TALKING PLANT: INTEGRATING PLANTS BEHAVIOR WITH AMBIENT INTELLIGENCE

Ioannis Calemis, Christos Goumopoulos, and Achilles Kameas

Research Academic Computer Technology Institute, Research Unit 3, Design of Ambient Information Systems Group, Patras, Hellas
{calemis, goumop, kameas}@cti.gr

ABSTRACT

The aim of this work is to bring the Ambient Intelligence (AmI) (1) concept in a new level by introducing it to living organisms – plants. In this paper we provide a concrete scenario where an augmented plant, an ePlant can be incorporated in a ubiquitous computing environment in order to work together with other augmented objects, artifacts, in order to provide to the environment status of its condition. The paper presents the enabling infrastructure and the tools that are used to make such an application, and discusses on how we can make these augmented items to collaborate in order to create mixed societies of plants and artifacts.

INTRODUCTION

The vision of Ambient Intelligence (AmI) (1) implies a seamless environment of computing, advanced networking technology and specific interfaces. In one of its possible implementations, technology becomes embedded in everyday objects such as furniture, clothes, vehicles, roads and smart materials, and people are provided with the tools and the processes that are necessary in order to achieve relaxing interactions with this environment. Computer Technology Institute through an EU funded research project, PLANTS (2), has brought the AmI concept in a new level by introducing it to living organisms – plants. While plants are able to perceive the environment and send signals to it, people aren't able to understand these signals and to fulfill the plants needs. With the introduction of specialized sensors and a ubiquitous communication environment, plants, not only are able to give the world their needs but also to take actions for themselves.

Our research effort adheres to the AMI vision where the virtual (computing) space will be seamlessly integrated with our physical environment. To bring this vision closer to reality several issues need to be addressed and solutions be provided:

(i) software infrastructure for pervasive computing that can support the integration between our physical space and virtual computing space;
(ii) sensors and sensor network that can capture and disseminate context information;
(iii) context-aware applications that use context information to create intelligent artifacts and services;
(iv) embedding computing into physical entities;
(v) tools and user interfaces for supporting ubiquitous computing.

This paper will provide a thorough example on how an augmented plant, an ePlant, can perceive its environment and how, through the use of appropriate middleware and tools, the ePlant can "talk" with other objects (artifacts), providing them the appropriate knowledge to take actions or to give events to the environment through to some interactive Artifact devices. The user can take this knowledge, which is perceived as the plants needs, and act accordingly, thus establishing a human-plant communication channel.

The rest of the paper is organized as follows. Section 2 outlines the conceptual model for composing Ubicomp applications where basic concepts and components are identified and described. The middleware that is used from all augmented Artifact physical entities is presented in Section 3. In section 4 the presentation of the scenario is introduced. Sections 5 and 6 present the artifacts and the tools that are used in the talking plant application. Section 8 gives the implementation of the scenario. Section 7 gives a brief state-of-the-art in the fields involved in this paper. Finally, section 8 concludes this paper.

A CONCEPTUAL MODEL FOR COMPOSING UBICOMP APPLICATIONS

In our approach, an application is realised through the cooperation of applications level components (nodes of a distributed system) in the form of established logical communication links between services and capabilities offered by the artifact 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 ubiquitous applications (3), Kameas and Al (4). To achieve collective desired functionality, one forms synapses by

associating compatible plugs, thus composing applications using Artifacts 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.
The basic definitions encapsulated in our conceptual framework Drossos et al (5) are:

Artifacts: An artifact is a tangible object which bears digitally expressed properties; usually it is an object or device augmented with sensors, actuators, processing, networking unit etc. or a computational device that already has embedded some of the required hardware components. Software applications running on computational devices are also excessively considerd to be artifacts.

ePlants: An ePlant is a kind of an artifact. A Plant is transformed into an 'ePlant' through the superimposition of a technological layer and may represent either a specific plant or a set of plants (a set may be defined in terms of specific plant species or a number of plants in a particular location). The scope of the system enables groups of ePlants to be organized into a large number of nodes, to create a hierarchical structure that evenly distributes the communication load and other resource (power, memory, computation) consumption and also facilitates distributed decision-making. These nodal groupings are considered as 'high-level' ePlants with the hierarchical structure adhering to the philosophy of the software component-engineering paradigm where composite components are synthesized from simpler ones.

eEntity: An eEntity is a generalization of the ePlant and the Artifact concept.

Artifact compositions: Two or more artifacts (simple or composite) can be combined in an artifact composition. Such compositions are the tangible bearers of **UbiComp applications** and are regarded as service compositions**;** their realization can be assisted by end-user tools.

bioGadgetWorld (bioGW): A bioGW is a kind of a UbiComp application. It represents a collection of of artifacts and ePlants that work together to achieve a number of goals.

Properties: Artifacts have properties, which collectively represent their physical characteristics, capabilities and services. A property is modeled as a function that either evaluates an artifact's state variable into a single value or triggers a reaction, typically involving an actuator. Some properties (i.e. physical characteristics, unique identifier) are artifact-specific, while others (i.e. services) may be not. For example, attributes like *color/shape/weight* represent properties that all physical objects possess. The service *light* may be offered by different objects. A property of an artifact composition is called an **emergent** property. All of the artifacts properties are encapsulated in a **property schema** which can be send on request to other artifacts, or tools (e.g. during an artifact discovery).

Functional Schemas: An artifact is modeled in terms of a functional schema: $F = \{f_1, f_2 \dots f_n\}$, where each function f_i gives the value of an observed property i in time t. Functions in a functional schema can be as simple or complex is required to define the property. They may range from single sensor readings to rule-based formulas involving multiple properties, to first-order logic so that we can quantify over sets of artifacts and their properties.

State: The values for all property functions of an artifact at a given time are the state of the artifact. For an artifact *A*, the set

$$P(A) = \{(p_1, p_2 \dots p_n) | p_i = f_i(t)\}$$

represents the state space of the artifact. Each member of the state vector represents a **state variable.** The concept of state is useful for reasoning about how things may change. Restrictions on the value domain of a state variable are then possible.

Transformation: A transformation is a transition from one state to another. A transformation happens either as a result of an internal event (i.e. a change in the state of a sensor) or after a change in the artifact's functional context (as it is propagated through the synapses of the artifact).

Plugs: Plugs are the constructs that we use to represent properties of artifacts in the digital space. Plugs are characterized by their direction and data type. Plugs may be output (O) in case they manifest their corresponding property (e.g. as a provided service), input (I) in case they associate their property with data from other artifacts (e.g. as service consumers), or I/O when both happens. Plugs also have a certain data type, which can be either a semantically primitive one (e.g. integer, boolean, etc.), or a semantically rich one (e.g. image, sound etc.). From the user's perspective, plugs make visible the artifacts' properties, capabilities and services to people and to other artifacts.

Synapses: Synapses are associations between two compatible plugs. In practice, synapses relate the functional schemas of two different artifacts. When a

property of a source artifact changes, the new value is propagated through the synapse to the target artifact. The initial change of value caused by a state transition of the source artifact causes finally a state transition to the target artifact. In that way, synapses are a realization of the functional context of the artifact.
To achieve collective desired functionality, one forms synapses by associating compatible plugs, thus composing applications using artifacts as components. Two levels of plug compatibility exist: Direction and data type compatibility. According to direction compatibility output or I/O plugs can only be connected to input or I/O plugs. According to Data type compatibility, plugs must have the same data type to be connected via a synapse. However, this is a restriction that can be bypassed using value mappings in a synapse. No other limitation exists in making a synapse. Although this may mean that meaningless synapses are allowed, it has the advantage of letting the user create associations and cause the emergence of new behaviours that the artifact manufacturer may have never thought of. Meaningless synapses can also be seen as having much in common with runtime errors in a program, where the program may be compiled correctly but does not manifest the desired by the programmer behavior.
The use of high-level abstractions, for expressing such associations, allows the flexible configuration and reconfiguration of UbiComp applications. It only requires that artifacts are able to communicate and they have to run the distributed management system (middleware) in order to "comprehend" each-other, so that users can access their services, properties and capabilities in a uniform way.

THE EPLANTOS MIDDLEWARE

The outline of the ePlantOS architecture is shown in Figure 1. This middleware is an upgraded version of the middleware presented in Drossos et all (5). Thus it encapsulates the basic features of the kernel and incorporates new modules, specifically developed for managing plant behaviour. The ePlantOS kernel implements the plug/synapse model manifesting the artifact's services and capabilities through plugs, while providing the mechanisms (API and protocols) to perform synapses with other artifacts via the application layer. Synapses can be considered as virtual channels that feed the lower communication levels with high-level data. Interfacing with networking mechanisms (transport layer) is done via the Java platform and finally data are transmitted through the physical layer to the other end of the synapse to follow the reverse process of transforming low-level information (e.g. messages) to high-level one (e.g. service requests). Data departing or arriving to plugs usually affect one or more of the device capabilities (sensors/actuators), while the kernel assumes the responsibility of translating those data to artifact behavior (e.g. activate a specific actuator in order to achieve a goal).

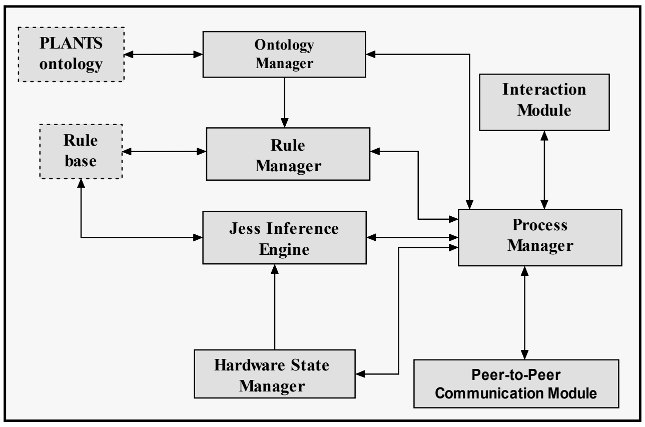


Figure 1. ePlantOS architecture outline

The ePlantOS kernel encompasses a Communication Module, a Process Manager, a Hardware State Manager, an Ontology Manager a Rule Manager and a Jess Inference Engine as shown in the Figure 1. The Communication Module Kameas et al (6) is responsible for communication between different ePlant/Artifact nodes. This module implements algorithms and protocols for wireless, connectionless communication (using the 802.11b/g protocol) as well as mechanisms for internal diffusion of information exchanged. The Process Manager is the coordinator module of ePlantOS. Some of its most important tasks are to manage the processing policies of the ePlantOS, to accept and serve various tasks set by the other modules of the kernel and to implement the Plug/Synapse model. The Hardware State Manager is a repository of the hardware environment (sensors/actuators) inside ePlantOS reflecting at each particular moment the state of the hardware. Through the Ontology Manager ePlants/Artifacts can obtain context-awareness and manifest higher-level behavior (7), Goumopoulos et al (8). Applications state their resource or service needs through concepts that are part of the artifact's ontology. Finally the Rule Manager and the Jess Inference Engine are responsible of the decision making process of the ePlant/Artifact. The decision-making process is based on a set of rules in operational representation forms that are applied on existent knowledge and allow the use of the ePlants ontology for reasoning providing inferential and validation mechanisms. The reasoning is on the definition of the PLANTS ontology, which may use simple description logic or user-defined reasoning using first-order logic.

SCENARIO

The prototype is an everyday application that takes place in the office, and aims at facilitating the user in

looking after his/her indoor plants, despite his/her pressing working time patterns. This ambient-intelligence scenario demonstrates, effectively, the concept of communicating plant, in the context of an every-day environment (e.g., an office) with several layers of decision-making.
The scenario is quite simple. A person has a plant in his office. However busy he may be, he still loves to take care of the plant. Several everyday objects are in his disposal for giving him an appropriate feedback (Lamp, MP3Player, Mobile Phone). Our aim is to form such an application (bioGW) where the plant will be able to provide to the human the appropriate feedback of his condition.The sequence of the scenario's interactions is shown in Figure 2.

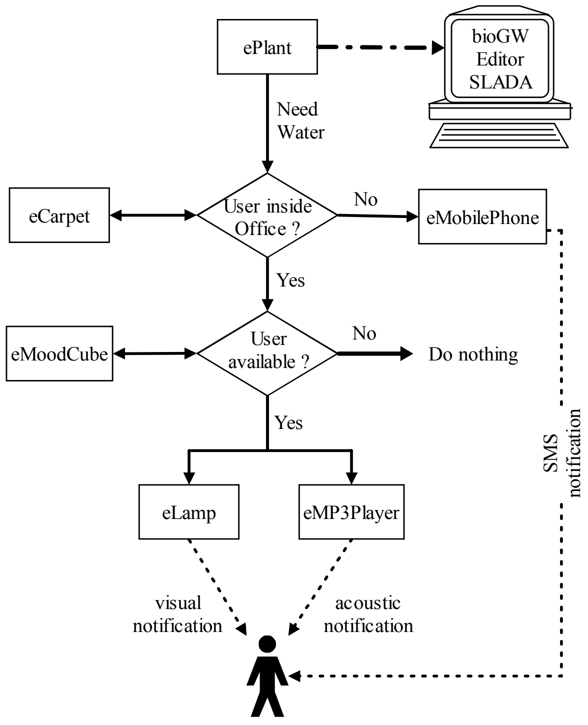


Figure 2. Indoor PLANTS bioGW flowchart diagram

The main actor is the ePlant. The ePlant decides whether it needs water or not using its sensors readings and the appropriate decision making mechanism incorporated in it. An augmented carpet is in the area, an eCarpet whose role is to provide information to the application if the user is inside his office. Similarly, a eMoodCube (a device where the user can set in certain position and change its color, reflecting the user's mood) provides an indication whether the user is available or not.
An augmented Lamp (eLamp) and an MP3Player (eMP3Player) are used to provide visual and acoustic notification respectively to the user. This notification is given according to the status of the eMoodCube. If the eMoodCube is set to "Do not disturb" status, the user is not notified at all. Lastly, in the case the user is not in his/her office, the application uses the eMobilePhone (an augmented Mobile Phone) to send him a SMS and inform him/her about the watering needs of the plant.

COMPONENTS TAKING PART IN THE SCENARIO

The application-level software components that are necessary for the realization of the above scenario are described as follows:

ePlant: The *ePlant* based on its sensor readings is capable of deciding whether it needs water or not. These sensors fall into 2 categories: *Thermistors*, that is sensors that can perceive the temperature of the plants leaves and the environment and *Soil Moisture Probes*, which are able to measure the moisture of the soil. The species of the plant is selected based on the available domain knowledge, from which specific rules will arise and become embedded into the ePlant's knowledge base. The rules combine the information given from the sensors above in order to provide a concrete decision on the current plant's state.

eMobilePhone: The *eMobilePhone* is a personal java enabled mobile phone, used for sending SMS to other mobile phones. There is the software part of the eMobilePhone that represents the entity of the mobile phone in the virtual world. When the eMobilePhone s/w receives a request to notify the user via an SMS, the software will connect through Bluetooth to the mobile phone device and send the SMS to the corresponding telephone number.

eLamp: The *eLamp* is an augmented floor lamp used for visual notifications. The lamps switch is controlled by an FPGA. This FPGA receives commands through the serial port and adjust the state of the *eLamp* accordingly.

eCarpet: The *eCarpet* is an augmented carpet with pressure sensors attached, used for sensing the occupancy of the office (i.e. if the user is in the office). Based on the sequence the pressure sensors are pressed the eCarpet is capable of deducing if someone is entering or leaving a room, thus if the user is present or not.

eMoodCube: The *eMoodCube* is an augmented Mathmos lamp with tilt switches attached, used for defining the current status/mood of the user, such as "available for acoustic nification", "available for visual notification", "do not disturb", etc.

eMP3Player: The eMP3Player is a winamp based mp3 player, used to play audio messages.

TOOLS FOR COMPOSING UBICOMP APPLICATIONS

In order to create the appropriate bioGW we need the support of certain tools. These tools are the Supervising Logic and Data Acquisition (SLADA) tool and bioGW Editor (bioGWE). The former is used to manipulate the rules of an eEntity (ePlant/Artifact) and to monitor its variables (data, actions, states) during its lifecycle, while the later is used for creating connections through eEntities and Composing bioGWs.

Supervising Logic and Data Acquisition (SLADA) tool

The main purpose of the SLADA tool is to provide the ability to the user to monitor and manage a remote ePlant. The main functionalities of SLADA fall into two categories:

- Monitoring of the Various components of the remote ePlant
- Altering the rule base of the ePlant

The logical communication channel that is used between the ePlant and SLADA is provided by the ePlantOS middleware. SLADA and ePlant exchange information through a specific Synapse between a specially designed Plug, the **Monitoring Plug**, or **MPlug**. The MPlug inherits its basic properties from the Plug class, therefore, has all the abilities of a single Plug. However, its functionality is enhanced in the following capabilities(Figure 3).

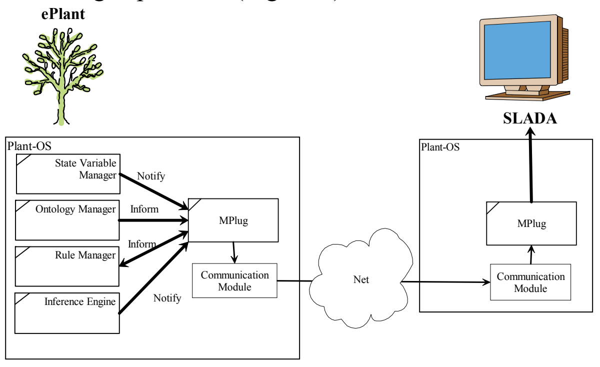
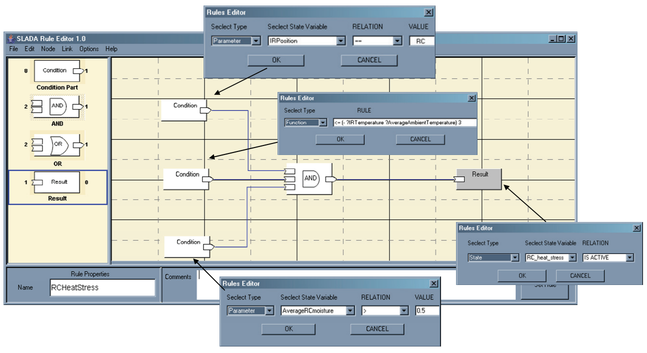


Figure 3. MPlug Communication Scheme

- **Monitoring of the low-level context of the host ePlant**. When created, the MPlug is connected to the state variable manager in order to retrieve the low level data of the host ePlant. From this point on any change on those data is reflected to the MPlug through an event based mechanism.
- **Monitoring of high level state**: The MPlug may receive high level monitoring data, provided by the inference engine, representing higher level states of the ePlant. These variables are processed in the same manner as the state variables.
- **Monitoring the variables of a remote ePlant**. The MPlug can receive the variables of a remote ePlant (via a synapse), and monitor the data they represent
- **Modifying the rule Base.** The MPlug provides the appropriate mechanism for requesting the rule base for the Rule Manager of the local ePlant, sending it to SLADA as well as sending the modified Rules back to ePlant

For modifying the rule base SLADA provides a complete, user friendly, Graphical environment, where the creator of the bioGW can modify the rules to express his needs (Figure 4)



Rules Editor: Seclect Type: Parameter; Seclect State Variable: IRPosition; RELATION: ==; VALUE: RC

Rules Editor: Seclect Type: Function; RULE: <= (- ?IRTemperature ?AverageAmbientTemperature) 3

Rules Editor: Seclect Type: State; Seclect State Variable: RC_heat_stress; RELATION: IS ACTIVE

Rules Editor: Seclect Type: Parameter; Seclect State Variable: AverageRCmoisture; RELATION: >; VALUE: 0.5

Rule Properties: Name: RCHeatStress; Comments

Figure 4. Rules Editing

bioGW Editor (bioGWE)

This tool is an extension of a previous tool called GadgetWorld Editor Mavromatti et al (9) and provides an extensive set of operations on ePlants, Synapses and bioGWs.

In order to operate the bioGWE the presence of eEntities within range is required. The bioGWE discovers all eEntities in the area. When a Discovery is initiated, the GEManager initiates the Peer Discovery Protocol and sends through the Plant-OS a Discovery Message. All the eEntities that receive this message send back to the editor an advertisement message containing the ePlant's T-Plug, which contains information about an eEntity's physical characteristics, as well any information about the eEntity's plugs. By using the eEntities found and provided in the editing pane the user can create its own bioGWs. The creation of the Gadgetworld can be done in a set of simple steps, After using that simple steps to connect all the appropriate plugs, the BioGadgetWorld is ready to be activated and used. The bioGWE then sends to the gadget of each synapse a synapse initiation message. Each gadget that receives a synapse initiation message proceeds with synapse activation. When the synapsing process is finalized, the ePlants inform the GE about the status of the synapse. From that moment on the eEntities are connected with each other form a concrete BioGadgetWolrd

IMPLEMENTATION OF THE SCENARIO

A high-level view of the plugs and synapses that are necessary for the implementation of the Indoor BioGadgetWorld is given in Figure 5.

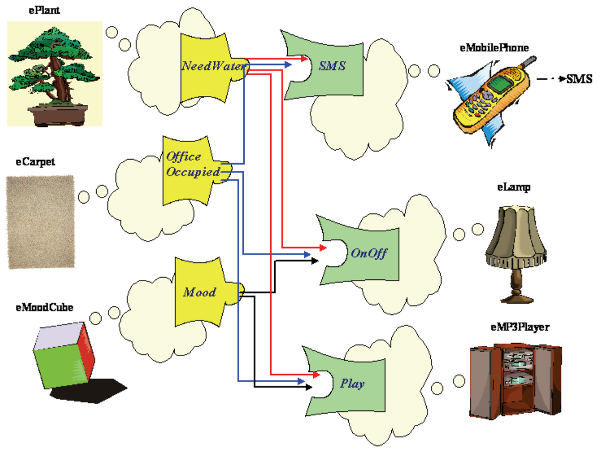


Figure 5. Plugs and synapses for the implementation of the Indoor BioGadgetWorld

We note that for the ePlant, eCarpet and eMoodCube the evaluation of the state (in case of the plant) or the service and capability (in case of an everyday object) is based on a **local decision making scheme**, because the assessment logic depends only on knowledge that is managed by the specific component through its attached sensor network and rule-base. On the other hand, the service provided by the eMobilePhone, eLamp and eMP3Player depends on a **global decision making scheme**, because the rules that govern the decision to offer a service have to take into account the state/capability information of several eEntities. For example, to decide whether to make the lamp blinking (visual notification service to the user) we have to take into account the state of the ePlant, provided by the *NeedWater* plug, the capability of the eCarpet to sense the presence of the user in the office, provided by the *OfficeOccupied* plug and the capability of the eMoodCube to map the mood of the user, through the *Mood* plug. Thus, to turn on or off the lamp we have to define a rule that takes into account all the above plugs. The following table summarizes the properties, plugs and functional schemas of each eEntity participating in the Indoor BioGadgetWorld.

TABLE 1 - BioGadgetworld Configuration

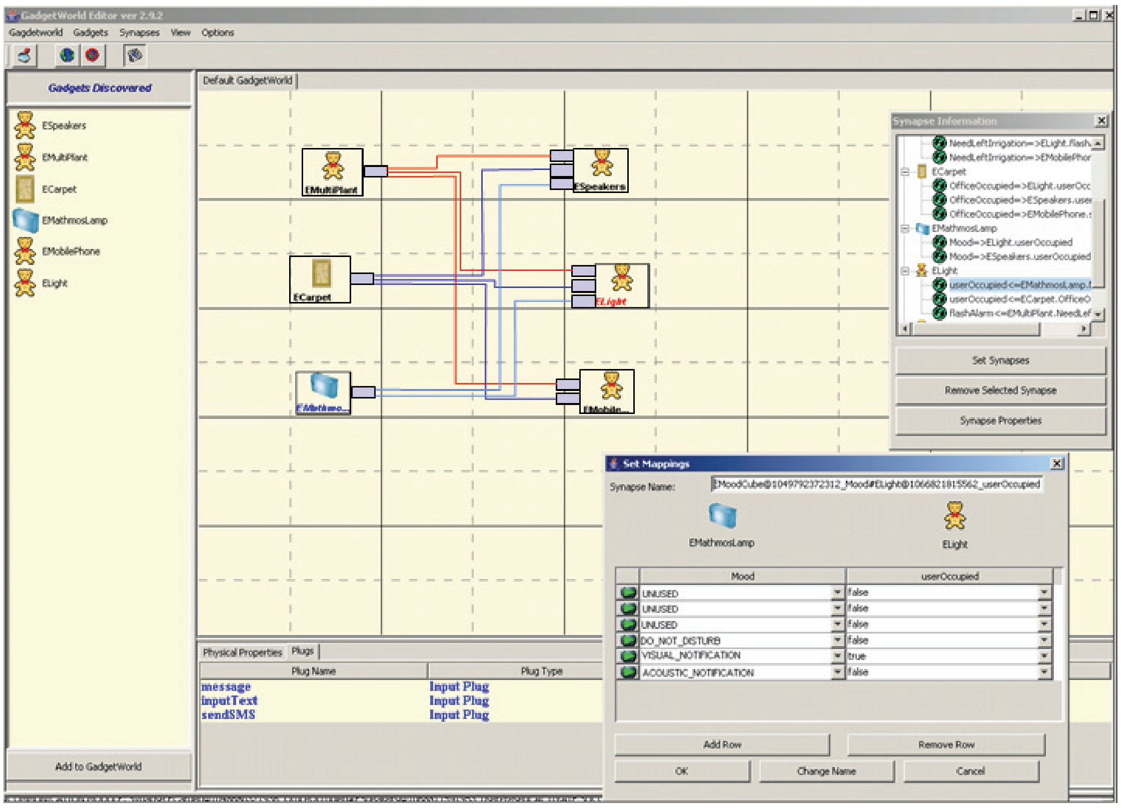
eEntity	Properties	Plugs	Functional Schemas

ePlant	Determining the state of the plant, whether the plant needs irrigation or not.	NeedWater: {OUT|Boolean}	`PlantTemp ← read(Thermistors)`
			`SoilHumidity ←`
			`read(MoistureProbe)`
			`AmbientTemp ← read(Thermistors)`
			`IF PlantTemp - AmbientTemp >`
			`ePlant.TempThreshold OR`
			`SoilHumidity <`
			`ePlant.HumidityThreshold`
			`THEN NeedWater ← TRUE`
			`ELSE NeedWater ← FALSE`

eCarpet	The carpet is placed at the entrance of the office. As the user enters or leaves he/she is forced to step over the carpet. The eCarpet monitors the direction of this movement and updates its plug accordingly.	OfficeOccupied: {OUT|Boolean}	`SensorArray ← read(SensorNetwork)`
			`OfficeOccupied ←`
			`FindMovementDirection(SensorArray)`

eMoodCube	As the moodCube is turned it changes its color and represents user's mood. Possible selections are:
	• DO_NOT_DISTURB,
	• NOTIFY_VISUAL,
	• NOTIFY_ACOUSTIC
		Mood: {OUT|Enumeration}	`Position ← read(Sensors)`
			`Mood ←`
			`MapPositiontoMood(Position)`

eEntity	Properties	Plugs	Functional Schemas
eMobilePhone	Send SMS Service (S1)	SMS: {IN|Boolean}	`IF ePlant.NeedWater AND NOT eCarpet.OfficeOccupied THEN S1()`
eLamp	Light service (S2)	OnOff: {IN|Enumeration}	`IF ePlant.NeedWater AND eCarpet.OfficeOccupied AND eMoodCube.Mood = NOTIFY_VISUAL THEN S2(BLINK)`
eMP3Player	Playing message service (S3)	Play: {IN|Boolean}	`IF ePlant.NeedWater AND eCarpet.OfficeOccupied AND eMoodCube.Mood = NOTIFY_ACOUSTIC THEN S3(A user-defined audio message)`



NeedLeftIrrigation=>ELight.flash
NeedLeftIrrigation=>EMobilePhor
ECarpet
OfficeOccupied=>ELight.userOcc
OfficeOccupied=>ESpeakers.user
OfficeOccupied=>EMobilePhone.s
EMathmosLamp
Mood=>ELight.userOccupied
Mood=>ESpeakers.userOccupied
ELight
userOccupied<=EMathmosLamp.
userOccupied<=ECarpet.OfficeO
flashAlarm<=EMultiPlant.NeedLef

Synapse Name: EMoodCube@1049792372312_Mood#ELight@1066821815562_userOccupied

EMathmosLamp ELight

Mood	userOccupied
UNUSED	false
UNUSED	false
UNUSED	false
DO_NOT_DISTURB	false
VISUAL_NOTIFICATION	true
ACOUSTIC_NOTIFICATION	false

Plug Name	Plug Type
message	Input Plug
inputText	Input Plug
sendSMS	Input Plug

Figure 6. Combining eEntities in the bioGWEditor to create the Indoor bioGW

The composition of the above bioGadgetWorld, in terms of components and synapses between them, is greatly facilitated by the use of the bioGWEditor (see Figure 6) Furthermore the definition of the functional schemas, in terms of rules, for the operation of the various eEntities in the Indoor bioGadgetWorld is facilitated by the use of the SLADA Tool and in particular its rule editor subsystem.
When the creation and activation of the BioGadgetwold is completed, the user should be able to use it seamlessly. As described in section 4 when the plant should need water, it would be able to communicate with the user through vocal, optical or text messages, and the user should make an action on for it.

RELATED WORK

Previous work on this subject can be defined in many fields. The first one is the application field where we need to define the work that has been done in the field of plant monitoring and automated irrigation processes. In that we can find many companies that provide sensors that can monitor the different plant parameters (Soil moisture, temperature, Chlorophyll etc.), such as Phytech Ltd.(10) And ICT Internactional (11). Phytech in particular has established a technique called Phytomonitoring that describes the process of a plant lifecycle monitoring. This however is not done automatically but through data collection from the sensors and user interaction on the rules.

Next fields are mainly technological. Getting in the Ubiquitous Environment Domains, we need to see what options are available. One key aspect is the middleware through which the artifacts can communicate through each other. Well known communication systems are JAVA RMI (12) Jini (13),(14) JXTA (15). Jxta in particular is a set of open, generalized P2P protocols, which allow any connected device on the network to communicate and collaborate as a peer. Project Jxta comes closest to be selected as the networking module for the ePlant/Artifact framework. Each ePlant/Artifact can function as a Jxta peer providing services through its Plugs. The Biogadgetworlds can be thought of as an analogy to groups in the JXTA terminology. But the currently available version of Jxta has been developed with the Java Standard edition in mind. Though ports to other Java editions and implementations on other platforms are available, most of them lose some functionality. Jxme, which is the port of Jxta to Java Micro Edition, is a bare-naked implementation and above that requires the services of a Jxta relay to function as a P2P peer.

In the field of Ubiquitous computing environments we can check the Disappearing Computer Initiative (16) where a numerous projects were successfully finished. In particular we can define Smarts-Its Holmquist L.E., et a (17), where small-scale smart devices can be attached to mundane everyday artifacts to augment these with a "digital self". These devices will be as cheap, as unobtrusive and as generic as state-of-the-art smart labels (i.e., RFID tags). In addition these devices will be enabled with perception of their environment, with peer-to-peer communication, and with customizable behaviour.

In the Disappearing Computer Initiative we can also mention eGadgets project (3), whose platform has been the base on designing the ePlantOS system for the PLANTS Projects. The main purpose of eGagdets was to develop and validate an architectural style for tangible, communicating artifacts [=a Gadgetware Architectural Style (GAS)]. Extrovert Gadgets are objects with communicative abilities. The objects and/or their environments can be enhanced by intelligence. A multitude of loosely coupled gadgets can be bound into ad-hoc interacting clusters which display collective function, thus forming a gadgetworld. The Gadgetware Architecture Style (GAS) provides a common conceptual framework for designers and people, to use e-gadgets as building blocks for composing gadgetworlds.

One interesting approach is the TinyOS (18) that is an event based operating environment designed for use with embedded networked sensors. More specifically, it is designed to support the concurrency intensive operations required by networked sensors with minimal hardware requirements. There are hundreds of TinyOS projects throughout the world. Other related research efforts are Gaia Román M. and Campbell R.H (19) that provides an infrastructure to spontaneously connect devices offering or using registered services. Gaia-OS requires a specific system software infrastructure using CORBA objects, while mobile devices cannot operate autonomously without the infrastructure; and BASE Román M. and Campbell R.H (20) which is a component-oriented micro-kernel based middleware, which, although provides support for heterogeneity and a uniform abstraction of services, the application programming interface requires specific programming capabilities by users

Whereas all the above may be used to formulate a Human-Plant environment, no concrete solution has ever been given. The only similar solution is the one given from PlantCare, where an autonomous application is introduced that takes care of houseplants using a sensor network and a mobile robot LaMarca A. et al. (21).However in this application, the emphasis is given on discussing technical challenges encountered during the deployment of the application. Our approach in contrast emphasizes the development of an architecture that views plants and associated computation as an integral part and allows the interaction of plants, artefacts and people to form a synergistic and scalable application.

CONCLUSIONS

In this paper we have given a practical way, on how, with the appropriate augemented plants and items (that we've called ePlants and Artifacts), middleware, and tools we can create a ubiquitous environment for our plant, through which it can communicate with us. The scenario is versatile, which means that we can alter with a little work the scenario, to embed other ways of communication, other messages or to make the plant able to serve itself. For example by adding to the scenario an irrigation system, and connecting the plant with the appropriate service we can modify the scenario, where the plant, under certain conditions to be able to irrigate itself. This can be done easily only by adding one more synapse from the ePlant to the Irrigation System, and setting the appropriate conditions for triggering this action.
Thus with the appropriate tools we can create a closer relation between Human-Plant, where the plant would be able to express its needs and the human to be able to satisfy them. So as the dog than needs to be fed barks to his boss, we've managed to make the plant “bark”

ACKNOWLEDGEMENT

This paper describes research carried within the PLANTS project (IST FET Open IST-2001-38900); more info can be found at http://www.edenproject/PLANTS. The authors wish to thank their fellow researchers in the PLANTS consortium.

REFERENCES

1. IST Advisory Group. (2003, September). Ambient Intelligence: from vision to reality. Available: http://www.cordis.lu/ist/istag-reports.htm
2. PLANTS Consortium: PLANTS roadmap, PLANTS IST-2001-38900 Technical Report, April 2004
3. e-Gadgets project website: http://www.extrovert-gadgets.net
4. Kameas A., et al.: An Architecture that Treats Everyday Objects as Communicating Tangible Components. In Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom03), Fort Worth, USA, 2003.
5. Drossos, N., Goumopoulos, C. and Kameas, A., A Conceptual Model and the Supporting Middleware for Composing Ubiquitous Computing Applications, Proc. of the IEE International Workshop on Intelligent Environments, Colchester, UK, 28-29 June 2005
6. Kameas A., et al, "eComp: An Architecture that Supports p2p Networking Among Ubiquitous Computing Devices", in Proceedings of the 2nd International Conference on Peer-to-Peer Computing (P2P'02), Linköping, Sweden, Sept 2002
7. The Plant Ontology™ Consortium, http://www.plantontology.org
8. Goumopoulos, C., Christopoulou, E., Drossos, N., and Kameas,The PLANTS System: Enabling Mixed Societies of Communicating Plants and Artifacts, A., Proc. of the 2nd European Symposium on Ambient Intelligence (EUSAI 2004), LNCS 3295, pp. 184-195, ISBN 3-540-23721-6, Springer-Verlag, Eindhoven, the Netherlands, November 8-10, 2004.
9. Mavrommati I., Kameas A. and Markopoulos P.: An Editing tool that manages device associations in an in-home environment. In Proceedings of the Second International Conference on Appliance Design (2AD), HP Labs, Bristol, UK, 11-13 May 2004, 104-111
10. PhyTech: Phytomonitoring™, http://www.phytech.co.il/phyt.html
11. ICT International http://www.ictinternational.com.au/plants.htm
12. "Jini Architectural Overview", Technical White Paper, Sun Microsystems.
13. JINI http://www.sun.com/software/Jini
14. "Java remote method invocation-distributed computing for java", White Paper, Sun Microsystems http://java.sun.com/products/jdk/rmi/index.html
15. Project JXTA web site 6 Conclusions and Future work http://www.jxta.org
16. Disappearing Computer initiative: http://www.disappearing-computer.net
17. Holmquist L.E., et al, "Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artifacts", in Proceedings of UBICOMP 2001, Atlanta, GA, USA, Sept. 2001
18. TinyOS http://www.tinyos.net/
19. Román M. and Campbell R.H., "GAIA: Enabling Active Spaces", Proceedings of the 9th ACM SIGOPS European Workshop, pp. 229-234, Kolding, Denmark, September 2000
20. Becker C., et al, "BASE - A Micro-broker-based Middleware For Pervasive Computing", in Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communication (PerCom03), Fort Worth, USA, 2003.
21. LaMarca A. et al.: PlantCare: An Investigation in Practical Ubiquitous Systems. 4th International Conference on Ubiquitous Computing (Springer-Verlag Lecture Notes in Computer Science Vol. 2498), Goteborg, Sweden, September 2002, 316-332.