

A Video-enhanced Environment for Distributed Extreme Programming

David Stotts

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, dual PC setup, and an enhanced projected video display. dXP supports the development of software by pairs of programmers that are non-co-located. One projector displays a shared PC desktop, and another projector displays a life-sized image of each collaborator to the other. Developers can add hyperlinks to the collaborator camera stream, integrating it with Web pages or other video streams. A digitized whiteboard serves as the projection surface for the video stream, opening several possibilities for support of pair design work. 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). In full form, this paper will be an experience report about experiments in using this environment for dXP software development.

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 [27,28]. 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 that show 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. Results from our first experiments have appeared recently [23,24]. These experiments have led us to develop the more video-enhanced environment described 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 and the experiments we are running.

2. Background: Collaborative systems and distributed workgroups

This section will be expanded in a full paper. 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.

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.

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. The DataWall [15] at MIT and the PowerWall [16] at the University of Minnesota are similar large display projects. Virtual WorkSpace [17] was intended as an environment to enable distributed collaboration over a network. 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. 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.

“Office of Real Soon Now” and VideoWindow

Our dXP environment is 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 of 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 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. *Video, when made large, was an effective and convincing communication tool.*



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.

The Clearboard project [18] showed that *increased eye contact and a sense of presence of the remote collaborator is important to providing effective work performance*. We believe a far more effective collaborative programming 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:

- PC workstation (2)
- High-resolution video projector2 (2)
- Firewire camera + PCI video capture card (1)

- Whiteboard digitizing system (1)
- wiring, cable, microphones, screen boards, etc.

The cost for a single office is about \$10,000. 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 over 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.

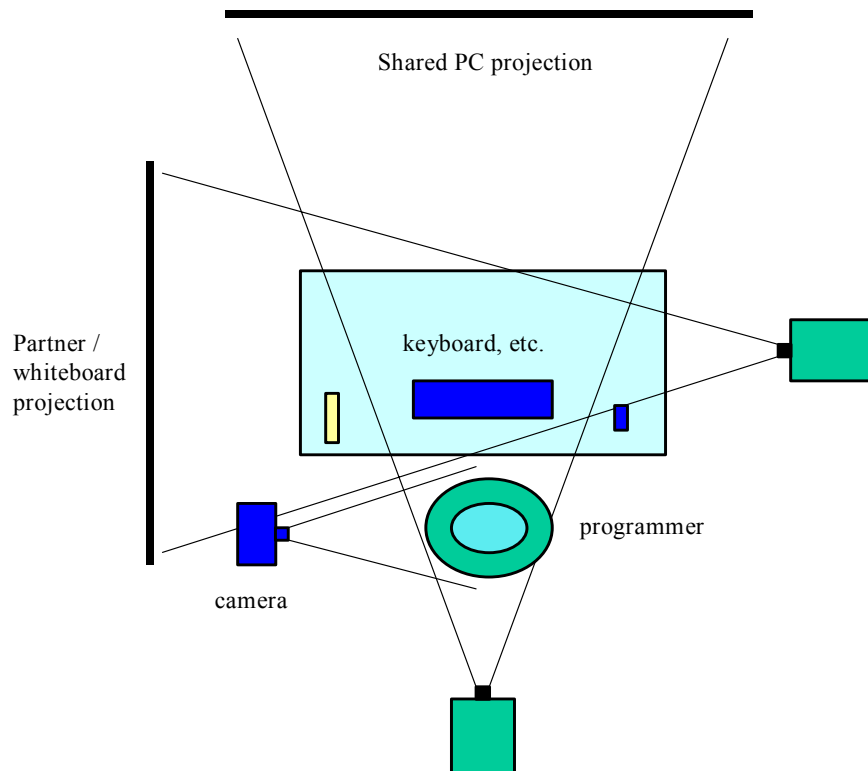


Figure 2: dXP workstation schematic layout

Figure 1 shows half the setup at UNC. Visible are the two projectors pointed at right angles to each other. Figure 2 shows a schematic of the office layout. 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 programming partner when working co-located.

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 in line with the projection wall, the remote collaborator has the impression that the communicator is looking at him or her. Each also sees 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.

Having large video display means that the programming pair can use it as a means of sharing drawings, diagrams, pictures, etc. However, our dXP environment provides another means for this joint design support as well. A

digitized whiteboard is used as the projections surface for the video image of the collaborator. When needed, the pair can switch from video image to whiteboard and use the second projector as a means of doing joint design work: drawings, notes, brainstorming. The ink drawn virtually on the board is captured and digitized and sent to the other projector to be displayed on his board. In this way, when done drawing, the board is clean and ready for re-projection of the video image.

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).

We intend to build shared artifact tools if 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 are with a single virtual PC obtained via NetMeeting, on traditional PCs. NetMeeting provides not only a shared desktop, but audio communications, text messages, and file exchange 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 [25,26]. 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). Figure 3 shows the head tracking window of the *OvalTine* system.

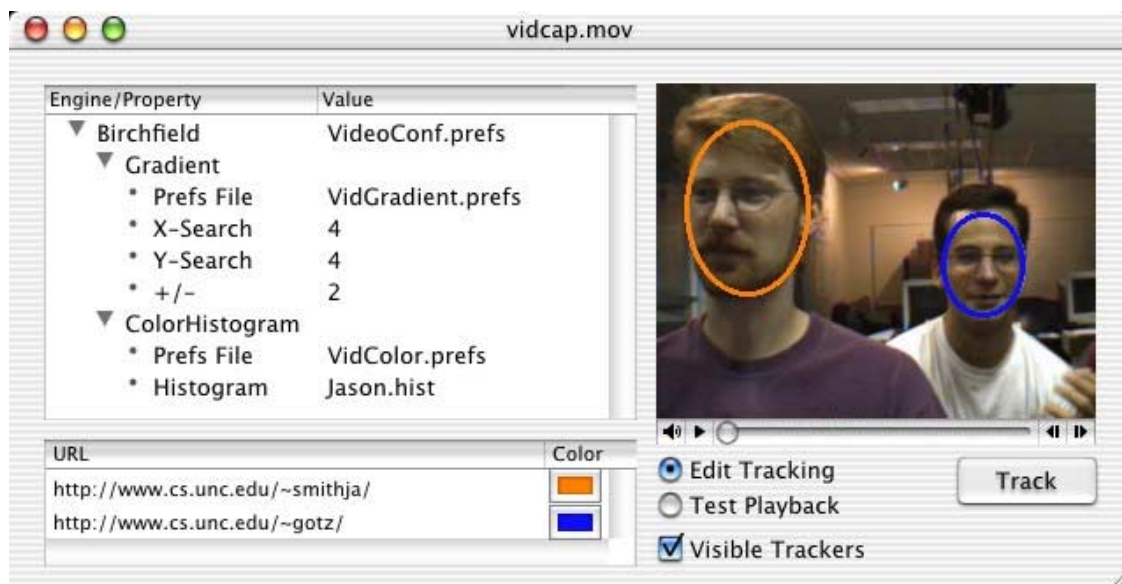


Figure 3: OvalTine face tracking window

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. Current Experiments

We are currently in the middle of several experiments using our dXP environment, and comparing it to other technical setups for dXP. A full paper will report the findings. We are developing several software systems with pairs using:

1. This video-enhanced dXP environment
2. A similar environment without the life-size video display / whiteboard
3. a dXP tool being developed at
4. a plug-in for the Eclipse IDE

Environment 2 is the same one we used to do the experiments reported in [23,24]. We found that the programming teams we very effective in using it; we now are trying to determine if the video enhancements provide additional benefits. Such benefits could include increased productivity, but they may also come in the area of enhanced enjoyment of the programmers in doing the collaborative tasks. Environment (3) is a modification of the open-source, Linux tool VNC that allows each programmer in a pair to have a cursor on the shared screen, unlike NetMeeting. This allows increased ability to remote gesture, highlight, even do separate tasks for a time.

Acknowledgements

This work was supported by NSF grant 9732577 and by EPA grant R82-795901-3 to the University of North Carolina at Chapel Hill, as well as by equipment from IBM.

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," *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," *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," *Proc. of ACM Conf. on Computer Supported Cooperative Work*, Los Angeles, 1990, pp. 169-181.
- [9] S. Conklin and M. Begeman, "gIBIS: A hypertext tool for team design deliberation," *Proc. of ACM Hypertext '87*, Chapel Hill, NC, 1987, pp. 247-251.
- [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.
- [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," *Proc. of ACM Conf. on Computer Supported Cooperative Work*, Toronto, 1992, pp. 226-232.
- [18] H. Ishii, M. Kobayashi, and J. Grudin, "Integration of inter-personal space and shared workspace: ClearBoard design and experiments," *Proc. of ACM Conf. on Computer Supported Cooperative Work*, Toronto, 1992, pp. 33-42.
- [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," *Proc. of ACM Conf. on Computer Supported Cooperative Work*, Los Angeles, 1990, pp. 1-11.
- [23] P. Baheti, L. Williams, E. Gehringer, and D. Stotts, "Exploring the Efficacy of Distributed Pair Programming," XP Universe 2002, Chicago, August 4-7, 2002; *Lecture Notes in Computer Science 2418* (Springer), pp. 208-220.
- [24] P. Baheti, L. Williams, E. Gehringer, D. Stotts, "Exploring Pair Programming in Distributed Object-Oriented Team Projects," Educator's Workshop, OOPSLA 2002, Seattle, Nov. 4-8, 2002, accepted to appear.
- [25] 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, TX, pp. 11-18.
- [26] Stotts, D. and Smith, J., "Semi-Automated Hyperlink Markup for Archived Video," *Proc. of Hypertext 2002 (ACM)*, College Park, MD, May 2002, pp. 105-106.
- [27] Olson, G. and J. Olson, "Distance Matters," *Human Computer Interaction*, vol. 15, 2001, pp. 139-179.
- [28] Teasley, S., L. Covi, M. Krishnan, and J. Olson, "Rapid Software Development through Team Collocation," *IEEE Trans. on Software Engineering*, vol. 28(7), July 2002, pp. 671-683.