A Middleware for Managing Sensory Information in Pervasive Environments

Costas Pontikakos and Theodore Tsiligiridis

Division of Informatics Mathematics and Statistics
Agricultural University of Athens
75 Iera Odos, 11855, Athens, Greece
tsili@aua.gr

Summary. Context and context-aware computing have attracted remarkable attention in recent years. Research into wireless sensor networks is rapidly moving from simulations to realistic testbeds. The work presented here investigates the architecture and design of an agent-based sensor network for monitoring and spraying control of olive fly in an ubiquitous precision farming environment. It contributes by providing a generic, flexible and extensible model for handling heterogeneous sensor data to be managed using a simple user interface. The innovative aspect stands for the development of a multi-agent system and focuses on the communication aspects of the proposed middleware architecture. The modeling allows the encapsulation of different modules as agents and integrates them, by utilizing a standardized XML schemas in order to hide the complexity to the users. The proposed communication design adopts a layered architecture approach, which is based on some software agents that solve different tasks and communicate their results and requests.

1 Introduction

Ambient Intelligence (AmI) relies on the areas of ubiquitous computing, ubiquitous communication and intelligent user interfaces aiming at seamless delivery of services and applications. The vision suggests an environment of potentially large number of embedded mobile devices, or software components, interacting to support user goals and activities. It also suggests a component-oriented view in which the components are independent and distributed. The environment is characterized by the autonomy, reactivity, distribution, collaboration and adaptation of its artifacts, and in this sense, they share the same characteristics as agents [8]. AmI requires these agents to be able to interact with numerous other agents in the environment around them in order to achieve their goals.

Agent technology has been considered an alternative for enhancing pervasive computing environments [2]. The environment provides the infrastructure that enables AmI scenarios to be realized. In relation to pervasiveness, it is

important to note that scalability, by means of the need to ensure that large numbers of agents and services are accommodated, as well as heterogeneity of agents and services must be facilitated by the provision of appropriate ontologies [3]. In most of the cases relevant context is assumed known in advance, and agents just learn how to adapt their decision to it. However, agent-based architectures that making an explicit separation between selection of relevant context from the decision making process of the agent have also been proposed [1]. Addressing all of these aspects will require efforts to provide solutions to issues of context architecture, operation, integration and visualization of distributed sensors, ad-hoc services, and network infrastructure.

Agricultural Information Systems (AIS) that provide access to electronic agro-environmental (e.g biological, climatic, meteorological, etc.) records are a step in the direction of providing accurate and timely information to farmers in support of decision-making. This has motivated the introduction of ubiquitous computing technology in precision farming based on designs which respond to particular conditions and demands. It is therefore important to realize how autonomous agents can cope with the complexities associated in implementing AmI environments for precision farming settings. Note that farms are distinguished by the distributed nature of the information, the intensive collaboration and mobility of their personnel, as well as their need to access agro-environmental information occasionally. We suggest an iterative user-centered development method to understand the farm activities and to envision farms AmI scenarios, in which the main systems components were identified as agents that respond autonomously in accordance with the context surrounding the activities performed at the farm. Specifically, based on the envisioned scenarios and the easy integration of components represented by autonomous agents, we have conceived a flexible middleware, called *Dacus_oleae* for spray control of olive trees. In the present work we focused on the communication aspects of the middleware architecture, allowing different context-aware applications to query, retrieve and use sensor data in a way that is decoupled from the mechanisms used for acquiring the raw sensor data.

2 Motivating AmI scenarios in Precision Farming

The olive fly (*Bactrocera (Dacus) oleae*) (Gmelin) (Diptera: Tephritidae) is the most serious insect pest that affects the olive tree (*Olea europea*) cultivation causing significant qualitative and quantitative consequences. Olive flies survive best in cooler coastal climate, but are also found in hot, dry regions. The optimum temperature for development is between 20^0C to 30^0C. High temperatures (35^0C or more) are detrimental to adult flies and to maggots in the fruit. However, since the flies are very mobile they have the ability to seek out cooler areas of the orchard and urban trees. The olive fly has three, and perhaps as many as five, generations per year depending upon local con-

ditions. Over-wintered adult populations decline to low levels, however new adults from over-wintered pupae begin to emerge in spring.

In order to monitor insect populations adequately, it is important to record both biological and climatic data. These data need to be collected in real time and give information about both adult and larval stages. The use of such information is only valid if it is incorporated into pest management models. In the case of olive fly, monitoring of adults in traps and observations of larval stages in fruit samples are coupled with climatic data (temperature, relative humidity etc.) to make predictions of damage and take preventive measures. Such climatic data is collected from automatic agro-climatic weather stations, which are capable of recording and storing great quantities of such data and sending it automatically to a central collection point.

Several approaches exist to treat the above pest problem [7]; however spray applications from the ground seem the most appropriate for several reasons, such as climatic, environmental protection, etc.. To avoid failures in the spray treatment care must be taken to ensure that: the experimental plants of olive trees are large enough; the population of olive flies tend to increase significantly; the olives must be in an advanced stage; the female to male insect ratio should be greater than one; the female insects must be in a mature stage; the temperature and the humidity levels should exceed a threshold.

The spray application takes place during a day using in most of the cases tractors. Each tractor covers one section of the spraying area. During the spray application several problems may arise. The first problem is that the air temperature and air speed values are unknown to the spraying attendant. In this case, the spraying could continue even when the meteorological conditions have been violated. The second problem arises because the spraying areas cannot be memorized by the attendant and therefore over or under spraying may occur. The third problem is that the spray volume is dependent on the coverage of olive trees and therefore the spraying attendant cannot easily determine if the number of the olive trees per area unit is low, medium, or high. The attendant is not aware of the existing areas inside the spraying area which must not be sprayed for some reason (i.e. domestic areas). The fourth problem arises because the olive fly population of the spraying area is not known to the attendant and therefore the spray volume per area cannot be determined at all. The fifth, but not last problem is the lack of communication between the supervisor and the attendants. Since the exact location of the attendant and the spraying coverage are not always known to the supervisor the whole spraying process can be altered.

3 The *Dacus_oleae* middleware

3.1 The *Dacus_oleae* WSN

The four-tiered architecture of the *Dacus_oleae* Wireless Sensor Network (WSN) shown in Fig. 1 is designed to transmit the data from an array of sen-

sors to a Field-zone Gateway (FzG), through their cluster-heads(Chs), using a two-way data stream over a wireless link. Depending on the geographical conditions and sensors density the WSN can be divided into several field-zones ($Fz.x\ ; x = 1, 2, \ldots k$). To facilitate query dissemination and efficiency in-network processing, a field-zone is further divided into clusters; however a cluster cannot be shared among different field-zones. Thus, field-zones as well as clusters are disjoint. In each cluster, there are many sensors coordinated by exactly one Ch. Sensors are responsible for all sensing-related activities; however sensors do not communicate with other sensors in the same or other clusters, and usually are independently operated. Chs have much more re-

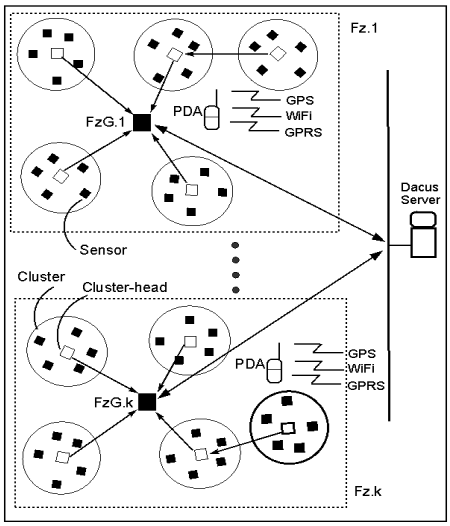

Fig. 1. Physical view of the four-tier architecture of *Dacus_oleae* WSN.

sponsibilities than sensors. First, a Ch receives raw data from all active sensors in the same cluster. It may also instruct sensors to be in sleep, idle, or active state, if some sensors are found to always generate irrelevant, inaccurate, or duplicated data, thereby allowing these sensors to be reactivated later when some existing active sensors run out of energy. Second, a Ch creates an application-specific local view for the whole cluster by exploring correlations among the data sent from sensors. Excessive redundancy in raw data can be alleviated, and the fidelity of captured information should be enhanced. Third, a Ch forwards the composite bit-stream toward a FzG that generates a comprehensive global-view for the entire field-zone. However, to maintain sufficient network connectivity Chs have no sensing capabilities. They are used as communication relays, aggregating traffic and routing the data to their FzG. The FzG will also play the role of a data repository; thus it will make decisions about what data to pass on, such as local area summaries and filtering in

order to minimize power use while maximizing information content. FzGs will then forward the data to the *Dacus_server*. To further improve communication, Chs and FzGs are installed at an appropriate height and optionally, can be involved in inter-relaying, if such activities are applicable and favorable.

The logical four-tiered architecture of the proposed WSN offers a flexible balance among reliability, redundancy, and scalability . Under this architecture, the primary goal of lower - tiers (sensors and Chs) is to gather data as effectively as possible, while the upper-tiers (FzGs and *Dacus_server*) are designed to move information as efficiently as possible. With this functionality partition, we can optimize the performance of individual tiers separately since they are designed for different purposes and have various concerns.

Finally, additional sensors could be used for environmental monitoring and to take measurements of atmospheric pollution. Thus, each field zone is equipped with a weather station registering the luminosity, air pressure, precipitation, wind strength and direction.

3.2 Design of *Dacus_oleae* Middleware

The proposed architecture focuses on the development and deployment of added value services and combines the capabilities of several technologies. Modules, in a form of agents communicate with each other sending and receiving data, while their collaboration is coordinated via the central receiving point. Thus, communication, coordination and cooperation are ensured. The agents are able to perform some actions on other objects, thus at least partially perceiving their environment and communicating among themselves. Each agent perceives a modification of its environment and receives messages from other agents. The agent then reacts to these stimuli by acting on the environment and sending messages to other agents according to its own methods and characteristics. The proposed model uses XML configuration files to simply add or remove sensors. It also adds new information recipients or listens to other existing sensors, by modifying specified XML files. The different sensors are abstracted by internal software drivers to facilitate the interaction. The model provides also an interface to interact with the data base in a simple way. Figure 2 shows a screen shot of the interface that can be used by an attendant in order to submit an SQL-like query. The input text fields are used to enter the threshold values for the sensor readings. Sensors detect temperature, humidity and air pressure. The values presented here are in a generic form, they do not relate to actual physical units, and for demonstration purposes, only an AND operator is used in the WHERE clause. Fig. 2 also shows a sample result for the query made. The result shows the readings received from all the sensors from Fz.2 with temperature greater than 22.5.

The queries can be adjusted according to the request. For example, the users may retrieve the most recent, average, maximum, or minimum value, received from any or all the sensors. Also, they may interesting in periodic sensor readings, namely; "temperatures from sensors of the Fz.3 every twelve

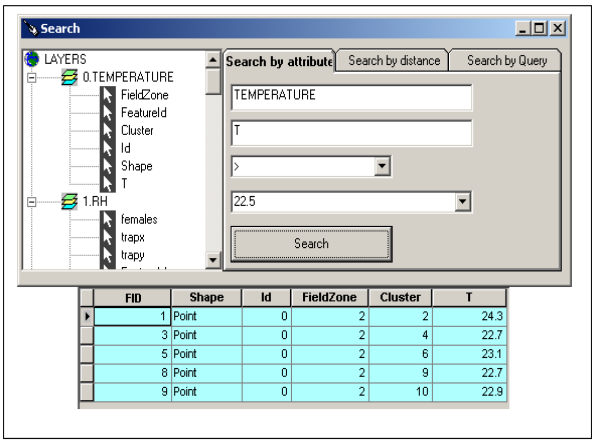

Fig. 2. GUI input and output of the SQL query (search by attribute): *select *from* temperature *where* $T > 22.5$

hours". In fact queries may consist of *select - from - where - group by - having* blocks to support selection, join, projection, aggregation and grouping. For example, to count the number of sprayed trees in the Fz.2 we use the query:

```
SELECT
  Count(trees.tree_sprayed) AS CountOftree_sprayed
FROM trees
GROUP BY trees.tree_sprayed, trees.fieldzone, trees.cluster
HAVING(((trees.tree_sprayed)="yes")AND((trees.fieldzone)=2));
```

4 Communication Module

The Communication (Com) module is a layered, hierarchically structured software, classified into different types of mobile agents and in accordance with their functionalities. It is related with the developed WSN, described in subsection 3.1, the network-based operations, the sensors' hardware, as well as the data acquisition mechanisms [4]. Methods regarding how WSNs can benefit from the use of autonomic techniques of multi-agent systems are discussed in [6] and others. The network operations of the Com module are distributed into four layers each one of which is controlled by specific task oriented mobile agents. Agents in each layer achieve their assigned task by collaborating with each other and with other agents in other layers and transmit the processed data to their destination layer. The architecture consists the following four layers; the Interface Layer (IL), the Field-zone Layer (FzL), the Cluster Layer (CL), and the Sensor Layer (SL). At the IL, an Interface Agent (IAnt) provides interaction, by means of flexible mediation services, between the users

and the *Dacus_oleae* WSN. At the FzL, the Field-zone Agent (FzAnt) acts as a gateway between the clusters of the sensors and the *Dacus_server*, whereas, at the CL, the Query Agent (QAnt) receives the required sensing data captured by the sensor acquisition devices, performs query dissemination and network processing. Finally the SL, is responsible to acquire the sensor data, namely it acts as data provider for the CL. Fig. 3 presents the architecture which operates as follows: The IAnt receives and translates the SQL-like user requests, and proceeds them into the appropriate FzL. The corresponding FzAnt accepts the queries and through the QAnts it includes them into the CL. The Chs obtain the sensor results from the SL and the QAnts pass the results to the IAnt through the FzAnt.

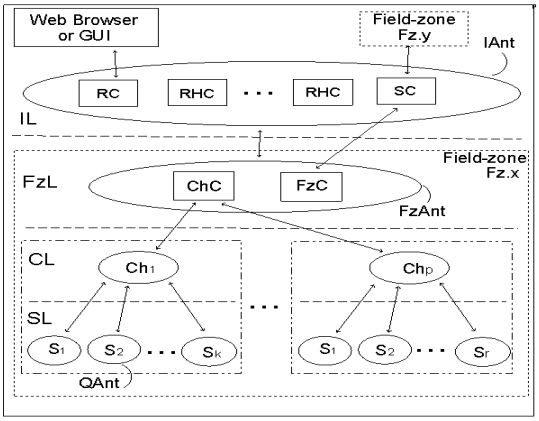

Fig. 3. The four layer, agent-based, architecture of the Com Module.

Interface Layer

The Interface Layer (IL) is responsible for accepting the user requests, processing them and displays the results in a predefined format. It is also responsible for the messages sent over several networks (i.e. SMS, E-Mail, etc.). Since it is possible to send an information structure to a recipient (e.g. an XML file), where the recipient can be a human, a different machine, or simply a software module, the chain can be extended indefinitely. Messages can also be stored in a separate database for further handling. The IL is the front-end of the system providing the user interface or services that are used to query the sensor network and thus it might address several scenarios. The user interface agents can be implemented by any portable device of the client, such as a laptop, or a PDA; however, for the time being it is running on a laptop. Note that the

design of the IAnt does not depend on the size and complexity of the WSN, but it is based on the *Dacus_oleae* application and the user profile.

Form-based queries are submitted by different type of users to the XML-based IAnt. As soon as the IAnt receives a query, it maintains the connection with the users interface; for example, if a user uses a web browser an HTTP connection is assumed (Fig. 3). The IAnt consists of the three well known component classes; The Receiving Component (RC), the Request Handler Component (RHC), and the Service Component (SC). The RC is always active and responsible for the processing of all user requests. For each user request an RHC instance is created; therefore the total number of RHC instances depends on the number of concurrent user requests. The obligation of the RHC is to process the user request, to format the query result for the user and to maintain the connection in progress (i.e, with the web browser or the GUI). Finally, the IL interacts with the FzL through the SC. The SC which is always active sends the queries to the desired FzAnts and the results received are combined and forwarded to the appropriate RHC.

Field-zone Layer

The Field-zone Layer (FzL) is responsible for accepting the user requests from the IL and forwarding them to the CL. It also handles all the information that comes from the CL. The experimental region comprises of several field-zones, each one of which is managed by its own FzAnt in the corresponding FzG. Note that the FzGs are assigned based on network topology and deployment conditions of Chs. They are also responsible for transmitting data either across their field-zone or to pass these data to another field zone, namely to the Field-zone Component (FzC) and Cluster-head Component (ChC) of the FzAnt, respectively. Both component classes interact wirelessly or through TCP/IP. The FzC interacts either with the *Dacus_oleae* server or with the other field-zones, maintaining the connection with the IAnt. The ChC interacts with the sensors through their Chs, maintaining the connection with the various QAnts and managing the radio communication with the ChC.

Cluster Layer

The Cluster Layer (CL) is responsible for accepting the user requests from the FzL and forwarding them to the SL. Each field-zone comprises of several clusters, each one of which is managed by its own Ch. For each sensor in a cluster there is a corresponding query agent (QAnt) responsible for the acquisition of sensor data, the filtering of inaccurate and irrelevant data, the aggregation and the processing of useful data, and the transmission of the desired results. Therefore the QAnts are located on the data collecting sensors and perform data acquisition and local computation in the corresponding cluster (Fig. 3). Their main functionality is to receive queries from the external component of the associated FzAnt and to re-transmit them to the desired sensors through

their Ch; any query data received from other FzAnts is immediately rejected. The sensor data is collected depending on the query. Every QAnt keeps information regarding the types of sensors available, their activity, their current energy level, as well as other complementary configuration details. Based on this information, as well as on the availability of sensors, query dissemination designs may be developed.

An interesting problem which is quite common in many agro-environmental applications like the *Dacus_oleae* application considered here is to periodically check the QAnts to see if any sensor failed to transmit the data. Usually the QAnts take turn in a TDM fashion (other protocols are also applicable) for sending the aggregation data to the CL and then to the FzL. In case some of the QAnts do not receive any respond to the query calls, real-time data can be received only from the functioning sensors. Thus, instead of computing the required parameter values with degradation (due to the non-functioning sensors) using only the available sensor readings, kriking methods may be applied in the available sensor readings to obtain more accurate result. A similar problem is to predict the required parameter values in different places from those the sensors are located. As it will be seen in section 5, kriking can be applied successfully, to predict olive fly population.

Sensor Layer

The Sensor Layer (SL) is responsible for acquiring the data from the sensors, gathering and passing it to the CL. It gets the raw data from the attached sensors and simply transforms it. The sensors themselves can be either software or hardware. Note that in software sensors raw data is taken by a software means e.g. a web service, an aggregation of the data coming form other sensors, or any another software component. The QAnt analyzes the data to maintain consistency and to filter inaccurate and irrelevant data. Then it sends the data to the FzL. The FzAnts perform data aggregation and the resulting data are sent to the IL where the IAnt make the data available to the users, displaying them on a GUI or on a web-based interface. Note that the QAnt has an additional function; as it abstracts the sensors from the other components it has the possibility to create further virtual sensors that rely usually on aggregations. The corresponding FzAnt has no means to find out whether this is a new sensor or a software one. This means, that the QAnt multiplexes the data and creates new sensors. The aggregation functions are clearly defined set of methods (e.g. SUM, AVG, etc.). The QAnt has no real information about the meaning of the data and simply executes these methods.

5 Simulation and Results

The proposed agent-based architecture is used in the design of a sensor application that monitors temperature, humidity, air pressure, light, and mobility.

The clusters, where sensors are deployed, are distributed into three field-zones, Fz1, Fz2, and Fz3, each one of which comprises of three, four, and four clusters, respectively (Fig. 4). It is useful to remind at this point that the QAnts are deployed on the sensors which provide the data, the FzAnts are deployed on the FzGs, whereas the IAnt is deployed on the client devices. Note that for each incoming user request, a RHC is created in a separate thread and it is managed by the *Dacus* web server. The control of ground spray was per-

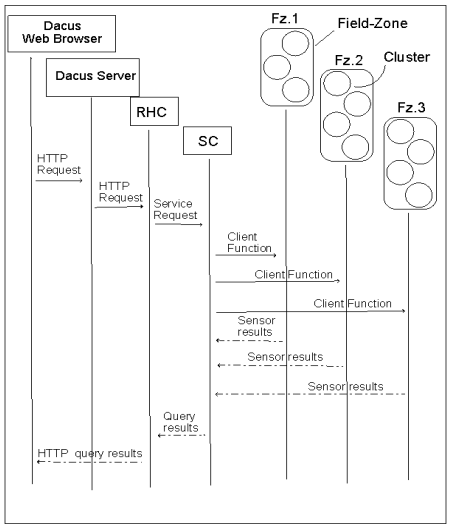

Fig. 4. Web-based query process in the pilot deployment of *Dacus_oleae* WSN.

formed by taking into account that for each field-zone the system provides the olive fly population per McPail trap and the volume of the required spray. Note that a trap only partially covers the spraying area. In each field-zone a tractor with a portable device(laptop) and wireless Internet access (Wi-Fi and/or GPRS) is assigned to perform the ground spray. The temperature is assumed to be in the range of 14^0C to 28^0C, whereas the air-speed does not not exceed 8m/s. Fig. 5 shows a screen shot of the simulation taken place which includes the following utilized information layers:

- *T, RH, AIR*: Provide the temperature, the relative humidity and the air-speed, respectively.
- *CLICK*: The layer is a GIS interactive menu. The user may invoke commands to update the GIS data, to browse though the web pages, etc..
- *ROADS*: Provides information about the roads of the spraying area.
- *TRAP*: Provides information of the location of each trap, the number of female and male insects it contains, the total number of insects and the insect population level per trap.

- *TRACTOR*: Shows the spraying areas designated by each tractor.
- *AREA*: Shows the boundaries of the area to be sprayed.
- *SPRAYAREA*: Shows the areas need to be sprayed. Also, the layer shows the olive-tree density in the field-zone (as an index taking values 1,2 and 3).
- *TRAPAREA*: Provides information about the trap area. For example, based on the level of the olive fly population per trap we developed a function that determines the extreme levels of the spray to be applied, using kriking methodology.

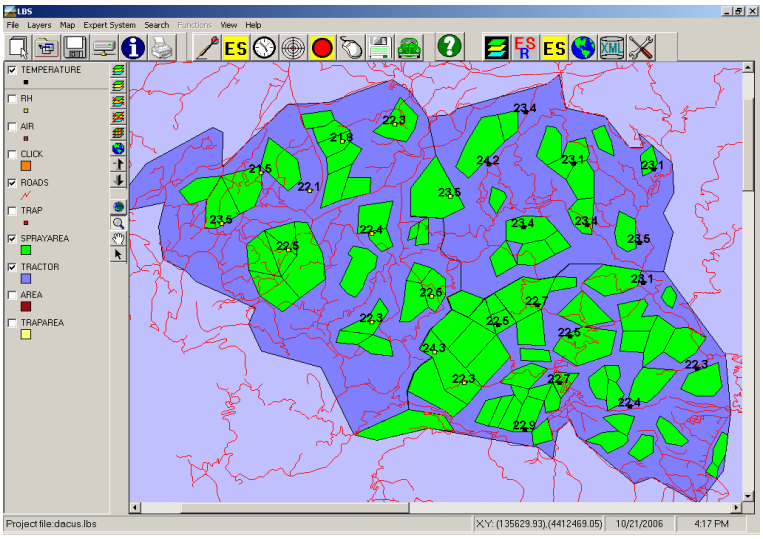

Fig. 5. The *Dacus_oleae* personalized options of the attendant.

During the spraying performance the knowledge base becomes aware about the location of the tractors in respect to the designated area of olive trees. As soon as a decision about the volume and the actual spray area is made the tractor attendant proceeds with the operation. If it is not allowed to spray the area, the system reports the attendant for the reason, i.e. temperature is too high, air-speed is too high, etc.. Note that when a tractor is in the area that is designated to be sprayed by another tractor, a location-based notification is sent to the attendant with the correct number of the tractor. Notifications are also sent by the system informing both the attendant and the spraying supervisor when a tractor is spraying within an area that has been sprayed. In such cases, the spraying supervisor may send a notification to both or to all the spraying attendants. Further, when a trap is within the range of a tractor the system notifies the attendant with an estimation of the number of insects that have been captured by this trap. An additional capability is

that the attendant may use multimedia data, such as photos of the spraying areas, sound files with advises, or orders on the spraying operation, textual information, network data concerning the spraying operation, etc.. Finally, as it is shown in Fig. 5 the attendant has the capability of personalizing the system with his options, namely, to update the database, or utilize help on the system's operation.

6 Conclusions

In this paper we proposed the design of a multi-agent middleware, called *Dacus_oleae*, for supporting a novel application for spray control and treatment of the olive fly pest problem. It provides the agents architecture, design and implementations that enable them to communicate and work together to disseminate and gather data in WSNs. The architecture consists of four layers of agents with different types of functionalities. It is open and adapts future technologies, encapsulating and integrating various modules as agents, by means of communication, with a clearly defined XML schema. The application integrates many interesting characteristics, including the independence of the positioning system and the location model support, the capability of the location based DSS, the user friendly environment with multimedia capabilities of the GUI, the flexibility in the development of new location aware application and services, etc.. Finally, the system provides efficient data dissemination, in cases where sensors are deployed in large areas with limited power.

References

1. Bocur O, Beaune P, Boissier O (2005) Representing Context in an Agent Architecture for Context-Based Decision Making. In: Proceedings of the Workshop on Context Representation and Reasoning (CRR05). Paris France EU.
2. Campo C (2002) Service Discovery in Pervasive Multi-Agent Systems. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagents Systems (AAMAS'02). Bologna Italy EU.
3. Chen H, Finin T, Joshi A (2003) An Ontology for Context-Aware Pervasive Computing Environments. The Knowledge Engineering Review 18(3):197–207.
4. Hill J, Szewczyk R, Woo A, Hollar S, Culler D, Pister K (2000) System architecture directions for networked sensors. SIGPLAN Not. 35(11):93-104.
5. Liu J, Jing H, Tang Y (2002) Multi-agent oriented constraint satisfaction. Artificial Intelligence 136:101-144.
6. Marsgh D, Tynan R, O'Kane D, O'Hare G (2004) Autonomic wireless sensor networks. Engineering Applications of Artificial Intelligence 17:741-748.
7. Montiel A, Jones O (2002) Alternative methods for controlling the olive fly, Bactrocera oleae, involving semiochemicals. Use of pheromones and other semiochemicals in integrated production. IOBC wprs Bulletin 25.
8. Tweedale J, Ichalkaranje N, Sioutis C, Jarvis B, Consoli A, Phillips-Wren G (2006) Innovations in multi-agent systems. Journal of Network and Computer Applications. In Press, Corrected Proof.