MTR 02B02B0000V00S000R00

MITRE TECHNICAL REPORT

Computational Embedded Training Strategies

MITRE-Sponsored Research Final Report

October, 2002

Brant A. Cheikes, Ph.D.
Abigail S. Gertner, Ph.D.
Laura Chung Kurland

MITRE
Center for Integrated Intelligence Systems
Bedford, Massachusetts

Table of Contents

Section 1

Introduction

This document constitutes the final report of the MITRE-Sponsored Research (MSR) project *Computational Embedded Training Strategies*, project 51MSR8AA. This report addresses requirements articulated in the MSR Handbook:

"A final report is required when a project has been completed. The report should be completed prior to the closing of the project while the project number is still valid for charging. This report should include the Project Title and Number, PI's Name, Names of Project Staff, Objective, Accomplishments, External Impact, List of Documentation, and any other pertinent information. The PI should publish the report on the MII along with a URL pointer to a demonstration, if appropriate."

1.1 Programmatics

The period of performance for 51MSR8AA was 04-OCT-1999 to 06-OCT-2002. Total funds allocated were $1.26M (FY00: $350K; FY01: $460K; FY02: $450K).

1.2 Project Staff

The following MITRE staff made significant contributions to this project over the course of its lifespan:

- Brant Cheikes (Principal Investigator)
- Andy Chisholm (Software Development)
- Marty Geier (WOSIT instrumentation)
- Abigail Gertner (Lead Designer/Developer)
- Brad Goodman (Research Support)
- Lisa Haverty (Instructional Design)
- Rob Hyland (User Action Tracking)
- Charles Kisner (CAASD Liaison)
- Laura Kurland (User Evaluation)
- Frank Linton (Research Support)
- Annette Riffe (Software Development)
- Linda Rodi (Instructional Design)
- Peter Schaefer (Research Support)

1.3 Background

Government workers today perform ever-increasing portions of their jobs with the aid of complex and often job-specific information systems. These systems have been growing in power and scope as they consolidate and streamline tasks that once were handled by teams of trained staff. The systems are upgraded regularly, and may be widely deployed around the world.

In the military, shortages of properly-trained operators have become common due to declining budgets, frequent personnel rotations, promotions, and attrition. Under these conditions, the two most common training approaches—classroom instruction at central training facilities and on-the-job training at field sites—are increasingly impractical and unaffordable.

One solution to providing affordable operator training in the workplace is to augment mission applications with *embedded training systems* (ETS)—task-specific training tools that operate in the same computing environment as the mission application to be used on the job. The goal of embedded training is to bring users quickly up to basic proficiency in system operation. After the initial training period, the ETS can remain available for review and refresher training.

In exploring the space of technical approaches to ETS development, we made the following assumption: To be effective learning-support tools, ETSs must support *highly interactive hands-on practice* and be capable of *individualizing their instruction* to the learning needs of each student. Consequently, we have focused our attention on ETSs that employ *intelligent computer-assisted instruction* (ICAI) techniques. ICAI systems use artificial intelligence (AI) techniques to deliver tailored instruction. Training provided by ICAI-based ETS is highly interactive because trainees practice problem-solving tasks on the mission application with guidance and feedback from the training system. Because trainees vary in knowledge and experience, the training component customizes its delivery to individual needs. The training system attempts to maintain an appropriate level of challenge, keeping students working at the edge of their competence, passing them quickly through material they demonstrably comprehend, and focusing attention on their weak skill areas.

1.4 Objectives

This project built upon the accomplishments of a previous cycle of MITRE-Sponsored Research: the project *Embedded Training for C4ISR System Operators* that was active during the period FY'97—FY'99. Those accomplishments can be summarized as follows:

- We learned how to *instrument* software applications. That is, we learned how to modify the run-time environment of a target software application in a way that enables other applications to receive real-time reports of actions on the graphical user interface (GUI) of the target application. This capability is critical for ICAI approaches to embedded training. We developed the WOSIT (Widget Observation, Scripting and Inspection Tool) technology for instrumenting X-Windows applications, and contributed to the development of the JOSIT (Java Observation, Scripting and Inspection Tool) technology for instrumenting standalone Java applications. Most significantly, we learned that we could effectively instrument applications without modifying the applications' source code. This enabled us to add an ETS to a given application after software development of that application had concluded.

- We learned how to track a software application user's progress against a detailed model of a known task. Given the ability to instrument a software application, we were able to develop algorithms for processing the real-time event reports from the instrumentation sub-system, comparing them to detailed models representing various ways a software user might accomplish tasks using the application, and determining where in the workflow the user currently was located. We designed a *command recognizer* that recognized meaningful software acts performed by means of sequences of GUI gestures.
- We learned how to provide relevant feedback. Once able to instrument and track users' activities, we developed simple mechanisms that allowed a prototype embedded trainer to answer two generic questions: *Where am I now?* (in the current task) and *What should I do next?* (either to continue the current task or initiate the next). These mechanisms illustrated a basic capability for automated support of trainees while they perform assigned exercises on the application to be learned.

As the previous cycle of research came to a close, we identified the following objectives for what became the current cycle of research:

- We need to develop principled instructional strategies for embedded training. Our first research cycle demonstrated that we could provide *relevant* feedback, in the sense that it relates to what the user is doing at any given moment. But what is really involved in providing effective training services? What sorts of instructional strategies and tactics should a computer-based training tool employ in order to help a learner become a skilled operator of a complex software application?
- We need to embody these principles in a computational artifact. To ground our work on instructional strategy design, we need to formalize our results in enough detail to allow them to be implemented in running software. This will give us a product that we can objectively test and evaluate.
- We need to evaluate the instructional effectiveness of the resulting artifact. The ultimate test of an instructional strategy is an experiment in which its impact on a human test subject, in terms of knowledge and skills acquired over time, is measured.

In Section 2, we summarize the project's accomplishments. Section 3 presents the details of a user-evaluation experiment conducted towards the end of FY02. General conclusions and suggestions for future work are presented in Section 4. The *List of References* lists publications that were produced during the project, and an appendix provides pointers to related web resources.

Section 2

Accomplishments

This section summarizes the accomplishments of the *Computational Embedded Training Strategies* project over its three-year lifespan. We begin with a description of the instructional strategies we designed and implemented (§2.1), then discuss the prototype Embedded Tutor Agent that we developed (§2.2). Our user evaluation, being one of the major accomplishments in FY02, is presented in detail in Section 3, rather than summarized here. In §2.3 we briefly describe three "spin-off" technologies created as a side benefit for the work on this project. We conclude with a discussion of this project's impact on MITRE's work program (§2.4) and the publications produced (§2.5).

2.1 Instructional Strategies

It seems intuitively obvious that at least one significant difference between "good" teachers and "bad" teachers is that the "good" ones routinely employ effective strategies for conveying domain knowledge and motivating their students to learn. These strategies influence many different aspects of instruction, including the selection, organization and sequencing of lessons, the choice of when to provide and when to withhold guidance and feedback, and the decision of what specific sorts of guidance and feedback to provide.

Early on we decided to focus our strategy-design efforts on decisions involving the choice and style of guidance and feedback. We decided to implement a simple, "hard-wired" strategy for selecting and sequencing exercises, that is, we designed our system to present exercises in a pre-selected sequence.

We made another fundamental choice early on in the project, namely, to fashion our embedded training system as a natural-language dialogue system, and to ground it on a formal theory of discourse (Grosz & Sidner, 1990).[1] As a result, we came to view computer-assisted learning devices as using natural-language utterances to implement their strategic choices. To this end, we formed a fruitful collaborative relationship with a team of researchers (Drs. Charles Rich, Candace Sidner, and Neal Lesh) at the Mitsubishi Electric Research Laboratory (MERL) in Cambridge. They had developed a software framework called COLLAGEN for implementing dialogue-oriented artificial agents. Collagen became the platform for the Embedded Tutor Agent developed over the course of this project, and served as a venue for code-exchange between MITRE and MERL.

We ultimately designed and implemented two broad instructional strategies for teaching software-application skills, which we called *demonstration mode* and *point-and-prompt mode*.

---

[1] B. J. Grosz and C. L. Sidner. (1990). Plans for Discourse. In P. R. Cohen, J. Morgan and M. E. Pollack, editors, *Intentions in Communication*, MIT Press, pp. 417-444.

The context for both strategies is an assigned hands-on exercise in which the learner completes a multi-step task using the target application.

In *demonstration mode*, the tutor demonstrates all steps of the task to the learner, providing explanatory comments as it goes. In *point-and-prompt* mode, the learner performs the steps of the task, but the tutor first *points* to the part of the target application's graphical user interface (GUI) to be manipulated, then *prompts* the student to perform a specific step. With these two strategies in its repertoire, our Embedded Tutor prototype was initially able to teach software tasks by first giving the learner an exercise in *demonstration mode*, then following it up with a similar exercise in *point-and-prompt mode*.

As our development efforts proceeded, and as we gained experience with our evolving prototype, we came to understand that it was not always appropriate to conduct an entire exercise in only one instructional mode. We realized that over a course of several exercises, skills learned in earlier exercises would be applied repeatedly in later exercises. Once learned, it seemed inappropriate for the tutor to demonstrate them. Yet later exercises would also introduce entirely new skills, in which case it made sense for the tutor to demonstrate them before asking the learner to perform them. So we introduced a *student model* into our prototype, which tracked each skill in the learning domain and kept a record of whether the student had seen it demonstrated or had performed it successfully (or unsuccessfully) after prompting by the Tutor. Drawing on the student model, we designed the Tutor to pursue a hybrid strategy, demonstrating skills that the student model indicated were new to the learner, otherwise pointing and prompting.

These were the strategies we were able to design and implement given the resources available to us. Clearly there are many other instructional strategies that would be worth investigating and implementing. For example, there is a variety of possible coaching strategies in which the Tutor gives the learner only high-level guidance (such as a general statement of a goal to achieve using the target software application), then remains mostly silent, only offering hints and advice when needed or upon request. Strategy development in these areas remains a fruitful direction for future work.

2.2 Development of Prototype

The most tangible accomplishment of this project was the development of an Embedded Tutor Agent for TARGETS (Terminal Area Route Generation, Evaluation and Traffic Simulation), a CAASD software prototype in the process of being transitioned to the Federal Aviation Authority. Dr. Abigail Gertner led both the design and implementation of the Tutor, building on the Collagen platform.

The final version of the Tutor consists of these major components:

- Collagen (agent framework);
- Protégé knowledge base management system (cf. http://protégé.stanford.edu);
- Protégé server (interface between Collagen and Protégé);

- JOSIT (GUI instrumentation);
- TARGETS (test-bed application);
- Student Model.

From the learner's perspective, three on-screen elements were significant:

1. The TARGETS application;
2. The Tutor's "home window" (cf. Figure 1);
3. The Learner's "home window" (cf. Figure 2).

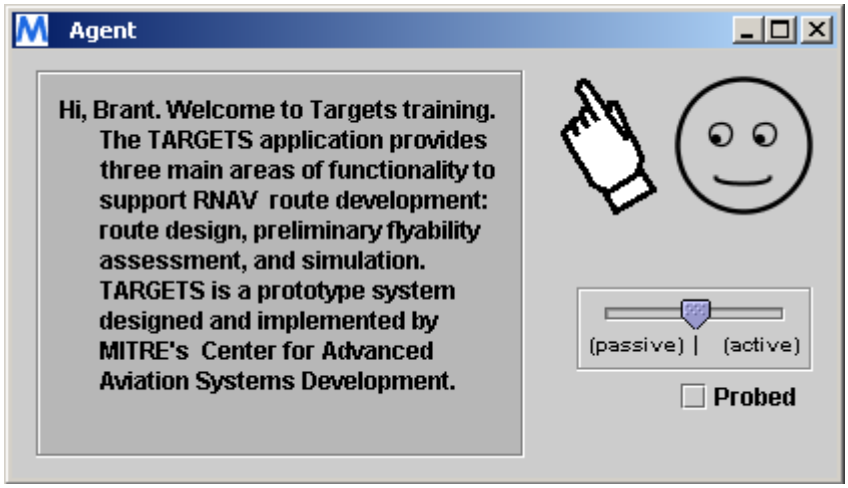

Figure 1: Tutor's Home Window

The Tutor's Home Window represents the Tutor. All instructional acts (natural language utterances) are presented in written text in this window. The pointing hand shown in Figure 1 also has the ability to move about the display, pointing to those parts of the GUI relevant to the current instructional step.

The Learner's Home Window represents the student, and enables the student to interact (in a somewhat limited fashion at present) with the Tutor.

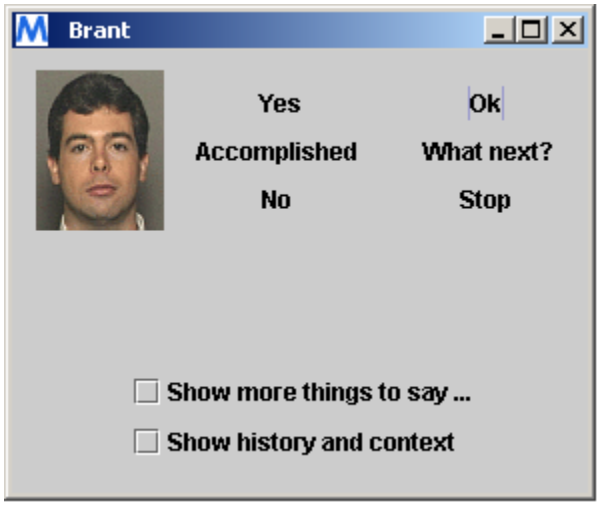

Figure 2: Learner's Home Window

In terms of coverage of the learning domain, the final implemented version of the Tutor supported tutoring on Area Navigation (RNAV) Route Generation, one-third of the TARGETS application functionality. The knowledge base contained models of 62 distinct *recipes*, each recipe defining a sequence of steps (called *acts*) for achieving a goal.

Recipes in the knowledge base were of two types: *instructional* recipes—recipes for achieving instructional goals—and *domain* recipes—recipes for achieving problem-solving goals using TARGETS. Of the 62 recipes in the knowledge base, 50 were domain recipes. This indicates that the Tutor's instructional knowledge was relatively compact and general purpose, and is consistent with our intuition that most of the knowledge that needs to be encoded in an ICAI-based tutoring system is domain knowledge rather than instructional knowledge.

2.3 Spin-Off Technologies

Three "spin-off" technologies were developed over the course of six years of research on embedded training:

1. Widget Observation, Scripting and Inspection Tool (WOSIT);
2. Java Observation, Scripting and Inspection Tool (JOSIT);
3. CORBA-Based Server for Protégé.

These are summarized below.

2.3.1 WOSIT

The Widget Observation Simulation Inspection Tool (WOSIT) is software that allows communication with a program's graphical user interface. WOSIT provides a way to:

- Observe user actions on a program's user interface;

- Inspect the properties of items on a program's user interface;
- Simulate user actions on a program's user interface.

In the current version of WOSIT, simulation functionality has not yet been fully implemented.

WOSIT requires no modifications to a program's source code, and no recompiling of the program is necessary. WOSIT runs unnoticeably in the background, and it uses negligible disk space, memory and processor time.

In terms of technical requirements, the application to be instrumented with WOSIT must (1) run on a UNIX platform that uses X-Windows, and (2) be dynamically linked with the X Intrinsics (Xt) and Motif (Xm) libraries.

WOSIT has been tested and developed on Solaris (versions 5 through 7) on Sun Sparcs, however any version of Solaris and brand of UNIX hardware should suffice. WOSIT is currently not supported on Linux.

WOSIT has been released under an open source software license, and is currently available from download both from MITRE's external website and from the Open Channel Foundation. As of this writing, 33 downloads of WOSIT have been logged.

WOSIT has influenced the Army's Advanced Training for DD2-N program, has been directly applied on the ENVI-Track program for the National Imagery and Mapping Agency (1.1 SY @ $210K in FY00), the Command Decision Training Army MOIE ($800K FY00-01), and is being used by the Naval Surface Warfare Center Dahlgren Division to support an internal embedded training development program.

2.3.2 JOSIT

JOSIT (Java Observation Simulation Inspection Toolkit) is an open Application Programmer's Interface for instrumenting applications written in the Java programming language. JOSIT seamlessly integrates with any Java application written using Sun Microsystem's standard library of Java graphical objects. JOSIT observes user actions, inspects the state of objects and scripts graphical events. Tools written in other programming languages may communicate with JOSIT via sockets.

Development of JOSIT was initiated under Brad Goodman's MSR *Intelligent Collaborative Learning Environments*. JOSIT was substantially extended, polished and released with funding from the embedded training MSR program.

JOSIT has been released under an open source software license, and is currently available from download both from MITRE's external website and from the Open Channel Foundation. As of this writing, 80 downloads of JOSIT have been logged.

JOSIT was directly applied on the Active Checklist for the Information Operations Planning Tool project (2.5 SY @ $400K FY00), and is an essential component of the Intelligent Agents for Distance Learning MSR.

2.3.3 Protégé Server

The Protégé Server is a package of Java software that implements a CORBA-Based Server for the Protégé knowledge-base editing environment. The Server software enables client applications to interact with Protégé knowledge bases using the Common Object Request Broker Architecture, a mechanism for distributed object communication.

The Server software has been released under the terms of the Mozilla Public License available at http://www.mozilla.org/MPL/. These are the same terms under which Protégé itself is released.

The Protégé Server has been directly applied on the Intelligent Agents for Distance Learning MSR, and on the Decision Support System for Computer Network Defense MOIE.

2.4 External Impact

This research has had measurable impact on MITRE's direct and indirect work program. On the direct side:

- Army DD2N Advanced Training (8.5 SY since '00). This program has its origins in a visit to MITRE Bedford in early 1998 by GEN (ret) Paul Gorman, at the invitation of David Lehman. GEN Gorman was interested in learning about MITRE's activities in computer-assisted instruction, and he heard briefings on a variety of activities, including the embedded training MSR program. He subsequently requested support from Dr. Brad Goodman at a briefing to the Defense Science Board. Soon thereafter a joint IDA-MITRE team was formed to develop a proposal to the Army Program Executive Office for C3 Systems (PEOC3S) at Ft. Monmouth recommending the development of an embedded training subsystem for the Army Battle Command System (ABCS). (WOSIT figured prominently in this proposal, though under the name GUI-SPT.) Funding commenced in FY00, and the MITRE effort has been ably managed by Dr. John Tyler and Dr. David Mireles.
- IOPT Active Checklist (2.5 SY @ $500K FY00). MITRE had been supporting the development of the Information Operations Planning Tool under an Advanced Concepts Technology Demonstration sponsored by Central Command and managed by the Air Force Information Warfare Center (AFIWC) in San Antonio. As the IOPT was being readied for a Military Utility Demonstration, AFIWC recognized the importance of ensuring that IOPT users were adequately trained. Our embedded training research was demonstrated, and this led to a funded program to rapidly develop a simplified embedded training capability for IOPT.
- NIMA ENVI-Track (1.1 SY @ $210K FY00). Our instrumentation capability caught the eye of a program manager within the National Imagery and Mapping Agency's Directorate of Operations, and we were tasked to prototype a capability to use instrumentation to capture and model analytic practices using the ENVI software for digital imagery analysis.
- USGC CRES (0.75 SY @ $150K FY99). Our skills in the area of instrumentation and modeling user activity led to a USGC summer study to assess the degree to which an enterprise message handling system's auditing component could provide data useful for rating the contribution of various information sources.

On the indirect program side:

- Command Decision Training MOIE ($800K FY00-01). The WOSIT technology was applied on this project, and it enabled the researchers to study how training on high-level decision making skills could be supported by computer-based tools.
- Core Dialogue MSR ($473K FY03). The insights that Abigail Gertner gained through the process of developing a dialogue-oriented tutoring system enabled her to effectively contribute to the development of a new MSR program focusing on grand challenges in dialogue system design.

2.5 Publications

Over the course of this project we participated actively in the academic research community, regularly publishing papers at major conferences (Artificial Intelligence in Education conference, Intelligent Tutoring Systems conference, User Modeling conference), and in the Journal of Artificial Intelligence in Education. The List of References contains a complete list of publications.

Section 3

User Evaluation

This section describes the design and presents the results of a formative evaluation conducted during August 13-15, 2002. Formative evaluation is an evaluation designed to help *form* a product, that is, establish its scope and design.

3.1 Test Subjects

MITRE subjects were recruited through an e-mail message sent to a CAASD (Center for Advanced Aviation Systems Development) community distribution list. Ken Wallace, our Subject Matter Expert, assisted in our recruitment efforts. There were two subject-selection criteria:

- familiarity with specific aviation terms (e.g., *waypoint*, *route*, *fix*, *NavAid*, *route*);
- lack of familiarity with the test-bed application (TARGETS).

Fourteen subjects were recruited, and all but one showed up for testing.

Test subjects had a variety of backgrounds: air-traffic control, pilots, area navigation (RNAV) route designers, avionics systems engineers, and mathematical modelers. Most were CAASD staff.

Several subjects had personal experience with computer-based training (CBT) applications. Three had made occasional use of MITRE's online courses, and one had used CBT in another job. One person was getting a degree in instructional design, hoping to create CBT programs.

3.2 Procedure

This subsection describes the testing procedure used during our formative evaluation.

3.2.1 Preparation for Testing

Subjects were first introduced to the purpose of this formative evaluation. They were given a brief history of the research project, then asked about their background and their experience with CBT and advanced training systems. They were told they would be trained on selected TARGETS tasks by a computer-based tutor, then asked to do the tasks again on their own, with minimal guidance.

3.2.2 Testing

Subjects were taught in one of two training conditions:

1. Speech+Written: the computer tutor communicates with the learner using both written text and spoken language;
2. Written-only: the computer tutor communicates using written text only.

In the Speech+ condition, seven subjects were tested. Three were AC5, three were AC4, and one was AC3. In the Written-only condition, six subjects were tested. Two were AC5, two were AC4, and two were AC3.

After using the tutor, subjects were tested on TARGETS without the tutor, but were given help by the experimenter if needed. Questions, comments, requests for procedural help and corrections of errors were recorded by the experimenter with additional logging backup provided by the tutor's built-in logging function (but without any tutoring going on).

Next, subjects were asked to verbally recall (without looking at the screen) the basic steps that they had learned from the tutor. Finally, subjects were given a questionnaire. They were asked to rate, check off, or write in (as provided) their responses to the questions. They were also encouraged to add additional comments if they felt it was necessary. The ratings were given on a seven-level Lickert scale.

The questionnaire had two forms: the Written-only Tutor form and the Speech+ Tutor form. They were identical except that the Speech+ form had three additional questions pertaining to the speech itself and also had tutoring preferences that included speech combinations. The questionnaire form (including the additional Speech+ questions) is included in Appendix A. Written comments are documented in Appendix B.

3.3 Analysis

Three measures were used to capture the results of the evaluation:

1. Verbal recall of the basic steps;
2. Performance on TARGETS;
3. Answers to the user questionnaire.

The results of the evaluation can be summarized as follows:

- Using the embedded tutor, subjects learned to perform six TARGETS tasks in approximately 20 minutes.
- There were some modality differences in performance and preference/satisfaction.
- There were some related user-interface and tutoring issues.

3.3.1 Verbal Recall

After subjects were taught and tested about the selected aspects of the TARGETS application, they were asked to verbally recall the basic steps. If they provided more detail, these were not counted as errors. Omissions were counted as errors.

The six basic steps were: *Start New Project*, *Import Runways*, *Import NavAids*, *Import Fixes*, *Create Route*, *Save Project*. *Start* and *Save New Project* involved one dialog window and several type-in fields, while the rest involved two windows or more, and used menu selection or check boxes.

Subjects were able to recall 5 out of 6 steps perfectly. On one middle task, 3 out of 13 subjects forgot to mention "Import Fixes."

There were no statistically significant differences between learning modalities, that is, between the Speech+ condition compared to the Written-only condition. The mean is indicated with an "M" instead of the X bar.

Speech+	(N=6)	M= 0.14	(Var. 0.14)
Written	(N=7)	M= 0.33	(Var. 0.27)

Not Significant (0.24)

3.3.2 Performance

After learning how to use TARGETS with the tutor, subjects were asked to perform the same task on TARGETS without the tutor. They were allowed to ask the experimenter for help, and she would provide unsolicited help if the user made an error that would cause the system to crash. Performance was measured three ways: time on task, accuracy and prompts.

3.3.2.1 Time On Task

When tutored by the embedded training system, subjects were consistently faster in the Speech+ condition than the Written-only condition by nearly 3 minutes. (The task was largely self paced.) This was not statistically significant, however, due primarily to the large variability it took for people to read the text in the written condition. Some people read aloud, which takes longer than silent reading. (This difference is important to remember in light of other results from the questionnaire.)

Speech+	(N=6)	M= 18.2 min.	(Var. 6.6)
Written	(N=7)	M= 21.7 min.	(Var. 32.3)

Not significant (0.11)

For solo TARGETS performance, subjects in the Speech+ tutoring condition consistently performed faster than subjects in the Written-only condition by over a minute. Although this difference was not statistically significant, it was almost so. (Again, this difference in performance is interesting in light of the questionnaire results.)

Speech	(N=6)	M= 3 min.	(Var 0.33)
Written	(N=7)	M= 4.2 min.	(Var. 2.2)

Not significant, but close (0.059)

3.3.2.2 Accuracy

There were two kinds of accuracy measures: Verbal and Performance. Each type of Accuracy measure will be discussed individually.

3.3.2.2.1 Verbal Recall Accuracy

After their solo performance on TARGETS, subjects were asked to recall the steps they had learned about TARGETS from the tutor. They were required to do this without using the computer. All subjects recalled five out of six tasks perfectly. Additions were not counted as errors, since some of the subjects' recollections were highly detailed and accurate. Three of the thirteen subjects forgot to mention the third task, *Import Fixes*. No differences in tutoring modality were detected.

Speech	(N=6)	M= 0.14	(Var. 0.14)
Written	(N=7)	M= 0.33	(Var. .27)
	Not Significant (0.24)		

3.3.2.2.2 Performance Accuracy

After their session with the Tutor, subjects were asked to do the same task again without the tutor. Subjects were assigned exercises having two component tasks:

1. Route Creation (a major task), and
2. Enlarge the Project Window (a minor task).

Half the subjects (6/13) were observed either to "fish around" (search for the Route Creation icon) or to choose the wrong icon. (More will be said shortly about the reasons for this.) There were no statistically significant differences in learning modality in terms of performance errors.

Speech+	(N=6)	M=0.71	(Var. 0.90)
Written	(N=7)	M=1.67	(Var. 1.87)*
	Not significant (0.09)		

*One subject made four errors. One error involved skipping an entire task (Import Fixes), which encompassed a set of subtasks. He also was one of the ones who forgot to mention it in Verbal Recall task.

3.3.2.3 Prompts

There were two kinds of prompts encoded in this evaluation: Conceptual and Procedural. Conceptual prompts involved subjects requesting information about a task that had to do with its purpose or its place in a larger context. For example, "Why would you want to create a route in the first place?" would be coded as a Conceptual prompt. A Procedural prompt had to with the mechanics of the actual step itself, such as clicking on Next to get the next set of instructions from the Tutor.

3.3.2.3.1 Conceptual Prompts

In each Tutoring condition, only one Conceptual prompt was needed—the question was along the lines of "why would you create a route anyway?" from a non-CAASD pilot. There were no statistically significant differences in learning modality.

Speech+	(N=6)	M= 0.14	(Var 0.14) (1 prompt)
Written	(N=7)	M= 0.17	(Var 0.14) (1 prompt)

Not significant (0.46)

3.3.2.3.2 Procedural Prompts

Subjects generally raced through the solo TARGETS performance for five of the six tasks. Learning modality did not show any significant differences in performance.

Speech+	(N=6)	M= 1.2	(Var 0.90)
Written	(N=7)	M=1.67	(Var 2.67)

Not significant (0.31)

Most procedural prompts fell into two categories:

1. Route Creation
2. Turn-taking (Mixed Initiative)

The reasons for these procedural prompts are discussed in the following section.

3.3.3 User Interface and Tutoring Issues

3.3.3.1 Route Creation

Route Creation required prompting and error recovery. Subjects had difficulty both locating the relevant icon and executing the route creation task itself. In TARGETS, the route creation task required putting down five waypoints at specific locations, then terminating the route by pressing the Escape key. A connector between each waypoint defines the air route.

More than half of the subjects needed to search for the green Route Creation icon. (Some were checking out the other icon names already.) Subjects were confused by the red Waypoint icon next to it primarily because of the yellow Tooltip name "Waypoint" that pops up. The name "waypoint" was more salient than the color difference. This is a TARGETS interface issue that bears some scrutiny and further testing.

Another UI issue was the delay between clicking on the screen and the appearance of a waypoint. The delay was sufficiently long enough to cause subjects to try clicking again.

3.3.3.2 Waypoints: Mixed Learning by Doing and Demonstration Paradigm

We had several instructional design problems with the route creation tutorial. The initial interactive tutorial for placing waypoints required placement of the waypoints at an exact

coordinate (latitude, longitude, minutes and degrees). This required painstaking accuracy and took a long time. After each waypoint, a subject had to click on Next to get the next coordinate from the Tutor. However, the connector between waypoints stretched toward the next window and was a source of human-factors confusion. As a result, we decided to demonstrate the placement of all five waypoints by the Tutor and have the student terminate the route creation process by hitting the Escape key.

Demonstrating the waypoint concept was successful in that subjects understood the idea. They understood they were supposed to click on the screen. However, subjects asked whether they were supposed to put the waypoints exactly where the tutor had and whether there was a "lat/long" list. In the next iteration of the demonstration, it might be preferable to re-word the tutor's instructions as, "Observe how waypoints are placed. When it is your turn to place waypoints, click your mouse on the screen at the desired locations".

Nearly all (12/13) subjects ended the route creation process without any hesitation by the hitting the Escape key.

3.3.3.3 Turn-Taking

Another instructional design issue detected prior to the evaluation required redesigning the tutor's approach to interacting with the student. In the original design, the Tutor would prompt the student by saying "OK?" at the end of each utterance. This indicated it was the student's turn, either to act as directed by the tutor, or to signal the tutor to proceed. To signal the tutor to proceed, the student was required to click a button labeled 'OK' on the tutor's interface to the student. Unfortunately, several TARGETS windows also had a button labeled "OK", which was used to accept the values displayed in the window. Thus, it was possible for there to be two different "OK" buttons on the screen, appearing in different windows, and used for different purposes—a clear human-factors defect. This led us to redesign the tutor-student interaction window, changing the label of the "OK" button to "Next".

Once formative testing began, it became apparent that the Tutor needed to clearly indicate when it was the subject's turn to respond or act. Subjects were often confused about whose turn it was, typically because the interaction was slow and so they became accustomed to just sit and wait. When they had waited long enough, and it became apparent that the Tutor was in fact waiting for *them,* they made comments like, "Oh, now *I'm* supposed to do [X]". Comments related to this issue appeared in the "What would you like to Change" section of the Questionnaire.

3.3.4 Questionnaire Results

The questionnaire was the same for both tutoring conditions, with the exception of three additional questions about the nature of the speech interaction itself. Subjects were asked to rate items on a Lickert scale from 1 to 7 and to provide free form comments on what they liked and what they would improve. Appendix A contains the questionnaire.

3.3.4.1 Tutoring

Subjects provided average to above average ratings for the Tutor itself as shown below. There were no significant differences between modalities. The average score is given below.

How well did the tutor explain the interface?	M=5.3
How well did the tutor explain the steps?	M=6.0
How well did the tutor give feedback?	M=4.7
How well did the tutor explain the "big picture"?	M=4.8
How well do you think you learned to use TARGETS?	M=4.9

3.3.4.2 Learning Modality Differences

Statistically significant differences appeared relating to learning modality in terms of motivation, pace and interaction. Subjects in the Speech+ condition definitely did not feel as satisfied with Tutor interaction as those from the Written only condition.

How well did the tutor hold your attention?

Speech+	M=4.1 (Var. 6.0)
Written	M=6.0 (Var. 2.4) sig. diff. (0.034)

How was the pace of the lesson?

Speech+	M=3.6 (VAR 3.6)
Written	M=5.0 (VAR 1.2) not sig. (0.06)

This perception of slowness is interesting because the actual data on Speech+ subjects showed that they were consistently faster both in the learning time on task and the solo TARGETS time on task condition.

I found the tutor interaction to be:

Speech+	M=3.9 (Var. 1.1)
Written	M=5.0 (Var. 0.8) sig. diff. (0.03)

Thus Speech+ subjects rated the Tutor as less able to hold their attention, slower, and more annoying.

3.3.4.3 Instructional Preferences

Learning modality differences were again apparent when subjects were asked how they would prefer to learn. They were asked to check off as many options as they wanted. Written-only subjects had three options: computer tutor (written) only, human instructor only and a combination of instructor and written tutor. Speech+ subjects had five options: speech only, written only, an instructor only, both speech and written, and a combination of an instructor, written, and speech.

Subjects demonstrated a clear preference for a computer tutor interacting using written text. They strongly liked the idea of a combination of a written computer tutor with a human instructor. This combination was preferred over either item alone.

Speech in any form was clearly not as desirable since less than half of those who were taught in the Speech+ condition thought speech was a good learning modality in combination with written text or with written and a human instructor. No one wanted to learn with Speech only.

I would prefer learning with:

	N = 13
Instructor and written tutor	(8/13)
Computer tutor/written only	(5/13)
Human instructor only	(4/13)

	N= 7
Speech and written	(3/7)
Instructor, written and speech	(3/7)
Speech only	(0/7)

The next section examines what subjects thought of the speech interaction itself.

3.3.4.4 Speech Interaction

Subjects were asked about general and specific aspects (loudness, clarity) of the speech interaction itself. They rated them below average to average, which does not account for the clear rejection of the Speech only tutor. During the session, subjects did not ask any sort of clarification of what the tutor said, but it should be noted that they also had the written text in front of them.

How did you like the speech interaction? M=3.6

How was the voice? Loudness (M=4.0) Clarity (M=3.9)

When listening to speech, subjects were observed to be fidgeting (e.g., drumming fingers) and sighing deeply. Subjects who were reading (silently or aloud) did not exhibit these signs. They seemed to be more cognitively engaged while they were reading, even though this took longer on average than the Speech+ condition. Thus, it appears that being lectured to, whether by computer tutor or by human instructor, is less cognitively engaging than reading.

3.3.4.5 Usefulness

Subjects were asked to select as many options as appropriate, and had the option to fill in their own opinions regarding how the tutor could be used, if at all. They strongly felt that the tutor was useful for help purposes, as a refresher, and for training (which also confirms a previous question that subjects strongly preferred learning with a computer tutor for an introduction and as a review). Subjects clearly did not choose "Not useful" (i.e., they did think it was useful) or "Job Aid".

Learning with a computer tutor is useful for:

Help	10/12
Refresher	8/13
Training	7/12
Job aid	0
Not useful	0
Other	4

In the "Other" fill -in option, four opinions suggested the tutor was useful for: "Complicated simulations - like TARGETS", "Professional development", "Qualification testing", and "As an introduction to TARGETS only".

3.3.4.6 Free-Form Comments

There were two questions that asked subjects to provide free-form input: "What did you like" and "What would you like to change."

3.3.4.6.1 Suggestions for Improvement

Most of the suggestions for improvement fell into two categories: speed and making the turn-taking clear. Subjects were clear with regard to increasing the speed of interaction: 8 out of 13 subjects felt this was important to improve. Furthermore, the Speech+ group felt this most strongly, since 6 out of 8 suggestions came from this group, which is consistent with their feelings about learning preferences.

Interestingly enough, for suggestions pertaining to turn-taking (7/13), most (5/7) of the comments also came from the Speech+ group. While somewhat speculative in this study, recent research literature on social interactions with computer-generated speech in the communicative media[2] is providing evidence that people expect computers to be more human if speech is involved and judge their applications accordingly. Humor, personality (introvert, extrovert), and other factors come into play with acceptance and credibility of what the computer "says". This certainly has implications for future human computer interaction design guidelines.

---

[2] Clifford Nass and Kwan Min Lee "Does Computer-Genrated Speech Manifest Personality? An Experimental Test of Similiarity Attraction", Proceedings of the CHI 200 Conference, The Hague, The Neterhlands (April 1-6, 2000)

Other scattered comments for improvement pertained to

- Feedback/tutoring (concept, more aggregation, specificity)
- Error handling
- Step selection/control
- User interface
- Motivation/appeal issues ("talking head")

3.3.4.6.2 What Users Liked

Users were generally pleased with the computer tutor. They felt it provided useful and effective instruction and they liked the idea of working on the actual application. What users liked about the system fell into the following categories:

- It was easy to use;
- Working with the actual system is helpful for learning;
- Detailed and clear instruction was provided on how to start using the system;
- It was efficient and quicker than reading a manual or getting human instruction;
- The pointing mechanism (an on-screen hand) was effective and highlighted relevant parts of the graphical user interface;
- The tutoring was self paced.

3.4 Conclusions

Our formative evaluation showed that the embedded tutor was effective at teaching six TARGETS tasks in under twenty minutes. The learning modality differences, while apparent, did not significantly affect performance. Written interaction was preferred over speech interaction because speech was perceived as slow and less motivating. Subjects saw the tutor as useful for help, review, and an introductory training. They felt that the tutor was easy to use and provided clear instruction on how to get started. Subjects liked the self paced aspect, and felt that working with the actual system was helpful for learning. This is consistent with the results of research on learning and instruction, where learning-by-doing has been shown to facilitate recall.

Subjects were clear about their principal concerns in their suggestions for improvement:

- Speed up interaction, improve error trapping;
- Add several levels of feedback;
- Add repeat-step and step selection;
- Address UI issues (includes clarifying turn-taking);

- Enhance aesthetics/appeal (change tutor's on-screen presentation, improve grammar and voice quality).

Section 4

Conclusions

Over the course of this project, much was learned, both broadly and deeply, about the technical challenges and opportunities presented by ICAI-based embedded training systems. Four conclusions stand out in our minds:

Highly interactive embedded training is technically feasible. The ability to instrument graphical user interfaces to obtain real-time information on user activity is a key enabler of ICAI-based embedded training. This research has convincingly demonstrated that we can add an embedded training system to a complex software application without requiring extensive, expensive modifications to that software application. Trends in the broader marketplace related to accessibility legislation (e.g., Section 508 of the Rehabilitation Act, as amended in 1998) suggest that WOSIT/JOSIT-like instrumentation capabilities are soon going to become standard features of computer operating systems and software applications.

Highly interactive embedded training is acceptable to users. Our user evaluation, albeit with a small subject population, indicates both that users seem to be able to acquire skills from an embedded training system, and that they find the experience acceptable.

GUI instrumentation is not a panacea. Without an ability to observe, inspect and simulate user actions on a software application, advanced embedded training simply is not possible. Our research has demonstrated that GUI solution is at least an "80% solution", but has limitations, particularly where visually-presented data is involved. In complex graphical situation displays, typical in military C4I applications, GUI instrumentation does not reveal everything that would be useful to a training application about what a user is seeing, doing and thinking. Clever instructional design can often mitigate these limitations, but not always. We consider it possible that full-scale embedded training solutions may demand a mixture of GUI instrumentation as employed in our research and custom modifications to the software application to be learned. We think this means that embedded training considerations must be accounted for at the earliest stages of an application's design, rather than at the end.

Development costs for ICAI-based embedded training systems remain high. The fact cannot be avoided that the complexity of developing an ICAI-based ETS is at least two orders of magnitude greater than conventional presentation-oriented Computer-Based Training, which is in turn an order of magnitude greater than developing instructor-led training. For advanced embedded training to be worth considering, a training domain must have two characteristics: (1) large and continuous flow of students, (2) stable curriculum. To achieve return on investment, the "cost per head" needs to be spread over a large student population, and training needs to be an ongoing process. In addition, the domain knowledge needs to be stable: the application needs to remain relatively stable in terms of functionality and interaction paradigm, and the skills to be learned need to be well-understood and relatively unchanging. In the absence of these characteristics, advanced embedded training systems are likely to be too costly to consider.

4.1 Directions for Further Research

This research left three areas relatively unexplored. These would be fruitful directions for further research:

- **Coaching Strategies.** Our intuition is that it is often productive to permit a student to flounder a bit, to make mistakes and learn from them, rather than holding them strictly to correct problem-solving paths. It is also important to transfer more initiative over time from the tutor to the student, to let students exert control over their learning, for the tutor to fade into the background as students become more skilled. How do we enable a computer-based tutor to employ these strategies?
- **Course Sequencing Strategies.** The goal of ICAI is to maintain an appropriate level of challenge at all times, keeping students working "at the edge of their competence", practicing learned skills in conjunction with new skills. Part of this involves deliberate selection of exercises, using an assessment of the learner's performance on "exercise N" to guide the selection of "exercise N+1". These sequencing strategies remain to be defined and tested.
- **Error Recovery Strategies.** During an embedded training session, the student is free to manipulate the application being learned, and it is not at all uncommon for students to make a variety of mistakes, changing the application's state before the training system has the ability to react. Sometimes these situations present learning opportunities—*teachable moments*, in the parlance of educational theory—and sometimes a tutor just needs to get a student quickly back on track. Effective error recovery strategies are needed, both to make embedded training robust in the face of student mistakes, and to take advantage of learning opportunities as they arise.

List of References

J. Rickel, N. Lesh, C. Rich, C. L. Sidner, and A. S. Gertner. (2002). *Collaborative Discourse Theory as a Foundation for Tutorial Dialogue*. In Proc. Sixth International Conference on Intelligent Tutoring Systems.

B. A. Cheikes and A. S. Gertner. (2001). *Software Instrumentation for Intelligent Embedded Training*. In Proceedings of the 2001 Interservice/Industry Training, Simulation & Education Conference, Orlando.

B. A. Cheikes and A. S. Gertner. (2001). *Teaching to Plan and Planning to Teach in an Embedded Training System.* In Moore, J. D., Redfield, C. L., and Johnson, W. L., editors, Artificial Intelligence in Education. Amsterdam: IOS Press, pp. 398-409.

J. Rickel, N. Lesh, C. Rich, C. L. Sidner, and A. S. Gertner. (2001). *Using a Model of Collaborative Dialogue to Teach Procedural Tasks.* In AI-ED 2001 Workshop on Tutorial Dialogue Systems, pp. 1-12, San Antonio.

J. Rickel, N. Lesh, C. Rich, C. L. Sidner, and A. S. Gertner. (2001). *Building a Bridge between Intelligent Tutoring and Collaborative Dialogue Systems*. In Proc. Tenth International Conference on AI in Education.

J. R. Davies, N. Lesh, C. Rich, C. L. Sidner, A. S. Gertner, J. Rickel. (2001). *Incorporating Tutorial Strategies into an Intelligent Assistant.* Proceedings of the International Conference on Intelligent User Interfaces. Santa Fe, NM.

A. S. Gertner, B. A. Cheikes, and L. A. Haverty. (2000). *Dialogue Management for Embedded Training.* AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications.

A. S. Gertner, L. A. Haverty, and B. A. Cheikes. (2000). *Tutorial Strategies for Embedded Training.* Workshop on Modeling Human Teaching Tactics and Strategies, Fifth International Conference on Intelligent Tutoring Systems.

B. A. Cheikes, M. Geier, R. Hyland, F. Linton, A. S. Riffe, L. L. Rodi, and H.-P. Schaefer. (1999). *Embedded Training for Complex Information Systems*. International Journal of Artificial Intelligence in Education, 10, 314-334.

B. A. Cheikes, M. Geier, R. Hyland, F. Linton, L. Rodi, H. Schaefer (1998). *Embedded Training for Complex Information Systems.* In Proceedings of ITS'98, 4th International Conference on Intelligent Tutoring Systems, San Antonio, August, pp. 36-45.

B. A. Cheikes. (2001) *Increasing System Usability by Teaming Trainers with Developers*. The MITRE Information Technology Advisor, 1(15): The MITRE Corporation.

B. A. Cheikes, B. A. Goodman and F. N. Linton. (2001) *More Effective Training Through Software Instrumentation.* Research News, 4(1): The MITRE Corporation.

B. A. Cheikes and B. A. Goodman. (1999). *Embedded and Collaborative Training Bring Users Up To Speed.* The Edge, 3(1): The MITRE Corporation.

Appendix A

Embedded Training User Questionnaire

Please answer the following questions on a scale from 1 to 7, where 1 corresponds to "not well" and 7 corresponds to "very well".

1) How well did the tutor explain the interface?
2) How well did the tutor explain the steps?
3) How well did the tutor give feedback?
4) How well did the tutor explain the "big picture" of what you are supposed to be doing?
5) How well did the tutor hold your attention?
6) How was the pace of the lesson?
7) How well do you think you learned how to use the TARGETS tool?
8) I found the tutor interaction to be

Annoying			neutral			enjoyable
1	2	3	4	5	6	7

9) I would prefer learning with a tutor (check all that apply)

__Not at all

__As an introduction

__During class with an instructor

__As a review

__Anytime

10) I would find learning with an embedded training computer tutor useful for:

__Help systems

__Training systems

__Refresher Courses

__Job Aid

__Not useful

__Other: (please explain)

11) Comments

What did you like about using this prototype?

What would you like to change?

For Written-Only Tutoring Mode

12) I would prefer learning with (check all that apply)

__computer tutor interaction only

__human instructor only

__an instructor and written interaction

For Speech+ Tutoring Mode

12) How did you like the speech interaction?

Annoying			neutral			enjoyable
1	2	3	4	5	6	7

13) How was the voice?

Loudness

Too loud			fine			too soft
1	2	3	4	5	6	7

Clarity:

Garbled			fine			Clear
1	2	3	4	5	6	7

14) I would prefer learning with (check all that apply)

__speech interaction only

__written interaction only

__only an instructor

__both speech and written interaction

__an instructor, written, and speech interaction

Appendix B

User Comments

	What did you like about using this prototype?
1	A creditable beginning.
2	Seemed easy to use; just needs more polish. Additional funding would provide additional improvement
3	Self paced, pointed to relevant parts of the interface.
4	Tutor was useful and I learned the minimum steps required to create an RNAV procedure.
5	It was detailed enough to give you a fairly good exposure to the material or application - in this case TARGETS.
6	It was working with the actual application.
7	Visual cue (pointy hand) effective.
8	I though there was a lot of potential. Good for new students if can be made to act much faster.
9	(none)
10	(none)
11	The overall concept is good. Having this prototype walk me through the necessary steps helps with the learning process.
12	I thought it was a good introduction to TARGETS and was probably quicker and more efficient than reading a manual or getting human instruction.
13	Very clear instructions and very helpful to do the training with the real TARGETS system.

Appendix C

Related Web Collections

- **Project Homepage:**
  http://info.mitre.org/pub/projshare/2002/51MS/R8AA/document/ET_MSR_Homepage.html

  This site collects all project-related information and documentation.

- **WOSIT Homepage:** http://www.mitre.org

  The Widget Observation, Scripting and Inspection Tool, a software tool for instrumenting Motif-based X-Windows applications, can be downloaded from this site. As of 9/9/02, there were 33 distinct downloads recorded.

  WOSIT is also available from the Open Channel Foundation:
  http://www.openchannelsoftware.org/projects/WOSIT

- **JOSIT Homepage:** http://www.mitre.org/

  The Java Observation, Scripting and Inspection Tool, a software tool for instrumenting standalone Swing-based Java applications, can be downloaded from this site. As of 9/9/02, there were 48 distinct downloads recorded.

  JOSIT is also available from the Open Channel Foundation:
  http://www.openchannelsoftware.org/projects/JOSIT

- **Protégé Server Homepage:**
  http://www.mitre.org/tech_transfer/protegeserver/index.html

  The CORBA-Based Server for the Protégé Knowledge Base System can be downloaded from this site. The Protégé homepage, which contains a link to the Protégé Server site, is here: http://protégé.stanford.edu

- **Collagen Homepage:** http://www.merl.com/projects/collagen/

  This site describes the Collagen agent framework developed by a team at the Mitsubishi Electric Research Lab in Cambridge, MA.

- **MITRE Advanced Training for Digital Divisions 2-N:**
  http://transfer.mitre.org/STF60-69/65/25365/Transfer/DD2-N_Training/

  This site collects materials pertaining to MITRE's support of the Army Program Executive Office for C3 Systems (PEOC3S) Program Advanced Training for Digital Divisions 2-N.