Building Smart Space Applications with PErvasive Computing in Embedded Systems (PECES) Middleware

K. Selvarajah, R. Zhao and N. Speirs
School of Computing Science
Newcastle University
Newcastle NE1 7RU, United Kingdom
Email: { k.selvarajah, ran.zhao1, neil.speirs}@ncl.ac.uk

***Abstract*— 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. PErvasive Computing in Embedded Systems (PECES) project develops the technological basis to enable the global cooperation of embedded devices residing in different smart spaces in a context-dependent, secure and trustworthy manner. This paper presents PECES middleware that consists of flexible context ontology, a middleware that is capable of dynamically forming execution environments that are secure and trustworthy. This paper also presents set of tools to facilitate application development using the PECES middleware.**

Keywords- Pervasive Computing; Middleware; Smart Spaces; Context Ontologies; Development Tools; Security;

I. PECES PROJECT

The objective of the PECES project [1] is the creation of a comprehensive software layer to enable the seamless cooperation of embedded devices across various smart spaces on a global scale in a context-dependent, secure and trustworthy manner. The benefits of pervasive computing and their undeniable impact on the economy and society have led to a number of research and development efforts. These efforts have enabled smart spaces to integrate embedded devices in such a way that they interact with a user as a coherent system. However, they fall short of addressing the co-operation of devices across different environments. This results in isolated 'islands of integration' with clearly defined boundaries.

Aura [2] middleware is based on the concept of smart environments that focuses on providing services in non-intrusive manner. BASE [3] is a service based middleware that supports the adaptation of communication protocols and technologies and it is based on the concept of smart peers. Integrated development of context-aware applications in smart spaces is presented in [4]. For many future applications, the integration of embedded systems from multiple smart spaces is a primary key to providing a truly seamless user experience. Nomadic users that move through different environments will need to access information provided by systems embedded in their surroundings as well as systems embedded in other smart spaces.

The PECES project and its consortium are geared towards addressing the challenges in pervasive computing environments in order to provide a truly integrated solution. The most innovative features of the PECES middleware are to enable the communication among heterogeneous devices across the different smart spaces using dynamic addressing, security and context ontologies.

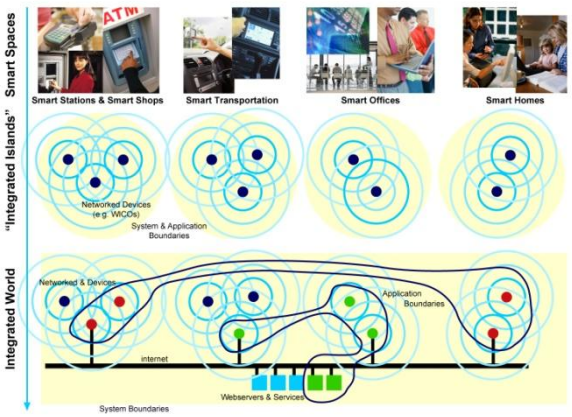

Figure 1: Pervasive Computing Vision

In this paper, the PECES middleware and development tools are discussed. Section II introduces the ontologies developed for the PECES project prototype application development. Section III describes the PECES middleware and its novel features such as role specification, dynamic addressing, local and remote gateway and security concepts. Section IV presents the PECES development tools that have been developed by the PECES consortium for middleware based smart space application development and testing. Conclusions are then presented in Section V.

II. PECES CONTEXT ONTOLOGIES

Context ontologies define a common vocabulary to share context information in a pervasive computing domain and provide machine interpretable definitions of basic concepts in

The work presented here is sponsored by European Commission under FP7 Programme (Grant agreement number FP7- 224342-ICT-2007-2).

DOI: 10.5176_2010-2283_1.4.103

the domain and relations among them. They offer quite promising and powerful mechanisms for defining, acquiring, understanding, processing and sharing context and inferring new knowledge based on available data and context [5].

The PECES project develops a general purpose, domain independent middleware which enables the seamless cooperation of embedded devices across smart spaces on a global scale in a context-dependent, secure and trustworthy manner. From a data perspective this means that a possibly very heterogeneous set of devices needs to communicate information among themselves in contexts and environments which cannot be predetermined statically. Meaningful communication, i.e., the understanding and correct interpretation of the type, content and context of the exchanged data is essential in this setting. While the type and content of information is dependent on the specific applications, the context in which the application executes can be generalized and described across application domains. The PECES ontologies are freely available from [12] and detailed information about the PECES ontologies can be found in [7].

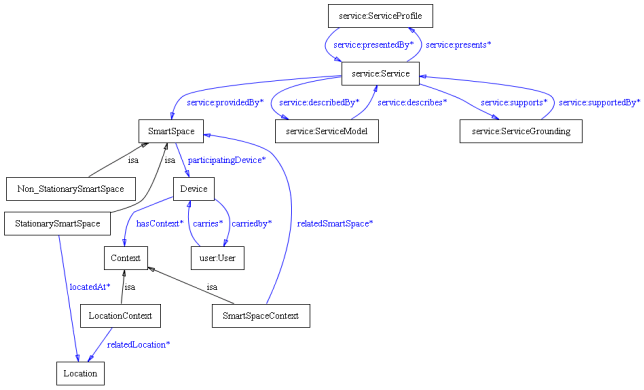

Figure 2: Contextual Concepts within a Smart Space

The basic concepts to model the contextual information of a smart space are *Device*, *Context*, *SmartSpace*, *Location* and *Service* and the relationships among them are shown in Figure 2. The *Device* concept provides vocabularies to model specification of devices inside smart spaces and the *Context* concept is used to extend sub-concepts such as *LocationContext* or *SmartSpaceContext* for representing a set of context instances. The *SmartSpace* concept is used to extend to different kinds of smart spaces. Two main categories of smart space are defined respectively by two subclasses *StationarySmartSpace* and *Non_StationarySmartSpace*. The *StationarySmartSpace* concept represents smart spaces having fixed location and the *Non_StationarySmartSpace* concept represents mobile smart spaces. To express the idea that a smart space provides a service, a *Service* instance is referred to by a *SmartSpace* instance using a *service:provides* property. Vice versa, to express a service is provided by a smart space, inversed property *service:providedBy* is used.

III. PECES MIDDLEWARE

The PECES project consortium decided to build the cooperation layer on top of the BASE middleware [3]. This enabled the project consortium to focus the development efforts on the novel and innovative features of the PECES middleware. BASE is freely available as open source under BSD license which facilitates the necessary modifications and extensions and enables the free reuse – even for commercial exploitation. The BASE middleware enables the communication between devices that are within communication range. Yet, in order to achieve the goal of providing cooperation layer that enables the seamless interaction within and across the boundaries of a single smart space, it was necessary to extend the BASE concepts. The extension of the BASE middleware focused communication gateway concepts, addressing concepts and smart space and security concepts. The figure below shows the BASE components as well as the new features added to achieve PECES project goal. More detailed information about the PECES middleware can be found in [8], [9], [11].

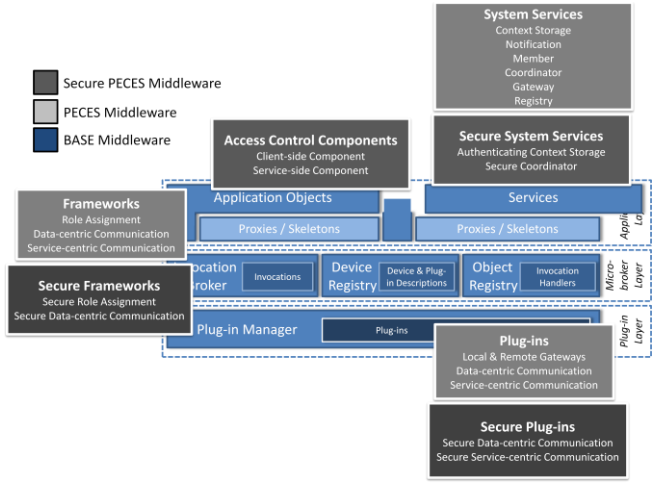

Figure 3: PECES Architecture

A. *Generic Role Assignment*

Due to the continuous changes in context and mobility of devices the underlying systems can be highly dynamic and the network topology can change frequently in the pervasive computing environments. So that it is vital to enable pervasive computing applications such as PECES based applications to adapt to the continuous changes in context and device availability. The responsibility for adaptation can be shifted between different entities. In cases where changes are infrequent, a user may manually configure and adapt the system. However, if changes are frequent, manual configuration and adaptation are clearly not a viable approach as they conflict with the goal of distraction free support for tasks. In order to mitigate this, the adaptation can be automated through the application. This approach relieves the user from performing manual adaptation but it complicates the development of applications and it may result in inefficiencies in cases where multiple applications implement and use similar adaptation mechanisms. As a result, the PECES middleware is aiming at automating the initial configuration and the continuous adaptation to changes in order to shield the user and the application developer from the accompanied complications.

In order to be suitable for a broad range of different systems and in order to minimize the utilization of resources that are required for automation, the PECES middleware provides configuration and adaptation support by means of a uniform abstraction. To create a uniform abstraction that is suitable for a broad range of different configuration tasks, it is necessary to introduce a clear separation between the result of a configuration, the computations that need to be done to produce it and the utilization of this result. This enables the reuse of the same basic mechanisms for different tasks. Generic role assignment provides such a uniform abstraction. More detailed information about the role assignment concepts can be found in [8].

A role can be assigned to any device as long as there are no further constraints that limit the assignment. To enable the automated computation of an assignment that reflects a particular goal of a configuration task, generic role assignment introduces rules. Rules define contextual constraints on the assignment of roles to devices. The simplest form of contextual constraint that is generally useful for all configuration tasks is a simple filter. An example of such a filter is to demand that all devices should be at a certain location. Another form of contextual constraint that is particularly relevant for PECES are so-called reference rules. Reference rules refer to a set of devices that has been assigned a particular role.

The set of rules together with their corresponding roles form a role specification. Given that the necessary contextual information can be captured by sensors or other types of information sources, one can use an algorithm to automatically assign roles to the devices whose context satisfies the constraints specified by their rules.

B. Smart Space Concept

A smart space can be defined as a group of networked devices that cooperate to support their users. The boundaries of a smart space are typically defined on the basis of a geographic location, e.g. a room or a building. However, such narrow definitions are not flexible enough to support the application prototypes in the PECES project. Obviously, these smart spaces cannot be defined on the basis of a single location. For example in applications based upon a car, the whole car, i.e. the smart space itself, is mobile.

In order to extend the definition, the addressing and grouping scheme can be used to support the formation of smart spaces based on arbitrary contextual properties. However, the resulting definition will be automatically restricted to devices that are residing in the same local network. This is a result of the fact that the formation process of basic groups is limited to a local network. Yet, for typical smart spaces local connectivity is guaranteed.

To support smart space formation, the PECES middleware introduces three additional components which are coordinator, member and gateway. These components can be easily motivated by looking at the anatomy of the smart spaces that are identified in the PECES Use-Case Specification [6]:

- Coordinator: A smart space consists of at least one coordinator device. This device is responsible for identifying members of the smart space based on role specification.
- Member: In addition to coordinator device, a smart space may contain additional devices that are dynamically entering or leaving the local network. Depending on the context, a member device might either be integrated or not. Currently, a member device can only be integrated at most into one smart space at a time.
- Gateway: Some devices that are part of a smart space may also be able to communicate with other devices through an Internet connection. Examples for such devices are smart phones or residential gateways as well as laptops that are equipped with a UMTS modem. In these scenarios, the PECES middleware gateway functionality provides connectivity for other devices in the smart space.

C. Communication Gateway

Due to the heterogeneity of devices and communication technologies, it is not safe to assume all future devices will be equipped with the same set of communication technologies. As an example consider that a sensor node might only be equipped with ZigBee 802.15.4 but not with Bluetooth in order enable energy-efficient communication. Thus, in order to enable a Bluetooth enabled device to communicate with such sensor nodes, it is necessary to use a device that is equipped with both technologies as a local gateway. Similarly, due to the associated costs and other factors, not all devices will have a direct connection to a global interconnection network like the Internet. In order to enable the communication between devices that are not directly connected to the Internet, it is necessary to enable some devices to act as remote gateways for others.

PECES middleware support local gateways as well as remote gateways. The main difference between these two types of gateways is that the local gateway locally shares the required knowledge. In the remote case, the knowledge sharing should be restricted to a minimum in order to avoid the costly distribution of frequently changing information. The remote gateways need to be realized differently in that they require an external entity to distribute the information that is distributed by means of device discovery in the local case. This information will be distributed by means of the Registry. More detailed information about the PECES Registry Interface can be found in [9].

D. Security Concept

PECES middleware introduce a basic trust model that is used as basis for the concepts and mechanisms of the middleware. These mechanisms enable the secure interaction of devices. To enable this, they span the management of cryptographic keys, the authentication of information – specifically context information and role assignments, the secure data- and service-centric communication as well as role-based access control. Although they do not introduce additional interaction features, together they span the whole set of security-related requirements identified in the PECES prototype applications.

The security mechanisms are modular and they introduce a certain degree of configurability that can be leveraged by application developers for optimization purposes. This enables them to define application-specific tradeoffs between security and application performance. In order to simplify the configuration of these mechanisms, the PECES project also provides appropriate development tools (will be discussed in the next section) that simplify basic security related tasks such as the distribution of keys and certificates during application development. The PECES middleware is designed to provide support for both secure and unsecure version of the PECES middleware application development. If security is not a requirement for any specific application, developers can make use of the unsecure version of the PECES middleware APIs for smart space application development.

IV. PECES DEVELOPMENT TOOLS

The development tools suite provides features to build and test applications based on the PECES middleware. The tools are implemented as Eclipse plugins [10]. The development tools suite provides several tools to support different activities during the application development, modelling and testing phases. The following sections explain the tool sets necessary for the application development which are the Peces Project Tool, the Peces Device Definition Tool, the Peces Ontology Instantiation Tool, the Peces Security Configuration Tool, the Peces Role Specification Definition Tool and the Peces Service Definition Tool. The PECES Development Tools are already available online and that can be installed in Eclipse IDE using "Installed New Software.." feature from the PECES project tools site: http://www.ict-peces.eu/eclipsetools.

A. *PECES Project Tool*

This is the first tool to start with the application development process. Using this tool, a PECES general project can be generated in the Eclipse workspace with three different folders (*ConfigurationTool*, *ModellingTool* and *TestingTool*) to keep different configuration, modelling and testing related files which will be generated by other tools in the development process. This project is used to provide interfaces with others tool using *.xml and *.owl files. For example, the following screenshot shows a *DEMOPROJECT* project created by the Peces Project Tool with *ConfigurationTool*, *ModellingTool* and *TestingTool* folders.

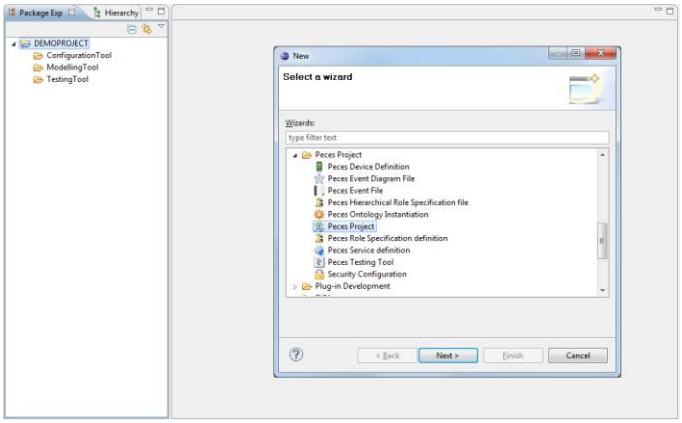

Figure 4: Screenshot of the Peces Project Tool

B. *PECES Device Definition Tool*

The This tool provides a graphical user interface (GUI) for application developers to specify the device description. The Peces Device Definition Tool can be used to define BASE/PECES middleware communication plugins such as IP, Bluetooth, ZigBee (e.g. *MxIPBroadcastTransceiver*, *MxIPMuticastTransceiver*, *EmulationTransceiver*), and device functionalities (e.g. *Coordinator*, *Gateway*, *Coordinator&Gateway*, *Member*) and also device names. The devices can be selected and placed in the editor area of the tool and necessary functionalities and communication plugins can be then defined by right clicking on the devices. After defining the device functionality, different colors will be shown according to the selected device functionality (e.g., a coordinator is red).

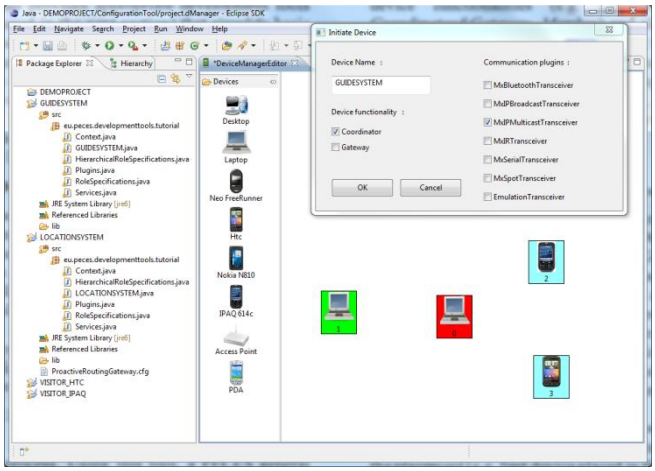

Figure 5: Screenshot of the Peces Device Definition Tool

Figure 5 shows an example application in which four devices are defined namely *GUIDESYSTEM*, *LOCATIONSYSTEM*, *VISITOR_IPAQ* and *VISITOR_HTC*. The *GUIDESYSTEM* is defined as the coordinator of the smart space (shown in red) and *LOCATIONSYSTEM* is defined as a gateway device (shown in green). Two member devices are *VISITOR_IPAQ* and *VISITOR_HTC* and those devices are shown in blue in Figure 5. The screenshot also shows four different Java projects which are automatically generated for each devices with necessary PECES middleware library (peces-2.0.jar).

After placing the selected devices in the workspace, device IDs are automatically generated according to the order of the placement (e.g. first device placed in the workspace will be given ID of 0, the next device will be given ID 1 and so on). Application developers may change the device name and device communication features as well as device functionalities such as coordinator, gateway and member but the device ID cannot be changed. The device related configuration details are recorded in project.xml file.

C. *PECES Ontology Instantiation Tool*

This tool provides a user interface for static context properties. The Peces Ontology Instantiation Tool enables the application developer to instantiate the devices. This tool supports all PECES ontologies (e.g., http://www.ict-

peces.eu/ont/device.owl) as well as other custom ontologies (e.g., http://www.daml.org/services/owl-s/1.1/Service.owl) which application developers may wish to use for their application. The Ontology Instantiation Tool automatically loads the participating device name and its assigned functionality information from the project.xml file which was generated by the Peces Device Definition Tool. The Peces Ontology Instantiation Tool provides GUI where application developers can add instances and link context properties.

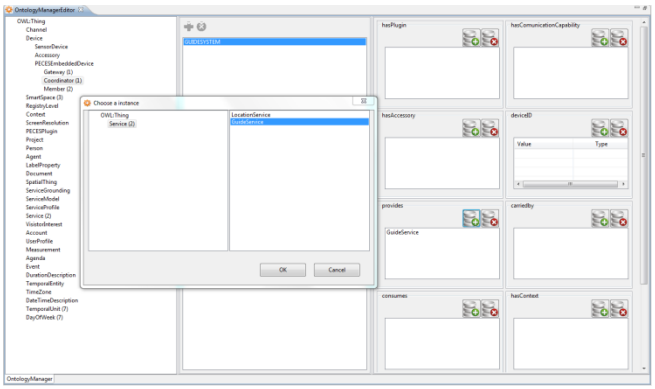

Figure 6: Screenshot of the Peces Ontology Instantiation Tool

D. PECES Role Specification Definition Tool

The PECES Role Specification Tool provides an interface where developers can define the different rules that the application will use to dynamically form groups of collaborative devices. In a PECES middleware, these rules are written essentially as constrained queries over the context properties of the devices. For that reason, the PECES Role Specification Definition tool loads the results of the PECES Ontology Instantiation Tool, showing on a tree-shaped diagram all the devices that have been defined in the project, and their properties (upon which the rules will be defined). When the process is completed, this tool generates necessary java code for role specification which actually defines the formation of the smart space.

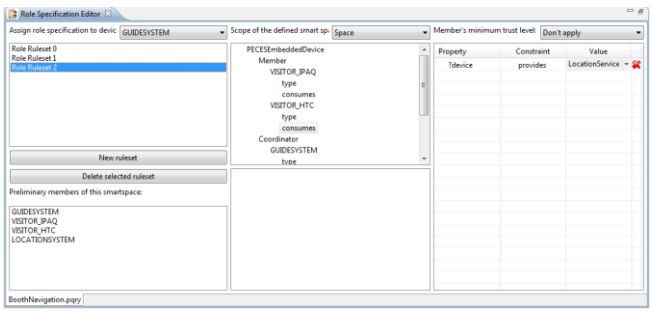

Figure 7: Screenshot of the Role Specification Definition Tool

E. PECES Service Definition Tool

The PECES Service Definition Tool provides a simple interface to the developers that allows the automatic generation of all the code needed to instantiate and make use of a PECES middleware based service. When the developer decides to make use of the PECES Service Definition Tool to define a service, a window shows a list of all the services that have been defined with the PECES Ontology Instantiation Tool. The developer can then simply choose the service to be defined.

Once the selection is performed, the main window of the PECES Service Definition Tool appears on the Eclipse IDE, showing the following options such as Device, Scope and Implemented functions. This tool permits the developer to define the interface that the service will offer to its clients (i.e. the functions that will be available to them). This definitions follow a format that is similar to any Java function. It means, the final function will have the following structure: [Returns] [Name]([Parameters]). For instance, "void getGuideLocation()".

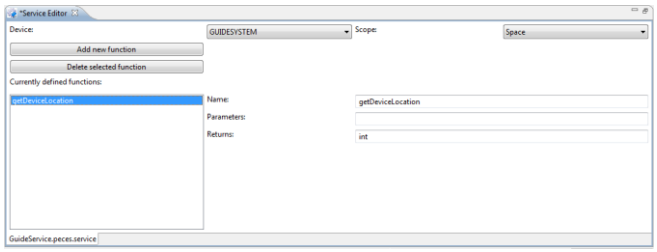

Figure 8: Screenshot of the Service Definition Tool

The PECES development tools are designed to provide support for both secure and unsecure version of the PECES middleware application development. So far the device projects are configured for unsecure version of the PECES middleware. If the application developers are interested in building secure middleware applications, then they should use the Peces Security Configuration Tool to configure secure middleware application and generate necessary keys and certificates for the devices.

F. PECES Security Configuration Tool

The PECES middleware uses the OpenSSL library to create necessary certificates and keys. As a result, the Security Configuration Tool integrates the OpenSSL toolkit to enable application developers to generate keys and certificates for smart space applications. The Security Configuration Tool provides an interface to gather necessary information for root certificates, intermediate certificates (trust chain) and client certificates. The necessary information gathered from the Java interface is passed to the OpenSSL command line interface with the use of AutoIT script files. The AutoIT "Send" command is used to send information. Figure 9 shows the interface needed to generate a root certificate. Developers should first generate a root certificate and can then generate necessary trust chain and client certificates.

Once the root certificate is generated successfully, the name of the security root certificate appears as a tree structure in the root Certificate section. To generate the first trust chain, developers should select the root certificate and then click on the "Trust Chain" button which provides an interface for trust chain configuration. More trust chains can be added as required for the application development.

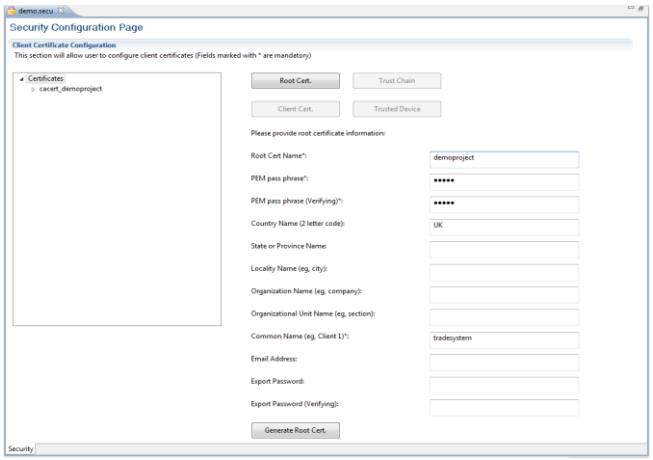

Figure 9: Screenshot of the Security Configuration Tool – Root Certificate Creation

Once the necessary certificate chains are created, they appear as trees in the Certificates area. To generate a Client certificate, developers must select the appropriate trust chain in the tree, and then click on the "Client. Cert" button to generate client certificate. The new interface provides feature to select the device for client certificate configuration. For example, here in Figure 10, the *GUIDESYSTEM* is selected as the device where the certificate will be deployed. When the process is completed, all necessary root and intermediate certificates are deployed in the "full" folder (full trust) in the certificate folder and keys and client certificates are also deployed in the certificate folder as shown in the Figure 10 below.

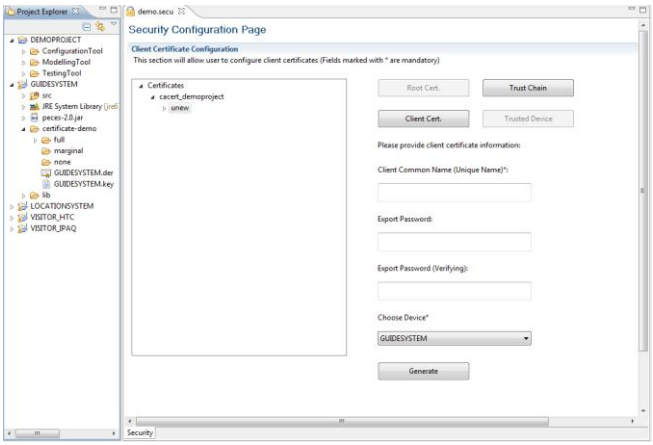

Figure 10: Screenshot of the Security Configuration Tool – Certificate Creation

The tool also provides a mechanism for a device to deploy certificates to other devices which are to be trusted. For example, the *GUIDESYSTEM* device can specify that the *LOCATIONSYSTEM* is to be trusted by copying the necessary certificates to the device.

V. CONCLUSIONS

One of the main objectives of the PECES middleware is to provide a cooperation layer that enables seamless interaction and coordination among devices in and across smart spaces in a secure manner. This paper presented the PECES middleware and a set of tools which provide support to build PECES middleware based application. The tools provided support for device configuration, ontology instantiation, security configuration, role specification and service definition.

ACKNOWLEDGMENTS

The work presented here is sponsored by European Commission under FP7 Programme (Grant agreement no FP7-224342-ICT-2007-2) and we would like to thank PECES project partners for their contributions.

REFERENCES

[1] PECES Project, http://www.ict-peces.eu, last accessed November 2011

[2] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: Towards Distraction-Free Pervasive Computing", IEEE Pervasive Computing, vol. 1, no. 2, pp. 22-31, Apr. 2002.

[3] C. Becker, G. Schiele, H. Gubbels, and K. Rothermel, "BASE - A Micro-broker-based Middleware For Pervasive computing", Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communication, pp. 443-451, Fort Worth, USA, March 2003.

[4] N. Dimakis, J. K. Soldatos, L. Polymenakos, P. Fleury, J. Curin, and J. Kleindienst, "Integrated Development of Context-Aware Applications in Smart Spaces" IEEE Pervasive Computing, vol. 7, no. 4, pp. 71- 79, Dec 2008.

[5] H. Chen, F. Perich, T. Finin, and A. Joshi. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In the 1st Annual Int'l Conf. on Mobile and Ubiquitous systems:Networking and Services, Aug. 2004.

[6] PECES Consortium, *PECES Use-Case Specification*, Deliverable D 1.2, PAS, http://www.ict-peces.eu, last accessed July 2010.

[7] PECES Consortium, *PECES Context Ontology and Query Specification, Deliverable D 2.1*, PAS, http://www.ict-peces.eu, last accessed July 2011.

[8] PECES Consortium, *PECES Addressing Scheme Specification, Deliverable D.3.1*, PAS, http://www.ict-peces.eu, last accessed July 2011

[9] *PECES Consortium, PECES Communication Mechanisms and Registry Interface Specification*, Deliverable D.3.2, PAS, http://www.ict-peces.eu, last accessed July 2011

[10] Eclipse IDE, *Eclipse Website*, http://www.eclipse.org/,, last accessed July 2011

[11] PECES Consortium, *PECES Secure Middleware Specification*, Deliverable D 4.1, PAS, http://www.ict-peces.eu, last accessed , last accessed July 2011

[12] PECES Ontologies, http://www.ict-peces.eu/ont/, last accessed July 2011

Neil Speirs obtained a 1st class Honours degree in Mathematics from Newcastle University in 1980 and a doctorate in Theoretical Physics from the University of Durham in 1985. Since 1987, he has been at the University of Newcastle upon Tyne where he is currently a Senior Lecturer in Computing Science. His main research interests are in fault-tolerance, reliability and mobile distributed systems. He is the Newcastle University Project Manager on the EU PECES Project.

Kirusnapillai Selvarajah completed his Ph.D. in Automatic Control and Systems Engineering at Sheffield University in August 2006. He obtained his B.Sc (Eng) with 1st class honours from the University of Moratuwa, Sri Lanka in 2001. He worked as a Lecturer in the Dept. of Electrical Engineering at the University of Moratuwa, Sri Lanka from 2001 to 2002. His research interests are Wireless Sensor Networks, Pervasive Computing, Embedded Systems and Swarm Intelligence. He has worked as a Research Associate on the EU PECES project and previously on the EU EMMA project.

Ran Zhao is a doctoral student in Computing Science at Newcastle University. Before he started his Ph.D. study, he completed his M.Sc. in Computing Science from Newcastle University in 2007. Before that, he received his B.Eng. in Software Engineering from South China University of Technology, China.