implemented in a single testing domain), implementing a testbed for experimentation in context-aware systems is also a future work for us. Such kind of testbed would allow a fair comparison among heterogeneous implementations of context-aware systems given a number and distribution of sensors providing services and a automatically-generated number of moving users consuming services.

Likewise, a qualitative comparison is also hard to outline since many of the published does not include any performance, evaluation, implementation details, etc. While some context-aware systems are focused on quality of context data instead of the system it self [8,23,10], others provide toolkits to facilitate the implementation of context-aware systems [14,21], or running platforms such as [43,28], a few of them use explicit KBS such as [9,1], we are the first researchers who propose a methodological approach to build KBS for context aware systems that can be integrated into a commercial, real location platform such as Appear.

As future work it would be interesting to include an adaptation of our methodology to be used with CARE [4] specifications. Also include algorithms to predict user behavior as in [5] or [39].

References

[1] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, M. Pinkerton, Cyberguide: A Mobile Context-Aware Tour Guide, vol. 55, 1997, pp. 55–56.

[2] J. Angele, D. Fensel, R. Studer, Domain and task modelling in MIKE, in: A. Sutchliffe (Ed.), Domain Knowledge for Interactive System Design, Chapman & Hall, 1996.

[3] J. Angele, S. Decker, R. Perkuhn, R. Studer, Developing knowledge-based systems with MIKE, Journal of Automated Software Engineering 5 (4) (1998) 326–389.

[4] A. Agostini, C. Bettini, D. Riboni, Hybrid reasoning in the CARE middleware for context awareness, International Journal of Web Engineering and Technology 5 (1) (2009) 3–23.

[5] M. Armentano, A. Amandi, Personalized detection of user intentions, Knowledge Based Systems 24 (8) (2011) 1169–1180. ISSN:0950-7051.

[6] T. Berners-Lee, E. Miller, The semantic web lifts off, European Research Consortium for Informatics and Mathematics (ERCIM) News 55 (2002) 9–11.

[7] J.A. Breuker, W. Van de Velde (Eds.), The CommonKADS Library for Expertise Modelling, IOS Press, Amsterdam, The Netherlands, 1994.

[8] N. Brgulja, R. Kusber, K. David, M. Baumgarten, Measuring the probability of correctness of contextual information in context aware systems, in: 8th International Conference on Pervasive Intelligence and Computing (PICom 2009), Chengdu, China, 2009, pp. 246–253.

[9] R. Burke, K. Hammond, B. Young, Knowledge-based navigation of complex information spaces, in: Proceedings of The National Conference On Artificial Intelligence, vol. 22, 1996, pp. 462–468.

[10] Z. Callejas, R. López-Cózar, Influence of contextual information in emotion annotation for spoken dialogue systems, Speech Communication 50 (2008) 416–433.

[11] G. Chen, D. Kotz, A Survey of Context-Aware Mobile Computing Research, Technical Report, TR2000-381, Dartmouth College, Hanover, NH, USA, 2000.

[12] G. Chen, M. Li, D. Kotz, Design and implementation of a large-scale context fusion network, MobiQuitous (2004) 246–255.

[13] N. Dahlback, A. Jonsson, L. Ahrenberg, Wizard of oz studies-why and how, Knowledge-Based Systems 6 (4) (1993) 258–266.

[14] A. Dey, G. Abowd, D. Salber, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, Human–Computer Interaction 16 (2–4) (2001) 97–166.

[15] J. Domingue, E. Motta, S. Watt, The emerging VITAL workbench, in: Proceedings of the 7th European Knowledge Acquisition for Knowledge-Based Systems, EKAW'93, 1993, pp. 320–329.

[16] H. Eriksson, T. Shahar, S.W. Tu, A.R. Puerta, M.A. Musen, Task modelling with reusable problem solving methods, Artificial Intelligence 79 (2) (1995) 293–326.

[17] A. Gokhale, D. Schmidt, Techniques for optimizing CORBA middleware for distributed embedded systems, in: Proceedings of the IEEE INFOCOM'99, New York, NY, 1999.

[18] A. Gomez-Perez, V.R. Benjamins, Overview of knowledge sharing and reuse components: ontologies and problem-solving methods, in: Paper presented at IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, 1999.

[19] D. Griol, N. Sanchez-Pi, J. Carbó, J.M. Molina, Context-aware approach for orally accessible web services, WI-IAT, in: IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, Italy, vol. 3, 2009, pp.171–174.

[20] T.R. Gruber, Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Report KSL-93-04, Stanford University, Stanford, CA, 1993.

[21] J.I. Hong, The context fabric: an infrastructure for context-aware computing (doctoral consortium), in: Extended Abstracts of ACM Conference on Human Factors in Computing Systems (CHI 2002), ACM Press, Minneapolis, MN, 2002, pp. 554–555.

[22] M. Hussein, J. Han, A. Colman, An Architecture-Based Approach to Context-Aware Adaptive Software Systems, Technical Report, #C3-516_04, Swinburne University of Technology, 2011.

[23] H. Kang, E. Suh, K. Yoo, Packet-based context aware system to determine information system user's context, Expert Systems with Applications 35 (2008) 286–300.

[24] P. Markopoulos, B. de Ruyter, S. Privender, A. van Breemen, Case study: bringing social intelligence into home dialogue systems, Interactions 12 (4) (2005) 37–44.

[25] F.N. Noy, M. Sintek, S. Decker, M. Crubezy, R.W. Fergerson, M.A. Musen, Creating semantics web contents with Protégé 2000, IEEE Intelligent Systems 16 (2) (2001) 60–71.

[26] Object Management Group (OMG), CORBA 2.5 Specification, 2000.

[27] A.R. Puerta, J.W. Egar, S.W. Tu, M.A. Musen, A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools, Knowledge Acquisition 4 (2) (1992) 171–196.

[28] M. Roman, C. Hess, R. Cerqueira, R.H. Campbell, K. Nahrstedt, Gaia: A middleware infrastructure to enable active spaces, IEEE Pervasive Computing 1 (2002) 74–83.

[29] N. Sánchez-Pi, V. Fuentes, J. Carbó, J.M. Molina, Knowledge-based system to define context in commercial applications, in: 8th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD), Qingdao, 2007.

[30] N. Sánchez-Pi, J. Carbó, J.M. Molina, Building a knowledge based system for an airport case of use, in: International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008), Advances in Soft Computing, vol. 50, 2009, pp. 739–747.

[31] N. Sánchez-Pi, J.M. Molina, Adaptation of an evaluation system for e-health environments, Knowledge-Based and Intelligent Information and Engineering Systems, Lecture Notes in Computer Science, vol. 6279, Springer, 2010. pp. 357–364.

[32] N. Sánchez-Pi, J. Molina, A multi-agent approach for the provisioning of e-services in u-commerce environment, Internet Research 20 (3) (2010) 276–295.

[33] N. Sánchez-Pi, E. Mangina, J. Carbo, J.M. Molina, Multi-agent system (MAS) applications in ambient intelligence (Aml) environments, in: Trends in Practical Applications of Agents and Multiagent Systems, Advances in Intelligent and Soft Computing, Springer-Verlag, Salamanca, Spain, 2010, pp. 493–500.

[34] B.N. Schilit, A System Architecture for Context-Aware Mobile Computing, Ph.D. Thesis, Columbia University, Department of Computer Science, 1995.
[35] A. Schmidt, Ubiquitous Computing-Computing in Context, Ph.D. Thesis, 2002.
[36] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, W. Van de Velde, Advanced Interaction in Context in Handheld and Ubiquitous Computing, Springer-Verlag, 1999. pp. 89–101.
[37] G. Schreiber, H. Akkermans, A. Anjewierden, R. Hoog, N. Shadbolt, W. Van de Velde, W. Wielinga, Knowledge Engineering and Management, The CommonKADS Methodology, The MIT Press, 1999.
[38] N. Shadbolt, E. Motta, A. Rouge, Constructing knowledge-based systems, IEEE Software 10 (6) (1993) 34–38.
[39] R.J. Tait, T.J. Allen, N. Sherkat, M.D. Bellett-Travers, An electronic tree inventory for arboriculture management, Knowledge-Based Systems 22 (7) (2009) 552–556.
[40] S.W. Tu, H. Eriksson, J. Gennari, Y. Shahar, M.A. Musen, Ontology-based configuration of problem-solving methods and generation of knowledge acquisition tools: application of PROTÉGÉ-II to protocol-based decision support, Artificial Intelligence in Medicine 7 (1995) 257–289.
[41] W.E. Walsh, G. Tesauro, J.O. Kephart, R. Das, Utility functions in autonomic systems, in: International Conference on Autonomic Computing, 2004, pp. 70–77.
[42] B. Wielinga, J.M. Akkermans, H.A. Hassan, O. Olsson, K. Orsvärn, A. Schreiber, P. Terpstra, V. Van de Velde, S. Wells, Expertise Model Definition Document, Technical Report, ESPIRIT Project P5248/KADS-II/M2/UvA/026/5.0, University of Amsterdam and Free University of Brussels and Netherlands Energy Research Centre, ECN, 1994.
[43] S.S. Yau, F. Karim, A context sensitive middleware for dynamic integration of mobile devices with network infrastructures, Journal of Parallel and Distributed Computing 64 (2004) 301–317.