

State of the art Smart Spaces: Application Models and Software Infrastructure

Abstract:

Smart spaces are ordinary environments equipped with visual and audio sensing systems, pervasive devices, sensors, and networks that can perceive and react to people, sense ongoing human activities and respond to them. Their ubiquity is evident by the fact that various state of the art smart spaces have been incorporated in all situations of our life. These smart space elements require middleware, standards and interfacing technologies to manage complex interactions between them. Here, we present an overview of the technologies integrated to build Smart Spaces, review the various scenarios in which Smart Spaces have been incorporated by researchers, highlight the requirements of software infrastructure for programming and networking them, and mention the contemporary frameworks for interaction with them.

Keywords:

Smart Spaces technologies
Smart Spaces scenarios
Programming smart spaces

Pervasive computing refers to the emerging trend toward numerous, easily accessible computing devices connected to each other and to an increasingly ubiquitous network infrastructure. This trend will enable companies to place computers and sensors in virtually every piece of equipment in buildings, homes, workplaces, factories, and clothes. The underlying principle of pervasive computing to make information available everywhere and anywhere led to the integration of the various computing devices into a single environment termed a “smart space”.

"**Smart spaces**" are ordinary environments equipped with visual and audio sensing systems that can perceive and react to people without requiring them to wear any special equipment. Pervasive devices, sensors, and networks, provide infrastructure for context-aware smart spaces that sense ongoing human activities and respond to them. Here, we present an overview of the technologies integrated to build Smart Spaces, the various scenarios in which Smart Spaces can be incorporated, software infrastructure for programming and networking them, and interaction with them.

1. Core Technologies

Important technologies include sensor-based collaborative interfaces using:

1.1 Pervasive Devices - An intelligent environment is likely to contain many different device types like wireless mobile devices, such as pagers, personal digital assistants, cell phones, palmtops, and smart devices, such as intelligent appliances (which have built-in active and passive intelligence), floor tiles with embedded sensors, and biosensors.

1.2 Perceptual Interfaces

1.2.1 Speech recognition: Speech or voice recognition is the ability of a machine or program to recognize and carry out voice commands or take dictation. It involves the ability to match a voice pattern against a provided or acquired vocabulary. Usually, a limited vocabulary is provided with a product and the user can record additional words. More sophisticated software has the ability to accept natural speech.

1.2.2 Speaker identification: the system uses speaker-dependent speech recognizers to model each speaker. During training, phonetically transcribed enrollment utterances are used to train context-dependent phonetic models for each speaker. During testing, a speaker-independent speech recognition component generates a phonetic transcription from the test utterance. This transcription is then used by the system to score each segment of speech against each speaker dependent phonetic model.

1.2.3 Gesture recognition: It is human interaction with a computer in which human gestures are recognized by the computer. Face recognition, eye motion, and lip reading are also being considered as ways to provide interaction.

1.2.4 Sensor fusion: The conversion of raw data into high-level context information, such as the user's current activity, requires applications to pre-process the data by filtering, transforming, and even aggregating the same or different types of distributed sensors to improve the quality of the derived context. This is called context fusion.

1.2.5 Gaze tracking: For a system to attend to a user it must not only perceive the user but it must also anticipate the user. The gaze-tracking cameras track the movement of our eyes to comprehend what we wish to be done.

1.3 Mobility and Networking - In an ideal smart environment, a large number of heterogeneous connected smart devices are deployed to collaboratively provision seamless services to users. Hence, Smart Spaces require support for interoperability, scalability, smartness, and invisibility to ensure that users have seamless access to computing whenever they need it.

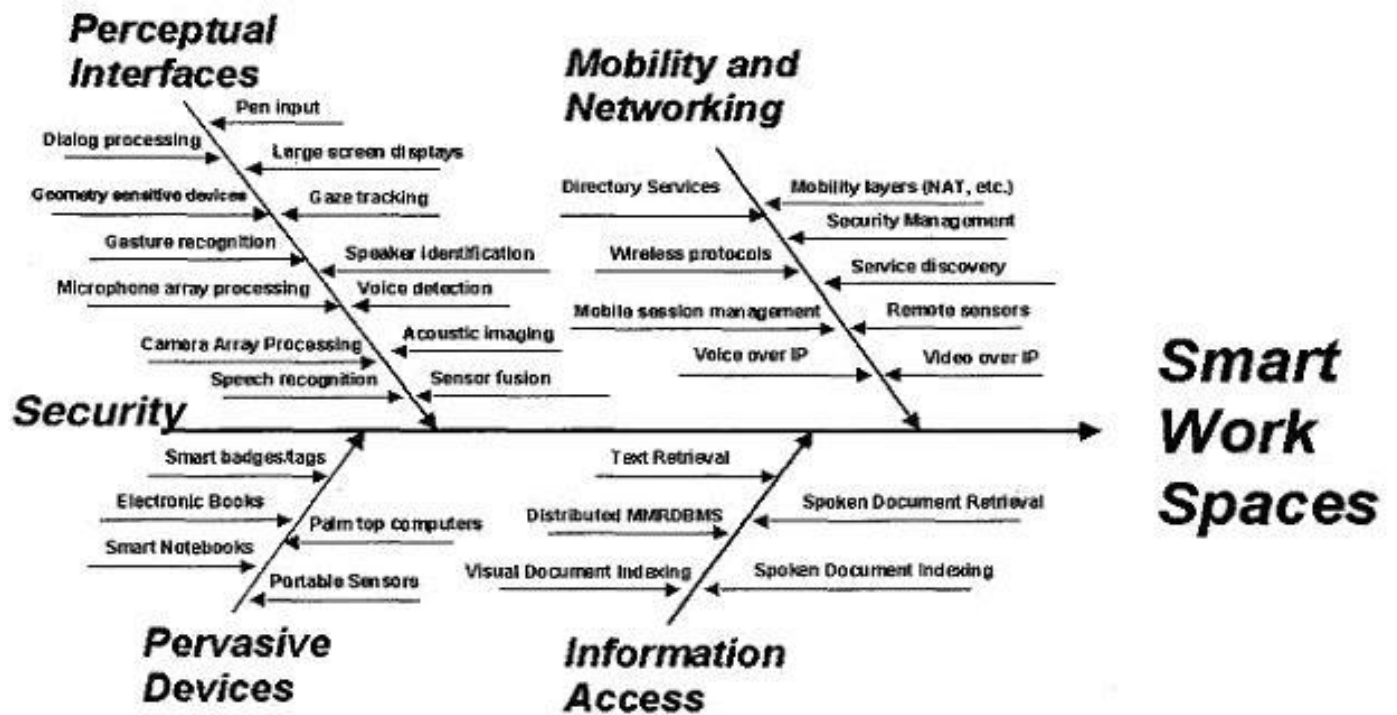
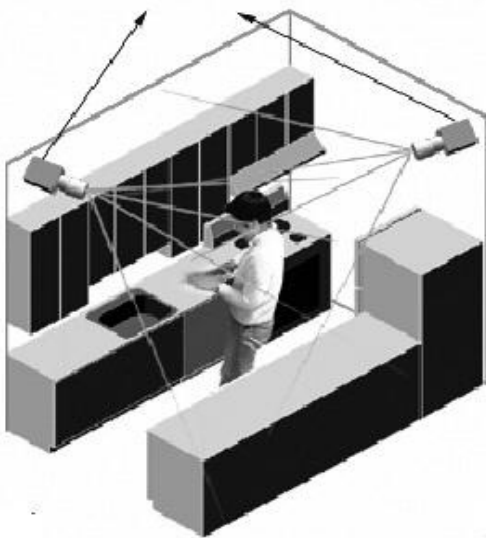


Figure 1: Various technologies integrated into Smart Spaces [1]

2. Smart Spaces Scenarios

2.1. Smart Homes - Researchers are developing a living laboratory [2] to study how people live with technology. The team members have proposed an integrated chassis infilled construction system in which the chassis components incorporate multipurpose “sensor packs.” These are standardized sets of sensors - fixed, wide-angle color cameras, microphones, and temperature sensors - that easily plug into the chassis beams at regular intervals and can provide computing applications with access to data on interior and exterior environmental conditions to support context-aware computing. Visually based real-time people- and object-tracking algorithms for environments that use these sensor packs have been developed.



Researchers will be able to acquire video, audio, and appliance- level data from every part of the environment. The data available can be used for various purposes like acquiring statistics about people's movements around the house, their activities, or even detecting the onset of various ailments like heart failure.

Figure 2: Model of the laboratory [2]

2.2. Interactive Workspaces: A prototype interactive workspace, called an iRoom [3] (interactive Room), contains three touch sensitive white-board displays along the side wall and a custom-built display called the interactive mural. In addition, there is a table with a display designed to look like a standard conference-room table. The room also has cameras, microphones, wireless LAN support, and several wireless buttons and other interaction devices. The infrastructure needed for the iRoom included PDAs, workstations, and laptops being simultaneously used in a workspace with heterogeneous software running on these devices. All of these must be accessible to one another in a standard way so that the user can treat them as a uniform collection i.e. any software framework must provide cross-platform support. The interfaces must be customized to different displays and possibly to different input-output modalities, such as speech and voice.

The iRoom provide three basic facilities to the user: Allow movement of data from one visual application running on a PDA to another, control any device or application from his or her current location, and support coordination of various applications. To provide people with the facility to identify various devices like projectors, screens, lights etc., a room controller used a small map of the room to indicate the lights, projectors, and display surfaces. Sample GUI applications used simple toggles and menus associated with objects in the map to switch video inputs to projectors and to turn lights and projectors on or off, but these were later expanded to make them available to a number of other devices by building them as Web applets and pages.

2.3. Interactive Museums -This work [4] implemented and deployed three computing tools at the museum, The Exploratorium (www.exploratorium.edu): The Informer, which provided the user with information related to the exhibit that the user was visiting; the Suggester offered ideas on implementing the functionality of an exhibit; the Rememberer, helped them to build a record of their experiences, which they can consult during or after their visit.

An electronic guidebook prototype combined the functions of informing, suggesting, and remembering. The three functions were delivered principally in the form of Web pages, which users could access on PDAs while at the exhibits and before and after the visit on any computer. The guidebook prototype provided content about the exhibits to users as they visit them; it also provides them with a record in the form of a personal scrapbook on the Web, for perusal after the visit. The guidebook delivered static Web pages to users' PDAs when they visited exhibits. Each exhibit had a homepage that contained links to other pages, which contained information about the exhibit and the underlying phenomena and suggestions about what to do with the exhibit.

The user picked up an identifier by pointing a handheld device at the physical object of interest. Attaching an identifier on or near the object and installing a sensor in the PDA held by the user or vice versa makes this work. Once the identifier is picked up, it is converted to a URL using a resolution service. The URL is dereferenced, and the resulting Web resource (the physical object's Web presence) is rendered on the PDA's Web browser as physical hyperlink. However, the findings of this project suggested that the users found the prototype to complex for use.

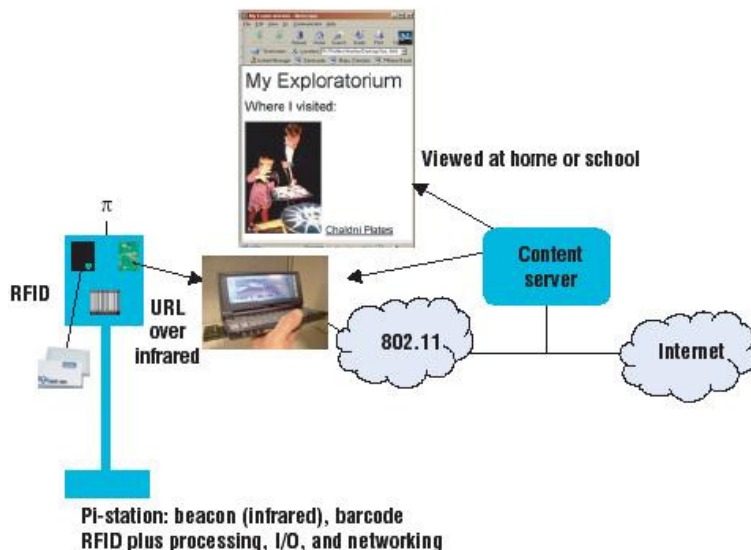


Figure 3: The prototype system [4]

2.4. Smart Classroom - This project [5] integrated involved a classroom having two wall-size projector screens, one board for displaying teaching materials and the other on a side wall which displayed remote students' images (captured by their cameras) and a computer-animated virtual assistant.. Remote students accessing the classroom via their computers can view the board's content via a client program. The classroom is equipped with several cameras which capture the teacher's actions (which are recognized and translated into an interaction command to the underlying tele-education software); and

live video of the classroom, (which the system broadcasts to remote students). The teacher also wears a wireless microphone to capture his or her speech.

A wall-size touch-sensitive SmartBoard device was used as the display screen. Teachers can use digital pens and erasers to write on the board or remove notes. Remote and local students both see the same lecture materials or notes on the board. A laser pointer served as an indicating tool and when the teacher pointed the laser pointer to highlight a point in the lecture, a red spot appeared at the same position on the remote students' view. A text-to-speech module notifies the teacher when certain events occurred. Biometric verification - face recognition and speaker-verification - is used to authorize teachers for using it.

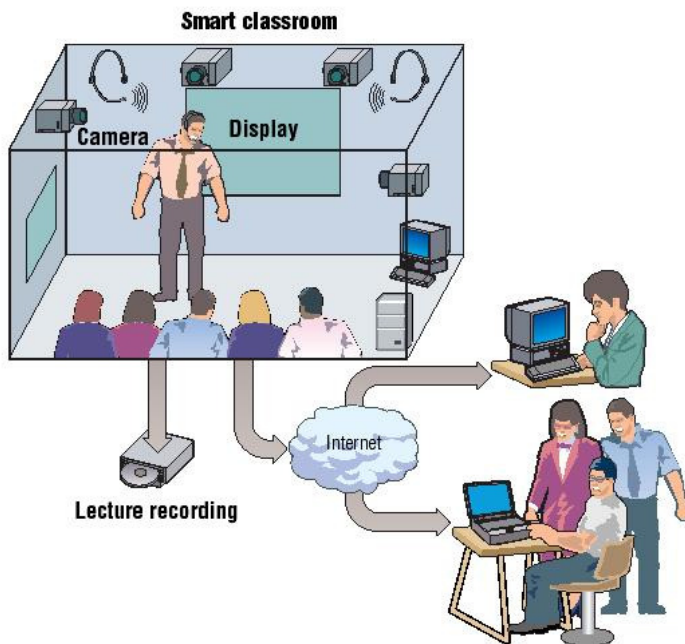


Figure 4: The Smart Classroom [10]

2.5. Gardens - UNSEEN [6], a knowledge-sharing experience machine for outdoor public spaces, collected real-time data on plants, insects, and animals through a multicamera vision system. UNSEEN had only image data as input. The site design consisted of nine planting areas arranged as eight parallel strips overseen by a four-camera machine vision system.

The system operated in three modes. In the first mode, all four cameras were checked for the highest degree of activity among the plant strips. Whichever camera registered the highest degree of change temporarily received the computing system's complete attention. The system then displayed the unaltered image stream on all three screens simultaneously in real time for visitors to observe. The second mode compared the garden's current state to previous states by the taking an image from each garden every afternoon for over a period of time and then comparing them. The third mode aimed to vary plant descriptions over time as a function of growth density and to speculate on the plants' future by detecting flowering of the plants. These dynamics of the various plants in the garden, as measured by the system, along with factual information published on them became the source of information to visitors.

2.6. Vineyards - Vineyards [7] have been turned into Smart Spaces by deploying a network of sensors to observe temperature data readings and to draw conclusions from that data regarding the spraying of pesticides or any other action that should be taken on the basis of that data. Also, models of systems have been developed which not only interpret data in a form meaningful to the vineyard owners but also suggest steps that should be taken or are automated to carry out the necessary steps.

3. Programming Smart Spaces

3.1 Why do we need to program Smart Spaces?

Any pervasive system consists of a number of heterogeneous elements that require integration. This integration is non-scalable and slow with the introduction of a new element leading to repeated testing of the system. Also, it is quite difficult to introduce a new element into a fully functioning closely integrated smart environment because of the unpredictability of the response of the elements already present in the system. Once a smart environment has been built keeping in mind a particular scenario - homes, classroom, labs, and gardens - it is quite difficult to incorporate new technology and concepts in it without disturbing the status quo.

Middleware and standards are needed which will enable developers to build applications independent of the physical elements like sensors et al. Thus, middleware should have the capability to support the elements of a smart environment

integrate themselves automatically into the environment without them being explicitly programmed. This self integration requires standards and protocols like UPnP and OSGi. The middleware should also be able to exploit semantics to provide additional services. The most significant aspect of middleware should be to provide the programmers a runtime environment with all physical elements of the Smart Space as ready to use services. This in turn accelerates prototyping and the development life cycle.

The software infrastructure of smart spaces should be transparent to the developers in a way that they should not be concerned with program and data migration or synchronization and coordination of distributed components. It should support context awareness by facilitating the gathering of information from sources such as sensors and resource monitors; performing interpretation of data; carrying out dissemination of contextual information to interested parties in a scalable and timely fashion; and providing models for programming context-aware applications. The infrastructure will be required to integrate software components, which may reside in fundamentally different environments, and even on different devices into compositions that can together achieve various tasks. It should be capable of supporting rapid development of specialized applications, such as agents, by the utilization of a framework of special-purpose languages that enable applications to be specified at a very high level of abstraction. The infrastructure should be powerful enough to support scalable, fault tolerant and distributed components.

3.2 The Solution

To address these issues mentioned, there is a need for middleware, networking and a software framework for interaction among the various pervasive devices that together constitute the Smart Space.

3.2.1 The **Open Services Gateway Initiative (OSGi)** (www.osgi.org) [8] is a platform and vendor independent framework to connect various devices in a local network and which provides general purpose support for Java applications called "Bundles". OSGi provides a shared execution environment that installs, updates, and uninstalls bundles without needing to restart the system. It defines only interfaces between the framework and services thus leaving the actual implementation to the developers.

3.2.2 An **Obj Interoperability Framework** [9] developed in Java facilitates networking all the components of a smart environment obliterates the need to program each device to recognize another device that it interacts with. Obj limits the categorization of device capabilities thus defining a general functionality of each device without specifying particular details for any device. This provides better compatibility between various devices.

3.2.3 To manage complex interactions among the interconnected computers and devices that make up a Smart Space, the two major approaches are: **universal user interface languages** and **user interface remoting**.

Universal UI languages describe user interfaces that are rendered by mapping a description's device-independent interaction elements to a target platform's concrete interface objects. These enable applications to be specified at a high level of abstraction. Universal UI languages include:

- **W3C XForms** (<http://www.w3.org/TR/2003/REC-xforms-20031014/>), which separate content from presentation. These provide support for structured form data and seamless integration with other XML tag sets.
- **Alternate Abstract Interface Markup Language (AAIML)**, developed by the International Committee for Information Technology Standards Universal Remote Console (INCITS/V2URC) (<http://v2.incits.org/>), is an XML-based language that would be used to communicate an abstract user interface definition for a service or device to a user's personal device which could act as a Universal Remote Console (URC).
- **AUIML** (Abstract User Interface Markup Language), an XML vocabulary which has been designed to allow the intent of an interaction with a user to be defined.
- **Extensible Interface Markup Language (XIML)** (<http://www.xml.org/>), an XML-based interface representation language for universal support of functionality across the entire lifecycle of a user interface: design, development, operation, management, organization, and evaluation
- **Extensible User Interface Language (XUL)** (<http://www.mozilla.org/projects/xul/>), standards-based interface definition language
- **Microsoft Extensible Application Markup Language (XAML)** (<http://longhorn.msdn.microsoft.com/lhsdk/core/overviews/about%20xaml.aspx>), which is based on Extensible Markup Language (XML) and enables developers to specify a hierarchy of objects with a set of properties and logic.
- **OASIS User Interface Markup Language (UIML)** Specification Technical Committee (<http://www.oasis-open.org/committees/uiml/>), an abstract meta-language that can provide a canonical XML representation of any user interface (UI).

- **User Interface eXtensible Markup Language (UsiXML)** (<http://www.usixml.org/>), a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces.
- **User Interface Markup Language (UIML)** (<http://www.uiml.org/index.php>) , it is designed to serve as a single language which permits creation of user interfaces for any device, any target language.

With **UI remotng**, user interfaces reside on a remote platform instead of the one running their applications. Some of the service discovery and management frameworks are:

UPnP: An extension of the Plug and Play (PnP) (www.upnp.org) initiative started by Microsoft, UPnP is an architecture for pervasive peer-to-peer network connectivity of intelligent appliances and wireless devices. It is designed to support zero-configuration, almost invisible networking, and automatic discovery for a breadth of device categories from a wide range of vendors. This means a device can dynamically join a network, obtain an IP address, convey its capabilities, and learn about the presence and capabilities of other devices. Finally, a device can leave a network smoothly and automatically without leaving any unwanted state behind.

No device drivers and common protocols are used instead. UPnP networking is media independent. UPnP devices can be implemented using any programming language, and on any operating system. UPnP does not specify or constrain the design of an API for applications running on control points; OS vendors may create APIs that suit their customer's needs. UPnP enables vendor control over device UI and interaction using the browser as well as conventional application programmatic control.

Jini (www.jini.org): This is a Java-based technology defined by Sun Microsystems. When Jini-enabled devices connect to networks, they establish impromptu Java-oriented networks that let users immediately access network resources and services. Devices register with the network when they connect, which makes them available to other devices. For example, when a printer is attached and gets registered, it makes its driver available on the network and this driver gets downloaded to clients when they need to use the printer.

Jini is based on Sun's Java language and uses Remote Method Invocation (RMI) as a major part of its communication system. RMI allows the passing of not only data through the network, but code as well. Jini uses Java marshalled objects to pass object references across the network. A marshalled object will contain the data to represent an instance of an object as well as a location to download the object definition if not available locally. As long as a device has a JVM installed or access to another network computer that can execute the Java code for them, it is relatively simple to set up a basic system since the Jini and Java APIs are encapsulating a lot of the basic leasing, underlying transport operations, and concurrency details. Jini should work with any network transfer protocol that has a JVM to support the network communications.

Salutation: Salutation (www.salutation.org) is a technique for service discovery and service management. The Salutation architecture is a royalty-free service discovery and service management product from the Salutation Consortium, a nonprofit corporation. Salutation is an open standard, independent of operating system, communications protocol, hardware platform, or vendor-imposed limitations. It was created to provide service discovery for a broad range of network appliances and equipment in a platform-, OS-, and network-independent environment. Devices can use it to advertise and describe their capabilities and discover the capabilities of other devices by using search features.

Salutation works around Salutation Managers (SLMs). The SLMs may be on the device or they may just be accessible by the device. The SLMs handle all the communication between devices by talking with other SLMs. They keep track of where the other SLMs are and what services are at that location. If any data need to be transferred between devices or function calls made to and from devices, this is done through the SLMs with Remote Procedure Call (RPC) and External Data Representation.

Microsoft.NET: The Microsoft.NET platform for Web Services is a development environment based on building applications with Web Services. The technique is similar to building distributed objects, but is based on HTTP and XML. Data is represented with XML and delivered in SOAP (Simple Object Access Protocol) messages via HTTP. A language called WSDL (Web Services Description Language) is used to describe services. An XML-based protocol called Disco is used to discover services at a site and a mechanism called UDDI (Universal Description, Discovery, and Integration) defines how to advertise services and how Web Service consumers can find services.

4. References:

- [1]. Lynne Rosenthal, Vincent Stanford, "NIST Smart Space: Pervasive Computer Initiative," p.6, *IEEE 9th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 2000
- [2].Intille, S.S., "Designing a home of the future," *IEEE Pervasive Computing*, vol.1, no. 2, 2002, pp. 76–82.
- [3].Johanson, B.; Fox, A.; Winograd, T., "The Interactive Workspaces project: experiences with ubiquitous computing rooms," *IEEE Pervasive Computing*, vol.1, no. 2, 2002, pp. 67–74
- [4]. Fleck, M.; Frid, M.; Kindberg, T.; O'Brien-Strain, E.; Rajani, R.; Spasojevic, M., "From informing to remembering: ubiquitous systems in interactive museums," *IEEE Pervasive Computing*, vol.1, no. 2, 2002, pp. 13–21
- [5]. Yuanchun Shi; Weikai Xie; Guangyou Xu; Runting Shi; Enyi Chen; Yanhua Mao; Fang Liu, "The smart classroom: merging technologies for seamless tele-education," *IEEE Pervasive Computing*, vol.2, no. 2, 2003, pp. 67–74
- [6] Bohlen, M.; Tan, N., "Garden variety pervasive computing," *IEEE Pervasive Computing*, vol.3, no. 1, 2004, pp. 29–34
- [7].Burrell, J.; Brooke, T.; Beckwith, R., "Vineyard computing: sensor networks in agricultural production," *IEEE Pervasive Computing*, vol.3, no. 1, 2004, pp. 38–45
- [8]. Choonhwa Lee; Nordstedt, D.; Helal, S., "Enabling smart spaces with OSGi," *IEEE Pervasive Computing*, vol.2, no. 3, 2003, pp. 89–94
- [9]. Edwards, W.K.; Newman, M.W.; Seciivy, J.Z.; Smith, T.F., "Bringing network effects to pervasive spaces," *IEEE Pervasive Computing*, vol.4, no. 3, 2005, pp. 15-17
- [10].Helal, S., "Programming pervasive spaces," *IEEE Pervasive Computing*, vol.4, no. 1, 2005, pp. 84-87
- [11]. Choonhwa Lee; Helal, S.; Wonjun Lee., "Universal interactions with smart spaces," *IEEE Pervasive Computing*, vol.5, no. 1, 2006, pp. 16-21
- [12] <http://xml.coverpages.org/userInterfaceXML.html>