**Technical Report TR02-009**

Department of Computer Science
Univ of North Carolina at Chapel Hill

# Hypervideo Support for Distributed Extreme Programming

**David Stotts, Jason Smith**
Dept. of Computer Science
Univ. of North Carolina at Chapel Hill

**Laurie Williams**
Dept. of Computer Science
North Carolina State University

Dept of Computer Science
Univ. of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175

stotts@cs.unc.edu

March 1, 2002

# Hypervideo Support for Distributed Extreme Programming

David Stotts, Jason Smith
Dept. of Computer Science, Univ. of North Carolina at Chapel Hill

Laurie Williams
Dept. of Computer Science, North Carolina State University

## Abstract

*We present a hardware/software system for support of distributed Extreme Programming, or DXP. It consists of a dual video projector setup with NetMeeting running on a single PC, and a hypervideo system we built called OvalTine. OvalTine allows the creation and automatic tracking of hyperlinks in a video stream, both archived and real-time (as in video conferencing). DXP supports the development of software by a pair of programmers that is non-co-located. One projector displays a shared PC desktop, and another projector displays a life-sized image of each collaborator to the other. OvalTine integration means developers can add hyperlinks to the collaborator camera stream, integrating it with Web pages or other video streams. The DXP work environment gives a better sense of "being there" to the pairs in a pair-programming team. We are experimenting with DXP to measure the productivity of distributed software developers working via different technical processes (paired, individual, full XP, traditional).*

## 1. Pair programming, XP, and distributed collaboration

Increasingly, programmers are working in geographically distributed teams. Escalating trends in teleworking, distance education, and globally distributed organizations are making these distributed teams an absolute necessity. These trends are beneficial in many ways, particularly for those in geographically disadvantaged areas. However, it is not believed that any of these arrangements makes a programmer more effective than if all the programmers were, indeed, co-located. Therefore, organizations must strive to maximize the efficiency and effectiveness of these unavoidably distributed programmers and teams.

This paper describes the development and study of a technique tailored for distributed programming teams. The technique is based on an emerging software engineering methodology known as *pair-programming* combined with nearly 20 years of widespread and active research in collaborative software systems. We aim to show that geographically distributed programmers benefit from using technology to collaborate synchronously with other programmers. Our objective is to demonstrate that the geographically distributed programmers who collaborate synchronously with other programmers will outperform geographically distributed programmers who work independently.

Professional interest in pair programming has risen dramatically in recent years with the success of an agile software development process called *Extreme Programming*, or *XP* [1,2] developed by Beck. XP is distinguished from more traditional development processes by emphasizing (even requiring)

- pair programming
- test-first code development at the unit level
- full regression test support (with JUnit [3,4])
- lack of up-front detailed design
- frequent code refactoring [5]
- on-site client
- expectation of requirements changes.

XP practitioners develop requirements conversationally with the client, and deliver frequent working prototypes for feedback and changes. No code is written unless it is needed (no programming for the "perhaps" future), and when a design grows to the point that it feels unwieldy it is refactored and re-architected before further extension.

The process is best known, though, for pair programming, which is at the heart of the productivity increases many XP teams are seeing.

Anecdotal and statistical evidence indicate that pair-programming—*two* programmers working side-by-side at *one* computer, collaborating on the same design, algorithm, code or test—is highly productive. Cockburn and Williams have produced statistical results which show that pair-programmers produced higher-quality products in essentially half the elapsed time as individual programmers [6]. We believe the pair-programming model can be modified for distributed development, and that we will see similar benefits.

To evaluate the effectiveness of distributed pair development, we are running controlled empirical studies with students at between North Carolina State University and the University of North Carolina at Chapel Hill. Students use interactive information technology over the Internet, such as *PCAnywhere* and *NetMeeting,* to jointly and simultaneously control a programming session and to speak with each other synchronously. Our early experiments have led us to develop the more video-enhanced environment desacribed here to support the remote synchronous collaboration required in DXP software development.

We first give some background on environments for support of collaborative activities. After that we describe the DXP hardware/software environment we built for distributed XP. We then describe the OvalTine system for hyperlinking in real-time video streams and show how it is being integrated in our DXP environment.

## 2. Background: Collaborative systems and distributed workgroups

The earliest example of a collaborative computer system was NLS-Augment by Engelbart [7], an initial version of which was demonstrated in the early 1960's. Engelbart's system used shared CRTs, audio connections, mouse, and keyboard to allow crude teleconferencing and shared examination of text files by users who were not co-located. From these early beginnings, collaborative software systems became the subject of widespread research more than 15 years ago, with the creation of the PC. Ongoing research tends to focus in three main areas: hardware to provide effective communications; software concepts that allow sharing of artifacts; and conceptual models of how people want to — or are able to — interact effectively.

Many collaborative system ideas developed in research labs have now found their way into practice, and we are seeing commercially viable products and services for supporting collaboration. These products are used to create virtual workplaces and allow people around the world to work on coordinated group efforts. The most commonly used collaborative tools are "chat rooms". Though simple in technical terms, chat rooms' wide usage demonstrates that *simple technology can be used very effectively*. Many commercial vendors support chat servers for exchange of text, audio, and video streams (Yahoo, Excite, PalTalk, and Microsoft to name a few) and hundreds of thousands of people use these services. Most usage of chat servers is for social interactions like game playing, but they are also used to support businesses. *We repeat that simple technology often gets you large gains.*

### *Shared software artifacts*

Numerous researchers have developed software for supporting human interactions within shared artifacts. These include general systems, like shared whiteboards for drawing, and shared editors for documents and multi-media streams. They also include special-purpose systems like ICICLE [8] for code inspection and review, and gIBIS for design review and logical argument construction [9]. Trellis [10] and MMM [11] are two systems for collaborative hypermedia and Web browsing. Trellis was built on the idea of structuring a hyperdocument as an abstract parallel process specification, allowing group interaction on the document to be defined in the document link structure. MMM extended this idea by extracting the process definition out of the document and allowed the group interaction rules (collaboration protocol) to be dynamically defined (as opposed to being hard-wired in the source code). In this way, group interactions can be developed and changed without changing the MMM software. Further research has led to methods for verifying the correctness of collaboration protocols [12,13].

### *Hardware and graphics for collaborations*

Many of the most visible developments in collaborative systems have come from computer graphics. While their display technologies have remarkable sophistication in their visual imagery, most do not support *distributed* collaborations. The CAVE [14] is a multi-person, room-sized, high-resolution, 3D video and audio environment developed at the University of Illinois at Chicago. It functions as virtual reality theater, made up of three rear-

projection screens for the front, right and left walls and a down-projection screen for the floor. Multiple users may sit in the space, wearing special glasses to decode the stereo projections. The CAVE is state-of-the-art in terms of visual impact and virtual reality presentation; it is limited in distributed collaboration support as it requires all participants to be present in the same space in order to work together. The DataWall [15] at MIT and the PowerWall [16] at the University of Minnesota are similar large display projects. The primary purpose of these large displays is to visualize and display very high-resolution data, often from large scientific simulations performed on supercomputers or from high resolution imaging applications. The large display areas allow small groups of collaborators to see the display clearly and without obstruction. It is possible to walk up to the display and point to features of interest, just as one would do while discussing work at a blackboard. However the emphasis is on graphics clarity, and not remote collaboration.

Unlike the previously mentioned display projects, Virtual WorkSpace [17] was intended as an environment to enable distributed collaboration over a network. It depends heavily on computer-generated graphics and virtual reality devices as well. ClearBoard [18] was similarly a non-co-located collaboration support system that allowed two users to appear to sit face to face, and see the shared work between them; emphasis was placed on drawing applications. The system required special hardware (the clear screen), and was not built with COTS technology (as is our environment). Experiments using ClearBoard showed that *increased eye contact and the sense of presence of the remote collaborator was important in providing effective work performance.*

The *Office of the Future* project [19] at UNC, under the direction of Henry Fuchs, seeks to combine network-based collaboration with the superior graphics and image manipulation capabilities of virtual reality systems. It is a long-term project that will not be generally usable for years and it will require expensive special-purpose support hardware and high-performance graphics engines.

***"Office of Real Soon Now" and VideoWindow***

Even with the remarkable graphics technology available at the high end of virtual reality systems, much remains unchanged from Englebart's early prototype. The dominant paradigm of human-computer interaction in the workplace — a single user sitting in front of a single display with limited resolution and a WIMP interface, attached to a single computer — remains in force and has become a barrier to effective communication and collaboration for group interactions. The research we are doing is trying to break that barrier with relatively simple technology, compared to the systems just described.

We are building a more effective collaborative environment for pair programming using the results of some simple wall-size display experiments at UNC [20, 21]. Whimsically termed the *"Office of Real Soon Now"* (a play on the name of the *"Office of the Future"*), it aims to get some of the benefits of large screens without waiting years and spending large amounts money. In this project, Bishop and Welch have produced double-width wall-sized displays for their offices using COTS projectors, video cards, and PCs. For their experiments they completely abandoned CRT displays and used only projected wall displays; after 3 years neither has any intention to return to CRTs. Benefits of the large wall displays include greatly reduced eye-strain; better interaction capabilities with students when discussing joint work; and expanded screen real-estate. Their experiments have concentrated on individual and co-located group use of the wall display technology, and have not involved networked collaborations.

Just as the "Office of Real Soon Now" seeks to have large-display benefits well in advance of the Office of the Future, we seek the "real soon now" benefits for distributed pair programming by using inexpensive COTS projection equipment, and ubiquitous PCs. The design of our DXP environment employs the principles uncovered at BellCore in the VideoWindow project [22]. In this experiment, two rooms in different buildings at BellCore (coffee lounges) were outfitted with video cameras and wall-sized projections. In essence, an image of one lounge was sent to the other and projected on the back wall, giving the illusion in each room of a double-size coffee lounge. The researchers discovered that many users found the setup to be very natural for human communication, due to its size. Two people, one in each room, would approach the wall to converse, standing a distance from the wall that approximated the distance they would stand from each other in face-to-face conversations.

**Figure 1: DXP Collaborative pair programming setup (half)**

## 3: A Collaborative Environment for Distributed Pair Programming

Our current experiments in distributed pair programming between UNC and NCSU have shown that programmer communication via a 19" to 21" display, while effective enough to allow development of good software, result in interactions that are stiff and limited in scope. The pairs so far have been given tools that support video interactions via webcam and postage-stamp-sized video windows. After initially turning the cameras on along with the shared PC and the audio, all pairs soon turned the cameras off to enhance performance. They reported that the video was too small to provide them with any communications enhancements. We believe a far more effective collaborative environment can be created with a wall-sized display produced by projectors, and that a corresponding improvement in distributed pair programming will result from this enhanced video support for collaboration.

The equipment package needed for one office is:

- High-resolution video projector2 (2)
- Firewire camera + PCI video capture card (1)
- PC video card to handle two screens seamlessly (1)
- wiring, cable, microphones, screen boards, etc.

The cost for a single office is about $8,000. We are working with four packages, outfitting two offices at each of the two research sites (UNC-CH and NCSU). This arrangement allows "local" distributed pairing at each site ove the LAN, as well as pairing across sites with a wider-area network. Each office has two projectors. One is primarily used for video display of the remote collaborator. The other is for display of the shared computer "screen." We are starting with an L-shaped screen setup, with the collaborator video image to the side of the programmer and the computer display to the front. We have placed the camera next to the projection wall rather than on the workstation in order to present each user with a view the other's office, with a side-view of the collaborator in the foreground. We will experiment later with different user placements and screen arrangements.

Figure 1 shows half the setup at UNC. Visible are the two projectors pointed at right angles to each other, and one of the screens. As the developer sits, he sees the shared PC desktop projected ahead, and the collaborator projected to the left. To communicate with the collaborator the developer turns to the left and speaks to the screen. The camera location with the screen gives a nice impression of the pair being face-to-face. This mimics the head movement needed to look at one's pair programmer when working co-located.

We are working to eventually mix the video imagery (allowing the collaborators to see each other) with digital display information (the source code being developed), but for the first realization we use one projector for PC display and one for camera/video display. Communication is via directional microphones placed in the vicinity of the workstations, so the participants will not be encumbered with headsets. Two distributed collaborators interact much like they do with local pairing; to talk, one will turn to the other (face the projection wall) and speak openly in the room. Since the camera is on the projection wall, the remote collaborator will have the impression that the communicator is looking at him or her. Each will see surrounding context and an image of significant size. The illusion created is a "joint office" with the video walls, much like the virtual coffee lounge of BellCore's project *VideoWindow* mentioned previously.

***Software Platform: Video with hyperlinking***

In addition to this hardware environment, we are developing for DXP software tools to more effectively support interaction between distributed pair programmers while developing programming project artifacts. We will eventually need session-logging capabilities to capture the activity in a pair programming session for study. As we progress from the simple early arrangement, we will need software to control the projectors and mix the two sources (collaborator video image and shared PC output) in different ways, allowing different arrangements of the screens in the office as well as different layouts of information on the screens. The mixing software will need to take network capacities into account in terms of the quality of video images sent between sites; with 2 offices at UNC and 2 at NCSU we will be able to experiment with local pairs (ultra-high-speed network connections) as well as inter-site pairs (standard Internet connections with associated loads).

As experiments progress, we will seek to identify areas in which collaboration among the programming pairs would benefit from software support and to build any shared artifact tools we may need (editors, inspection tools, etc.). We intend to build tools as needed, but our first experiments are to determine the effectiveness of the simplest approaches, using the observation made earlier that *simple technology often reaps large benefits*. Thus our first experiments have been with a single virtual PC obtained via NetMeeting, on traditional PCs. NetMeeting provides not only a shared desktop, but audio communications as well.

One novel aspect of the DXP software environment is integration of the video stream from the camera with *OvalTine*, a hypermedia tool we developed at UNC to allow embedding of hyperlinks in video streams. In the section following we give an overview of the structure and image analysis techniques used in OvalTine to do video hyperlinking.

Having hyperlinking capabilities in the DXP video widow gives users unique tools for organizing software development. Potential uses include creating hyperlinks off words on the collaborator's whiteboard, effectively making the video image a virtual page. A user can also attach a notepad to the collaborator himself (the face), so that ideas needing discussion can be noted as they are thought of; when pairs switch, face recognition software will allow the previously annotated information to be retrieved via a mouse click on the collaborator's face. Another possible use is linking programmers' images to the code they have most recently worked on, or are responsible for. Such a use would exploit the reason OvalTine was created -- to allow video streams to be properly integrated with textual and image-based hypermedia documents (i.e., web pages and databases).

OvalTine allows hyperlink annotations in both real-time streams and in stored video. The later capability means that for study of DXP itself, the video window can be captured, archived, and then marked up with hyperlinks via OvalTine. Researchers studying the collaborators will be able to form video webs from the various DXP sessions. We are sure there are other uses for hyperlinks that the programming pairs will discover during experimentation when the OvalTine-enhanced DXP environment is fully online.

## 4. OvalTine: Hypervideo with Real-time Anchor Tracking

One of the problems keeping video from being a fully first-class data component of hypermedia documents is the difficulty of treating the objects depicted in video as identifiable, linkable content. Rather, video tends to be manipulated as frames of pixels with no further subdivisions. When link markup is done on video streams, it is done manually frame-by-frame. We have been working with OvalTine, a system for tracking objects in video streams so that hypermedia link anchors can be associated with the objects in the video frames. The OvalTine tracking system can be used to do automated link markup of video streams. While our previous work with OvalTine presented object tracking in real-time streams [23], we have recently developed extended techniques for markup of stored (archived) data. Our results allow hypermedia structure to be generated and added to large digital libraries of video data.

Every current popular method for adding active regions to video requires manual selection of video objects, on a frame-by-frame basis. No research efforts in automation have yet made it into common practice in a widely used system such as those from Apple (QuickTime, [24]). By contrast, an automatable object tracking system is much more desirable, both for real-time applications, and for the automated addition of hyperlinks to the vast amount of archived video currently in existence.

A good overview of the issues and technologies in current hypervideo systems can be found in class notes at Texas A&M [25]. In the terms defined in this taxonomy, we are working on a system for automatically specifying mostly spatio-temporal links in hypervideo. Hypercafe [26] is often cited in hypervideo discussions; however, it is a presentation system mostly and does not support the dynamic and automatic link anchor creation we are exploring in OvalTine. The Multimedia Systems Lab at IISc India is doing work on object tracking in MPEG streams [27]. This project seems similar in scope and goals to OvalTine. They are tracking object in an MPEG stream, where as OvalTine is architected to be modular and extensible to define tracking and linking concepts at an abstract level, and to be applicable to different image and video formats with minimal extensions. Most other systems, however, that apply to hypervideo involve manual anchor creation when authoring hypervideos.

### *Two tracking modes*

We have previously reported on the OvalTine system [23] and discussed the various distributed system architecture issues involved in storing and serving video hyperlinks in a client/server implementation. This section presents our continuing work with the system, demonstrating the use of the basic real-time image tracking algorithms for use in automated markup of stored video data. In the terms of the taxonomy we outlined in [23] this is a *Server/Archived* scenario, and we have chosen to implement a *Manual* object selection scheme.

In our initial OvalTine implementation, we demonstrated real-time tracking of faces in live video streams, such as the one from the collaborator camera in the DXP environment (an example of real-time videoconferencing). This tracking allows hypermedia link anchors to be associated with objects in the video window, creating a first-class hypermedia capability for video data. The face in the video frame becomes a live link, a selectable target for the user to click on to trigger some action. As the face moves around the video screen, the live target area moves with it, providing the illusion that the face itself is the hyperlink anchor. The target is optionally made visible by means of a simple highlight oval that moves with the face. Clicks within the oval count as a click on the face, and the enveloping hypermedia layer follows the link associated with the face. The links targets are URLs entered when the object is first selected as a link anchor (at the initiation of tracking); selecting one triggers the display of a web browser window with the proper URL loaded.

We have developed a second, equally important use for the OvalTine technology – automating the addition of link markup to stored (non-real-time) video data. There is increasing interest in video data being incorporated in hypermedia structures (which we will hereafter refer to as *hypervideo data*). Digital libraries are growing in popularity and scope, and video is an important component of such archives. All major news services have vast video archives, valuable “footage” that would be of use in education, historical research, even entertainment. As noted earlier, the current best practices for link markup in video require completely, or considerably, manual markup of the video frames with the active, or hot, areas that serve as link anchors in hypervideo data. Broad access to vast caches of stored video “footage” will only be possible with automated link markup methods.

We do not specify to what a link anchor may refer, nor do we limit the type of objects that may be designated as a link anchor. The goal of our work is to explore the creation and maintenance of hyperlinks in video streams, and to automate these procedures as much as possible.

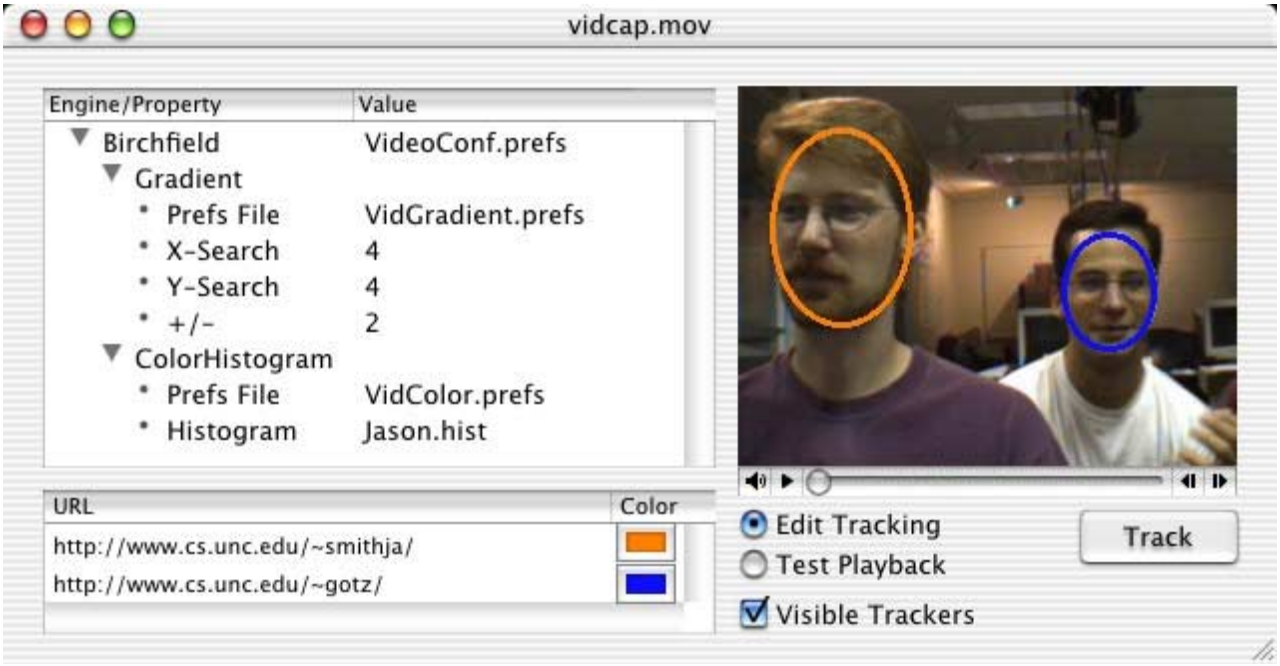


**Figure 2. OvalTine tracker interface**

***User Interface Examples: Ovals, trackers, sprites, links, and multi-links***

OvalTine is set up to allow link markup of stored video as an editing task. Figure 2 shows the video display window, the properties window, and the URLs window. The video display window presents the video data and shows the tracked objects as (optionally) outlined ovals. For real-time applications, such as video conferencing, the playback controls are inoperative. For markup of stored video, the user can do the standard *start*, *stop*, *pause*, and *slider* frame selection operations on the video stream.

The tracking properties window allows the user to select the type(s) of tracking algorithms to apply to the video frames. OvalTine's architecture has been structured with well-known design patterns [28], allowing multiple tracking algorithms to be chained together and applied in sequence. A user can even apply a different chain of trackers to each different oval if desired; OvalTine spawns a separate tracking thread for each oval. Some trackers work better than others in varying images; the selection of specific trackers to use depends on image properties such as color variability, background complexity, object motion, texture, etc.

Once a link anchor (oval) has been established in the video window, the user can associate one or more URLs with that anchor to be targets of the link(s). These URLs show up in the URL window in the lower left. Figure 2 shows one URL for each oval, and the association is made by the color of the tag. For real-time links, mouse events are trapped in the video window, and the tracker information is used directly for the current frame to determine which stored URL is to be activated.

Stored video has an extra layer atop the base video data. During markup, the video stream is played and tracked as if in real-time. The areas that the trackers identify as active link anchors in each frame are captured and stored in a Sprite layer for Apple Quicktime [24]. OvalTine then uses Quicktime to overlay the hotspot layer onto the video image during playback, and traps mouse clicks in these areas for processing through the associated URL information in OvalTine's data store.

To edit, the user selects a starting frame with the slider controls, and then designates one or more objects to be tracked. The video is started with the "track" button, and the tracker chain for each oval causes the link anchors to follow the objects as they move in the video frame. The sprite infrastructure captures the layout information needed to maintain the link anchors in association with the video data. At any point the user may pause the video, add or delete ovals, and continue with tracking.

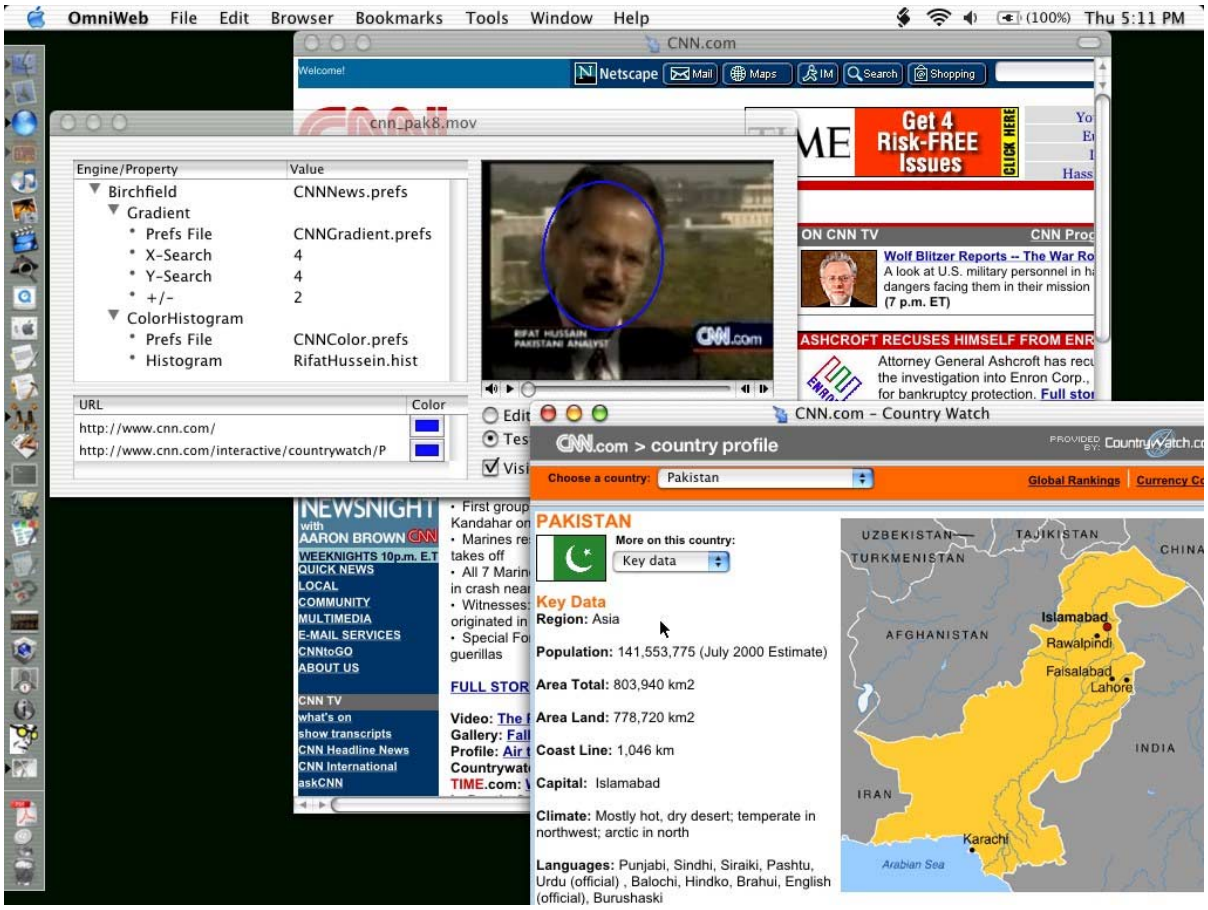


**Figure 3. OvalTine application showing marked up video stream**

URLs can be added to the tracked objects at any point, either during tracking, or during playback editing of the marked-up video data. Though not shown, ovals can be linked to any first-class Web data, including another OvalTine video clip. There is also a "lost" link palette (not shown) that collects the URLs associated with objects that are being tracked, but move out of the video window. Any URLs associated with such an object are taken out of the URL window and saved in the "lost" list. This is a convenience that makes it easier for the user to re-associate these URLs if the tracked object should reappear in the video window and need to be tracked again.

Figure 3 shows OvalTine in use while viewing a marked-up video stream. Here we see a CNN clip where the face of the reported has been annotated with 2 different links. One link is to the CNN home page (seen displayed in the background). The other link is to an article and map on Pakistan, which is the topic of the video; this page is shown in the foreground.

There are several ways to manage multiple links on each oval. One is simply to left-click the mouse on the oval in the video window. This will cause the top link in its URL list (shown in the URL window, lower left) to be followed and displayed. Another method is to right-click on the oval and a pop-up menu will present all the URLs for that oval for selection of one to follow. A third method is to directly click on the desired URL in the URL window; these URLs are only displayed in the window while their associated ovals are in the video window. In the editing mode

(as opposed to playback) the order of URLs in the URL window for each oval can be rearranged via click-and-drag operations so that new defaults can be established.

Finally, there is a mode the user can select during editing that causes *all* the associated links on an object to be followed concurrently when an oval is left-clicked. Each link target will be displayed in a separate browser window. This mode is related to the concurrent and synchronized browsing possible with the multi-head/multi-tail links found in MMM [29,30] and first studied in Trellis [31,32].

## 5. Conclusions

We have presented the DXP hardware/software environment for experimenting with distributed pair programming and distributed Extreme Programming. We are using this environment to determine if the productivity increases that XP and pair programming gives to co-located programming teams can also be demonstrated to apply to development teams that are geographically distributed. The DXP environment uses large-screen projections of the shared PC desktop and a video image of the collaborator to enhance the sense of "being there" for a non-co-located programming pair. One of these enhancements is OvalTine, a software system that allows the automated creation and tracking of hyperlink anchors in video streams, both real-time and archived. OvalTine is integrated in the video window of the DXP environment, allowing the programming pairs to create hyperlinks in the video as if it were web data.

## References


[1] Beck, K., *Extreme Programming Explained*, Addison-Wesley, 2000.

[2] Wells, J. D., "Extreme Programming: A Gentle Introduction," 2001, available on-line at http://www.extremeprogramming.org/

[3] Beck, K., and Gamma, E., " JUnit Test Infected: Programmers Love Writing Tests," *Java Report*, July 1998, Volume 3, Number 7. Available on-line at: http://JUnit.sourceforge.net/doc/testinfected/testing.htm

[4] Beck, K., and Gamma, E., "JUnit A Cook's Tour," *Java Report*, 4(5), May 1999. Available on-line at: http://JUnit.sourceforge.net/doc/cookstour/cookstour.htm

[5] Fowler, M., *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999.

[6] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," presented at eXtreme Programming and Flexible Processes in Software Engineering -- XP2000, Cagliari, Sardinia, Italy, 2000.

[7] D. C. Engelbart and W. K. English, "A Research Center for Augmenting Human Intellect," presented at AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference, San Francisco, CA,, 1968.

[8] L. Brothers, V. Sembugamoorthy, and M. Muller, "ICICLE: Groupware for code inspection," presented at Proc. of Computer Supported Collaborative Work, Los Angeles, 1990.

[9] S. Conklin and M. Begeman, "gIBIS: A hypertext tool for team design deliberation," presented at Proc. of ACM Hypertext '87, Chapel Hill, NC, 1987.

[10] R. F. P. D. Stotts, "Petri Net Based Hypertext: Document Structure with Browsing Semantics," *ACM Trans. on Information Systems (ACM)*, vol. 7, pp. 3-29, January 1989.

[11] M. Capps, B. Ladd, and D. Stotts, "Enhanced Graph Models in the Web: Multi-client, Multi-head, Multi-tail Browsing," *Computer Networks and ISDN Systems*, vol. 28, pp. 1105-1112, 1996.

[12] J. Navon, “Specification and Semi-Automated Verification of Coordination Protocols for Collaborative Software Systems Ph.D. Thesis,” in *Department of Computer Science*. Chapel Hill, NC: University of North Carolina, 2001.

[13] P. D. Stotts, R. Furuta, and C. R. Cabarrus, “Hyperdocuments as Automata: Verification of Trace-based Browsing Properties by Model Checking,” *ACM Trans. on Information Systems*, vol. 16, pp. 1-30, January 1998.

[14] D. Jones, “What is a CAVE TM?,” , pp. http://www.sv.vt.edu/future/vt-cave/whatis/, 1999.

[15] MIT, “DataWall: Seamless, full motion ultrahigh resolution projection display,” pp. http://vlw.www.media.mit.edu/groups/vlw/DataWall-overview.htm, 2000.

[16] UMN, “Welcome to the PowerWall,” , pp. http://www.lcse.umn.edu/research/powerwall/powerwall.html, 1998.

[17] H. Takemura and F. Kishino, “Cooperative Work Environment using Virtual Workspace,” presented at Proc. of CSCW '92, Toronto, 1992.

[18] H. Ishii, M. Kobayashi, and J. Grudin, “Integration of inter-personal space and shared workspace: ClearBoard design and experiments,” presented at Proc. of CSCW '92, Toronto, 1992.

[19] H. Fuchs, “The Office of the Future,” pp. http://www.cs.unc.edu/~raskar/Office/.

[20] G. Bishop, , pp. http://www.cs.unc.edu/~gb/office.htm, The Office of Real Soon Now.

[21] G. Bishop and G. Welch, “Working in the Office of 'Real Soon Now',” *IEEE Computer Graphics and Applications*, pp. 76-78, July/August 2000.

[22] R. S. Fish, R. E. Kraut, and B. L. Chalfonte, “The VideoWindow System in Informal Communications,” presented at Proc. of CSCW '90, Los Angeles, 1990.

[23] Smith, J., D. Stotts, and S.-U. Kum, "An Orthogonal Taxonomy for Hyperlink Anchor Generation in Video Streams using OvalTine," *Proc. of Hypertext 2000 (ACM)*, May, 2000, San Antonio, Texas, pp. 11-18.

[24] Apple Computer, “Introduction to Wired Movies, Sprites, and the Sprite Toolbox”, http://developer.apple.com/techpubs/ quicktime/qtdevdocs/REF/refWiredIntro.htm

[25] Francisco-Revilla, L., “A Picture of Hypervideo Today”, http://www.csdl.tamu.edu/~l0f0954/academic/cpsc610/p-1.htm, 1998.

[26] Sawhney, N., D. Balcom, and I. Smith, “HyperCafe: Narrative and Aesthetic Properties of Hypervideo”, Hypertext ’96 Proceedings, ACM, Washington, D.C., 1996, pp. 1-10.

[27] Multimedia Systems Lab at IISc, “Object tracking and hypervideo”, http://serc204a.serc.iisc.ernet.in/research/track.htm

[28] Gamma, Helm, Johnson, and Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison-Wesley, 1995.

[29] Capps, M., B. Ladd, D. Stotts, "Enhanced Graph Models in the Web: Multi-client, Multi-head, Multi-tail Browsing," Computer Networks and ISDN Systems, vol. 28 (Proc. of the 5th WWW Conf., May 6-10, 1996, Paris), pp. 1105-1112.

[30] Ladd, B., M. Capps, D. Stotts, and R. Furuta, "Multi-head/Multi-tail Mosaic: Adding Parallel Automata Semantics to the Web," *World Wide Web Journal*, O'Reilly and Associates Inc., vol. 1 (*Proc. of the 4th International WWW Conference*, Boston, December 11-14, 1995), pp. 433-440.

[31] Stotts, P.D., and R. Furuta, "Petri Net Based Hypertext: Document Structure with Browsing Semantics," *ACM Trans. on Information Systems* (ACM), vol. 7, no. 1, January 1989, pp. 3-29.

[32] Furuta, R., and P. D. Stotts, "Programmable Browsing Semantics in Trellis," *Proc. of Hypertext '89 (ACM)*, Pittsburgh, November 1989, pp. 27-42.