Expert Systems with Applications 39 (2012) 2907–2918

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

iConAwa – An intelligent context-aware system

Özgün Yılmaz *, Rıza Cenk Erdur

Ege University, Department of Computer Engineering, Bornova, 35100 Izmir, Turkey

ARTICLE INFO

Keywords:
Context-awareness
Software agent
Context ontology
Context reasoning
Rule-based reasoning

ABSTRACT

Context-awareness becomes an increasingly important concept in the development of mobile and ubiquitous systems. Applications and services, which run in these kinds of highly dynamic environments, should be aware of and adapt to their contexts. Context-aware applications improve and enrich people's interactions with devices, computers and other people.

In this paper, design and development of iConAwa, which is an intelligent context-aware multi-agent system proactively providing mobile users with context-aware information and services, is described. In iConAwa, mobile users can get information and services about nearby resources (attraction points) according to their context and also communicate with each other by exchanging messages. Context and point of interest ontologies are developed in OWL. Context and points of interest are modelled in a flexible and extensible way by the developed ontology models. Knowledge sharing and knowledge reuse are also provided by using these ontology models. iConAwa makes use of rule-based context reasoning which provides derivation of high level implicit context from low level explicit context. With this approach context reasoning is decoupled from the source code of the system. JADE agent development framework is used to develop the agents and Jena semantic web framework is used to manipulate ontologies and for rule based reasoning.

1. Introduction

In recent years, with major advances in technology, mobile devices such as PDAs and smart mobile phones and wireless networks have entered our daily lives. Computing becomes increasingly mobile and ubiquitous nowadays. With these technological improvements although computers are shrinking in size, their computing power are increasing consistently. PDAs, moreover smart phones have sufficient computing power to run some desktop applications. As for wireless networks, wireless networks are becoming more and more widespread and they are covering many locations nowadays. All these improvements enable access to resources, data and information anytime, anywhere and introduce a new research field called ubiquitous computing which is a highly dynamic and complex environment (Chen & Kotz, 2000; Wang, Zhang, Gu, & Pung, 2004).

Traditional distributed systems which assume running in a fixed environment are not suitable for these kinds of highly mobile scenarios (Chen & Kotz, 2000) because factors like location, social environment, network bandwidth, connectivity status and communication costs of the user or the mobile device are continuously changing (Schilit, Adams, & Want, 1994). Context-awareness is an important step for applications running in these kinds of highly dynamic environments. Context-aware applications can adapt to changing situations easily and by distinguishing relevant information from irrelevant information they provide better functionality (Belotti, Decurtins, Grossniklaus, Norrie, & Palinginis, 2005).

Context-aware systems are suitable for agent-based development. Basically, agents are systems which behave autonomously to reach their goals in an environment (Wooldridge, 2002). Running environments of context-aware applications and agents are similar. Both of these systems focus on ubiquitous computing and run in open world and dynamic environments. Context-aware applications require automatic decision making and taking an action. Agents being autonomous, reactive and proactive (goal-oriented and initiative holder) makes agents suitable for use in context-aware systems.

In this paper, design and development of iConAwa which is an intelligent context-aware multi-agent system proactively providing mobile users with context-aware information and services, is described. In iConAwa, mobile users can get information and services about nearby resources (attraction points) according to their context and also communicate by exchanging messages with each other.

The significance of iConAwa is that the developed system is a context-aware multi-agent system and makes use of reasoning. The context is modelled with the context ontology using OWL (Web Ontology Language). Rule-based context reasoning is performed over the context ontology. The system has expert system

* Corresponding author. Tel.: +90 232 3111010; fax: +90 232 3399405.
E-mail addresses: ozgun.yilmaz@ege.edu.tr (Ö. Yılmaz), cenk.erdur@ege.edu.tr (R.C. Erdur).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.08.152

characteristics, includes intelligent agents showing proactive and autonomous behavior and is an intelligent system in all of these aspects. iConAwa system presented in this paper is original and different from all other related work by combining these features in one system.

The reasons for using an ontology-based context model are to provide knowledge sharing, logical reasoning and knowledge reuse (Gu, Wang, Pung, & Zhang, 2004; Wang et al., 2004). Also attraction points or points of interest (POI) are modelled with the POI ontology.

In iConAwa, mobile user logs into the system using his/her mobile device. Each user has a client agent running in the mobile device. Client agent sends user's current location information consisting of latitude and longitude values to the context agent when the user's context changes. Mobile user's personal information such as personal profile, preferences, etc. is stored in the context ontology which is stored on the server side. Context agent running on the server side combines these contextual data and obtains user's context and then sends a list of points of interest and nearby users which are suitable for the user's context to the client agent to be shown to the user. This process includes decision making. The user can send messages to nearby users. Point of interest list is sorted according to the match degrees of the points of interest. A point of interest can be an activity place, museum, historical or cultural place, restaurant, movie theater, shopping store, etc. Points of interest are stored in the POI ontology. POIs are also shown to the user on a map. The user can scroll through the map and change map type and zoom level. Google Static Map API is used for providing maps.

The users can also get services via service agents each specific to a certain point of interest which can run in remote computers.

The agents are developed using JADE (Java Agent Development Framework) (JADE, 2010) and coded in Java programming language. Jena (2010) a semantic web framework is used to manipulate ontologies and make rule-based reasoning. Context and POI ontologies are developed using Protégé (Protege, 2010) ontology editor.

The subsequent sections of this paper are as follows: In Section 2, an overview of context-awareness is given. In Section 3, system architecture of iConAwa, context agent, client agent, service agent, context and points of interest ontologies are presented and context reasoning is discussed. In Section 4, a case study is presented. In Section 5, related research is surveyed. Research related to context-awareness, context-aware applications and context-aware agent-based applications are reviewed and the applications are compared. iConAwa system is evaluated according to related work. In Section 6, conclusion is discussed.

2. Context-awareness

Many definitions of context have been made in the field of mobile computing by different researchers. According to Schilit and Theimer (1994) who first mentioned the term, context consists of location, nearby objects and people and also changes to those objects and these are the most important aspects of context (Dey, 2001). However some researchers argue that this definition is too specific. Context encompasses all situations related to the user and the application. Because use of context can change from application to application, it is not appropriate to list what context consists of Dey (2001).

Schilit divides context into three categories (Chen & Kotz, 2000; Schilit et al., 1994):

- *Computing context:* computing context might consist of network connectivity, communication costs, network bandwidth, nearby resources and specifications of the mobile computer such as CPU speed, memory, display resolution, etc.
- *User context:* user context might consist of user profile, location, nearby people and current social situation.
- *Physical context:* physical context might consist of lighting level, noise level, temperature, etc.

In addition to these, time is also an important context. Time context might consist of time of a day, week, month, and season of the year. If context data are recorded across time, then context history is obtained. Context history is crucial for some applications (Chen & Kotz, 2000).

Schmidt et al. (1999), who try to define context formally, define context as "knowledge about the user's and IT device's state, including surroundings, situation, and to a less extent, location". According to Dey (2001), context is "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". Dey (2001) states that this definition makes it easier to find out which information constitutes the context for a specific application.

Primary contextual information such as location, identity, time, and activity act as indices into other contextual information. For example, from a user's identifier, information such as that user's phone number, address, birth date, friends, and relationships with other people, etc. can be obtained. With an entity's location, nearby objects, people and activities can be determined (Dey & Abowd, 1999). By combining contextual information, current situation can be understood more precisely. For example, user's current location and current time can be combined with the user's calendar to reveal user's current social situation such as having a meeting, sitting in the class, waiting in the airport, etc. (Chen & Kotz, 2000).

Context-aware computing was first discussed by Schilit and Theimer (1994) as software that "adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time" (Dey, 2001). Thus context-aware software is software that adapts to its context and changes occurring in this context.

Schilit et al. (1994) divides context-aware applications into four categories:

- *Proximate selection:* proximate selection is a user interface technique where nearby objects are emphasized or made easier to choose.
- *Automatic contextual reconfiguration:* it is a process of adding new components, removing existing components or changing the relationships between components as a result of context changes.
- *Contextual information and commands:* it is a type of application which manually executes commands for the user with respect to current context.
- *Context-triggered actions:* it is a type of application which automatically executes commands for the user with respect to current context. In this kind of applications, how a context-aware system will adapt to current context is determined by defining simple IF–THEN rules.

2.1. Sensing the context

In this section, how context can be obtained is discussed. To use context in an application, first context should be sensed by sensors. Context which is directly obtained from the data coming from sensors is called low level context. Sensors can be implemented as software or hardware sensors. For example, temperature, lighting level, location, etc. can be sensed by hardware sensors. On the other hand, personal information, status of a service, etc. can be

sensed by software sensors (Chen & Kotz, 2000; Gonçalves, Costa, & Botelho, 2008).

Some contextual information is static and some are dynamic, they continuously change over time. If the context information does not vary much over time then it is called static context. Static context can be acquired directly from the user or a service and then stored in a central repository. On the other hand, dynamic context is usually acquired from the sensors instantaneously. User's name, nationality, birth date, etc. are examples of static context, whereas location, current time, current temperature, etc. are examples of dynamic context (Gonçalves et al., 2008; Henricksen, Indulska, & Rakotonirainy, 2002). In the following parts of this subsection, how to sense main contextual information, such as location, direction, network bandwidth, time, nearby people and objects, is discussed.

Because location is an important contextual information which changes when the user moves, a reliable location-tracking system is crucial for context-aware applications. The Global Positioning System (GPS) is a very convenient positioning system for outdoor use. GPS system includes more than 20 satellites orbiting around the earth. A GPS receiver can calculate its position utilizing these satellites with an accuracy of 10–20 m (Chen & Kotz, 2000). GPS can be utilized anywhere in the world without any mandatory registration for free. By attaching a GPS receiver to the user's mobile device, location of the user can be tracked. The main advantages of the GPS system is that; it covers everywhere around the globe outdoors, it is accurate and free. GPS receiver can also determine speed and direction of a moving person.

On the other hand, GPS system does not work indoors because the GPS signal cannot penetrate most buildings. Implementing a positioning sensor which can operate indoors and also which is inexpensive, scalable, robust and small in size is a challenging task. Most researchers build their own location tracking systems (Chen & Kotz, 2000). Olivetti Active Badge system (Want, Hopper, Falcao, & Gibbons, 1992), Xerox ParcTab (Want et al., 1996) and the Cyberguide (Abowd et al., 1997) projects make use of infrared technology to track location. The Personal Shopping Assistant (Asthana, Crauatts, & Krzyzanowski, 1994), the Pinger (Hull, Neaves, & Bedford-Roberts, 1997) and the Pinpoint (Werb & Lanzl, 1998) systems use radio waves. The Cricket (Priyantha, Chakraborty, & Balakrishnan, 2000) location-support system use both radio and ultrasonic waves to track location. The indoor tracking schemes mentioned above all track location automatically. On the contrary, if the system is to be used in a building consisting of rooms, then manual location tracking based on the rooms can be implemented, provided that users slide their badge or press their finger on the fingerprint reader before entering and leaving a room. In this situation users should be willing to cooperate, otherwise it can lead to errors.

To summarize, there are two general approaches to tracking location of the mobile device. One approach is that the central system calculates the location of the mobile device by tracking the beacon signal broadcasted from the mobile device. The other one is that the mobile device calculates its own location (e.g. GPS) by listening to signal beacons broadcasted from the base stations. In the latter case, mobile device knows its own location, thus user privacy is maintained and the user can send its location information only to selected modules (Chen & Kotz, 2000). In addition to these, if the location history of a user is taken into account, then the direction the user is going can be determined.

Network context might also be an important context for mobile context-aware applications because connectivity status, bandwidth and communication costs of a wireless network can vary frequently. For the applications to be able to adapt to network bandwidth changes, system support is necessary. Developed as a user-level module, the Odyssey system (Noble et al., 1997) provides API calls for the applications to be notified if the network bandwidth changes.

Another low level contextual information is time. Time context can be obtained from the built-in clock of the mobile device. Nearby people and objects can be determined if the system is aware of the locations of people and objects (Chen & Kotz, 2000).

2.2. Context reasoning

In addition to low level contextual information, high level contextual information might be needed too. However it is really hard to sense high level context (Chen & Kotz, 2000). A solution to this problem is to combine low level context data and deduce a conclusion from this. If the context model used in the application is formal, then logical reasoning can be made over the contextual information. By context reasoning, high level implicit context can be derived from low level explicit context. Also consistency of the context can be checked and inconsistencies can be eliminated. For example, if a person is located in bedroom and the bedroom is located in the house, this person is located in the house can be deduced. Similarly, if a person is located in living room and TV is located in the living room and the TV is on then the person is watching TV can be deduced (Gu et al., 2004; Wang et al., 2004).

2.3. Modelling the context

Existing context models can be classified into three categories (Gu et al., 2004):

- *Application-oriented approach:* most context-aware systems model the context this way. These models are specific to applications and they lack formality and expressiveness. In these kinds of models, context is represented by key-value pairs, XML (Extensible Markup Language) or another XML based markup language (Chen & Kotz, 2000).
- *Model-oriented approach:* in this approach, context is usually modelled using conceptual modelling approaches and these models are formal. Context models based on ER (Entity-Relationship) were proposed by several projects (Harter, Hopper, Steggles, Ward, & Webster, 2002; Wu, Siegel, & Ablay, 2002). ER based context models can be easily developed with relational databases. In another project, Henricksen et al. (2002) model context with ER and UML (Unified Modelling Language). This model was further rearranged with extended ORM (Object Role Modelling) (Henricksen, Indulska, & Rakotonirainy, 2003).
- *Ontology-oriented approach:* finally, context can be formally modelled with ontologies, comprising of first-order predicates. Context model in GAIA system is encoded in DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer) (Ranganathan & Campbell, 2003). In other projects (Chen & Finin, 2003; Wang et al., 2004) context model is encoded in OWL. In iConAwa, context is also modelled using OWL, because DAML+OIL is merging into OWL and DAML+OIL is becoming obsolete.

From the above categories, application-oriented approach is not formal and does not support knowledge sharing across different systems. Because it is not formal, it is application-specific and lacks expressiveness. Model-oriented approach is formal but it does not support knowledge sharing and context reasoning. Ontology-oriented approach is formal and supports knowledge sharing and knowledge reuse. In addition to these, it supports context reasoning based on Semantic Web technologies (Gu et al., 2004). Rule-based reasoning or description logic reasoning can be used to employ context reasoning over ontologies. In iConAwa ontology-oriented approach is used.

3. iConAwa

3.1. Motivation

In this section, a motivating scenario for describing the iConAwa system is presented. According to this scenario, a person walking in the street starts the client application through his/her mobile computer which has an Internet connection and a GPS receiver and then starts a new session by specifying user name and password. Then user agent sends authentication and location information obtained from the GPS receiver each time the user context changes, to the context agent through the Internet. Then context agent sends lists of nearby points of interest and people who permit sharing of their location information to the user agent, considering user's context which consists of current location, time and user's personal profile. Client application presents this information to the user in a text area and also by pinpointing each POI on a map. The user can scroll through the map, change zoom level and map type. A point of interest can be a restaurant, movie theater, activity place, museum, historical or cultural place, etc. Coordinates, description of the point of interest, related keywords to the user's profile and preferences and its match degree to the user's context are also presented to the user. A point of interest with a higher matching degree is emphasized to the user. In this scheme the user is also able to interact with points of interest. For instance when the user chooses a movie theater; movies currently being screened, movies' match degree to the user's preferences, movie session information and related prices are shown to the user and then the user can take a ticket to the desired movie. Navigation information including current geographic location, speed and moving direction can also be presented to the user. If there are other users nearby and if these users allow sharing of their location information, then these nearby users' account names are shown to the user. If these nearby users allow sharing their personal information and there are common user interests between the user and these people then these nearby users which have common interests are emphasized and shown to the user.

3.2. System architecture

In this section, system architecture of iConAwa is presented. This architecture is capable of fulfilling the scenario described in Section 3.1. The system is a context-aware multi-agent system. System architecture is shown in Fig. 1. Server side consists of a server computer, context agent and DF (Directory Facilitator) agent of the JADE platform running on the server, context and points of interest ontologies. DF agent provides yellow pages directory service to agents as specified by FIPA (Foundation For Intelligent Physical Agents) Agent Management Specification (FIPA, 2004). Server computer is connected to the Internet.

Client side consists of mobile computers or smart phones each having a GPS receiver, client agent running on the mobile device and a user interface related to the client agent. Also there can be service agents each specific to a certain point of interest running in a remote computer. These service agents provide services related to a point of interest. Client agent obtains user's current location in latitude and longitude values from the GPS receiver of the mobile device. Because context agent communicates with client agents through the Internet, server computer and client's mobile device must have an Internet connection. Context agent registers to the DF agent of the JADE platform, specifying the service it provides. Client agents find out context agent's unique name (ID) by querying the DF agent with the service description. Client agent, running in the mobile device, sends context information automatically as the user's context changes via ACL messages. Context agent composes user's context by combining low-level dynamic contextual information such as location, time and user's personal profile which is stored in the context ontology. Then context agent compares and matches this context with the points of interest and sends information relevant to the user's context back to the client agent again via ACL messages. Client agent presents this information to the user through the user interface. This message sent from the context agent includes a list of points of interest, their descriptions and their match degree to the user's context. User can get services over the Internet using his/her mobile device. These points are also shown to the user on a map. The map is updated as the user's context changes. Google Static Maps API is used in the map component of the iConAwa system. Also user's high level social context is derived from low-level context by context reasoning. Social context is composed of other nearby users. If there are any nearby users who allow sharing of their location information, then the social context information is also sent to the user. Then the user can send messages to nearby users.

Java programming Language and JADE agent development framework are used in the development of iConAwa. A second version of client agent is developed for the JADE-LEAP environment which is obtained by addition of LEAP plug-in to the JADE environment. Hence this version of client agent is capable of running on devices with limited resources such as smart phones or PDAs.

In following sections, context and point of interest ontologies, context agent, client agent and service agent are described respectively.

3.3. Ontologies

In this section, context and point of interest ontologies are described. While context is modelled with the context ontology, points of interest (attraction points) are modelled with the POI ontology. These ontologies are used by the context agent running on the server side. Context and points of interest are modelled with the use of ontologies and these models are flexible and extensible. These issues are further discussed in the following subsections.

Ontologies are encoded in OWL (Web Ontology Language). OWL is chosen because RDFS (RDF Schema) is more restrictive and inferior in terms of expressiveness. OWL is chosen instead of DAML+OIL because DAML+OIL is becoming obsolete.

OWL enables the definition of domain ontologies and sharing of domain vocabularies. Ontology refers to the subject of existence and is the shared conceptualization of concepts, properties and relationships between the concepts in a specific domain (Wang et al., 2004).

Protégé ontology editor is used for developing the ontologies. Protégé is an open source and extendable ontology editor written in Java. It is developed at Stanford University (Protege, 2010). Ontologies can be developed with ease using Protégé.

3.3.1. The context ontology

In iConAwa system, an ontology-based context model is used which is encoded in OWL. The advantages of modelling the context using an ontology are as follows (Wang et al., 2004):

- *Knowledge sharing:* using an ontology based context model provides computational entities running in pervasive computing environments, such as agents and services to have a common set of concepts about context while interacting with each other. As a result, semantic interoperability is achieved for exchanging and sharing contextual information between different systems.
- *Logical reasoning:* another advantage is the built-in reasoning capability over the context ontology. High level implicit contextual information can be derived from low level raw context data by performing reasoning over the context ontology. Also

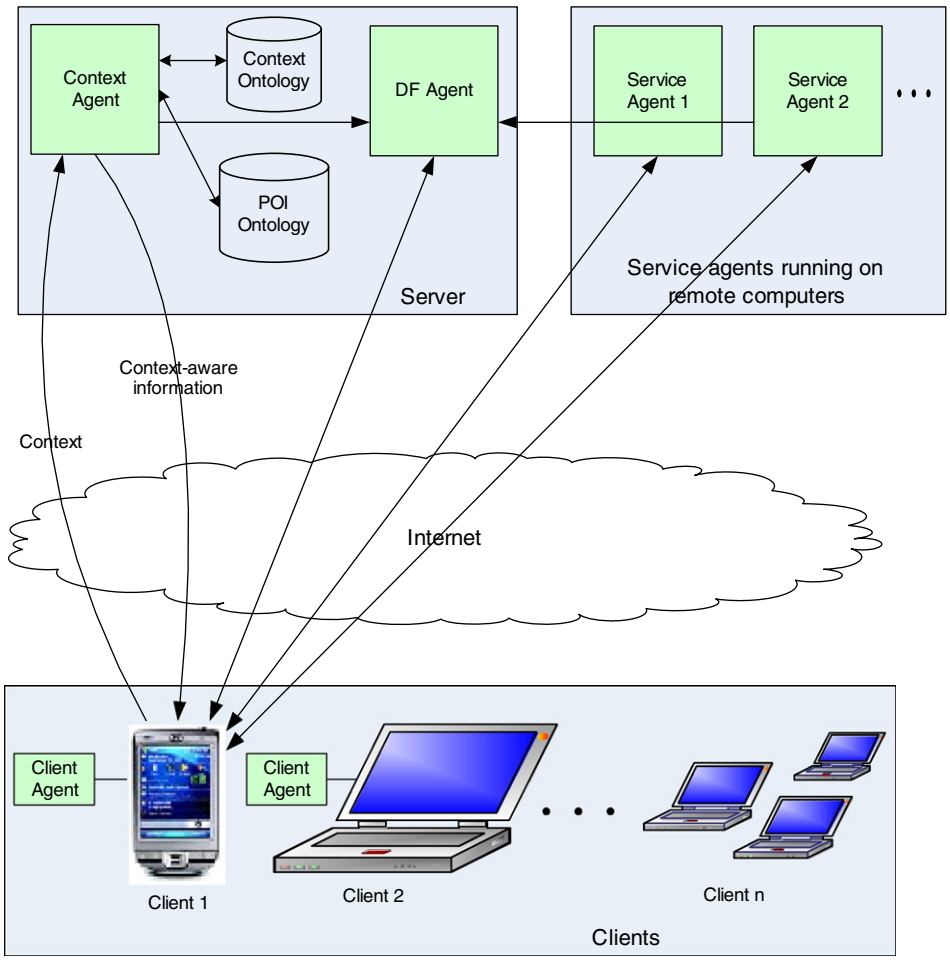

Fig. 1. System architecture of iConAwa.

inconsistent context information caused by erroneous sensor inputs can be realized by reasoning. Rule-based reasoning or description logic reasoning can be used.

- *Knowledge reuse:* by reusing well-defined ontologies from different domains, large scale context ontologies can be created without starting from scratch.

In the context ontology, context is composed of location, time, mobile device properties, network, user's personal profile and user preferences. The context ontology is modelled with the use of classes, object properties, data type properties and restrictions. Context, location, time, mobile device, network, personal profile and personal preferences are modelled with ontology classes for flexibility and extensibility reasons as shown in Fig. 2. Context class is related to these classes with object properties.

In the context ontology, location context is composed of latitude and longitude classes, again for flexibility and extensibility purposes. Latitude and longitude values are stored as strings in latitude and longitude classes. Because latitude and longitude values are stored as strings, decimal numbers or degree-minute-second format or any other format can be used. Time context is composed of date and time values. Mobile device context is composed of brand, model, CPU speed, memory and screen resolution of the mobile device. Network context is composed of network type, communication costs and network quality parameters including network bandwidth which is an important factor for mobile applications. User context is composed of personal profile and preferences. Personal profile class stores user ID, password, name, address, occupation and other personal information. Every user must have one and only one user profile and this is mandated by restrictions. Preferences are organized into a class hierarchy. Thus flexibility is achieved. A user can have many preferences (food, film, etc.). All of the mentioned classes can be extended furthermore to meet current user requirements.

User specific context is formed by creating individuals from the above mentioned ontology classes. Mobile device properties and user's personal profile constitute static context. Static context can be stored in the ontology file. On the other hand, location, time and network parameters may change continuously in an application where the users are mobile and this information constitute dynamic context. It is intended that dynamic context should be handled on the fly programmatically.

3.3.2. The point of interest (POI) ontology

In iConAwa, points of interest are modelled with the POI ontology. In the POI ontology, a point of interest is composed of name, location, (opening and closing) time, type, keywords describing the POI and services provided by the POI. Again these concepts are modelled with ontology classes for flexibility and extensibility reasons as shown in Fig. 3.

In the POI ontology location is modelled similar to the context ontology. POI type describes the point of interest type. It can have string values like 'museum', 'restaurant', 'movie theater', etc. Keywords describe the services given in that POI. Each POI may have many keywords associated with it. Services class stores the detailed description of the services provided given in that POI.

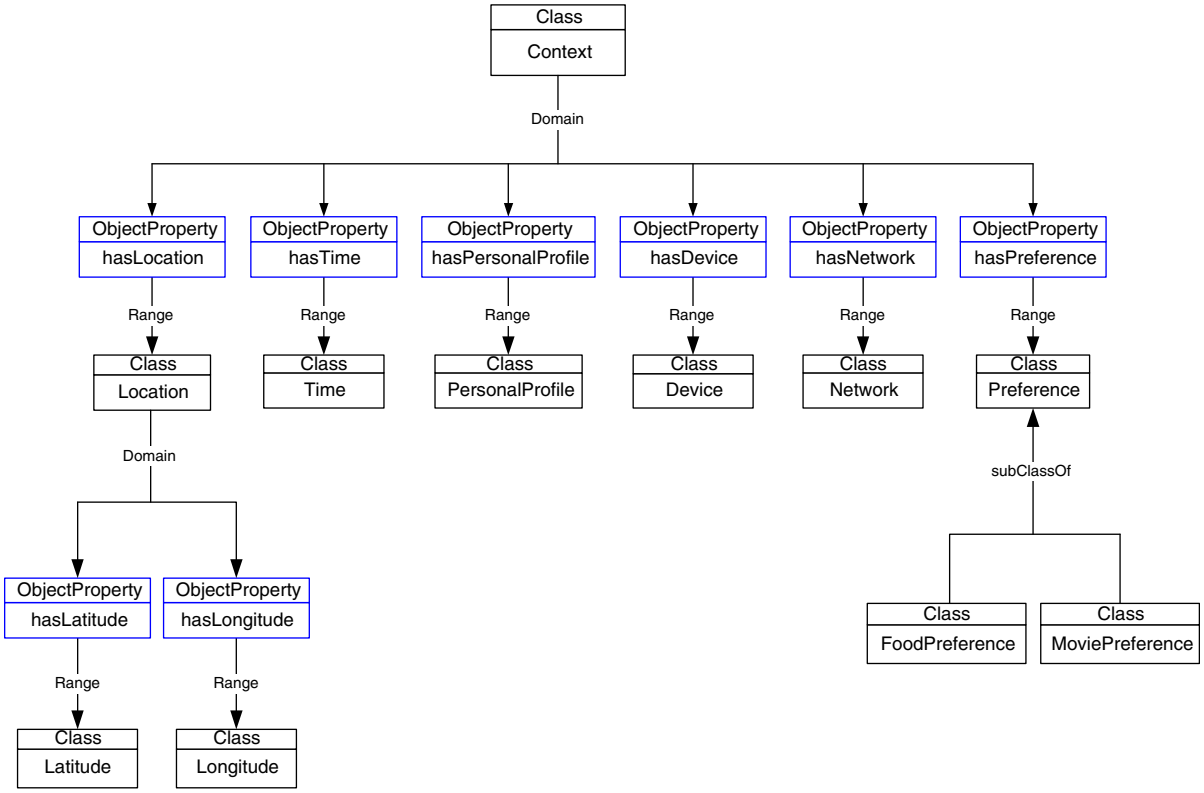

Fig. 2. General view of the context ontology.

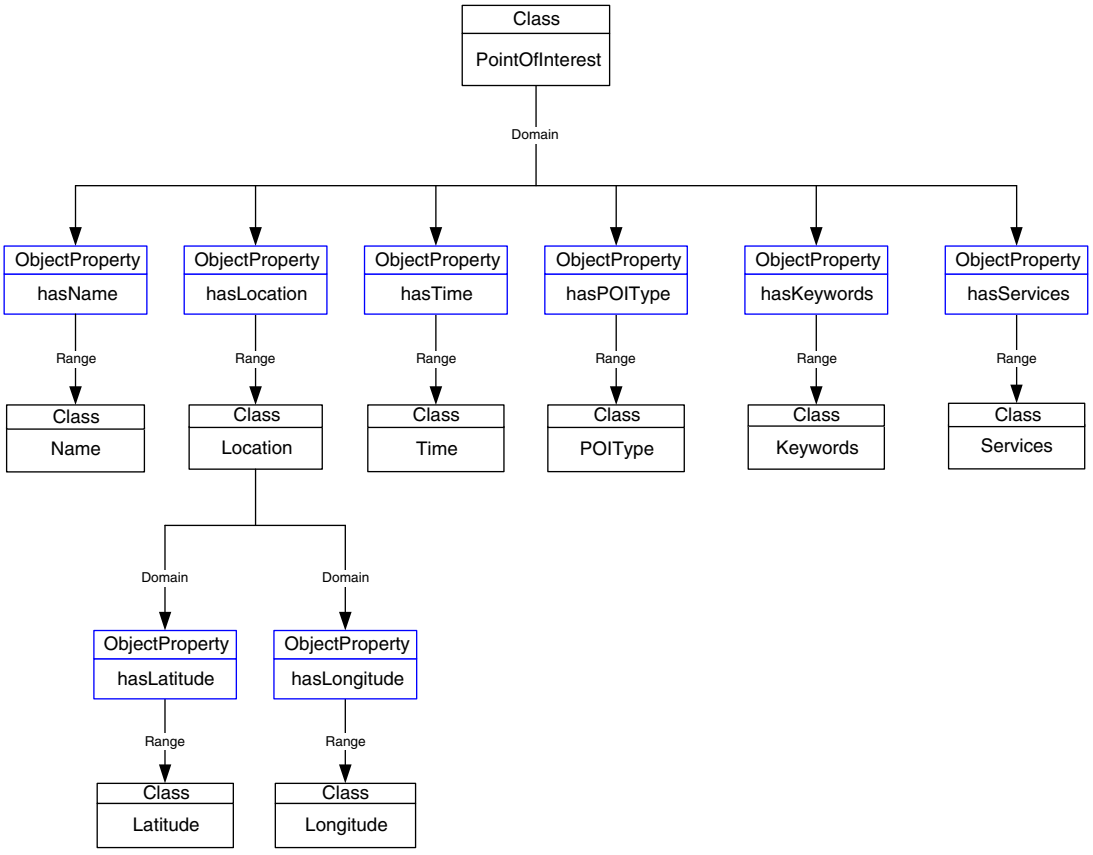

Fig. 3. General view of the point of interest (POI) ontology.

A point of interest individual must be created from the POI class for each point of interest. These individuals are to be stored in the ontology file.

3.4. The context agent

In this section, the context agent of the iConAwa system which provides mobile users with context-aware information is described. Context agent uses Jena framework to manipulate ontologies and make rule-based reasoning. Operation of context agent includes decision making. Context agent decides which context-aware information should be sent to the client agent. For example, if the user has moved and the points of interest that should be sent to the client agent are same as the previous ones then context agent does not send all of the information about point of interests, but only sends new distance values to the client agent. By this strategy, only necessary information is sent to the client agent, reducing the network communication overhead that would occur otherwise.

When the application starts, context agent registers itself to the DF agent stating the given service description. Client agents get the context agent's ID by querying the DF agent.

Context agent gets request messages from the client agents. Request messages include user ID, password and location information. If the user ID or the location is missing or the user ID or the password is wrong, then a refuse message stating the error occurred is sent to the client. If the sent information is valid then latitude and longitude values are extracted from the request message. Then context individual is directly accessed in the context ontology with the user ID and then the coordinate values for that individual is set to carry out reasoning. For each point of interest in the POI ontology, the distance to the client is calculated using the Haversine formula. If the distance is less than a predefined threshold and the point of interest is open at that time, then match degree of the point of interest is calculated with respect to keywords and user's interests and preferences. Match degree is equal to the number of matching keywords. If distance and time conditions are met, even if the match degree is 0, point of interest is added to the list of points of interest that will be sent to the user. Then the list is sorted according to match degree values in descending order. This list includes information about points of interest such as its name, match degree, type, location, current distance to the user, opening and closing times, given services, keywords and matched keywords. This list is sent to the client agent by an inform message. Also other nearby users and nearby users with similar interests are discovered by making rule-based reasoning over the context ontology. This information is also sent by another inform message to the client agent if there are any nearby users. Then the user can send messages to these nearby users.

If there are same parts in a message which were sent previously, then only the different parts are sent to the client to optimize network usage.

3.5. The client agent

In this section, the client agent is described. Each client agent is affiliated with a user interface and vice versa. Mobile user can log in and make requests to the context agent by using the user interface. When the user's location changes, client agent makes a new request to the context agent automatically without the need of the user intervention. Every time the client agent makes a request by specifying user ID, password and location, a request message is forwarded to the context agent. The information received from the context agent is shown to the user from the client user interface. JADE version of the client user interface is shown in Fig. 4.

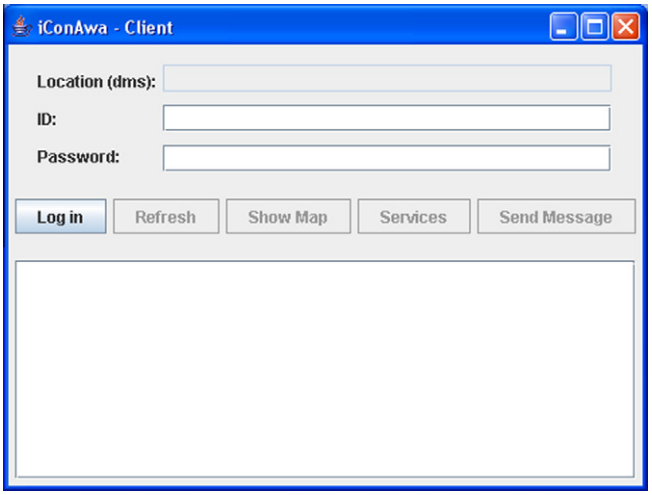

Fig. 4. JADE version of the client user interface.

3.6. The service agent

In iConAwa, mobile users can get services from service agents specific to each point of interest. The client agent can search for a service agent specific to a point of interest by querying the DF agent. After the client agent gets a certain service agent's ID, it can perform operations supported by the service agent. These operations might include making reservations, taking a ticket, etc. The user can get services about a point of interest by clicking the "Services" button in the client user interface and then searching the POI name in the services window frame. The communication between the client agent and service agents is achieved by exchanging ACL (Agent Communication Language) messages.

3.7. Interactions of context agent

In this section, interactions between context agent and a client agent are described in a sequence diagram. These interactions are shown in Fig. 5. The context agent and the client agents interact with each other by exchanging messages. Client agent sends request messages to the context agent and the context agent replies to these request messages by sending either inform or refuse messages.

Interactions of the context agent and a client agent:

1. The context agent registers itself to the DF agent by specifying "Contextaware" as the service description.
2. The client agent queries the DF agent for the agents which specified "Contextaware" as their service description.
3. DF agent replies client agent's query by sending agent identifier of the context agent.
4. Client agent sends request message to the context agent. This message includes user ID, password and current location.
5. If the request message contains missing or wrong information then a refuse message is sent to the client agent stating the error occurred.
6. If the request is valid than an inform message containing context-aware information which is arranged according to the user's context is sent to the client agent.
7. If there are other nearby users which allow sharing of their personal information, then user identifiers of these nearby users and nearby users with similar interests are sent to the client agent.

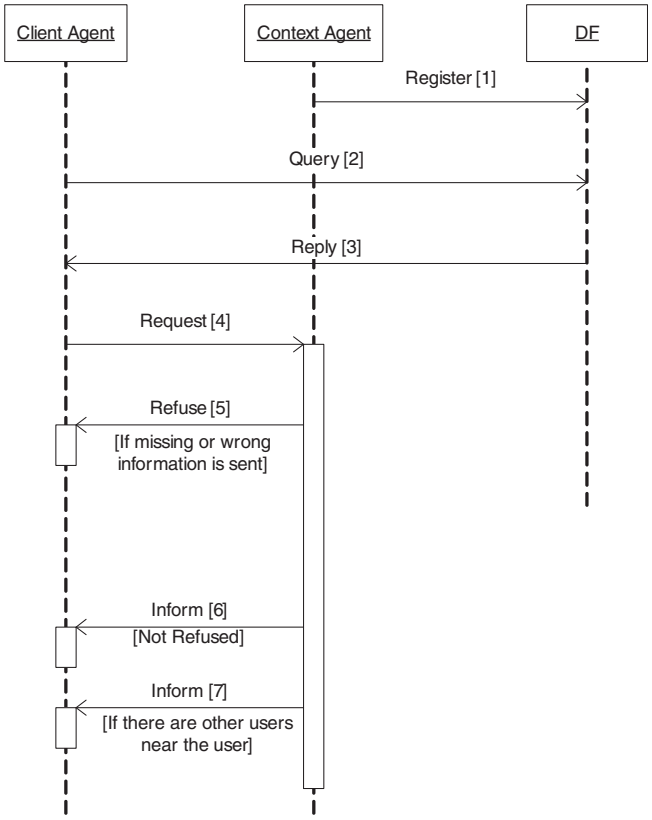

Fig. 5. Interactions of the context agent with a client agent in iConAwa system.

3.8. Context reasoning

In this section, rule-based reasoning made over the context ontology is described. High level social context is derived from low level contextual information by making rule-based reasoning over the context ontology. In this case, social context is composed of nearby users. Nearby users and nearby users with similar interests are discovered by reasoning.

Rules are defined in a file and the context agent acquires rules from this file. System administrator can add new rules or update existing rules by simply updating this file. New deductions inferred from new rules are shown to the user without any need to change the application's source code. Hence context reasoning is separated from the application's source code and a flexible and easily extensible application is implemented. By defining new rules, further functionality for the context reasoning part can be achieved.

Backward chaining technique is used for reasoning because of the possibility of huge number of facts and little number of rules. Also the system is highly dynamic because of users continuously changing their location, so the facts also change as the user moves. This makes forward reasoning less feasible for the situation.

The content of the rules file is as follows:

```
@prefix ns: http://cs.ege.edu.tr/ContextOntology#
[FindNear: (?c1 ns:near ?c2) <-
(?c1 ns:hasLocation ?loc1)
(?c2 ns:hasLocation ?loc2)
(?loc1 ns:hasLatitude ?lat1)
(?loc1 ns:hasLongitude ?lon1)
(?loc2 ns:hasLatitude ?lat2)
(?loc2 ns:hasLongitude ?lon2)
(?lat1 ns:Lat_Value ?latval1)
(?lon1 ns:Long_Value ?lonval1)
(?lat2 ns:Lat_Value ?latval2)
(?lon2 ns:Long_Value ?lonval2)
isNear(?latval1,?lonval1,?latval2,?lonval2)
notEqual(?c1,?c2)]
[FindNearAndMatch: (?c1 ns:nearAndMatch ?c2) <-
(?c1 ns:near ?c2)
(?c1 ns:hasPersonalProfile ?prof1)
(?c2 ns:hasPersonalProfile ?prof2)
(?prof1 ns:Interests ?interest)
(?prof2 ns:Interests ?interest)]
```

Rules file contains two rules: *FindNear* and *FindNearAndMatch*. Namespace is defined by *@prefix* statement. Rules are defined as backward chaining rules. *FindNear* rule defines *near* object property and *FindNearAndMatch* rule defines *nearAndMatch* object property. By *FindNear rule*, *near* relation (object property) is defined between context individuals which are near and are different from each other. To achieve this Jena framework is extended by defining a new primitive called "isNear". This primitive is registered to the framework as a built-in primitive instantiated from *NearFunctor* class which extends *BaseBuiltin* class of the Jena framework. This primitive checks if the coordinates of the users are less than or equal to a predefined value which is 100 meters. Haversine formula is used for calculating the distance between the coordinate values.

By *FindNearAndMatch* rule, *nearAndMatch* relation is defined between context individuals which are relationed with *near* property and which have at least one interest in common. Object property *near* defined in *FindNear* rule is used in defining *FindNearAndMatch* rule.

Context agent continuously updates location of each logged in user's context individual in the context ontology when a user's location changes. Jena updates inference model as the context ontology is changed. Nearby users are obtained by finding context individuals which have near relation with the user's context individual. Nearby users with similar interests are obtained by finding context individuals which have *nearAndMatch* relation with the user's context individual. Then user can communicate with nearby users by sending messages.

4. Case study

In this section, iConAwa is described with a case study where two users are provided with context-aware information. For this goal, two context individuals and their related individuals are created using Protégé ontology editor. *Context_1* is first user's context individual, whereas *Context_2* is the second user's context individual. Other information related to the users is shown in Table 1.

In POI ontology, three unreal points of interest are created. The names of these POI individuals are *Atlas_Cinema*, *Deniz_Cinema* and *Ege_Restaurant*. Further information relating to these points is shown in Table 2.

Table 1
User's context individuals and their related properties.

	1st User	2nd User
Context individual	*Context_1*	*Context_2*
Password	c1	c2
Interests	Cars, history, architecture	History, architecture, photography
Film Preferences	Comedy, adventure, horror	
Food Preferences	Meat	

Table 2
POI individuals and their related properties.

	Atlas Cinema	Deniz Cinema	Ege Restaurant
POI individual	Atlas_Cinema	Deniz_Cinema	Ege_Restaurant
Location	38 27 33 N 027 12 36E	38 27 46 N 027 12 40E	38 27 38 N 027 12 49E
Type	Film	Film	Food
Keywords	Adventure, action, thriller	Drama, comedy, horror	Meat, pizza, fast food

To realize this scenario, one context agent and two client agents are created and started. According to the scenario, when the first user logs in, entering his/her user ID and password, information describing points that are closer than 3 km and open at the time are listed in descending order according to their match degree to the user in a text area as shown in Fig. 6. The location information is obtained from the GPS receiver of the mobile computer. Also the user can see POIs on a map as shown in Fig. 7. The user can scroll through the map, change zoom level and map type of this map. The map is updated automatically as the user's context changes. The user can get services by clicking the "Services" button. "Send Message" is disabled because there are currently no nearby users.

Then the second user logs in as shown in Fig. 8. Note that this time only two points are listed, because distance of the third point is greater than 3 kilometers. Because the second user does not have any preferences, match degrees of the points are 0.

According to the scenario, the second user moves to the same location as the first user as shown in Fig. 9. In addition to listing

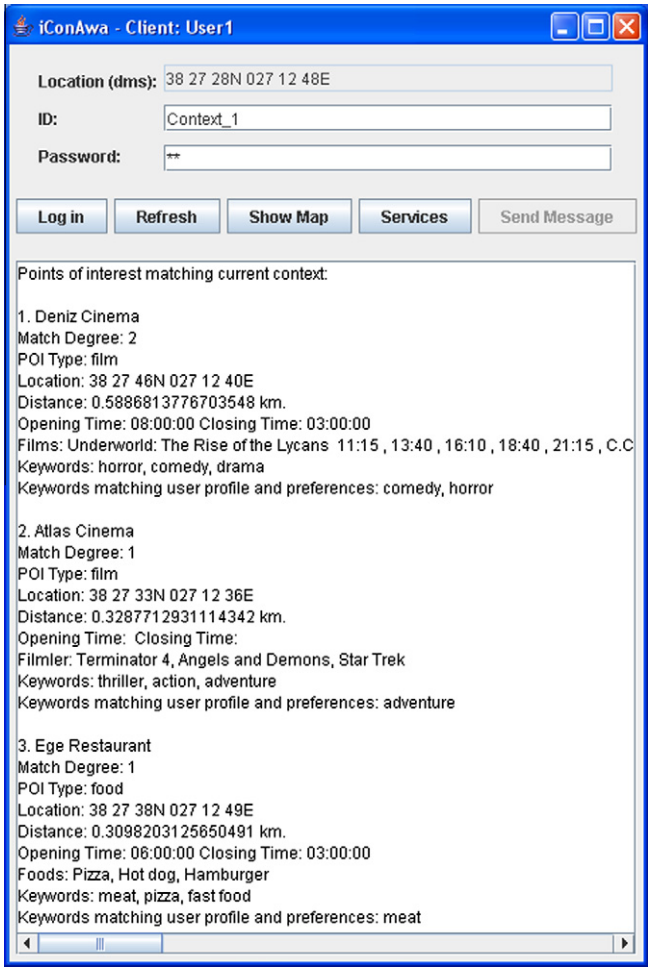

Fig. 6. Providing first user with context-aware information.

of points and their information, this time nearby users and nearby users with similar interests discovered from reasoning are shown to the user. Because first user is near to the second user, first user's user ID is shown to the second user. Also these two people have common interests such as "history" and "architecture". The second user is also notified about this situation. The second user can send messages to the first user by clicking the "Send Message" button.

5. Related work

In this section, related work about modelling the context, context-aware applications where context is modelled using ontologies and agent-based context-aware applications are reviewed. CONON, an ontology-based context model; COMPASS, a context-aware application which makes use of context and points of interest ontologies; Gulliver's Genie and CRUMPET which are agent-based context-aware applications are described. Finally iConAwa system is compared with these surveyed work and then evaluated accordingly.

5.1. CONON

In this related work, CONON (context ontology), an extensible context ontology which is developed to model the context and support logic-based context reasoning, is proposed. Context ontology is encoded in OWL. Context is thought mainly consisting of location, user, activity and computing environment contexts. Context is modelled by combining upper ontology and domain-specific ontology. Upper ontology is high level and more general. General properties of basic contextual entities are stored in the upper ontology. Domain-specific ontology consists of ontologies which define the details of general concepts and their features in each sub-domain (home, office, vehicle, etc.). Therefore, it is discussed that reuse of general concepts and flexibility to add specific concepts in different application domains are provided. Also, by logical reasoning over the ontology model, consistency of the contextual information is checked and high level implicit context is derived from low level explicit context. Rule-based reasoning and description logic reasoning are used (Wang et al., 2004).

In this related work, it is concluded that ontology-based context models are applicable and are suitable for modelling the context and providing reasoning support.

5.2. Gulliver's Genie

Gulliver's Genie is a context-aware mobile tourist guide application. In Gulliver's Genie, contents which can be comprised of text, image, audio and video are presented to the user according to the user's context proactively through user's PDA. Context consists of location, time, direction, user and mobile device contexts. Context model is not ontology-based. While the user is using the system; location, orientation and nearby points of interest are shown to the user on a map and the map is updated dynamically. Also user can watch presentations describing points of interest. Presentations are downloaded from the server using a pre-caching strategy, because of low bandwidth of wireless networks. When the mobile user heads for a direction, the points of interest that the user will arrive are predicted and downloaded before the user arrives there. Thus when the user arrives, presentations about nearby points of interest are ready for watching. The presentations and the content presented to the user are specific to the user's profile (O'Grady & O'Hare, 2004; Schwinger, Grün, Pröll, Retschitzegger, & Schauerhuber, 2005).

Client side consists of PDA, GPS receiver and GPRS (General Packet Radio Service) wireless network technology. Each PDA has

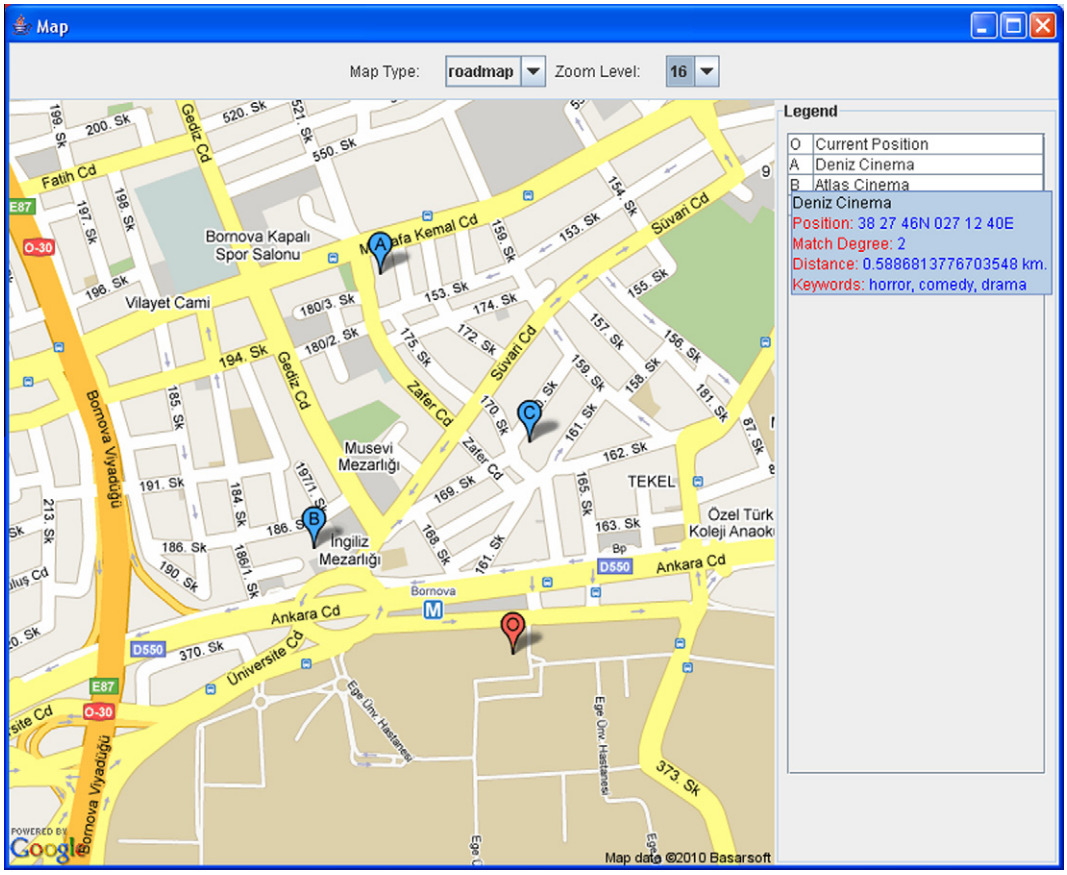

Fig. 7. Providing first user with context-aware information shown on a map.

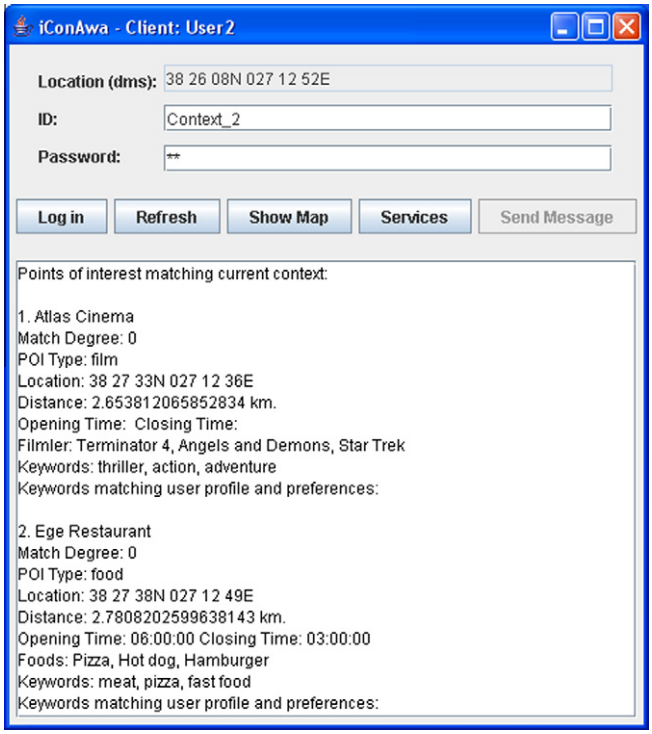

Fig. 8. Providing second user with context-aware information.

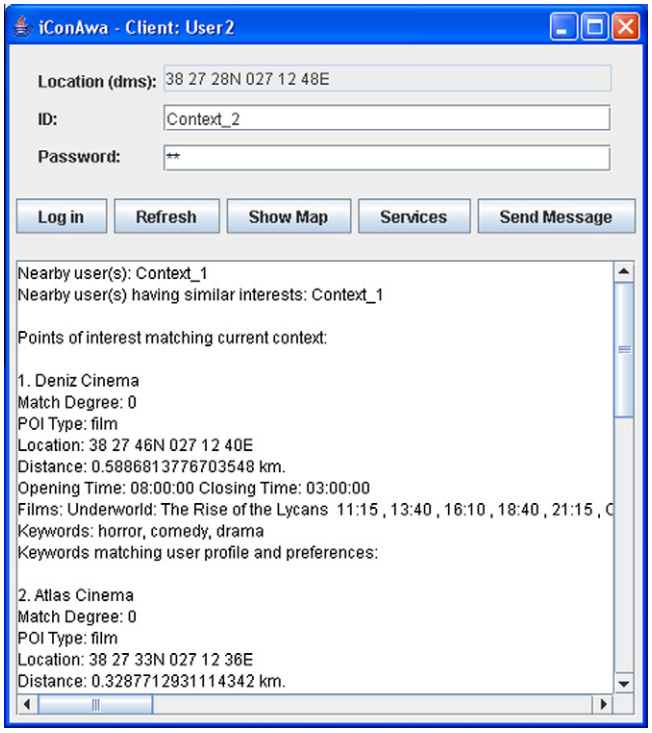

Fig. 9. Providing second user with context-aware information when the user moves to the same location as first user.

a GPS receiver and wireless network support. Detection of the user's orientation is done through an electronic compass. On the server side, presentation data, maps and user profiles are stored in a database (O'Grady & O'Hare, 2004).

Gulliver's Genie is a multi-agent system. BDI agent architecture is used as the agent architecture and the system makes use of reasoning through BDI agents.

In this related work, it is concluded that agents and even more, BDI agents can run in mobile devices and can be used in context-aware systems.

5.3. Compass

COMPASS (Context-aware Mobile Personal Assistant) is a mobile tourist application which provides users with context-aware recommendations and services. COMPASS is built on the WASP (Web Architectures for Services Platform) platform, which supports web service based context-aware applications. COMPASS keeps the records of third party service providers which provide information about points of interest such as museums, restaurants, hotels, etc. Services are defined in OWL. Context is derived from context services. Context consists of location, time, user, weather and traffic contexts. It is stated that high level implicit context is derived from domain-specific rules, but it is not stated whether rule-based reasoning is used or not. For example, whether a user is walking or is in a vehicle can be determined by looking at the user's speed and position properties (road, sea, walking area, air) or by looking if the mobile device is mounted to the vehicle or not. Context and points of interest models are ontology-based. As the context changes, map and points of interests on the device display are constantly updated (Pokraev et al., 2005; Schwinger et al., 2005; van Setten, Pokraev, & Koolwaaij, 2004).

In this related work, context-awareness and recommendation system properties are combined. It is also concluded that context-awareness and recommendation systems complement and improve each other.

5.4. CRUMPET

CRUMPET (Creation of User-friendly Mobile Services Personalized for Tourism) is a European Union project which aims at creation of user-friendly mobile services personalized for tourism. Using this application, users can request information and suggestions about current activities and points of interest. User's location and points of interest can be seen on a map via a PDA. When the user gets near to certain points of interest, related information are presented to the user proactively. Context consists of location, mobile device, network and user contexts. Location information is extracted from GPS receiver. Mobile device context consists of display size, resolution, color depth and graphical capability information and it is taken into consideration. User interests are learned dynamically by considering context history and user interactions. CRUMPET is a multi-agent system. Only the user model and broker agents are ontology based. A general context ontology model is not present and there is no reasoning (Poslad et al., 2001; Schmidt-Belz, Poslad, & Zipf, 2002; Schwinger et al., 2005).

In this related work, an agent-based scalable mobile tourist guide which gives personalized and context-aware support to mobile users and includes an adaptable ontology-based user model is proposed.

5.5. Evaluation

In this section, related work are reviewed and compared to iConAwa as shown in Table 3. In CONON, an ontology-based context model is proposed. High level context is obtained from low level contextual information by reasoning. Gulliver's Genie context-aware application is agent-based, employs reasoning, but does not use an ontology-based context model. COMPASS context-aware application is not agent-based, uses ontology-based context and points of interest models, but it is uncertain whether rule-based reasoning is used or not. CRUMPET context-aware application is agent-based, only the user model is ontology-based, but it does not include an ontology-based context model and does not employ context reasoning.

Table 3
Comparison of iConAwa with other related work.

Work	Agent-based	Context ontology	Reasoning
CONON	–	Yes	DL/Rule-based
Gulliver's Genie	Yes	No	Yes
COMPASS	No	Yes	Yes
CRUMPET	Yes	No	No
iConAwa	Yes	Yes	Rule-based

iConAwa system described in this paper is context-aware, agent-based, includes ontology-based context and points of interest models, and rule-based reasoning is made over the context model. By applying rule-based reasoning over the context ontology, high level and implicit context is obtained from low level contextual data. iConAwa system is novel and different from the previously reviewed related work by employing a combination of the above features.

6. Conclusion

Latest technological achievements have enabled ubiquitous computing where users can have access to information and resources anytime and anywhere. Context-awareness is an important step forward for the applications running in these environments for them to adapt to these kinds of highly dynamic and complex environments. Context-aware applications can adapt to changing situations easily and by distinguishing relevant information from irrelevant information they provide better functionality for the user.

Software agents are software entities that behave autonomously and adapt easily to dynamic and heterogeneous environments. Context-aware systems are suitable for agent-based development. Running environments of both context-aware applications and agents are similar. Both of these systems focus on ubiquitous computing and run in open world and dynamic environments. Context-aware applications require automatic decision making and taking an action. Agents being autonomous, reactive and proactive makes agents suitable for use in context-aware systems.

In this paper, design and development of iConAwa system, an intelligent context-aware system, developed for proactively providing mobile users with context-aware information is described. iConAwa is a context-aware and multi-agent system. Context and point of interest ontologies are developed in OWL. Context and points of interest are modelled in a flexible and extensible way by the developed ontology models. Knowledge sharing and knowledge reuse are also provided by using these ontology models. iConAwa makes use of rule-based context reasoning which provides derivation of high level implicit context from low level explicit context. The rules used for reasoning are defined in a rules file. In this approach, context reasoning is decoupled from the source code of the system. In all of these aspects, the system is an intelligent system. By defining new rules in the rules file, new functionalities can be added to the system with ease without the need to change the source code of the programs. And hence a flexible and extensible system is developed.

In development of iConAwa, Java programming language and JADE agent development framework are used. JADE provides the communication infrastructure and the directory service of

iConAwa. For manipulating ontologies and making rule-based reasoning, Jena semantic web framework is used. Jena framework is futher extended by defining a new primitive to find nearby users.

iConAwa system proposed in this paper is original because of; the system being a context-aware and multi-agent system, ontology based context and point of interest models and also because the system makes use of rule-based reasoning over the context ontology to derive new high level context. The system also makes decisions to reduce network usage. iConAwa system presented in this paper is novel and different from other related work by combining these features in one system.

References

Abowd, G. D., Atkeson, C. G., Hong, J., Long, S., Kooper, R., & Pinkerton, M. (1997). Cyberguide: A mobile context-aware tour guide. *Wireless Networks, 3*, 421–433.

Asthana, A., Crauatts, M., & Krzyzanowski, P. (1994). An indoor wireless system for personalized shopping assistance. In *WMCSA '94: Proceedings of the 1994 first workshop on mobile computing systems and applications, Washington, DC, USA: IEEE Computer Society* (pp. 69–74).

Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M. C., & Palinginis, A. (2005). Modelling context for information environments. In *Ubiquitous mobile information and collaboration systems. Lecture notes in computer science* (Vol. 32, pp. 43–56). Berlin Heidelberg: Springer.

Chen, H., & Finin, T. (2003). An ontology for a context-aware pervasive computing environment. In *IJCAI workshop on ontologies and distributed systems*.

Chen, G., & Kotz, D. (2000). A survey of context-aware mobile computing research. Technical Report Dartmouth College Hanover, NH, USA.

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

Dey, A. K., & Abowd, G. D. (1999). Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22 College of Computing, Georgia Institute of Technology.

FIPA. (2004). Fipa agent management specification. SC00023K.

Gonçalves, B., Costa, P., & Botelho, L. M. (2008). Context-awareness. In M. Schumacher, H. Schuldt, & H. Helin (Eds.), *CASCOM: Intelligent service coordination in the semantic web Whitestein series in software agent technologies and autonomic computing*. Birkhäuser Basel.

Gu, T., Wang, X. H., Pung, H. K., & Zhang, D. Q. (2004). An ontology-based context model in intelligent environments. In *Proceedings of communication networks and distributed systems modeling and simulation conference, San Diego, CA, USA* (pp. 270–275).

Harter, A., Hopper, A., Steggles, P., Ward, A., & Webster, P. (2002). The anatomy of a context-aware application. *Wireless Networks, 8*, 187–197.

Henricksen, K., Indulska, J., & Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. In *Proceedings of the first international conference on pervasive computing*.

Henricksen, K., Indulska, J., & Rakotonirainy, A. (2003). Generating context management infrastructure from high-level context models. In *Proceedings 4th international conference on mobile data management*.

Hull, R., Neaves, P., & Bedford-Roberts, J. (1997). Towards situated computing. In *Proceedings of the 1st IEEE international symposium on wearable computers, Washington, DC, USA: IEEE Computer Society* (pp. 146–153).

JADE. (2010). Introduction to jade. Accessed 04.08.10.

Jena. (2010). Jena – A semantic web framework for java. Accessed 04.08.10.

Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J., & Walker, K. R. (1997). Agile application-aware adaptation for mobility. In *Proceedings of the sixteenth ACM symposium on operating systems principles*.

O'Grady, M. J., & O'Hare, G. M. P. (2004). Gulliver's genie: Agency, mobility & adaptivity. *Computers & graphics, special issue on pervasive computing and ambient intelligence – Mobility, ubiquity and wearables get together, 28*, 677–689.

Pokraev, S., Koolwaaij, J., van Setten, M., Broens, T., Costa, P. D., Wibbels, et al. (2005). Service platform for rapid development and deployment of context-aware, mobile applications. In *Proceedings of the 4th IEEE international conference on web services, Washington, DC, USA: IEEE Computer Society* (pp. 639–646).

Poslad, S., Laamanen, H., Malaka, R., Nick, A., Buckle, P., & Zipl, A. (2001). Crumpet: Creation of user-friendly mobile services personalised for tourism. In *Proceedings of second international conference on 3G mobile communication technologies*.

Priyantha, N. B., Chakraborty, A., & Balakrishnan, H. (2000). The Cricket location-support system. In *Proceedings of the sixth annual ACM international conference on mobile computing and networking, New York, NY, USA: ACM* (pp. 32–43).

Protege. (2010). Protege. Accessed 04.08.10.

Ranganathan, A., & Campbell, R. (2003). A middleware for context-aware agents in ubiquitous computing environments. In *Proceedings of ACM/IFIP/USENIX international middleware conference*.

Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. In *Proceedings of the workshop on mobile computing systems and applications, IEEE Computer Society* (pp. 85–90).

Schilit, B., & Theimer, M. (1994). Disseminating active map information to mobile hosts. *Network, IEEE, 8*, 22–32.

Schmidt-Belz, B., Poslad, N. A. S., & Zipf, A. (2002). Personalized and location-based mobile tourism services. In *Proceedings of mobile HCI '02 with the workshop on mobile tourism support systems*.

Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. V., & Velde, W. V. D. (1999). Advanced interaction in context. In *Proceedings of first international symposium on handheld and ubiquitous computing, London, UK: Springer-Verlag* (pp. 89–101).

Schwinger, W., Grün, C., Pröll, B., Retschitzegger, W., & Schauerhuber, A. (2005). Context-awareness in mobile tourism guides – A comprehensive survey. Technical Report Johannes Kepler University Linz.

van Setten, M., Pokraev, S., & Koolwaaij, J. (2004). Context-aware recommendations in the mobile tourist application compass. In *Proceedings of the second IEEE annual conference on pervasive computing and communications workshops, IEEE Computer Society* (p. 18).

Wang, X. H., Zhang, D. Q., Gu, T., & Pung, H. K. (2004). Ontology based context modeling and reasoning using OWL. In *Proceedings of the second IEEE annual conference on pervasive computing and communications workshops, IEEE Computer Society* (p. 18).

Want, R., Hopper, A., Falcao, V., & Gibbons, J. (1992). The active badge location system. *ACM Transactions on Information Systems, 10*, 91–102.

Want, R., Schilit, B. N., Adams, N. I., Gold, R., Petersen, K., Goldberg, D., et al. (1996). The parctab ubiquitous computing experiment. In *Mobile computing. The Kluwer international series in engineering and computer science* (Vol. 353, pp. 45–101). US: Springer.

Werb, J., & Lanzl, C. (1998). Designing a positioning systems for finding things and people indoors. *IEEE Spectrum, 35*, 71–78.

Wooldridge, M. (2002). *An introduction to multiagent systems*. John Wiley & Sons Ltd.

Wu, H., Siegel, M., & Ablay, S. (2002). Sensor fusion for context understanding. In *Proceedings of IEEE instrumentation and measurement technology conference*.