Edward A. Fox
Guest Editor

Intelligent Interactive Video Simulation of a Code Inspection

The need for technological solutions to learning, in *the software engineering field is increasing. The Advanced Learning Technologies Project (ALT) has developed a highly interactive, high-fidelity simulation of group process communication. The first course demonstrating these techniques is on the formal technical review* known *as code inspection.*

Scott M. Stevens

The Advanced Learning Technologies Project (ALT) at the Software Engineering Institute is applying advanced hardware and software technologies to software engineering education. The content domain of the project is the formal technical review of Ada code. The specific type of review methodology taught is inspection. Code inspection is a formal review process that has proven to be an effective technique for defect identification. Appropriately applied, code inspection is also a management tool that provides visibility into the software development process. The purpose of the project is to demonstrate and promote advanced learning technologies that are applied to software engineering education.

The project has four broad, overlapping goals. First, it demonstrates the applicability of technologically advanced educational media to software engineering problems. These media include Digital Video Interactive (DVI), hypertext, hypermedia, and intelligent tutoring systems. Second, it transitions modern software engineering methods into practice by providing a technology-based course. Third, it is taking the lead in raising the awareness of the academic community, the government, and industrial training communities on technology-intensive software engineering education. Fourth, it investigates methods and techniques to reduce the cost, time, and effort necessary to develop a technology based software engineering education.

The National Science Board (NSB) recognizes the need for improved technology based solutions to engineering education [18]. In calling for the application of new technologies to engineering instruction the NSB states:

> The mechanisms and modes of delivery of instruction have taken on significance nearly as great as the content, with the advent of the new technologies, especially the computer. Ways must be sought to exploit the power of these technologies in the learning process, in the interests of increased efficiency and effectiveness of learning and lower overall costs.

The need for technological solutions to learning is especially important in the software engineering discipline. Software engineering has been a difficult area for educators because it continues to be human-intensive, especially in areas such as project management and technical review. Most of the instruction necessary to learn skills in these areas depends on group-paced learning and on instructors highly skilled in designing and delivering group-process oriented instruction. As a result, the quality of software engineering education often suffers; instructors are primarily technical people, skilled in and oriented toward software engineering *technical* issues, but not experienced in instructional design or group leadership.

Developing the teaching expertise necessary for these courses is difficult. IBM uses software developers and project managers, with an average of ten years experience as educators, for a two week course on software engineering. Before they are allowed to teach this class, they are given nine months of full-time training in how to teach this course [19]. Obviously, most institutions do not have the resources necessary for this approach. Advanced learning technologies can capture this expertise and deliver it when and where it is needed. These technologies have proven to be fruitful, cost effective teaching tools in numerous disciplines.

The Advanced Learning Technologies Project demonstrates their efficacy in the field of software engineering through a course on and simulation of an Ada code inspection. The course is an intelligent tutoring/interactive digital video (intelligent video) system that provides self-paced education, while it also replicates the interactive and adaptive aspects of classroom-style, group-paced education.

Portions of this paper were presented at the Third Annual Ada Software Engineering Education and Training Symposium (Denver. Col.. June 14-18), 1988.

This work was sponsored by the Department of Defense.

Advantages of IV/ITS

Reasons for choosing an intelligent video approach include decreased cost and learning time and increased learning effectiveness. A typical classroom course has a modest, onetime development cost and high delivery cost. A single industrial course may be given thousands of times at hundreds of locations over an average five-year lifetime. Each time it is given an instructor presents the course, grades papers, and provides follow-up tutoring. By comparison, intelligent video has a higher development cost but much lower delivery cost. The only continuing delivery costs are management costs that are incurred to provide students access to the materials and equipment. The courseware presents the material, grades the student, and provides follow-up tutoring. Studies with all of these factors considered have shown traditional interactive video courses to be three to five times less expensive than conventional classroom training [9, 21].

Studies by the armed services have shown that students learn material more fully and in less time with interactive video, even when compared and contrasted with computer-based training. In one army study [10], three groups of students took a course on missile electronic troubleshooting. One group took a standard lecture course, one a computer-based training course, and one an interactive video course. After the courses the students were given actual system faults to troubleshoot. The lecture group solved 25 percent of the faults. Both the computer-based training group and the interactive video group found 100 percent of the faults. While the classroom and computer-based training groups took about the same amount of time to find the faults, the interactive video group found the faults in one-half the time of the other two groups.

In another study, an intelligent tutoring system for electronic troubleshooting gave technicians with 20 hours on the system the equivalent of four years of on the job training [13]. Studies in other domains have consistently found similar results [3, 8, 10, 20].

Transitioning the software engineering content into practice is a fundamental goal of the ALT Project. None of intelligent video's advantages would be helpful, however, if the learning technology was not accepted. Fortunately, there is a growing installed base of interactive video systems in colleges, industry, and the armed services. Of all U.S. organizations with 50 or more employees, 15.5 percent used interactive video to deliver job-related training in 1986, up from 11.6 percent in 1985. Thirty-six percent of all organizations with over 10,000 employees use interactive video [24]. With the lower cost projected for DVI systems in 1990 and DVI's significantly greater capabilities compared to videodisc, an even greater acceptance of DVI can be expected.

Interactive video technology has matured to the point where it is clearly a viable medium for the delivery of education and training. Videodiscs have been developed in project management, written and oral communications, physics, chemistry, mathematics, electronic troubleshooting skills, and medical diagnoses [23]. Videodiscs have also been developed for training in operating systems (UNIX©) and computer languages (C). One company, EmTech, has just begun developing analog interactive videodiscs for software engineering education. At present no other interactive video, much less DVI projects, deal with computer science or software engineering. Evidently the immaturity of these disciplines is the main reason for this as the medium is being used very successfully in other equally abstract areas.

Content: Inspections

The content area that was chosen is one of formal technical reviews, in this case, inspection. Inspections were developed in the 1970s by Michael Fagan at IBM [5] and have been used extensively at both IBM and AT&T. Inspections can be used by a large number of people on a software project and are applicable during the development, testing, and maintenance phases of a software project.

> Software inspections or, more correctly, software development workproduct inspections are meetings where development workproducts-such as design specifications, test plans, and code-are "read." That is, the workproduct is examined meticulously and systematically by its author and his or her peers. The examination tries to find defects, mismatches between the workproduct and the specification or between the workproduct and the standards. The sharp focus on detecting defects is the key to the effectiveness of in-process inspections ... Checklists of common defects for the type of workproduct being inspected guide and enhance the meeting's defect detection process [6].

The model of an inspection simulated by the ALT Project includes four separate roles: a moderator, a reader, a recorder, and a producer. Each role in an inspection involves specific duties, while all roles require the participant to prepare for the review beforehand and act as a reviewer of the code during the inspection. Preparation includes becoming thoroughly familiar with the code. Additional responsibilities associated with each role include:

Moderator: Typically, the moderator arranges for the inspection meeting, contacting other inspectors, setting time and place, and distributing materials. The moderator also leads the inspection, controlling pace and mediating disputes.

Reader: The reader moves the inspection by reading and summarizing the code sequentially. He or she also raises action items related to pieces of code.

Recorder: The recorder records defects found and decisions and issues raised in the inspection. The recorder provides the defect log to the moderator after the inspection.

° UNIX is a trademark of AT&T.

Producer: The producer is the author of the work product. He or she is responsible for determining that the code satisfies entry criteria, explains the product when necessary, and reworks the product after the inspection.

The teaching of code inspections invariably requires references to programming style and standards. This project is using Ada programs as the work product for inspection. Thus, the code inspection simulation permits the introduction and discussion of Ada programming style and standards. This is accomplished through access to a library of materials and discussion of style during a simulation of an inspection.

For the ALT course, a single, modestly sized piece of code was desired for the inspection product. In 1978 Glenford J. Myers of the IBM Systems Research Institute performed a controlled experiment in inspections [16]. Myers wrote a PL/I program based on an Algol program written using techniques of program-correctness proofs by Naur [17] in which six errors were discovered by Goodenough and Gerhart [7]. Myers translated the program to PL/I and introduced several more errors, bringing the total to 15.

The PL/I program was translated into Ada for the ALT system. In addition, there are a number of stylistic points of discussion that have been introduced. This brings the number of points of discussion directly related to the documents being inspected to almost 70. The program has been divided into three separate artifacts for inspection. This permits a student to use the simulation several times, using different code each time.

It is clear from years of training in inspections at organizations such as IBM and AT&T that for someone to become proficient at the inspection process they must have experience in it. For this reason group process simulations are typically used in inspection training. Traditional training in inspections will last two days. During the first day information on inspections and group process will be delivered through a lecture. The second day is devoted to the group process simulation of the inspection, where students perform an inspection and analyze the process they have just experienced. This method affords the students experience in one or at most two roles in the inspection. ALT's goal is to provide a technological approach to each aspect of inspection training, both the simple information transfer associated with the lecture format and the knowledge and skill discovery associated with the group process simulation.

ALT Intelligent Video Simulation (A Cure for the Common Code)

Our intent was to create a total learning environment for inspections. In this environment a student can receive a lecture at his or her own pace, peruse the holdings of a library, study code to be inspected with debugging and hypertext tools, or participate in an inspection.

The ALT system simulates a mythical company *Ultimex.* The user takes the role of a new software engineer at Ultimex. The top level user interface is surrogate travel through Ultimex's office building. The user has access to six rooms: the Auditorium, the Training Room, the User's Office, the Library, the Coffee Room, and the Conference Room. These rooms provide the user with information on and experiences in inspections, as well as tools supporting the inspection process.

In the Auditorium, a person using the system the first time gets general information about inspections and their value. Here, the user also receives information on Ultimex and the course they are about to take. NASA Apollo and Space shuttle films and short dramatizations are used to illustrate the importance of software quality and the potentially life threatening aspects of software errors. This is principally a motivational section, the "hook" that interests the students in the rest of the course, and takes approximately ten minutes.

After the Auditorium, the student will be led into the Training Room where they will learn how to navigate through the simulated world, use both the window interface, and the natural language interface. Here they will also have access to instruction on inspections and group process. The student is given information such as the differences between walkthroughs, audits, and formal inspections, as well as descriptions of the inspection process, types and checklists. The material in the Training Room is presented through motion video, graphics with narration, and text with graphics. Depending on their needs, users may spend from fifteen minutes to one and one half hours in the Training Room.

From the Training Room users may go into the Library where they have easy access to a large amount of information. This is a hypermedia library containing text, graphics, video, and audio materials. Users will be able to view what looks to them like videotapes. These are a series of vignettes illustrating what goes into a good inspection, what happens to make inspections go wrong, as well as tapes on group process. The Library *card catalogue* provides access to a large database of print materials. Included are items such as checklists, NASA Ada style guidelines, approximately 1000 slides from traditional courses on inspections developed by the jet Propulsion Laboratory and the Software Engineering Institute, and 12 of the seminal papers on inspections.

Some of the material from the Training Room will be accessible in the Library as well. Also, while the user is in the Training Room suggestions will be made as to material for further study. This material will then be highlighted in the Library. Access to both the Library and the Training Room will be available to the student at all times.

An important aspect of inspections is preparation. In their Office the user has a number of tools to help study the code in preparation for the inspection. To help facilitate this process a windowing environment and toolset was created. Off the shelf environments were investigated. None, however, provided the performance nor integration with the chosen expert system language and DVI library that was required. The toolset, a source level debugger and hypertext system, was designed to give the user assistance in the preparation task, permit the system to monitor student work, and to make the process paperless.

Source level debuggers are quite common and provide an excellent instrument for investigating the operation of another's code. The source level debugger included in the ALT system allows students to trace a program line by line, set break points, run the program stopping just at those break points, and examine how selected variables change during execution.

The hypertext tool links specific lines code to the specification that generated it and specifications to the code that was generated from them. This way the user may more easily trace code to the requirements that created that code and vice versa. In addition, error reporting forms are provided for the user. These include space for information on the specific error as well as broader comments, and are linked automatically to the particular line of code under study. The code, specifications, and error report forms complete with the hypertext links are available to the user in the Conference Room during the simulation of the inspection.

While we recognized the need for this type of tool independently, work by Soloway et al. has shown the benefits of creating documentation to compensate for delocalized plans [22]. The hypertext system, although not intended to be a production level tool, points to the type of tools we believe would be useful for dealing with the problem of delocalized plans in software production and inspection.

While they work in their Office, users are asked by the secretary which role they would like to take when they actually perform the inspection. As noted earlier, experience in performing an inspection is crucial to acquiring the skills needed for success in the inspection process. The ALT system creates a high fidelity simulation of an inspection, giving, we believe, the requisite experiences to become proficient in the inspection process.

Since each role is unique, the user is given the choice of taking the role of moderator, reader, or recorder. The role of the producer of the code is always simulated by the system; there is no way for the students to use their own code for the inspection. If such complete automatic analysis of code was possible, inspections themselves would be unnecessary. Since code developed by the user cannot be inspected, the user can never have the pride of ownership that causes much of the conflict associated with software reviews. While technically we could have allowed the student to act as the producer of the code in the inspection, we prohibit them from assuming this role.

Depending on which role has been taken, the student may schedule the inspection (as moderator) or inform the group when they are ready to perform the inspection (if reader or recorder). The student then moves to the Conference Room to perform the inspection. Here the student will see the other three members of the inspection team enter the room, sit down and begin discussion. This begins the simulation of an inspection and is the central piece of the ALT system. Several features of Digital Video Interactive are critical to the ALT system's ability to create this simulation.

DVI permits up to seventy-two minutes of full screen, full motion video to be stored on one compact disc. The simulation of the inspection requires approximately ten hours of audio and two hours of motion video with audio. We are able to increase the apparent storage by saving fractional screen images and composing the full image from still images plus motion sequences.

In analog video, images are fixed during production and post-production. For practical purposes, the image on a videodisc or video tape cannot be altered significantly during playback. If the information of interest takes up only a fraction of the screen one full frame of analog video storage must still be allocated. Since images are stored digitally in DVI neither of these limitations is true. We are able to compose the visual image during playback. Since we wished to store the images of the actors separately we could save significant storage space by saving only the section of the images that is of interest.

The actors for the inspection simulation were shot in a blue environment as shown in Figure 1. Camera placement was held rigid during two weeks of production through a worm drive gear head. The camera itself, a new generation CCD model, was extremely stable electronically. Lighting was meticulously checked throughout the production. Rigorous attention to such detail allows for the precise registration necessary to compose the image during playback.

After production the table was repainted and a background placed in the scene. The scene with table and background alone was then taped again as shown in Figure 2 and saved as a still image which the actors are keyed onto during playback as we see in Figure 3. The space for each actor is approximately one sixth of the full screen. During playback only the actor speaking moves. Stills of the other actors may be changed periodically. The viewer, looking at the speaker, gets the impression of normal motion video at one sixth the storage requirement.

Even with this technique, not enough full motion video could be stored on a single CD-ROM to hold all of the data necessary for the ALT system to simulate a wide ranging discussion between four participants-a discussion where the user may ask questions, respond to questions, or change the topic of conversation at any point. In terms of information transfer to the user, the critical data is audio. An analysis was made to determine the most probable discourse. The data necessary for this interaction, along with other high interest monolouges were produced and stored as motion video with audio. The remainder was recorded and stored as audio only. This audio, approximately ten hours of AM quality, is presented in conjunction with still images. These

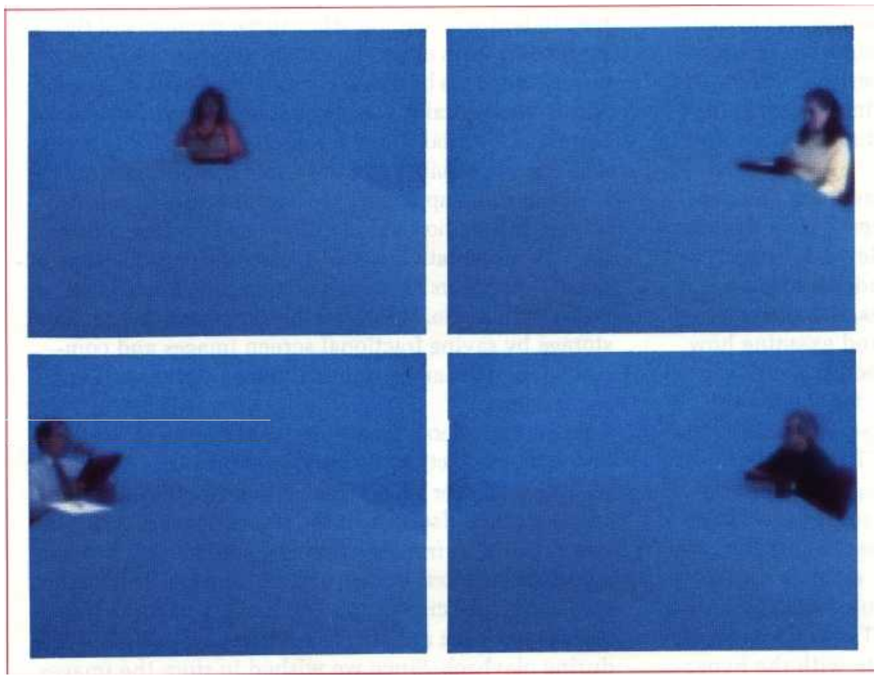

FIGURE 1. Blue Screen Images of Simulation Participants

stills (over 2,500 are available to the system) are continually changed during the audio presentation to give an animated effect, drawing the viewer's attention to the actor speaking. With these techniques the user's impression is one of viewing and taking part in a continuous discussion.

The producer of the code is part of the inspection team no matter what role the user takes and is always in the center of the image. The image is point of view. Depending on which role the user takes, two of the other three team members will be keyed into the image to complete the scene. In order to allow any combination of team members, one actor will have to be presented screen right for one combination and screen left for another. Rather than producing and storing one actor's scenes twice, once for left presentation and once for right, only one series is stored and if necessary mirrored at playback, saving approximately twenty percent of the total video stored as shown in Figure 4.

Each student will spend from one to two hours in the inspection, receiving a natural feedback on his or her performance from the intelligent tutoring system driving the simulation. If the interactions are inappropriate, simulated members of the inspection team will respond accordingly. Still, more direct feedback will be desired by many. On conclusion of the inspection the user will be given performance feedback in the Coffee Room. During the inspection a global history of the simulation is kept. The tutoring system uses this history to analyze the student's performance. From this analysis the system's video trainer will explain directly both the positive and the negative aspects of the user's performance in the inspection.

Prior to the introduction of DVI, hardware capabilities were unavailable to produce a simulation of this type. But even with the impressive functionality of DVI, it is the software that is critical to the creation of a high fidelity, pedagogically sound simulation.

Expert System Driven Simulation

To simulate the interactions between participants in the inspection, to provide the ability for the user to take any role, and to provide tutoring in the inspection process, a rule-based expert system was developed to model the participants. This expert system is used to define the "personalities," to control the dialogue between the simulated members of the inspection team, personae, and the user, to interpret the response from the user, and to intelligently control the visual presentation.

The expert system is composed of over one hundred rules in Ops/83. This was chosen because it is capable of forward chaining, it is has acceptable performance on an IBM AT, and it interfaces well with C (the language of necessity for DVI development). The rule base was developed from analyses of taped inspections, analyses of normal conversations, and considered the work of Bales [1, 2]. The rule base is used to make decisions in areas such as who should speak, the tone they should take, the content (context space search), and who should be addressed.

To model different personalities, several emotional attributes are defined for each individual simulated by the system. Attributes include defensiveness, aggressiveness, talkativeness, and the tendency to make irrelevant, humorous comments.

State variables keep track of the history of the conversation. These variables capture information on the current topic of discussion, the length of focus on this topic, how the discussion is resolved, the opinion on the topic for each participant, the current speaker, the person focused on in the current comment, and the

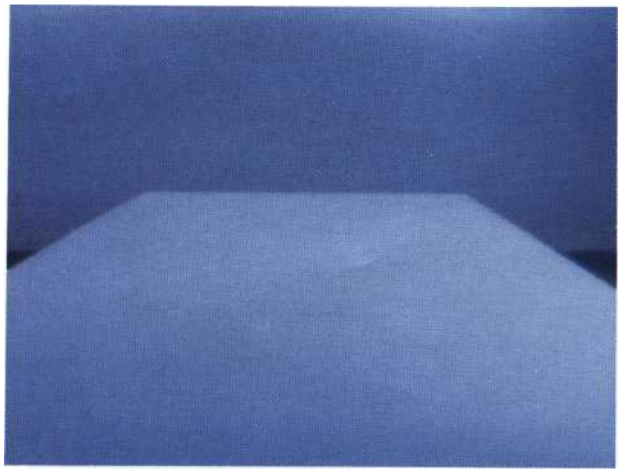

FIGURE 2. Still Background Image

person addressed in the current comment.

Rules control the conversation and model the personalities of the personae. Our paradigm is somewhat different from typical intelligent tutors. The student receives a natural feedback on his or her performance. After the inspection simulation, more direct feedback is available through an outlet interview with the trainer in the Coffee Room, prior to exiting from the system.

Development of the rule base proceeded from an English version of the rule to pseudocode and finally to its Ops/83 implementation. For example, one rule deals with a participant who has been dominating the conversation. This participant may be the user or one of the simulated personae. Another simulated persona has tired of this and wants to say something about it. Who brings this up, what they say, and how they say it depends on a number of factors including personality variables set for the personae, the history of the conversation, what this persona's role is, and what role the user has taken. The English version of the rule is shown in Figure 5, the pseudocode version in Figure 6, and the Ops/83 implementation in Figure 7.

An example of items in the rule base's working memory are personal attributes as shown in Figure 8. This includes information such as when this person last spoke, the last comment made that was relevant to the conversation at hand, and whether this person has been dominating the conversation. Also included here is the name of the person. The system cannot know a priori which role the user is going to take or who a persona is going to address. When personae are addressing one another they are actually addressing a role, i.e., moderator, recorder, producer, or reader. The system knows if it is about to address a simulated participant

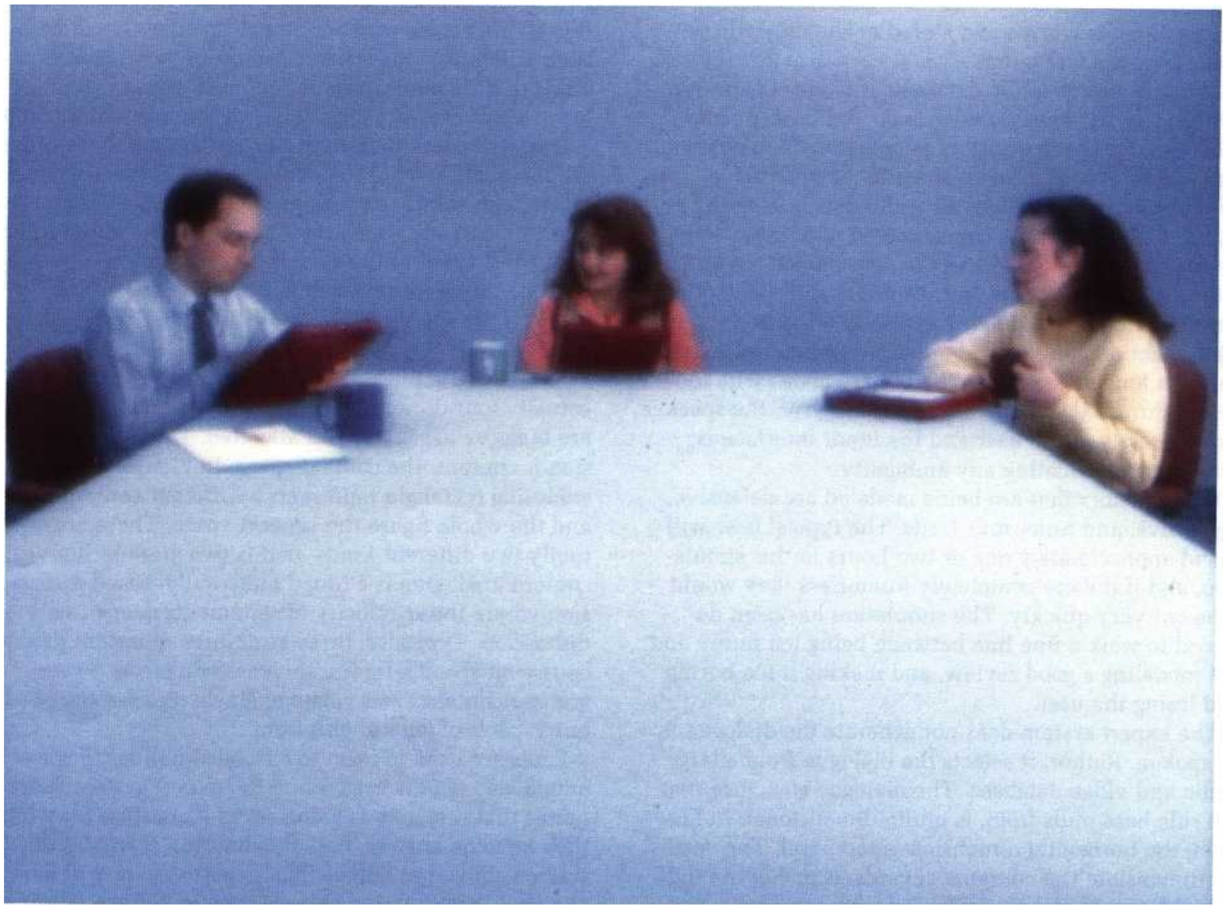

FIGURE 3. Final Composed Image

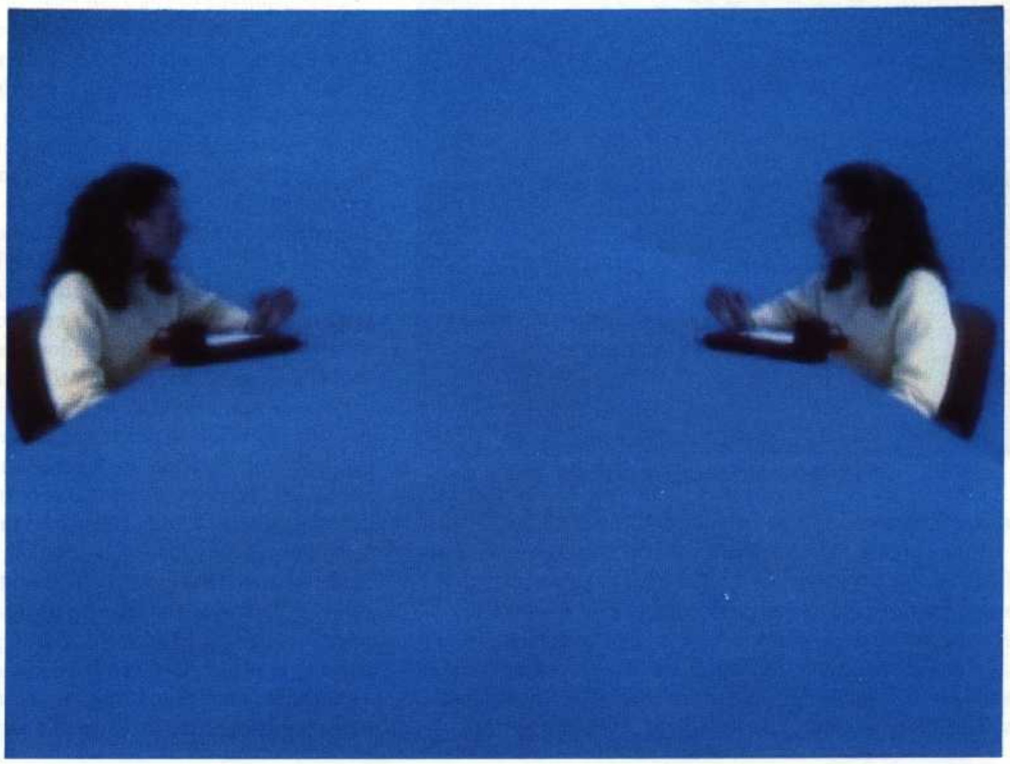

FIGURE 4. Mirrored Image of Actress for Simulation

whose name is known and stored or the user whose name may not be known.

There is a large audio database of names taken from a list of the 120 most popular male and female names [4]. Thus, for the majority of people the video characters will be able to address them by name. This is possible since names, like all audio data, are stored as digital files that can be concatenated with other sentences as needed. Sentences are constructed such that if the user's name is not in the audio database no direct form of address is needed. For example the sentence "Bill, what do you think about this?" stands without the name as long as there is no ambiguity about who is being addressed. When addressing the user, the speaker looks directly at the user and the input interface is presented, eliminating any ambiguity.

Other factors that are being modeled are defensive, aggressive, and humorous traits. The typical user will spend approximately one or two hours in this simulation, and if it were completely humorless they would tune out very quickly. The simulation has been de signed to walk a fine line between being too funny and not modeling a good review, and making it too boring and losing the user.

The expert system does not generate the dialogue to be spoken. Rather, it selects the dialogue from a large audio and video database. The dialogue structure that the rule base pulls from, is multi-dimensional. In Figure 9 the horizontal dimension is temporal. The vertical dimension, the columns of cards, is one of the individual being modeled. Each card represents what the persona may have to say at any point in the conversation. Within each card we model the affective dimension, where the upper areas represent very passive kinds of statements, middle are more neutral, and the lower areas are very aggressive or defensive statements.

During the simulation, the expert system determines the temporal level of the conversation. It then determines who wants to speak and their affective state. The dialogue data are then searched to find the appropriate audio and video. If the search is successful, the persona has something to say on this topic and is permitted to speak. If the search fails, the system must still behave reasonably.

This is accomplished since the dialogue structure is actually four dimensional. The first three dimensions are temporal, speaker, and affective. The fourth dimension is content, the context space. In Figure 10 each enclosing rectangle represents a different content area and the whole figure the context space. There are actually two different kinds, that is two granularities, of content area. One is a broad temporally-based discussion where the specificity of statements deepen with discussion. Typically, these structures represent discourse on specific topics, such as code errors. Finer grained content areas relate to single phrases acting as quick pools of data to pull from.

Pools are used to react to special situations. For example, one pool is used when the expert system determines that someone is being asked a question for which they have no answer, (i.e., the dialogue search as described above has failed). The expert system will look at a pool that contains phrases such as "I have nothing to say," and pull an item from there. Since situations

One of the participants in the inspection has spoken too frequently during the last few minutes and someone (participant Y) other than this person is willing to talk.

If the user is taking the role of the moderator then usually the user will get various types of feedback such as other participants beginning to lose interest or falling asleep. Frequently, participant Y will make a comment to the user about the problem.

If the user is not the moderator (or occasionally when the user is the moderator) and if participant Y is fairly aggressive, then Y tells the person talking too much to talk less. If the user is the moderator and this problem has been brought up to the user before then participant Y will more forcefully tell the user there is a problem (i.e. someone is dominating the conversation and the user, as moderator, should do something about it). If Y is not aggressive, Y will simply suggest that others talk more. This rule adjusts the propensity to talk of the individuals in the group. Other rules interact to after the defensiveness of the person (user excluded) who is talking too much.

FIGURE 5. English Language Version of Production Rule

such as this may occur several times and at any point in the conversation, we store these comments as pools that are accessible whenever needed.

These pools also allow us to bridge between areas. Due to changes in personality factors and intervention by the user, speaker order is indeterminate. This requires individual script atoms that are independent of immediate order. Acknowledgement of the previous statement, however, is key to a realistic dialogue. This acknowledgement is handled with agreement and disagreement pool topics.

For example, at one point in the conversation the reader may be able to say "It's a nice day," the moderator "I think it's a dreary day," and the producer "I think it's a beautiful day." If the reader speaks first a pool statement may be combined with the producer's statement to create: "It's a nice day" (reader), "I agree, I think it's a beautiful day" (producer). On the other hand, if the moderator speaks first: "I think it's a dreary day" (moderator), "I don't agree, I think it's a beautiful day" (producer).

The four dimensional structure provides multiple levels of granularity for phrase generation. It thus provides great flexibility for the expert system in generat-

```
Let:
    X =Any Reviewer
    Z1 = The Set of Reviewers Excluding X
    U = The User (Student)
    M =Moderator

    Pt(X) = Level of Talkativeness for Person X
          0 < Pt(X) < 1
    Pa(X) = Level of Aggressiveness for Person X
          0<Pa(X)<1
    Pd(X) = Level of Defensiveness for Person X
          0<Pd(X)<1

    id(X) = Incremental Change in Defensiveness for Person X
          0 < id(X) < .4
    dt(X) = Decremental Change in Talkativeness for Person X
          -.4 < dt(X) < 0
    newval(trait, person(s), change/person)
          Where newval(d, X, id(X)) -> Pd(X) = Pd(X) + id(X)
    All-Y-X -> For every person Q


Then the Production Rule is:

X has spoken MOUTHLIMIT of the time out of the last STATECHECK comments &
Y is in Z1 & Pt(Y) > 0.6 ->
    if U = M then 75% of the time this rule fires do this:
          Provide feedback to U on the problem, e.g. have Z1 fidget and
          look annoyingly at X, or have Z1 begin to fall asleep and
          snore a little. Possibly, have Y address U with the problem.
    else (either U 1= M or 25% of the time when U = M)
          if Pa(Y) > 0.5 ->
                if U = M & U was mildly scolded before ->
                      have Y scold M
                Y prompts X to talk less, others to talk more;
                newval(d, X, id(X))

                newval (L X, 2 - dt(X))
          else
                Y just Prompts others to talk more;
          newval(t, ALL - Y - X, it(ALL-Y-X));
```

FIGURE 6. Pseudocode Production Rule

```
rule p shut up
  {
   &T (spoketoomuch);
   &Y (person user = Ob; role <> &T.talker; talkative > 60);
   &G (global history);
   &U (person user= 1b);
   &C (conversation current = 1b);
   &H (recent history comment made = &G.num speeches);
   - (historycheck);
   10.65] -->

   -- Person &T.talker has been speaking too much. Y tells him to shut
   -- up, and may also scold the moderator in the process.

   local &waitforM: logical;

   if (&U.role = moderator)
      call scoldmoderator(&Y, &T.talker, 38, &waitforM)
                        --topic 38 = SHUT UP
   else
      &waitforM = Ob; -- No need to wait for moderator!
  if (&waitforM = Ob)
      { -- Do not wait for the moderator. Have talker be addressed
        -- by Y.
      call say(&Y, 38, &T.talker, &T.talker, &G, &C, &H);
         --topic 38=SHUT UP
      call newval(talkative,ALL, &Y.role, &T.talker, 1, up);
      if (&Y.aggressive > 50)
           { -- If Y is aggressive there is an increased
               emotional response --
           call newval(defensive, &T.talker, NONE, NONE, 1, up);
           call newval(talkative, &T.talker, NONE, NONE, 2, down);
      };
  };

};
```

FIGURE 7. Ops/83 Production Rule Implementation

- spoke_last--When did this person last speak
- last relevant--When was the last relevant comment made
- num speeches--How often has this person spoken
- name--What is the name of this person
- talkative--How talkative is this person
    - uptalk--Increment value for talkative trait
    - downtalk--Decrement value for talkative trait
- defensive--How defensive is this person
    - updef--Increment value for defensiveness trait
    - downtalk--Decrement value for defensiveness trait
- aggressive--How aggressive is this person
    - upagg--Increment value for aggressiveness trait
    - downagg--Decrement value for aggressiveness trait
- wisecracker--How humorous is this person
    - upwise--Increment value for humorous trait
    - downwise--Decrement value for humorous trait
- role--What role is being taken
- user--Is this the user
- prepared--Is this person prepared

FIGURE 8. Working Memory Item: Personal Attributes

ing conversations as well as responding to user input.

A unique feature is the ability to run the simulation without taking a role. In this case the system models all four individuals. They carry on a conversation among themselves, performing a complete inspection with no user intervention. Trainers and educators can set up particular scenarios. For example, an instructor could have the system model a very aggressive moderator, defensive producer, and talkative reader and then let the system play out the scenario to illustrate the consequences. As in real life, discussions will vary depending on the personalities of the participants. When the same conversation is begun with new personalities, a different discussion will ensue.

An interesting part of the expert system is one that we are calling Hitchcock, after the director. This is the visual director of the system. Now, for the first time with digital video, we can manipulate the images when we play them back. We have to do that for the ALT system.

We want the expert system to behave intelligently in presenting images, much like a director would do when creating or editing an image. Here, the system does it during playback. For instance, if the user is dominating a conversation, the system may present a slightly lower camera angle of the participants on the screen. Years of experience and many studies have shown that images such as this tend to convey to viewers the impression that the viewers themselves are dominating the scene [11, 12]. Conversely, if the user is being very passive, we may wish to present a close-up from a low angle, where the people in the scene now appear more dominant. From a history of the scene Hitchcock also makes a judgement on whether the presentation of a wide shot, medium shot, or close-up is called for.

Natural Language Interface

From the beginning, one of our largest concerns was how the user should supply input. Ideally you would be able to speak into the system, and it would be capable of continuous speaker independent speech recognition, including the comprehension of affective effects of tone. Obviously, that technology is not here today. Even free-form natural language text input on a system the size of an IBM AT is not feasible with the reliability we required.

To overcome these problems we developed a natural language interface based, in part, on the work of Harry Tennant [15]. Tennant has done work in a menu based natural language system: Natural Access. In Natural Access sentences are constructed from sentence fragments. While allowing the user great freedom and power in sentence construction, this method restricts the domain and syntax to a level manageable by our target machine.

Figure 11 illustrates this interface. With it you construct sentences, address comments to individuals, and add tone and affect to your statement. Affect and tone are handled by the sentences created and by selection of tone icons. Tone ranges from passive (by selecting the cool blue face) up through aggressive (by selecting the red flame face).

The construction of the interface shell was straightforward. The most significant problem was the analysis of the sentences to be constructed. For an interface such as Natural Access the domain is typically an application such as a spreadsheet, where analysis is relatively simple. The discourse in an inspection is quite free in its form and broad in its domain, however.

Elliot Soloway had performed a significant amount of inspection analysis during a research project [14]. His research team found that there were specific topic areas that people spent most of their time on, during an inspection, such as looking at clarity and correctness. We recognized that this analysis might be extended to look for particular sentence types. After discussions with Soloway it was determined that a joint project would allow for the extension and tailoring of this research to a form that would provide the foundation for the ALT language interface sub-system.

To complete this work, Robin Lampert was brought to the SEI from Soloway's project. Lampert worked with project members to extend and refine her earlier research. This part of our work found seven main categories of discourse during an inspection which were labeled: clarity, correctness, simulation, consistency, design rationale /choice, procedural, and general discussion. Within each category a sufficiently small number of sentence types were identified to make the development of a menu based natural language interface feasible as shown in Figure 12.

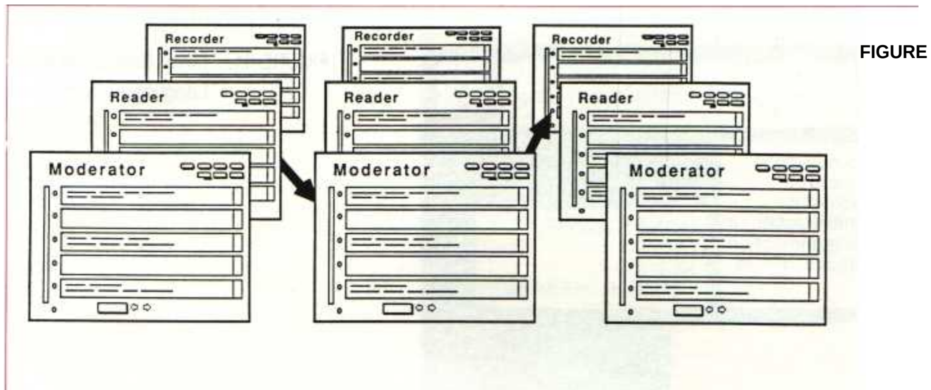

FIGURE 9. **Single Topic Dialogue Structure**

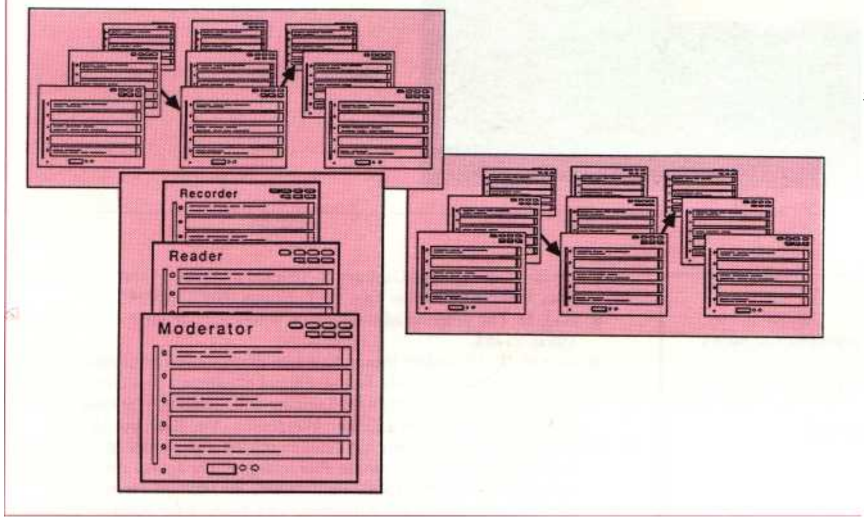

FIGURE 10. **Four Dimensional Dialogue Structure: Content (rectangular groups), Speaker (individual cards), Affective (areas within cards), and Temporal (horizontal extension within a content structure)**

For example, the user may wish to say:

"This line when bufpos = maxpos causes an error." "This line" refers to a line of code and would be highlighted by the user.

This sentence is an example of the basic sentence structure:

<object> when <code-action> causes an error.

To create this sentence a user selects the <object> from the OBJECT list as in Figure 11. The portion called <code-action> is a sub-section of SIMULATION with construction of specific identifiers by keyboard input allowed. The line referred to by "This Line" might be selected by pointing with the mouse and would remain highlighted during the discussion.

Early experience with this interface suggests that users adapt to it in a short time. More importantly, they are able to quickly input the content if not exactly the form of sentence they wish.

Conclusions

The ALT system has several modes of operation. Software engineers and students use all aspects of the system. Managers are given a presentation designed to highlight management uses of inspections and typically would not perform the inspection simulation. Educators and trainers can use the resources in the Library along with a projection monitor as a powerful presentation system for lectures on inspections. The power of interactive technologies is, when designed properly, their ability to adapt to the user.

Using expert system technology to model personalities and control the simulation creates an exceptionally interactive simulation. This allows, for the first time, the creation of high fidelity simulations of interactions between people. Rather than on-the-job education in areas such as reviews and project management, where errors are costly, the ALT system demonstrates that it now becomes possible to give students and professionals technology based experience, where errors are free of dire consequences.

The ALT Project has just begun formal evaluation. The final CD-ROM disc is expected to be pressed in the fall of 1989. Early use of prototype versions indicate that users quickly adapt to the interface and find the simulation engaging. The experience in other domains suggests that benefits of the DVI, hypermedia, and intelligent tutoring learning environment of the ALT system include: inexpensive dissemination of materials, increased learning productivity, continuous assistance to the learner, practical experience in the methodology, and the ability to adapt the material to a diverse group of users. It is believed that as the capabilities of CD-

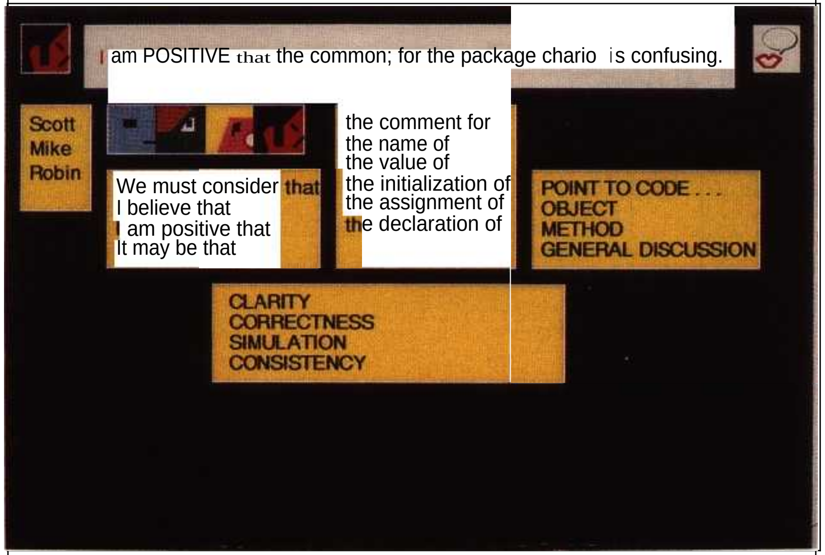
I am POSITIVE that the common; for the package chario is confusing.

FIGURE 11. Menu Based Natural Language Interface

<object> is (not) an ERROR. (? !)

<object> has (in)CORRECT <programming construct-aspect>

<object> does (not) FULFILL the REQUIREMENTS.

<object> when <code-action> causes an ERROR.

<object> is (un)NECESSARY.

<object> is in the WRONG POSITION.

<Object> SHOULD BE <(user input ?), variable-value-boundary>

FIGURE 12. Sentence Types for the Category of Correctness

ROMs and digital video become commonplace on workstations, high quality, technology-intensive education will also become common.

Acknowledgments. The Advanced Learning TECHNOLOGIES Project team includes Judy Chiswell, instructional designer and Robin Lampert, research assistant. A special note *of* thanks goes to Michael Christel, project software engineer.

REFERENCES

1. Bales, R. *Interaction Process Analysis.* Addison-Wesley, Cambridge, Mass., 1951.
2. Bales, R. *Personality and Interpersonal Behavior.* Holt, Rinehart, and Winston, New York, 1970.
3. Bernd, R. Interactive videodisc in the army. In *Proceedings of the Fifth Annual Conference on Ineractive Videodisc in Education and Training* (Aug. 24-26, Arlington, Va.). Society for Applied Learning Technology, Warrenton, Va., 1983, pp. 14-16.
4. Bonanza Books, Inc. *The Modern American Encyclopedia of Names for Your Baby.* Bonanza Books, New York, 1981.
5. Fagan, M.E. Design and code inspections to reduce errors in program development. *IBM Syst. J.* 15, 3(1976), 182-211.
6. Fowler, P. In-process inspections of workproducts at AT&T. *AT&T Tech. J.* 65, 2(Mar.-Apr. 1986), 102-111.
7. Goodenough, J., and Gerhart, S. Toward a theory of test data selection. *IEEE Trans. Softw. Eng. SE-1,* 2(June 1975), 156-173.
8. Hon, D. The promise of interactive video. *Perf. & Instr. J. 22,* 9(Oct. 1983), 21-23.
9. Ketner, W. Videodisc interactive two dimensional equipment training. In *Proceedings of the Fourth Annual Conference on Video Learning Systems Videodisc for Military Training and Simulation.* Society for Applied Learning Technology, Warrenton, Va, 1982, pp. 18-20.
10. Kimberlin, D. U.S. army air defense school distributed instructional system project evaluation. In *Proceedings of the Fourth Annual Conference on Video Learning Systems Videodisc for Military Training and Simulation* (Aug. 25-27, Arlington, Va.). Society for Applied Learning Technology, Warrenton, Va., 1982, pp. 21-23.
11. Kraft, R. The influence of camera angle on comprehension and retention of pictorial events. *Mem. & Cogn.* 15, 4(1987), 291-307.
12. Kraft, R. Mind and media: The psychological reality of cinematic principles. In *Images, Information & Interfaces: Directions for the* 1990's, D. Schultz and C.W. Moody, Eds. Human Factors Society, New York, 1988, pp. 13-36.
13. Lesgold, A. A coached practice environment for an electronics troubleshooting job. In *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Issues and Complimentatry Approaches,* J. Larking, and R. Chabay, Eds. Lawrence Erlbaum, Hillsdale, N. J. To be published.
14. Letovsky, S., Pinto, J., Lampert, R., and Soloway, E. A cognitive analysis of a code inspection. In *Empirical Studies of Programming,* G. Olson, S. Sheppard, and E. Soloway, Eds. Ablex Publishers, Norwood, N. J., 1987, pp. 231-247.
15. Miskoff, H. *Understanding Artificial Intelligence.* Texas Instruments, Dallas, 1985.
16. Myers, G. A controlled experiment in program testing and code walkthrough/inspections. *Commun. ACM 21,* 9(Sept. 1978), 760-768.
17. Naur, P. Programming by action clusters. *BIT* 9, 3(1969), 250-258.
18. NSB Task Committee On Undergraduate Science and Engineering Education. Undergraduate science, mathematics and engineering education. Tech. Rep. NSB 86-100, National Science Board, Washington, D.C., 1986.
19. Pietrasanta, A. Software engineering education in IBM. In *Issues in Software Engineering Education.* R. Fairley and P. Freeman, Eds. Springer-Verlag, New York, 1989, pp. 5-18.
20. Pietri, J., Jr. An IBM perspective. In *IEEE Videoconference Seminars via Satellite Optical Discs: An Information Revolution* (Fob. 26). IEEE and The Learning Channel, New York, 1987.
21. Reeves, T., and King, J. Development, production and programming of an interactive videodisc adult literacy program. In *Proceedings of the Eighth Annual Conference on Interactive Videodisc In Education and Training* (Aug. 20-22, Washington, D. C.). Society for Applied Learning Technology, Warrenton, Va, 1986, pp. 44-49.
22. Soloway, E., Pinto, J., Letovsky, S., Littman, D., and Lampert, R. Designing documentation to compensate for delocalized plans. *Commun. ACM 31,* 11(Nov. 1988),1259-67.

23. Stevens, S. Interactive videodisc: A background report. Tech. Rep. Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, 1987.
24. Training Magazine. *Annual Industry Report.* Market Report, Training Magazine, Minneapolis, 1986.

CR Categories and Subject Descriptors: D.2.4 [Software Engineering]: Program verification; D.2.9 [Software Engineering]: Management-*software quality assurance;* F.3.1 [Logics and Meanings of Programs]: Specifying and Verifying and Reasoning about Programs; 1.2.1 [Artificial Intelligence]: Applications and Expert Systems; 1.3.m [Computer Graphics]: Miscellaneous; 1.6.3 [Simulation and Modeling]: Applications; J.4 [Computer Applications]: Social and Behavioral Sciences; K.3.1 [Computers and Education]: Computer Uses in Education
General Terms: Human Factors
Additional Key Words and Phrases: Code inspections, digital video, digital video interactive, human computer interaction, intelligent tutoring systems, intelligent video, interactive video

ABOUT THE AUTHOR:

SCOTT M. STEVENS is a senior member of the technical staff at Carnegie Mellon University's Software Engineering Institute and the Project Leader for the Advanced Learning Technologies Project (ALT). His research interests include exploring the role of cognitive sciences. Author's Present Address: Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.