Collaborative Learning with the Distributed Building Site Metaphor

Fabien Costantini and Christian Toinard
Center for Study and Research in Information Technology, National Conservatory of Industrial Arts and Crafts (CNAM), France

We describe a new approach to interactive and collaborative learning in an industrial training environment. Users learn in the 3D world where multiple instructors can observe, advise, and correct in real time. Our distributed building site metaphor provides distribution services for sharing a virtual world and enables different collaboration styles.

Distance-learning environments generally focus either on broadcasting knowledge, a virtual classroom, or educational games. When the solutions use a virtual environment (VE), it's to immerse the students in a virtual classroom where they use videoconferencing to interact with teachers and other students. Sometimes, a student can play with the VE to get feedback from an educational game.

Many of the current collaborative learning solutions however have prohibitive limitations. For example, they don't have adapted distributed VEs. They don't address the requirements of concurrent learning because they don't preserve the work's progress (that is, learners must redo their work because of concurrent operations), and they limit parallel learning. Moreover, they don't let instructors easily switch between different learning styles. Client–server approaches are also inefficient in many ways. They introduce a bottleneck and a failure point in the system. Requiring a specific quality of service (QoS) from the underlying network limits a solution's ease of deployment and openness. (See the sidebar "State of the Art in Distance Learning" for more information.)

Our Distributed Building Site Metaphor (DBSM) system provides several improvements. The basic idea behind our proposal is that a learner uses the VE to interact with the distant participants. Thus, we augmented that interaction so that the virtual scene mediates human interactions. For example, an instructor can introduce a virtual scene in a meeting that learners can interactively modify. The virtual scene becomes the learning subject, and the learners use the scene to respond and interact with each other. Typically, the collaboration enables modifications, annotations, and sharing.

Thus, we've improved collaborative learning with real-time interactions in a shared virtual scene. The participants collaborate in real time during a meeting, but they can take work home to continue learning offline. In later meetings, learners introduce their homework into the shared scene and can merge and compare different solutions. During a meeting, the participants in the distributed environment find the best answer to a given problem together.

We can easily deploy our solution on any Internet protocol (IP) network infrastructure. It doesn't require a specific quality from the underlying network such as resource reservation. Our solution also doesn't transmit the scene events with a classical reliable multicast, which doesn't scale well due to the positive acknowledgment problem. In addition, it doesn't implement an ordered multicast[1] (that is, a causal and total order) that introduces a high overhead. Moreover, a reliable or ordered properties of the multicast protocols don't preserve the work's progress. In our solution, parallel work isn't limited and we achieve consistency by using a lightweight protocol that's processed uniquely at the appropriate time. A peer, a system agent providing the end-user collaboration services, can check if its copy is up to date by multicasting a request to get the version of distant objects. With version numbers, the peer can request retransmission of out-of-date objects.

Finally, we can deploy our solution securely over the Internet. It uses standardized authentication tools embedded in any email client. Users can set up a project and schedule meetings using email. We also use email to distribute a session key, so we can maintain confidentiality.

DBSM features

We developed the DBSM concepts and technologies under a European Aeronautic Defense

State of the Art in Distance Learning

Researchers have proposed several different styles of distance learning in the literature. Several solutions focus on broadcasting knowledge. Egan et al.[1] used television to broadcast courses. In that case, the main drawback of television instruction is the lack of interaction between faculty and students.

Others use the Web to support asynchronous activities. Online programs use Web pages to access course materials, announcements, electronic libraries, and other information. Online activities can include forums using threaded bulletin boards, chat rooms, and email. In addition, the Web lets students submit assignments online in multimedia formats and receive their professor's online reviews of assignments in the same formats. However, students still don't have direct interaction with professors and other students.

Maly et al.[2] developed a virtual classroom where a student has a conventional classroom experience through a workstation. The teacher and students interact with videoconferencing. They share tools like Netscape and PowerPoint through a window-sharing engine. The environment incorporates XTV,[3] an X-Windows-based group-collaboration system. Thus, any participant can take control of a window to multicast his inputs to the distant participants. XTV is based on the Remote Multicast Protocol protocol[4] that achieves a reliable and ordered multicast. The key disadvantages of the environment are speed and bandwidth. First, an extraordinary load is put on the reliable multicast protocol because it must replicate the X traffic. Second, the environment only works on high-speed network. Typically, the participants are connected through a high-speed local area network, so the environment is devoted to Intranet use.

Arikawa et al.[5] proposed a server for managing an augmented 3D scene. They rendered graphical objects and videos at the same time onto the display. Incorporating live videos into virtual spaces makes the virtual space more attractive. To share the virtual space, the multimedia application uses high-speed computer networks with asynchronous transfer mode links at 155 Mbps. To give smooth motion from one space to another, a prefetching technique is necessary, where the multimedia application prefetches adjacent virtual spaces when required.

This solution resembles MPEG-4,[6] where a server broadcasts a virtual scene including videos. In contrast, MPEG-4 multiplexes and synchronizes the data associated with media objects so that they can be transported over network channels providing a quality of service (QoS) appropriate for the nature of the specific media objects. If a user wants to send information to other users, the interaction must be sent to server and processed before being forwarded to the distant users.

Saini-Eidukat, Schwert, and Slator[7] studied how to immerse students in a VE that provides them with feedback. The system computes the student interactions, thus reacting like an educational game. Their purpose was to facilitate the design of educational games, so they didn't focus on how to share a virtual world among students.

Collaborative environments let a stand-alone application be shared among participants. The general idea is that inputs occurring at a user interface are caught by a cooperation module/library that is integrated in the local application. Thus, the cooperation module/library sends the local inputs to the server that distributes them to the distant applications. For example, Saddik et al.[8] used JavaBeans and Java Remote Method Invocation (RMI). The different solutions from the literature[8,9] all use a central server. The server receives a message from a replica, processes the message, and sends out further messages to the other replicas. The Java Shared

Space Company, Research Center (EADS-CCR) contract. (See Sgambato's thesis for the content this contract covers.[2]) The DBSM merges disconnected works into a shared scene that satisfies the functional relationships between the different works. After a global-scene merging, meeting participants interact efficiently through the DBSM by updating the shared scene. The DBSM implements a distributed designation, distributed protection, and ownership transfer. Thus, it supports parallel working and work progression both online and off-line.

The DBSM has two phases. In the management phase, a group of participants forms and uses email to communicate securely. The collaboration phase lets the participants share a 3D scene. We can subdivide the collaboration phase into the scheduling and working phases. Costantini et al.[3] give the basic principles of the DBSM. Here, we give a wider description and we add detail to the working phase.

Preparation and scheduling

During the management phase, the DBSM provides two major services:

- *Project negotiation*: A project manager and selected participants communicate by email to set up a project. We achieve security using the Secure Multipurpose Internet Mail Extension (S/MIME) for authentication and confidentiality through X509 certificates. The negotiation phase doesn't require a specific tool.
- *Management transfer*: Managers send an email to transfer the responsibility of their project. That email contains a signed text certifying the transfer.

After the project negotiation phase, a scheduling phase lets the project members prepare a

Data Toolkit uses the same principle where broadcasting by the server can be replaced by a reliable multicast protocol.

The main drawback of a central server is a poor performance. First, a central server introduces a bottleneck in the system. Second, these solutions generally use multiple transmission control protocol (TCP) connections either in straightforward manner or through RMI invocations. Third, using a reliable multicast doesn't always improve the performances and doesn't guarantee a consistent progression of the work.

Distributed virtual environments are guided by the historical requirements coming from battlefield simulations. Thus, several works consider how to reduce the network traffic due to a high number of moving objects.[10-13] Generally, DVEs provide a way to divide the scene statically or dynamically for scalability. These systems don't support parallel work[10] or don't guarantee the work consistency.[11,12,14] Moreover, client–server pitfalls[12,14] or QoS requirements[11] limit these solutions.

References

1. M. Egan et al., "Learner's Perceptions of Instructional Delivery Systems: Conventional and Television," *The American J. Distance Education*, vol. 2, no. 4, Aug. 1993, pp. 47-55.
2. K. Maly et al., "Interactive Distance Learning over Intranets," *IEEE Internet Computing*, vol 1, no. 1, Jan./Feb. 1997, pp. 60-71.
3. H. Abdel-Wahad and K. Jeffay, "Issues, Problems, and Solutions in Sharing X Clients on Displays," *J. Internet-Working Research and Experience*, vol. 5, no. 1, Mar. 1994, pp. 1-15.
4. B. Whetten, T. Montgomery, and S. Kaplan, "A High Performance Totally Ordered Multicast Protocol," *Theory and Practice in Distributed Systems*, LCNS 938, Spriner Verlag, Berlin, 1994, http://research.ivv.nasa.gov/RMP.
5. M. Arikawa et al., "Real-Time Spatial Data Management for Scalable Networked Augmented Virtual Spaces," *IECIE Trans. Information and Systems*, vol. E82-D, no.1, 1999, pp. 099-112.
6. R. Koenen, *Coding of Moving Pictures and Audio, MPEG-4 Overview* (V.18 – Singapore Version), ISO/IEC JTC1/SC29/WG11-N4030, Int'l Organization For Standardization, Mar. 2001, http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm.
7. B. Saini-Eidukat, D.P. Schwert, and B.M. Slator, "Designing, Building, and Assessing a Virtual World for Science Education," *Proc. Int'l Conf. Computers and Their Applications*, 1999, pp. 59-65.
8. A. Saddik et al., "Collaborative Working with Stand-Alone Applets," *Proc. Intelligent Mutimedia and Distance Education* (ISIMADE 99), 1999, pp. 203-209.
9. C. Kuhmünch, T. Fuhrmann, and G. Schäppe, "Java Teachware—The Java Remote Control Tool and its Applications," *Proc. World Conf. Educational Multimedia and Hypermedia and World Conf. Educational Telecommunications* (ED–MEDIA 98), Association for the Advancement of Computing in Education, 1998, http://www.informatik.uni-mannheim.de/~cjk/publications/ed-media98.
10. J.W. Barrus, C. Waters, and D.B. Anderson, "Locales: Supporting Large Multiuser Virtual Environments, *IEEE Computer Graphics and Application.*, vol. 16, no. 6, Nov. 1996, pp. 50-57.
11. Defense Modeling and Simulation Office, *HLA Data Distribution Management Design Document Version 0.5*, US Dept. Defense, Washington D.C., 1997, http://www.dmso.mil/index.php?page=64.
12. O. Hagsand, "Interactive Multiusers VEs in the DIVE System," *IEEE MultiMedia*, vol. 3, no. 1, Spring 1996, pp. 30-39.
13. M.R. Macedonia et al., "Exploiting Reality with Multicast Groups," *IEEE Computer Graphics and Applications*, vol. 15, no. 5, Sept. 1995, pp. 38-45.
14. W. Broll, "DWTP-An Internet Protocol For Shared Virtual World," *Proc. Int'l Symp. Virtual Reality Modeling Language* (VRML 98), ACM Press, New York, pp. 49-56.

meeting. It provides the following services:

- *Scheduling*: A subset of the project members uses email to schedule a meeting.

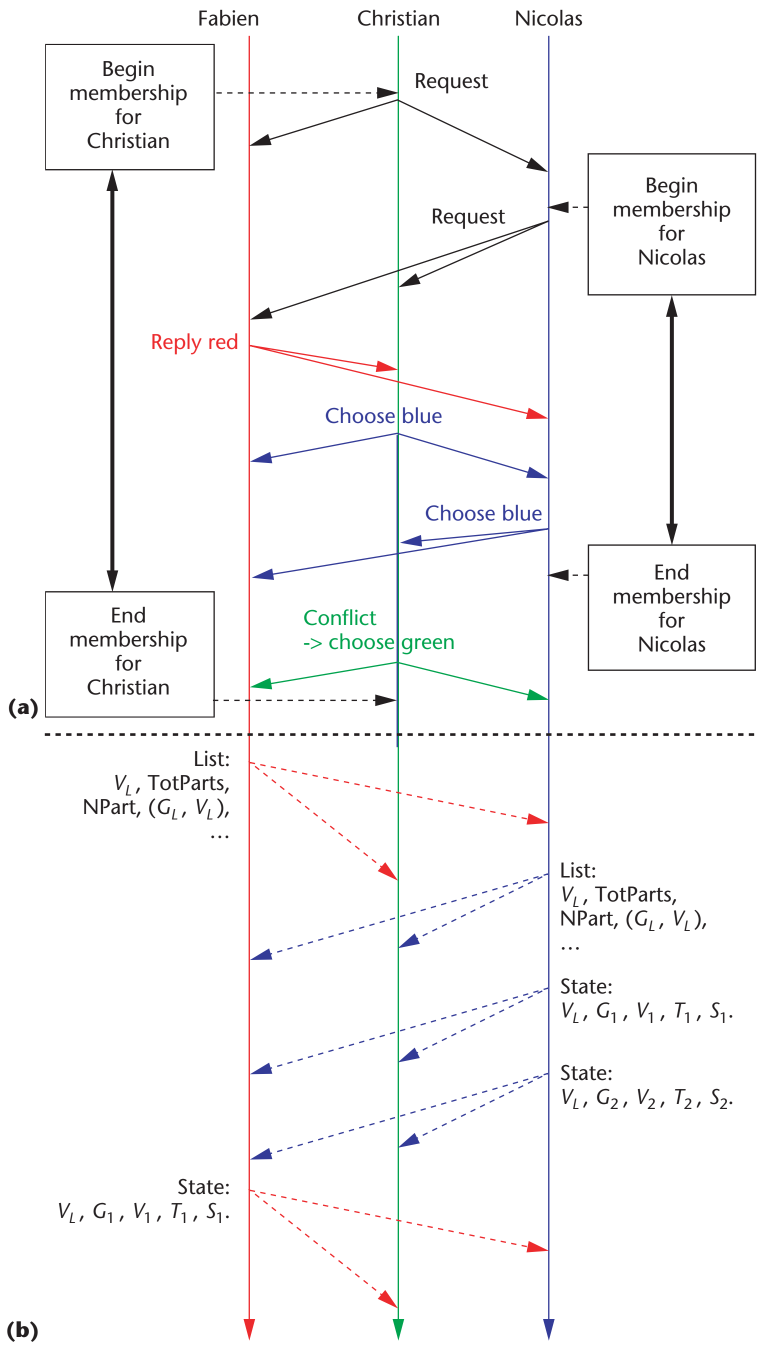
- *Address allocation*: During the scheduling phase, the project members negotiate a multicast address by email. In practice, each recipient can use a local directory service (that is, Lightweight Directory Access Protocol) to reserve the requested address. We can omit this reservation procedure and detect and resolve a conflicting address automatically during the meeting.

- *Distribution of a session key*: The scheduling phase serves also to distribute a private key K, which the participants will use as a session key during the scheduled meeting. That private key is transmitted securely by email using S/MIME to authenticate the participants and encrypt the key.

Meeting

A meeting runs in three phases. The first phase is real-time gathering and global-scene merging. Thus, each participant gets a copy of the shared scene using an application peer. Peers communicate through the DBSM services. During a second phase, different operations (object creation, deletion, or modification) process the shared scene. In the third phase, participants leave the meeting and take with them a subset of the shared space.

Real-time meetings. An entering peer multicasts participants' names as they join. Distant participants reply by multicasting their names. Thus, a new participant maintains his knowledge



Fabien
Christian
Nicolas
Begin membership for Christian
Request
Begin membership for Nicolas
Request
Reply red
Choose blue
Choose blue
End membership for Nicolas
End membership for Christian
Conflict -> choose green
(a)
List: V_L, TotParts, NPart, (G_L, V_L), …
List: V_L, TotParts, NPart, (G_L, V_L), …
State: V_L, G_1, V_1, T_1, S_1.
State: V_L, G_2, V_2, T_2, S_2.
State: V_L, G_1, V_1, T_1, S_1.
(b)

Figure 1. (a) Real-time membership and (b) gobal scene merging. The distributed membership protocol maintains for each communication peer the meeting membership and colors of the participants. The global-scene merging builds a copy of the shared scene for each communication peer by aggregating the different pieces at the right functional places.

of the meeting membership locally. The protocol allocates a session color to each participant. (See Cosstantini et al.[3] for more on real-time meetings.) The top of Figure 1 gives an example of this phase. It shows Christian entering concurrently with Nicolas. The protocol resolves the conflicting color (blue) at the end of Christian's membership phase. Only Christian's peer multicasts a new message since it has the smallest IP address. Finally, note that the mechanism can have minor side effects like when Christian's color changes from blue to green during the same meeting.

Global-scene merging. After the system achieves a real-time meeting, it processes a global-scene merging. Each peer multicasts a list part [List: V_L, TotParts, NPart, (G_1, V_1), …, (G_N, V_N)], where V_L is the version number of the list, TotParts is the total number of parts forming that list, NPart is the part number, and each couple (G_X, V_X) defines the global name and the version number for an object X. Using TotParts list messages, a peer announces the objects it enters into the meeting. Afterwards, the announcing peer multicasts an object state [State: V_L, G_X, V_X, T_X, S_X] for each object X of the list where T_X is the type of X and S_X is the state associated with V_X. Thus, a distant peer recovers all the announced objects. At the end of the scene merging, each peer has a copy of the shared scene.

Because a distant peer has a list of the objects, it can request retransmission of a missing object X by asking for a version number equal or higher to V_X. Thus, an emitting peer simply only has to retransmit its current version V'_X of X ($V'_X \geq V_X$).

Because each state transports a list version V_L, a receiving peer that misses the corresponding list message can ask for a retransmission by requesting the list part for the object G_X. Generally, the emitting peer will retransmit the corresponding list part. When the emitter adds or retrieves an object, the requested version V_L is no longer available because the list has changed. In that case, the emitter retransmits all the new parts with $V'_L \geq V_L$. After receiving the new list, the requester must request retransmission of the missing states based on the received list.

We should note that a selective list [SLIST: V_L, T_L, TotParts, NPart, (G_1, V_1), …, (G_N, V_N)] transmits only object references of type T_L. A selective list is a subset of the list V_L. Figure 1b illustrates the messages exchanged during the global-scene merging phase.

Real-time operations. A peer modifies an object X by multicasting a message [State: V_L, G_X, V_X, T_X, S_X]. A receiver uses the event either to cre-

Related Work

Other collaborative learning solutions use a server to download the scene[1] and manage the meeting membership.[2] Hagsand[1] used a reliable multicast with negative acknowledgements to continuously provide fresh updates and recover errors. Several peers reply to the negative acknowledgement. Broll[2] developeds a classical reliable multicast with positive acknowledgements for the events requiring reliability. However, neither Hagsand nor Broll guarantee that users are modifying the latest state of the corresponding object. Generally, they don't give a clear definition of the consistency they tackle. Hagsand used a token-passing algorithm to schedule the concurrent requests. To guarantee a copy's freshness, Broll periodically transmitted the state of the VRML nodes and required precise clock synchronization.

Handley and Crowcroft[3] also considered freshness as a consistency property and used synchronized clocks to maintain a fresh copy. They don't guaranty that a site modifies an object starting from its latest state since two modifications occur simultaneously. Generally, their solutions don't integrate security properties or define how to allocate a multicast address. They also don't consider how to disseminate a global scene tree among different workers for disconnected improvements and reintegration into the shared environment.

References

1. O. Hagsand, "Interactive Multiusers VEs in the DIVE System," *IEEE MultiMedia*, vol. 3, no. 1, Spring 1996, pp. 30-39.
2. W. Broll, "DWTP-An Internet Protocol For Shared Virtual World," *Proc. Int'l Symp. Virtual Reality Modeling Language* (VRML 98), ACM Press, New York, pp. 49-56.
3. M. Handley and J. Crowcroft, "Network Text Editor (NTE) A Scalable Shared Text Editor for the Mbone," *ACM Computer Sommunications Review*, vol. 27, Oct. 1997, pp. 1997-208.

ate a new object or update an existing object. The receiving peers don't acknowledge that message.

When a peer deletes or creates an object, it changes its list (**`myList:=myList-X`** or **`myList:=myList+X`**) and increments the list version (**`myListVersion++;`**). Afterwards, the peer multicasts a message [Deletion: V'_L] or [State: V'_L, G_X, V_X, T_X, S_X], where V'_L equals **`myListVersion`**. A receiving peer removes or adds G_X and updates its local knowledge of the distant list. Thus, an emitter multicasts a small deletion or update message instead of using the list messages. The receiving peers don't need to acknowledge these operations.

Shared scene tree. The global scene is a scene tree where each node contains graphical and collaboration attributes. The application maintains that tree and inherits the collaboration attributes from the DBSM classes. Each peer builds a copy of that global scene tree by collecting the different subtrees through the global-scene merging. A central server doesn't maintain the global scene. It's accessible when all the participants reach the meeting, but an absentee doesn't prevent the scene from merging. If participants are absent, the shared scene is a subset of the global scene.

The scene tree is an application tree that gives high-level, concise information about the graphical objects. For example, a pipe is defined by its diameter and a set of routing points. So a peer multicasts a small amount of data for the update of an object. For example, a peer transmits the diameter and routing points but not the different polygons to draw the pipe. Thus, a receiver will compute the corresponding polygons locally.

Collaboration attributes include ownership and protection attributes. Protection attributes prevent distant participants from observing or modifying the corresponding object. There's only one owner of a given object at a time. Only the owner can modify the corresponding object, but the ownership can be transmitted.

Distributed designation. The DBSM locally computes a unique name when a designer creates a new node. A unique name contains the IP address (**`@IP`**) of the creation machine, a local time stamp, and the position within the tree—for example, **`@IPA,stampA,1.3`** corresponds to the first child and third grandson of the node **`@IPA,stampA`**. Only a node's owner can create a child for that node. For example, A creates the node **`@IPA,stampA,1.3`** as owner of **`@IPA, stampA,1`**. That way, unique names are defined in a distributed way by each computation peer. Moreover, the system can create those distributed names before the part reaches the meeting. A participant doesn't need any name server to create a unique name. The father–son relationships are implicitly maintained by the distributed names, so the system doesn't have to explicitly store those relationships.

A time stamp includes a date corresponding to the local creation date and a random number. Therefore, a peer can change its IP address with-

The ownership transfer guarantees that a modification is processed starting from the latest state of the corresponding object.

out any difficulty. An already created name remains unique in time and space because a name is unchanged during the node life. The new IP address will produce distinct names because the probability that two machines will use the same address to produce the same name (same local time and same random number) is close to zero. If in the future another machine allocates the same IP address with completely desynchronized clocks, the random numbers will discriminate the two names.

Responsibilities and permissions. The DBSM provides the basic services that let the application respect the different responsibilities and permissions. The application uses the project definition (transmitted during the project negotiation) and protection attributes of the nodes to control the requested operation's validity.

Refreshing. Since receiving peers don't acknowledge updates and deletions, a distant peer can lose events. Using the refresh service, a peer requests a list from a distant peer. The requester replies by multicasting the list messages. The requesting peer uses the received list to remove or add objects. If the objects already exist, that list lets the peer request retransmission of newer versions. Therefore, the system only transmits missing versions. A retransmission request is a kind of negative acknowledgement (NACK), but NACKs aren't generated each time a version is missing or to recover all of a given object's state. The refresh service is used typically on application demand to process an accurate copy of the scene.

The retransmission request and the reply are multicast. Thus, a peer observing a retransmission request from another peer waits for the reply. Finally, a peer packages different retransmission requests into a single multicast message, so only a small number of retransmission requests are multicast.

Consistency through the ownership transfer. The ownership transfer guarantees that a modification is processed starting from the latest state of the corresponding object. The owner refuses the ownership transfer when the user disables a write attribute. Otherwise, the requesting peer receives the current state and the ownership at the same time.

A transfer runs in three phases. First, the requester multicasts a message to locate the owner. Second, the owner replies to the requester with a point-to-point message including the ownership and the node's current state. Third, when receiving the reply, the granted peer sends a point-to-point acknowledgement that terminates the transfer.

The ownership transmission must be reliable, so a local number is associated with the request. The owner replies with the same number. When receiving the reply, the requester sends an acknowledgment, including the same number. The requester resends a request in absence of a response. The granting peer resends the reply in absence of an acknowledgment. This avoids faulty situations where an object has no owner because of a transmission error.

Real-time awareness. Because each modification (update and deletion) is multicast, a receiving peer is aware of the distant modification. Moreover, a receiving peer speeds up the delivery of a recent update by discarding an older version. This reduces the workload because the receiving peer doesn't process out-of-date updates.

Parallel work. Two distant peers process simultaneously two different tasks dealing with two distinct nodes. Two tasks for the same node are serialized to guarantee the consistency property. The peer that currently owns the node processes the first task. Afterwards, the owning peer processes the ownership transfer with the distant peer to carry out the second task. Thus, the second task starts processing the node after the first task ends.

Security and address allocation. By using the session key K, the DBSM symmetrically encrypts each message. The shared secret K is distributed securely with S/MIME when the participants schedule the meeting. If another application uses the same multicast address, the DBSM will automatically detect the situation because the message can't be decrypted using the session key K. The peer with the smallest network address looks for a free address by listening for activity on that address. Afterwards, the peer transmits the new address to the distant peers. On receiving the proposed address, each peer will join the new multicast address. This mechanism lets the system allocate and exchange securely a new address during the meeting.

Leaving. Departing participants select the nodes that they want to keep in their isolated workspace. All the owned nodes are automatically included in their isolated workspace, but they can also select other nodes. Thus, departing participants define the subset of the shared scene they include in their isolated workspace. At the end of the selection, the peer multicasts a leaving message to inform the distant participants of its departure. Then, the peer leaves the multicast address.

The leaving message doesn't require an acknowledgment. An unreliable departure suffices as the peer already has the selected nodes within the user's isolated workspace. When it's leaving, the peer will normally save the isolated workspace within a local store. Thus, designers can recover their isolated workspace at any time. Furthermore, the saved workspace lets users make improvements without a network connection. Users can introduce the results of their offline work in a subsequent meeting.

Collaborative training of industrial designers

DBSM was originally integrated into a virtual prototyping tool to support collaborative design of aeronautical systems. Here, we present a novel approach where that collaborative virtual prototyping environment (the virtual prototyping application that integrates DBSM) is used to instruct engineers to build aeronautical systems. Experimented engineers (instructors) teach novice engineers to build aeronautical systems in interactive design session. The novice engineers learn at the same time how to use the collaborative virtual prototyping environment and how to build aeronautical systems. First, we describe how an instructor gives cooperative lessons. Second, we present how learners cooperate to realize a virtual prototype for a design exercise.

Collaborative teaching

The instructor comes to the meeting with a prepared ventilation system to teach the scaling of the different pipes. Through the DBSM, each learner gets a copy of the ventilation system scene. Afterwards, the instructor starts a chat object to explain the case study. The application uses the DBSM to transmit an update for the chat object. In fact, the instructor can create different chat objects to distinguish the explanations. A receiver uses the refresh request to regularly ask for a list of the chat objects.

We present a novel approach where that collaborative virtual prototyping environment (the virtual prototyping application that integrates DBSM) is used to instruct engineers to build aeronautical systems.

The instructor uses the collaborative virtual prototyping environment to instruct the learners in using the environment and to give lessons on aeronautical design. The instructor uses the generic interface to design a virtual mockup. Because his work drives the distant interfaces of the learning group, the distant participants observe the instructor's work.

The instructor can use a shared viewpoint to navigate the students within the scene. Shared pointers let the instructor designate specific scene elements. A shared viewpoint is a classical object whose attributes update the distant viewpoints. A learner can take control of a shared viewpoint or pointer. For this purpose, the application draws a tree of the objects and the user selects the desired object. Before updating the selected object, the system transfers ownership so that the requesting learner has the most current object. Thus, the requester can see the latest position of the current owner before getting control.

The application uses the DBSM to catch the user interactions and transmit them to the distant peers. The instructor adds some text describing the interaction. The application automatically creates a new object including the text, a step number, and the user interaction. When receiving the corresponding event, a peer processes it as a local interaction and creates the object to dis-

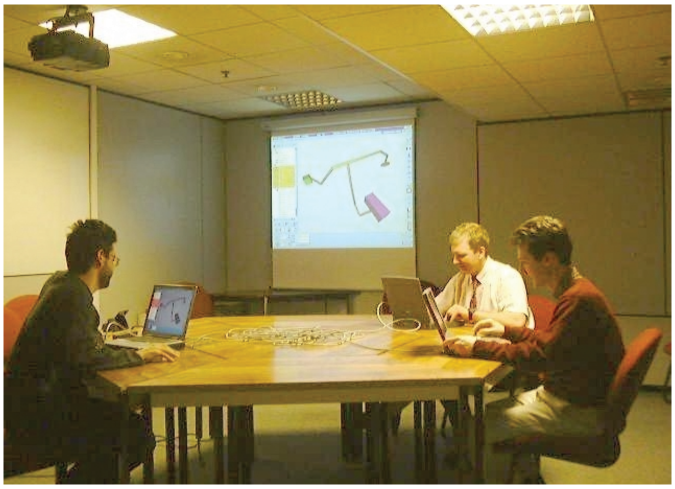


Figure 2. Mutual learning. Learners collaborate to mutually design a ventilation subsystem.

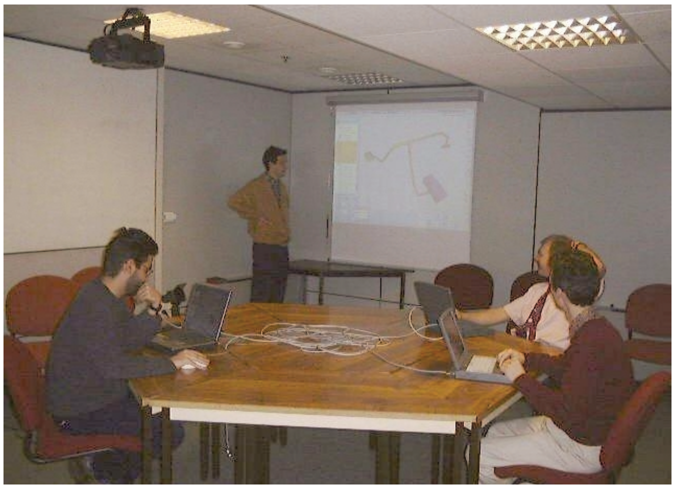


Figure 3. Collocated teaching. Collocated teaching lets the instructor give oral explanations using a video projector.

play the text and step number. A receiver periodically uses a refresh request to ask the emitter to send the list of the tutoring objects. Thus, a faulty peer can request missing tutoring objects. The texts appear in order of their step numbers. A learner can replay locally the interactions through the step numbers. Therefore, the refresh service can recover a loss.

Finally, the instructor modifies in real time the pipes to achieve the required debit and pressure for the ventilation system. Thus, he shows directly within the 3D scene how he solves a specific problem. The instructor uses his expertise to teach a difficult case and the shared scene mediates the knowledge transmission.

Our system can use videoconferencing tools like Microsoft NetMeeting instead of chat objects. Thus, the participants could communicate with external tools while sharing the virtual scene.

Mutual learning

The participants also learn from each other by observing distant interactions and by collaborating on a task. For example, if an instructor proposes a design exercise for a ventilation subsystem, learners can collaborate to build the ventilation subsystem. By observing the distant modifications, a designer can learn how to do the requested subsystem.

Moreover, any designer can help another participant by directly modifying distant objects. This guarantees a consistent progression of the work. The participant requesting ownership will recover the latest state of the corresponding object. Thus, a participant won't have to redo the same work because of conflicting actions. The DSBM avoids these problems because it serializes the conflicts.

Figure 2 shows three learners sharing a 3D scene. Because they're in the same room, they don't need to use a chat facility or a videoconferencing tool. In Figure 3, the instructor advises the learners using the scene displayed with a video projector. In a distant configuration, the instructor uses the chat facility, shared viewpoints, shared pointers, and annotations to give advice.

Because our solution also supports distributed tutoring, several instructors can collaborate with a group of learners in a shared scene.

Homework review

At the end of a lesson, learners typically move their work to their isolated workspace. That way they can do homework and improve a subset of the global shared scene while they're offline.

At a later meeting, the different learners can share their homework and merge them into a global scene. Each learner can also introduce different proposals into the meeting. Thus, the participants study different alternatives.

The system processes real-time modification so learners collaborate to find the best solution. Typically, a peer uses the refresh service to resynchronize the copies with fresh states. Afterwards, learners can propose direct improvements to the solution by creating, deleting, or modifying objects. The DBSM guarantees that the learners recover a consistent state for the involved object before processing their interactions—that is, a work progression guarantee.

Implementation issues

The DBSM uses the Adaptive Communication Environment (ACE),[4] so it runs on both Unix and Windows. DBSM is a C++ library. An object application inherits from the DBSM classes.

The DBSM takes 6.75 seconds to merge a shared scene of 1,500 objects between two participants with 750 objects at each side on a Sun UltraSparc 10. This time includes the transmission time and the processing time.

The average size of an update message is 150 bytes for simple objects like a cube, sphere, or cylinder. Thus, the transmission time is 120 μs on a 10-Mbps network and 35 ms on a 33.6-Kbps line. The latency between a local interaction and the update of a distant copy is 16 ms at 10 Mbps and 100 ms in a wide area network at 33.6 Kbps.

Conclusion

The DBSM enables collaborative training and mutual learning in a consistent, up-to-date shared virtual environment.

Currently, we are working on a new and free implementation to replace DBSM and better support various applications, such as collaborative virtual prototyping, distributed games, and distributed simulation. An important lesson we learned is that integrating the cooperation services into an existing application must be easy. The new solution will use design patterns to improve reuse and ease integration. Moreover, we are studying integration of off-the-shelf simulators (such as hydraulic or electrical simulators). Thus, designers could collaborate to quickly set up a distributed simulation within the shared virtual scene. We started that latter work within the European Information Society Technology/Advanced Information Technology, Virtual Early Prototyping Open Platform—the IST/AIT-VEPOP project. **MM**

Acknowledgment

The figure images are courtesy of European Aeronautic Defense Space Company, Research Center (EADS_CCR).

References

1. C. Toinard, G. Florin, and C. Carrez, "A Formal Method to Prove Ordering Properties of Multicast Systems," *ACM Operating Systems Review*, vol. 33, no. 4, 1999, pp. 75-89, http://cedric.cnam.fr/~toinard/Recherche.
2. A. Sgambato, *Distributed Building Site Metaphor*, enginering thesis, Nat'l Conservatory of Industrial Arts and Crafts (CNAM), 1999.
3. F. Costantini et al., "An Internet Based Architecture Satisfying the Distributed Building Site Metaphor," *Proc. Multimedia Computing Track* (IRMA 2000), Idea Group Publishing, Hershey, Pa., 2000, pp. 151-155, http://cedric.cnam.fr/AfficheArticle.php?id=250.
4. D.F. Box, D. Schmidt, and T. Suda, "ADAPTIVE: An Obkect-Oriented Framework for Flexible and Adaptive Communication Protocols," *Proc. Fourth Conf. High-Performance Networking*, Intl' Federation for Information Processing, 1992, pp. 367-382.

Fabien Costantini is a PhD student in computer science at the Center for Study and Research in Information Technology at the National Conservatory of Industrial Arts and Crafts (CNAM-CEDRIC), Paris, France. Before that he worked with a European company manufacturing aeronautical systems, where he participated in the conception and development of a large collaborative virtual prototyping tool using the distributed building site metaphor. His research focuses on a general distribution framework that relies on DBSM principles as well as new architectural design patterns for implementing highly reusable and portable object-oriented software solutions.

Christian Toinard is an associate professor in computer Sciences at National Engineering Scholl in Electronic, Computer Science, and Radio Communications of Bordeaux (ENSEIRB), France. His research activities are at the CNAM-CEDRIC. He is also a technical leader in the European Information Society Technology/Advanced Information Technology, Virtual Early Prototyping Open Platform (IST/AIT-VEPOP) project funded by the European Commission. He has published several articles on cooperative virtual prototyping of aeronautical systems and object-oriented distributed systems.

Readers may contact Toinard at the Centre for Study and Research in Information Technology (CEDRIC), 292 rue Saint-Martin 75141 Paris Cedex 03 France, email toinard@cnam.fr, http://cedric.cnam.fr/~toinard.