© 2004 VSB-TUO, Ostrava, ISSN 1562-3580

A Flexible Software Middleware for Interactive Learning Environments

Dirk GROENEVELD
Dept. of Computer Science, University of California, Irvine, dgroenev@uci.edu

Tara C. HUTCHINSON
Dept. of Civil and Environmental Engineering, University of California, Irvine, thutchin@uci.edu

Falko KUESTER
Dept. of Electrical Eng. and Computer Science, University of California, Irvine, fkuester@uci.edu

KEYWORDS: *digital classrooms, middleware, device integration*

ABSTRACT: *While the outside world is becoming ever more complex, classroom environments have barely changed over hundreds of years and students now bring cell phones, PDAs and laptops into rooms that are still dominated by chalk boards. The introduction of digital interactive classrooms finally provides us with a means to close this technology gap while creating an opportunity to transform many students that currently prefer to be passive listeners to students that interact with the instructor, their peers and the course material on different levels suitable for individual learning styles. The presented VizClass project is working towards creating these capabilities by creating a ubiquitous computing environment specifically geared towards engineering students. VizClass is equipped with a number of different devices such as touch screens and large 3D projection surfaces, and the corresponding software middleware, while keeping the overall complexity of using this environment at a level comparable to that of markers, whiteboards and erasers. Application development for this environment faces many of the challenges that are common to digital rooms. The software has to deal with numerous concurrent events and remain responsive enough for interactive work even in large installations. At the same time, it has to support a wide variety of different devices, ranging from the students PDAs over tablet PCs to wall-sized touch screens. It also has to remain reliable in order to facilitate, rather than hinder, the instruction of possibly hundreds of students every day. This paper describes VizION, the middleware layer that has been created to ease software development for collaborative digital classroom environments. Following the example of distributed computer networks, it supports an arbitrary number of devices, called nodes, for data input, output and processing. System usability and the overall quality of service, including responsiveness and failure tolerance, are fundamentally important since VizClass is intended to replace the traditional low-tech, yet reliable chalkboard-based environment.*

1 INTRODUCTION

1.1 Motivation

The increasingly pervasive nature of computer technology is resulting in their nearly transparent integration into homes, offices and schools. In particular at Universities the pervasive nature of information technology has created a challenging divide between chalkboard-based classroom environments that have changed relatively little over the past hundred years and the technology that students are taking for granted in their daily life. PocketPCs, laptops, mobile phones and pagers are just a few of the devices that have made it into our work environment and that currently have to be shed when we transition into the traditional classroom. However, the integration of some of these devices into our teaching environment can have a broad range of benefits, such as real-time student-teacher interaction on digital contents, real-time student confidence evaluation via surveys and quizzes or collaborative work between big student teams using networked devices, just to name a few. This has resulted in increased attention being paid towards the development of technology based environments requiring a middleware layer that allows control and coordination of many heterogeneous devices, while satisfying a multitude of scalability and quality of- service requirements. In particular, the challenge of combining heterogeneous components into a homogenous system that is easy to use by educators and students and encourages developers to build upon it has received increasing attention. The underlying middleware has to be able to seamlessly interface with existing drivers and software, avoiding replication of code, while communicating with device drivers and applications that are not openly accessible. At the same time, the

middleware has to be flexible enough to accommodate a broad range of different devices that may become part of a pervasive computing system, ranging from USB sticks, handheld devices and laptops all the way to wall-sized projection systems. Consequently, such systems should be able to cope with components running on a variety platforms and operating systems. Furthermore, ubiquitous computing environments tend to be very volatile. Parts might enter or leave the system without warning, some 1 appliances might fail unexpectedly, while most components will be connected with something as unreliable as a wireless or wired network. In this environment, the middleware has to ensure the correct operation of all devices, which have to remain unaffected by change, at a level that goes unnoticed by the end user of the system. To address these challenges we have developed VizION, the Interface Operating Network, which provides a stable framework for the deployment of information technology classrooms.

1.2 Related work

Over the past five years, a number of information technology (IT) environments have been developed at Universities and research laboratories to provide testbeds for the development of new software systems that can facilitate the integration of new modes of instruction and information sharing. These environments are aimed at infrastructure support for classrooms, office activities, tele-conferencing and meeting spaces. Examples include the iRoom Interactive Workspaces Project [1], eClass [2] and the Office of the Future project [3]. These and other projects require the development of dedicated middleware in support of the ubiquitous computing environment, such as iROS [4], GAIA [5] and PICO [6].

The Interactive Workspace Project, more commonly termed iRoom at Stanford University [1] uses a broad array of interactive devices, such as multiple large 2D and 3D displays and hand held and other portable devices to provide an interactive research working space. A software layer developed by Shankar et al. [4], termed iROS (Interactive Room Operating System), is used to provide the underlying middleware architecture integrating the different interaction devices and computational activities together. The iRoom project brings together interaction at a broad range of scales while using a variety of interfacing techniques. Both wired and wireless devices are integrated into the environment. For example, the user can control multiple displays with a handheld device such as an iPAQ, or remotely activate devices during working sessions, such as scanners, printers, etc.

The eClass project, formerly called Classroom2000 [2], has been developed by the Future Computing Environments group at the Georgia Institute of Technology. Since 1995, eClass has supported teaching at Georgia Tech in the areas of Computer Science and Engineering, by providing a digital environment for course lectures and discussion sessions. The environment supports real-time playing, recording, and animation of lectures. A java-based client-server system, ZenPad, records and stores lecture video clips, slides, and annotations during each lecture. Lectures collected and compiled by ZenPad provide a detailed record of the discussion, including the original presentation, annotations compiled during the presentation, and a full audio record of the session. eClass has a unique 2D display layout for lecturers, primarily including a digital active LiveBoard, and two projection surfaces for passive display of information.

The Office of the Future project was developed at University of North Carolina at Chapel Hill to provide an immersive space allowing 3D graphical model display and manipulation for the future office environment. The primary focus of the office of the future project is to capture and display 3D data dynamically and on random surfaces within the space it is constructed in and to broadcast this data in real-time to remote collaborators.

These and other systems are providing state-of-the-art interactive digital workplaces, whether it is for an office, educational, or entirely research environments. The different middleware frameworks address the same challenges, while being targeted towards different applications. For example, iROS uses a centralized event heap for communication between the different parts of the system [7]. In contrast, our VizION framework uses a highly distributed approach that avoids centralized entities wherever possible. A similar approach is supported by PICO. However, PICO goes beyond the pervasive computing room and targets large installations the size of entire cities. GAIA is highly distributed and focuses on room-size installations and implements many features as part of the framework that in our case are simply applications running on top of our VizION framework.

2 VIZCLASS - A PILOT PROJECT

The testbed environment for VizION is VizClass, our ubiquitous computing classroom [8]. The room offers two main modes of presentation that can be combined arbitrarily, a 2D system and a 3D system. The 2D system consists of three large digital whiteboards that are mounted side-by-side to provide a large (4.4m x 1m) working environment. These whiteboards can simultaneously function as a computer screen and as a blackboard, and are the main device facilitating data input in VizClass. The 3D system consists of a passive stereo display with an effective projection surface of 2.4m by 1.8m. Users can interact with all displays either through wireless mice and keyboards, or in the case of the touch sensitive digital whiteboards, via touch events registered by the screen.

3 VIZION CONTROL LAYER

3.1 A System of Nodes

In order to reach many of the goals stated above, we decided to design VizION as a system of relatively independent elements, called nodes. Each node is basically a program that is running on a computer, communicating with other nodes running on arbitrary computer platforms. It is possible, and often desirable, to run multiple nodes on one computer, allowing every node to perform a small task, such as controlling a projector or a touchscreen while presenting the user with a friendly interface, or controlling other nodes. All nodes are connected via a normal Ethernet network in order to make the room act intelligently (Figure 1).

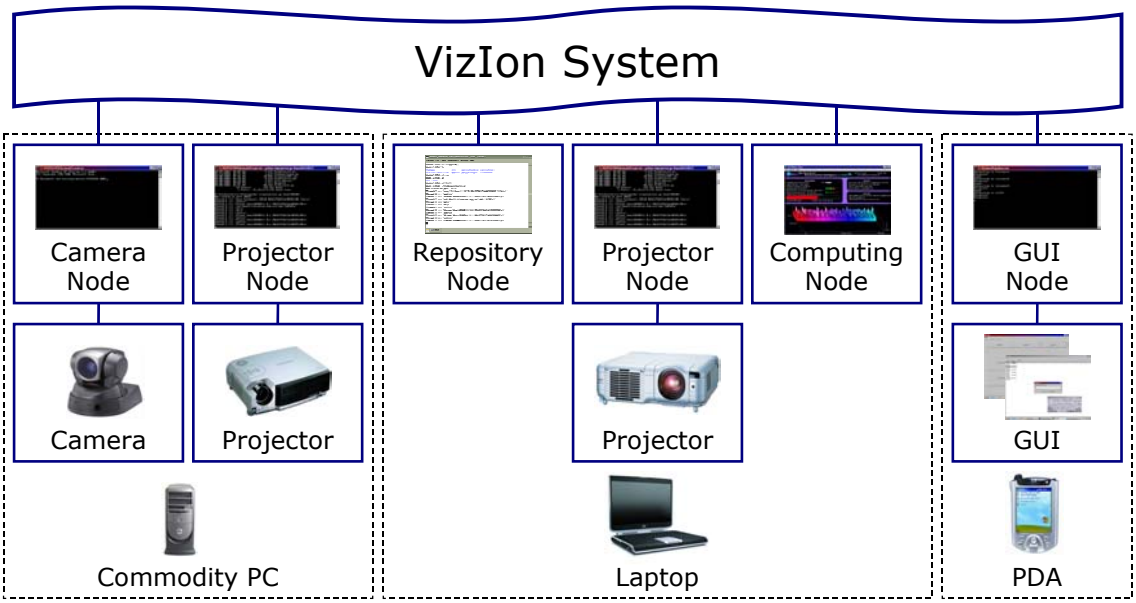

Figure 1 – The Structure of the VizION framework

Splitting the system into a large number of small, independent entities has a number of advantages. The most important one is that it makes the system very robust. If one node fails, only the specific service that this one node was providing will be affected while the remaining system stays fully operational. Another major advantages of the design is scalability. VizION will grow as VizClass grows and more services are integrated into it and VizION s design makes it easy to add new nodes to accommodate this growth. Since nodes can be added to, and removed from a running system, they also provide a means to deal with highly volatile environments. For example, when a new device enters the room, the nodes running on it announce their presence and tie the device into the system. When a device subsequently leaves, VizION will detect this event and safely remove the corresponding nodes.

3.2 Node Structure

While different nodes can provide very different services, they all share a common structure that is necessary to facilitate the communication between them. Each node listens for incoming transmissions from other nodes on at least one so-called network port. Every transmission consists of a request and a response message pair. Almost all of the coordination between the nodes can be accomplished using this structure. In our case, a number of default requests that every node has to respond to is defined and guarantees that this node is a valid member of the VizION system.

3.2.1 Node Attributes

A fundamentally important request mechanism is that for a node s attributes. Node attributes provide a means for distributing static information and are based on a list of keys and corresponding values that describe it. Attributes are read-only, meaning their number and values do not change while the node is running. Even though nodes may have a different number of attributes, a common subset of four attributes is defined for every node (Figure 2).

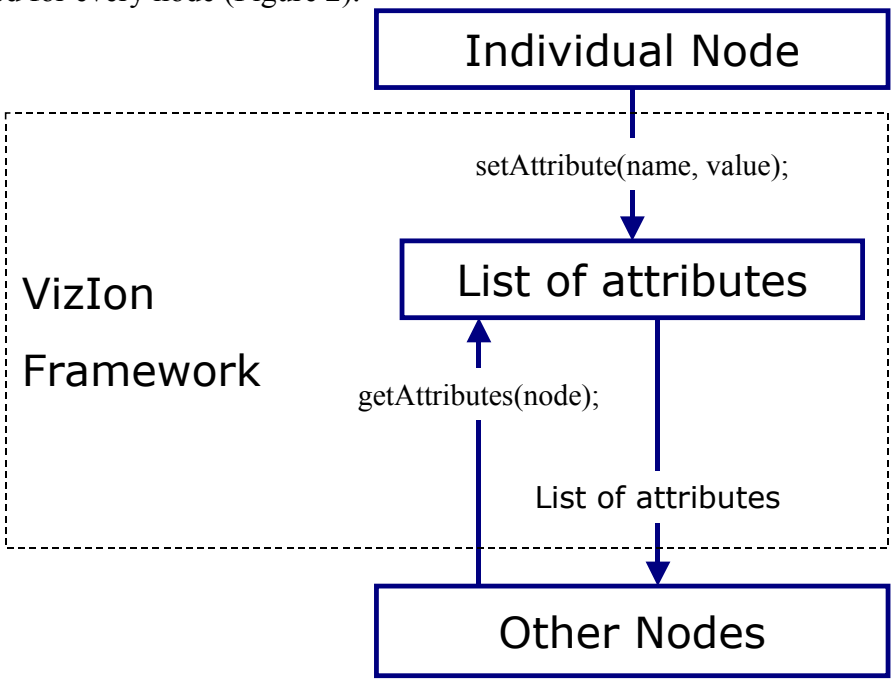

Figure 2 – A schematic of the attribute system implemented in VizION

The first common attribute is the NodeName which provides programmers and users of the system with an easy mechanism to identify a node. The second standard attribute is called NodeType and provides high-level information about the capabilities of the node. For example, a node driving a screen would have its type set to Screen . By convention, nodes of the same type have exactly the same interface beyond the commonalities that all VizION nodes share. They have the same attributes, the same messages and the same streams. Another attribute, called NodeVersion is used to store the version of the node. As nodes are being developed, the version number will increase and provide a means to uniquely identify different implementations of the same node over time. The other common attribute is the NodeId, which stores the unique ID of every node. This ID is guaranteed to be unique not only in the entire VizION system, but also in time. When a node is restarted, it will get a different ID and by comparing this attribute, specific node failure and restart events can be detected.

3.2.2 Messages

Message are a primary communication channel between nodes in VizION and nodes can exchange messages via a special type of request mechanisms. A message will contain information about a particular action and parameters that detail the reason why the message was sent. Receiving nodes can subsequently react to that message in the appropriate form. Every node has a fixed list of messages that it sends out and different nodes have different sets of messages while nodes of the same type are guaranteed to support the same message set. For example, a node may request a list of messages from another, and then subscribe to the messages on that list. Subsequently, whenever a node generates a message, it is automatically sent out to every node that subscribed to it. This subscriptions can be made dependent on the parameters of the message and if the parameters do not match a defined pattern that the subscriber provided, the message is not being sent. This mechanism facilitates the reduction of unnecessary network traffic within the VizION middleware. A node receiving a message can either accept or deny it. If the message is being denied, or if the delivery fails for any other reason, the recipient is automatically unsubscribed and will not receive further messages of that kind. VizION relies on this mechanism to keep its system state clean. Most of all,

it allows the system to automatically drop subscriptions of nodes that no longer exist. Figure 3 shows the message handling exposed to the developer of a node.

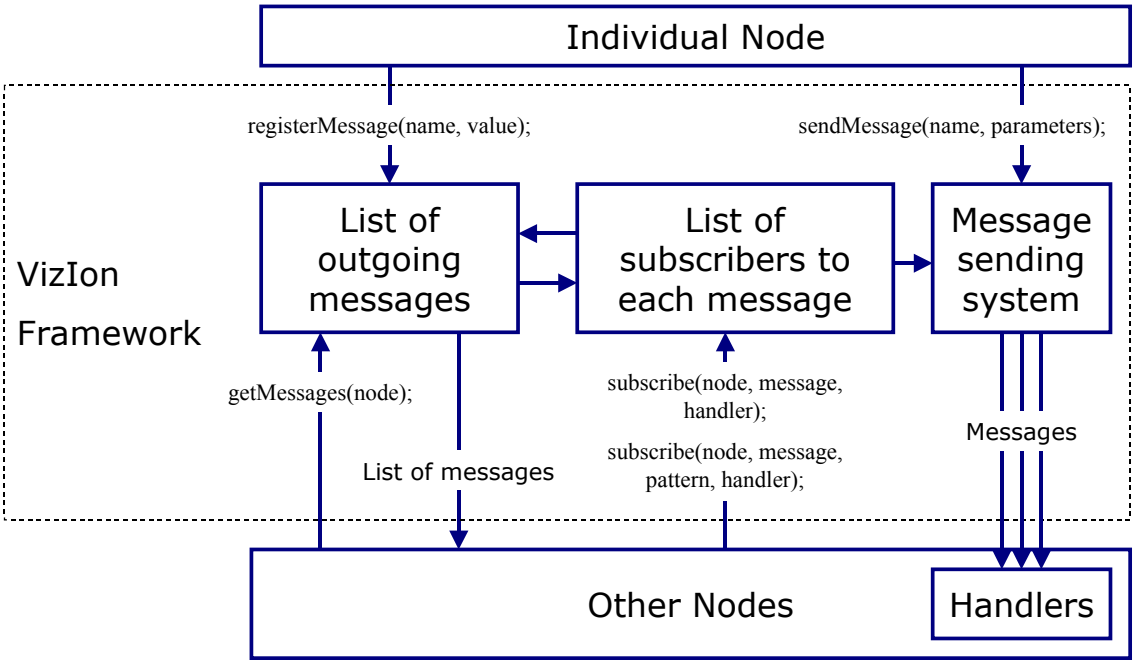

Figure 3 – A schematic of the message system implemented in VizION

3.2.3 Streams

Streams are the primary way to extend the functionality of VizION nodes. A stream listens on a network port for other incoming transmissions. Unlike the main VizION communication channel though, the format of these transmissions is completely free. Programmers of nodes are encouraged to design streams in the spirit of VizION, and tools are provided to make this as easy as possible. This provides developers with a great deal of flexibility, an important aspect of any ubiquitous computing middleware. One node may request a list of available streams from another node and the response will provide a list of names of the streams, accompanied by the information needed to contact that stream. This approach is illustrated in Figure 4. While a node is running, the number and stream types are guaranteed to remain the same until the node restarts, and that nodes of the same type have the same streams.

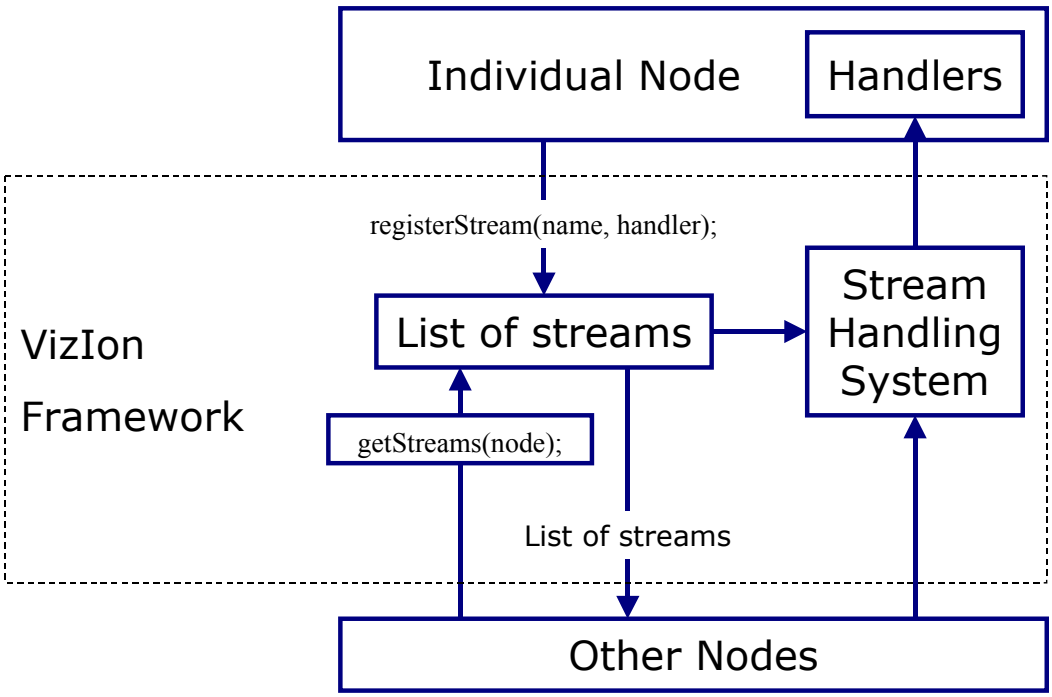

Figure 4 – A schematic of the stream system implemented in VizION

3.3 Node Repository

The repository is a special node in VizION tasked with keeping an updated list of all the nodes in the system and making this list available through a stream. Every new node joining the system, is required to announce its presence to the repository.

While nodes that fail ideally would inform the repository beforehand, this is not always possible. In order to accommodate unexpectedly failing nodes, the repository periodically verifies if all of the registered nodes are still operational, and otherwise removes them from the list. Whenever a change in the repository is detected, this information is broadcast of all active nodes, which in turn can perform required adjustments.

The concept of the repository goes against the paradigm of keeping the system fully distributed, introducing some of the disadvantages of a centralized implementation. While solutions exist that would allow us to replace the repository with a distributed mechanism, it turns out that the added complexity would bring very little value to a real life system. In our case, it is reasonable to assume that at least one computer in the room stays operational at all times and that this system would run the repository and that this repository saves its state to the hard drive whenever a change is detected. In the case of system failure, VizION can restore its previous status once the computer returns to operation. Even though this would mean that it is impossible to add new nodes to the system while the repository is offline, all other VizION functionality stays intact, and once the repository has been restarted the system works as before. If the restart delay is small, all changes will be completely transparent to the user.

4 THE LOGIN SYSTEM

To illustrate the flexibility and modularity of VizION, one particular set of nodes is presented in detail. The nodes of the types Login and LoginGUI work together to provide the user with a consistent login interface to the room. These modules address the challenge of controlling systems that consists of multiple computational nodes (in our case eight), running traditional operating systems such as WindowsXP. Since the instructor may use multiple PCs at the same time, it has to be possible to centrally log into all of the systems at the same time. To facilitate this, we have developed login nodes that run on each machine registered with the VizION system. These nodes wait for login information consisting of the username and password. Once this information is received via a stream, the nodes automatically log the user into the participating computers and broadcast a message to inform the rest of the VizION system.

The other component of the login system is represented by the LoginGUI node. This node waits for login credentials from the user, either in form of a username and a password, or, more conveniently in form of a USB stick that has the credentials saved on it. Once it receives these credentials it requests a list of login nodes from the repository and sends the login information to all of them, logging the lecturer into all the systems in the room. Thus, the USB stick essentially becomes a key to VizClass, unlocking the capabilities of our digital classroom.

5 CONCLUSIONS

We present VizION, a software framework that addresses and simplifies some of the common problems inherent to the development of IT classroom environments. We describe VizION in the context of a testbed project termed VizClass and formulate the corresponding middleware using independent nodes that communicate with each other through a standardized system of messages. Utilizing these distributed nodes, VizION achieves the stated goals of failure tolerance and scalability while making it easy to build new applications on the foundation that it provides.

In the future, we plan to integrate a broader range of wearable devices via the VizION framework. From a usability perspective, the next step will include an improvement of the login system. The password entry using the somewhat cumbersome on-screen keyboard will be replaced by authentication support based on a public key stored on either a wireless device or a USB stick. Another step will be to transparently integrate student laptops and PDAs. Students should be able to download arbitrary course material for later review, run simulations on their own machines to evaluate solution strategies for different sets of data, and then present them to the entire class or directly interact with lecture material. As more courses are taught in VizClass, additional feedback will become available, allowing us to refine system capabilities.

ACKNOWLEDGMENTS

This research is supported by the National Science Foundation, under Grant Number EIA-0203528. Support for the hardware infrastructure was provided by the California Institute for Information and Telecommunications (Cal-IT2) and the Henry Samueli School of Engineering at UC Irvine. In addition, the first author was partially supported by the Summer Undergraduate Research Program (SURP) at UC Irvine. The above support is greatly appreciated.

REFERENCES

[1] JOHANSON, B., FOX, A., and WINOGRAD, T. *The interactive workspaces project: Experiences with ubiquitous computing rooms*, IEEE Pervasive Computing Special Issue on Overviews of Real-World Ubiquitous Computing Environments, April 2002.

[2] ABOWD, G. *Classroom 2000: An experiment with the intrumentation of a living educational environment*, IBM Systems Journal, vol. 38, no. 4, pp. 508 530, October 1999.

[3] RASKAR, R., WELCH, G., CUTTS, M., LAKE, A., STESIN, L., FUCHS, H. *The office of the future: A unified approach to image-based modeling and spatially immersive displays*, in Computer Graphics Proceedings, Orlando, Fl, July 1998, ACM, Annual Conference Series.

[4] PONNEKANTI, S., JOHANSON, B., KICIMAN, E., FOX, A. *Portability, extensibility and robustness in iros*, in Proc. IEEE International Conference on Pervasive Computing and Communications, Dallas-Fort Worth, TX, March 2003, Percom.

[5] ROMAN, M., HESS, CH., CERQUEIRA, R., RANGANATHAN, A., CAMPBELL, R.H., NAHRSTEDT, K. *A middleware infrastructure for active spaces*, Pervasive Computing, IEEE, vol. 1, no. 4, pp. 74 83, October 2002.

[6] KUMAR, M., SHIRAZI, B., DAS, S., SUNG, B., LEVINE, D. *Pico: a middleware framework for pervasive computing*, Pervasive Computing, IEEE, vol. 2, no. 3, pp. 72 79, September 2003.

[7] JOHANSON, B., FOX, A. *The event heap: A coordination infrastructure for interactive workspaces*, in Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 02), Callicoon, NY, June 2002.

[8] HUTCHINSON, T., KUESTER, F., KIN, S.-J., LU, R. *Developing an advanced it-based visualization classroom for enhanced engineering learning*, in International Conference on Engineering Education, Valencia, Spain, July 2003, pp. 1 9.