The Smart Classroom: Merging Technologies for Seamless Tele-Education

The Smart Classroom integrates voice-recognition, computer-vision, and other technologies to provide a tele-education experience similar to a real classroom experience.

Yuanchun Shi, Weikai Xie, Guangyou Xu, Runting Shi, Enyi Chen, Yanhua Mao, and Fang Liu
Tsinghua University

Tele-education systems promise wider access to education and support for lifelong learning. These systems are either asynchronized or synchronized. Asynchronized systems are relatively simple. An organization can use the Internet to publish hyperlinked multimedia content and reach a wide audience. Yet, most current courseware is simply textbook material transferred to HTML; instead of reading the book, students read the screen. In most cases, live instruction catches students' attention and interest much more effectively than static materials. Real-time interactive virtual classrooms therefore play an indispensable role in distance learning. In this type of tele-education, multimedia communication systems let teachers and students in different locations participate in the class synchronously. Most systems are desktop-based, however, so the teacher must remain at the computer, using the keyboard and mouse to operate the class—an awkward experience for a teacher.

By applying smart space technologies in a real classroom, the Smart Classroom project bridges the gap between tele-education and traditional classroom activities in terms of the teacher's experience and seamlessly integrates these two currently separate educational practices. (See the sidebar for a discussion of other projects seeking to merge pervasive computing technologies and education.) More specifically, we extend the user interface of a legacy desktop-based tele-education system—SameView[1,2]—into the 3D space of an augmented classroom. In the Smart Classroom, teachers can use multiple natural modalities while interacting with remote students to achieve the same effect as a teacher in a classroom with local students.

Figure 1 gives an overview of the Smart Classroom. The system turns a physical classroom into a natural user interface for tele-education software. Teachers in the Smart Classroom can move freely, using conventional teaching methods to instruct remote students. Because they are in a real classroom environment, they can accommodate local students at the same time. Simultaneously instructing local and remote students also requires a smaller workforce than separate on-campus and tele-education operations. Furthermore, the lecture, recorded as hypermedia courseware, is available for playback after class.

Classroom setup

Our prototype system is in a 3.2×7.6-meter room divided by a curtain. The class takes place in the main part of the room. The other section is filled with computers required by the system, purposefully hidden from view to exemplify the "invisible computer" notion.

The classroom has two wall-size projector screens, one on the front wall and the other on a side wall (see Figure 2). The *MediaBoard* is a touch-sensitive SmartBoard screen for displaying teaching materials. Remote students can view the board's content via a client program (assuming remote stu-

1536-1268/03/$17.00 © 2003 IEEE ■ *Published by the IEEE CS and IEEE ComSoc*

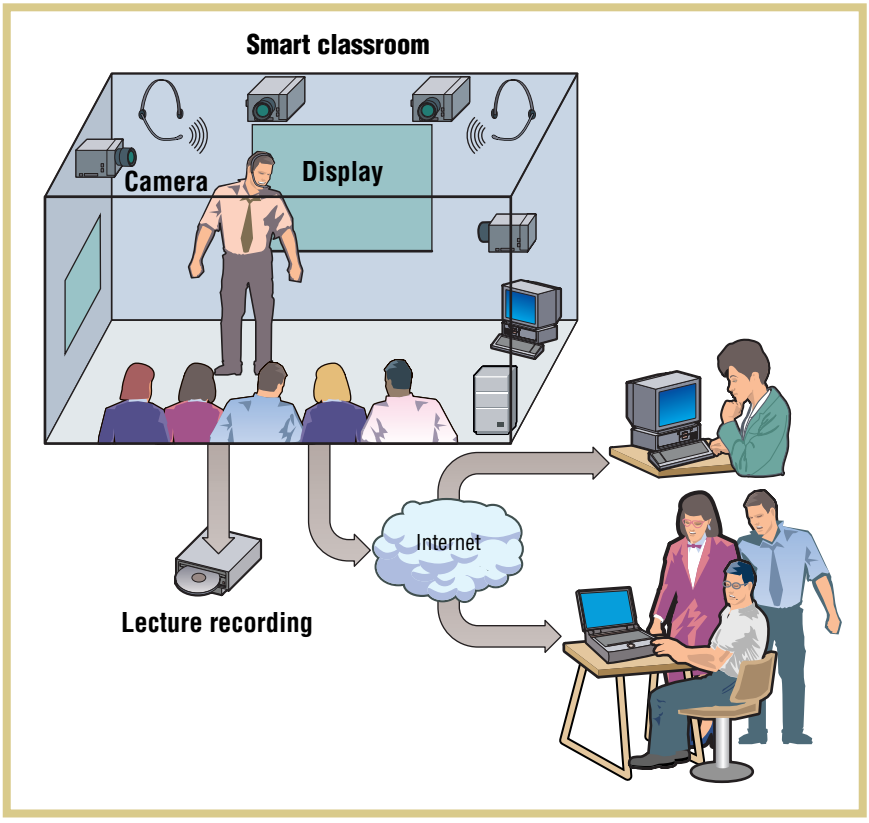


Figure 1. The Smart Classroom system. Integrated modules let teachers interact with remote students as though they were in the physical classroom.

dents access the classroom via their computers). The *StudentBoard* displays remote students' images and a computer-animated virtual assistant. When a remote student takes the floor, live video and audio of the student is projected here too. The teacher and local students view and interact with remote students through the StudentBoard.

The classroom is equipped with several cameras. Some capture the teacher's actions, which are recognized and translated into an interaction command to the underlying tele-education software. Other cameras capture 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.

Numerous software modules provide classroom functionality. For example, SameView provides the interaction channels between local and remote students, and various perception modules let the teacher interact with remote students as though they were local. The system's software infrastructure coordinates the distributed modules.

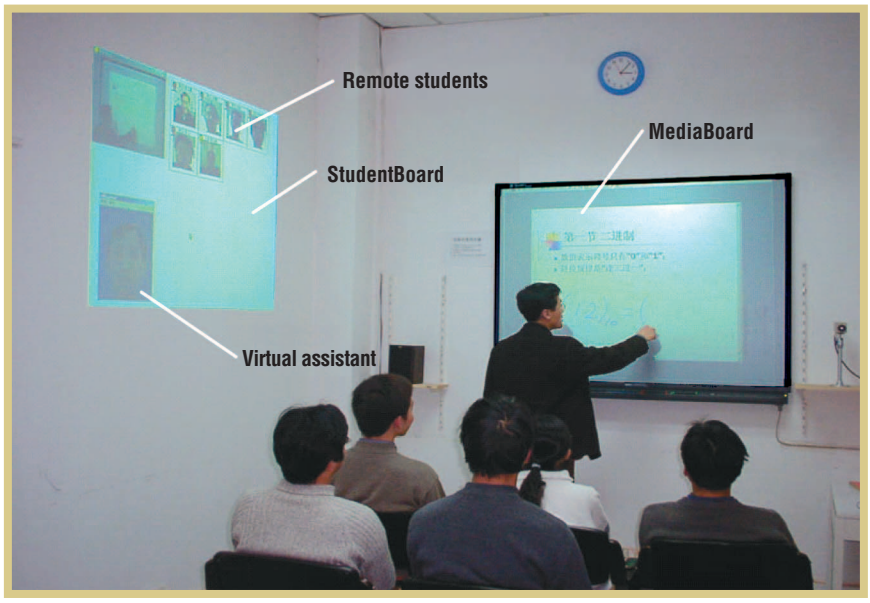


Figure 2. A snapshot of the prototype Smart Classroom. Teaching materials are displayed on the MediaBoard at the front of the class, while remote student images are displayed on the StudentBoard, located on a side wall.

Component technologies

The system involves a number of component technologies that make the interaction between the teacher and remote students as smooth as that in a physical classroom. With the exception of IBM China Research Lab's speech recognition, our lab developed all these component technologies. Due to space limitations, we only describe how the technologies function in the Smart Classroom project. We direct interested readers to the works cited for descriptions of the technologies' internal algorithms.

Remote student telepresence

When merging tele-education with real classroom instruction, the teacher and local students must be made aware of the remote students. In the Smart Classroom, when remote students log on using the provided client program, the client program uses cameras on the students' com-

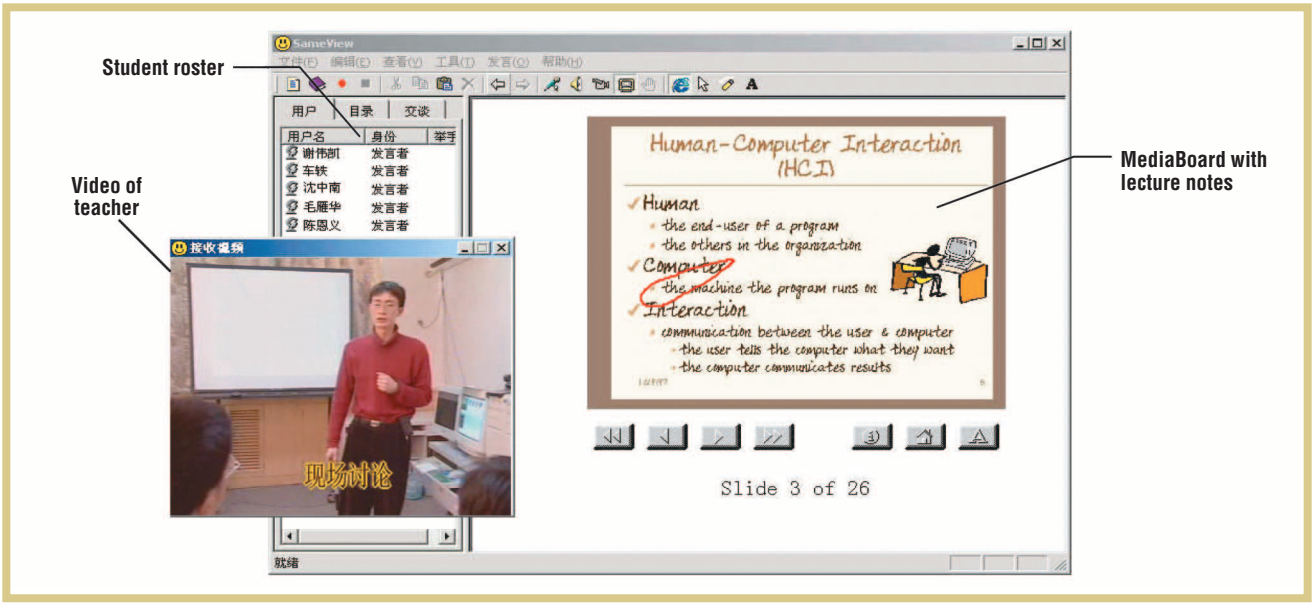


Figure 3. Remote students' client interface. Students can watch live video of the teacher, view lecture notes on the MediaBoard, or chat with the teacher or other students.

puters to capture their images and transfer them to the classroom's StudentBoard. Presenting live video of all the remote students would certainly add color to the class, but doing so would place too much of a burden on the network and CPU. Thus, the StudentBoard only displays the live video of remote students who are explicitly given the floor.

We allocate the same fixed amount of space for each remote student image and place them side-by-side. This approach, however, is not scalable when there are more remote students than fit in the StudentBoard's display space. Our current solution is to limit the number of students that can request the floor and display their images only. To better address the problem, we are seeking more sophisticated visualization methods. One plan is to render each image as if the remote students were seated in a large hall—the students' image size decreases as their virtual distance from the teacher increases.

Pen-based user interface

Teachers in real classrooms often write on chalkboards or whiteboards. Most current tele-education practices, however, confine teachers to their computers by requiring them to use the mouse and keyboard. We address this problem by adopting the wall-size touch-sensitive SmartBoard device as the display screen for MediaBoard. A SmartBoard accommodates most mouse functions. To move to a new page, for example, the teacher need only touch an icon on the display. Teachers can use digital pens and erasers to write or remove notes and to diagram on the SmartBoard directly (see Figure 2).

With the help of the underlying real-time multimedia communication software, remote students see the same lecture materials or notes on the SmartBoard as local students do. Figure 3 shows the client program for remote students. A remote student who has the floor can also write on this board freely—for instance, a student might write the solution to a question the teacher has just posed.

Laser pointers as interaction tools

In most current desktop-based tele-education systems, teachers use a mouse or keyboard to relay classroom arbitration decisions to the computer. In the Smart Classroom, a computer-vision-based module called Laser2Cursor lets teachers accomplish this task more intuitively: the teacher aims the laser pointer at a remote student's image (highlighted to indicate that it is the current selection), and by *clicking* on the selected image—fixing the laser spot on the point of interest for a second and then circling that point—the teacher gives the floor to a remote student or revokes it at will.

The module also works for the MediaBoard, where the laser pointer serves as a remote mouse. Using this module, a teacher need not approach the MediaBoard to run a slide presentation. Moreover, when the teacher points the laser pointer at the MediaBoard to highlight a point in the lecture, a red spot is visible at the same position on the remote students' view.

Laser2Cursor embodies a number of ideas not seen in previous work on laser pointer tracking.[3,4] First, we have developed a training process to improve the system's adaptability. By learning the background of the image captured by the cameras and parameters such as color segmentation and motion detection thresholds, our system automatically adapts to new environments. Second, to improve the system's robustness, we integrate multiple cues such as color, motion, and shape in the laser spot detection. Because most people's hands are unsteady, when a per-

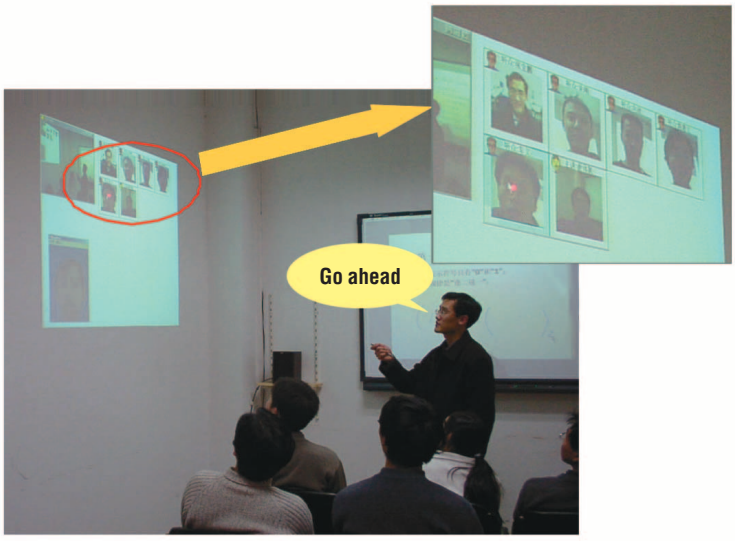


Figure 4. A teacher highlights a remote student's image using a laser pointer and issues the voice command, "Go ahead," to give the student the floor.

son aims a laser pointer, the spot's exact position usually jitters. We use this characteristic as an additional cue to detect the spot. Next, a Kalman filter smoothes the spot's trajectory, which tends to be irregular and rough. Finally, the module uses a finite state machine to track the spot's behavior—that is, whether it is moving or clicking.

To test the module's usability, we asked several students in our lab to select items on the MediaBoard using a laser pointer. The module yielded satisfactory performance, with most testers agreeing that spot detection is precise and that moving the cursor is almost as easy as it is with a mouse. Still, some students complained that clicking was relatively difficult. As a remedy, we offer users an optional interaction modality: the voice command.

Speech-capable virtual assistant

In the Smart Classroom, a speech-recognition module lets the teacher perform some common tasks using voice commands. For example, the teacher might say, "jump to the previous page," or "go ahead" to give the floor to a remote student whose image is highlighted with the laser pointer, as Figure 4 shows. To obtain reliable recognition, we limit the number of valid voice commands. Moreover, because the system's recognition engine is set to work in voice-command-oriented mode, it seldom mistakes the teacher's continuous speech for voice commands.

The Smart Classroom also uses a text-to-speech module[5] to notify the teacher when certain events occur. For example, when remote student Tom asks for the floor by clicking a menu item on the client program, the system advises the teacher, "Tom is asking for the floor."

Because teachers might feel strange interacting with a lifeless classroom, we adopted a virtual assistant to represent the Smart Classroom. On the StudentBoard, a computer-animated human image (see Figures 2 and 5) with facial expressions and lip movements synchronized with the text-to-speech module's synthesized voice represents the virtual assistant.[6] Thus, the system provides a natural metaphor and the teacher interacts with an assistant that understands voice commands and produces proper vocal notification and feedback.

Biometrics-based login

The Smart Classroom must identify a lecturer before authorizing use of the classroom facilities. By combining face-recognition and speaker-verification technologies,[7,8] the Smart Classroom can identify a teacher automatically and provide an undeterred login experience.

On entering the Smart Classroom, the teacher stands in front of a camera and says his or her name (or any other words

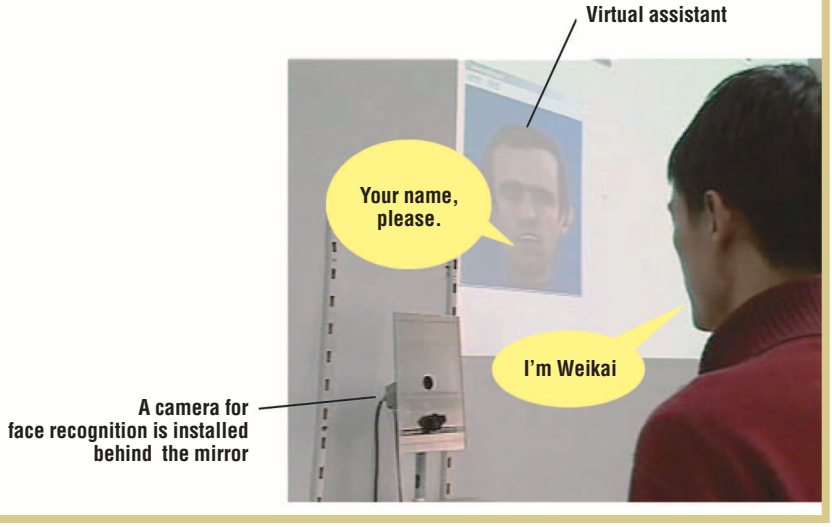


Figure 5. A teacher logs on to the Smart Classroom using a biometrics-based module combining face-recognition and speaker-verification technologies.

Figure 6. Different scenes delivered to remote students according to the class context: (a) teacher writing on MediaBoard; (b) teacher showing a model; (c) teacher having a discussion with local students.

the teacher has chosen). If both the face-recognition and speaker-verification modules respond positively, the virtual assistant replies with a greeting that places the Smart Classroom at the teacher's disposal, as Figure 5 illustrates. The system uses the teacher's identification information to load the proper voice model into the speech-recognition module, if the teacher has trained the model in advance. This improves voice-recognition accuracy during the lecture.

Smart Cameraman

In a conventional classroom, students naturally follow the class's focus as it changes from time to time. For instance, when a teacher writes a formula on the blackboard, the formula becomes the center of attention; when a teacher holds up a model, the model becomes the object of general interest. Yet in most current real-time teaching systems, remote students only see a fixed scene regardless of the changing context of the class, which hampers their understanding of the instruction. Although a human camera operator can select the proper view for live video, the high labor cost often makes this infeasible. More importantly, teachers and students generally agree that having such a person in class would make them uneasy.

The Smart Cameraman overcomes this problem. This component consists of an array of cameras and a decision module. By drawing clues from observed classroom activities, the Smart Cameraman automatically distinguishes among several activities during a typical class. Using this information, the decision module selects the most appropriate view. We define several context types for this module:

- *Teacher writing on the MediaBoard*. Whenever the teacher is writing on the MediaBoard, the module selects a close-up view of the board, as Figure 6a shows.
- *Teacher showing a model*. When the teacher holds up a model, the camera zooms in on it, as Figure 6b shows.
- *Remote student speaking*. When a remote student is speaking, live video of the student will be delivered to other remote students. Meanwhile, it will be displayed on the StudentBoard.
- *Other*. In all other situations, the module selects the classroom overview, as Figure 6c shows.

By tracing the SmartBoard's mouse events, the Smart Cameraman knows when the teacher is writing on the MediaBoard; by monitoring floor control events, it can tell when a remote student is speaking. Meanwhile, a separate computer-vision-based hand-gesture-recognition module decides whether the teacher is holding up a model.[9] For reliable recognition, we currently restrict the teacher to a predefined area when displaying models. We model the class context using a finite-state machine whose state transitions are triggered by recognized cues. To filter possible transient misrecognition, each state has an associated minimum steady time.

In systems such as the Smart Classroom, in which various modules must be connected and coordinated, the software infrastructure plays a key role.

Software infrastructure

In systems such as the Smart Classroom, in which various modules must be connected and coordinated, the software infrastructure plays a key role. The infrastructure provides the runtime environment and common services for the modules as well as a development interface for building modules that can be seamlessly integrated into the system. The Smart Platform[10] (see our Web site, http://media.cs.tsinghua.edu.cn/

Pervasive Computing Technologies in Education

Many research projects have sought to augment traditional education with pervasive computing technologies. They cover different aspects of educational activities.

As many students might agree, it is difficult to keep up with both their notes and the lecture. DePauw University's Debbie project (http://acad.depauw.edu/~dberque/debbie/index.htm) tackles this problem by providing teachers and students with pen-based tablet PCs and associated software. As the teacher displays digitalized lecture materials or writes notes on his or her tablet, the system distributes a synchronized view to the students' devices, and students can save it with their notes. Laurent Denoue and Patrick Chiu at Fuji-Xerox Palo Alto Laboratory (FXPAL)[1] assume students have handheld devices with relatively inconvenient means of inputting text. In their system, a word or phrase entered by one person can be quickly reused by others. With the concerted efforts of all group members, note taking becomes more efficient.

Other researchers attempt to capture the classroom experience, where not only the note and slides, but also the live video and audio and other class threads can be recorded in a synchronized and structured manner. A student can use the recorded documents to review the lecture after class, or to make up for a missed class. GaTech's Classroom 2000,[2] McGill University's Intelligent Classroom,[3] Cornell University's Lecture Browser (www.cs.cornell.edu/zeno/projects/lecture browser/Default.html), and the conference room at FXPAL[4] are different implementations of the same facility. In these projects, the captured classes are stored in a Web-accessible format so they readily become courseware for tele-education. Rather than have students passively view recorded documents, David Bargeron and his colleagues at Microsoft Research propose adding communication channels such as email so a student can interact with the teacher when viewing the recorded class.[5] As they note, however, such interaction is asynchronous.

Other projects seek to automate routine classroom tasks. In McGill's Intelligent Classroom, the computer system does the housekeeping chores, so lecturers need not worry about things such as light or drape control or A/V device configuration. Northwestern University's Intelligent Classroom project focuses on automated lecture video production, in which the camera uses plan-recognition technologies to track the lecture's focus.[6] AutoAuditorium (www.autoauditorium.com), a commercially available product, shares a similar objective but is more suitable for business presentations.

Although our system has many similarities to these works, it distinguishes itself by allowing the teacher to simultaneously interact with local and remote students in a similar fashion. Work at the University of Toronto's Knowledge Media Design Institute (http://epresence.kmdi.utoronto.ca) is similar to ours in that remote students can participate in a real classroom via the Internet. Our system provides a rich variety of bidirectional interaction capabilities, however, whereas in their work the interaction channel from remote students to local teachers or students is limited to plain text chatting. Furthermore, many tele-education projects have used computer-supported cooperative work or videoconference products such as Microsoft's NetMeeting, which involve desktop-based teaching—exactly what we try to overcome.

REFERENCES

1. L. Denoue, P. Chiu, and T. Fuse, "Shared Text Input for Note Taking on Handheld Devices," *Proc. ACM Human Factors in Computing Systems* (CHI 2002), ACM Press, 2002, pp. 794–795.
2. G.D. Abowd, "Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment," *IBM Systems J.*, special issue on pervasive computing, vol. 38, no. 4, 1999, pp. 508–530.
3. J.R. Cooperstock, "Classroom of the Future: Enhancing Education through Augmented Reality," *Proc. Conf. Human-Computer Interaction* (HCI Int'l 2001), Lawrence Erlbaum Assoc., 2001, pp. 688–692.
4. P. Chiu et al., "Room with a Rear View: Meeting Capture in a Multimedia Conference Room," *IEEE Multimedia*, vol. 7, no. 4, Oct.–Dec. 2000, pp. 48–54.
5. D.M. Bargeron et al., *Asynchronous Collaboration Around Multimedia and Its Application to On-Demand Training*, Microsoft Research tech. report 99-66, 1999.
6. D. Franklin, "Cooperating with People: The Intelligent Classroom," *Proc. 15th Nat'l Conf. Artificial Intelligence* (AAAI 98), AAAI Press, 1998, pp. 555–560.

~pervasive, for runtime code and the software developer's kit) is not limited to the Smart Classroom but addresses common areas of concern for any smart space project, including these issues:

- The components in a smart space might not have been designed to cooperate with each other. To assemble them with least effort, we need a high-level abstraction model and coordination mechanism.
- Many perceptual interface modules are prone to runtime errors or exceptions, making it necessary to occasionally restart them, especially during joint debugging. A loosely coupled system is therefore essential not only to ensure runtime error resilience but also to facilitate system development.
- Many modules must exchange real-time data at a constant rate—that is, they exhibit characteristics of stream-oriented communications. For instance, the Laser2Cursor module sends the

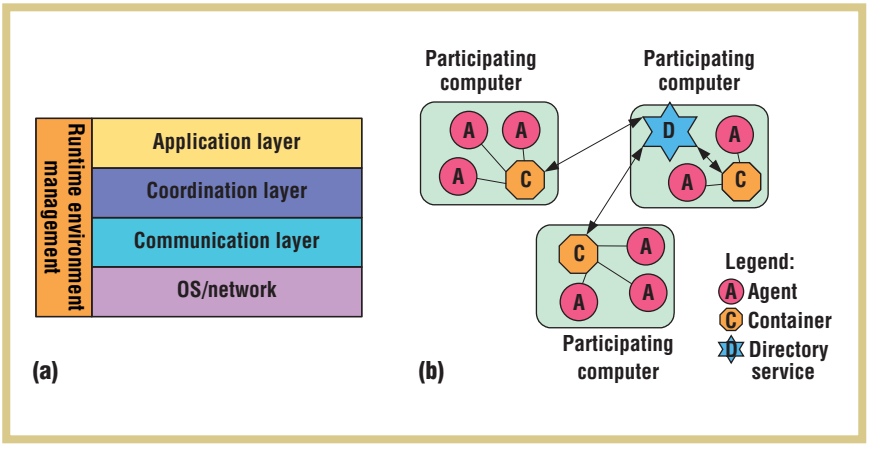


Figure 7. The Smart Platform. (a) The platform architecture consists of four layers and a vertical runtime environment management mechanism to manage agents. (b) The two-tier runtime structure protects against transient network failures and requires no configuration.

updated cursor position to the MediaBoard module 10 times per second. In such cases, communication latency is of serious concern.

- Developers of smart space component technologies are not necessarily distributed computing experts and thus prefer a simple system structure and development interface to a sophisticated one.

Architecture and runtime structure

In the Smart Platform, each module is modeled as an agent that has its own goal and cooperates with others through an interagent language. An agent encapsulates not only the module's state and behavior but also the logic for activating its behavior, and interagent communication occurs on a relatively high level, usually the intention level. These features are suitable for constructing a system involving diversified component technologies, such as the Smart Classroom.

Figure 7a shows the Smart Platform architecture. Between the bottom operating system/network layer and the upper application layer are the communication and coordination layers. The former enhances the underlying network layer's quality of service; the latter lets agents collaborate in a structured and coherent manner to achieve the system's overall goal. A vertical runtime environment management mechanism manages tasks such as agent registry, agent lifecycle, and so on.

The Smart Platform's runtime structure, shown in Figure 7b, has two tiers. Each computer in the system hosts a *container* component, which provides a local runtime environment for agents on the same host. Globally, a *directory service* component is responsible for agent registration, message dispatch, and system management.

On startup, agents connect to their local container on a predefined local socket, while the container and the directory service establish connection with a self-discovery mechanism based on IP multicast. More specifically, the directory service always listens on a multicast address. The container sends a "hello" message to this address. On receiving the message, the directory service responds with a "hello-ack" message to the container, specifying its IP address and port where the container can establish a connection to it.

The two-tier structure has two significant advantages: resilience to transient network failures and zero system configuration. When the network connection recovers from a transient failure, only the container must renegotiate the connection with the directory service; the connections between local agents and containers will remain unaffected. The automatic directory service–container negotiation mechanism also frees developers and users from having to configure system components to tell them the locations of other components.

Interagent communication

The Smart Platform coordinates agent communication via a message-group-based publish-and-subscribe model. Each agent can publish messages or subscribe to any message group. If an agent designates an unknown message group, the system creates the new group immediately. Therefore, the agents need not start up in a rigid relative order. When an agent publishes a message to a group, the system forwards it to all subscribers by invoking a callback function on the receiving side. This coordination model enables a loosely coupled system with these features:

- An agent need not have prior knowledge of other agents' names or locations. As long as it is working with the correct message group, the directory service will handle all message routing.
- Agents communicate in an asynchronous and nonblocking way, restricting the spread of failures and facilitating system recovery after an agent fails.

Two kinds of message groups are available:

- Message-oriented, for communications without tight temporal constraints

In the Smart Platform, each module is modeled as an agent that has its own goal and cooperates with others through an interagent language.

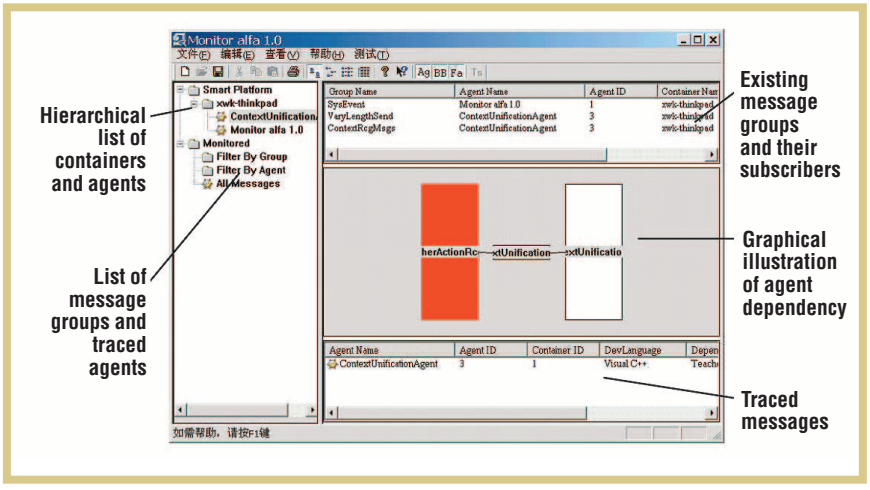


Figure 8. Monitor agent user interface. The system-level agent lets users remotely launch or kill an agent, trace agent messages, or send messages manually.

- Stream-oriented, to transmit data that are sensitive to transmission delay and delay variation

Messages in the message-oriented group are transferred along the agent-container-directory service-container-agent path, with the directory service acting as message dispatcher and replicator. For the stream-oriented messaging service, the directory service assigns an IP multicast address to each streaming group, and the system transfers the message to this address directly, using the real-time transport protocol (RTP). Such peer-to-peer communication effectively reduces transmission delay. RTP and the associated buffering scheme on the receiver's side lessen the variation in message delivery latency. Furthermore, IP multicast ensures that subscribers in the same group receive messages simultaneously.

The Smart Platform message format conforms to the XML standard both on the agent level and the wire protocol level. XML's extensibility means developers can add more descriptive fields to their messages without disturbing agents that only understand old versions of messages. The XML-based wire protocol will help our system interoperate with others in the future.

Runtime environment management and debugging support

Users or developers of a distributed system often find it cumbersome to have to walk from computer to computer to start agents or check their running states. To avoid this, the Smart Platform uses an agent dependency-management mechanism and a centralized system-monitoring and debugging mechanism.

When registering with the directory service, each agent should specify two arguments: **OfferedSvcs**, a list of services the agent offers; and **ReliedSvcs**, the services it relies on for normal operation. Through these arguments the directory service learns the inter-agent dependencies. Furthermore, by recording agents' location history, the directory service can automatically negotiate with the correct container to launch an agent whose service is demanded by a running agent. Thus, a group of agents can be loaded in one action.

The containers and directory service have built-in system state collection and control capabilities, which can be accessed through a system-level agent called the *monitor*. Figure 8 shows the monitor's user interface, which includes the host name and IP address of the computers participating in the runtime environment, the running agents and their status, and the interagent dependency topology. System developers can use monitor agents to remotely launch or kill an agent, trace an agent's messages, or manually send a message to a group or an individual agent.

Agent development interface

The Smart Platform agent development interface is fairly simple and straightforward. In most cases, agents only need five primitives: **Register**, **Subscribe**, **OnMsgNotify**, **Publish**, and **Quit**. The interface is provided as an abstract class, available in both C++ and Java. We also implemented a custom AppWizard for Visual C++, with which developers can generate an agent's skeleton code in a single step. In addition, we provide a standard setup program that automatically installs and configures required components for new computers, making deployment much easier.

We performed an informal usability study of the Smart Platform by training Smart Classroom project members to use it. The seven participants differed in research backgrounds and distributed computing expertise. Our observations showed that most of them grasped the Smart Platform principles and could use its SDK in less than an hour. Half of the trainees deployed the Smart Platform on their own computers and began developing agents without further help.

In informal user evaluations with several teachers from different departments at our university, our prototype system received positive comments, especially for its rich and intuitive interactive channels for working with remote students. In collaboration with Tsinghua University's Distance Learning School, we plan large-scale testing of the system in Fall 2003.

As portable computing devices proliferate, teachers will more likely take portable computers or PDAs into the Smart Classroom. Thus, our future work will seek to configure the interfaces to support the available devices and seamlessly add mobile devices to the Smart Classroom. P

ACKNOWLEDGMENTS

The research presented here is supported by the National Natural Science Foundation of China, 863 High-Tech Plan, the Ministry of Education, China, and IBM China Research Lab.

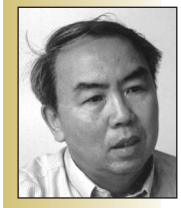
REFERENCES

1. C.Y. Liao, Y.C. Shi, and G.Y. Xu, "AMTM—an Adaptive Multimedia Transport Model," *Proc. SPIE Int'l Internet Multimedia Management Systems*, Int'l Soc. for Optical Eng. (SPIE), 2000, pp. 141–149.
2. Y.Z. Pei et al., "Totally Ordered Reliable Multicast for Whiteboard Application," *Proc. 4th Int'l Workshop on CSCW in Design*, Univ. de Technologie de Compiegne, Compiegne, France, 1999, pp. 253–256.
3. C. Kirstein and H. Muller, "Interaction with a Projection Screen Using a Camera-Tracked Laser Pointer," *Proc. Multimedia Modeling* (MMM 98), IEEE CS Press, 1998, pp. 191–192.
4. D.R. Olsen and T. Nielsen, "Laser Pointer Interaction," *Proc. ACM Human Factors in Computing Systems* (CHI 2001), ACM Press, 2001, pp. 17–22.
5. J.H. Tao and L.H. Cai, "A Neural Network Based Prosodic Model of Mandarin TTS System-2," *Proc. Int'l Conf. Spoken Language Processing* (ICSLP 2000), China Military Friendship Publish, vol. II, 2000, pp. 75–78.
6. H. Zhang et al., "Robust Pose Estimation for 3D Face Modeling from Stereo Sequences," *Proc. IEEE Conf. Image Processing* (ICIP 2002), vol. III, IEEE Signal Processing Soc., 2002, pp. 333–336.
7. F. Xie, G.Y. Xu, and E. Hundt, "A Face Verification Algorithm Integrating Geometrical and Template Features," *Proc. 2nd IEEE Pacific-Rim Conf. Multimedia* (PCM2001), Springer, 2001, pp. 253–260.
8. Z.Y. He and Q.X. Hu, "A Speaker Identification System with Verification Method Based on Speaker Relative Threshold and HMM," *Proc. 6th Int'l Conf. Signal* (ICSP 2002), People's Posts and Telecomm. Publishing House, 2002, pp. 488–491.
9. H.B. Ren and G.Y. Xu, "Human Action Recognition in Smart Classroom," *Proc. 5th IEEE Int'l Conf. Automatic Face and Gesture Recognition*, IEEE CS Press, 2002, pp. 399–404.
10. W.K. Xie et al., "Smart Platform: A Software Infrastructure for Smart Space (SISS)," *Proc. 4th Int'l Conf. Multimodal Interfaces* (ICMI 2002), IEEE CS Press, 2002, pp. 429–434.

the AUTHORS

Yuanchun Shi is a professor in the State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science, Tsinghua University, China. Her research interests include pervasive computing, human-computer communication technology, and distributed multimedia processing. She received her PhD in computer science from Tsinghua University. She is a member of the IEEE and the vice chairperson of the Young Scientist Committee of the China Computer Federation. Contact her at shiyc@tsinghua.edu.cn.

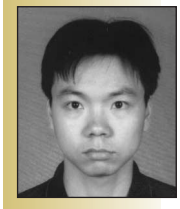
Weikai Xie is a PhD candidate in the Department of Computer Science at Tsinghua University, China. His research interests include software infrastructure for smart spaces, formal context data representation, and context-reasoning models in context-aware computing. He received a BS in computer science from Tsinghua University. Contact him at xwk@media.cs.tsinghua.edu.cn.



Guangyou Xu is the chair professor of the Department of Computer Science, Tsinghua University. His research interests are in computer vision, human–computer interaction, and multimedia computing. He graduated from the Department of Automatic Control Engineering, Tsinghua University, China. He is a senior member of the IEEE and a standing member of the Council of China Image and Graphic Association. Contact him at xgy-dcs@tsinghua.edu.cn.

Runting Shi is a senior undergraduate in the Department of Computer Science's Pervasive Computing Group at Tsinghua University, and a part-time student in the Internet Media Group, IBM China Research Lab. Her research interests include software infrastructures for smart spaces, media streaming, peer-to-peer computing, and scalable multicast infrastructures. Her homepage is available at http://media.cs.tsinghua.edu.cn/~shirunting.

Enyi Chen is a PhD candidate in the Department of Computer Science, Tsinghua University. His major research interests include smart spaces, mobile and wireless networks, human–computer interaction, and multimedia systems. He received his MS in computer engineering from Northern Jiaotong University, China. Contact him at chenenyi00@mails.tsinghua.edu.cn.



Yanhua Mao is an MS candidate in the Department of Computer Science, Tsinghua University. He received a BS in computer science from Tsinghua University. He is now working on software infrastructures for smart spaces. Contact him at maoyanhua@tsinghua.org.cn.

Fang Liu is an MS candidate in the Department of Computer Science, Tsinghua University. His research interests include computer vision, graphics, and human–computer interaction. He received a BS in computer science from Huazhong University of Science and Technology. Contact him at liuf00@mails.tsinghua.edu.cn.