TOOLS FOR BUILDING CONTEX AWARE AND SECURE PERVASIVE COMPUTING APPLICATIONS

Ran Zhao, Kirusnapillai Selvarajah and Neil Speirs
School of Computing Science, Newcastle University, Newcastle NE1 7RU, U.K.

Keywords: Smart Space, Middleware, Pervasive Computing, Wireless Networking, Context Ontologies, Modelling, Testing, Dynamic Addressing, Communication Gateway, Eclipse.

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. The PErvasive Computing in Embedded Systems (PECES) project has developed 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 PECES project has also developed a set of tools which enable application developers to build and test context aware and secure pervasive computing applications. This paper introduces the PECES middleware that consists of flexible context ontology and a middleware that is capable of dynamically forming execution environments that are secure and trustworthy. This paper presents a set of tools, namely Peces Project Tool, Peces Device Definition Tool, Peces Ontology Instantiation Tool, Peces Security Configuration Tool, Peces Event Editor Tool, Peces Event Diagram Tool and Peces Testing Tool which enables developers to build and test pervasive computing application with the PECES middleware.

1 PECES PROJECT

The objective of the PECES project (PECES project, 2009) 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 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 such as ambient intelligence, ubiquitous and pervasive computing. The benefits of these visions and their undeniable impact on the economy and society have led to a number of research and development efforts. These include various European projects such as EMMA (EMMA, 2006) that that develop specialized middleware abstractions for different application areas such as automotive and traffic control systems or home automation. These efforts 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 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.

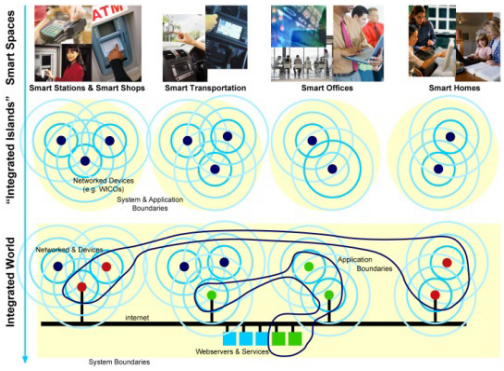

Figure 1: Pervasive computing vision.

Zhao R., Selvarajah K. and Speirs N..
TOOLS FOR BUILDING CONTEX AWARE AND SECURE PERVASIVE COMPUTING APPLICATIONS.
DOI: 10.5220/0003823601420149
In *Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems* (PECCS-2012), pages 142-149
ISBN: 978-989-8565-00-6

PECES is committed to developing 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 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.

Three different prototype applications have been identified to evaluate the novel features provided by the PECES project (PECES project, 2009). To enable the application devlopment process, a set of tools have been developed with the PECES project consortium. This paper introduces the PECES middleware and context ontologies and then focuses on the development tools that have been developed by the project consortium to enable PECES middleware based application development and testing. Section 2 introduces the ontologies developed for the PECES project prototype applications. Section 3 describes the PECES middleware and its novel features such as role specification, dynamic addressing, local and remote gateway and security concepts. Section 4 describes related work of development environment for pervasive computing applications. The PECES developments tools are discussed in Section 5. Conclusions are then presented in Section 6.

2 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 domain and relations among them. They offer powerful mechanisms for defining, acquiring, understanding, processing and sharing context and inferring new knowledge based on available data and context.

The PECES project has developed 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. Figure 2 shows an example of smart space ontologies developed for the project. The PECES ontologies are freely available from (PECES project, 2009) and detailed information about the PECES ontologies can be found in (PECES project, 2009).

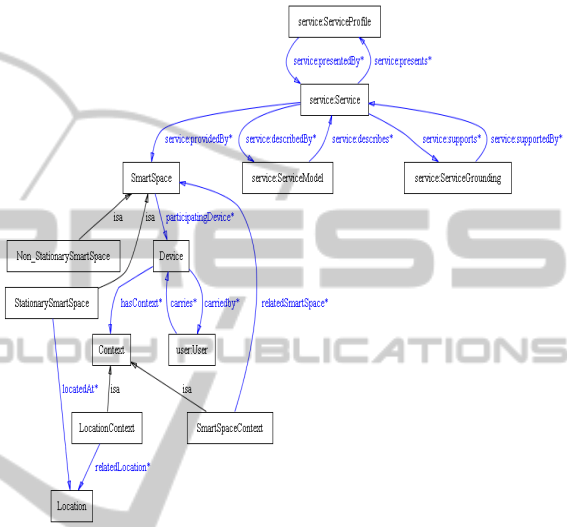

Figure 2: Contextual concepts within a smart space.

The basic concepts to model the contextual information of a smart space are Device, Context, SmartSpace, Location and Service. 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 Non_StationarySmartSpace and StationarySmartSpace. The StationarySmartSpace concept represents smart spaces having fixed location and the Non_StationarySmartSpace concept represents mobile smart spaces. To express a smart space provides a service, a Service instance is referred by a SmartSpace instance using service:provides property. The service:providedBy property is used to express the fact that a service is provided by a smart space.

3 PECES MIDDLEWARE

The PECES project consortium built the targeted cooperation layer on top of BASE middleware (Becker, Schiele, Gubbels and Rothermel, 2003). This enabled the project consortium to focus the development efforts on the novel and innovative features of the PECES middleware. The BASE middleware 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.However, 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.

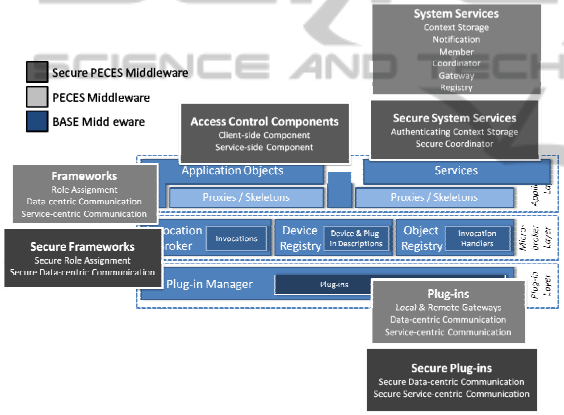

Figure 3: PECES middleware architecture.

The extension of the BASE middleware focused communication gateway concepts, addressing concepts and smart space and security concepts. Figure 3 above shows the BASE components as well as the new features added on top of that to achieve PECES project goal. More detailed information about the PECES middleware can be found in (PECES project, 2009).

PECES middleware supports 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 detail information about the PECES Registry Interface can be found in (PECES project, 2009).

3.1 Generic Role Assignment Concept

The extended PECES middleware provides configuration and adaptation support by means of generic role assignment. 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. More detailed information about role assignment concepts can be found in (PECES project, 2009)

3.2 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 (PECES project, 2009).

3.3 Security Concept

The PECES consortium extends the PECES middleware to derive a secure middleware. For this, PECES consortium introduced 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 that have been identified in the PECES Requirements Specification and thus, they are sufficient to be applicable to a broad range of scenarios.

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 consortium provides set of tools (will be discussed in this paper later) that simplify basic security related tasks such as the distribution of keys and certificates during application development.

4 RELATED WORK

A Middleware based application framework for Active Space applications was proposed in (Roman and Campbell, 2003). The Active Space consists of the Gaia middleware OS (Roman and Campbell, 2000) managing a distributed system composed of different kinds of system. The framework focuses on providing an application framework that leverages the functionality provided by the Gaia middleware OS to assist developers in the construction of Active Space application.

The Distributed Trust Toolkit (DTT) (Lagesse, Kumar, Paluska and Wright, 2009) proposed a framework for implementing and evaluating trust mechanisms in pervasive computing systems and introduced two new abstractions: trust groups and trust blocks.

UbiWise, a simulator for ubiquitous computing system was proposed in (Barton and Vijayaraghavan, 2002). UbiWise concentrates on computation and communication devices situated within their physical environments. UbiREAL simulator (Nishikawa, et al., 2006) was proposed for realistic smart space systematic testing.

A suite of tools used to construct domain models and knowledge-based applications with ontologies was provided by Protégé (Protégé, 2004). Protégé also provide a Java API for other developer to build their own tools and applications.

The tools discussed above provide limited support for application developers, as they have been designed for different goals and concepts. They only offer little methodological support for role assignment, context-awareness and security, which are the core features of the PECES middleware. The PECES development tools aim to empower application developers to make effective use of the concepts and mechanisms provided by the PECES middleware. The following section presents the set of tools that have been developed by the PECES project consortium to enable application developers to build and test context aware and secure smart space applications.

5 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. The development tools suite provides number of tools to support different activities during the application development and testing process. The following sections explain the tool sets which are the Peces Project Tool, the Peces Device Definition Tool, the Peces Ontology Instantiation Tool, the Peces Security Configuration Tool, the Peces Event Editor Tool, the Peces Event Diagram Tool and the Peces Testing Tool. The PECES Development Tools are freely available online (with PECES middleware library) 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.

5.1 Peces Project Tool

This is the first step of 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. The project generated by this tool is used to provide interfaces with others tool via *.xml and *.owl files.

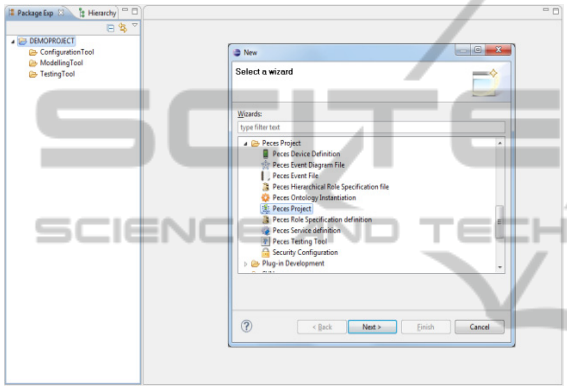

Figure 4: Screenshot of the peces project tool.

5.2 Peces Device Definition Tool

The Peces Device Definition Tool provides a graphical user interface (GUI) for application developers to specify the device description. This tool can be used to define BASE/PECES middleware communication plugins such as IP, Bluetooth, ZigBee (e.g. MxIPBroadcastTransceiver, MxIPMuticastTransceive, EmulationTransceiver), and device functionalities (e.g. Coordinator, Gateway, Coordinator&Gateway, Member) and also device names. Developers can choose the EmulationTransceiver plugin if they would like to emulate their application. The devices can be selected and placed in the Editor area of the tool and necessary functionalities can be defined by right clicking on the devices.

Figure 5 below shows an example application in which four devices are defined (GUIDESYSTEM, LOCATIONSYSTEM, VISITOR_IPAQ, 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 below. The figure also shows four different Java projects which are automatically generated for each devices with necessary PECES middleware library (peces-2.0.jar).

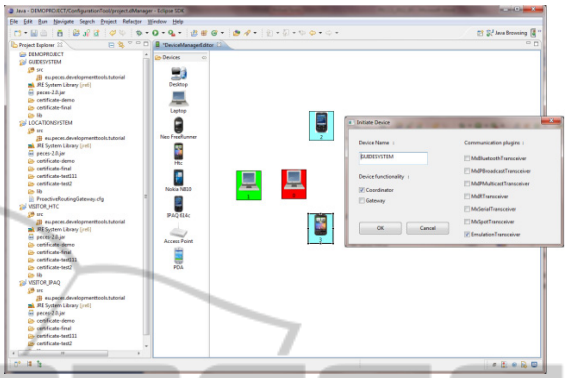

Figure 5: Screenshot of the peces device definition tool.

5.3 Peces Ontology Instantiation Tool

This tool implements the features discussed in the Configuration Tool section of D5.1 and 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 as well as other custom ontologies 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.

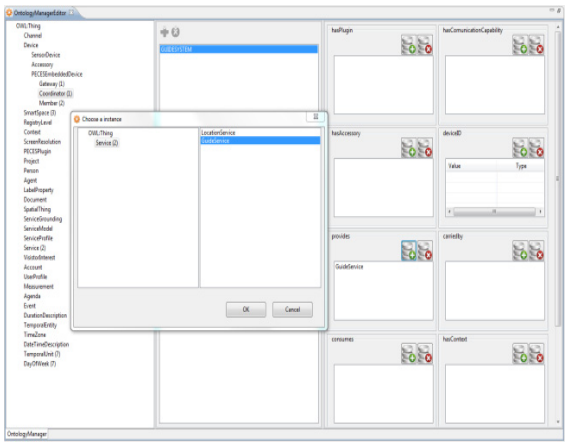

Figure 6: Screenshot of the peces ontology instantiation tool.

5.4 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 application. The Security Configuration Tool provides an interface to gather necessary information for root certificate, intermediate certificate (trust chain) and client certificate. The necessary information gathered from the Java interface is passed to the OpenSSL command line interface with the use of script files.

Figure 7: Screenshot of the peces security configuration tool – root certificate creation.

Once necessary certificate chains are created, they appear as trees in the Certificates section of the tool. To generate a Client certificate, first, developers must select the appropriate trust chain in the tree, and then click on the "Client. Cert" button to generate client certificate. The interface provides feature to select the device for client certificate configuration. 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.

Up to this point, developers have configured Java projects for devices with the PECES secure middleware. The next step is to define the smart space using the PECES Role Specification tool which provides an interface where developers can define the different rules that the application will use to dynamically form groups of collaborative devices. Actually, 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. The final step of the application development process to use the Peces Service Definition Tool to generate the necessary service related Java code and Skeleton and Proxy for any defined services in the Peces Ontology Instantiation Tool.

5.5 Peces Event Editor & Diagram Tools

The PECES Event Editor tool is used to define single event definition. There are five different events can be generated using the Peces Event Editor Tool which are Delay Events, Device SwitchON Event, Device SwitchOFF Event and Context Event and Connection Event.

The Connection Event Editor consists of a full featured graphical editor where application developers can connect or disconnect any devices already defined by the Peces Device Definition Tool. Figure 8 shows a connection event which generated for the example devices defined in the Peces Device Definition Tool.

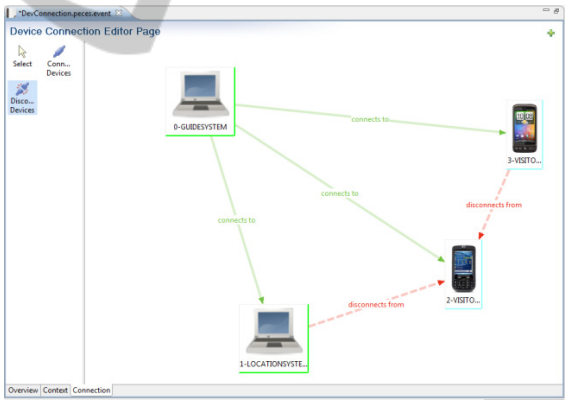

Figure 8: Screenshot of the peces event editor tool.

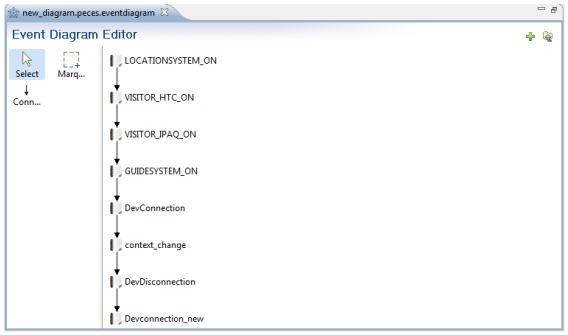

Figure 9: Screenshot of the peces event diagram tool.

After generated all necessary events by the Peces Event Editor Tool, application developers can now use the Event Diagram Tool (Figure 9) to make those events as a sequence. The Event Diagram Tool enables application developer to model smart spaces, dynamics connections and dynamic contexts for testing. This tool generates test cases as a XML file which is used by the Peces Testing Tool during the testing. Figure 9 shows an Event Diagram Tool with the events generated for this example application.

5.6 Peces Testing Tool

The Testing Tool enables application developers to test the modelled application defined by the Peces Event Editor Tool and Peces Event Diagram Tool. For this purpose the Testing Tool generates another new emulator Java project to centrally control other device related events. In addition to the necessary plugins, this new Java project should install the EmulationTransceiver plugin. This emulator Java project (run as new JVM process) is used to control connections (add and remove connections) between devices which is defined by the events.xml. This project is also used to control the context changes (add or remove) of any device. The Testing Tool records all important middleware and application related events in a common log file. The Peces Testing provides three different editor pages: Execute page, TestLog page and Visualise page.

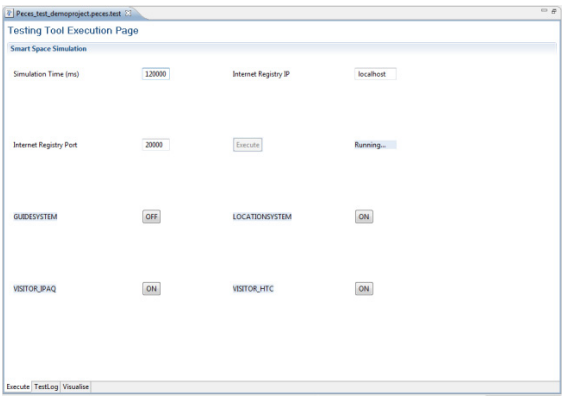

Figure 10: PECES testing tool execute page.

The initial device status information is displayed (for example, all devices are “OFF”) in the Testing Tool Execute Page. Application developers are able to define the required time to test the application specified by the previous tools. Developers can also provide Internet Registry IP and port information if they want to test that application with the internet registry. The Execute” button enables developers to run the application. As seen in Figure 10, the status of each device is shown during test (three devices are “ON” and one device is “OFF” at a particular time).

The TestLog page provides detailed information of the test performed. Middleware and application related events are logged with specific device name, absolute time and other relevant and available information.

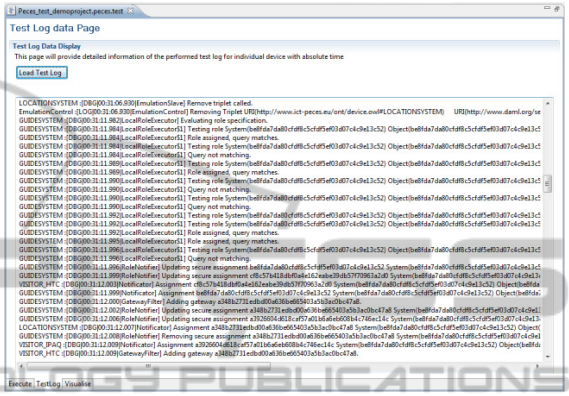

Figure 11: Testing tool TestLog page.

The Testing Tool Visualisation Page provides features to visualise the smart space network status based on the test log data events with relative time. The Page lists analyzed important events occurred during the test. The “List of Events” may contain event such Device Switch ON, Device Switch OFF, Connection, Disconnection and Smart Space Establish, Smart Space Join, etc. The status of the system can be viewed by double clicking on the name of the specific event. Double clicking on the particular event from the “List of Events” provides not only the particular event but also the status of the whole network and its devices, its connections and smart space activities, etc. at a particular time.

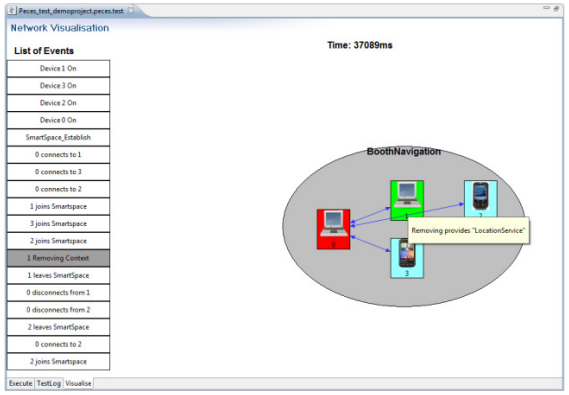

Figure 12: Screenshot of the Visualisation Page just after devices joined the smart space.

Figure 12 shows the visualisation of the system at time 37089 ms (after test started). It displays the four devices which at that time joined with the "BoothNavigation" smart space define in this example application. Figure 13 shows that the *LOCATIONSYSTEM* left the smart space at 42170 ms (after the test started) due to its context property changes ("*LocationService*" was successfully removed).

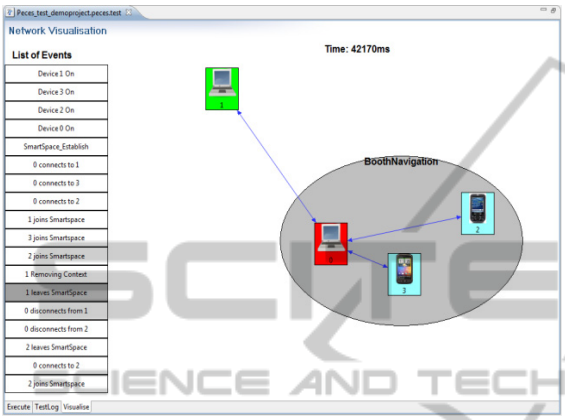

Figure 13: Screenshot of the visualise page just after the "locationservice" context was removed.

6 CONCLUSIONS

The main objectives of the PECES middleware are to provide a cooperation layer that enables seamless interaction and coordination among devices and a development tools suite to build and test the pervasive computing applications. This paper presents a set of tools which provide support for PECES middleware based application development and testing. The tools provide support for device configuration, ontology instantiation, security configuration and role specification. The tools also enable dynamic modelling of the network connections and context changes. Finally, the tools provide support to test the smart space application performance and visualise the test results. It is the authors' intention to present complete set of tools developed for the PECES project as a live demo at the conference.

ACKNOWLEDGEMENTS

The work presented here is sponsored by EC under FP7 programme (FP7-224342-ICT-2007-2) and authors also would like to thank all the project partners for their contributions.

REFERENCES

Barton, J., and Vijayaraghavan, V., 2002. UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment, [online] Available at: <http://www.hpl.hp.com/techreports/2002/HPL-2002-303.html> [Accessed December 2010]

Becker, C., Schiele, G., Gubbels, H., and Rothermel, K., 2003. BASE - A Micro-broker based Middleware For Pervasive Computing, In *IEEE, 1st International Conference on Pervasive Computing and Communications*, pp. 443-451, Fort Worth, USA, March 2003

EMMA Project, (2006). [online] Available at: <http://www.emmaproject.eu> [Accessed May 2011].

Lagesse, B., Kumar, M., Paluska, J., and Wright, M., 2009 DTT: A Distributed Trust Toolkit for Pervasive Systems, In: *IEEE, Pervasive Computing and Communications Conference*. Galveston, Texas, USA, March 9-13, 2009

Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., and Ito, M., UbiREAL: Realistic Smartspace Simulator for Systematic Testing, In: UbiComp. *8th International Conference on Ubiquitous Computing*, LNCS4206, pp. 459-476, Irvine CA, USA, September. 2006.

PECES Project, (2009). [online] Available at: <http://www.ict-peces.eu> [Accessed May 2011].

Protégé. (2010). [online] Available at: <http://protege.stanford.edu/> [Accessed May 2011]

Roman, M., and Campbell, R., 2000. Gaia: Enabling Active Spaces, In: *SIGOPS EW'00, 9th ACM SIGOPS European Worksho*p, pp.229-234, Kolding, Denmark, September 2000. New York: USA.

Roman, M., and Campbell, R., 2003, A Middleware-based Application Framework for Active Space Applications, In: *ACM/IFIP/USENIX, International Conference on Middleware*. Rio de Janeiro, Brazil June 2003

Jeremy J. Carroll., Ian Dickinson., Chris Dollin., Dave Reynolds., Andy Seaborne., Kevin Wilkinson., In *ACM, 13th International World Wide Web conference on Alternate track papers & posters*, New York, NY, USA 2004