Pervasive Computing in Embedded Systems: Designing Cooperative Applications for Real Environments

Alberto Zambrano Galbís
Department of New Technologies
ETRA Research and Development
Valencia, Spain
azambrano.etra-id@grupoetra.com

***Abstract*— The dramatic growth of the amount of information that is made available through computer systems and the increasing need to access relevant information anywhere at any time are more and more overwhelming the cognitive capacity of human users. Instead of providing the right information at the right time, current computer systems are geared towards providing all information at any time. For many future applications, the integration of embedded systems from multiple smart spaces is a primary key to provide a truly seamless user experience. The project PECES has worked during the last two years to offer the technological basis to enable the global cooperation of embedded devices residing in different smart spaces in a context-dependent, secure, and trustworthy manner. The main output of this paper relies on the set of tools developed to create PECES based applications in an easy and understandable way for developers.**

Keywords-pervasive; embedded; smart space; WICO; security; middleware; context; ontology.

I. INTRODUCTION

The dramatic growth of the amount of information that is made available through computer systems and the increasing need to access relevant information anywhere at any time are more and more overwhelming the cognitive capacity of human users. This is an immediate result of the design goal of providing transparent access to all available information that guides the development of today's information and communication technology. Thus, instead of providing the right information at the right time, current computer systems are geared towards providing all information at any time. This requires humans to explicitly and repeatedly specify the context of the required information in great detail.

The vision of Pervasive Computing aims at solving these problems by providing seamless and distraction-free support for user tasks with devices that are invisibly embedded into the environment. In order to provide task support in an unobtrusive and intuitive way, the devices are equipped with wireless communication and sensing technology. This allows them to cooperate with each other autonomously, i.e., without manual intervention, and it enables them to perceive relevant parts of the physical world surrounding their human users.

Together with the richer input and output capabilities realizable by the joint utilization of these embedded devices, this can greatly reduce the cognitive load that is put on users when they need to access information.

While there are various approaches towards enabling the vision of Pervasive Computing, existing approaches are mostly focusing on concepts to realize smart spaces, such as smart meeting rooms or offices. However, truly seamless support for user tasks requires the development of one system that exposes a single and unifying image to its human users. This requires the integration of multiple smart spaces with each other and with information system infrastructure that exists today as shown in Figure 1.

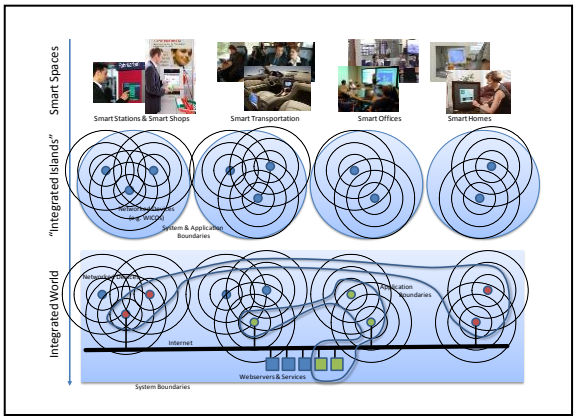

Figure 1. Pervasive Computing Vision

The increasing number of devices that are invisibly embedded into our surrounding environment as well as the proliferation of wireless communication and sensing technologies are the basis for visions like ambient intelligence, ubiquitous and pervasive computing, whose benefits and impact on the economy and society are undeniable. Efforts in related projects have enabled smart spaces that integrate embedded devices in such a way that they interact with a user as a coherent system. However, they fall short of addressing the cooperation of devices across different environments. This results in isolated 'islands of integration' with clearly defined boundaries such as the smart home or office. For many future applications, the integration of embedded systems from multiple smart spaces is a primary key to provide a truly seamless user experience. Nomadic users that move through different environments will need to access information provided by systems

 ISBN: 978-1-61208-171-7

embedded in their surroundings as well as systems embedded in other smart spaces. Depending on their context and on the targeted application, this can be smart spaces in their vicinity such as 'smart stores', or distant places with a specific meaning such as their home or their office or dynamically changing places. The project PECES has worked during the last two years to offer the technological basis to enable the global cooperation of embedded devices residing in different smart spaces in a context-dependent, secure, and trustworthy manner.

The result is a comprehensive software layer that consists of a flexible context ontology, a middleware that is capable of dynamically forming execution environments that are secure and trustworthy, and a set of tools to facilitate application development.

This paper will provide an overview of the results of the research carried out in PECES project, showing how a developer can make use of the software layer provided by using the development tools to create applications that allow the collaboration of embedded devices across different smart spaces, being them co-located or remote. Section II will describe the main building blocks of the software solution proposed, Section III and IV show how to develop and applications using PECES respectively and Section V makes an overview of the process to design test cases. Finally, Section VI describes one of the applications that have been developed and successfully implemented in a real environment as a matter of fact of the applicability of results.

II. THE MAIN BUILDING BLOCKS OF THE SOFTWARE SOLUTION PROPOSED

As mentioned in the previous section, the software layer provided consists in three key components and the applications that allow the collaboration of devices across different smart spaces are built on it. These components are: a context ontology, a middleware and a set of tools to help developers to build the applications.

The context ontology is the basis for capturing the context of the cooperating objects and specifying groups of cooperating objects in an abstract manner.

The middleware consists in a set of application-independent services that enables the dynamic and context-aware formation of a secure execution environment from a set of cooperating objects. This encompasses an addressing and grouping scheme with associated gateway concepts to enable the interaction of cooperating objects between smart spaces, a distributed registry for cooperating objects to enable the dynamic formation of an environment on the basis of applications requirements and all the associated concepts and protocols to ensure that environments can be formed in a secure manner and that the data-oriented communication between cooperating objects is secure.

The development tools aims at simplifying the formation of groups as well as the description of the context of the cooperating objects that are part of the applications. These tools have been created to support developers who want to create applications using PECES middleware.

III. DEVELOPMENT OF APPLICATIONS USING PECES

Structure of a typical PECES application shows the structure of a typical PECES application, where several devices, characterized by their context properties, are grouped in collaborative smart spaces according to their needs, capabilities and context, regardless of whether they can establish local communication or they contact across Internet. They can cooperate to create local or global smart spaces. The locally available services can only be accessed by those devices which are inside the communication range while globally available services are published in the internet and accessible remotely by any device.

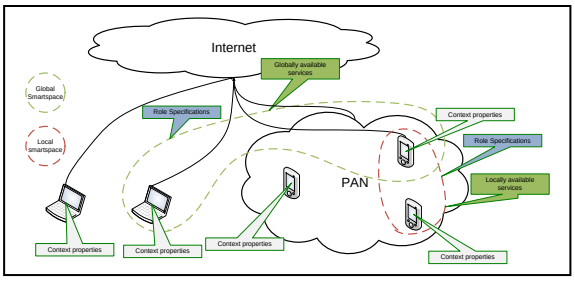

Figure 2. Structure of a typical PECES application

Devices are grouped into smart spaces in an intelligent manner, based on their context properties. Smart spaces are defined by so called "Role specifications", being a Role Specification a set of rules that a device must fulfill in order to become member of a certain smart space.

In order to allow a flexible, open and human-readable way of defining these constraints in the Role specifications, the PECES project has adopted the use of ontologies, developing a custom extensible set of ontologies called "Context Ontologies". The context ontologies have three main objectives inside the PECES middleware:

- Model the context properties of the devices (for instance, services, device's capabilities, locations, ownerships, etc.).
- Model human-readable relationships between these properties (for instance, "device offers service", "device is located at location" or "device is owned by person".
- Provide an engine that allows the middleware to perform queries over the context properties, by using the defined relationships (for instance, "select all devices located at a certain location", "select all devices owned by a certain person", and combinations of type "select all devices located at a certain location and owned by a certain person" or "select all devices offering the service that is required by a certain person").

The middleware offers a set of context properties that allow the operation of the middleware and the prototype applications developed in the project. The context ontologies can be easily extended to support further applications, in case new concepts and relationships are needed.

 ISBN: 978-1-61208-171-7

Summarizing:

- Devices are characterized by a set of context properties
- Role specifications define the characteristics a device must have in order to become member of a smart space
- The formation of a smart space is not limited physically, since three different types of smart spaces can be defined:
  - Device level smart spaces: intra-device.
  - Local smart spaces: restricted to directly reachable devices, independently of the communication channel used (Ethernet, WiFi, Bluetooth...)
  - Internet smart spaces: publicly defined smart spaces, reachable by any device with access to the internet
- PECES services offered by a certain device will be available to the other partners of the smart spaces it is part of

IV. IMPLEMENTATION OF APPLICATIONS USING PECES

A set of development tools has been developed inside the PECES project, to assist developers in the design of new applications using the PECES middleware. These development tools are provided as an Eclipse plugin.

A. *Project Set up*

The development of a PECES application implies working with several different projects within the Eclipse environment. Usually, a developer will have to deal with two projects:

A PECES project, which will be the working basis. This kind of projects contains special files that store the description of the whole system, and that are built step by step during the development process, using the different modules provided in the PECES development tools.

Several JAVA projects with PECES nature. These projects contain the actual code that takes part of the different software pieces that compose the whole application. Usually, it will exist one JAVA project per device taking part in the application.

Basically, all the things needed to be used with the PECES development tools will be created within the PECES project. The content of the other projects will be automatically created, based on the description of the application provided by using the PECES development tools. At the end, the JAVA projects will contain the structure of the final software pieces, including all the PECES-related instantiations and initialisations. Work beyond will include the actual implementation of the services and the application logic.

B. *Instantiating Devices*

The first step in the definition of a PECES application is the definition of how many kinds of devices will participate on it. Usually, this corresponds to the number of software pieces that will be necessary in order to run the whole application. For instance, a simple service provider/consumer application would have two software pieces, thus two devices. Nevertheless, a more complex application could have several different software pieces collaborating among each other.

Once the number of devices has been decided, the PECES Device Definition can be used to define them. This task will result in the creation of several new JAVA projects (as many as devices get defined), where the different software pieces of the application will be built.

The devices needed to run the application have to be instantiated, providing them a name and assigning each device the extra PECES functionality that will be deployed in it:

- Coordinator: the device will be then in charge of defining and managing one or more smart spaces.
- Gateway: the device will be able to provide Internet access to other devices.

The devices not being coordinators or gateways are just members of the smart space.

As part of the instantiation, the developer has to select the communication plug in to be deployed in the device based on its features, namely:

- MxBluetoothTransceiver: for devices with Bluetooth capabilities.
- MxIPBroadcastTransceiver: for devices with IP-based network capabilities, using datagram sockets.
- MxIPMulticastTransceiver: for devices with IP-based network capabilities, using multicast sockets.
- MxIRTransceiver: for devices with IRDA (infrared) capabilities.
- MxSerialTransceiver: for communication via serial connection over USB on Sunspots.
- MxSpotTransceiver: for radio stream communications on Sunspots.
- EmulationTransceiver: needed for the debugging tasks with the PECES development tools.

Figure 3 shows a screenshot of the development tools interface with the different type of devices available and those involved in the smart space application under development.

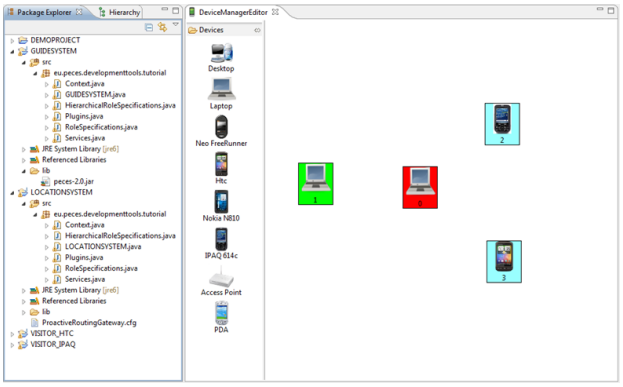

Figure 3. Development Tools Screenshot

C. *Defining Context Properties*

The context properties of the devices are the key elements on the intelligent behaviour of the PECES applications. Devices use their available possibilities to be

aware of their context, and the PECES applications react to this context by building new groups of devices and bringing new services in function of the situation of any device at any moment.

In order to model the behaviour of the PECES application, it is necessary to specify which kind of context information will be useful in order to reason which devices must be able to take part of which groups, thus being able to communicate with its partners and make use of their services.

The PECES applications model these context properties using ontologies. Ontologies are formal models of generic concepts and the relations among them. They provide an easy way of modelling the real world, and therefore any logical condition over the context properties of the devices that may be specified for the correct operation of the application. Examples of context properties and conditions that can be specified with ontologies could be "All red devices owned by John Doe", "All red devices owned by John Doe and located in Valencia" (combination of several properties).

The PECES project already delivers a set of context ontologies that covers the basic concepts necessary to build-up applications, and some further concepts used within the project specific use-cases. These ontologies can always be extended to cover new concepts necessary for new applications.

PECES development tools provide an ontology editor which automatically creates the instances of all the devices defined with the PECES Device definition tool. Therefore, the work of the developer will just focus in the following points:

- Instantiating all smart spaces that will compose the application
- Instantiating all the services to be implemented and used in the application
- Instantiating all the properties of the devices, and relate them to the proper devices

D. Role Specifications

The basis of any PECES application is its ability to build up groups of collaborative objects in an intelligent manner, based on their characteristics and the patterns provided by the application designer. As it has been mentioned in previous sections, the characteristics of a certain device have to be formally modelled by using the context ontologies. The next step is the design of constraints using these characteristics that can be used later on to dynamically build the smart spaces and group all devices with a common background that can collaborate with each other to achieve the objective of the application.

With this objective PECES provides a Role Specification editor with specific tasks:

- Assign a specific Role Specification to each device. It sets which coordinator will be in charge of specifying the roles, thus managing the corresponding smart space.
- Scope of the defined smart space. It specifies the level where the role specification will be published to (device level, space level –local- or Internet level).
- Member's minimum trust level. In case the application uses security concepts, this field specifies the trust level a coordinator must have in another device to allow it to become member of the smart space.

A role specification defines which devices will be members of a certain smart space. It is composed by one or more rule sets. Each rule set defines certain constraints to be applied on the devices' properties (for instance, "a device must be red"). Any member fulfilling one or more rule set will become member of the smart space. A device fulfils a certain rule set only if all the constraints contained there are fulfilled (i.e., an AND condition is applied inside a rule set). Figure 4 shows a number of examples which clarifies this explanation.

		Devices					
		Red		Green		Blue	
		Small	Big	Small	Big	Small	Big
Role specification 1	Rule set 1a **Red & small**	X					
	Rule set 1b **Green & big**				X		
	Member	X			X		
Role Specification 2	Rule set 2a **Blue**					X	X
	Rule set 2b **Red and small**	X					
	Member	X				X	X

Figure 4. Example of Devices Role Specification

E. Services

The PECES middleware facilitates the implementation of services that, once implemented in a device, can be shared among other members of the own smart space. Therefore, services are an important piece of the whole PECES application structure. In cooperation with other elements of the middleware, it is possible to design services that will be available only to certain types of devices, services that will be available only to devices that can be trusted or even services with several interfaces that will be accessible or not based on the trust level or characteristics of the client devices.

PECES provides a Service Editor which supports developers in the implementation of services. Developers will have to specify which device implements the service and the availability of the service – device level, space level or Internet level-.

A service is composed by one or more methods (interface of the service) which can be called by clients, and which generate a result based on the parameters received. For each of the defined methods, PECES Service Editor tool will create in the proper project an empty function with a "TODO" comment inside, indicating to the developers where to include the actual implementation of the service.

 ISBN: 978-1-61208-171-7

F. *Hierarchical Role Specifications*

There are applications where it can be useful to join all members of smaller smart spaces into a single bigger smart space. For instance, in a city full of smart cars, grouping all devices attached to a car and implementing local user-oriented services, it could be interesting to define a super-group with all smart cars allowing the broadcast of traffic information messages among all smart cars.

PECES provides a Hierarchical Role Specifications definition tool to allow the developers to easily create all the code necessary to define and instantiate such kind of smart spaces:

The Hierarchical Role Specification editor offers the following options:

- At a coordinator level, it specifies the device that will instantiate the hierarchical role specification.
- At available smart spaces level, it shows all the smart spaces defined in the project.
- At a selected smart spaces level, it holds the list of smart spaces that will take part of the hierarchical smart space.

G. *Security Aspects*

The PECES middleware offers a security layer that adds extra functionalities to the application. Its use is completely optional. The basis of the security layer is the following:

- Every device carries a certificate, signed by a certain authority or by another certificate.
- Every device stores public certificates of other devices, classified along three different trust levels:
    - Full trust: certificates of devices with the maximum level of trust.
    - Marginal trust: certificates of devices with a lower level of trust.
    - No trust: certificates of not-trusted devices.
- Every role specification can be associated to a certain trust level, which is the minimum trust level the coordinator must hold with another device in order to assign him the role. Figure 5 shows a graphical example which clarifies this concept.

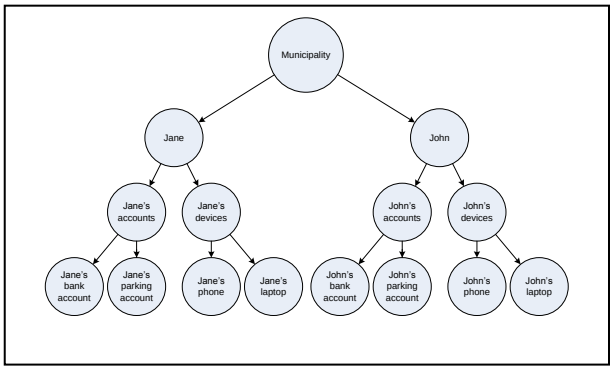

Figure 5. Example of trust levels

PECES Security Configuration tool assists the developer in the creation of the certificates necessary for the security layer to work. Basically, one security configuration will be needed for each certificate authority (roots in the trust level).

The security configuration implies the Root Certificate Configuration, to allow the configuration of the Certification Authority and the Client Certificate Configuration, to allow the design of a certification chain.

V. DESIGNING TEST CASES

The PECES development tools offer all the necessary mechanisms to run the application under development in a testing environment, where the reactions of the different software pieces to different events and changing situations can be triggered and observed, thus helping in the validation of the development process.

The tools offered are able to structure the testing process in a set of test cases. A test case is understood as an experiment; i.e., all the software pieces are run in parallel, the situation to be tested is induced, and the reactions and behaviour of the different pieces is observed (via its console output and graphical visualizations).

A test case is hence defined by a sequence of events that are induced in the testing environment where the different software pieces are run. The sequence of events is defined by the tester, with the objective of triggering and checking a certain behavior of the application.

When defining a test case, the developer will have to define the set of events to be used, ordering them in the proper sequence afterwards.

The context of the devices under test can be modified by introducing device context change events. This allows the developer to introduce artificial changes in the context of the devices under test, thus inducing changes in the behavior of the application.

The tool also allows the developers to introduce connection link change events to define which devices can interact with each other. This is very useful when testing local interactions between devices, or the behavior of the operation when one of the devices in on longer available.

Finally, the PECES development tools offer an execution environment where the software pieces to be deployed in different devices can be run, and certain conditions (the events previously defined) can be induced, causing reactions and interactions between the devices of the application that can be observed and analyzed.

Once the simulation is finished, the developer can access a test log to observe an aggregated and ordered version of the console output of all devices. This tool provides all information coming from the log of the PECES middleware, and further user custom messages the different software pieces can print. The lines in the shown log comply with the following rules:

- Messages are ordered as they are produced, independently of which is their source. This facilitates the observation of interactions and cause-effect relations between the different devices
- First item in every line identifies the source of the message (name of the device)

 ISBN: 978-1-61208-171-7

- Messages coming from the logging facilities of the PECES middleware begin with some information contained between brackets ([]), namely
- Type of message (ERR, DBG,LOG)
- Instant when the message is produced
- Class printing the message

The developer can use the logging facilities of the middleware in order to ensure that this format is always followed in the log files.

VI. IMPLEMENTATION OF APPLICATIONS USING PECES

One of the key challenges PECES technology addresses is to provide the user with a seamless experience when he/she moves through different smart spaces, being them physical or virtual. A delicate balance between usefulness, security and non-intrusiveness must be kept. Technology must be there all the time, but the user must not see it, he/she has to perceive just the benefits brought by the applications enabled by PECES technology.

In this context, a Smart Access Control prototype has been developed to validate the PECES' main features in a real environment. To get an idea of the scenario, imagine the user, John Smith, travelling in his car. He has a PDA where he planned his trip – a visit to one of his main customers to hold an important meeting. The moment he got in the vehicle, all smart devices on board – from the PDA to the in-car satellite navigator - became aware of each other's presence. PECES enabled their mutual discovery and their dynamic interaction. Based on the interests of the user, the devices present the possible functionalities available and offer the user a number of services.

The first service provided seamlessly to the user is the localization of a parking near the meeting location. The navigator automatically sets as destination point the parking entrance and the system books a parking lot for John.

Whist he is driving, the car joins the smart space of the cars in the area and receives real time notifications of the traffic incidents, allowing the recalculation of the route until the destination.

In the way to the customer's office, there is an access control. The smart car is automatically registered and the user is charged the corresponding tax.

When John gets to the parking entrance, his car number plate is recognized by a CCTV camera, the barrier opens and John parks the car in a parking lot booked for him. At the same time, John's personal data is transferred with the requested security to the parking system for invoicing. While he parks, the reception management system of the building negotiates with John's personal device his personal access to the building. He leaves the car and reaches reception. Once he is in the building, he will get access to all the locations and services that the system assigns to users with a 'guest' profile. Another user working in the customers' company will get access to different locations and services than John, such as for example the schedule of his/her department meetings or the monthly payment day.

Once John is back in the parking, he gets into his car and approaches the exit. The camera recognizes the plate number and automatically opens the barrier and invoices John, who receives a message with the amount of money he has been charged. Figure 6 shows a schema of the smart access control application.

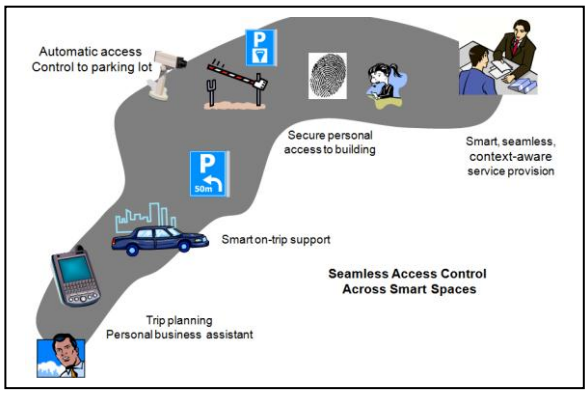

Figure 6. PECES Smart Access Control Application

ACKNOWLEDGMENT

PECES is a project funded by the European Commission under the Seventh Framework Program. PECES Consortium has actively participated in the elaboration of the contents of this paper based on the work performed during the two years the project has been running..

REFERENCES

[1] A. Zambrano, Z. Rak, S. Kirusnapillai, "Use Case Specification," December 2008
[2] W. Apolinarski, M. Handte, P. J. Marrón, A. Zambrano, Z. Rak, S. Kirusnapillai, "Middleware Prototype," September 2010
[3] A. Zambrano, Z. Rak, S. Kirusnapillai, "Development Tools Specification," April 2010
[4] A. Zambrano, Z. Rak, S. Kirusnapillai, "Development Tools Prototype," June 2011
[5] W. Apolinarski, M. Handte, P. J. Marrón, A. Zambrano, Z. Rak, S. Kirusnapillai, "Middleware Prototype," November 2010.1109/SCIS.2007.357670.

 ISBN: 978-1-61208-171-7