An ontology-driven system architecture for precision agriculture applications

Christos Goumopoulos* and Achilles D. Kameas

Computer Technology Institute,
Designing Ambient Information Systems Group,
N. Kazantzaki, 26500 Rio Patras, Hellas, Greece
E-mail: goumop@cti.gr
E-mail: kameas@cti.gr
*Corresponding author

Alan Cassells

Department of Zoology, Ecology and Plant Science,
National University of Ireland Cork,
Distillery Fields, North Mall, Cork, Ireland
E-mail: a.cassells@ucc.ie

Abstract: Our research has been performed in the context of the EU-funded R&D project PLANTS. In this paper, we describe an ontology-driven architecture for developing systems that can be used in precision agriculture applications. Central to our approach is the use of an ontology, which views plants and associated computation as an integral part and allows the interaction of plants and artefacts in the form of synergistic mixed societies. PLANTS ontology sets up a conceptual framework that combines the knowledge about sensors, actuators and other domain concepts available, on the one hand, and the biological studies about plant stressing and sensing mechanisms and consequent plant behaviour, on the other hand, to make plants a proactive component of agricultural systems.

Keywords: ontology; system architecture; proactive computing; precision agriculture.

Reference to this paper should be made as follows: Goumopoulos, C., Kameas, A.D. and Cassells, A. (2009) 'An ontology-driven system architecture for precision agriculture applications', *Int. J. Metadata Semantics and Ontologies*, Vol. 4, Nos. 1/2, pp.72–84.

Biographical notes: Christos Goumopoulos received his diploma and PhD Degrees in Computer Science from University of Patras, Hellas. Currently he is a Cooperating Professor in the Hellenic Open University and serves as a term appointed Assistant Professor in the University of Patras and as teaching staff in the Technological Educational Institute of Patras. His research interests include software engineering, programming languages and compilers, resource scheduling, distributed computing, ubiquitous computing and awareness management, middleware and ontological knowledge representation. He has more than 30 publications to his credit in international journals, books and conferences in these areas.

Achilles D. Kameas is an Assistant Professor with the Hellenic Open University, where he teaches software design and engineering. He is also R&D manager with CTI, where he is the head of Research Unit 3 and the founder of DAISy group (http://daisy.cti.gr). He has published over 50 papers on books, journals and international refereed conferences, co-edited more than five books and participated in several conference and workshop committees. His current research interests include architectures, ontologies and tools for engineering of ubiquitous computing applications. He is a voting member of IEEE, IEEE CS, ACM and ACM SIGCHI.

Alan Cassells received his BSc and MSc from the National University of Ireland Dublin. He undertook postgraduate research as the Universities of Glasgow and Wales (UK) and received his PhD from the latter. He was a postdoc at the Universities of Warwick and Oxford, Lecturer in the University of London (UK) and Professor of Botany at the National University of Ireland Cork from 1979 until his retirement in 2007. He was Managing Director of Plant Biotechnology (UCC) Ltd. for ten years. He has edited/co-edited seven books and published over 100 papers in plant pathology and plant biotechnology.

1 Introduction

Precision agriculture is an agricultural concept relying on the existence of in-field variability across an array of cropping systems (Koch and Khosla, 2003). Thanks to developments in the field of wireless sensor networks as well as miniaturisation of sensor systems, new trends have emerged in the area of precision agriculture. Wireless networks allow the deployment of sensing systems and actuation mechanisms at a much finer level of granularity, and a more automated implementation than has been possible before. Sensors and actuators can be used to precisely control for example the concentration of fertiliser in soil based on information gathered from the soil itself, the ambient temperature, and other environmental factors. Incorporating feedback into the system through the use of sensors, actuators, and adaptation algorithms will allow a more fine-grained analysis that could adjust flow rate and duration in a way that is informed by local conditions. One can imagine the use of such precise information in particularly sensitive high-value crops such as wine grapes, citrus fruits and strawberries.

At present, the information gathered by sensor networks deployed in a field are mainly used for monitoring and reporting on the status of the crops (Burrell et al., 2004; Zhang et al., 2004). However, agricultural environments make a good candidate for using proactive-computing approaches for applications, which require a faster than human response time or which require precise, time-consuming optimisation. For example, irrigation is a major issue in many farms. An ideal proactive system would optimise water usage in different areas of the farm by using the water available – particularly where water is a limited, shared resource. Being able to water plants more selectively and precisely on the basis of individual plant requirements and the water available, would reduce water wastage. Frost detection and pest detection are other examples of applications in agriculture that would benefit from proactive approaches.

Our research has been performed in the context of the EU-funded R&D project PLANTS (Cassells et al., 2006). In this paper, we describe an ontology-driven architecture for developing hybrid systems that can be used in precision agriculture applications. A hybrid system consists of various entities including software components, hardware components (sensors, actuators and controllers), datastores (knowledge base, raw data, metadata), biological elements (plants) and environmental context. By positioning sensors around particular plants, the delivered technology is capable of reacting (via actuators) to stimuli (perceived via sensor networks), aiming to maintain an optimised plant state and support efficient plant growth.

The remainder of this paper is organised as follows. Section 2 presents the background concepts related to hybrid systems. We explain the concept of mixed societies of communicating plants and artefacts, we identify the analogy with a context management process at a high-level system view and we give an illustrative example of the interactions between the elements of a mixed society. Section 3 discusses the PLANTS ontology requirements analysis and design. The main categories of knowledge that must be represented in the ontology are specified. The ontology is organised hierarchically into the PLANTS Core Ontology (encodes general knowledge) and the PLANTS Higher Ontology (encodes application-specific knowledge). Section 4 presents the core modules of the distributed management system, namely PLANTS system. First, the architecture is outlined and then the basic modules are described. A detailed example application from the precision agriculture domain is given in Section 5. Related work is discussed in Section 6 and finally the conclusions and our future work plans are given in Section 7.

2 Background

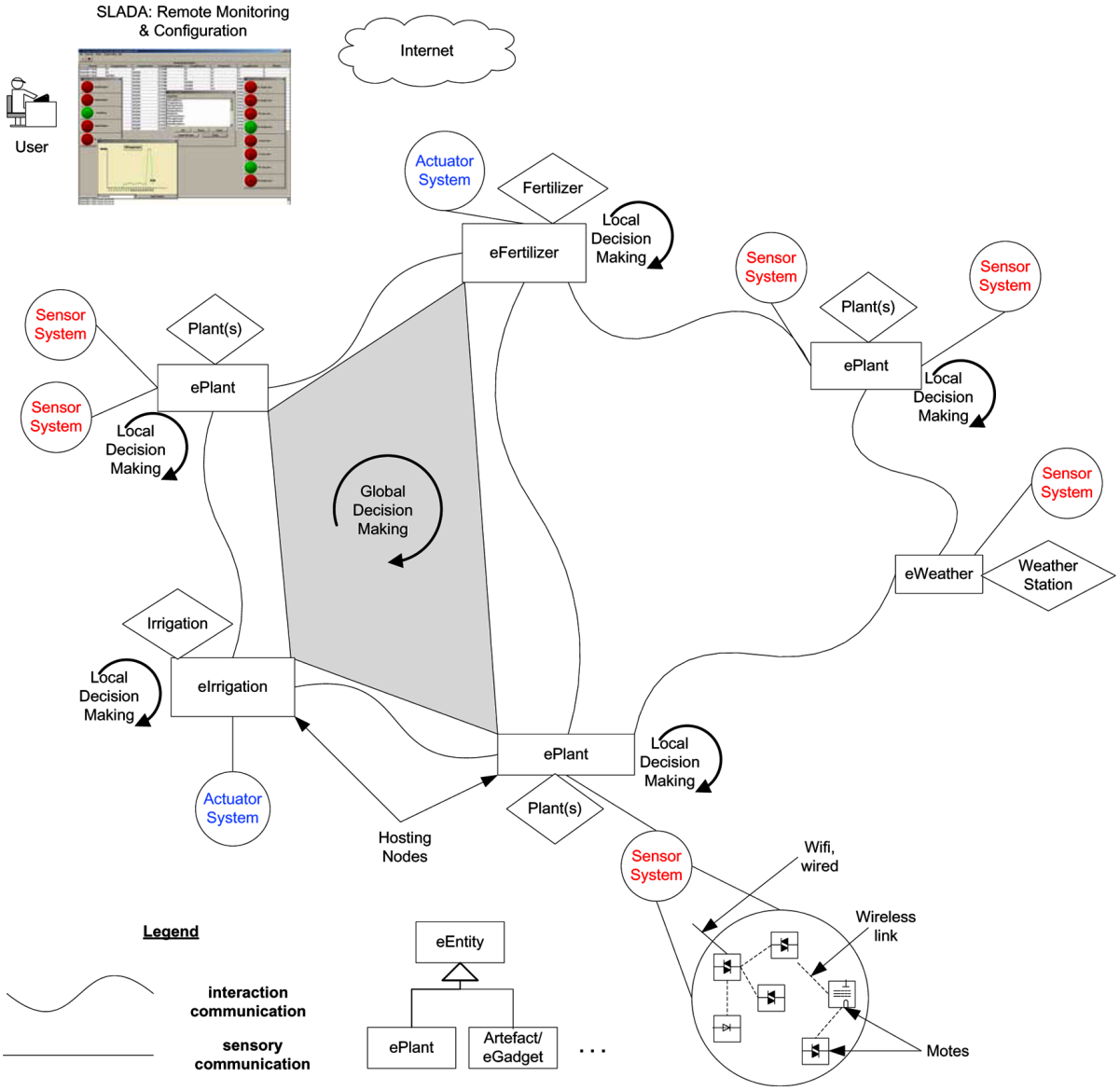
2.1 Mixed societies of communicating plants and artefacts

It is well known that plants show visible signs (e.g., leaf substance and colour), measurable signs (e.g., heat and chlorophyll irradiance) and emit volatiles (e.g., stress chemicals) that can be measured. These are all elements of plant language that we can perceive through technology (Goumopoulos et al., 2004). Accordingly, the concept of 'communicating plant' emerges, where a plant, augmented with sensors and actuators, provides a report on the specific plant status and directly addresses the plant's requirements.

The communicating plant fits then well within the vision of Ubiquitous Computing (Weiser, 1991) where the virtual (computing) space will be seamlessly integrated with our physical environment. By giving plants a 'digital self', plants can communicate their properties in digital space. Furthermore, regarding plants as virtual 'components', which can communicate with other artefacts in the digital space, we can shape mixed societies of them. From an engineering perspective, a mixed society of communicating plants and artefacts can be regarded as a multi-layered, hierarchical distributed system, which will globally manage the resources of the society, its function(s) and its interaction with the environment.

In Figure 1, the basic elements of such a society are shown. The components *ePlant*, *eIrrigation*, *eFertiliser*, etc., depicted in the scheme represent in the digital space the corresponding physical or tangible entities (hosting nodes). These components are generalised by the term *eEntity*. Thus, in principle, an entity in the physical space becomes an eEntity in the virtual space by the superimposition of a technological layer consisting of hardware/software modules that enable it to sense the environment (using sensors), act upon it (using actuators), store and process data locally and communicate with other eEntities or software programmes (e.g., tools) via wireless or internet communication links.

Figure 1 Basic elements identified in mixed societies of communicating plants and artefacts (see online version for colours)



An *ePlant* component, in particular, may represent the digital self either of a specific plant or a group of plants (a group may be defined in terms of a specific plant species or in terms of plant vicinity, a number of plants in a geographical region) and is responsible for the back-end computation with respect to the sensor network computing. Through a software layer, ePlant communicates with the sensor network, implements a decision-making scheme for assessing plant states and alarms and handles the interaction with other eEntities.

eEntities that represent domain-specific objects with the capabilities of information processing and exchange are also called *Artefacts* or *eGadgets*. These artefacts have the capability of communicating with other artefacts based on local networks, as well as accessing or exchanging information at a distance via global networks. In our case, artefacts may represent expressive devices (speakers, displays, etc.), resource-providing devices (e.g., lamps, irrigation/fertilisation/shading system) or any other everyday object (e.g., cell phone, camera).

Sensor systems range from standalone sensor devices to wireless sensor networks monitoring micro-climates in a crop field. Standalone sensor devices may be shared among a number of ePlants (e.g., owing to cost constraints) so that the context needs to be determined. Wireless sensor networks are based on hardware platforms like Mica2, Mica2Dot and Intel systems, called *motes* (Bellis et al., 2005). On the other hand, the *actuator systems* will allow the plant to influence the environment that it resides in.

The communication in the distributed system is divided into two levels: *sensory communication*, which refers to the communication between an eEntity hosting node and its sensor/actuator systems, and the *interaction communication*, which refers to the communication between the eEntities. In that way, we separate the interaction services from the context of application.

The interaction of artefacts and ePlants entails the triggering of autonomous *local decision-making* and *global decision-making* procedures. The term local decision-making is used when a node can reach a decision

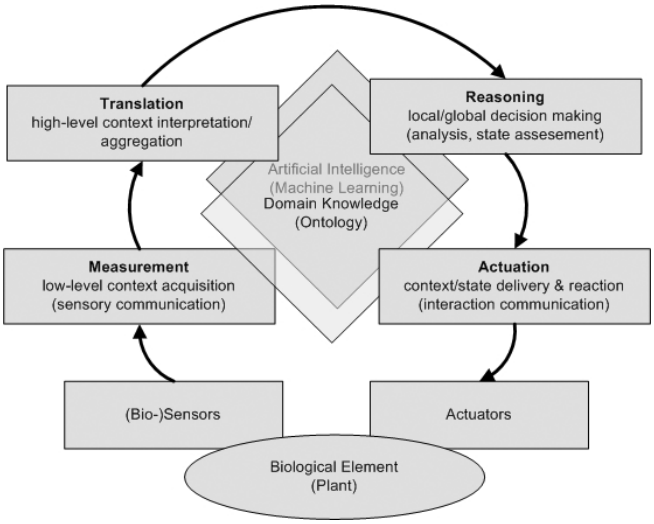
based on local knowledge, i.e., does not need to interact with other nodes in the distributed system. This is in contrast to global decision-making where the nodes of the distributed system communicate and exchange data that can be used to perceive a global state and trigger a global decision. For example, upon determining the local state of a plant, a decision may be required for an action to be followed. In the case of an artefact (e.g., a lamp or valve), the local decision-making (or resource management) mechanism resolves conflicts when multiple ePlants request a common resource (e.g., light or water). Distributed mechanisms can also be considered to alleviate similar situations, when ePlants and related artefacts are coordinated for detecting/maintaining a global state in the context of a group of distributed nodes.

2.2 Context management process

At a high level, the process performed by the distributed system discussed in Figure 1 can be viewed as a plant/environmental context management process. We model this process as a *measurement-translation-reasoning-actuation* control cycle (Figure 2). A mechanism for low-level context acquisition, which reads plant/environmental signals from sensors, starts this cycle. Signals range from selected electromagnetic wavelengths through to volatile organic molecules. This information is probably not initially in a format that can be used by the system to make decisions or reach a conclusion. In a second phase, the signals are interpreted and high-level context information is derived. For example, temperature and soil moisture sensors return an analogue signal (voltage value), which must be then converted, after a calibration phase, to a digital format. This signal conditioning phase is usually performed within the motes using specialised Analogue to Digital conversion circuitry, implemented in such a way as to optimise both network data throughput and system battery life, by avoiding unnecessary send/receive messages. A mote typically contains a microcontroller equipped with in-system memory and can be programmed to handle analogue to digital conversion of sensor data. More details on this specialised process may be found in Bellis et al. (2005).

Aggregation of context is also possible, meaning that semantically richer information may be derived based on the fusion of several measurements that come from different homogeneous or heterogeneous sensors. The determination of photo-oxidative stress, for example, requires monitoring of chlorophyll fluorescence in conjunction with ambient light level signals so as to adjust supplementary light levels. As another example, the determination of water stress requires the monitoring of a plant's leaf temperature, the ambient temperature and the soil moisture content. The aggregation of context is an operation that is performed at the higher levels of the system, usually at the hosting node.

Figure 2 Plant/environmental context management process



Having acquired the necessary context, we are in a position to assess the state of the plant and decide an appropriate response activation. Adopting the definition from Artificial Intelligence, a state is a logical proposition defined over a set of context measurements (Russell and Norvig, 2003). This state assessment will be based on a set of rules, which are either obtained as part of a time-consuming and labour-intensive manual process, or as part of a more advanced scheme by utilising learning capabilities within the system. The low (sensor) and high (fused) level data, their interpretation and the decision-making rules are encoded in an ontology.

The reaction may be as simple as to turn on a light, or to send a message to the user, or a composite one such as a request to add water directly to the soil in the pot in case of drought stress, or as spraying mist in case of heat stress. This means that the system has to differentiate between the two kinds of water stress and evaluate the appropriate response. Such a decision may be based on local context or may require context from external sources as well, e.g., a weather station supporting prediction of plant disease spreading.

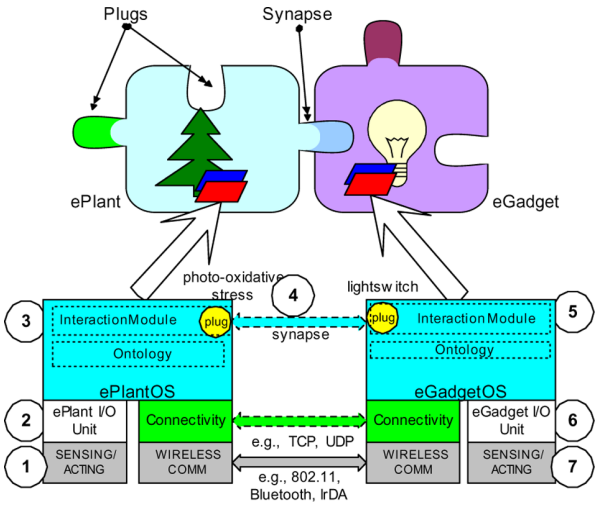
2.3 An interaction example

In our approach, an application is realised through the cooperation of nodes of the distributed system in the form of established logical communication links between services and capabilities offered by the artefacts and the states and behaviours inferred from the plants (in each case services/states are provided through access points called plugs). The plug/synapse model provides a conceptual abstraction that allows the user to describe mixed society applications (Drossos et al., 2007). To achieve collective desired functionality, one forms synapses by associating compatible plugs, thus composing applications using eGadgets and ePlants as components. The use of high-level

abstractions, for expressing such associations, allows the flexible configuration and reconfiguration of mixed society applications with the use of appropriate editing tools.

In Figure 3, we depict a simple mixed society of an ePlant associated with an eGadget (eLamp), so that when the chlorophyll fluorescence signal of the plant is below a certain level, implying a photo-oxidative stress situation, the light of the lamp must be turned on to a specific level of luminosity, until chlorophyll fluorescence signals indicate optimal photosynthetic efficiency again. A synapse has been formed between the 'photo-oxidative stress' plug of the ePlant and the 'lightswitch' plug of the eLamp. The interaction module that implements the plug/synapse model is compatible between the two components and thus their interaction is feasible.

Figure 3 An example of ePlant/eGadget interaction (see online version for colours)



In step 1, the biosensor/bioactuator network transforms selected plant (chlorophyll fluorescence) or other environmental (ambient light intensity) signals into digital signals. In step 2, ePlant's I/O Unit reads the digital signals (sensory communication), which will then be interpreted to a high-level unit of information, for instance, to an aggregated composite signal, and this information is transferred to the PLANTS system, which acts as a middleware (ePlantOS). In step 3, the context received by the middleware is applied to the rules encoded in the ontology so that a state of the plant is determined. Then, the decision for an action may come in the form of a command or a request for a service. The ontology of the ePlantOS may specify, for example, the luminosity of the requested light, based on the plant species at hand. A new context type that is not specified in the current ontology will trigger an update of the ontology before it can be used into a decision-making rule. In step 4, the system passes the information to the connected eGadget through the established logical channel (synapse). The connectivity and wireless communication layers implement the lower layers of the network stack. Finally, in steps 5–7, the middleware of the eGadget receives the information and acts upon the eGadget by using the eGadget I/O unit that in turn activates an actuator through the sensor/actuator network.

3 PLANTS ontology

3.1 Requirements

The distributed management system that governs plant/artefact societies is closely bound to knowledge that must be presented and managed. The knowledge that must be represented can be divided into the following categories:

I *Mixed societies conceptualisation*

According to Uschold and Gruninger (1996), an ontology is a tool that can conceptualise a world view by capturing general knowledge and providing basic notions and concepts for basic terms. In the same way, we turn to ontologies to conceptualise the terms of mixed societies as to enable the communication among ePlants and eGadgets. For a feasible communication among ePlants and eGadgets, a common language and a common perception of their world is required. The terms of this common language as well as the basic concepts of mixed societies are described in the PLANTS ontology. Therefore, the PLANTS ontology contains the description of the semantics of the basic terms, such as: *eGadget*, *ePlant*, *Plug*, *Synapse*, *Sensor*, *Actuator* and *bioGadgetWorld* (denotes a mixed society), and the definition of the relations among them.

II *Plant characterisation*

One of the main objectives is to study plant eco-systems to understand sensing and communication mechanisms that will be used as models for specification of plant–artefact interfacing mechanisms. The knowledge emerged from these studies can be divided into various categories.

a *Knowledge regarding the plant itself*. In this category, knowledge such as the name and the species of the plant is described. Additionally, this category contains knowledge about the growth and the development stages of plants.

b *Knowledge regarding plant parameters being monitored by sensors*. This category contains information about the available sensors that can monitor the plant parameters as well as relative knowledge like the range of values, the threshold values and the interpretation of the aforementioned values.

c *Knowledge regarding the implied state of plants*. This category contains information relevant to the plant stressing and sensing mechanisms and the signals that plants perceive and send to the environment. Specifically, the possible states of a plant implied by its parameters monitored by sensors are part of this knowledge. For example, the representation of stresses, like the water stress, diseases and symptoms, belongs to this category of knowledge.

d *Knowledge regarding environmental parameters*. The knowledge about the environmental parameters that we can measure and monitor is essential to define the state of a plant. For example, parameters like the temperature, the humidity, CO_2, the light and the soil moisture play a major role. The description of these parameters, their range and threshold values are also represented.

III *Sensor and actuator systems characterisation*

The sensors and the actuators play a crucial role in precision agriculture applications. In particular, the use of sensors requires a description that specifies their type, the parameter they measure, the range of their values as well as their sensitivity and accuracy. Because we may use different sensors for the same plant (or environmental) parameter, we design an abstract structure about sensors and using an intermediate interpretation connect their outputs to specific parameters. This intermediate interpretation allows the interfacing with both existing and future sensor systems when these become available.

IV *Rules for decision-making*

This category refers to the knowledge that supports the decision-making process. This knowledge is represented as a set of rules, which are used for various decisions. We mention below some of them.

a *Sensors*. A design principle of our approach is to abstract the system from the real sensors that we are using. Specifically, if we assume that a sensor measuring the temperature provides to the system a value within a specific range, we can use any available sensor and with a set of rules we can calculate the value that the system recognises from the sensor's outputs.

b *Diagnosis of the plant state*. There is a need for a set of rules that will take into account both plant and environmental parameters and the description of a plant to diagnose a plant's state.

c *Local decision-making*. The local decision-making is based on a plant's state and its description and determines the possible actions of an ePlant, like the request for a resource. Correspondingly, similar rules will support the decision-making of an eGadget to select a policy for the resource management.

d *Global decision-making process*. These rules are similar to the rules used by the local decision-making process that defines the reaction of an eEntity. The main difference is that the rules of the global decision-making process have to take into account the states of other eEntities and their possible reactions. The rules that are activated in the global decision-making process are related mainly to user-defined policies.

3.2 Design

The PLANTS ontology is designed so that it enables the semantically meaningful interaction between plants and artefacts via the conceptualisation of plants domain knowledge. The PLANTS ontology thus represents the necessary knowledge to meet system requirements and support its functionalities.

Since the interoperability among system components (ePlants/eGadgets) is based on their ontologies, the existence of different ontologies could result in inefficient interoperability. An awkward solution to this issue could be the merging of all existing ontologies into a global one that would inevitably result into a very large knowledge base. This solution is undesirable for two reasons: first, it does not respect the limited memory capabilities of the artefacts, and second, it requires the continuous synchronisation of all eEntities. Another solution could be the use of a client/server model where all eEntity ontologies are stored centrally and each eEntity can have access to it. This solution conflicts, however, with the autonomous nature of system components.

We designed the PLANTS ontology, keeping in mind the design criteria proposed by Gruber (1993) for efficient developing of ontologies:

- *Maximum monotonic extensibility*: new general or specialised terms can be included in the ontology in such a way that it does not require the revision of existing definitions.
- *Clarity*: terms that are not similar (common-sense terms vs. specialised domain ontologies) are placed in different taxonomies.

According to this, the PLANTS ontology is divided into the following two different layers: the *PLANTS Core Ontology* (*PLANTS-CO*) and the *PLANTS Higher Ontology* (*PLANTS-HO*). The solution that we propose allows each eEntity to have a different ontology with the condition that all ontologies will be based on a common vocabulary. Specifically, the PLANTS Core Ontology will contain the common vocabulary, and the PLANTS Higher Ontology will represent eEntity-specific knowledge using concepts represented into PLANTS-CO.

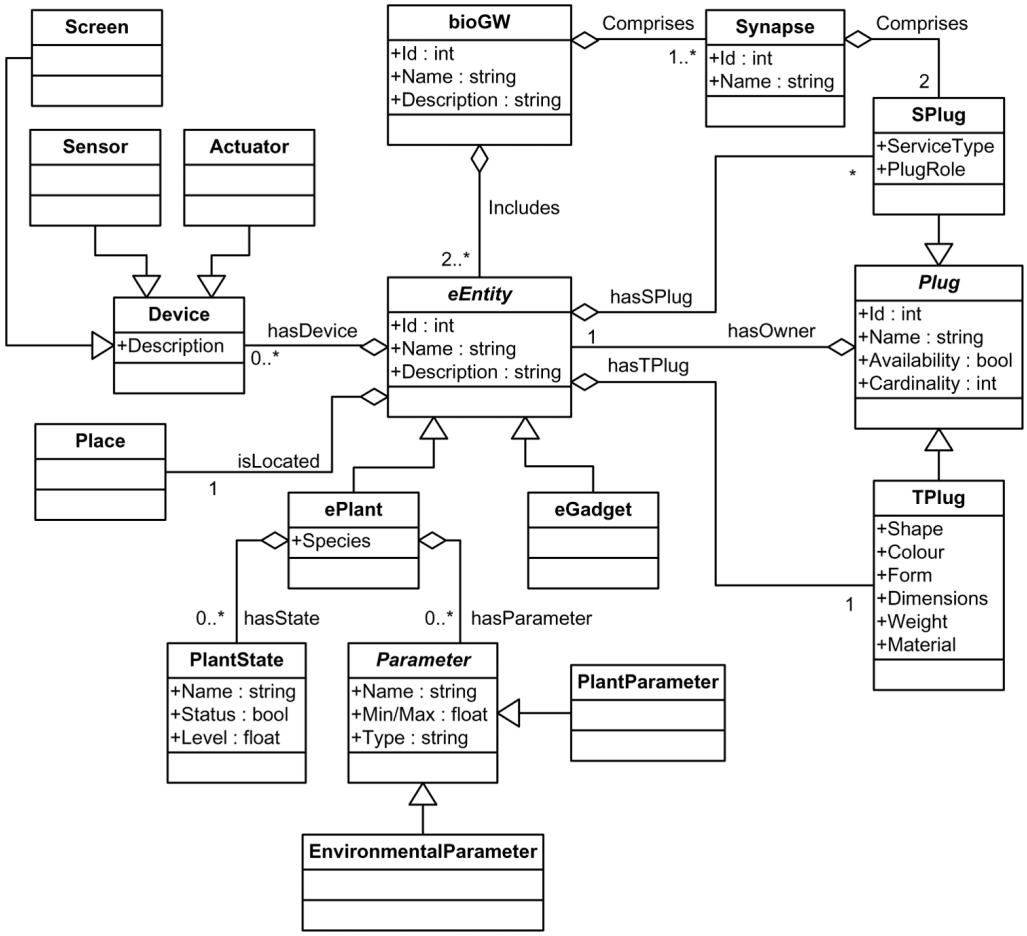
Cranefield and Purvis (1999) proposed using a subset of the Unified Modelling Language (UML) (Fowler and Scott, 1999) together with its associated Object Constraint Language (OCL) for representing ontologies. In the ontology modelling diagrams presented in this paper, we assume the following conventions:

- ontology classification hierarchy will be expressed by UML generalisation, represented by lines with hollow arrow heads pointing to the super class
- class properties will be presented by UML aggregation, an association with a diamond at the aggregate end of the link.

3.3 PLANTS Core Ontology (PLANTS-CO)

The PLANTS-CO represents the common language among eEntities, thus it has to contain all the semantic description of the basic concepts of mixed societies and their inter-relations. All eEntities share the same PLANTS-CO. A key requirement of the PLANTS-CO is to contain only the indispensable semantics for the interoperability of eEntities to be very small and even eEntities with limited memory capacity may store it. The PLANTS-CO is static and cannot be changed either by an eEntity manufacturer or by a user. In Figure 4, a UML representation of the PLANTS-CO basic classes and their inter-relations is shown.

Figure 4 PLANTS-CO: basic classes and their inter-relations



In the PLANTS ontology, an artefact will be represented as a class, *eGadget*, which has a number of properties. These properties are divided into two categories: the first one contains the physical properties, which describe the eGadget as a tangible object; the second one contains the digital properties, which manifest the digital self of the eGadget and the plugs owned by the eGadget and expose its services.

The *ePlant* concept will be represented as another class with its properties. Additionally, the PLANTS ontology will describe the digital properties of ePlant, such as its plugs.

The notion of *Plug* will be represented in the PLANTS ontology as another class. From the user's perspective, plugs make visible the entities' properties, capabilities and services to people and to other entities. Plug *cardinality* is the maximum number of synapses that the plug can participate-in and plug *availability* denotes whether the plug can participate in another synapse. Plug-availability is determined based on plug-cardinality and the number of existing connections a plug has. The Plug class is divided into two disjoint subclasses: the *TPlug* and the *SPlug*. The TPlug describes the physical properties of the object that is used as an eGadget or the plant that is used as an ePlant and lists all the eEntity's Plugs and Synapses; there is a cardinality restriction that an eEntity must have exactly one TPlug. On the other hand, an SPlug represents the eEntity properties, capabilities and services; an eEntity has exactly one SPlug per service offered and thus can have an arbitrary number of SPlugs. SPlugs are characterised by their *service type*, which describes the type of the service offered (e.g., 'light', 'temperature', etc.).

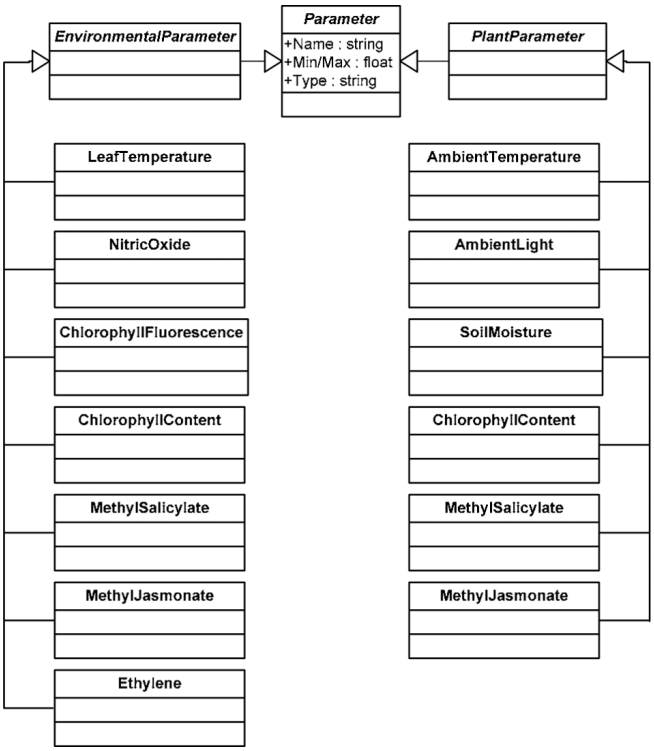
The *Synapse* class represents a synapse among two plugs (for the plug/synapse logic see the example in Section 2.3). A synapse may only appear among two SPlugs. The cardinality restriction that stands for a synapse is that it represents the connection among only two SPlugs. For each SPlug that participates in a synapse, a special role will be declared through PLANTS ontology.

With the *bioGW* (bioGadgetWorld) class, the PLANTS ontology describes the mixed societies that are created. The bioGW properties represent the eGadgets and the ePlants contained in a bioGW and the synapses that compose it. There are two cardinality constraints: first, a bioGW must contain at least two eEntities, and second, at least a synapse must exist between their plugs.

The sensors and the actuators play a significant role in the targeted application domain, especially with respect to the ePlant concept, thus we incorporate into the PLANTS ontology the classes *Sensor* and *Actuator*. These classes are special cases of the general class *Device*. For instance, a description of sensors defining their type, their measurement range and accuracy is provided.

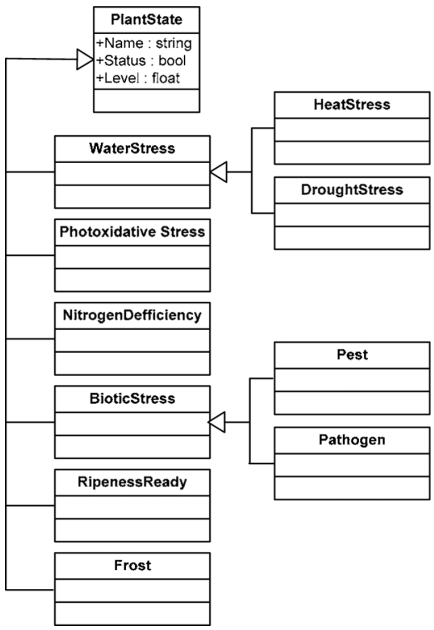
We have added the concepts *Parameter* and *PlantState* and we defined that an ePlant may have parameters and states. The concept Parameter is further classified into two concepts: the *Plants Parameters* and the *Environmental Parameters*. Each Parameter has a number of properties, like minimum value, maximum value, default value, the type of its value (e.g., integer, float, Boolean, etc.). Additionally, there is a relation between a parameter and the sensor that measures it. Both the classes 'Plants Parameters' and 'Environmental Parameters' are divided into subclasses that define specific parameters. In Figure 5, we illustrate examples of these classes.

Figure 5 PLANTS-CO: Subclasses of concept 'Parameter'



To represent the various plant states, the PLANTS-CO contains the concept 'PlantState'; a state may be activated and if it is activated we may use an activation level to further define the state of an ePlant. In Figure 6, we illustrate subclasses of the PlantState.

Figure 6 PLANTS-CO: Subclasses of concept 'PlantState'



3.4 PLANTS Higher Ontology (PLANTS-HO)

The PLANTS-HO represents both the description of an eEntity and its acquired knowledge. These descriptions follow the definitions contained in the PLANTS-CO. Specifically, the knowledge stored into the PLANTS-HO is represented as instances of the classes defined into the PLANTS-CO. For example, the PLANTS-CO contains the definition of the concept SPlug, while the PLANTS-HO contains the description of a specific SPlug represented as an instance of the concept SPlug. Consequently, the PLANTS-HO is not a standalone ontology, as it does not contain the definition of its concepts and their relations.

Since the PLANTS-HO represents the private knowledge of each eEntity, it is different for each eEntity. Contrary to PLANTS-CO, which size is required to be small enough, the size of PLANTS-HO will depend only on eEntity's memory capacity. Apparently, PLANTS-HO is not static and it can change over time without causing complications to plants–artefacts communication. To reflect the fact that PLANTS-HO contains both static information about the eEntity and dynamic information emerged from its knowledge and use, PLANTS-HO is divided into the *PLANTS-HO-static* and the *PLANTS-HO-volatile*.

The PLANTS-HO-static represents the description of an eEntity containing information about the eEntity plugs, its sensors and actuators, its parameters and its states, as well as its physical characteristics. For example, the PLANTS-HO-static of the 'eStrawberry' ePlant contains the knowledge about the species of ePlant (e.g., 'strawberry'), its parameters (e.g., 'AvgTemp' and 'AmbientAvgTemp'), its state (e.g., 'DroughtStress') and its SPlug (e.g., 'Need_Irrigation').

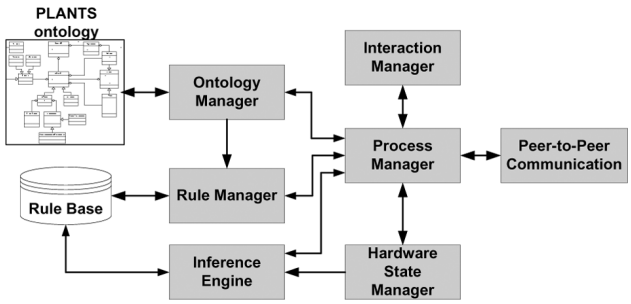
On the other hand, the PLANTS-HO-volatile of an eEntity contains information derived from its use and acquired knowledge. Specifically, it describes the synapses

to which the eEntity plugs are connected to, the applications to which it takes part, as well as information about the capabilities of other eEntities that has acquainted through communication. The PLANTS-HO-volatile is updated during the eEntity's various activities, like the establishment of a new synapse.

4 PLANTS system architecture

The deployment of mixed societies is supported by the PLANTS system whose architecture is composed of a few basic modules that are briefly explained in this section. The outline of the system architecture is shown in Figure 7.

Figure 7 System architecture outline



The *Process Manager* is the coordinator module of the system and the main function of this module is to monitor and execute the reaction rules defined by the supported applications. These rules define how and when the infrastructure should react to changes in the environment. The *Hardware State Manager* maintains a repository of the hardware environment (sensors/actuators) inside PLANTS system reflecting at each particular moment the state of the hardware. The *Interaction module* implements the interaction scheme between plants and artefacts, in the form of the Plug/Synapse model. The *Peer-to-Peer Communication Module* is responsible for application-level communication and interaction between the various eEntity nodes. This module implements algorithms and protocols for wireless, connectionless communication (using the 802.11 b/g protocol) as well as mechanisms for internal diffusion of the information exchanged.

The *Ontology Manager* module has been defined for the manipulation of the knowledge represented into the PLANTS ontology and to provide the other modules of the system with parts of this knowledge with a level of abstraction. This means that only the Ontology Manager needs to understand PLANTS Ontology and be able to use it; all other system modules can query the Ontology Manager (through the Process Manager) for the information that they need without any knowledge about the ontology language and its structure. Therefore, any changes that may be done to the PLANTS Ontology affect only the Ontology Manager and the rest of the system is isolated from them. The Ontology Manager provides methods that query PLANTS ontology for the definition of specific concepts, and for the existing instances of specific concepts, like the environmental parameters. The Ontology Manager is also responsible for the addition of knowledge into the PLANTS ontology. For example, when a new Synapse is established between two Plugs, the relevant knowledge must be added to the PLANTS ontology.

The *Rule Manager* is the mechanism that manages the rule base of an eEntity, and its basic functionality is to provide to the other modules the rules that define an eEntity's logical operation. The basic operations of the Rule Manager are to query about the rules of an eEntity and to update them. For the initialisation of the decision-making process apart from the rules, the initial facts are necessary that represent low-level environmental/plant context sensor measurements or inferred plant states. In this respect, the Rule Manager is also responsible for the creation of the initial facts of a specific eEntity. For example, an initial fact is the definition of the existence of the eEntity with its specific parameters, states and SPlugs (reactions) that participate in its rules. To create this initial fact, the Rule Manager needs to have knowledge about the eEntity that is stored in the PLANTS-HO-static. For this, it queries the Ontology Manager through the Process Manager for any information that it needs, like what are the parameters, the states and the SPlugs of the eEntity.

The *Inference Engine* is the module of the system architecture that supports the decision-making process. This module exploits the Jess rule engine (Java Expert System Shell) (http://herzberg.ca.sandia.gov/jess/). The execution of this module is started based on the initial facts (defined by the Rule Manager from knowledge emerged from the PLANTS ontology through the Ontology Manager) and the rules stored in the rule base. The Inference Engine module is informed for all the changes of parameters values from sensor measurements through the Hardware State Manager. When the Inference Engine is informed for such a change, it runs all its rules. When a rule is activated, the Inference Engine informs for the activation of this rule and for the knowledge that is inferred the Process Manager that is responsible to transfer this knowledge to any module that needs it.

Regarding the *Rule Base,* currently the rules are stored in a file in Jess format and the concepts that appear are emerged from the PLANTS ontology. This is an approach of building rules on top of ontologies. Additionally, we have also stored the rules in a simple XML-like format that we have designed targeting to a simpler manipulation of rules from various tools. Our target is to use the Semantic Web Rule Language (http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/) for the representation of the rules, which is an other approach that specifies an ontology for rule syntax.

The specification of the PLANTS ontology was performed using the Protégé ontology development tool (http://protege.stanford.edu/) based on the OWL standard language.

5 Example application

The example application described in this section is composed of a strawberry plant where the plant is controlling irrigation and supplementary light. Irrigation is applied according to the specific requirements of the plants in different parts of the crop array, thus illustrating the precision delivery of agricultural inputs.

5.1 Plant/environmental parameters

The plant/environmental parameters explored for the application development are: Plants' leaf Temperature (PT), Chlorophyll Fluorescence (CF), Ambient Temperature (AT), Ambient Light (AL) and Soil Moisture (SM). For each signal, a different type of sensor is required. Table 1 summarises the signals and the corresponding sensors used as well as the associated knowledge that will be stored in the PLANTS ontology for supporting the monitoring and decision-making process.

Table 1 Plant/environmental signals and sensors

Signal	*Measuring sensor*	*State assessment*	*Possible actions*
CF	PAM meter[1]	Photo-oxidative stress; photosynthetic efficiency	Light control; estimate/adapt threshold values for providing input resources
PT	Thermistor	Drought stress; heat stress	Irrigation/misting
SM	Probe EC-10[2]	Drought stress	Irrigation
AT	Thermistor		
AL	PAR meter[3]	Photo-oxidative stress	Light control

[1]Junior PAM, Gademann Instruments: http://www.gademann.com/

[2]ECHO probe model EC-10: http://www.ech2o.com/specs.html

[3]Skye SKP215 Quantum Sensor: http://www.alliance-technologies.net

Heat stress can occur independently of water stress when the ambient environmental temperature gets very high and plant transpiration cannot maintain leaf cooling. Therefore, if the plant has adequate water (determined by the SM probe) but the plant temperature is high this means that it is heat-stressed and requires misting to cool it. However, if the temperature is high and the moisture content low, then pot irrigation is required. The CF and AL parameters are used to determine photo-oxidative stress and adjust supplementary light.

5.2 Prototype setup and evaluation

The prototype setup consists of an array of 96 plants placed in a glasshouse, arranged in an array of 12 by 8. The setup consists of four different zones: Left-Edge (LE), Right-Edge (RE), Left-Centre (LC), Right-Centre (RC) and also one zone specified for misting, which coincides with the RC zone. The setup integrates the thermistors and soil moisture probes into one system that can irrigate when required and also determine when to stop the irrigation. This deployment takes into account differences in the location of the plants in the overall area and will allow for independent irrigation of edge or centre zone plants as required. Each zone can be controlled using individual solenoids. Misting can be applied only to the RC owing to infrastructure limitations.

A total of ten motes are required to implement the above prototype: eight modules are used for connecting the various sensors, each one 'supervising' the sensors in the neighbourhood of an array of three by four plants, one module is sensorless and is used as a communication relay with the hosting node, and one module is used for controlling the irrigation system. The sensor nodes are manually placed however the mapping to the zones is administered at a higher level in the hosting node (ePlant), as part of its description. For energy-efficiency and power consumption considerations, the sensor nodes are reporting data once per 5 mins. The data collected by the sensor nodes is gathered by the hosting node, for local processing and logging. Interaction then is possible between the hosting node and other devices for managing the delivery of agricultural input according to local or global decision-making schemes.

The application business logic is expressed upon a set of plant parameters, plant states and actions to be performed. Table 2 illustrates such variables defined in the ontology of the application.

Table 2 Application business logic variables

Parameters	*States*	*Action requests*
AmbientAvgTemp	"Z"DroughtStress	"Z"NeedIrrigation
"Z"AvgTemp	"Z"HeatStress	"Z"NeedMisting
"Z"AvgMoisture		

The "Z" prefix in the name of a variable is substituted by one of the possible zone names of the crop array (LE, RE, LC, RC). For the NeedMisting variable, the prefix can be omitted since there is only one zone specified for misting. Two additional parameters must be defined for the prototype to be properly working; the duration of irrigation/misting and an idle time that specifies the amount of time the rules should be disabled, after the action is performed. This is to allow the ecosystem to absorb the changes. The values used for the application were 1 min and 4 hr, respectively.

The actual logic of the prototype is captured in a set of rules. Table 3 contains the applicable rules for the RC zone. Rules for evaluating the plant states and actions to be performed are shown. *Confidence Factor (CF)* values are also included. CF values in square brackets are defined by the domain-expert, while in curly brackets by the system Inference Engine. The user, for example, can specify a policy where actions with confidence below 50% should not be triggered but the user should be notified.

Certainty factors may apply both to facts and to rules, or rather to the conclusion(s) of rules. Conditions for rules are formed by the logical '*and*' and '*or*' of a number of facts. The certainty factors associated with each condition are combined to produce a certainty factor for the whole condition. For two conditions *P1* and *P2,* it holds that: *CF(P1 and P2) = min(CF(P1), CF(P2))* and *CF(P1 or P2) = max(CF(P1), CF(P2))*. The combined CF of the condition is then multiplied by the CF of the rule to get the CF of the conclusion.

Table 3 Application rules with confidence factors shown

Rule	**Body**
RCDrought Stress *[CF=0.8]*	IF RCAvgTemp−AmbientAvgTemp>0.75°C *[CF=0.9]* THEN RCDroughtStress ← TRUE ELSE RCDroughtStress ← FALSE *{CF=0.72}*
RCHeat Stress *[CF=0.9]*	IF RCDroughtStress *{CF=0.72}* AND RCAvgMoisture>60% *[CF=0.9]* *{CF=min(0.72, 0.9)=0.72}* THEN RCHeatStress ← TRUE ELSE RCHeatStress ← FALSE *{CF=0.65}*
RCNeed Irrigation *[CF=1]*	IF RCDroughtStress *{CF=0.72 }* AND NOT RCHeatStress *{CF=0.65 }* *{CF=min(0.72, 0.65)=0.65}* THEN RCNeedIrrigation ← TRUE ELSE RCNeedIrrigation ← FALSE *{CF=0.65}*
Need Misting *[CF=1]*	IF RCDroughtStress *{CF=0.72 }* AND RCHeatStress *{CF=0.65}* *{CF=min(0.72, 0.65)=0.65}* THEN NeedMisting ← TRUE ELSE NeedMisting ← FALSE *{CF=0.65}*

On the agronomic part of the experiment, the instrumentation of the strawberry field with the wireless sensor network and the plant-driven irrigation leads to a notable reduction in water consumption (15–20%) with respect to traditional agricultural practices involving user-defined timed irrigation based on rules of thumb. The latter was applied in a parallel setup for the same growing period (early development stage) of the crop. The deployment of smart water management on a large farming scale is extremely important, given the irrigation needs of the agricultural sector (irrigation uses up to 80% of total water in some regions) and the decreasing availability of water for irrigation.

The use of the PLANTS ontology for the organisation of concepts and definition of operational semantics has been successfully tested and revealed the advantages of this approach. Using PLANTS ontology for defining application business logic emphasises system flexibility and adaptability. In that sense, our system architecture can be regarded as a reflective architecture that can be adapted dynamically to new requirements.

By specifying the rules structure and semantics in an ontology that defines various parameter/states types as well as the arguments that the rules are based on, we can use the ontology to verify rules validity. This also makes easier the inclusion of environmental/plant context parameters in rules, since we know the rules structure and the kinds of values different arguments can take.

Finally, using ontological descriptions allowed us to have heterogeneous entities interoperate and interact with one another in a meaningful way dictated by the domain under consideration.

6 Related work

Attempts to use environmental sensor networks to improve crop cultivation by monitoring and reporting on the status of the field are reported by Burrell et al. (2004), Fukatsu and Hirafuji (2005) and Zhang et al. (2004). These approaches provide decision-support to the user who responds by providing the required treatment. This is in contrast to our plant-driven distributed management system that imposes a proactive-computing model for the crop treatment.

The practical issues of building a Ubiquitous Computing application, called PlantCare, that takes care of houseplants using a sensor network and a mobile robot are investigated by LaMarca et al. (2002). The emphasis is given on discussing technical challenges encountered during the deployment of the application. Our approach, in contrast, emphasises the development of an architecture that views plants and associated computation as an integral part and allows the interaction of plants and artefacts in the form of synergistic and scaleable mixed societies. An ontology-based conceptual model is defined for composing applications, which ensures a balanced behaviour both to ambient nature applications where interactions through high-level concepts and user empowerment is the focus, and agricultural nature applications where the integration of a large number of plant

and environmental sensors and the complexity of the communication and the decision-making processes are the focal points.

In the biology, botany, organic computing and bioinformatics domains, there are activities on building ontologies that partially address principles of PLANTS (The Plant Ontology Consortium, http://www.plantontology.org; Sequence Ontology, http://song.sourceforge.net; Gene Ontology Consortium, http://www.geneontology.org). These activities aim to develop and share structured controlled vocabularies for plant-specific knowledge domains like plant anatomy, temporal stages, genes and biological sequences. Central to our approach is the use of an ontology, which provides not just a conceptual description of the domain knowledge, but furthermore the use of rules and constraints (axioms) in operational representation forms allow the use of the ontology for reasoning providing inferential and validation mechanisms. The reasoning is based on the definition of the ontology, which may use simple description logic or user-defined reasoning using first-order logic.

7 Conclusion and future work

We have been involved with a facet of precision agriculture that concentrates on plant-driven crop management. By monitoring soil, crop and climate in a field and providing a decision-support system, it is possible to deliver treatments, such as irrigation, fertiliser and pesticide application, for specific parts of a field in real time and proactively. In this context, we have presented in this paper an ontology-driven framework for developing precision agriculture applications.

Moving our research towards to a more autonomous system with self-adaptation and self-learning characteristics, we have been exploring ways of incorporating learning capabilities in the system. Machine-learning algorithms (Mitchell, 1997) can be used for inducing new rules by analysing logged data sets to determine accurately significant thresholds of plant-based parameters and for extracting new knowledge and extending the PLANTS ontology. By providing intelligent decision-making, we can replace the typical, explicitly coded actions to situations and conditions (which can only prescribe a fixed set of predicates) with a multi-level and more knowledge-intensive decision-making framework coupled with reasoning under uncertainty and machine-learning techniques. Both supervised (experimentation-driven or user-mediation) and non-supervised learning algorithms are needed for realising the self-regulation properties of the system that goes beyond the usual distinction of closed vs. open adaptive systems.

Different ontology-based systems may use different terms to describe the same concept and may follow different policies to perform the same task. For example, the name of plants could be different for different areas or for different languages. As a consequence, applications and services developed for one system often cannot be ported in other systems. One solution is standardisation, though it is often difficult to achieve. Another solution is to develop and discover mappings and relationships between different ontology-based systems, a process called *ontology alignment* (Ehrig, 2007). The ontology alignment can be described as: given two ontologies each describing a set of discrete entities (which can be classes, properties, rules, predicates, or even formulas), find the correspondences, e.g., equivalence or subsumption, holding between these entities. To make our system available on a larger scale and adaptable to other systems developed for different geographical areas with different needs and different environmental issues, we aim to apply ontology alignment to find those elements that have the same intended meaning.

Acknowledgement

Part of the research described in this paper was conducted in the PLANTS project (IST FET Open IST-2001-38900); the authors thank their fellow researchers in the PLANTS consortium for their input and support and the anonymous reviewers for their suggestions for improving this paper.

References

Bellis, S.J., Delaney, K., O'Flynn, B., Barton, J., Razeeb, K.M. and O'Mathuna, C. (2005) 'Development of field programmable modular wireless sensor network nodes for ambient systems', *Computer Communications – Special Issue on Wireless Sensor Networks and Applications*, Vol. 28, No. 13, pp.1531–1544.

Burrell, J., Brooke, T. and Beckwith, R. (2004) 'Vineyard computing: sensor networks in agricultural production', *IEEE Pervasive Computing*, Vol. 3, No. 1, pp.38–45.

Cassells, A., Goumopoulos, C., Morrissey, A. and Tooke, F. (2006) 'New crop of technology reveals plant health', *ICT Results*, http://cordis.europa.eu/ictresults/index.cfm?section=news&Tpl=article&BrowsingType=Features&ID=81342

Cranefield, S. and Purvis, M. (1999) 'UML as an ontology modelling language', *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden, *CEUR Workshop Proceedings*, Vol. 23, pp.46–53.

Drossos, N., Goumopoulos, C. and Kameas, A. (2007) 'A conceptual model and the supporting middleware for composing ubiquitous computing applications', *Journal of Ubiquitous Computing and Intelligence*, Vol. 1, No. 2, pp.174–186.

Ehrig, M. (2007) *Ontology Alignment – Bridging the Semantic Gap*, Springer, New York, NY, USA.

Fowler, M. and Scott, K. (1999) *UML Distilled Second Edition, A Brief Guide to the Standard Object Modeling Language*, Addison Wesley, New York, NY, USA.

Fukatsu, T. and Hirafuji, M. (2005) 'Field monitoring using sensor-nodes with a web server', *Journal of Robotics and Mechatronics*, Vol. 17, No. 2, pp.164–172.

Goumopoulos, C., Christopoulou, E., Drossos, N. and Kameas, A. (2004) 'The PLANTS system: enabling mixed societies of communicating plants and artefacts', *Proc. 2nd European Symposium on Ambient Intelligence (EUSAI 2004)*, 8–10 November, Springer-Verlag, Eindhoven, the Netherlands, LNCS Vol. 3295, pp.184–195.

Gruber, R. (1993) 'A translation approach to portable ontology specification', *Knowledge Acquisition*, Vol. 5, No. 2, pp.199–220.

Koch, B. and Khosla, R. (2003) 'The role of precision agriculture in cropping systems', *Crop Production J.*, Vol. 8, pp.361–381.

LaMarca, A., Brunette, W., Koizumi, D., Lease, M., Sigurdsson, S.B., Sikorski, K., Fox, D. and Borriello, G. (2002) 'PlantCare: an investigation in practical ubiquitous systems', *Proceedings of the 4th International Conference on Ubiquitous Computing*, Goteborg, Sweden, September, Springer-Verlag, LNCS, Vol. 2498, pp.316–332.

Mitchell, T.M. (1997) *Machine Learning*, McGraw-Hill, New York, NY, USA.

Russell, S. and Norvig, P. (2003) *Artificial Intelligence: A Modern Approach*, Pearson Education, Upper Saddle River, NJ, USA.

Uschold, M. and Gruninger, M. (1996) 'Ontologies: principles, methods and applications', *Knowledge Engineering Review*, Vol. 11, No. 2, pp.93–155.

Weiser, M. (1991) 'The computer for the 21st century', *Scientific American*, Vol. 265, No. 3, pp.94–104.

Zhang, W., Kantor, G. and Singh, S. (2004) 'Demo abstract: integrated wireless sensor/actuator networks in an agricultural application', *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, ACM Press, November, Baltimore, Maryland, USA, p.317.

Websites

Gene Ontology Consortium, at http://www.geneontology.org

Jess – the Rule Engine for the JavaTM Platform, at http://herzberg.ca.sandia.gov/jess/

PLANTS FET Open project, Enabling Mixed Societies of Communicating Plants and Artefacts, IST-2001-38900, at http://cordis.europa.eu/fetch?ACTION=D&CALLER=PROJ_IST&QM_EP_RCN_A=66398.

Protégé – A Tool for Ontology Development and Knowledge Acquisition, at http://protege.stanford.edu/

Sequence Ontology, at http://song.sourceforge.net

SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

The Plant Ontology Consortium, at http://www.plantontology.org