VOLUME I

Proceedings of the Contributed Sessions 1993 Conference on Intelligent Computer-Aided Training and Virtual Environment Technology

Sponsored by

Software Technology Branch
NASA/Johnson Space Center
and
U.S. Army Training and Doctrine Command
and
RICIS/University of Houston-Clear Lake

May 5-7, 1993

Gilruth Recreation Center
NASA/Johnson Space Center

Edited by

Patricia R. Hyde and
R. Bowen Loftin
University of Houston-Downtown

Virtual • Environment • Technology

Intelligent • Computer • Aided • Training

(NASA-TM-111097) PROCEEDINGS OF
THE 1993 CONFERENCE ON INTELLIGENT
COMPUTER-AIDED TRAINING AND VIRTUAL
ENVIRONMENT TECHNOLOGY, VOLUME 1
(NASA. Johnson Space Center) 125 p

N96-14974
--THRU--
N96-14996
Unclas

G3/61 0067590

Preface

The 1993 Conference on Intelligent Computer-Aided Training and Virtual Environment Technology attracted a larger than anticipated number of participants. Both the number and quality of accepted contributed papers was higher than in the antecedent 1991 Conference on Intelligent Computer-Aided Training. All those who played a part in the success of the conference are to be commended for a job well done.

This Proceedings is organized in the same manner as the conference's contributed sessions, with the papers are grouped by topic area. The table of contents which follows is laid out in this manner.

To the surprise of the Conference Program Committee the number of contributed papers devoted to virtual environment technology increased from a handful in 1991 to 33 in 1993. This is evidence to the increasing attention being given to this technology in many areas, including aerospace and military training.

Although not documented in these Proceedings, the exhibits portion of the conference added a great deal to the overall success of the Conference. The vendors who attended were exceptionally pleased with the numbers of visitors to their booths and with the serious interest in the products that those visitors displayed.

Looking ahead to the next conference, planned for early 1995, the Conference Program Committee believes that attention will be given to the integration of Intelligent Computer-Aided Training and Virtual Environment Technologies as a means to enhance the efficacy of both technologies when applied to training and education tasks.

Finally, a great debt of thanks is owed to Bob Savely for his insightful guidance, to Chris Ortiz and Ellis Henry for their role in organizing the exhibits, to Carla Armstrong (now Colangelo) for arrangements and registration support, to the Program Committee for a job well done, and to Patricia Hyde for her dogged determination in editing these Proceedings.

R. Bowen Loftin
Houston, Texas
April, 1994

Proceedings of the

1993 Conference on Intelligent Computer-Aided Training and Virtual Environment Technology

edited by
Patricia R. Hyde
and
R. Bowen Loftin
University of Houston-Downtown

A1: VE Training for Space Flight

Patrick J. Kenney David B. Palumbo	Instructional and Technological Development Activities of a Synthetic Solar System Exploration Environment	1
Michael Goldsby Abhilash Pahdya Ann Aldridge James Maida	A Virtual Reality Browser for Space Station Models	2
Steve Tanner Kellie Miller	The Use of High Fidelity CAD Models as the Basis for Training on Complex Systems	10

A2: Virtual Environment Hardware

Arden Strasser	Improved Visualization of Virtual Environments	14
John W. Worthington K. Duncan W.G. Crosier	Network and User Interface for PAT Dome Virtual Motion Environment System	15
Brian Vandellyn Park	The Personal Motion Platform	20

A3: Knowledge Acquisition for ICAT & VE

Tim Saito R. Bowen Loftin	Issues in Capturing and Representing Domain Knowledge and Polygonal Assignment for Virtual Environments	26
Pamela K. Fink L. Tandy Herren	A Knowledge Engineering Taxonomy for Intelligent Tutoring System Development	27

David C. Wilkins	Generic Expert System Shells and Apprenticeship Tutoring: The Representation and Use of Explicit Control Knowledge	39
David B. Palumbo David Germain Will Lidwell	The Impact of Cognitive Task Analysis as a Knowledge Acquisition Method in Intelligent Computer Assisted Training Systems	40
	A4: Multimedia in ICAT Systems	
Charles P. Bloom Frank Linton Brigham Bell Edwin Norton	The Learn, Explore and Practice (LEAP) Intelligent Tutoring Systems Platform: Multimedia Integration	41
Peter T. Bullemer Rose W. Chu Nagi Kodali Mike Villano	The Home and Building Control Fundamental Tutor: A Design Framework for the Multimedia Instruction of Declarative Concepts	58
Stephen Desrosiers Debra Bettis	SMART—Situated Multimodal Advanced Real-time Trainer	59
	B1: VE in Training & Education I	
Sharon A. Stansfield	A Computer-based Training System Combining Virtual Reality and Multimedia	60
Brett Achorn Norman Badler	A Virtual Training Environment with Simulated Agents	65
Andrew F. Payer J. Mark Voss Laurie Spargue	Development of Virtual Environments For Medical Educaiton and Training	69
	B2: Virtual Environment Software I	
Dr. Gerald Pitts Mark Robinson Steve Strange	A Microbased Shared Virtual World Prototype	70
Georges G. Grinstein David A. Southard J. P. Lee	Virtual Environment Architecture for Rapid Application Development	75
C.F. Codella R. Jalili L. Koved J.B. Lewis	A Development System for Multi-User, Distributed Virtual Environments	83

Chuck Kosta Dr. Patrick D. Krolak	Rapid Prototyping 3D Virtual World Interfaces within a Virtual Factory Environment	84

B3: Models in ICAT Systems

Rose W. Chu Mike Villano	Application of Knowledge Space Theory for Student Modeling	92
Randall W. Hill, Jr. W. Lewis Johnson	Impasse-Driven Tutoring for Reactive Skill Acquisition	93
John D. Farquhar J. Wesley Regian	The Effects of a Dynamic Graphical Model during Simulation-based Training of Console Operation Skills	111
Bradley J. Wiederholt T. Kiki Widjaja Joseph Y. Yasutake Hachiro Isoda	Advanced Technology Training System on Motor Operated Valves	117

B4: ICAT Commercial Applications

Charles P. Bloom Brigham Bell Frank Linton Edwin Norton	The Learn Explore and Practice (LEAP) Intelligent Tutoring Systems Platform	128
Minoru Kiyama Yoshimi Fukuhara	An Authoring System for Practice Environment in the Network Service Field	145
Clifford M. McKeitham, Jr.	Integrated Intelligent Training and Job Aiding for Combustion Turbine Engines	153
Amir Hekmatpour Gary Brown Randy Brault Greg Bowen Larry Grant	FTDD973: A Multimedia Knowledge-Based System & Methodology for Operator Training & Diagnostics	161

C1: VE in Training & Education II

Christian-Arved Bohn Wolfgang Krueger	Embedding Speech into Virtual Realities	171
Major Donald Pryor James Larsen	Virtual Instrument Technology	179

Laurie McCarthy, Michael Pontecorvo, Frances Grant, Randy Stiles — Spatial Considerations for Instructional Development in a Virtual Environment.......... 180

R. Bowen Loftin, Mark Engelberg, Robin Benedetti — Virtual Environments for Science Education: A Virtual Physics Laboratory... 190

C2: Virtual Environment Software II

Peter Astheimer — Sounds of Silence--How to Animate Visual Worlds with Sound.......... 191

Long V. Truong — Living Color Frame System: PC Graphics Tool for Data Visualization Applications.......... 203

Lac Nguyen, Patrick J. Kenney — Design Strategies and Functionality of the "Visual Interface for Virtual Interaction Development" (VIVID) Tool.......... 212

C3: ICAT Architectures & Authoring Systems

Brigham Bell, Charles Bloom, Frank Linton, Edwin Norton — The LEAP Intelligent Tutoring Architecture.......... 218

Michael Orey, Ann Trent, James Young, Michael Sanders — Streamling ICAT Development Through Off-the-Shelf Hypermedia Systems.......... 219

P. Ahmed, A. Al-Dhelan, A. Shah — On the Construction of a Knowledge-Based Intelligent Tutoring System (KB-ITS).......... 234

Kimberly C. Warren, Bradley A. Goodman — Engineering Intelligent Tutoring Systems.......... 235

C4: ICAT Education & Medical Applications

Rodger Marion, Bruce R. Niebuhr, B. Chris Renten, John Bernstein — An Idealized Computer-Based Patient Record for Teaching Medical Diagnosis and Care Planning.......... 244

K. Macura, R. Macura, V. Toro, E. Binet, J. Trueblood — Case-Based Teaching System for Radiology.......... 252

Madeline Kovarik — Intelligent Computer Aided Training Systems in the Real World: Making the Technology Reachable for Everyday Educators.......... 256

Noah Rifkin, Hans ten Cate, Alison Watkins — ICAT and the NASA Technology Transfer Process.......... 260

D1: Assessing VE for Training

Mark S. Schlager, Randall J. Mumaw, Douglas G. Hoecker — Analytical Tools For Designing Virtual Environment Training Systems: Nuclear Power Plant Maintenance Applications.......... 261

Donald R. Lampton, James P. Bliss, Bruce W. Knerr — Assessing Human Performance in Virtual Environments.......... 270

Joseph P. Hale II — Marshall Space Flight Center's Virtual Reality Applications Program.......... 271

D2: VE & Human Systems I

Joseph Psotka — Virtual Egocenters as a Function of Display Geometric Field of View and Eye Station Point.......... 277

Joseph Psotka, Sharon Davison — Cognitive Factors Associated with Immersion in Virtual Environments.......... 285

D.E. Parker, D.L. Harm, F.L. Florer, K. Davino, N.C. Skinner, J.T. Kuennen — Attitude Awareness Training: An Aid to Effective Performance in the Virtual Environment?.......... 298

D3: ICAT Theory & Natural Language

Frank Linton, Brigham Bell, Charles Bloom, Edwin Norton — Tutoring Methods and Strategies in LEAP.......... 305

Michelle R. Sams — An Intelligent Foreign Language Tutor Incorporating Natural Language Processing.......... 315

Wayne A. Nelson — Knowledge Acquisition and Interface Design for Learning-on-Demand Systems.......... 321

Bruce Porter, Art Souther — Improving the Explanation Abilities of Tutoring Systems.......... 328

D4: ICAT Applications in the Military

Tucker Maney Henry Hamburger	VIS/ACT: The Next Episode	338
Sylvia Acchione-Noel Joseph Psotka	Mach III: Past and Future Approaches to Intelligent Tutoring	344
Walter G. Butler Richard R. Laferriere Philipp A. Djang	Training Mix Model	352

E1: VE Applications in Engineering

Cory W. Logan Henk Prenger Jim Clark Liew Wu Michael E. Goldsby Jim Maida	Virtual Environment and Computer-Aided Technologies Used for System Prototyping and Requirements Development	361
Steve Bryson	The Virtual Windtunnel: Visualizing Modern CFD Datasets with a Virtual Environments	390
Mark Bolas	Practical VR–Five Years of Lesson's Learned	396

E2: VE & Human Systems II

K. M. Duncan D. L. Harm W. G. Crosier J. W. Worthington	Using Virtual Environment Technology for Preadapting Astronauts to the Novel Sensory Conditions of Microgravity	397
D.L. Harm F.E. Guedry D.E. Parker M. F. Reschke	Development and Implementation of Inflight Neurosensory Training for Adaptation/Readaptation (INSTAR)	402
Robert S. Kennedy Kevin S. Berbaum Lawrence Hettinger Michael Lilienthal	Ataxia and Other Posteffects of Flight Simulators: Should Virtual Reality Systems Have Warning Labels?	407

E3: Knowledge Acquisition for ICAT

Vincent J. Kovarik Jr.	Autononmously Acquiring Declarative and Procedural Domain Knowledge for ICAT Systems	408

Will Lidwell, David Palumbo — Automating Knowledge Acquisition Processes: An Analysis of the Psychological and Technological Issues Surrounding the Automated Elicitation of Expert Knowledge. 415

David Germain, Bob McNenny, David Palumbo — Beyond Knowledge Acquisition - A Blueprint for Knowledge Construction. . . . 416

Debra Bettis, Stephen Desrosiers, David Mulcihy, Ken Ruta — Robo-Cat–An Intelligent Robotics Trainer. 417

E4: ICAT Applications in Aerospace

Mark Rolincik, Michael Lauriente, Harry C. Koons, David Gorney — An Intelligent Computer-Aided Tutoring System For Diagnosing Anomalies Of Spacecraft In Operation. 418

Capt. Keith B. Shoates, Patrick Moran — Training Augmentation Device (TAD) for the Air Force Satellite Control Network (AFSCN). 419

Instructional and Technological Development Activities of a Synthetic Solar System Exploration Environment

Patrick J. Kenney (LinCom) and David B. Palumbo
Advanced Knowledge Transfer Group
University of Houston - Clear Lake
Houston,TX 77058

Advanced graphics tools, interactive simulations, intelligent computer-aided trainers, and expert systems all offer greater interactivity for educational and training systems . Astronomy and planetary science instruction are taking advantage of these systems by augmenting curriculums with computer-enhanced pictures of celestial bodies and animated series of still photos to provide learners with greater insight to planetary mechanics and properties. However, even with these more action-based computer systems, there remains a fundamental problem - limited exercise of the extensive human perceptual system.

Synthetic (virtual) environments, can be more encompassing than traditional computer-human interaction because of their highly interactive nature and responsiveness to the user's perceptual and cognitive needs. Not only can instructional material be presented through a variety of sensory modes, but it can also be manipulated to achieve various perspectives and reinforce supporting concepts. This presentation will discuss technological and instructional development activities, operational functionality, and cognitive implications of a synthetic solar system environment. It will provide the learner with three primary capabilities: 1) exploration of planets and objects, 2) experiments on and around the planets, and 3) the ability to apply "what-if" scenarios to known systems and processes. Potential uses for this environment include planetary science and orbital mechanics instruction, as well as satellite and space vehicle operations prototyping and training.

N96- 14975

A Virtual Reality Browser For Space Station Models

Michael Goldsby, Abhilash Pandya, Ann Aldridge
Lockheed Engineering and Sciences Company
2400 NASA Rd. 1, mail code C-44
Houston, Texas 77058
goldsby@graf10.jsc.nasa.gov

James Maida
NASA, Lyndon B. Johnson Space Center
maida@graf6.jsc.nasa.gov

ABSTRACT

The Graphics Analysis Facility at NASA/JSC has created a visualization and learning tool by merging its database of detailed geometric models with a virtual reality system. The system allows an interactive walk-through of models of the Space Station and other structures, providing detailed realistic stereo images. The user can activate audio messages describing the function and connectivity of selected components within his field of view. This paper presents the issues and trade-offs involved in the implementation of the VR system and discusses its suitability for its intended purposes.

1. INTRODUCTION

This paper describes a Virtual Reality browser made at the Graphics Analysis Facility (GRAF) at NASA/JSC in Houston.

The GRAF is a service and research group providing a variety of graphics services to NASA and NASA contractors. In broadest terms, the GRAF provides the means for its customers to visualize the products of their imagination. It helps them visualize equipment that does not yet exist and maneuvers that have yet to be carried out. The GRAF maintains an extensive repertoire of detailed geometric models of the Space Shuttle, its payloads, and the various Space Station designs in different stages of construction. From them, it produces color printouts, transparencies and animations on video tape.

A VR capability finds a place as an extension of the work the GRAF already does. It gives the customer another means of viewing the models. Its advantages and drawbacks compared to the other media used at the GRAF are discussed below.

2. DESCRIPTION OF THE SYSTEM

2.1. Hardware

The system's hardware components are shown in Figure 1. They consist of two Silicon Graphics Reality Engines, up to two more Silicon Graphics workstations, an Apple Macintosh Super Mac personal computer, a Virtual Research head-mounted display (HMD) with earphones, and two Bird magnetic trackers. The Reality Engines are used to render a stereo pair of images which are routed to the displays in the HMD. The Mac plays pre-recorded messages on request, which are heard through the earphones of the HMD.

2.2. Operation

Wearing the HMD, the user can "fly" through one of the geometric models in the GRAF database (see Figure 2). A magnetic tracker positioned on top of the HMD gives his current direction of gaze. Another tracker is used as a "throttle" to control the user's travel. In the present system, the user may only travel in the direction of gaze with a forward or backward sense. The throttle tracker is held in the hand and has only three meaningful positions: forward (palm down), backward (palm up) and stationary (palm sideways). (An earlier version of the system allowed the throttle tracker to set direction and speed of travel, but most of us VR neophytes found that too confusing.) The user can inquire about the object in the center of his field of view. If there is a pre recorded message corresponding to that object, it is played through the headphones of the HMD. The message may include information about the function and connectivity of the selected object. The user presently registers the inquiry through the mouse of one of the Reality Engines, but work is in progress to replace the mouse button clicks with throttle-hand gestures. A click on the right mouse button arms the inquiry, causing a large red cross to appear at the center of the field of view. Another click of the right button selects the object lying under the cross, or a left button click disarms the inquiry; in either case the cross disappears.

2.3. Software

2.3.1. Software Structure

The system's software is structured as a set of servers and a single client. The client may be thought of as representing the "ego" of the user. There are two drawing servers (one for each eye), a tracking server, and a sound server. In addition, a small sound production program runs on the Mac. A diagram of the software components is given in Figure 3. Each drawing server produces one member of the stereo image pair. The tracking server reads the Bird trackers and returns the current orientation and position of the user. The sound server conveys requests for pre-recorded message to the Macintosh.

An advantage of using a client-server approach is that the client and servers may be distributed about the local-area network for performance or convenience. Each drawing server runs on one of the two SGI Reality Engines. The tracking server runs on another SGI workstation because the trackers happen to be connected to it. The client and sound server can run on any workstation on the network.

2.3.2. Conversion of an Existing Drawing Program to VR Use

An in-house program called DMC is used by the GRAF to view the models on the workstation screen and to produce hardcopy stills and animations. This same program is used to draw the images for the VR system. It might be of interest to anyone considering doing something similar to know what we had to go through to convert an existing drawing program to VR use.

DMC already produces 3D perspective projections from user-selected viewpoints. To make it usable for VR, it was necessary to do the following:

1) drive DMC from the client program rather than interactively from the workstation keyboard;

2) make the projection match the geometry of the HMD; and

3) make the viewing transformation match the current user position and orientation; and

4) synchronize the appearance of the new images for the two eyes.

It proved easy to make these changes, in part because we had access to the DMC source code.

The first item can be accomplished in various ways, for instance by creating an additional UNIX process to take the place of the user. By making some simple additions to the source code we could do it even more directly, by making the entire VR episode an extended user command.

To make the projection match the HMD geometry, we followed the outline of Robinett and Rolland [1]. The Virtual Research helmet uses the LEEP optics, the same as the VPL EyePhones described in [1]. In both systems, the optical axes do not pass through the center of the LCD display screens. DMC did not allow off-center viewing windows; that capability had to be added to it. We made no correction for the optical distortion introduced by the LEEP system [1].

DMC already computes the viewing transformation according to user-specified position and orientation.

On double-buffered workstations, image synchronization can be obtained by synchronizing the swapping of the buffers. To separate the swapping of the graphics buffers from the rest of the drawing procedure required changes to the DMC source code. Because we sometimes had to run the two drawing servers on machines of different speed, synchronization was clearly necessary. (It was probably advisable in any case, since clipping can cause a difference in the complexity of the two images; the off-center viewing windows only tend to increase the difference.)

2.3.3. Data Structures for the Selectable Messages

In order to draw a correspondence between the pre-recorded messages and the visible images, it was necessary to superimpose another level of organization on the model data. In this case it was easy to do, because the describable units corresponded to subtrees of the tree representing the image. We simply kept a list of describable elements (read in from a file, hence easily altered) in which the node name of the graphical subtree for the element was listed opposite the file name of the pre-recorded message. For more involved interactions with the model, however, it seems likely that describing the connection between the interaction and the displayed image could well be the most difficult part of the problem.

3. TRADE-OFFS

The models built by the GRAF are extremely detailed. For that reason, it takes a long time to draw them. Refresh times of a second on the SGI Reality Engine are not uncommon. Because of the low resolution of the displays in the HMD (208 by 139 pixels), most of that detail is lost, anyway. Because of the great redraw time, the illusion of continuity is lost. The slow refresh and low resolution interfere with the sensation of immersion. However, the latest models (and they change daily) with their detailed geometry are what our customers are interested in seeing.

The media typically used by the GRAF now are printed images, transparencies and animation recorded on video tape. Our VR system may be compared to these media:

	real-time selection of viewpoint by the user	continuity	resolution	stereo images	sound
VR	Y	N	low	Y	Y
animation	N	Y	high	N	Y
stills	N	N	high	N	N

The VR system has the advantage of providing immediate access to a large collection of models of interest to the GRAF's customers. Also, it provides features that other media lack (stereo images and real-time viewpoint selection). Moreover, it was easy to implement, since it uses an already-existing drawing program. But it has the disadvantages of low refresh rate and low resolution.

4. FUTURE DEVELOPMENT

Several avenues of future development are planned. One is to represent the user by a human model in the environment. The GRAF has a long-standing involvement in biomechanical research and modeling and maintains an anthropometrically correct human modeling capability. The model will be used to do reach studies and is also expected to increase the sense of immersion.

Movement within the virtual environment will also be made more convenient. An "elevator" mechanism will be added to allow the user to rise and fall without changing his head orientation, and a reorientation mechanism will be implemented to allow the user to change his head-up direction. Some more intuitive way of flying will be devised, perhaps using a joystick or a third magnetic tracker. At present, the system reads the orientation of the head-tracking sensor but ignores its position; position movements will be included so that the user can look around an object without having to fly around it.

We are also investigating ways to increase the refresh rate of the system (see Appendix).

We plan to study the effectiveness of the system as a tool for visualization, learning and design. Several of the designers of the Single Launch Space Station have shown interest in the system and said that they found it helpful to view their designs in it. Their comments made apparent the need for better motion control. They also expressed a desire for lower lag time (no surprise to us) and higher resolution.

ACKNOWLEDGMENTS

We would like to thank Lorraine Hancock, Mary Helm, Kim Tran and Jennifer Nguyen for their modeling support.

REFERENCES

[1] Warren Robinett and Jannick P. Rolland, "A Computational Model for the Stereoscopic Optics of a Head-Mounted Display", *Presence*, Vol. 1, No. 1, Winter 1992.

APPENDIX

Several approaches to speeding up the drawing rate suggest themselves:

1) Rebuild the models to simplify them.
2) Change the drawing algorithm or the model data structures.
3) Algorithmically simplify the models.
4) Switch among smaller local models depending on the user's position.

Rebuilding the models would be uneconomic; it would require more effort than it is practical for us to expend on it.

The drawing algorithm and data structures are already efficient; changing them without changing the number of polygons to be drawn would not change the drawing speed very much.

It is reasonable to ask if there is any kind of algorithmic simplification which could be applied to the data, either off-line or on the fly. (One sort of algorithmic simplification is common and is used in this system, namely clipping. It is viewpoint-dependent, hence must be done on the fly.) For instance, might it be possible to change the level of detail of the model or the resolution of the image depending on the viewer's position? Note that simplifications such as these might imply changing the data structures or drawing methods.

An algorithmic simplification at a gross level would be to switch among different local models as the user moves about. For instance, when the user is outside the station, all data structures for the interior detail might be omitted, and when he is inside a module, all structures exterior to that module might be omitted.

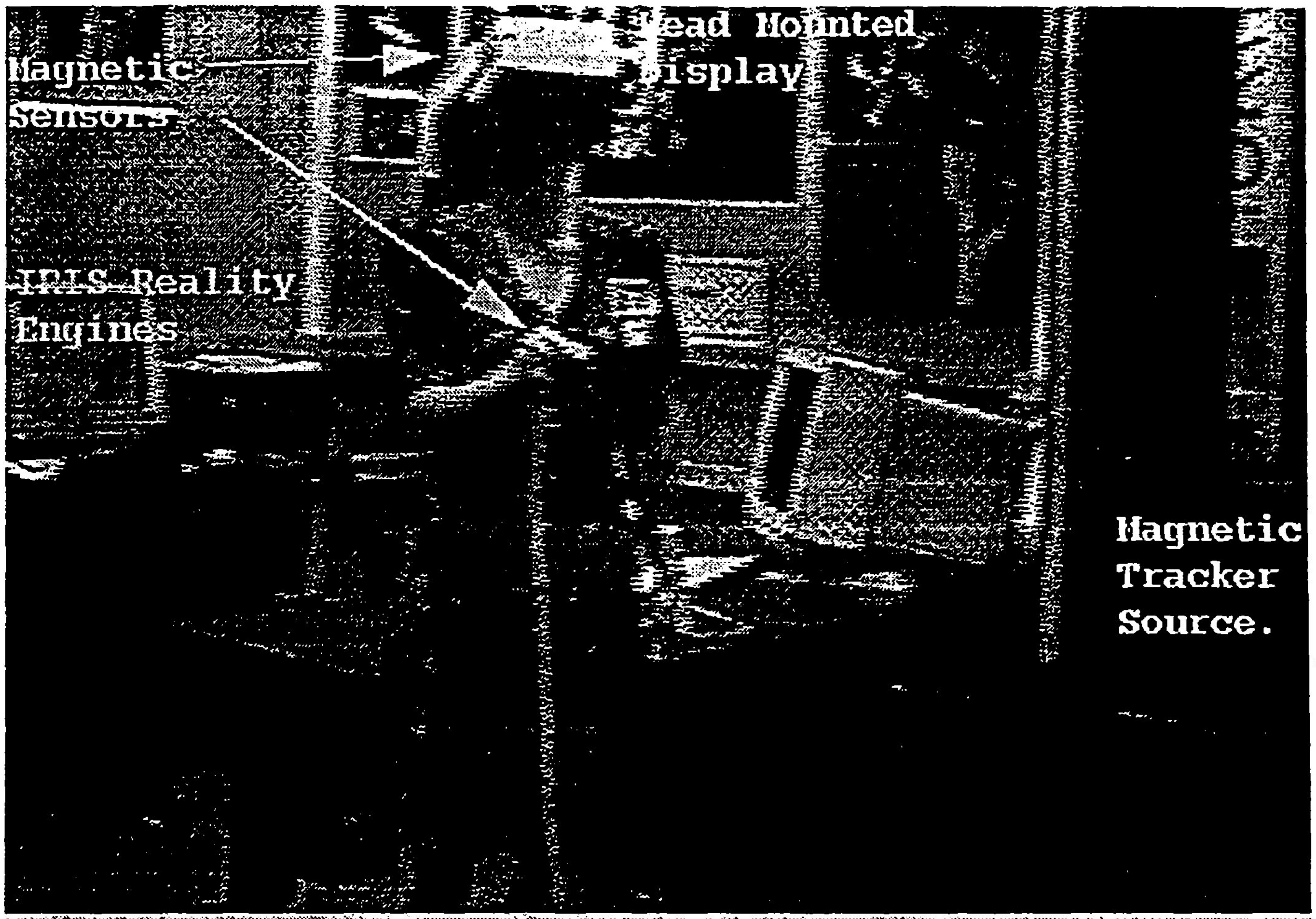


Figure 2 - Virtual Reality in Use.

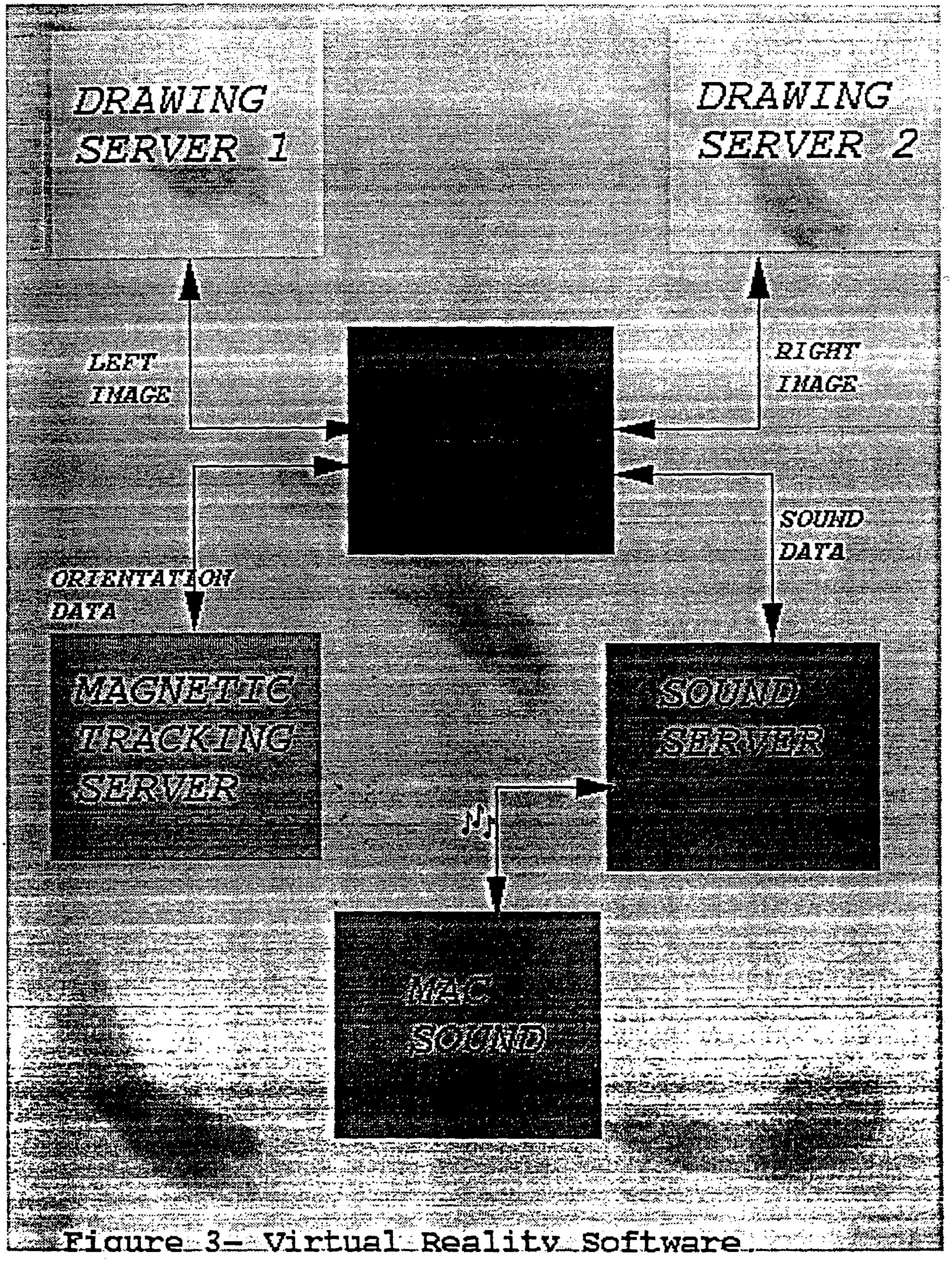


Figure 3- Virtual Reality Software

N96- 14976

The Use of High Fidelity CAD Models as the Basis for Training on Complex Systems

Kellie Miller and Steve Tanner
kellie@hsvaic.boeing.com
steve@hsvaic.boeing.com
Boeing Missiles and Space Division
499 Boeing Blvd., MS JY-58
P.O. Box 240002
Huntsville, AL 35824-6402

Abstract

During the design phases of large and complex systems such as NASA's Space Station Freedom (SSF), there are few, if any physical prototypes built. This is often due to their expense and the realization that the design is likely to change. This poses a problem for training, maintainability, and operations groups who are tasked to lay the foundation of plans for using these systems.

The Virtual Reality and Visualization Laboratory at the Boeing Advanced Computing Group's Huntsville facility is supporting the use of high fidelity, detailed design models that are generated during the initial design phases, for use in training, maintainability and operations exercises. This capability was used in its non-immersive form to great effect at the SSF Critical Design Review (CDR) during February, 1993.

Allowing the user to move about within a CAD design supports many efforts, including training and scenario study. We will demonstrate via a video of the Maintainability SSF CDR how this type of approach can be used and why it is so effective in conveying large amounts of information quickly and concisely. We will also demonstrate why high fidelity models are so important for this type of training system and how it's immersive aspects may be exploited as well.

Introduction

High fidelity Computer Aided Design (CAD) models provide an excellent basis for conventional design and analysis of complex systems. Boeing's Huntsville Visualization Laboratory has demonstrated that they also provide a good substrate for training and operations planning activities. In conjunction with SSF design engineers, we have shown how CAD models can be used to graphically demonstrate functional sequencing, identify the location of critical maintenance access points, and, with the addition of accurate human models, demonstrate the accessibility of these maintenance points. This demonstration was very favorably received at the SSF CDR in February of 1993.

The current visualization capability is based on the use of a proprietary visualization tool that allows interactive viewing of a shaded, three dimensional image of a CAD model in real time. Both immersive and non-immersive versions of the tool exist. The view perspective is interactively controlled with a mouse or other input device. Alternatively, a predetermined viewing path through the model provides a convenient and repeatable method of conducting a "tour" of the model. This path is specified by a B-spline.

While fully immersive, stereo views of the models are possible with the tool, the resolution of our Head Mounted Display (HMD) equipment is inadequate for viewing the complex models. This proves less restrictive than expected. The emphasis on engineering analysis favors high resolution displays over the wide field of view at lower resolution provided by HMDs.

This paper will discuss four aspects of this work at the Huntsville Visualization Lab. We will describe our efforts to perform maintainability analyses, expansion of the application domain to operations planning and training, planned enhancements to our tool suite and future directions of this research.

Maintainability Analyses

Maintainability analysis studies the design attributes of a piece of equipment that are salient to the maintenance of the equipment. Critical to this analysis are a high fidelity model of the equipment, a set of models of typical human forms, and a relatively high fidelity simulation of the kinematics of the equipment and the human form. Of only slightly less interest is a simulation of the physics of the environment in which the maintenance will be performed.

An example maintainability analysis was conducted using a human model and an actual equipment rack. This analysis was presented at the SSF CDR in February, 1993. It showed the step by step procedure used to remove a pump assembly from a rack. We imported a model of the 95th percentile American male in the zero-g posture to provide a human form for this analysis. Other standard human models are available but were not used in the example study. These models would be used to conduct an exhaustive maintainability study.

The equipment models that were used in our example analysis were imported directly from the SSF Engineering Database. The visualization software allowed each model to be operated on independently, i.e. displayed, hidden, translated, rotated, etc. The amount of detail imported from each CAD model was bounded by the consideration of the overall complexity of the set of models to be viewed and was specified when the models were translated.

In order to demonstrate the rotation of the rack into a maintenance position and the translation of the pump out of the rack, multiple models of the rack, the pump and the human form, each in different orientations and positions, were loaded simultaneously and the operator turned the display of the nominal models off and the display of the corresponding translated (or rotated) models on manually.

Access to utility disconnect points was demonstrated by moving the viewing perspective through the access path that would be used by the astronaut performing maintenance. This provided a rough idea of how interactive analysis would look. The maintainability team liked the approach and announced their intention to model additional maintenance procedures in this fashion.

Operations Planning

Operations planning historically is not feasible until the first hardware prototypes are built. It is difficult to foresee operational constraints from studying two dimensional CAD drawings. This forces the consideration of operational issues to be deferred until late in the design process. Most design decisions have been committed to by this time and design interdependencies have proliferated. Design changes at this stage are typically prohibitively expensive.

Imagine the enthusiasm that an opportunity to effect the operational aspects of a design early in the design process generates in the operations planning staff. Analysis of this sort is being pursued with the SSF operations planning organizations. These organizations can evaluate design proposals for operational characteristics before the first prototype is built. The results of these analyses can be fed back to the design engineers for incorporation in design revisions.

Another benefit of early involvement by the operations planning staff is an early start on developing the operator training for a piece of equipment. Operator training requires understanding of operational procedures. The early development of these procedures allows training developers to start their activity much earlier in the equipment development cycle. As a result of their early involvement, training developers can participate in the analysis of equipment and operations designs adding a training perspective to and exerting a timely influence on these designs.

Other beneficial side effects of earlier training development include higher quality training resulting from more time to develop training materials and earlier availability of training materials. Earlier availability of training materials increases flexibility in the scheduling of training programs. All of these benefits can be equated with higher quality operations procedures and training programs at lower cost.

Planned Enhancements

Several capabilities are required to effectively realize the planned maintainability analyses, operations planning, and training development activities. While turning multiple copies of a model in different locations and orientations on and off is adequate to demonstrate the potential benefit of this approach, interactive rotation and translation of models must migrate into the basic functionality of the tool suite.

Furthermore, kinematic modelling of object movement, i.e., constraining the movement of a model to the set of motions that are physically possible, is also required to fully realize the potential benefits of this approach. Kinematic modelling is only half of the picture, however. Behavioral modelling must be provided to generate completely interactive simulations of the environments of interest.

Much of this functionality could be provided through the addition of a scripting language. Ideally this language would provide functional abstraction, conditional execution and hooks for executing external programs. These capabilities are currently undergoing detailed analysis and design.

Future Directions

Simulation packages of adequate fidelity for modelling various behavior of interest already exist but mechanisms to provide connectivity between the visualization environment and these external simulation packages must be established. This allows us to use existing simulation work directly and also simplifies the scope of any built in scripting language.

A method of integrating descriptions of detailed behavioral models with visual models must be developed. A general world model building tool would allow information at various levels of abstraction to be added to the world description without the danger of inconsistencies creeping in that is inherent in separately constructed models.

A grand vision is emerging of a complete virtual implementation of a program from start to finish, before any hardware is built. This would include a virtual prototype of not only the equipment, but the facility that would produce the equipment, the operations procedures associated with the equipment, and any special training requirements of the equipment. Such extensive simulation would allow the trade-offs between proposed projects to be examined in depth before any resources were allocated to the implementation of any one project.

One could even imagine a suite of tools for building immersive simulations that were available as an immersive simulation, giving new meaning to the phrase "Programmers' workbench." The work that is already in progress at our lab is taking the first steps toward realizing this vision.

Conclusions

This work has met with much enthusiasm from the engineers working directly on various design activities. They appreciate the ease with which mistakes can be found in the three dimensional representation of the CAD drawings. The operations planning group appreciates the early opportunity to evaluate the operational feasibility of the design and provide design feedback at a stage where minor redesign is still more economically feasible. And the training developers appreciate the extra time before development in their schedule to develop the training materials

While the non-immersive tool is extremely valuable to present activities, fully immersive applications are not yet useful. This is primarily due to the lack of resolution of current HMD technology. We anticipate this issue to be addressed by enhanced HMDs in the next two or three years. The addition of a scripting language will allow exploration to occur at a much faster pace.

The connectivity with other behavioral modelling software is inherently more difficult. World building software that integrates visual and behavioral modelling is much further from availability.

Bibliography

Aukstakalnis, Steve and David Blatner. *Silicon Mirage: The Art and Science of Virtual Reality*. Berkeley, CA: Peachpit Press, Inc., 1992.

Helsel, Sandra K. and Judith Paris Roth. *Virtual Reality: Theory, Practice, and Promise*. Westport, CT: Meckler Publishing, 1991.

Rheingold, Howard. *Virtual Reality*. New York: Summit Bks, 1991.

Improved Visualization of Virtual Environments

Arden Strasser
Virtual Reality, Inc.
485 Washington Avenue
Pleasantville ,NY 10570
914-769-0900 914-769-7106

Efforts into human interface hardware have not matched those in software libraries for many types of simulation-based training. The limitations of viewing flat panels are composed to large screen displays, infinity displays, 3-D glasses, boom-mounted displays and binocular head-mounted displays (HMDs).

Only spatially-tracked head-mounted displays provide the ability to "fly-though' large datasets of terrain, architecture, manufacturing, and human anatomy. Important requirements include: field of view, image overlap, resolution, center of gravity, see-through and head-tracking latency and repeatability.

The future directions of HMD designs are discussed, as well as some novel scenarios for a ship's bridge simulation and on-board pilot training.

N96- 14977

Network And User Interface For Pat Dome Virtual Motion Environment System

J. W. Worthington, K. M. Duncan W. G. Crosier
Krug Life Sciences
1290 Hercules, Suite 120
Houston, Texas 77058

ABSTRACT

The Device for Orientation and Motion Environments Preflight Adaptation Trainer (DOME PAT) provides astronauts a virtual microgravity sensory environment designed to help allieviate the symptoms of space motion sickness (SMS). The system consists of four microcomputers networked to provide real time control, and an Image Generator (IG) driving a wide angle video display inside a dome structure. The spherical display demands distortion correction. The system is currently being modified with a new graphical user interface (GUI) and a new Silicon Graphics IG. This paper will concentrate on the new GUI and the networking scheme.

The new GUI eliminates proprietary graphics hardware and software , and instead makes use of standard and low cost PC video (CGA) and off the shelf software (Microsoft's Quick C). Mouse selection for user input is supported.

The new Silicon Graphics IG requires an Ethernet interface. The microcomputer known as the Real Time Controller (RTC), which has overall control of the system and is written in Ada, was modified to use the free public domain NCSA Telnet software for Ethernet communications with the Silicon Graphics IG. The RTC also maintains the original ARCNET communications through Novell Netware IPX with the rest of the system. The Telnet TCP/IP protocol was first used for real-time communication, but because of buffering problems the Telnet datagram (UDP) protocol needed to be implemented. Since the Telnet modules are written in C, the Ada pragma "Interface" was used to interface with the network calls.

INTRODUCTION

The main purpose of the Preflight Adaptation Training (PAT) laboratory at the Johnson Space Center (JSC) is to help reduce or eliminate Space Motion Sickness (SMS) symptoms and to improve neurosensory performance during the initial days of flight as well as helping in readaptation to earth's gravity following a space flight. The JSC PAT laboratory has two trainers; a Tilt-Translation Device (TTD) and the Device for Orientation and Motion Environments (DOME). This paper will discuss the networking scheme and graphical user interface (GUI) for the DOME system.

The DOME system creates a virtual reality environment to simulate some of the sensory rearrangements that astronauts experience on-orbit. A computer generated scene is projected onto the dome surface. Projection onto the dome surface requires distortion correction. The trainee can experience passive motion stimuli or can control his/her own motion with a six degree of freedom hand controller. Head movements can also be enabled to drive the scene. The microcomputer kinematics control can be set to simulate various motion environments e.g. microgravity.

Four microcomputers (PCs) and two Silicon Graphics (SGI) Crimson VGXT graphics computers, used as Image Generators (IG), drive the DOME system. The PCs communicate with each other over an ARCNET network using calls to a proprietary virtual token ring network through the Novell Netware IPX kernel. One of the PCs also communicates with the SGI Crimsons using NCSA Telnet network calls over an Ethernet connection. The Ethernet also connects the SGI machines with each other.

The instructor/operator selects various training environments using one PC which is known as the Instructor Operator Station (IOS). The IOS employs a graphical user interface with mouse selection.

The other PCs are: the Real-Time Conroller (RTC) which is the 'executive' or has basic control of the system and is the PC which communicates with the SGI Crimsons; the Real-Time Positioning System which accepts trainer device input from the trainee or instructor and provides kinematics processing; and the Data Logging and Display System (DLDS) which can display selected signals as on an oscilloscope.

Problems with special hardware and proprietary software led to rewriting the software and changing the display for the IOS. The fact that the SGI machines use Ethernet for network communication required that the RTC be able to support an Ethernet connection as well as maintain the ARCNET communication with the rest of the system. Budget constraints forced a low cost solution for these items.

GRAPHICAL USER INTERFACE

History

The original IOS used Multisynch monitors with a 768x1024 resolution for the user display. In order to generate this display special graphics boards were installed. These special graphics boards were accessed by making calls to special software written in FORTRAN; therefore, the original IOS was written in FORTRAN.

The display output could best be described as pages rather than windows since they were static in nature and selection was a lot like turning to a page in a book. Several pages were provided by the subcontractor; they were quite detailed and the resolution allowed much information per page. Some of the pages were supposed to allow the user to change some of the parameters in the system's computer kinematics and input control.

Problems

The subcontractor had originally expected to allow the user to select items by using a mouse. However, for some reason perhaps because of memory conflicts with the graphics boards and mouse driver, or interfacing the FORTRAN code with assembler calls to Int33H, the mouse never worked correctly. Our efforts in getting the mouse to work within this setup also failed. It was relativley time consuming to use cursor keys on these large display pages waiting for the cursor to arrive at the desired location.

Changes that the user made to the DOME parameters on the IOS exhibited inconsistent results when sent to the RTC. Values were wrong or missing which meant the desired change was not made. The major problem appears to have been the difference in data representations in FORTRAN on the IOS machine transferring to Ada on the RTC machine. We made modifications that helped, but there were still inconsistencies.

The special graphics cards failed. This would have left the DOME inoperable except that we had written a very terse non-graphic IOS using Ada which allowed system operation. This failure pointed out the dangers of relying on special hardware items. The manufacturer was a local business so the boards were repaired in less time than if they would have to have been shipped. Also, the graphics cards and software were geared to the Multisynch monitors only. Even though we have two of these monitors, each one failed and needed repair within a few weeks of each other. This again pointed out how much we were relying on special hardware.

As trainer use increased we found parameters that we would like to add to the IOS to allow the instructor/operator to select. The acquisition of the SGI IGs along with Paradigm Simulation's VisionWorks software provided some new capabilities that the IOS needed to let the user select. The software code as received from the subcontractor and the myriad FORTRAN graphics calls did not lend itself easily for modification. Also, most of the pages that were originally provided were now useless.

Solution

When the monitors failed it was time to find a new way to support the IOS. The terse Ada program was a place to begin, but interfacing it with C code graphics calls caused conflicts and did not work. We were also on a very limited budget at this time, so we needed to use materials we had on hand. We had a CGA monitor and graphics card and Microsoft Quick C which provides an extensive graphics library. A short test program proved that using Quick C with CGA would be fairly easy and very economical.

Mouse support was easily implemented by using Int33H functions called from the Quick C code.

By this time the requirements for what the user would need to be able to select from the IOS were firmly established, although these requirements do change over time. The pages that allowed selection of unused items were eliminated. New pages were added to provide support for the new capabilities provided by the VisionWorks software and SGI IGs e.g. 'floating' models and 'spinning'. The display should still be thought of as pages, but with a modular programming structure new pages can be programmed quickly. The worst limit posed by the CGA graphics is the resolution. Selections have to be relatively large so items per page are limited more severely than with the previous system or EGA.

The system allows user direct entry for some parameters. There has been no problem 'packaging' the C structured variables into a network packet that the Ada code on the RTC retrieves and uses, or vice versa.

The current IOS greatly decreased the time required for the instructor/operator to set up a session in the DOME. The mouse selection is much faster than the cursor keys. By making the pages pertain to what is required, the instructor/operator upon selecting an option (eg 'spinning' the scene) allows the system to immediately respond only to that item, whereas before all parameters were passed to the RTC. Having modularized and well-commented code in a familiar language has made this possible. This also allows for easy modification - adding or deleting pages for desired functions, and there is no 'wasted' code for unecessary pages.

Future

In the near future we hope to incorporate the IOS onto a Macintosh. This will allow true windowing rather than displaying pages. The choice of the Macintosh over Windows on a PC also stems from what is on hand. We already have development tools and experience in Macintosh programming, but we would have to purchase the Windows Development Kit and learn it. Also, our end users (the neuro lab scientists) use Macintosh computers at their desks and are already familiar with its menu scheme, dialog boxes, scroll bars, etc.

Modular programming structures will continue to be used; or perhaps object oriented class structures. Higher display resolution will be gained. It is also possible that we could combine the functions of the IOS and one of the other systems onto this single machine.

NETWORKING DOME COMPUTERS

History

The DOME system is set up to use networked microcomputers to allow a degree of parallel processing for the real-time operation. The Real Time Positioning System (RTPS) receives input from the hand controllers and helmet position. It performs calculations relating to the orientation of the hand control to the subject, thresholding, and kinematics. Its output is a six degree of freedom vector of changes in position (Delta Position Vectors or DPVs) in body coordinates which it sends to the Real Time Controller (RTC) over the network. The RTC takes the DPVs from the RTPS and transforms them into world coordinates, integrates these into a new world position vector (eyepoint), and sends these eyepoints to the IG for display. Therefore, the RTPS can be processing the next subject input while the RTC is finishing with the previous input. This turns out to be a low cost method of implementing parallel processing of the math intensive equations necessary for the real time operation of the trainer. Our original image generator had a PC front end which was connected to the same network in order to receive the eyepoints from the RTC. Additionally we have a Data Logging and Display System (DLDS) PC which can be used to show various signals in real-time. This PC is also connected to the network.

The network architecture chosen by the subcontractor uses ARCNET cards and coaxial cable at the physical level. The network uses Novell Netware IPX as the kernel. The applications make calls to a library of functions which implement a proprietary virtual token ring network called Micronet developed by Microsim. The ARCNET has a throughput of about 2.5Mbps.

Ongoing problems with the original IG ultimately led to the acquisition of the two SGI Crimsons.

Problems

The specifications for acquiring the new SGI machines required that communication between them and our system be implemented with an Ethernet connection using the TCP/IP protocol. It was determined that the RTC would be the PC which would connect to the SGI. This raised several questions. What software would we need for Ethernet and TCP/IP on the PC? Would the RTC be able to maintain its ARCNET capabilities and take on the Ethernet connection? If not, would we need a bridge or some other interconnection device; or, as an alternative, would it be worth the time, effort and cost to modify the software in all of the PCs in order to have one Ethernet network? Would the RTC Ada code interface with the network software if it was not written in Ada? What would be the cost to modify the system?

Solution

Knowing the possibility that memory conflicts and Interrupt Request (IRQ) conflicts could occur we decided to go ahead and try to support both networks on the RTC. If it worked, fine; if not, we would still need the Ethernet and TCP/IP software for another PC, which would be used solely for communication with the SGI machines, or if we modified the system to use only Ethernet.

An Ethernet card was ordered. It is supported by the NE2000 driver, could support both thin-net or thick-net cabling, and had four selectable IRQ lines. A transceiver for adapting from the AUI Ethernet port on the SGI to a BNC connection for thin-net was secured.

The networking software required would only need to be a small subset of what was available. It would need to support creating a socket, opening a connection, sending packets, and receiving packets. It turned out that NCSA at the University of Illinois, Champaign-Urbana, had TCP/IP software avaliable in the public domain by the name of NCSA Telnet. The source code was provided and downloaded over the Internet. Only the necessary modules needed for the DOME system were compiled from the C source files and placed into a library along with the application functions that referenced the Telnet functions.

In order for the Ada code in the RTC to access the network calls written and compiled into a C library, package ETHNET was created. This package contained Ada procedures for the Ethernet network and used pragma INTERFACE and pragma INTERFACE_NAME to access the C functions. The Ada code also had to be linked with the C library file.

With the software installed and the hardware in place, a test was conducted between the RTC and an SGI Crimson. The proper IRQ line was finally set on the Ethernet card, a good address for installing the NE2000 driver was selected, and the communication between the machines worked fine. Expanding the test to include the ARCNET calls to the other machines also worked fine; we did not have any conflicts by using two different network cards in the RTC PC.

While testing in real-time it was discovered that the SGI code was taking the TCP/IP packets and buffering them about three at a time. This slowed down the real-time performance and caused 'jerky' motion. It was finally decided to change from the TCP/IP protocol to the UDP (datagram) protocol because UDP is a 'don't care if received' type protocol and SGI would not buffer these packets. The NCSA Telnet code when compiled for the PC included the datagram service functions so this change was made quickly. The datagram service solved the real-time problem and is working very well.

Future

The current system PCs are 286 and 386 based machines running at 8 to 10 MHz. We hope to minimize the reliance on networking by combining the fuctionality of the RTC, RTPS, and IOS into one or two PCs since the power of PCs has greatly increased over what the current system has.

Futhermore, we plan to have a single network supporting the datagram service and TCP/IP over the Ethernet. This will eliminate the proprietary network and enable greater control over the networking software.

The Personal Motion Platform

Brian Vandellyn Park
Flogiston Corporation
462 Capehill
Webster, TX 77598
713-280-8554

Presented at the 1993 Conference on Intelligent Computer Aided Training and Virtual Environment Technology, May 5-7th, 1993, Gilruth Recreation Center, NASA/Johnson Space Center

Abstract

The Neutral Body Posture experienced in micro gravity creates a biomechanical equilibrium by enabling the internal forces within the body to find their own balance. A patented reclining chair based on this posture provides a minimal stress environment for interfacing with computer systems for extended periods. When the chair is mounted on a 3 or 6 degree of freedom motion platform, a generic motion simulator for Cyberspace is created. The Personal Motion Platform provides motional feedback to the occupant in synchronization with their movements inside the digital world which enhances the simulation experience. The turnkey system can be integrated with existing head mounted device (HMD) based simulation systems.

Introduction

Neutral Body Posture.
The Neutral Body Posture (NBP) is the fundamental posture of humans in the absence of gravity. It is a minimal stress posture caused by the internal muscles and body structure finding their own biomechanical equilibrium and balance in the absence of external forces (1). Since Skylab, designing equipment conducive to this posture has been a NASA requirement (2). Astronauts work, rest and sleep in this natural posture.

The Flogiston chair has been developed to apply neutral body posture concepts to Earth bound applications, based on the concept that the posture is just as valid for relaxation in one gravity as it is in the micro gravity of outer space. The NBP is made practical in a one gravity environment by compromising the posture with Savasana, a yoga posture of relaxation practiced for thousands of years (3). The resulting posture is called the Flogiston posture, and is as near to the NBP as gravity will allow. The postures are illustrated in Figure 1.

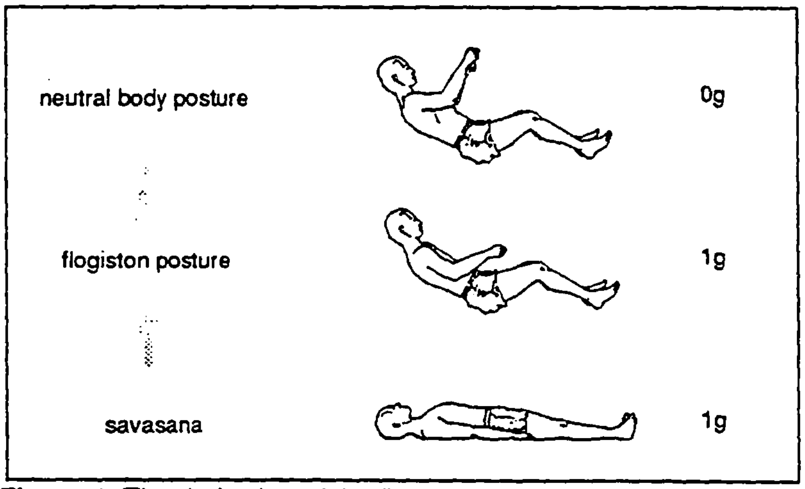


Figure 1. The derivation of the flogiston posture

N96- 14978

The Personal Motion Platform

Brian Vandellyn Park
Flogiston Corporation
462 Capehill
Webster, TX 77598
713-280-8554

The Neutral Body Posture experienced in microgravity creates a biomechanical equilibrium by enabling the internal forces within the body to find their own balance. A patented reclining chair based on this posture provides a minimal stress environment for interfacing with computer systems for extended periods. When the chair is mounted on a 3 or 6 axis motion platform, a generic motion simulator for simulated digital environments is created.

The Personal Motion Platform provides motional feedback to the occupant in synchronization with their movements inside the digital world which enhances the simulation experience. Existing HMD based simulation systems can be integrated to the turnkey system. Future developments are discussed.

We now have four years of subjective experience with the Flogiston chair. In that time we have learned that:

1. The modified Neutral Body Posture is valid in a one gravity environment by providing the optimum minimal stress posture for relaxation, concentration and focus.
2. The flexibility of the hull is a positive feature which adds to the comfort of the occupant.
3. The dynamic motion of the chair is perhaps the most stimulating feature of the chair.
4. The modular hull with various suspensions is the most versatile approach. This makes the chair adaptable to many applications.

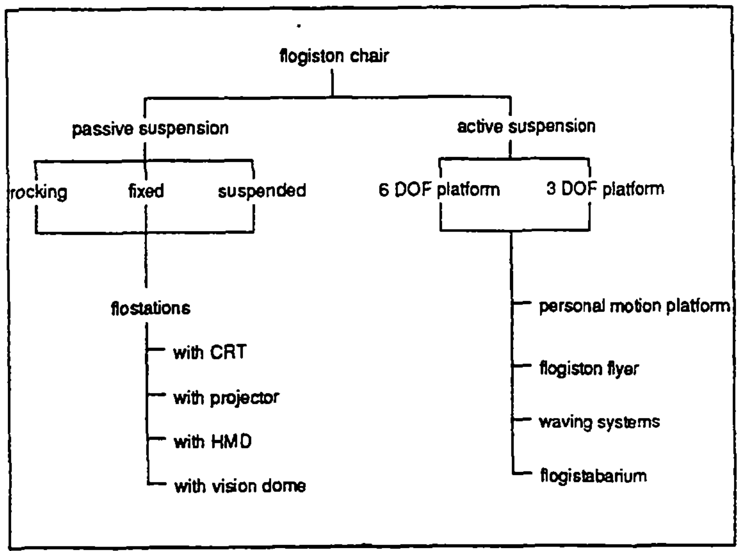


Figure 2. **Flogiston chair family tree**

A family of environments based on the chair is being developed. See Figure 2. For example, a flostation is a chair system that integrates the tasks of work, rest, play and learning. A flostation that uses an LCD projector is shown in Figure 3. This provides a suitable environment for multimedia learning, amongst others.

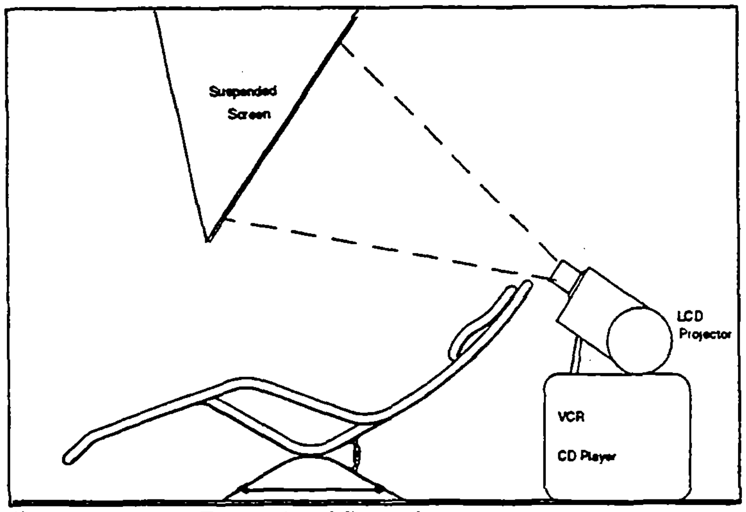


Figure 3 Example of flostation

We believe that the posture is also fundamental to the exploration of Cyberspace. Immersion in a synthetic environment requires the reduction of the awareness of the real world as much as possible to provide the illusion of being in an alternate reality. This should include the awareness of the effects of gravity on the physical body. By supporting the body in the minimal stress posture of the chair, the subjective gravity effects are reduced, and the occupant soon feels that they are floating in Cyberspace. This state is naturally conducive to micro gravity simulation.

The occupant reclines with their back at 30° to the horizontal, so that their normal perspective is also reclined. Thus the x,y, and z coordinates of the virtual world are also inclined. When the occupant moves forward in Cyberspace, the chair moves upward against gravity to enhance the feeling of movement. The orientation of Cyberspace with reference to real space can readily be adjusted in software.

In normal operation the occupant rests his hands on the chair arms so that his hands fall naturally on the motion controls. Flogiston proposes that the controls should be designed to support the hand in the neutral hand posture to minimize potential carpal tunnel effects. When the occupant wants to reach out to touch a virtual object, he raises his arms above his body in real space, to where they would normally be in zero g, which is less fatiguing than holding the arms out horizontally against gravity.

The chair is protected by a US utility patent granted in 1992. Any structure which supports the body similar to the modified neutral body posture infringes the patent.

Motion Platform

Flight simulators have used motion platforms since their early development in the 1940s. Motion enhances the realism of the experience and improves the acceptance of the simulation. Three degrees of freedom systems for pitch, roll and heave, and six degrees of freedom systems for pitch, roll, yaw, surge, sway and heave are available as standard products.

The addition of motion to VR systems is a new concept, but is as fundamental to immersion in Cyberspace as it is with flight simulators. For simulating EVA applications, the platform will simulate the dynamics of working while floating in micro gravity with a sensitivity that neutral buoyancy tank simulation is unable to reproduce, due to the viscosity of water dampening the small but significant inertia effects. For example, the reactionary forces applied to the astronaut when he contacts a structure could be fed back as motion to the platform so that he feels the experience as much as sees it. This could be as beneficial as tactile feedback.

The System

The personal motion platform combines the advantages of the modified neutral body posture with the dynamics of a motion system for an individual. Flight simulators are large, complex and expensive systems, whereas the personal motion platform is relatively simple, compact and inexpensive. By modeling the complete environment in VR, including the occupant's immediate surroundings, little external structure is needed except to support the body in the right posture and provide motion.

The system, illustrated in Figure 4, consists of a monocoque hull which supports the body in the modified neutral body posture, mounted to a small six degree of freedom motion platform. Controllers are mounted to each arm of the chair at the optimum location and orientation. Alternate hardware which can be added to the chair include a zero mass HMD support, a motion sensor stand, and a body restraint harness. The design of the hull allows fasteners to be mounted at any point on the under surface for the attachment of additional equipment.

A suitable 6 degree of freedom platform has been identified. It consists of a hydraulically servo actuated motion base assembly, servo controllers, hydraulic power supply and a digital control system. The control system consists of an Intel 80486 and two TMSC30 digital signal processors. This interfaces through Ethernet to the VR processor for control. Real time motion control is performed by the control system. Multi level redundant safety mechanisms are utilized in hardware and software to ensure that the system will function safely.

An alternate gas spring/linear motor motion platform with 3 degrees of freedom (DOF) has also been identified as a candidate for a low cost system suitable for game or home use.

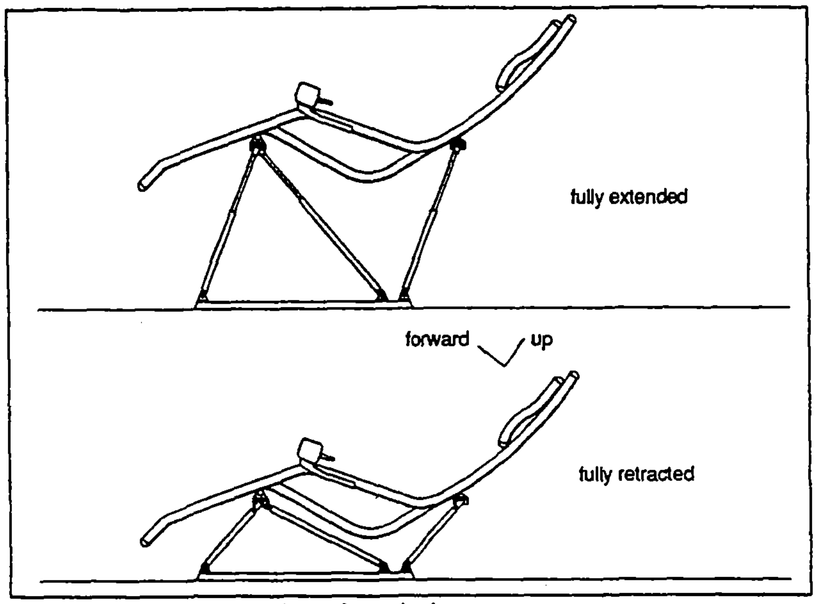


Figure 4. The personal motion platform.

The system provides:

1. Reproduction of the neutral body posture familiar to astronauts, thus increasing the fidelity of the simulation.
2. An extremely comfortable posture which enables the astronaut to remain in simulation for extended periods with minimal physical fatigue.
3. Direct feedback of accelerations in 6 degrees of freedom.
4. A complete environment for generic cyberspace immersion and control.

Operation

A block diagram of the system is shown in Figure 5.

The system operates as follows:
1. The virtual reality processor contains a math model of the synthetic environment created by the customer. The environment dynamics are computed, with resultant body axis accelerations in the six degrees of freedom. For example, accelerations can be derived from the controls on the chair, or from collision detection with virtual objects. These computed accelerations, which are results of the math model, should match what the actual environment would produce in real life.

2. The acceleration signals are passed from the virtual reality processor via Ethernet or direct connection to the PMP processor. These signals are fed into Motion Drive Laws which are basically adaptive filters that convert the accelerations to position degree of freedom commands and limit the excursion of the motion platform to prevent the pistons from ending against their stops. The Motion Drive Laws consider the frequency response of the human, the size of the motion system and the type of the vehicle. Kinematics calculations convert the degree of freedom space to leg length space. These signals are converted to analog voltages and sent to the six channel servo controller mounted on the platform base.

3. The servo controller sends the signals to the servo cylinders which provide positional feedback to correct the position. The self contained hydraulic power supply provides the power to cylinders. This results in motion of the Flogiston chair.

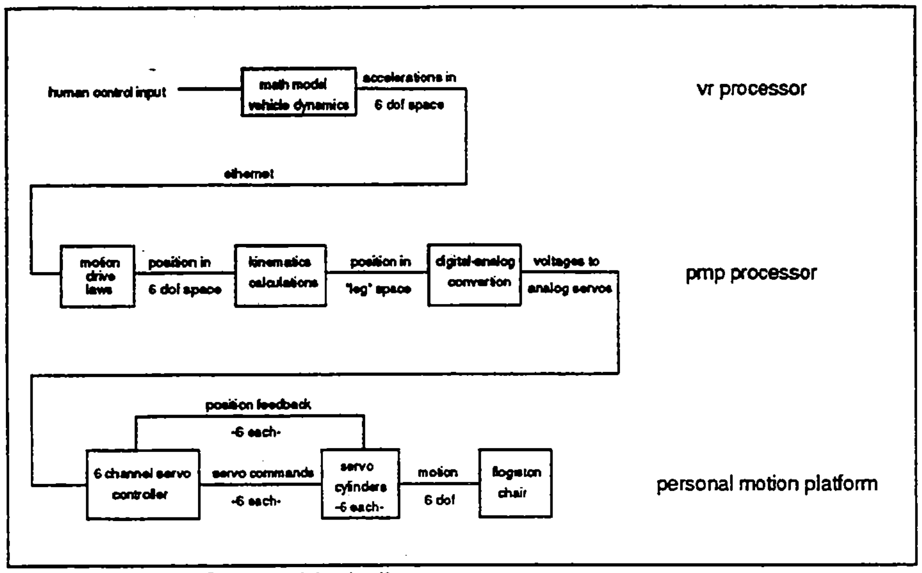


Figure 5. System block diagram.

Performance

See Table 1 below:

	Excursion (minimums)	Maximum Velocity	Max. Acceleration	Acceleration Onset
Pitch	± 18 °	± 20 °/s	± 60 °/s^2	± 100 °/s^3
Roll	± 19 °	± 20 °/s	± 60 °/s^2	± 100 °/s^3
Yaw	± 21 °	± 20 °/s	± 60 °/s^2	± 100 °/s^3
Vertical	± 7 in	± 20 in/s	± 0.5 g	± 1 g/s
Longitudinal	± 9 in	± 20 in/s	± 0.5 g	± 1 g/s
Latitudinal	± 7 in	± 20 in/s	± 0.5 g	± 1 g/s

Table 1. PMP Performance

The velocities and accelerations quoted are for a 2000 lb. payload design. The chair plus occupant typically weigh less than 350 lbs, so that the platform will be able to respond much faster if needed. The excursions can also be changed with choice of alternate cylinder lengths.

Frequency response: 90 degrees of phase lag at 1.5 Hz.

Smoothness: Less than 0.06 g's with one actuator exercised sinusoidally.

Actuator Acceleration Instability: Less than 0.06 g for any static position.

Servo Actuator Crosstalk: Less than 10% between any two actuators.

Servo Actuator Position Drift: Less than 5% over a 6 hour period.

Applications

The personal motion platform provides the sensation of movement in virtual reality for research, biomedical studies, training, entertainment, health, and stress management. In this fashion it is a generic cyberspace motion simulator that can be applied to any application where movement can enhance the immersion experience.

For micro gravity simulations, the reclining axis, posture and low resistance movement provide a sensitive platform for simulating EVA work on Space Station Freedom, satellite servicing and retrieval, MMU simulation, and so on. The platform could also be used to simulate work inside Space Station Freedom to verify workstation layouts, serviceability of racks, workspace volumes, etc.

Additional applications for the PMP may include:

1. Research into the effects of micro gravity on the body. If the phase lag between control and response is purposely increased motion sickness could be induced, measured and in theory, reversed.

2. Preflight conditioning the astronauts to micro gravity. Programs could operate the chair cyclically to raise and lower the chair continuously and set up a wave motion that affects the inner ear for example. HMD visuals could enhance the experience.

3. Post flight reconditioning of the astronauts to a one G environment after 90 to 120 days on station. An exercise regimen while lying in the chair could accelerate response to one G.

References

1. B.N. Griffin. Design Guide. The Influence of Zero-G and Acceleration on the Human Factors of Spacecraft Design. NASA/Johnson Space Center 1978.

2. NASA-STD-3000. Space Station Man-Systems Integration Standards. Vol. IV. NASA 1985.

3. Yogiraj Sri Swami Satchidananda. Integral Yoga Hatha. Holt, Rinehart and Winston 1970. pages 84-85.

Issues in Capturing and Representing Domain Knowledge and Polygonal Assignment for Virtual Environments

Tim Saito and R. Bowen Loftin (UH-D)
LinCom Corporation
Mail Code PT4
NASA/Johnson Space Center
Houston, TX 77058

Depending on the application (medical, simulation, games, etc.) and its presentation style, the caliber of knowledge directly affects the quality of a virtual reality (VR) experience. Virtual environments require comprehensive, diverse knowledge components for their creation and maintenance. Two perspectives within VR, suspension of disbelief and viewer-centered perspective, affect the relationships in which the domain knowledge is presented and vice versa.

The development of virtual environment challenges "traditional" knowledge acquisition (KA) methods and strategies. Knowledge components would be infused into graphical objects, trees (object hierarchy based on parent-child relationships) and polygons (object capacities). Creating environment rules graphically using object oriented icons related to domain specific or meta-level knowledge is explained. Body and physical representations demand a visual manifestation of the viewer and pose unique quandaries for KA. We will postulate on the balance of knowledge types and their roles in creating the essence of the domain of an artificial world.

Next, the issue of knowledge representation and polygonal assignment will be touched on. Finally, we will suggest methods in which knowledge could possibly be acquired and organized to operate within a virtual environment.

N96-14979

A Knowledge Engineering Taxonomy for Intelligent Tutoring System Development*

Pamela K. Fink, Ph.D. and L. Tandy Herren, Ph.D.
Southwest Research Institute
6220 Culebra Road
San Antonio, TX 78228
(210) 522-3762

ABSTRACT

This paper describes a study addressing the issue of developing an appropriate mapping of knowledge acquisition methods to problem types for intelligent tutoring system development. Recent research has recognized that knowledge acquisition methodologies are not general across problem domains; the effectiveness of a method for obtaining knowledge depends on the characteristics of the domain and problem solving task. Southwest Research Institute developed a taxonomy of problem types by evaluating the characteristics that discriminate between problems and grouping problems that share critical characteristics. Along with the problem taxonomy, heuristics that guide the knowledge acquisition process based on the characteristics of the class are provided.

1.0 INTRODUCTION

Practical experience in knowledge-based systems indicates that how a system is designed and implemented depends on the characteristics of the task that the system is to perform or teach. Researchers in artificial intelligence are beginning to analyze the relationship between knowledge representation, system architecture, and task type. Such work is related to a continuing focus in the psychological and educational literature on the relationship between task characteristics and instructional strategy in curriculum development and on the relationship between task characteristics and human factors engineering.

The growing focus on task characteristics in knowledge-based system development signals the maturation of the field from one searching for the single best representational format and inferencing strategy to one that recognizes the diversity of problems and the need to address each according to its characteristics. Not surprisingly, human cognition appears to utilize multiple representational formats and reasoning techniques. Different representations store different types of information (e.g., scripts store simple, well-structured sequences of events while rules store single-level inferences) and dictate appropriate methods for reasoning about that information. The flexibility to employ the best and most appropriate representation and reasoning strategy in a given situation undoubtedly contributes to a human's ability to effectively solve problems.

In the development of an intelligent tutoring system (ITS), domain expertise of the appropriate kind and level for teaching must be implemented in the expert module to be used by the instructional portion of the system for training. This essentially constitutes the design and implementation of a specialized knowledge-based system that contains the knowledge and problem solving skills that must be taught to the student. The development of an ITS is equivalent to the development of several very different expert systems that all must work together, requiring extensive effort on the part of experts from all of the fields concerned. Because of the extensive effort involved in ITS development, it is crucial for researchers to evaluate the nature of the task to be taught and to pair it with an appropriate knowledge acquisition strategy.

Due to the amount of knowledge that must be embedded in ITSs, they can be very expensive to build. They require many hours of knowledge engineering in which the task is characterized and the knowledge is collected for task performance and for instruction. Then, the design of each of the four major modules needs to be developed and finally implemented. Depending on the size and complexity of the task to be taught, this extensive process can take many person-years of effort to complete. Some work has been done to reduce the amount of effort required to build an ITS. In most cases, these are software tools that support the design and implementation process by providing frameworks and authoring facilities for the various components of an ITS. Such tools work much the way a knowledge-based system development tool works for the development of knowledge-based systems. They reduce the effort required to write the code. However, they provide little direction on how to go about acquiring the knowledge that must be implemented into the code.

To acquire the knowledge, the ITS or knowledge-based system developer must use either a traditional verbal or an automated method of knowledge acquisition. The process of knowledge engineering has long been marked as a bottleneck in any knowledge-based system development. This is not surprising since the process involves humans who are expert in a given domain communicating with other humans who are not expert in the domain about their expertise and problem solving skills. The knowledge engineering experts are, however, expert in computer programming and knowledge representation and they must be able to organize, structure, and convert the domain expertise as they understand it into a computer program. All of this communication results in a situation similar to the "telephone game," where something is lost or misinterpreted with each communication act. The problem, of course, is magnified by the fact that the individuals involved do not tend to speak the same "language" and the concepts being communicated can be quite complex. A good, simplified description and illustration of the knowledge engineering process is given in Yost and Newell (1989).

The knowledge acquisition problem has been acknowledged since the early days of knowledge-based system development. Because the expert module of an ITS is essentially a knowledge-based system, ITS development suffers from the same difficulty with knowledge acquisition. Many person-years of effort go into the development of a single knowledge base. The ideal solution, of course, is to skip the knowledge engineer and have the domain expert generate the knowledge base directly. The problem with this approach is that the domain expert usually has no real knowledge or experience in computer programming and knowledge representation. Research into the development of tools to support the knowledge acquisition process has been in two directions: 1) to support the knowledge engineer in structuring and encoding the knowledge acquired and 2) to provide an interface that would allow the domain expert to enter his/her knowledge directly into the computer program. However, the knowledge engineering process remains very time consuming and expensive.

2.0 FOUNDATION FOR THE NEXT GENERATION KNOWLEDGE ACQUISITION TOOLS

Completely automated knowledge acquisition tools that allow a domain expert to generate a sophisticated, complex knowledge-based system with no support from an individual knowledgeable in computer programming and artificial intelligence is a lofty goal. Current understanding of human learning and cognition is not at a level to allow the development of completely general, domain independent automated knowledge acquisition tools capable of developing arbitrarily complex knowledge-based systems in any domain. In addition, most domain experts do not have the time or inclination to submit themselves to extensive, tedious question-answer sessions with a computer program. Another approach to the knowledge acquisition problem that leaves the human knowledge engineer in the loop might be a better short-term solution.

It has been readily acknowledged among practitioners that the knowledge engineering process, as performed currently, is more an art than a science. Of course, there are techniques with names attached to them such as structured and unstructured interviews, example cases, etc., and there are recommended practices such as taping the interviews, generating transcripts, and then editing and modifying the resulting knowledge acquired based on domain expert feedback. However, the quality of the resulting system is still highly dependent on the personalities and skills of the individuals involved. The key to a successful knowledge engineering process (i.e., a useful, working knowledge-based system) is the ability to extract the general problem solving methodologies utilized and the specific domain knowledge needed to find a solution to a given problem.

Thus, the goal of the research reported on in this paper was to develop a methodology that could be used to typify the expert problem solving strategies and the types of specific domain knowledge needed that would allow the knowledge engineer to categorize the task. This initial categorization could then provide further methodologies for understanding the key characteristics of the given kind of task. These techniques could eventually be automated, but initial experience with, and evaluation of, the techniques should be gained through the use of good human knowledge engineers. The approach to generating such methodologies and classes was to study a broad spectrum of problem solving tasks, analyze the problem solving techniques used, and pair knowledge engineering techniques to the acquisition and codification of specific problem solving strategies needed in an ITS. The result of this process is a proposed taxonomy of problem solving tasks characterized by problem solving strategies and types of domain knowledge, as well as some recommendations concerning knowledge acquisition methodologies relevant to the particular task.

Recently researchers have become more and more interested in basing their selection of a knowledge acquisition technique on the characteristics of the task. The ultimate goal is to find a way to attenuate the well-known knowledge acquisition bottleneck. Tailoring the knowledge acquisition process to the task will reduce the problems associated with eliciting expertise.

For example, Chandrasekaran (1985) describes six generic problem-solving tasks in knowledge-based reasoning, including classification, state abstraction, knowledge-directed retrieval, object synthesis by plan selection and refinement, hypothesis matching, and assembly of compound hypotheses for abduction. These tasks correspond to problem solving methods that can be combined to perform knowledge-based reasoning for an application. Bylander and Chandrasekaran (1987) suggest that generic tasks can be associated with a specific knowledge acquisition methodology.

Boose and Bradshaw (1987) proposed a set of knowledge acquisition strategies that link with appropriate problem-solving methods. The knowledge acquisition strategies establish distinctions between alternatives, problem decomposition, combining and propagating information, testing of knowledge, combining multiple sources of knowledge, incremental expansion of knowledge, and providing process guidance. However, Kitto and Boose caution that a knowledge acquisition system must be able to acquire knowledge about aspects of the problem-solving process that are unique to a given application.

McDermott (1988) presented a preliminary taxonomy of problem-solving methods based on the assumption that there are families of tasks that share abstract control knowledge and that the abstract control knowledge provides strong guidance as to what knowledge is required and how that knowledge should be encoded. He refers to the control methods as role-limiting methods because they strongly guide knowledge collection and encoding, i.e., task specific knowledge fills one of a few number of roles within the control framework. McDermott argues that it is likely there are hundreds of role-limiting methods. He discusses the following role-limiting methods and knowledge acquisition tools:

- cover and differentiate - implemented in MOLE
- propose and refine - implemented in SALT
- qualitative reasoning - implemented in YAKA
- acquire and present - implemented in KNACK
- extrapolate from a similar case - implemented in SIZZLE

What researchers in psychology and artificial intelligence have neglected to specify is how to determine in the first place whether or not an application task involves a certain characteristic. How do you identify a diagnostic task? How do you know if the task requires declarative or procedural knowledge? Or both? Does it matter? This issue is not entirely self evident but is, in fact, the reason that knowledge acquisition remains largely an art. Good knowledge engineers have little difficulty discerning the fundamental requirements of a task, but there is no formal method for making these judgments and no indication of how accurate they are once the judgments are made. The research reported on this paper explored these issues.

3.0 RESEARCH APPROACH TO DEVELOPING A KNOWLEDGE ACQUISITION TAXONOMY

The approach taken in this research was a very pragmatic one. The work was based on years of experience in performing knowledge acquisition for the design and implementation of numerous knowledge-based and intelligent tutoring systems in a wide variety of problem solving domains. One of the underlying goals has been to minimize the amount of time required with the expert. Experience has shown that expert time is usually very limited and it helps to make the most of the time that you do have. Another underlying goal of the research was to develop an approach to knowledge acquisition that would help a knowledge engineer characterize the problem solving task in some useful way and to scope the effort. Thus, much of the questioning is oriented towards acquiring knowledge of the inputs to the task, the result or outputs from the task, reasoning mechanisms that operate on the input and generate the output, the environment in which the problem solving takes place, and the attributes of the experts who perform the task. The effort was less concerned with total accuracy of the knowledge obtained than with discovering the problem solving approach(es) used. Accuracy can come later when the software is under development and the expert can observe the system's explicit behavior and suggest concrete corrections.

The research proceeded by identifying a set of representative tasks, interviewing experts in the selected tasks, classifying these tasks into a taxonomy based on their characteristics, and generating recommendations for performing knowledge acquisition based on the task classification. In some sense, the work to develop a taxonomy needed to be done before we could make sense of the results gained from a series of knowledge acquisition interviews. But, at the same time, the knowledge acquisition interviews supplied the information on tasks that could support the generation of a taxonomy. Thus, because this was an initial research project, a bootstrapping approach had to be taken. Therefore, two interviews were performed for each problem solving task included in the study. This allowed us to gain some insight into the problem solving tasks, generate some hypotheses and classification schemes, and then informally test these hypotheses and classifications through a second interview. The following sections describe the four main phases of the research that resulted: 1) performing the first interview, 2) analyzing the results of the first interview, 3) performing the second interview, and 4) analyzing the results of the second interview and developing the taxonomy.

3.1 The First Interview

In developing the first interview, three major aspects had to be addressed. First, a set of problem solving tasks had to be selected for analysis. Second, a means by which each task could be characterized and rated had to be developed so that a comparison could be made between tasks. Third, a knowledge acquisition approach had to be identified/developed that could be used to acquire the information on each problem solving task. Each of these aspects of the approach is addressed below.

In selecting a set of problem solving tasks for analysis in this research, the goal was to find a representative set with as much variety as possible. The following list provides the task types that we felt needed to be represented in the set selected for interview.

- diagnostic task
- training task
- high performance task
- form fill-out
- managerial/supervisory task
- design task
- planning task
- monitoring with a time factor
- perceptually-oriented task
- bin-packing/np-complete task (exponential/combinatorial growth problem)
- numerical task
- data intensive (no real time factor), e.g., acquire and present

As a result, we interviewed experts in the following areas:

- (diagnostics) medical diagnosis
- (diagnostics) equipment diagnosis
- (training) training pilots
- (training) training a foreign language
- (high performance) flight controller console operations
- (high performance/knowledge-rich) surgery

·**(form fill-out)** contracting
·**(people-oriented)** personnel management/leadership training
·**(design)** software design
·**(planning)** acquisition program management
·**(monitoring/time constrained)** air traffic control
·**(perceptual)** weather forecasting
·**(bin-packing)** cargo loading
·**(numerical)** accounting
·**(data intensive/no time constraint)** DRAIR generation
·**(planning)** scientific protocol design

Obviously, these tasks were still rather broad and we did not intend to try to perform knowledge acquisition on the entire area indicated by the job label. Instead, we planned on allowing the individual expert's specific job requirements to narrow the scope in each problem solving area. Individuals considered experts in the selected areas were identified and asked to take part in the effort. Only one expert in each area was selected. The focus of the research was on task type, not on how numbers and types of experts could alter the knowledge acquisition process.

To generate a set of task characteristics to support the rating of tasks, a literature search was performed in both the psychology and artificial intelligence fields to identify existing approaches to problem solving taxonomies. This search helped to identify a list of 123 characteristics that fell into 16 different categories on which a particular problem solving task could be rated. This list was intended to be relatively exhaustive by including all the attributes that might be relevant to how problem solving tasks could be categorized. We suspected that some redundancy existed, and that some of the characteristics would not prove to be relevant for the problem of knowledge acquisition. However, we wanted to be sure that as much as possible was considered when analyzing a problem solving task.

The of characteristics compiled for this study originated from a compilation of existing taxonomies (Gawron, Drury, Czaja, and Wilkins, 1989). The list of characteristics fell into 15 categories:

Reasoning Techniques -	characteristics related to the manipulation of information during task performance
Problem-Solving Techniques -	search strategies for finding a solution
Inputs -	data input to the problem solving process
Task Complexity -	characteristics of the data array and problem solving space that influence the task
Technical Dimension -	the degree to which the task requires a technical background or skill
Motor Processes -	physical requirements of the task
Information Processing -	characteristics of the way data is manipulated during task performance
Problem Solving Tasks -	cognitive operations on the data
Recall -	the role of memory in the task
Perceptual Processes -	the perceptual requirements of the task
Environment -	physical and psychological characteristics of the environment in which the task is performed
Personal Characteristics -	characteristics of the experts required to perform the task
Type of Domain -	knowledge-rich and/or high performance
Hardware/Equipment/Tools -	items or devices used during the task
Communication Processes -	verbal and nonverbal communication required during task performance

We selected these categories and characteristics because of their relevance to knowledge acquisition and system development. We felt that some of the characteristics would be related to how knowledge acquisition should be conducted and others would be more related to how an intelligent system for the task should be designed.

The technique used in performing the initial knowledge acquisition for each task involved the use of a structured interview. Based on our experience in knowledge acquisition, we believed it to be the most general approach to acquiring knowledge when little was yet known about a particular task. As a result, a series of questions was generated directly from the list of task characteristics. These questions were designed to elicit information that would help to discern the importance of the various task characteristics to the task under consideration.

These questions provided a guide to ensure that we obtained information about each of the 123 characteristics during the interview. We framed the questions such that they were understandable to a lay person and grouped and ordered them into a logical progression from general requirements and inputs, to data processing requirements and outputs. This interview usually required approximately one and a half hours to perform and two researchers performed the interview together. All interviews were conducted at the expert's workplace.

3.2 Evaluating the First Interview

Based on the interview performed with the human experts, two raters reached agreement on the ratings for each task on the 123 attributes. The raters used a 5 point Likert-type scale in which 0 indicated the attribute is not relevant at all for the task and 4 indicated that the attribute is critical for task performance. The ratings were used as a measure of the importance of a characteristic for task performance.

The major goal of the data analysis was to develop a principled way of classifying tasks based on the knowledge acquired from the first interview and to make recommendations concerning how to proceed with the knowledge acquisition process based on this classification. The data generated from the first set of interviews consisted of a 16 by 123 matrix with a rating of zero to four for each of the 123 characteristics for each of the 16 tasks. We wanted to answer a number of questions concerning this data, including what characteristics are relevant for a meaningful classification of the tasks, which tasks are related based on their characteristics and which were not, and what the key characteristics are about a class of tasks that would affect the way we would want to proceed with the second interview.

3.2.1 Statistical Analyses

The ratings of the characteristics entered into a factor analysis and cluster analysis. Some characteristics were dropped from the analysis because the ratings for all 16 tasks was 0. Additionally, we collapsed the scores across a category of characteristics if a summary score for that category made sense. For example, a summary score for the category "inputs" reflects the degree to which the task requires external information. A summary score for the category "reasoning techniques" would not produce a meaningful index.

A principle components factor analysis with a varimax rotation was performed on the characteristic ratings. The factor analysis revealed three main factors in the characteristics. The first factor consisted of characteristics related to design tasks, the second factors consisted of characteristics of tasks involving a physically-based skill, and the third factor corresponded to characteristics of tasks involving statistical reasoning and mental modelling. These factors were evident in the task clustering. In a cluster analysis of the tasks, the first cluster, software design and protocol design, includes the design tasks we examined. The second cluster, pilot training and surgery, are tasks that require physically-based skills. The relevance of the third factor is less clear. Protocol design, medical diagnosis, weather forecasting, and drair generation require statistical reasoning while software design, weather forecasting, console operations, surgery and medical diagnosis require the use of mental models. This factor does not clearly map to the cluster analysis.

3.2.2 An Analysis Based on Experience

The issue of whether or not a task is formally trained was not clearly delineated in the characteristics by which we were rating each task. However, it was clear from our initial interview how an expert became an expert in the field and we found ourselves recommending an examination of the curriculum for the second interview for any task that was formally trained in the operational environment. As a result, we categorized the tasks by whether or not an individual received formal training in performing the actual task(s) that constituted their area of expertise. The tasks fell into two relatively equal sets where aircraft piloting, air traffic control, console operations, weather forecasting, cargo loading, foreign language, surgery, and medical diagnosis all are heavily trained before an individual is allowed to perform the task, and where equipment diagnosis, form fill-out, program management, DRAIR generation, software design, leadership, accounting, and protocol design are not formally trained. Tasks appearing in the latter set tend to require at least a general background education in a particular field such as business, computer science, or operations research as a foundation on which they can build expert skill through experience on the job.

Once these two sets of tasks were generated, we then examined the other characteristics of the tasks to see if there were any unifying themes. Among the formally trained tasks the unifying characteristics seemed to be an issue involving human safety, as well as quite often a physical component. They were also highly proceduralized, even if the experts did not necessarily follow a procedure once they became expert. Individuals entering the field are taught a procedure to follow that allows them to become efficient problem solvers. Tasks that were not formally trained tended to be less well-defined in terms of goals and results. Such tasks do not tend to have definitive right and wrong answers and problem solving tends to be oriented towards breaking the task down into relatively independent subtasks. This approach helps to deal with the complexity and inexactness of the problem.

In addition, in those tasks that involved a procedure the skills required for the task tended to build sequentially on one another. So for example, when a person learns to fly a plane they begin by learning how to fly level, then learn to take-off, then learn to land. Each skill builds on the previous skill. However, the skills required in the set of tasks not formally trained tended to be componential. That is, the problem solver has a toolbox of skills relevant for different aspects of the problem solving process. They learn each of the skills independently. For example, the expert in protocol analysis had skill in experimental design, statistical analysis, research methodology, and electronics. These are independent skills and are learned during a formal education in a content area, such as biomedical engineering. Based on these observations, we labeled the first set of tasks procedurally-oriented tasks and the second set we call componential tasks.

3.2.3 Results and Conclusions from the First Interview

When all of the various analyses of the data from the first interview were compared, we began to see patterns in the data that confirmed our experiential analysis. For example, the eight cluster statistical analysis generated the following clusters:

- pilot training, surgery
- software design, protocol design
- cargo loading, accounting, program management, leadership, equipment diagnosis, DRAIR generation, form fill-out
- medical diagnosis
- foreign language training
- air traffic control
- weather forecasting
- console operations

The cluster analysis corresponds greatly to our informal analysis. The first cluster, software design and protocol design, and the large third cluster primarily contain tasks that do not receive formal training in the operational environment, such as program management, accounting, and drair generation. However, these tasks require a lot of background knowledge often obtained through formal education. The remaining clusters are tasks that receive a large amount of training in the operational environment. Surgical training stems from years of internships and residencies, air traffic control involves training specific to each airport at which the controller works, and console operations requires both a formal educational relevant to the specific console and continual on-the-job training to remain proficient.

In addition, the factor analysis of the characteristics indicated that the clusterings were due to differences among the design, high performance, and model-based/statistical factors that roughly correspond to the attributes we saw intuitively as being key to the commonalities that were responsible for the generation of the two sets of tasks.

This information was used to help guide the work in generating the approaches used in the second set of interviews.

3.3 The Second Interview

Once a clustering of the tasks became apparent, we examined the implications for the second interview for each task. However, when we compared what we believed we should do for the second interview based on our own intuition versus what seemed appropriate based on the clusterings, we saw that the clusterings did not have as much impact as expected on the desired approach. Based on this assessment, we determined that the second interview is still too early in the knowledge acquisition process for much delineation to take place in terms of how to proceed with the knowledge acquisition task. For example, even though a wealth of information is available through existing curriculums for those tasks that are formally trained, we sometimes felt that the second interview was too early to effectively utilize the information. In most cases, we believed that going through an example in some form would be the most appropriate approach to the second interview. The differences among tasks was exploited mainly in how the second interview should elicit the example and how many examples should be examined.

The major exceptions to the use of an example were foreign language training, software design, program management, and leadership training. Foreign language training was an exception because an example is almost meaningless, so examination of a component of the curriculum was recommended. Software design and program management were exceptions because the tasks are so large and ill-defined at their highest level that an example would have little meaning even if it could be formulated. Thus, we selected one of the components of software design and of program management on which to focus further discussion. This would allow an iterative and principled approach to breaking the task down into sub-tasks until a task of a workable size was found. Leadership training was an exception because it was unclear how an example could even be formulated from which a discussion could evolve. Because there are a number of tools that are used in leadership training (TQM) we chose to examine one of those tools, namely team building.

One minor exception to the use of examples occurred with aircraft pilot training. For this task, the desired approach was to examine an example, but we wanted to examine the easiest examples, preferably from the first few flights that a student pilot would take. In this case a well-defined curriculum existed, but examples would be most useful in furthering our understanding of the task. Of course, they teach the students to perform the task, as well, through the use of well-selected examples.

In the cases where the second interview should consist of eliciting an example, the differences among tasks was exploited mainly in how the second interview should elicit the example and how many examples should be examined in that second interview. In general, if the task is data intensive, then a sequence of examples that built on increasing data complexity was used. In cases where the task included a strong procedural component, the example was used to provide structure, but general rules were expected as part of the outcome of the interview.

3.4 Evaluating the Second Interview

Once a second interview for a task was performed, we again rated the tasks along the set of characteristics. However, this set consisted of 124 characteristics because we broke the training characteristic used in the initial ratings into general education vs. specific training in the operational environment for each of the sixteen tasks. This was due to the impact of a formal training approach on the knowledge acquisition process.

The second rating was performed independently of the ratings given after the first interview. This allowed us to look at how much the ratings changed from one interview to the next, thus providing some indication of how our impression of each task changed. Based on the ratings given for the 124 characteristics for each of the sixteen tasks, the same analyses were performed on the characteristics matrix to see if any significant changes occurred in how the tasks clustered based on the second interview.

The factor analysis revealed three main factors in the questionnaire. The first factor consisted of characteristics associated with tasks involving a physically-based skill, having perceptual requirements, and requiring spatial and temporal reasoning. The second factor corresponded to knowledge rich tasks involving no perceptual requirements or environmental/psychological stressors. The third factor was related to tasks involving means-ends heuristic search in a data-driven fashion.

As in the first set of analyses, these factors were related to the clustering solution in the cluster analysis. Air traffic control, surgery, and pilot training as well as console operations and medical diagnosis have physical skill and perceptual requirements. The remaining tasks can be characterized as primarily knowledge-based and involving no perceptual requirements or environmental/psychological stressors. The relationship of the third factor to the clustering solution is less clear. Many of the tasks (e.g., medical diagnosis, console operations, and weather forecasting) are data-driven and require some means-ends analysis (e.g., program management and pilot training).

The clusters are fairly similar to those that emerged from the first ratings. Cargo loading, accounting, program management, form fill-out, and drair generation clustered in both solutions as did pilot training and surgery. Weather forecasting and foreign language training never clustered with other tasks. In the second clustering solution, medical diagnosis and console operations clustered, which was not surprising because they both involve diagnosis. There was inconsistency between the cluster from the original ratings of software design and protocol design and the cluster from the second ratings of software design and leadership training. We determined that software design and protocol design are very different tasks. Protocol design involves a knowledge of experimental and statistical methods and is much more formalized than software design. Software design is a true design task, involving decomposition and propose/refine and generate/test problem solving strategies.

Also, the second clustering solution confirmed our contention that two main categories of tasks exist: those formally trained and those not. Three clusters that constituted air traffic control, surgery, pilot training, console operations, medical diagnosis, and weather forecasting include the formally trained tasks. With the exception of cargo loading, the tasks in the other five clusters are not formally trained.

4.0 RESULTS AND CONCLUSIONS

Based on the results of the second interview, the delineation between tasks that are formally taught and those that are learned more or less on the job remained from the first to the second interview. Task clusterings remained very similar from the first to the second interview, though the characteristics themselves were not always rated the same across the tasks. When examining the amount of knowledge we felt that we acquired from an interview and the ease with which it was acquired, the fact that a task was explicitly trained played a tremendous role. Those individuals who had taught, or were teaching, the task were much better prepared to talk about the task in an organized and explicit fashion. They were also much more capable of generating examples and explaining them since this is something that they have had practice in. When the task is not formally trained, the knowledge acquisition process is much more dependent on the skills of the knowledge engineer to guide the interview and make sense of the information collected. This is not always possible with only a couple of hours of discussion with an expert. However, it is also important to be able to detect when the expert does not know enough about the area of interest and to try to find another expert. We had two experts in this effort that, if we were tasked with actually building a tutoring system, would need to be supplemented with other experts more involved with performing the actual task on a day-to-day basis.

Based on this experience, we concluded that, in general, if the task has a distinct starting and stopping point and the task is well-defined, then in the second knowledge acquisition session with an expert, the knowledge engineer should try to go through at least one example. This example should be selected from a continuum of easy to hard, starting with easier ones. What makes the task easier or harder should be identified up front and explained. There are a number of refining conditions as follows:

- If the problem solving task extends over more than a couple of hours, then you should break the task down further into sub-tasks and examine them before proceeding to an example.

- If the task is distinctly procedural, then the example should be used only as a way of eliciting the general procedure and following how it is applied to a specific situation. There is no point in getting specific information on an example and then having to try to generalize from it when the generalization exists in the expert's head.

·If the task is very data intensive, then a summary of the relevant data elements, where they come from, and how they are used should be discussed before the example is started.

·If the task uses a complex interface such as a console, cockpit, or set of tools, then these should be discussed in terms of their components and their functions before going into an example.

·If the task is formally taught, then the examples should be selected from the early part of the curriculum and discussed as they progress to more complex ones.

·If the task is ill-defined, but a curriculum exists, then use the curriculum to guide the interview process.

·If the task is ill-defined or takes too long to go through an example, then the interview process should continue to explore the task area to determine subtasks that can be examined further.

Thus, the recommendation for the second interview is heavily example-based. However, a word of caution is in order. The examples that we discussed with the experts were not usually easily reduced to a series of attributes and values that could be entered into a knowledge acquisition tool for generalization into rules. The examples often had many aspects and were quite complex. In addition, the examples were always used as a way of structuring the interview, they were not the sole source of knowledge from the interview. The experts provided many insights into the relevance of various attributes and their effect on the problem solving task. In addition, there was often a procedure implicitly used within the process that would not necessarily be apparent just from a few examples or even many hundreds of examples. Thus, just a collection of examples from which we would generalize would be a highly inefficient and ineffective way to acquire knowledge for building a tutoring system or a knowledge-based system. Use of examples can provide a much richer medium for obtaining knowledge than simply the collection of attributes and values as they relate to a solution.

The problem solving taxonomy to support knowledge acquisition developed as a result of this research appears in Figure 1. The first few levels of the taxonomy, including how the task was learned by the expert, the complexity/data requirements of the task, and the continuity with which the task is performed, are areas not addressed previously in any other research. Other attributes of tasks, such as procedural orientation and construction vs. classification, have been addressed by other researchers. However, we believe these latter attributes only affect the knowledge acquisition process in the later stages of an intelligent systems development effort.

This research provided an efficient and effective method of approaching the first few interviews in the knowledge acquisition process. The total time required of an expert in this methodology for the first two interviews ranged from two to four hours. A knowledge engineer using the initial interview and classification scheme should be able to readily classify the task according to the hierarchy. This should provide some heuristics for conducting follow-on interviews, selecting tools, and selecting problem solving methods. In addition, many of the characteristics assessed by the initial interview, while not related to knowledge acquisition, provide input to other design decisions in the development of an ITS or knowledge-based system.

This study generated a number of hypotheses about how to conduct knowledge acquisition for a variety of tasks and therefore provides some direction in how knowledge acquisition should proceed. However, the hypotheses need to be verified through more formal experimentation than was possible in this initial effort.

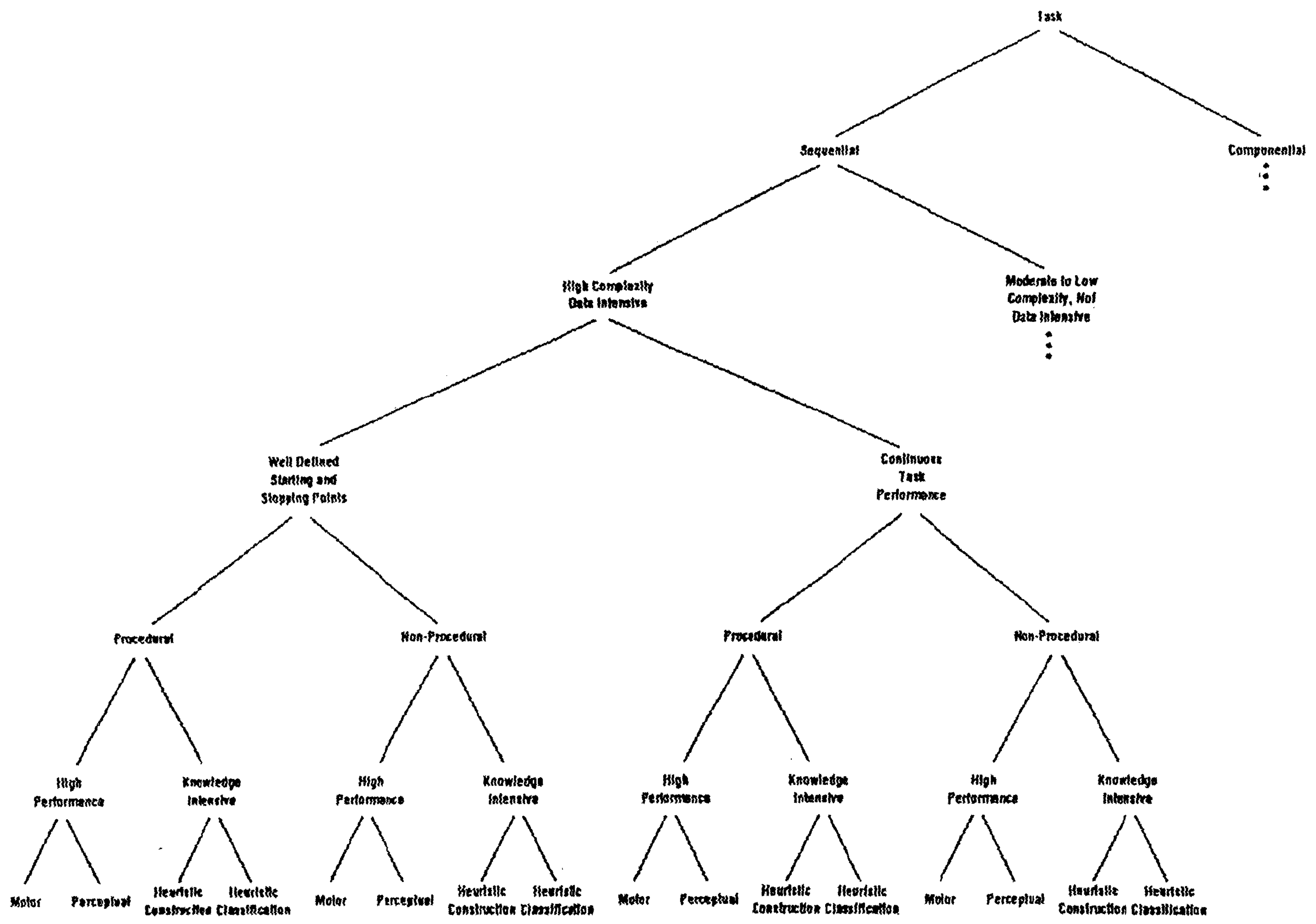


Figure 1. The problem solving taxonomy to support knowledge acquisition

REFERENCES

(1) Boose, J. H. and Bradshaw, J. M. (1987). Expertise transfer and complex problems: Using AQUINAS as a knowledge acquisition workbench for expert systems. International Journal of Man-Machine Studies, 26, 21-25.

(2) Bylander, T. and Chandrasekaran, B. (1987). Generic tasks for knowledge based reasoning: the "right" level of abstraction for knowledge acquisition. International Journal of Man-Machine Studies, 26, 231-244.

(3) Chandrasekaran, B. (1985). Generic tasks in knowledge based reasoning: Characterizing and designing expert systems at the "right" level of abstraction. Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach Florida, New York: IEEE Computer Society, 294-300.

(4) Gawron, V. J., Drury, C. G., Czaja, S. J., and Wilkins, D. M. (1989). A taxonomy of independent variables affecting human performance. International Journal of Man-Machine Studies, 31, 643-672.

(5) McDermott, J. (1988). Preliminary steps toward a taxonomy of problem-solving methods. In S. Marcus (Ed.), Automating Knowledge Acquisition for Expert Systems. New York: Kluwer Academic Publishers.

(6) Yost, G. and Newell, A. (1989). "A Problem Space Approach to Expert System Specification," Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 621-627.

Generic Expert System Shells and Apprenticeship Tutoring: The Representation and Use of Explicit Control Knowledge'

David C. Wilkins
University of Illinois
Department of Computer Science
405 North Mathews Avenue
Urbana, IL 61801
wilkins@cs.uiuc.edu

Our goal is to advance the state-of-the-art of second generation expert system shells for heuristic classification so that they can effectively support the teaching of the domain knowledge base via apprenticeship tutoring. Heuristic classification expert systems are very widespread, encompassing most extant expert systems, and are used throughout science, engineering and defense in applications such as data interpretation, monitoring and diagnosis.

This paper describes the design of the Minerva expert system shell, a descendent of Neomycin; it shows how its design facilitates critiquing the problem solving of a student in terms of the knowledge base for an application domain, such as airplane or medical diagnosis. A key aspect of Minerva is an explicit representation of three levels of knowledge: an application domain level, a strategy level and a scheduler level.

Our most recent advance is an explicit representation of the scheduler control level, whose purpose is to determine the relative plausibility of the alternative problem-solving actions at a given point in time. Our results have shown that in order to follow the line-of-reasoning of a student, the amount and complexity of the knowledge at the scheduler level is equal to that of the application domain level. Further, we have shown that the needed scheduling knowledge can be organized and semi-automatically learned using a technique called recursive heuristic classification. The Minerva shell recursively calls itself each problem-solving cycle to solve the important scheduler control problem.

The Impact of Cognitive Task Analysis as a Knowledge Acquisition Method in Intelligent Computer Assisted Training Systems

David B. Palumbo (UHCL), David Germain (MDC), and Will Lidwell (Nimbus Technologies, Inc.)
Advanced Knowledge Transfer Laboratory
University of Houston-Clear Lake
2700 Bay Area Blvd., Box 263
Houston, TX 77058
713-283-3565 713-283-3599
IN%"PALUMBO@CL4.CL.UH.EDU"

Design and development of ICAT systems are tied to the successful representation of expert understanding and performance in complex performance domains. As the cognitive sciences yield a more fully developed understanding of the underlying processes involved in performing these functions, a shift from behavioral task analysis (which focuses primarily on procedural knowledge) to cognitive task analysis (which also addresses declarative and strategic knowledge) has occurred. This shift is important as we begin to develop more complex instructional systems that attempt to provide the learner with knowledge that may not be readily broken down into observable behaviors.

Cognitive task analysis, which addresses both the overt and covert knowledge required to complete a particular task, holds a key position in both the research and application of intelligent instructional systems. There are currently a variety of methodologies that attempt to extract and represent the knowledge and processes (declarative, procedural, and strategic) that are required for successful completion of complex tasks.

This session will attempt to distill the most salient features of a variety of cognitive task analysis methodologies and present work in progress at The Advanced Knowledge Transfer Laboratory to come up with a more effective and efficient means of extracting and representing expert knowledge in the declarative, procedural, and strategic domains for use in ICAT system development.

The Learn, Explore and Practice (LEAP) Intelligent Tutoring Systems Platform Multimedia Integration

Charles Bloom, Brigham Bell, Frank Linton & Edwin Norton
U S WEST Advanced Technologies
4001 Discovery Drive, Suite 270
Boulder, CO 80303
cbloom@advtech.uswest.com

ABSTRACT

This paper describes the prototype LEAP (Learn Explore and Practice) intelligent tutoring systems (ITS) platform being developed at U S WEST Advanced Technologies (AT) to train U S WEST customer service representatives (CSRs) how to sell telecommunications services and products. Domains of this type require the CSR to simultaneously carry on a conversation with the caller, manipulate database applications to find out information about the caller and enter information regarding service registration, and look up information about availability and service capabilities from reference documentation located on their desks. These services and products that the CSRs sell (and the information about them that the CSR must be facile with) are updated frequently, producing an ongoing learning problem in order to "stay current." In addition, telecommunications companies are highly regulated, and CSRs must be knowledgeable about and comply with all federal and state regulations under penalty of law. LEAP employs two interrelated, instructional modes: Procedural instruction, where students can exercise their customer interaction skills, and declarative instruction, where students can review services information. In procedural training mode, students work through typical customer interactions in a tutoring environment that simulates their actual working environment. Their goal is to handle some customer request regarding VMS, or transition a non-VMS call to a VMS sales opportunity. In declarative training mode, students can study multimedia lessons (e.g., animations, video segments, etc.) on information prerequisite to specific types of customer interaction skills. In addition, students can navigate between modes during instruction based on the topic currently being studied or practiced. Instruction in LEAP is student initiated. However, LEAP does assess student performance and uses that assessment to: Make recommendations about what to study or practice next, determine how to apply its different instructional methods, initiate interventions during procedural training sessions, and provide student performance feedback at the end of a procedural session. In addition, LEAP allows instructional designers to adjust LEAP's instructional and student modeling parameters to further individualize the delivered instruction. The first application of the LEAP ITS Platform is teaching CSRs how to operate and sell residential Voice Messaging Service (VMS).

INTRODUCTION

Intelligent tutoring systems are dynamically organized "intelligent" instructional programs that employ independent representations of domain, instructional and student knowledge to enable them to provide individualized instruction much like that provided by a personal human tutor. The goal of an Intelligent Tutoring System (ITS) should be to provide real-time, context-appropriate and cost-effective training that enables learners to perform appropriate domain tasks in the right manner and at the proper time. An ITS that is able to achieve this goal should produce a decrease in the time required to migrate learners from novice to expert, while increasing the number of trained personnel successfully reaching a more knowledgeable level. Unlike conventional computer-based training (CBT) programs that deliver instruction statically and uniformly, an ITS can be:

- Proactive — actively teach difficult and abstract concepts and skills
- Reactive — guide students in exploratory learning in a simulated environment or microworld
- Adaptive — tailor training scenarios to the student's learning progress

An ITS accomplishes this by employing a basic architecture that consists of the four interacting components depicted in Figure 1:

- Domain Model — a representation of the knowledge to be tutored to the student, also used as the standard for evaluating the student's performance (Anderson, 1988; Wenger, 1987).
- Instructional Model — a representation of the knowledge of how to tutor the student (i.e., the instructional methods to be employed in the tutor and how they are employed) (Halff, 1988).
- Student Model — a dynamic representation of the student's knowledge state (VanLehn, 1988).
- User Interface — the communication channel between tutor and student (Burns & Capps, 1988).

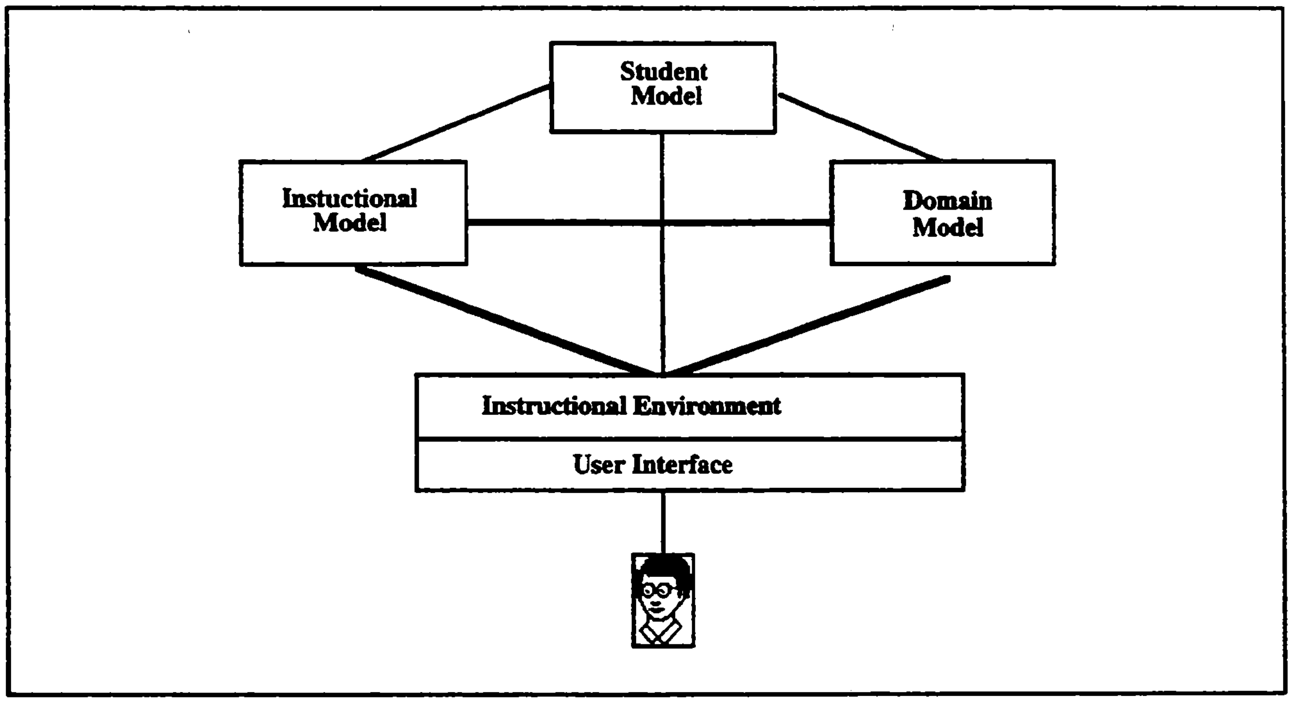


Figure 1. Traditional ITS Architecture

The domain selected for the first application of the LEAP ITS Platform is teaching U S WEST Home and Personal Services (H&PS) customer service representatives (CSRs) how to operate, sell and register customers for residential Voice Messaging Service (VMS). This domain was selected because it is representative of a broader class of domains that includes the sales and service of US WEST products across business units. This choice of domains makes it possible to identify portions of the ITS that are reusable both for extending its scope to cover additional business units, and for developing LEAP-based ITSs for other products and services across business units.

VMS is an Enhanced Service Product offered by U S WEST that answers incoming calls placed to the subscriber when the called number is busy or if the called number does not answer. (Enhanced Service Products are products that are optional, competitive, and not regulated.) VMS allows subscribers to record their own personal greeting or else use a prerecorded greeting. When messages are waiting, the subscriber is alerted by a special "stutter" dial tone. Messages can be retrieved either from home or away from home by calling a special access number using a push-button phone and entering a personalized security code. A representation of the high-level tasks related to VMS that a CSR performs is presented in Figure 2.

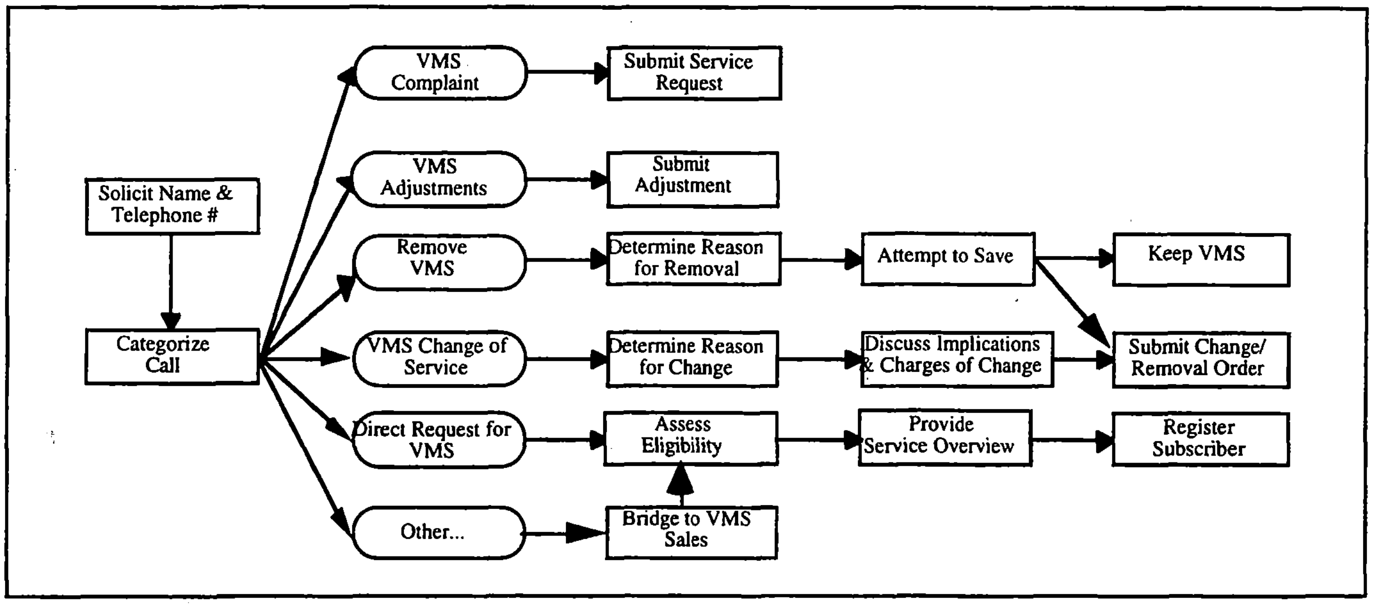


Figure 2. VMS Related Tasks

The VMS sales process begins with the CSR asking the caller's name and telephone number, which is then entered into a database (via a computer workstation) to pull up the customer's account information. At this point, the experienced service representative will engage in a short conversation with the customer to determine the nature of the call. In the VMS domain, calls can be categorized into one of three types: (1) Calls from a customer who already has VMS, (2) calls infolving a direct request for VMS, or (3) calls that have nothing to do with VMS.

There are basically two types of calls á CSR can receive from a customer who already has VMS: Calls in which the customer is complaining about VMS service or billing, or calls in which the customer is requesting some change to the service. For the first type of call, the CSR's task is to initiate the appropriate work order. For the second type of call, the CSR's task is more complicated involving:

- Determining the customer's reasons for wanting a change or removal of service
- Discussing with the customer the implications of these changes
- In the case of a request for removal of service, attempting to save the sale
- Initiating the change or removal if the customer still wants it.

In the case of calls that have nothing to do with VMS, the CSR must look for or make an opportunity to transition the conversation into a discussion of VMS (or some other product or service) capabilities and benefits. To do this, the CSR needs to have skills in the areas of:

- Evaluating customer data (either from credit information, the current conversation, or information volunteered by the customer) for VMS selling clues
- Transitioning the conversation to VMS sales
- Making selling points (i.e., explaining benefits and features)
- Countering specific customer responses.

Once the caller becomes interested in the service, or in the case where a caller is making a direct request for VMS, the CSR then assesses the compatibility with the subscribers other services and equipment, and discusses the legalities of selling VMS. Following that, the CSR might go over the specifics of VMS, and finally, go through the process of registering the subscriber for VMS.

During the process, the CSR has to simultaneously (1) carry on a conversation with the caller, (2) manipulate certain databases and applications on their computer terminal to find out information about the subscriber and enter information regarding VMS registration, (3) look up certain information about availability and service capabilities from reference documentation located on their desks, and (4) prepare mailings of VMS information for the caller.

An integral part of the CSR's job in selling and registering customers for VMS is interacting with the Service Order Negotiation and Retrieval (SONAR) application. SONAR is the primary system used by CSRs for collecting customer information and initiating orders for phone service. SONAR is a mainframe application that CSRs access over an internal U S WEST ethernet local area network (LAN) via terminal emulation software on their desktop workstations.

In an attempt to conceptualize and represent these real-world conversations in a manner that is usable in the LEAP Tutor, we developed an abstract representation of these conversations in terms of a task grammar. The basic unit of the task grammar is the situation-action (SA) pair. Situations can be either customer statements, requests or questions, or database application output or configurations. Actions can be either responses by the CSR to customer statements or to application information, commands or data entered into the application by the CSR, or actions focused around information processing and decision making. All conversations are made up of sequences of SA pairs. Conversations are grouped together to reflect different types of scenarios the could occur between caller and CSR (i.e., "New Connect," "Change of Service," etc.). Branches within conversations are based on customer situations (e.g., callers equipment or type of service, etc.). SA pairs that are conceptually related are referred to as sub-topics. Sub-topics are reusable portions of conversations that can appear in several different conversation scenarios. Figure 3 contains an example of a task grammar representation.

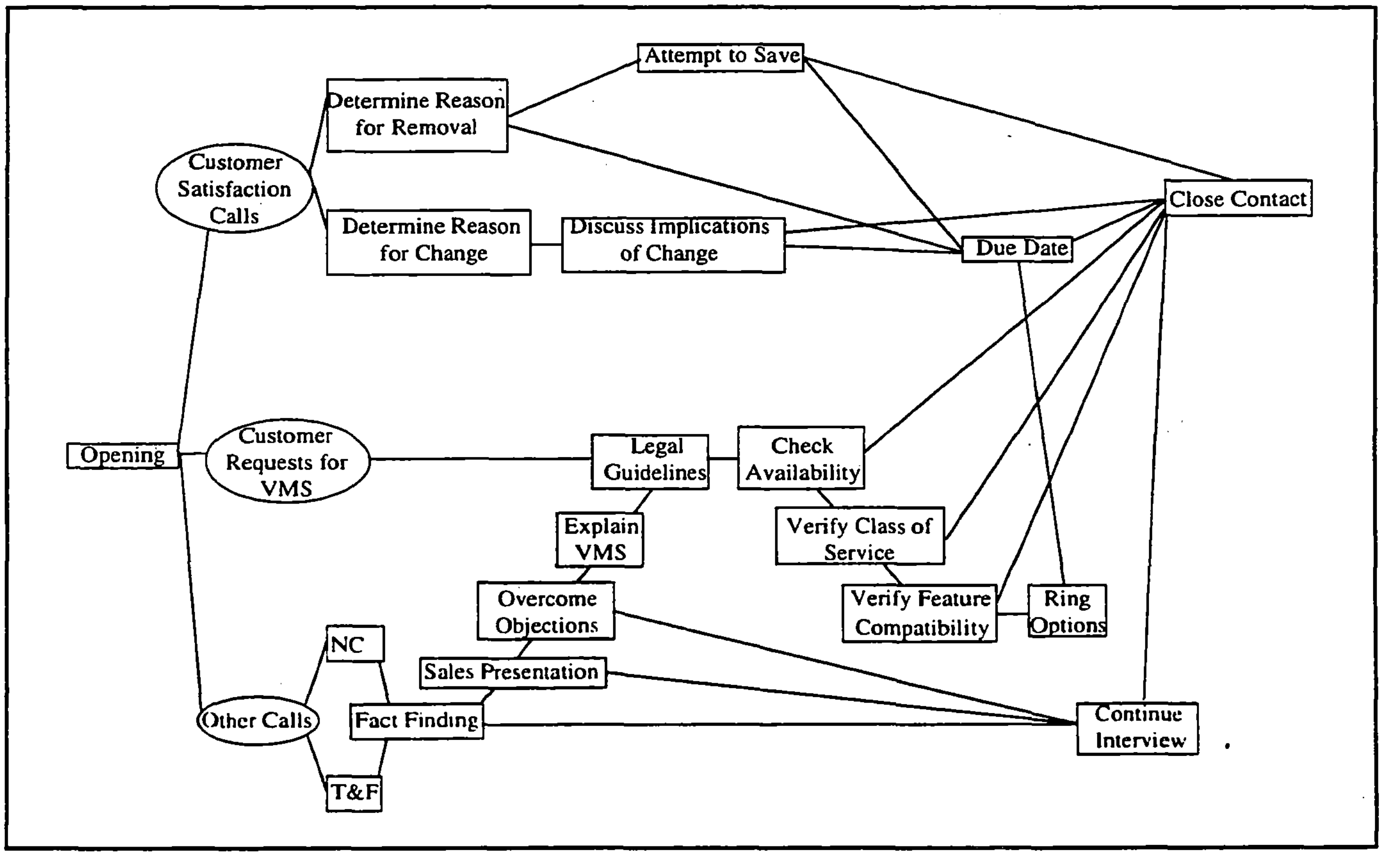


Figure 3. Example Task Grammar Space

The Task Grammar Space can be conceived of as a network representation of all of conversation sub-topics, and the interconnections between those sub-topics observed in the operational environment. In other words, the task grammar is a representation of the complete collection of possible customer-CSR conversations that LEAP will support. Each conversation scenario (i.e., VMS Change of service conversations, Direct request for VMS on new connect, etc.) is made up of several sub-topics (and their associated SA pairs) of a customer-CSR conversation. Each sub-topic contains a specification of all allowable SA pairs (including identified conversation errors).

Conversations or scenarios can be constructed by combining all of the individual sub-topics along any one branch (from left to right) of the Task Grammar Space. For example, the scenario of servicing a customer's direct request for VMS could involve some or all of the following: Opening, Legal Guidelines, Check Availability, Verify Class of Service, Verify Feature Compatibility, Ring Options, Dues Dates and Close Connection. The specific sub-topics employed would depend upon "customer" responses or constraints (i.e., if the service is not available in the customer's area, the scenario would contain only Opening, Legal Guidelines, Check Availability and Close Connection).

LEAP TUTOR FUNCTIONAL ARCHITECTURE

This section describes the instructional modes that the LEAP Tutor employs to teach CSR's how to perform the VMS sales and service related tasks discussed previously. Figure 4 contains a conceptual diagram of LEAP Tutor.

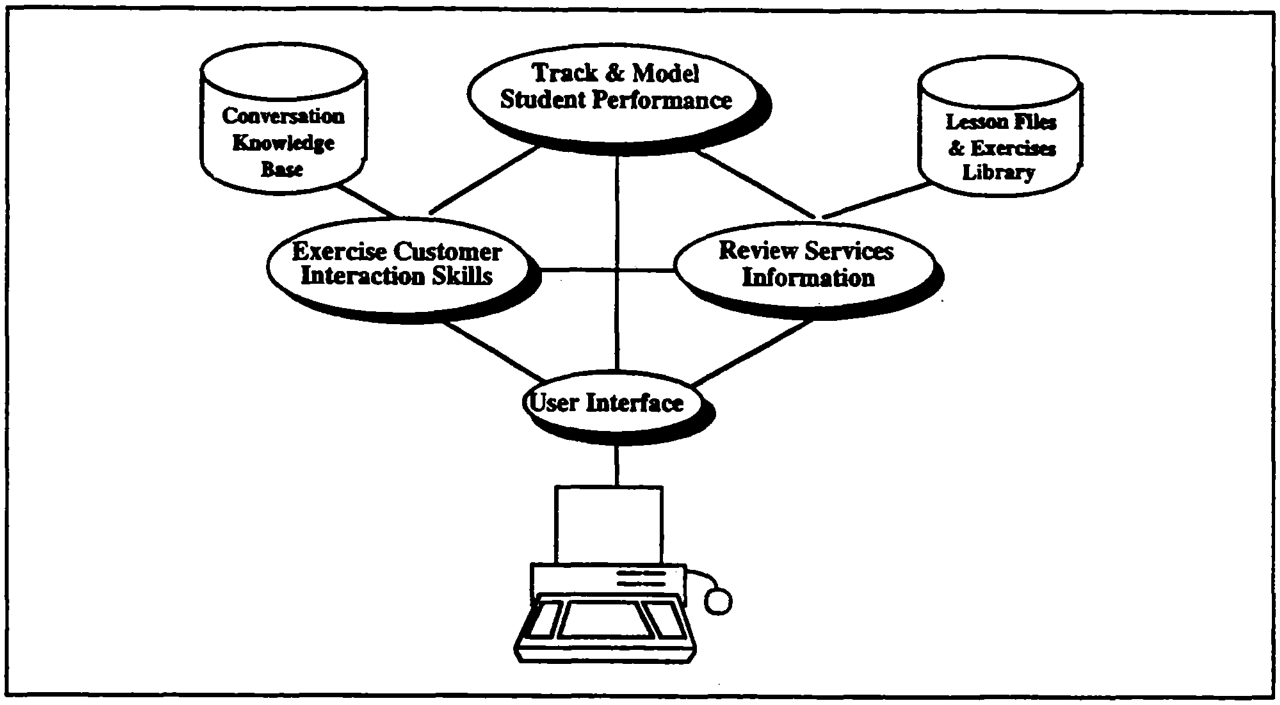

Conversation Knowledge Base
Track & Model Student Performance
Lesson Files & Exercises Library
Exercise Customer Interaction Skills
Review Services Information
User Interface

Figure 4. LEAP Tutor Conceptual Diagram

The LEAP Tutor employs two major instructional modes: Review Services Information (RSI) and Exercise Customer Interaction Skills (ECIS). In RSI mode the student can study interactive multimedia lessons on information prerequisite to VMS, or else perform study exercises on the topic selected. In ECIS mode, the student works through typical conversations (or else individual parts of conversations) where the goal is to handle some customer request regarding VMS, or else transition a non-VMS call to a VMS sales opportunity. Instruction in the LEAP Tutor is student initiated (i.e., the student selects both the mode, and the topic, grammar or scenario to study or practice). However, the LEAP Tutor will have the capability for tutor initiated recommendations (i.e., the student can request LEAP to recommend a topic, grammar or scenario for the student to study or practice) should the student want LEAP to take on that responsibility.

The LEAP Tutor also has the capability to assess the student's performance, in both instructional modes, and to use those assessment results to:

- Provide an introspective evaluation of the student's skill level on each topic and sub-topic (as listed in the LEAP Tutor's top-level topic list) on a five point scale ("Untried," "Needs Practice," "Almost," "Good," and "Excellent")
- Make recommendations about topics or scenarios to study or practice
- Initiate instructional interventions
- Provide student performance feedback at the end of the session.

The LEAP Tutor's top-level topic list provides the organizational structure for both RSI lessons and ECIS conversations. Therefore, the topic list provides explicit links between information in the two modes enabling the LEAP Tutor to:

- Recommend conversations to practice based on lessons just studied or exercises just practiced (and the student's performance on those exercises)
- Recommend lessons to study based on conversations just rehearsed (and the student's performance on those conversations)
- Support dynamic transitions between conversations (or parts of conversations) and appropriate lesson materials during ECIS practice (and vice versa for RSI study)

The LEAP Tutor recommends topics to study or practice based on a consideration of three factors: Topic sequence, relation to last topic studied, and user's topic-level proficiency. After recommending a topic, the LEAP Tutor then recommends a method of studying the topic: Either "Review Services Information," or else one of the "Exercise Customer Interaction Skills" sub-methods. The algorithm the LEAP Tutor uses for selecting a study method to recommend is as follows: First, see a demonstration of the topic in use in ECIS mode. Next, study the topic in RSI mode. Then, practice applying that knowledge in simulated conversations in ECIS mode (the types of simulated conversations will discussed in the next section). Finally, take a test on that topic (i.e., perform a conversation without the tutor's assistance). In addition, students can elect to perform drill and practice exercises on the declarative materials, or examine the flow of particular customer/CSR contacts.

The instructional objectives of the LEAP Tutor are consistent with several features of the minimalist approach to training and learning (Carroll, 1990):

- Teach people what they need to learn in order to perform their jobs. In the LEAP Tutor, most training takes place in an environment that closely emulates their real world job environment, and the tasks they perform on the tutor are representative of tasks they will perform on the job.

- Allow learners to start immediately on meaningful and realistic job tasks, and allow them to work on those tasks in any order. In the LEAP Tutor, the student can decide what they want to do and when to do it. Although the LEAP Tutor has the capability to recommend instruction, the student has the choice of whether to go along with that recommendation.

- Keep the amount of passive instruction (i.e., reading) to a minimum. Even though there are close to twenty topics covered in the LEAP VMS Tutor, only three of those topics, those covering prerequisite information necessary to support task performance, have declarative lessons. In addition, the declarative lessons are presented in interactive, multimedia formats to increase the level of activity or involvement of the student in the lesson.

- Training materials should explicitly support the recognition and recovery from error, both to make the learning materials more robust and complete, and to train learners in error recovery skills. The LEAP Tutor contains explicit training on errors and error recovery.

In the subsections that follow, we describe each of the LEAP Tutor's major functions.

Review Services Information

The Review Services Information (RSI) mode of the LEAP Tutor supports the delivery of declarative study materials in interactive multimedia formats, and drill and practice exercises on the topics identified as prerequisite to operating, selling and registering customers for VMS.

In RSI mode, the student selects a topic from the LEAP Tutor's top level list of topics. Not all topics have multimedia lessons or exercises. However, all topics do have an introduction and an overview. In those topics that do have multimedia lessons, those lesson materials are developed using multiple media (i.e., text, audio, graphics, photographs, animations and videos) as appropriate.

The lessons have been produced to a grain size that supports entry into the entire lesson at a topic level, or entry into parts of the lesson as they apply to parts of conversations. This second capability is particularly useful in enabling students to review information from part of a lesson to support their conversation rehearsals.

For example, prior to commencing any practice sessions, a student is advised to view the three prerequisite declarative lessons. One of those lessons deals with legal issues. The sub topics of that lesson include: Enhanced Service Product Guidelines, Consumer Protection Law Requirements, Anti-Trust Requirements, FCC Requirements, and U S WEST Full Disclosure Requirements. When viewing the lesson in RSI mode, the student is able to interactively view the lesson, selecting what to review and in what order to review them. However, the tutor does recommend they review ALL lesson materials.

On the other hand, suppose a student is practicing a conversation and gets to a point where they are to discuss the legal ramifications of selling competitive enhanced services products. If that student is unsure of exactly what these legal ramifications are, they may selectively review that portion of the "Legal" lesson that deals with this topic by pressing the related information button. Once they have reviewed as much of this information as they want or need, they can return to the conversation and continue where they left off.

The Study Exercises function presents students with one of four types of drill and practice exercise on the topic selected or studied. These drill and practice exercises are integrated with the LEAP Tutor's student model to help in the selection of items and for updating the model in terms of correct or incorrect responses. The types of drill and practice exercises are as follows:

- *Checklist Flashcards* — The student is presented with a series "clues" with accompanying questions and potential answer checklists about that clue. At the end of the series the student reviews their performance as compared to the tutor's "experts."

- *Audio Flashcards* — The student is presented with a series of auditory "cues" (i.e., prerecorded customer statements) to which they record a reply. At the end of the series the student reviews their performance as compared to the tutor's "experts."

- *Application Flashcards* — The student is presented with a series of application (i.e., SONAR) screens specifically configured to contain pertinent account information. The student must respond to specific "Yes-No" questions about that account information, as well as indicated additional account characteristics interactively on the application screen.

- *Question and Answer Flashcards* — The student is presented with a series of questions to which they must type a response into a specific input field. At the end of the series the student reviews their performance as compared to the tutor's "expert."

Exercise Customer Interaction Skills

The ECIS mode of the LEAP Tutor employs two different ways of exercising customer interaction skills: Rehearsing Conversations or Examining the Contact Flow.

Rehearse Conversation

The LEAP Tutor's Rehearse Conversation mode supports students working through conversations representative of problems or tasks faced on the job. As students work through the conversations , the LEAP Tutor will provide feedback and hints when students have difficulties. At the end of each scenario, the LEAP Tutor will provide summary feedback.

Rehearse Conversations is the most complex part of the LEAP Tutor's instructional process. The Rehearse Conversation instructional environment and philosophy is similar to that employed in the SHERLOCK ITS developed at the University of Pittsburgh's Learning Research and Development Center (Lesgold, Lajoie, Bunzo, & Eggan 1992). In SHERLOCK, students interact with an accurate emulation of their real-world environment. The objective is to produce a situated learning environment in which students can acquire and exercise their knowledge and skills. However, SHERLOCK is a non-directive tutor -- it does no proactive tutoring. However, the domains for which the LEAP Tutor was developed does require some proactive tutoring.

As a model of proactive tutoring for the LEAP Tutor's Rehearse Conversation mode, we have chose to employ an apprenticeship approach to teaching the skills and knowledge necessary for competent domain performance. The apprenticeship approach involves using methods like observation, coaching and successive approximation rather than didactic teaching (Collins, Brown & Newman, 1989). In addition, apprenticeship approaches embed the learning of skills and knowledge in their social and functional context.

There are four distinct study methods employed in the Rehearse Conversation mode: Demonstrate, Critique, Accompany and Practice. However, since Practice is the most complex method, with many of its features employed in the other methods, we will discuss Practice first.

Practice. The Practice method is the primary learning method employed in the LEAP Tutor. In practice, the LEAP Tutor proceeds through an entire conversation, with the tutor playing the role of customer, and the student playing the role of CSR. Conversations can be characterized by sets of situation-action pairs.

- Situations:
  - — Customer statements, requests or questions (Verbal Situations)
  - — Database Application (SONAR) output or configurations (Operational Situations)
- Actions:
  - — Responses by the CSR to customer statements, requests of questions, or to SONAR information (Verbal Actions)
  - — Commands or data entered into SONAR by the CSR (Operational Actions)
  - — Actions focused around information gathering, information processing and decision making (Cognitive Actions)

Within a Practice session, the LEAP Tutor employs four specific study methods as defined by Collins et al. (1989) to facilitate learning: Skimming, Scaffolding, Fading, and Feedback (there is also a feedback method employed at the completion of a conversation).

Skimming takes place when the LEAP Tutor has determined that the student already knows a specific situation-action pair. To skim means that the tutor presents both the situation and its appropriate action, so that the student need only click on the action to continue the conversation. The application of skimming is probabilistic: When the situation-action pair is new to the student, he or she will always have to perform it. When the situation-action pair is known to the student, most of the time it will be skimmed. However, occasionally, the student will be asked to respond. This element of unpredictability contributes to maintaining the student's attention to the task as a whole.

Scaffolding takes place when some material is unknown to the student and not part of the current focus of instruction. In scaffolding (as in skimming and demonstrate), the tutor presents both the situation and its appropriate action and the student needs only click on the action to continue the conversation. Scaffolding also affords the student the opportunity to observe the tutor perform activities beyond their current capabilities and begin forming a model of that part of the task.

Skimming and scaffolding make it possible to support a situated learning model: Students can practice specific skills or parts of conversations in the context of entire conversations. In addition, the tutor does not waste time making the student either re-do tasks that are already well known, or else have difficulty with tasks that they are not yet capable of performing.

The third within practice study method employed in the LEAP Tutor is Fading. Fading consists of the gradual removal of "supports" until the students are on their own (Collins et al., 1989). In the LEAP Tutor, fading is used to support learning of situation-action pairs where the situation has no explicit features. For example, before selling VMS to a customer, CSRs are expected to remember to tell the customer that this is a competitive service available from other providers. There is no explicit situation to stimulate this action - the CSR must remember to make this "competitive service disclosure" on their own (a legal requirement). To support learning of responses to implicit situations, the LEAP Tutor will initially present them an explicit situation, which it gradually "fades," ultimately requiring the student to perform this action from memory as they would have to in actual conversations.

The fourth within practice study method employed in the LEAP Tutor is feedback. As the student works their way through a conversation, if their action is correct, the conversation continues. If, on the other hand, their action is incorrect, the LEAP Tutor provides them with a hint and allows the user a second try before supplying the student with the correct action and continuing the conversation.

As an example of a practice session, conversations begin with LEAP initiating a customer call which the student hears over a headset similar to the kind used on the job. The student then responds, prompting another customer statement and so on until the conversation is complete. Verbal responses are input as follows: First the student responds verbally (the LEAP Tutor records their responses for playback/feedback purposes). After indicating to LEAP that their verbal response is complete, the student is then presented with a menu of pre-defined responses and instructed to select the one that is closest to their verbal response. These responses are abstract representations of possible CSR responses (e.g., "Ask caller's name").

When the response selected by the student is incorrect, the LEAP Tutor provides them with an immediate intervention (a hint) and then allows them to attempt to select the correct response. If their second response is also incorrect, the tutor supplies them with the correct response and then continues with the practice session.

At times, instead of a verbal response, the student is expected to take some action in the database application (SONAR). This they do directly, with the LEAP Tutor monitoring, evaluating and responding to their actions. When the student's action is correct, the application monitoring software passes their input to the host application and continues the practice. When the student's action is incorrect, the application monitoring software notifies the student that their answer was incorrect, provides them with a hint, and recommends they try again. After a second incorrect action, the application monitoring software provides by the student and the host application with the correct action and continues the conversation.

There are four important points to note about practicing conversations in the LEAP Tutor:

- Customer statements are heard rather than read (emulates the "job" environment)
- CSRs verbal responses are recorded for use in end-of-conversation feedback/reviews
- Menu selection is employed as a work around our inability to parse spoken natural language
- The application students interact with in LEAP training IS the same application they will use on the job.

Demonstrate. The Demonstrate method is similar to Collins et al.'s "Modeling" method. The tutor shows the student how a task is performed, demonstrating both sides (customer and CSR) of a VMS-related contact. This includes all verbal statements made by either the customer or the CSR, the cognitive actions (strategic decisions) made by the CSR, and all database application actions (SONAR). The student observes the demonstration and builds a conceptual model of the task, in this case, a telephone contact with application use.

When a student observes a conversation using the Demonstrate method, they receive "credit" with the student model even though they did not do any of the conversation (or application) interactions.

Critique. The Critique method (still under development) is envisioned to be similar to Collins et al.'s "Articulate" method. The goal is to get the student to "use" their knowledge by articulating reasons for specific actions or decisions. The tutor presents a conversation (as in Demonstrate), and the student is assigned the task of monitoring that conversation for specific activities (thereby articulating their knowledge). These activities could be errors made during the conversation, or specific strategies used, or even specific actions taken.

During the Critique exercise, if the student correctly notes the occurrence of some item, the tutor provides them with positive feedback and continues with the exercise. If the student fails to note the occurrence of some item, or if the student notes an incorrect item, the tutor provides them with an appropriate hint and lets them try and identify the correct item. If the student gets the item correct on this second try, the receive positive feedback and the exercise continues. If the student fails to note the correct item on this second try, the tutor provides them with the correct answer before continuing the exercise. It is important to note that the student receives feedback on the critique items, not conversation items in Critique exercises.

At the end of a critique exercise, the student goes through a feedback session in which the conversation and their observations are played back for them, with errors and correct responses highlighted.

Critique exercises are designed to focus on the sub-topic (and in some cases, the topic) level. Therefore, the assessments provided to the student model reflect those levels. In addition, since students "observe" a conversation, they also receive credit with the student model for having seen the conversation and application situation-action pairs employed in that exercise.

Accompany. The accompany method is based on the principles of part-task training and reduction of cognitive load (Sweller, 1988). In accompany mode, the tutor presents both sides (customer and CSR) of the conversation, while the student is expected to perform the database application actions. Thus, the student "accompanies" the conversation on SONAR.

In an Accompany exercise, the student's receive "credit" with the student model for having seen the exercise's conversation situation-action pairs, and receive student model assessments for their performance on the exercise's application situation-action pairs.

Session Feedback. At the end of each rehearse conversation session, the LEAP Tutor evaluates the student's performance and prepares and presents feedback. The LEAP Tutor provides for several different types of feedback, selectable by the student. The types of feedback available are patterned after those implemented in SHERLOCK (A. Lesgold, personal communication, December 1992), and will include:

- *Summarize progress* — The student is presented with a graphical representation of the student model indicating LEAP's assessment of the student's status with regard to the various topics.
- *Session Playback* — The student is presented with a step-by-step walk-through of their rehearse conversation session, with expert responses available upon request.
- *Selective Review (Summary and Playback)* — The student can select a portion of the previous rehearse conversation session to review (this includes both a summary assessment and a step-by-step playback with expert responses available). The selective review can be either by conversation component (i.e., legal issues, verify feature compatibility, etc.) or by student response type (i.e., verbal, operations or implicit).

- *Global Facet Review* — The student is presented with a checklist in which they will be asked to rate their performance on a number of areas (e.g., conversational tone, conciseness and clarity of explanations, sales effectiveness, effectiveness of handling customer problem, etc.). This will prompt the student to keep these factors in consideration during subsequent rehearsals, and could also be used during instructor reviews.

Following completion of the rehearse conversation feedback, the student returns to the LEAP Tutor's top level screen set where the student's proficiency levels for the course topics have been updated and presented. From this point, the student proceeds as before, either selecting the next topic and method, or else asking the tutor for a recommendation.

Test. Test mode is the LEAP Tutor's means for measuring the student's knowledge and skill levels under as realistic conditions as possible. In Test mode, the student must perform entire conversations -- there is no skimming or scaffolding, all implicit situation cues are completely faded, access to the multimedia lesson materials is blocked, and no coaching hints are provided by the tutor.
Students can reach test model by one of two methods. First, when the student reaches some criterial level of performance in practice sessions, the LEAP Tutor recommends they take a "test." Second, students can access test mode at any time as a means of self-evaluating their readiness.

Examine Contact Flow

In Examine Contract Flow mode, the LEAP Tutor supports students' explorations of the underlying conversation structure (i.e., the task grammar network) by allowing them to navigate the network, select which branch of the network to take, and back up and try different alternatives. Student's can examine the flow of customer/CSR contacts in either of two ways: Explore or Browse. Browse enables students to see the tutor's response to conversational situation they (the student) select. Explore enables students to not only select conversational situations, but also select the response to those situations and have the tutor evaluate their selection.

In a Browse session, the students receive "credit" with the student model for those situation-action pairs they examine. In an Explore session, the students receive student model assessments for the situation-action pairs they interact with.

User Interface

The user interface is the means by which individuals interact with a system. Further, it is the user interface, more than any other function, that serves to define for the user what that system does, and how well it does it. This is particularly important in an intelligent tutoring system because users of ITSs are by definition working with concepts they do not understand very well (Miller, 1988). A bad user interface generally results in a system that is not only difficult to use, but one that users do not want to use. In an ITS, in addition do determining how students interact with the system, the user interface also defines how students interact with the domain being tutored. Keeping these issues in mind, the user interface was designed to provide an intuitive window into the LEAP system, and the domains in which it tutors. The remainder of this section discusses the design principles to which the LEAP Tutor user interface adheres.

The first principle is that all interactions in the LEAP Tutor user interface will take place using a first-person (i.e., direct manipulation), point and click implementation. In other words, all activities are carried out by single-action manipulation of graphic objects that map directly onto the task and domain of study. The second principle is that all objects in the LEAP Tutor user interface that can be manipulated or used in any way for information conveyance given a current system state should be visible to the user (i.e., no pull down menus or keystroke-mouse combinations to see additional options). The third principle is that interface objects should be grouped according to functionality. For instance, objects concerned with system navigation are kept together and kept separate from objects concerned with dialog control (which themselves are kept together). The fourth principle is neither require nor allow the user to control the presentation or appearance of any of the interface objects. Students need to expend their mental energies on learning the domain in question, not on customizing the user interface. The final principle is that system functions should be broken down into screen-sets, with all activities possible for a given function supported within the screen-set's objects.

For example, when a student rehearses a conversation, all of the interface objects required to support that rehearsal are contained within its screen-set (i.e., dialog objects, dialog control objects, feedback objects, session trace objects, and system navigation objects). If during that rehearsal the student chooses to review an associated declarative lesson, transitioning to the lesson results in their exiting the rehearsal screen-set (via a navigation button) and entering the study topic screen-set. While in the study topic screen-set the student would neither see nor have access to any of the rehearse conversation screen-set objects, unless they were to return to that function (via another navigation button). This type of conceptually focused organization minimizes the amount of external processing a student must commit to learning and using the interface, and focuses their processing on the task at hand - learning the tutor's domain. Figure 5 presents a hierarchical view of the LEAP Tutor's screen-sets.

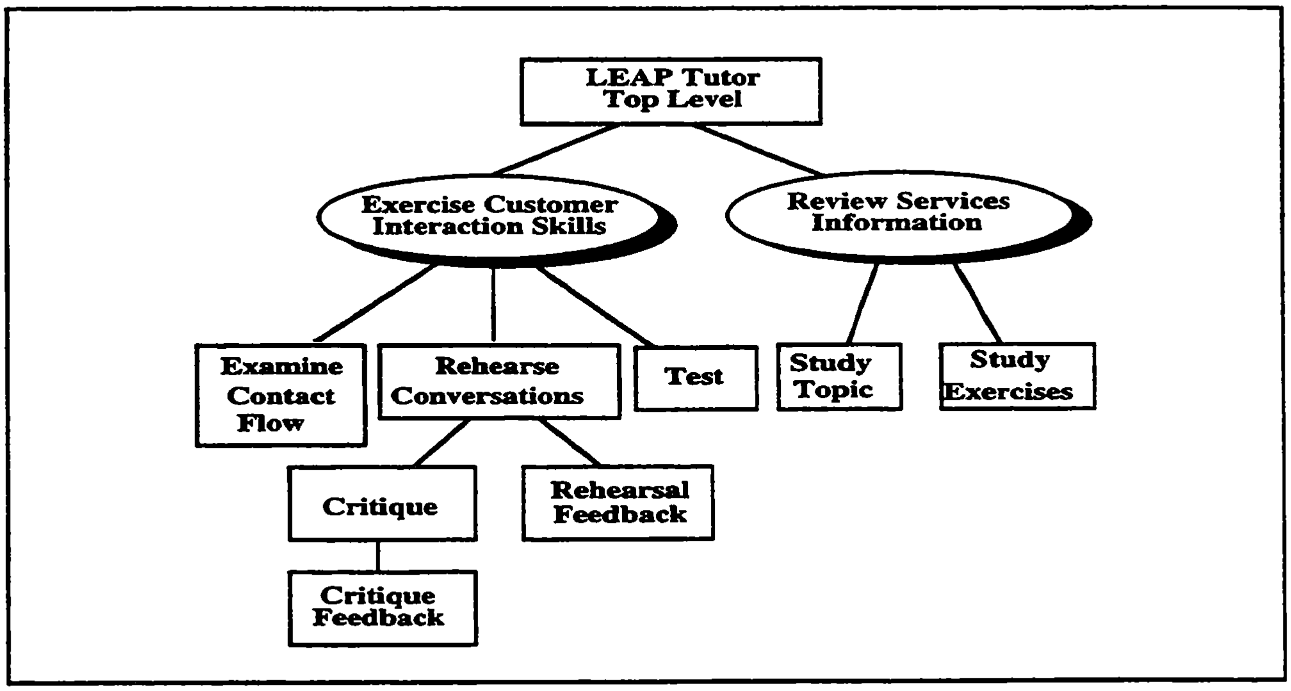


Figure 5. LEAP Tutor Screen-Set Hierarchy

As depicted by the rectangles in Figure 5, there are nine distinct screen-sets within the LEAP Tutor:

- *LEAP Tutor Top Level* — This screen-set provides the student with one window containing the set of control objects for accessing various tutor functions (i.e., navigation and recommendation), and a second window containing a selectable scrolling, hierarchical list of topics available for study and their proficiency with those topics, and a set of selectable objects representing the methods available for studying or practicing those topics.

- *Rehearse Conversation* — (used in the Demonstration, Accompany and Practice methods) This screen-set provides the student with one window containing the set of rehearse conversation control and navigation objects for accessing available tutor functions (i.e., Related Information, Help, Exit), another window containing SONAR, a third window where all dialog control activities take place, a fourth window for the coach to provide feedback, hints, and context information, and an optional fifth window containing a scrollable trace of the current session.

- *Critique* — This screen set is a super-set of the Rehearse Conversation screen set. In addition to all of the objects from the Rehearse Conversation screen set, the Critique screen set will also contain a checklist window containing lists of selectable objects (e.g., check buttons).

- *Rehearsal Feedback* — This screen set is a super-set of the Rehearse Conversation screen set. In addition to the Rehearse Conversation screen set objects, the Rehearsal Feedback screen set will contain an expanded feedback window.

- *Critique Feedback* — This screen set is a super-set of the Critique screen set. In addition to the Critique screen set objects, the Critique Feedback screen set will contain an expanded feedback window.

- *Test* — This screen set is a subset of the Rehearse Conversation screen set in which the feedback and context windows are removed, and the navigation object for viewing related information is disabled and grayed out.

- *Examine Contact Flow* — This screen-set provides the student with one window containing the set of Examine Contact Flow control and navigation objects, a second window where all dialog control activities take place, a third window for the coach to provide feedback, hints, and context information, and an optional fourth window containing a scrollable trace of the current session.

- *Study Topic* — This screen set provides the student with one window resembling an "interactive" book. The book contains "index markers" which are buttons providing interactive access to the various lessons. Each page of the book represents a particular sub-topic of a lesson and can contain text, graphics, animations and/or video sequences. In addition to continual access to the books index markers, each page also contains navigation buttons for "paging" forward or backward, for exiting the lesson and returning to the LEAP Tutor's top-level screen set, or for exiting to a specific Rehearse conversation session on the topic being studied.

- *Study Exercises* — This screen set provides the student with one window where all exercise interactions take place. At the top of the window is an area for control buttons (i.e., Next, Record, Playback, Exit). Below that is an area where the specific exercise instructions are presented to the student. Below the instructions is a "Clue" area, and below the clue area is an area for student responding (i.e., checklists, areas for typed input, interactive application screens). In the special case of an audio study exercise, the student sees only the control area and instructions.

Student Model

The student model is essentially a dynamic data structure that maintains a "model" of each student within and across training sessions. Consistent with previous research (Villano & Bloom, 1992), the LEAP Tutor uses the student model to:

- Adaptively assess the student's mastery of the learning materials
- Represent the student's progress through the learning materials and recommend topics or conversations the student needs to study or practice
- Select and present instructional interventions at appropriate levels of understanding
- Provide the student with performance feedback

To accomplish these objectives, the student model employed in the LEAP Tutor is comprised of four major components:

- *Student Model Data Structure* — A dynamic representation of the knowledge and skills to be learned with associated confidence estimates
- *Student Model Performance Modeling Function* — Mechanism for tracking all student activities within the tutor and mapping them to the student model data structure
- *Student Model Updating Function* — Statistical techniques for going from a student's individual actions to estimates of how well the student knows each identified item of knowledge or skill
- *Student Model Diagnostic Algorithm* — Techniques for interpreting the student model data structure and applying the various learning techniques employed in the tutor

The LEAP Tutor's student model data structure is modeled after an Operator Function Model - OFM (Mitchell, 1987). Consistent with an OFM, the LEAP student model represents scenarios in a hierarchical network, with scenarios and major topics at the highest levels, and individual conversation activities at the lowest level. Actions can be verbal (i.e., responses by the CSR to the customer or to SONAR output), operational (i.e., a commands or data entered into SONAR by the CSR), or cognitive (i.e., information gathering, information processing, and decision making). Figure 6 presents an example of a subset of the LEAP student model representation.

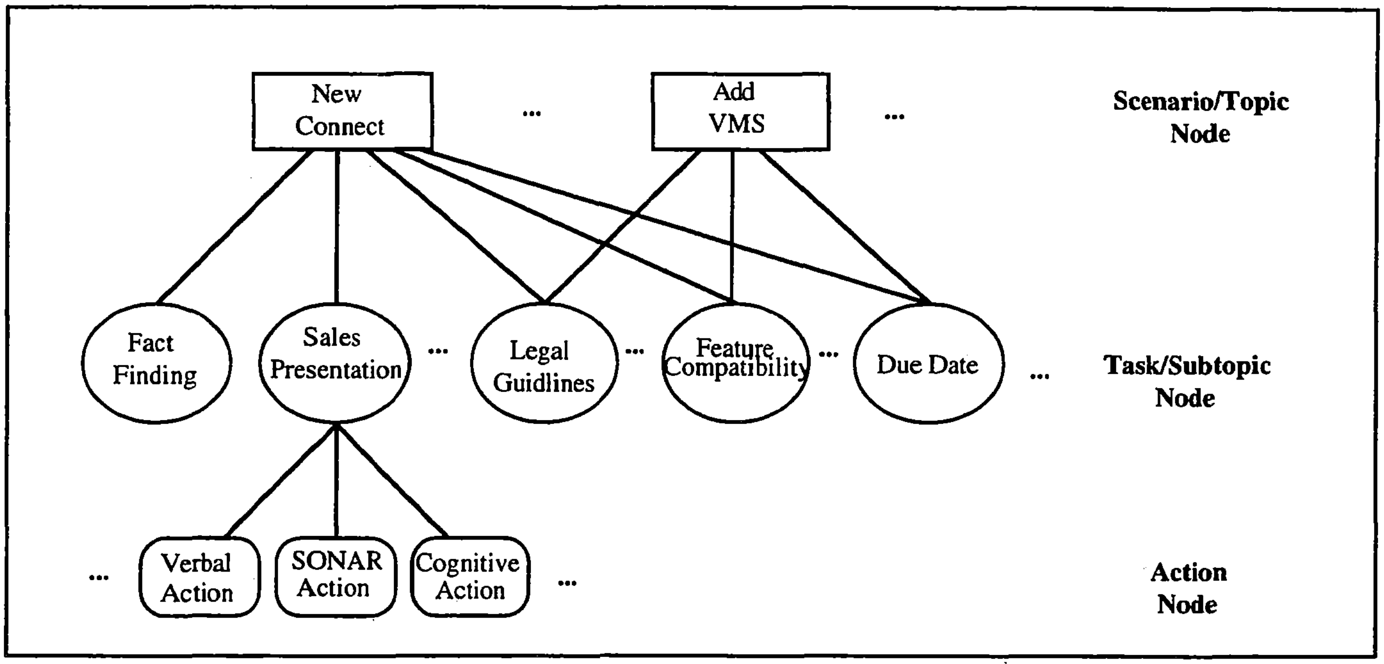


Figure 6. LEAP Student Model Representation

The top level, referred to in Figure 6 as the Scenario or Topic Node level, represents all of the possible conversational scenarios that the CSR can engage in (and supported by the LEAP Task Grammar). The second level, referred to in Figure 6 as the Task or Sub-Topic Node level, represents all of the major tasks or parts of conversations within each scenario or topic. The lowest level represents the individual actions (described previously).

In ECIS mode, for each student, the LEAP Tutor records three pieces of data for each situation-action pair: An average correct score, the number of consecutive correct or incorrect responses, and the number of times a pair is seen but not practiced. These data points are combined using a weighting function (the individual weights for each factor are interrelated and adjustable as described in the section on Parameter Editing). The computed value is a measure of the LEAP Tutor's confidence that the student can perform the activity associated with the situation. The value is updated each time a student sees or responds to a situation-action pair. The values from the situation-action pairs are propagated up to parts of conversations (i.e., sub-topics), and ultimately up to the conversation or topic level.

In the case of a Critique conversation, the LEAP Tutor records two pieces of data at the sub-topic and topic level: An average score on each critique exercise, and the number of consecutive correct or incorrect responses on each Critique exercise. These data points are combined using an adjustable weighting function and combined with the other ECIS sub-topic and topic estimates.

In RSI mode, the LEAP Tutor records three pieces of data for each topic and sub-topic: The number of times a lesson is reviewed, an average score on each Study Exercise, and the number of consecutive correct or incorrect responses on each Study Exercise. These data points are also combined using an adjustable weighting function, and also combined with the ECIS estimates propagated to the sub-topic and topic levels to provide the tutor's overall assessments. These combined estimates are then converted into one of five categorical labels (i.e., "Untried," "Needs Practice," "Almost," "Good," or "Excellent"). The categorical conversation is also adjustable, so that instructional designers or field instructors can vary the "scores" a student needs to move from one level to the next.

Given a lack of consensus in the student modeling literature on methods for updating confidence estimates based on observable data (Bloom, Villano, VanLehn, Jones, Watson & O'Bannon, 1992), the parameters used in the LEAP Tutor's student model updating function were made to be adjustable.

The diagnostic algorithm of the LEAP Tutor's student model is based on two assumptions: (1) The effect of practice on learning skills and knowledge can be fitted to a traditional, negatively accelerating learning curve, such that the more one practices, the greater that person's odds of responding correctly. (2) The need for practice can be fitted to a negatively decelerating learning curve, such that as skills increase, the need for practice decreases.

As stated previously, each student has their own student model representation that is saved across training sessions, enabling the student to resume their instruction based on their performance on pervious sessions. In addition, student models will only be viewable by their owners. The intent of LEAP is to provide CSRs with a coached practice environment that allows the student to monitor their own learning progress, it is not intended to allow supervisors to use LEAP's findings to grade or assess the CSRs. To protect each student, the student model will also contain a user-model component. This user model component will contain information about the user (i.e., user name, user preferences, etc.) as well as a private password for security protection.

PARAMETER ADJUSTING

One of the more powerful features of the LEAP ITS Platform is its Parameter Adjusting function. The LEAP Tutor makes its instructional decisions on what topic to recommend, what conversation to execute, how to evaluate a student's responses and how to apply its within-practice instructional methods (i.e., skim & scaffold) based on sets of parameters that can be adjusted by course developers and instructors.

The rationale for this approach is as follows: (1) We were able to identify what factors influenced these decision making strategies (each has been used in some previous instructional system or method), but not the relative contributions of each when taken together. And, (2) parameterization of these factors would provide us with the means of conducting systematic empirical investigations into this issue.

The LEAP Parameter Adjuster has been designed so that the weights given to each factor can be easily modified. The weights (and hence the contributions) of each factor can range on a continuous scale from zero (no contribution) to 100 (full contribution). In addition, these factors are interrelated on all but the determination of the application of within-practice instructional methods, so that increasing the weight of one factor will produce uniform decreases to the weights of the other factors (and vice versa).

Topic choice is based on three interrelated factors: (1) Sequential ordering of topics (as represented in the topic/sub-topic structure), (2) relationship to the last topic/sub-topic studied, and (3) the student's proficiency on each topic/sub-topic. The sequential order of topics represents the sequence determined in accord with current instructional design practices. In other words, if an instructor were going to build the "best and only" route through the course, it would be represented by this sequential order. However, different students learn best in different orders, so we have the two other factors that can produce a "resequencing" of the topics as recommended to the student. Relation to the last topic influences recommendations on a conceptual relatedness rather than sequential level. With regard to proficiency, at present the author can select from among three predefined prioritization settings that differ with regard to the way the tutor recommends student progress through topics in the "Untried" to "Almost" categories:

- *Prefer "Almost" to "Needs Practice" to "Untried"* — Elevate students proficiencies on individual topics/sub-topics to good before proceeding to other topics.

- *Prefer "Needs Practice" to "Untried" to "Almost"* — Elevate students proficiencies on all individual topics/sub-topics to "Almost" before proceeding to "Good."

- *Prefer "Untried" to "Needs Practice" to "Almost"* — Elevate students proficiencies on all topics/sub-topics one level at a time.

Conversation choice is based on six factors (5 of which are interrelated): (1) relation to selected topic, (2) complexity of conversation (in terms of the number of SA pairs in the conversation), (3) the number of times the conversation has already been seen or practiced by the student, (4) the students current skill level on the conversation, (5) an independent teacher preference rating, and (6) the student's preference for hard vs. easy conversations (the only independent factor).

At present, student response evaluation is only implemented for ECIS' mode. Plans are in progress to extend this capability to RSI mode and to increase the number of parameters employed in ECIS mode to include response latencies and times. Student response evaluation is based on three interrelated factors: (1) Average correct score when asked, (2) the number of consecutive correct (or incorrect) when asked, and (3) the number if times the item is seen but not asked.

The application of within-practice instructional methods (i.e., skim and scaffold) is the only Parameter Adjusting function in which the parameters are not interrelated. In this function, the author can set the parameters for skim (In Topic) and scaffold (Out of Topic) separately. However, each uses identical parameters: (1) Question the student when not known, (2) question the student when known, and (3) Decrease questioning when known.

With regard to skimming information that is "In Topic," the question when not known parameter is set to always as a default -- the objective is to get students to practice that information within the topic they are studying that they do not yet know. The question when known parameter is set at the midpoint as a default and refers to how often do you want the student to get prompted to respond to something they already know -- just enough to keep the information active. The decrease questioning when known factor refers to the how gradually you want the system to stop asking the student something they have demonstrated they know.

With regard to scaffolding information that is "Out of Topic," the question when not know parameter is set to never as a default -- we do not want students to get frustrated trying to answer items the tutor has no reason to believe they know. However, the more times a student sees and item that is unknown and out of topic, the tutor's assessment is updated to the point where is reflects some level of knowing for those items not practiced by seen. This is where the question when known-out of topic factor comes in -- we begin to ask the student to respond to items that are out of topic, but they should have some knowledge of. The default setting for this parameter is at the midpoint. The decrease questioning when known is the same for Out of Topic scaffolding as for In Topic skimming.

HARDWARE AND SOFTWARE REQUIREMENTS

The LEAP Tutor is envisioned to ultimately migrate towards delivery on U S WEST Information Technologies' Intelligent WorkStation (IWS). The IWS is a platform composed of front-end UNIX X-terminals with 19 inch color monitors supporting 8 bit color displays, that is connected to several application and communication serves via a high speed (10 megabits per second) ethernet local area network (LAN). The IWS is proposed to be capable of supporting installations varying in size from one to 160 CSRs. The IWS provides access to the database application the LEAP Tutor will need to interact with, the Service Order Negotiation and Retrieval (SONAR) system. The SONAR application itself resides on an IBM 3270 mainframe accessed through the IWS application/communication server and displayed on the front-end X-terminal using a 3270 emulator.

For the purpose of evaluating the LEAP tutor's instructional effectiveness, we will be using a Sun Sparc 10 Model 41 with 128 MB of RAM as a server, and an external 2 GB disk drive for multimedia file storage. The Model 41 is a dual processor model, with one processor to be used as a server for the LEAP software and the second to be used for multimedia file serving. The server will be connected to the IWS LAN to enable access to SONAR. Connected to the LEAP/multimedia server via a dedicated LAN will be a series of Sun Sparc 2s and IPXs running in stand-alone mode. This configuration provides us with the ability to test the tutor without adversely impacting the IWS LAN, and provides us with a work-around the IWS limitations in the areas of audio and video processing. The LEAP system is built in common LISP, using a separate user interface code written in C++. In addition, the LEAP system uses two other pieces of software: (1) A standard 3270 terminal emulator for access to SONAR, and (2) a Bellcore application called XFUR, used to interact with the terminal emulator to allow the LEAP Tutor to control and monitor student actions taking place in SONAR (Lefkowitz & Farrell, 1991).

REFERENCES

Anderson, J. R. (1988). The expert module. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Bloom, C. P., Villano, M., VanLehn, K., Jones, J., Watson, P. K., & O'Bannon, M. (1992). *Application of Artificial Intelligence Technologies to Training Systems: Computer-Based Diagnostic Testing System*. (Research Rep. No. AL-TR-1992-0072). Brooks Air Force Base, TX: Air Force Human Resources Directorate, Technical Training Research Division.

Burns, H. L., & Capps, C. G. (1988). Foundations of intelligent tutoring systems: An introduction. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Carroll, J. M. (1990). *The Nuremberg funnel*. Cambridge, MA: The MIT Press.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive Apprenticeship: Teaching the crafts of reading, writing and mathematics. In L. B. Resnick (Ed.), *Knowing, learning and instruction*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 453-494.

Halff, H. M. (1988). Curriculum and instruction in automated tutors. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Lefkowitz, L. & Farrell, R. (1991). *Assessing the applicability, utility and feasibility of WITS intelligent tutoring for your operations system*. (Technical Memorandum No. TM-STS-019544). Piscataway, NJ: Bellcore.

Lesgold, A., Lajoie, S., Bunzo, M., & Eggan, G. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin & R. Chabay (eds.), *Computer-assisted instruction and intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 201-238.

Miller, J. R. (1988). The role of human-computer interaction in intelligent tutoring systems. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Mitchell, C. M. (1987) "GT-MSOCC: A domain for research on human-computer interaction and decision aiding in supervisory control systems," *IEEE Transactions of Systems, Man, and Cybernetics*, vol. 17, pp. 553-570.

Sweller, J. (1988). Cognitive load during problem solving: Effects of learning, *Cognitive Science*, *12*, pp. 257-285.

VanLehn, K. (1988). Student modeling. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Villano, M. & Bloom, C. P. (1992). *Probabilistic student modeling with knowledge space theory*. (Research Rep. No. AL-TP-1992-0046). Brooks Air Force Base, TX: Air Force Human Resources Directorate, Technical Training Research Division.

Wenger, E. (1987). *Artificial intelligence and tutoring systems*. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

The Home and Building Control Fundamentals Tutor: A Design Framework for the Multimedia Instruction of Declarative Concepts'

Peter Bullemer, Rose Chu. Nagi Kodali[1] and Michael Villano[2]
Honeywell Sensor & System Development Center
MN65-2300, 3360 Technology Drive
Minneapolis, MN 55418

Bullemer at (612) 951-7554
bullemer_peter@ssdc.honeywell.com

In Honeywell's Home and Building Control (HBC) division, new employees for engineering, sales, technical support or administration must attend an introductory course on basic control fundamentals. In the past, the course comprised stand-up classroom instruction involving many instructors each teaching distinct domain topics. In response to a request to explore alternative cost-effective training methods, we developed the HBC Fundamentals Tutor, a computer-based training system that permits employees to complete a week's worth of classroom instruction at their own pace and in their own work environment using widely available 80386 PC hardware. The Tutor is presently being used in over 75 branch offices.

The HBC Fundamentals Tutor contains 56 lessons that teach a broad spectrum of declarative concepts using text, static graphics, digitized images, and animation. We propose a design framework for supporting the instruction of declarative concepts that range from concept definitions to system behaviors. Key features of the design framework include the functional organization and multilevel progress indicators used to support navigation in this self-directed study environment. These design features address students' need for explicit cognitive aids to track their progress and location in the course curriculum. Moreover, we present the classes of interaction objects developed to access different types of declarative concepts. For each type of declarative concept, we illustrate and discuss the features, we outline the planned design enhancements that will provide intelligent tutoring capabilities.

[1]Now at Mechanical Engineering Department, Anderson Hall, Tufts University, Medford, MA 022155
[2]Now at NYNEX Science & Technology, Inc., 500 Westchester Avenue, White Plains, NY 10604

N96- 14980

SMART--Situated Multimodal Advanced Real-time Trainer

Stephen Desrosiers and Debra Bettis
McDonnell Douglas Aerospace
13100 Space Center Blvd.
Houston, TX 77059
(713) 280-1536 (713) 280-1689
desrosie@sweetpea.jsc.nasa.gov

We have begun work on an intelligent support tool (SMART) for operating and maintenance procedures. This work extends McDonnell Douglas's recent success in developing electronic job aids for onboard applications.

SMART is based on advanced AI technology that derives the sequential, situational, operational, and temporal views of operating and maintenance procedures from a single internal representation of the procedures.

SMART guides the astronauts through detailed operating and maintenance procedures. It also helps the astronauts decide when and how it is safe to deviate from the established procedures.

SMART's user interface presents the sequential and procedural views of the procedures to the astronauts. The views are supplemented with vivid color graphics of internal system functions, digitized photographs of flight hardware, and full-motion video. SMART also allows the astronauts to keep their own personal notes attached to a procedure. SMART will be delivered on portable hand-held systems.

SMART provides highly descriptive, realistic, "hands-on" training which is available "on-demand" wherever and whenever it is needed.

N96- 14981

A Computer-based Training System Combining Virtual Reality and Multimedia*

Sharon A. Stansfield
Sandia National Laboratories
Albuquerque, NM 87185

April 28, 1993

Abstract

Training new users of complex machines is often an expensive and time-consuming process. This is particularly true for special purpose systems, such as those frequently encountered in DOE applications. This paper presents a computer-based training system intended as a partial solution to this problem. The system extends the basic virtual reality (VR) training paradigm by adding a multimedia component which may be accessed during interaction with the virtual environment: The 3D model used to create the virtual reality is also used as the primary navigation tool through the associated multimedia. This method exploits the natural mapping between a virtual world and the real world that it represents to provide a more intuitive way for the student to interact with all forms of information about the system.

1. Introduction

DOE applications, such as Environmental Restoration and Waste Management (ER&WM) and nuclear weapons dismantlement will require complex, special-purpose robotic systems. Training the operators of such systems will be both expensive and time-consuming. This is due to limited access, the expense of duplicating the system for training, and the negative consequences of making a mistake while learning. This paper presents a computer-interactive training system that combines virtual reality (VR) with multimedia and artificial intelligence (AI) to provide an extended training environment for such applications. Virtual reality is used as the primary training tool, in much the same way that it has been used traditionally for flight simulators and other similar applications. A multimedia component, containing additional information about the application and equipment in the form of video, text, voice, and audio may also be accessed during interaction with this virtual environment. The 3D graphical model used to create the virtual reality is also used as the primary navigation tool through the associated multimedia. In addition, AI techniques are incorporated to aid in creating realistic simulations, in navigating the hypermedia and, eventually, in tutoring the student .The rest of this paper describes the prototype system. Section 2 presents the application and the virtual training environment in more detail. Section 3 presents the multimedia component. Section 4 discusses the addition of intelligent agents for both training and media navigation.

2. Virtual Reality for Training

The initial application of this work will be in conjunction with the in-house Underground Storage Tank (UST) demonstration being developed at Sandia National Laboratories. This demonstration will provide a testbed for a number of technology development activities related to the retrieval of hazardous radioactive waste from

*This work has been performed at Sandia National Laboratories and supported by the U.S. Department of Energy under Contract DE-AC04-76DP00789.

underground storage tanks. Because of the hazardous nature of these tanks, robotic systems will be used to retrieve and store the radioactive materials. This provides special challenges for robotic technology development: Tanks found in Hanford WA., for example, contain waste in the form of liquid, sludge, and hard salt cake. They also contain risers, pumps, cooling pipes, debris, and other obstacles. Obviously, the systems ultimately used for such work will be complex, consisting of many tools, sensors, and manipulators. Operating such systems will be a difficult task at best.
The in-house UST demonstration testbed provides a mock-up of a UST environment. It consists of a variety of robots, sensors, and tools. This entire robotic system is under computer control. At the highest control level, human operators use graphical programming techniques to provide commands to the system. Flat screen displays provide feedback concerning the status of the devices and sensors[Christensen 92].
An on-going development effort is underway to provide a virtual reality-based interface to this system. This interface provides a VR model of the UST testbed, including simulation capabilities for the manipulators and tools. The operator may move around in this virtual environment, using voice commands and hand gestures to program accurate graphical models of the actual robots. These task-level commands then automatically generate device-level programs to be down-loaded to the robots. The operator may also preview these operations for correctness via simulation and edit them interactively before sending them to the robotic devices. During operations, the simulation is driven by the positions of the robotic devices and the readings from the various sensors to allow the operator to monitor the state of the operations. The positions and states of the manipulators, tools, and sensors are accessed and used to upgrade the graphical models in the virtual environment.
The virtual reality testbed for this work consists of a Fakespace, Inc. mechanically-tracked stereoscopic viewer (BOOM), a Polhemus FastTrack wrist tracker, a Dragon Systems voice recognition system, and a Silicon Graphics Inc. workstation with audio. The software platform is the Cim-Station robot modeling and simulation package from SILMA, Inc. The VR interface will be used both for training the operators of the system (simulation mode) and during actual operations (command and monitoring mode). The primary difference between these two modes is whether the actual robotic devices are in the loop or not. Figure 2 conveys the idea behind this work. The figure shows an operator in the Fakespace BOOM, the CimStation model of the UST testbed, and the actual testbed.

3. The Multimedia Component

The operator may access additional information about the devices from the virtual environment by putting the system into training mode. In this mode, a selected object provides an access point into a hypermedia system containing information in the form of video, text, and still photos. The hypermedia structure is a graph containing *menu* nodes for navigation and *action* nodes for accessing the media. In the case of the UST application, additional information is associated with each of the robots and with a 3D icon which represents the underground storage tank application itself. An object is selected by "touching" it with a 3D pointer driven by a joystick and pressing the joystick button. This device will soon be replaced by a hand-held, magnetic-based position tracker. Spoken commands will be used for making selections. Figure 3 shows the system with the various devices used for viewing the information.

Information associated with each robot includes still photos, system diagrams, video of the actual device in operation, and the manuals for that device. Manuals are divided into reasonable chunks based upon the structure of the original text. Any figures associated with the text are also available. The UST icon invokes a text report on the Hanford site, photos of actual tanks, diagrams of remediation plans, and a DOE video describing the problem. Menus allow the user to navigate through the information for a particular device or icon.

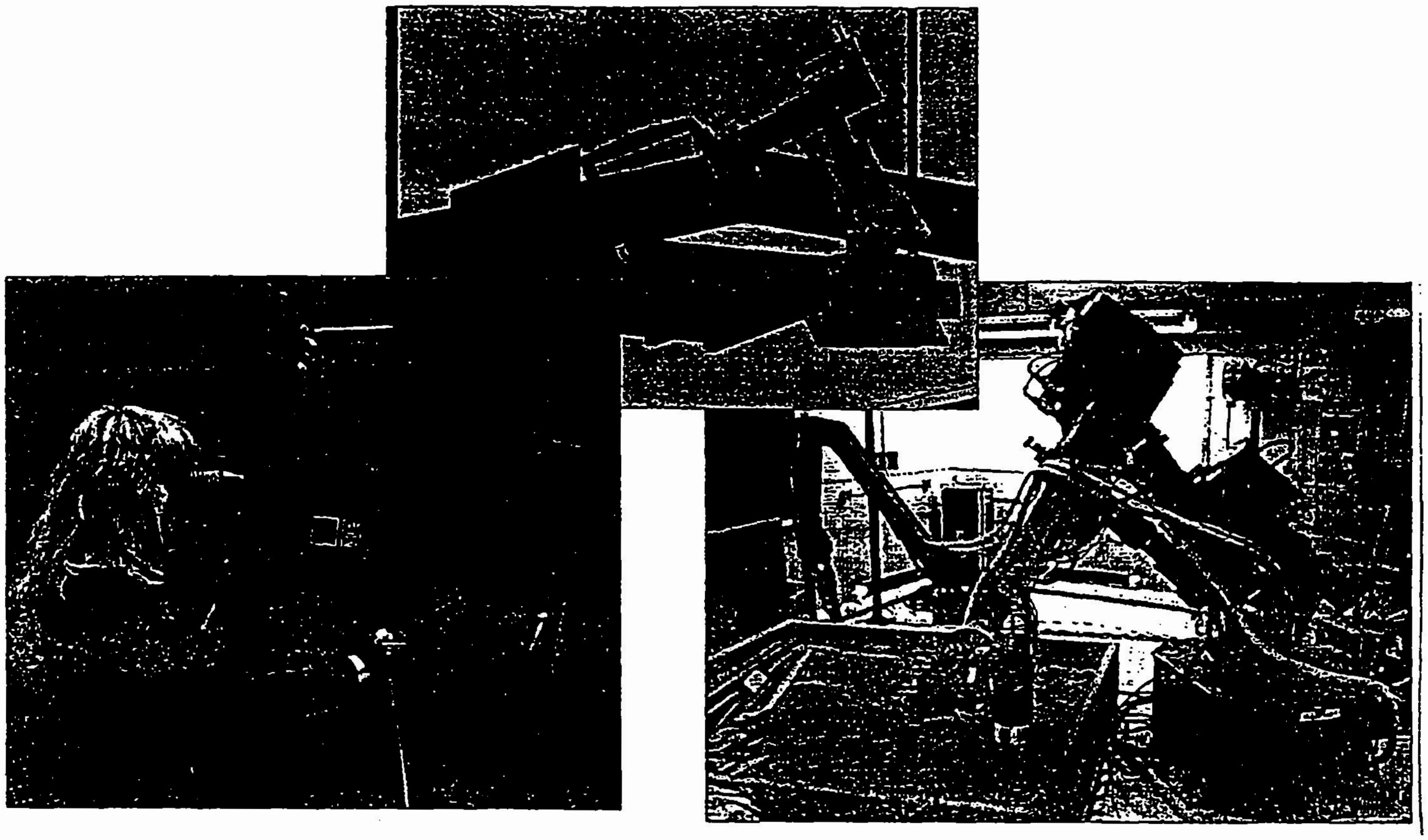
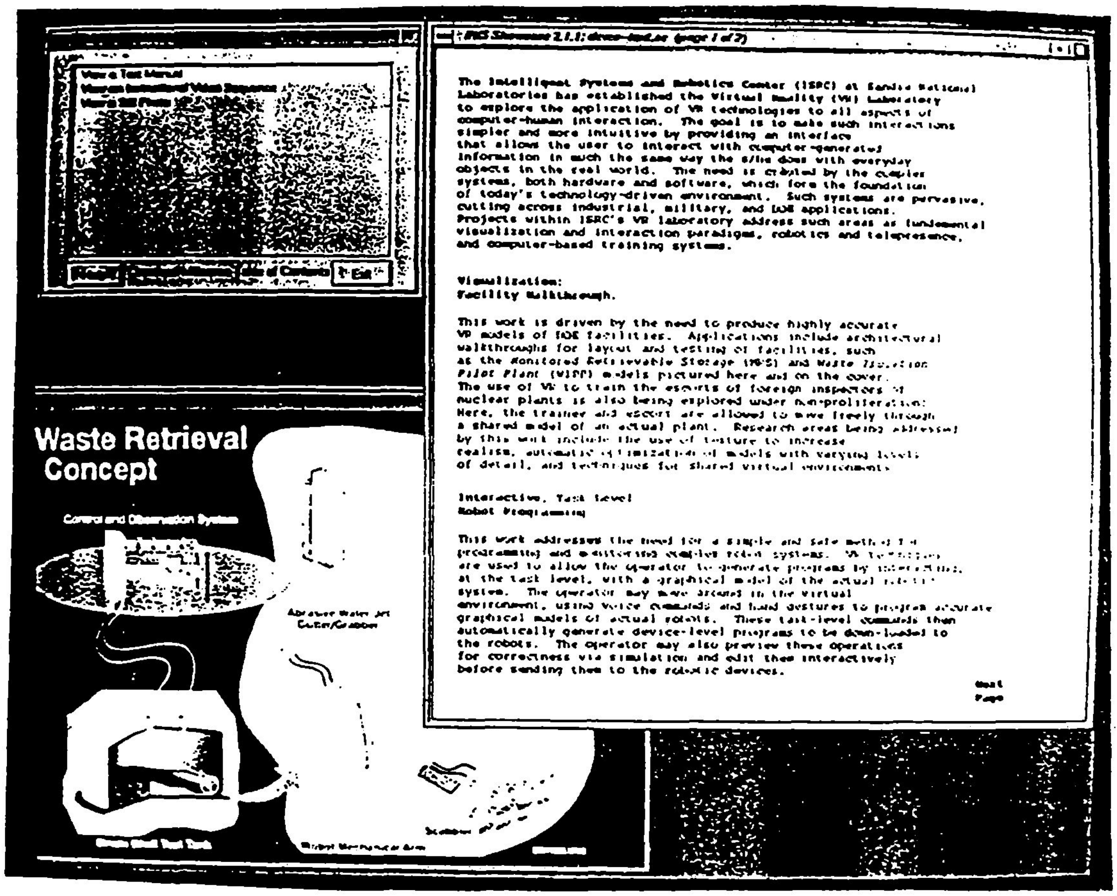


Figure 1: VR for robot operator training.

Figure 2: The VR-based multimedia training system.


Waste Retrieval Concept
Control and Observation System
Abrasive Water Jet Cutter/Grabber
Robot Manipulator Arm
Next Page

The Intelligent Systems and Robotics Center (ISRC) at Sandia National Laboratories has established the Virtual Reality (VR) Laboratory to explore the application of VR technologies to all aspects of computer-human interaction. The goal is to make such interactions simpler and more intuitive by providing an interface that allows the user to interact with computer-generated information in much the same way the s/he does with everyday objects in the real world. The need is created by the complex systems, both hardware and software, which form the foundation of today's technology-driven environment. Such systems are pervasive, cutting across industrial, military, and DOE applications. Projects within ISRC's VR laboratory address such areas as fundamental visualization and interaction paradigms, robotics and telepresence, and computer-based training systems.

Visualization:
Facility Walkthrough.

This work is driven by the need to produce highly accurate VR models of DOE facilities. Applications include architectural walkthroughs for layout and testing of facilities, such as the *Monitored Retrievable Storage* (MRS) and *Waste Isolation Pilot Plant* (WIPP) models pictured here and on the cover. The use of VR to train the escorts of foreign inspectors of nuclear plants is also being explored under non-proliferation: Here, the trainee and escort are allowed to move freely through a shared model of an actual plant. Research areas being addressed by this work include the use of texture to increase realism, automatic optimization of models with varying levels of detail, and techniques for shared virtual environments.

Interactive, Task Level
Robot Programming

This work addresses the need for a simple and safe method for programming and monitoring complex robot systems. VR techniques are used to allow the operator to generate programs by interacting, at the task level, with a graphical model of the actual robotic system. The operator may move around in the virtual environment, using voice commands and hand gestures to program accurate graphical models of actual robots. These task-level commands then automatically generate device-level programs to be down-loaded to the robots. The operator may also preview these operations for correctness via simulation and edit them interactively before sending them to the robotic devices.

Figure 3: A representative screen.

Spoken word audio also provides guidance. A near-term addition will allow the user to speak preferences as well as to select them with the mouse. Figure 3 shows a representative screen for the multimedia component.

4. Adding Intelligence

Ultimately, the training system will embody intelligence in three forms: Intelligent behaviors during simulation, intelligent navigation tools for moving through the hypermedia component, and intelligent tutoring. Current emphasis is on the first two. The fundamental representation of an object is being extended to allow the invocation of intelligent behaviors for that object based upon the state of the simulation. For example, a pipe obtains the knowledge that it has been cut and behaves accordingly: it becomes two pieces, one of which falls to the ground. Likewise, waste contains the knowledge that it is radioactive and can communicate this state to the sensor object which can behave accordingly by creating the proper readings. Such intelligent behaviors add realism to the training simulator by allowing the user to run different scenarios. Emergency situations and equipment failures may also be invoked to train the operator to handle them.

Intelligence will be added to the hypermedia component to aid in navigation. Initially, this intelligent navigation tool will utilize typed, weighted links and a set of simple rules to allow the system to provide a list of prioritized suggestions as to other, related information that the user might also be interested in accessing. Future work will enhance this navigation tool by integrating machine learning techniques. Learning algorithms will be applied to allow the system to learn from the behavior of previous users. The system will then use this knowledge to guide current users through the hypermedia information.

Future work also includes the addition of an intelligent tutoring component that will interact with the student during training. In addition to information about the environment and tasks, the student will be able to access interactive tutorials concerning the operations being ed. The system will also be capable of monitoring the student's performance during a training session and logging this performance. The system will then offer critiques and advice concerning where additional train nd/or study is needed and where the relevant material may be found.

5. Summary

This paper has provided an overview of the intelligent, c puter-based training system being developed at Sandia National Laboratories for use in training the operators complex, automated machines. This system combines virtual reality with multimedia and artificial intelliger to extend the basic VR training paradigm to a more complete teaching tool. The virtual environment provid he underlying simulation for training the operators of robotic systems, such as those used for remediation o derground radioactive waste storage tanks. The VR model also provides the primary access points into rela information about the system and application in the form of video, text, still photos and sound. AI technique e used to create more realistic simulations and to help the user navigate through the multimedia. Future wor ill incorporate an automated tutoring component to assist the student in learning about the system and tasks.

References

[Christensen 92] Christensen, B. K., Drotning, W. D., and T borg, S. "Model Based, Sensor Directed Remediation of Underground Storage Tanks", *Journal obotic Systems*, pp. 145-159, 1992.

N96-14982

Virtual Agents in a Simulated Virtual Training Environment

Brett Achorn and Dr. Norman I. Badler
Department of Computer & Information Science
University of Pennsylvania
Philadelphia PA 19104-6389

ABSTRACT

A drawback to live-action training simulations is the need to gather a large group of participants in order to train a few individuals. One solution to this difficulty is the use of computer-controlled agents in a virtual training environment. This allows a human participant to be replaced by a virtual, or simulated, agent when only limited responses are needed. Each agent possesses a specified set of behaviors and is capable of limited autonomous action in response to its environment or the direction of a human trainee. The paper describes these agents in the context of a simulated hostage rescue training session, involving two human rescuers assisted by three virtual (computer-controlled) agents and opposed by three other virtual agents.

1 INTRODUCTION

Virtual training environments are gaining in popularity because of many attractive features, some of which are increased safety, reduced training cost, and simulated scenarios that would be impossible to duplicate in live-action training. However, once one uses a computer to create the training environment, it is possible to take a further step and simulate some of the participants in the exercise who are present in only a supporting role. This is presently done at the level of vehicular entities, such as airplanes and tanks, in battlefield simulations. The replacement of human participants with simulated agents is possible when the responses required of the agent are well-defined and only limited autonomous action is needed. Not only would the use of simulated agents reduce the number of participants needed for a training session, but it would also afford trainers greater control over elements of the training exercise.

In order to be effective, a simulated agent must be designed with several considerations in mind [5]. These are:

1. Constructing a reasonable, interactive computer graphics human model
   The model of the agent must not only produce an image that is believable, but the large number of degrees of freedom in a human body must be managed to simplify control. There must be some way to control both local and global positioning.

2. Incorporating biomechanical correctness
   Since the human body is capable of a wide range of motion, an agent must be able to closely match this range, both in its capabilities and limits. Also, without some notion of what strength can be exerted at different joints, an agent will tend to perform actions that are not considered efficient or natural by human standards.

3. Developing human-like behaviors
   To simplify control and to produce realistic motions, it is desirable to incorporate a prepackaged set of actions and skills into an agent. [1] Such behaviors would include continuing actions like balance and self-collision avoidance [6], rhythmic motions such as walking [3], and skills such as grasping [6] or sitting.

4. Permitting the agent to perceive and affect the environment
   An agent's senses, whether direct like vision or indirect like hearing, must be modeled to account for both the kind of information returned and the limits to their field of detection. An agent must also be able to interact with at least some of the objects in its environment, whether it be to carry something or to open a door.

5. Allowing tasks to be guided by standing commands, instructions, or warnings
   In order to be effective, an agent must be able to accomplish some set of programmed instructions, deal with unplanned events, and perform tasks while obeying a list of prohibitions or guidelines. Each of these will govern different elements of how an agent will react in a particular circumstance.

6. Allowing interaction and communication between agents
   During a training session, agents may acquire information that will be useful to other agents, or a person in the simulation may wish to issue commands to the simulated agents. To accommodate these situations, there needs to be a mechanism to allow some form of communication between people and agents as well as agents and other agents.

Although virtual agents will vary due to their particular application, there will be common architectural elements to allow them to accomplish these objectives. Though research is continuing, some of these elements can be tested and demonstrated by using them to drive an animation of a prototype agent performing a particular mission. The difficulties encountered in such an animation point to areas needing further refinement and provide a guide for future research.

2 AGENT QUALITIES

As the agent performs its tasks, it will need to respond correctly to the environment, events, and agents around it. Some of these responses will come from modeling physical characteristics, but most will depend on the ways that the agent receives information, decides what tasks to carry out, and uses the skills programmed as basic actions. The qualities needed to achieve these responses include realistic modeling, high-level actions, limited perception and knowledge acquisition, predefined responses and policy, instruction acceptance, and communication.

2.1 Realistic modeling

In order to move and act in a believable manner, a simulated agent must not only incorporate reasonable geometry, but also data on human motion patterns and strength. It may also be necessary, in some cases, to incorporate dynamic simulation to some extent. Whatever the case, there are too many degrees of freedom in a human figure to base motions simply on geometry. Realistic motions tend to minimize the effort exerted by the agent, but even that is not enough to drive a complex activity such as walking [3,4].

2.2 High-level actions

Using high-level action primitives simplifies agent control and provides a more natural way to describe how an agent should behave. As long as actions do not conflict, the agent should be able to perform multiple actions simultaneously and to move smoothly from one action to the next. An example of the first case is simultaneously walking and looking at an object. An example of the second case is walking to a door and then opening it. Especially in the second case, it is important for an agent look ahead in order to behave in a reasonable manner. If it gets too close to the door, for instance, it may not be able to open it. The positioning for the agent will in fact depend on the type of door and the way in which it will be opened, which requires the ability to reason about objects in the environment.

2.3 Limited perception and knowledge acquisition

An agent may begin a simulation with incomplete knowledge about its environment, and its senses may not provide complete information about the area around it. To overcome this, an agent must augment its normal behavior with actions to acquire more information, position itself to best handle unexpected events, and maintain a system of expectations and assumptions that can be modified as it gains more information. Such a system of assumptions may also dictate what actions an agent will take in order to fill out its picture of the environment. An agent may also have to cope with perceptions that provide only general and indirect knowledge, such as hearing. In this case, some assumptions about the information itself may be required, based on the agent's mission and environment.

2.4 Predefined responses and policy

A simulated agent is not meant to be able to reason its way unassisted through a complex simulation. Instead, it carries out a set of tasks assigned to it at the start or during the simulation. However, there needs to be another set of plans to carry out in case some predefined event occurs, and a set of prohibitions against certain kinds of behaviors or situations. The first kind of plan could be a response to spotting a member of an opposing group, with considerations for the conditions under which the contact was made and what task the agent is trying to accomplish. A prohibition might take the form of a restriction from moving into an unlit area or a rule against executing a particular action when a fellow agent might be injured. In both cases, the prohibition forces the agent to avoid certain actions or at least consider other, preferable alternatives.

2.5 Instruction acceptance

As an agent goes through a simulation, it may receive instructions from a human-controlled agent. The form of the instructions will probably be from a limited, mission-specific vocabulary, from gestures, or by set signals. An agent must be able to interrupt tasks to accomplish others, know how to prioritize tasks when no explicit importance is given, and incorporate the new instruction into its current plan of action. Although the number of instructions may be very limited in a particular simulation, the situations in which they are given can make handling them very complex. [2]

2.6 Communication

As agents acquire information about their environment and accomplish parts of their mission, they may have to relay this information to other simulated or human-controlled agents. While simulated agents could exchange data in a standard format, communication with people could be much more difficult. Probable methods of communication would be through status messages in the person's field of view or even schematics and images for more complicated information that would be difficult to describe without some kind of dialogue that might be beyond the agent's capability.

3 APPLICATION

A reasonable first test of a simulated agent is to try to produce an animation through the use of the qualities described above. One such animation was recently made at the University of Pennsylvania Computer Graphics Research Lab. This animation simulated a training session where three simulated agents were lead by two human-controlled agents to rescue a hostage from three terrorists, all of whom were simulated agents. While the animation showed that the human model and motions worked well in general, it also revealed some problem areas. These included the difficulties involved in managing multiple behaviors simultaneously, recovering from extreme postures such as crouching, and providing an easy way for an agent to manipulate objects in its environment.

Work is also continuing on other elements of the simulated agent architecture. One project, named AnimNL for "Animation from Natural Language", is addressing the issues involved in the higher levels of agent control such as interpreting instructions, working with incomplete knowledge, and reasoning about the objects in the environment. The goal of the project is to construct a complete pipeline to drive an animation from natural language instructions, which will use many of the same techniques that will be needed to direct a simulated agent.

4 CONCLUSION

Although much work remains to be done to achieve the desired agent qualities, significant progress has been made in developing realistic models and producing human motions and behaviors. By addressing the different qualities that will be needed to produce a useful simulated agent, it is possible to break down the elements of the architecture into modules that can be tested and refined. Animation provides a very useful tool for determining the validity of a particular simulated agent architecture by revealing both the quality of the animation as well as the level of animator control needed to produce a correct sequence of actions.

FUNDING

Support for this research was provided by ARO Grant DAAL03-92-G-0342, as well as ARO Grant DAAL03-89-C-0031 including participation by the U.S. Army Research Laboratory (Aberdeen), and Natick Laboratory; U.S. Air Force DEPTH contract through Hughes Missile Systems F33615-91-C0001; DMSO through the University of Iowa; NSF CISE Grant CDA88-22719; and NSF Grant IRI91-17110.

REFERENCES

[1] Badler, N., Phillips C., and Webber, B. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, 1993.

[2] Crangle, C. On Saying 'Stop' to a Robot. *Language and Communication* 9(1), pp. 23-33, 1989.

[3] Ko, H. and Badler, N. *Curved Path Human Locomotion that Handles Anthropometrical Variety.* Department of Computer & Information Science, Technical Report MS-CIS-93-13. University of Pennsylvania, 1993.

[4] Ko, H. and Badler, N. Intermittent Non-Rhythmic Human Stepping and Locomotion. *Proceedings Pacific Graphics '93*, Seoul, Korea, 1993.

[5] Webber, B. and Badler, N. *Virtual Interactive Collaborators for Simulation and Training.* Department of Computer & Information Science, Technical Report MS-CIS-93-46. University of Pennsylvania, 1993.

[6] Zhao, X. *Grasp System Update - Grasp as a Motion.* Department of Computer & Information Science, Technical Report MS-CIS-93-46. University of Pennsylvania, 1993.

BIOGRAPHY

Brett W. Achorn is a Ph.D. student in computer graphics at the University of Pennsylvania through a NDSEG fellowship. His current interests are human figure animation and animation control. He received a BS in Computer Science and a BS in Mechanics from Michigan State University in 1992.

Norman I. Badler is the Cecilia Fitler Moore Professor and Chair of Computer and Information Science at the University of Pennsylvania and has been on the faculty since 1974. Active in computer graphics since 1968, his research focuses on human figure modeling, manipulation, and animation. Badler received the BA degree in Creative Studies Mathematics from the University of California at Santa Barbara in 1970, the MSc in Mathematics in 1971, and the Ph.D. in Computer Science in 1975, both from the University of Toronto. He is Co-Editor of the Journal *Graphical Models and Image Processing*. He also directs the Computer Graphics Research Laboratory with three full time staff members and about 40 students.

Development of Virtual Environments For Medical Education and Training

Andrew F. Payer (UTMB), J. Mark Voss and Laurie Spargue (LinCom)
Department of Anatomy and Neurosciences
University of Texas Medical Branch
Galveston, TX 77550

The University of Texas Medical Branch at Galveston(UTMB) and Lincom Corporation are developing an anatomical virtual environment which will enable medical students and surgeons to emerse themselves within a specific region of the human body. This technology will provide important new instructional methods for anatomy and will be the first step towards virtual surgery where complex, high-risk surgical procedures can be simulated. This would eliminate risk to an actual patient and would allow comprehensive qualitative analysis of the student's or surgeon's surgical performance. Various emergency conditions could be presented to the student to evaluate the reactions to and abilities to compensate for unexpected conditions. The instructor then could utilize the replay the surgery to enable the student to fully understand the correct procedures.

The current project is utilizing CT and MRI scan data from a skull and heart which is then converted to a complex polygonal representation utilizing a version of the Marching Cubes algorithm. Since the polygonal representations are too complex to render within current workstation capabilities, images are rendered on a frame by frame basis and stored on laser disks which can be sequenced to provided stereoscopic images. The first application is a guided tour of the human skull which will incorporate not only the virtual images but other medical images and information to provide a comprehensive medical multimedia educational system.

N96-14983

A Microbased Shared Virtual World Prototype

Dr. Gerald Pitts, Mr. Mark Robinson and Mr.Steve Strange
Trinity University
Department of Coomputer Science
715 Stadium Drive
San Antonie, TX 78212

INTRODUCTION

Virtual reality (VR) allows sensory immersion and interaction with a computer-generated environment. The user adopts a physical interface with the computer, through Input/Output devices such as a head-mounted display, data glove, mouse, keyboard, or monitor, to experience an alternate universe. What this means is that the computer generates an environment which, in its ultimate extension becomes indistinguishable from the real world. "Imagine a wraparound television with three-dimensional programs, including three-dimensional sound, and solid objects that you can pick up and manipulate, even feel with your fingers and hands... Imagine that you are the creator as well as the consumer of your artificial experience, with the power to use a gesture or word to remold the world you see and hear and feel. That part is not fiction... three-dimensional computer graphics, input/output devices, computer models that constitute a VR system make it possible, today, to immerse yourself in an artificial world and to reach in and reshape it." (2). Our research's goal was to propose a feasibility experiment in the construction of a networked virtual reality system, making use of current personal computer (PC) technology. The prototype was built using Borland C compiler, running on a IBM 486 33 MHz and a 386 33 MHz. Each game currently is represented as an IPX client on a non-dedicated Novell server. We initially posited the two questions: 1. Is there a need for networked virtual reality?; and 2. In what ways can the technology be made available to the most people possible?

WHAT SHARED VIRTUAL REALITY HOLDS

With more and more people sharing information from remote sites, virtual reality is a perfect marriage between the person to person interface and the person to information interface. Using an earlier example, if two architects designing a building were separated geographically, they could meet in a virtual reality model of the building, walk through it together, and discuss the pros and cons. They could also "summon" other sources of data, utilizing applications and resources native to their own computers. People will be able to interact as if they were in the same room together without the necessity of physical proximity. "Communications has improved because the nuances of spoken language are transmitted instead of the impersonality of Morse code. Shared virtual reality can make the world a smaller place and at the same time provide an unlimited world to explore." (3). Architects can design 3D blueprints of buildings and use a VR simulation to test the ergonomic and structural attributes before construction begins. This will increase safety, accelerate construction, and help prevent costly mistakes in design. Biology students could study human anatomy, stripping away flesh, bone, and muscle to peer at the inner workings of the body, without touching a corpse. "The shift of focus from working with alphanumeric data to working in a three-dimensional representation of that data can alter dramatically both the manner in which we work with computers and our productivity and enjoyment when working with them."(1).

THE PERSONAL COMPUTER SOLUTION

Looking at the promise that virtual reality holds for so many wide-open areas, it seems only logical that its power be made available to as many people as possible. However, the average consumer cannot afford the $15,000 for the a head mounted display, data glove, and other virtual reality equipment typically used for a VR station. For a shared experience, add at least this amount to include another station and networks. The tightly controlled budgets typical in today's academic, business, and government institutions should look closely to insure that such costly hardware is needed. An economical answer to shared virtual reality could be found in the form of personal computers. Current PC technology possess the computational power necessary to generate and maintain realistic, believable virtual worlds, as demonstrated in PC software such as Ultima Underworld or Alone in the Dark. PC's are affordable and accessible, easily holding the largest percentage of the computer market.

One of the most important factors in supporting PC based virtual environments is that we believe that immersion into a virtual world is a product of the mind. It is precisely because of this mental immersion that pulling away from a good book is so difficult. People do not depend on expensive hardware to immerse the mind. Are video games popular because of HMD's or Gloves? It is the game itself and often the ability to interact with another in the particular electronic world. Sharing a computerized experience greatly enhances immersion into the world because human beings are used to interacting with others on a daily basis. This is an integral part of what we term "reality". An individual does not have to rely on the expensive specialized hardware to have a meaningful virtual reality experience. It was this realization that sparked the construction of our prototype.

PROTOTYPE DETAIL

The Reality Forge involves separate computers running individual VR application shells, each configured to suit the computer's own unique hardware. A chief element of the system is its modularity, in the sense of hardware transparency. In other words, if one user had a head-mounted display device, the VR shell could be configured to utilize that piece of equipment, while another user might only have a black and white monitor. This would not change how the users interact in the virtual environment. The core of the VR shell is a virtual reality processor that receives transaction requests from some source (input from the local user or data packets from a remote user), interprets the request, and generates an appropriate response (update the visual image through coordinate transformations, assemble and send a data packet to a remote user, etc.). (See Figure 1). The virtual reality processor directly accesses the object coordinate database in order to translate the three dimensional world to a two dimensional screen and to initiate coordinate updates that will be reflected in all remote databases. The Data I/O Handler controls communication between parts of the shell as well as assembling and disassembling data Packets to be transmitted over the Novell network to other VR shells.

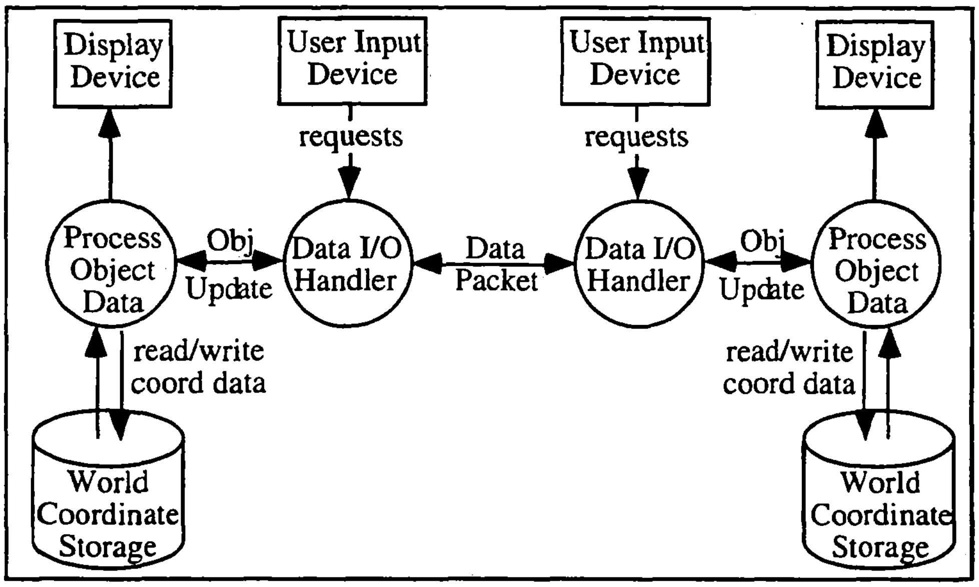


Figure 1 Simplified diagram of The Reality Forge

It was important for our VR system to be as "generic" as possible. There are two reasons for this: 1) Portability of the shell for ease of transference between platforms and uniformity of VR processing techniques. This allows users to upgrade the shell easily as computing power increases. These upgrades will include graphic enhancements (better shading techniques, shadows, etc.), full stereo sounds, and other equipment as the technology evolves (for example, tactile feedback equipment). And 2) Transparency of the VR processor promotes easier application development. (See Figure 2). For example, the implementation of a simple tank game on top of the VR shell took approximately 20 work hours.

Obstacles

It is important to realize that some of the methods employed in this project are not necessarily the most efficient. The program code is the physical manifestation of the actual research, its goal, as stated before: to explore the possibility of a PC-based networked virtual reality.

Graphical Representation of the World

Human beings can see through the reflection of light off objects into the eye. Light rays bouncing off of objects behind the person do not enter the eye and are not seen. Seeing is not so simple for a computer. To simulate human sight, only what a person standing in a real world would see would be rendered by the machine. Objects, especially those which are positioned directly behind the camera wreak havoc with the image if drawn to the screen. To help the computer to draw only what is in this viewing area, objects can be divided into three categories: wholly included, wholly excluded, and straddling. Wholly included objects can be entirely seen from the vantage of the person in question. Wholly excluded are those objects which are entirely out of sight. Straddling objects are those which can be partially seen, such as a long wall view from the middle, looking to one end. The unseen portion of the object must be mathematically clipped away where the object intersects the outside of the viewing area, leaving only wholly included objects. The computer then sorts these objects based upon how far away they are from the person. The most distant objects are drawn first by finding which faces point toward the viewer and filling them in with the specified face color. Closer and closer objects are drawn in this manner until all of the objects have been drawn. Drawing objects from farthest away to nearest gives a realistic rendering where objects in the foreground obscure objects in the background. This method for depth sorting is known as the Painter's algorithm. (4).

Collision Detection

Running into things may seem very natural to some people, but collision detection can provide some tricky problems. Pretend we have two cubes floating in a virtual space and we wish to see if they have collided. The accurate way to do this would be to check and see if any of the vertices have intersected any of the faces on the other cube. (cube 1,vertex 1 with face 1,2,3,4,5,6; vertex2 with...cube 2,vertex 1 with face 1,2,3,4,5,6...) This is seventy-two complicated three dimensional geometric checks to see if a collision has occurred each time either moves. A faster method would be to surround each object in the world with a boundary sphere. Each time an object moves the distance between it and other objects are computed. If the distance is less than the sum of the objects' radii, the objects have encroached upon each others' boundary spheres and therefore, have "collided".

Object Data Structure

A suitable data structure for virtual objects is essential to allow for future extension as well as provide efficient access to the object data. As the power of the VR system grows, so does the complexity of the objects in the virtual world. Accessing object data (relative coordinates, etc.) must be quick to provide fast calculations and re-draws. Their is the possibility of data structures needing frequent updating of object responses (a projectile becoming a land mine) as well as fast changes in shared views of the world. Time based movement and rotation ensured objects seen on different speed machines will move at the same rate per second. Objects which are not assigned to represent users may have a prescribed set of responses to certain actions. If a user touches a sphere in a virtual space, the sphere could move away from the user (a transformation change), turn a different color (a change in representation), or both. Object responses are encapsulated within the object's structure as pointers to mathematical time-based functions describing motion. (the projectile shell contains a pointer to a function which describes gravity)

Object Databases

One of the trickiest problems faced in the design of a shared virtual reality system is the physical location of the world database and insuring consistency among all users. Centralization of the object database necessitates each user pulling the world data from a central location, thereby increasing the likelihood of network bottlenecks and data collisions. A distributed database system would greatly reduce the amount of network traffic but would also pose the difficulty of requiring a data synchronization scheme. The Reality Forge utilizes distributed object databases native to each user's shell with a simple synchronization technique to ensure uniformity of object motion and interaction.

Communication Between Computers

To facilitate the VR systems networkability between different platforms and allow users to take full advantage of their existing hardware, a widely implemented communication protocol must be employed. Netware IPX was initially chosen for its ease of use, but as of February, it was decided to eventually switch to TCP/IP, a widely-used cross-platform protocol. Because this protocol is a connectionless communication service, data exchange between machines was accomplished through the passing of packets. Loss of data packets from queue overflow and overhead of assembly/disassembly compounded the synchronization problem between shells. Therefore, the synchronization algorithm had to be extended to compensate for this to keep the multiple databases "looking" similar.

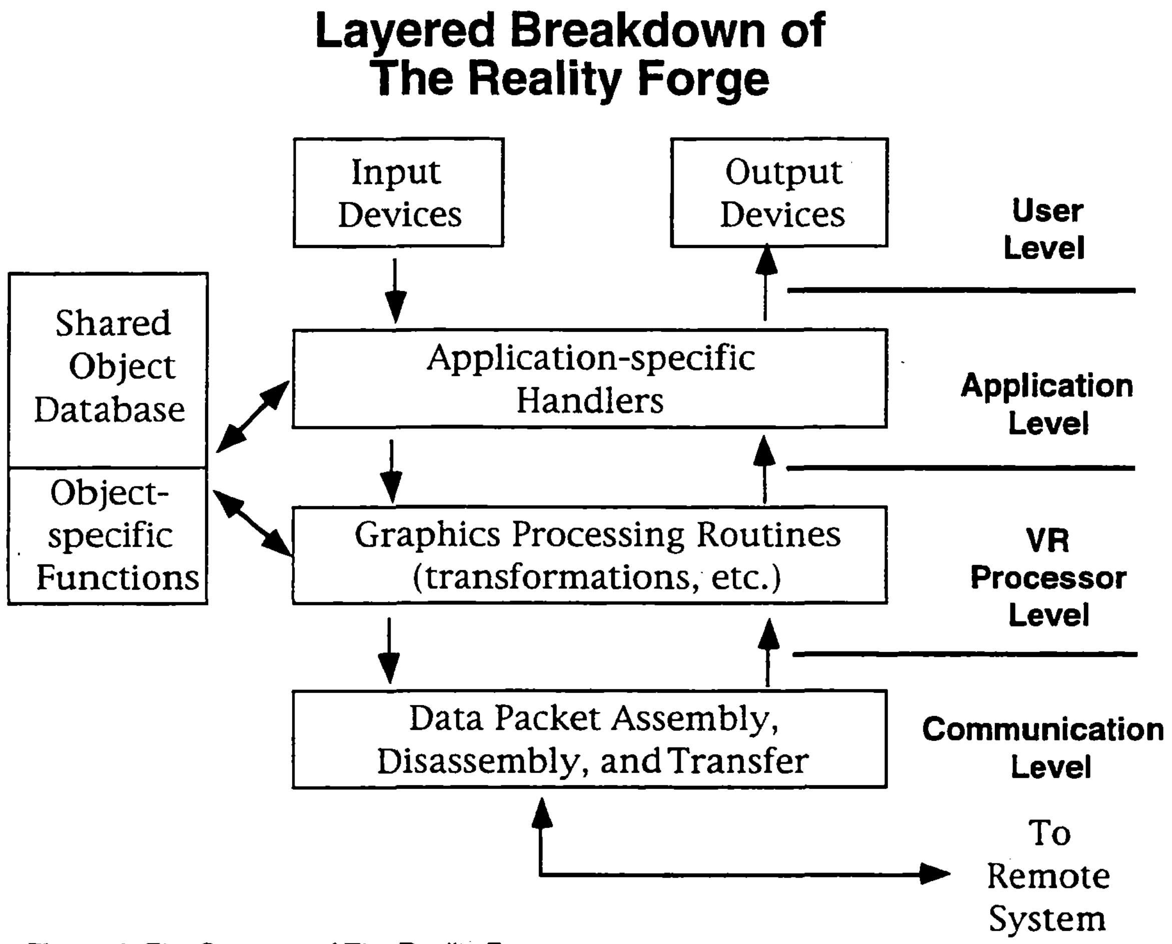

Layered Breakdown of The Reality Forge
Input Devices
Output Devices
User Level
Shared Object Database
Object-specific Functions
Application-specific Handlers
Application Level
Graphics Processing Routines (transformations, etc.)
VR Processor Level
Data Packet Assembly, Disassembly, and Transfer
Communication Level
To Remote System

Figure 2 The Structure of The Reality Forge

Multi-user Interaction

When multiple, independent users interact in a shared virtual world, conflicting actions can be generated. For example, if one user wishes to "grab" an object at the same time another user tries to do the same, then a conflict arises and some solution must be determined. As the number of users increase, so does the likelihood of action conflicts. We have employed a simple conflict resolution routine where one of the shells is designated as an arbitrator. A VR shell will react to a user request for movement or possession of an object only after attaining permission from the arbitrator. Permission is granted to the request received first.

CONCLUSION

If virtual reality is a human/computer interface, then networked virtual reality serves as a human/human interface, providing an incredible interpersonal communication tool. This shared virtual reality presents problems far beyond those associated with just the linking of machines. This project has been an exploration into the difficulties involved with such a task, as well as the "ground up" development of a PC-based VR system (dubbed The Reality Forge). Networked virtual reality will reach its full potential in the very near future and in order for it to achieve maximum accessibility, it must be geared toward the resources of the current average user. Armed with the results of this experiment, we believe that this is certainly an attainable goal. Keeping virtual reality financially exclusive will do nothing but dwarf interest and investment into it. In order to avoid this crippling mistake, networked virtual reality must be built around technology that is both affordable and accessible.

BIBLIOGRAPHY

1. Aukstakalnis, Steve. *Silicon Mirage.* Berkley, California: Teachtit Press, 1992.

2. Rheingold, Howard. *Virtual Reality.* New York, New York: Simon & Schuster, 1990.

3. Teixeira, Kevin. Kevin Pimentel. *Virtual Reality: Through the New Looking Glass.* New York: McGraw-Hill, 1993.

4. Van Dam, A. J.D. Foley. *Fundamentals of Interactive Computer Graphics.* Reading, Massachusetts: Addison-Wesley, 1984.

N96-14984

Virtual Environment Architecture for Rapid Application Development

Dr. Georges G. Grinstein, David A. Southard, J. P. Lee
The MITRE Corporation,
Bedford, MA 01730-1420

ABSTRACT

We describe the MITRE Virtual Environment Architecture (VEA), a product of nearly two years of investigations and prototypes of virtual environment technology. This paper discusses the requirements for rapid prototyping, and an architecture we are developing to support virtual environment construction. VEA supports rapid application development by providing a variety of pre-built modules that can be reconfigured for each application session. The modules supply interfaces for several types of interactive I/O devices, in addition to large-screen or head-mounted displays.

Key words: Virtual reality, virtual environments, software, architecture, prototyping.

INTRODUCTION

One of MITRE's duties as a Federally Funded Research and Development Center (FFRDC) is to objectively evaluate and compare current technologies, and to recommend courses of action for numerous government programs. As such, we have actively been involved in assessing workstation, graphics, and user-interface technology. Since these areas form the basis for virtual environment technology (VET), we are in a good position to further explore the intrinsics and the impact of this new technology [1-3]. We are currently developing MITRE's Virtual Environment Architecture (VEA), to be used as a foundation for several applications in battle management, mission planning, electric utility simulation, and medical support [4,5].

VEA supports event-driven simulation, using a real-time clock with variable time expansion, and a flexible mechanism for integrating application-specific modules, such as knowledge bases, planners, and simulation models. It is written in C++ on a Silicon Graphics RealityEngine system, and executes object modules in parallel when multiple processors are available. VEA can support a wide variety of user interface devices. Voice, gesture, aural, and visual interactions are supported for the creation of multimodal,
multisensory, and highly interactive environments. Visualizations can be presented on large screen or head-mounted stereoscopic displays.

Requirements for rapid prototyping

Rapid prototyping of environments, objects, behaviors, synthetic tools, and evaluation of new peripheral devices is an important part of the virtual environment design process. Virtual environments are typically very large entities, comprising diverse source code for graphics, expert systems, knowledge bases, and peripheral device drivers. Distributed systems for collaborative activities further complicate the integration of new technologies. As software developments and hardware offerings continue forward, a virtual environment application must be able to accommodate rapid change. Requirements for rapid prototyping are needed to face the larger domain of requirements placed on virtual environments, and are critical to the overall solutions of the problems VET places on developers.

In a comparative study of four virtual environment architectures (including VEA), Masterman [6] analyses the functional requirements for virtual environments, and the solutions offered by each of the representative systems. The functional requirements are: *interactive response* times to user inputs; *multiple interface devices*, and *multiple modes* of input and output; *distributed processing; easy integration* of application code; *extensibility* to new interfaces and applications.

System usability ultimately depends on minimizing the latency between user actions and their manifestation in the virtual environment. Interactive response must always be maintained in the environment. This is complicated by the requirement that multiple devices of a multimodal and multisensory nature are the rule in such applications. Distributed computing is inevitable, which means that integration of additional systems and devices occur across multiple, possibly remote platforms. Ease of integration allows maximum portability and must consider the addition of separate, autonomous application code written in heterogeneous languages. Finally, ease of extension is required for the evaluation of new concepts and products without a large programming effort.

These requirements are accommodated by several common design themes. Object-orientation results in a highly modular environment, and it addresses extensibility and ease of integration. Object behavior can also be easily encapsulated. Mechanisms for parallelism, in line with present and future hardware capabilities, address the interactive response and distributed computing requirements. Asynchronous, message-based communication results from the object-oriented and parallelism approaches, and it is necessary for distributed computing. The layering of device abstraction further assists the integration and extension issues, as the device interface must accommodate change because of the continual introduction of new devices for virtual environments.

Why a new architecture?

We wanted to take specific approaches to each of the design themes. On the basis of our experiences and observations, these approaches seemed best:

Object Orientation. VEA is implemented in C++. The kernel, devices, and abstract domain objects are all treated as autonomous objects. Most system protocols are implemented high in the class inheritance hierarchy, so that object instances derive most of their systems behavior from their base classes. In contrast, for example, VEOS (Virtual Environment Operating Shell) is written in primarily in XLisp, with application and interface modules in C [7]; Division's dVS uses an object-oriented C library interface [8]; and IBM's VUE (Veridical User Environment) uses a rule-based dialog manager [9].

Parallelism. VEA assumes a tightly coupled, shared memory, multiprocessing model of computation. In this respect VEA differs from most other architectures, which assume a looser configuration, distributed over a local area network. VEA's approach is consistent with current super graphics workstation architectures (as exhibited by Silicon Graphics, for example), as well as our perception of architectural trends. Network distribution will be supported in the future by adding network adapter software modules. Contrast VEOS with its loose, heterogeneous model; dVS with its specific selected hardware platforms; and VUE, similar to VEOS, with a loose and heterogeneous model.

Message-Based Communication. VEA defines two modes of communication: messages and events. Messages are synchronous (blocking); events are asynchronous (non-blocking). Both messages and events can be executed in parallel.

Layering of Device Abstraction. VEA defines two layers for device abstraction: *filters* and *managers*. Filters provide an interface between devices and generalized I/O events. Managers transform the I/O events into higher-level events.

Integration and extensibility

There are several aspects of VEA that provide for ease in integration and extensibility. New sessions with alternate devices, displays, and users can be reconfigured without recompiling by selecting a different configuration file or by altering any of the environment parameters. Object knowledge and behaviors are independent and separated from the interaction style. This provides the capability of having dramatically different environments. We have provided templates for generic device classes that support quick integration of new devices: most of the interface is provided as boilerplate. Finally we have developed VEA with the intent of eventually supporting multiple users. There is much work to be done in understanding the ramifications of users sharing a design space. For example, what happens when two users grab the same object and attempt each to modify it in different ways?

HARDWARE AND SOFTWARE PLATFORM

We decided to target super graphics workstations from Silicon Graphics, since we are familiar with their products, and because it meets the requirements for a high-performance graphics rendering across a wide product line. The RealityEngine graphics option provides hardware texturing and anti-aliasing support. In principle, at least, portability can be achieved by replacing the graphics module with alternative code, as all platform-specific graphics code resides in one object module.

I/O Devices

After having evaluated numerous devices and software packages, we realized that applications require a variety of I/O devices; however, software to support them is scarce. Table 1 summarizes the devices we are currently working with, which we have categorized into generic classes. These devices were evaluated and their limitations characterized through experimentation and empirical observations in our virtual environment laboratory. Software drivers have been written as necessary.

Table 1. I/O Device Categories

Class	*Examples*
Posture	VPL DataGlove, Exos Dexterous HandMaster
Locators	Polhemus, Ascension Flock of Birds, SpaceBall, Global 3D Controller
Pointers	mouse, Logitech 2D/6D mouse, Origin Instruments DynaSight tracker
Graphics	Large-screen dual-projector stereoscopic display, Virtual Reality Group head-mounted display, SGI RealityEngine
Tactile	Exos tactile display
Speech	Voice Navigator (input), Voice Impact (output)
Audio	IDI synthesizer
Text	keyboard
Button	buttons box, mouse, SpaceBall
Valuator	dials

For voice I/O we are using inexpensive commercial products such as Macintosh-based Voice Navigator for voice input. Even with its speaker-dependent limitation it provides for an interesting integration demonstration. We plan to use Voice Impact for output. It records and generates voice messages. We also plan to experiment with a MIDI sound synthesizer for non-speech audio output.

Software tools

Numerous software tools were acquired and evaluated. We selected the following tools: C++ for the object oriented programming, Software Systems' MultiGen for object modeling, IRIS Performer for visual simulation support, and NASA's C Language Integrated Production System (CLIPS) for the knowledge-bases, which will be used to model intelligent object behavior. IRIS Performer provides excellent rendering

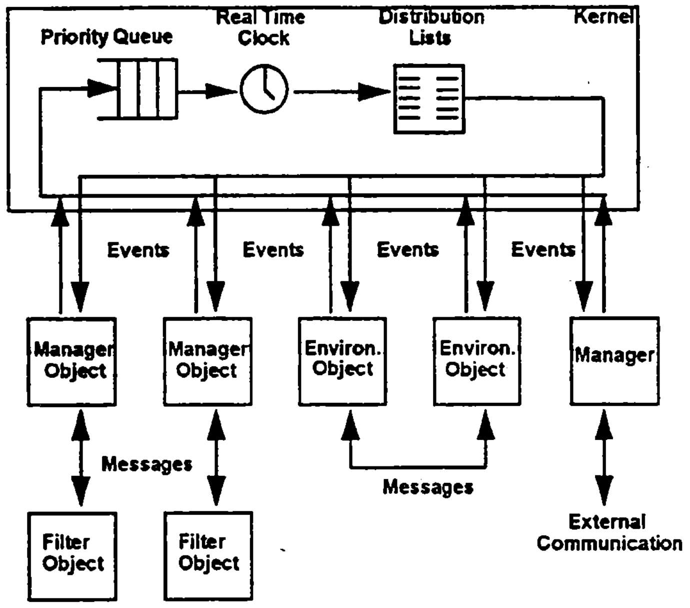


Figure 1. VEA Overview

performance on the whole line of workstations, takes advantage of multiprocessing capabilities on multi-CPU systems, and is the most cost-effective among the visual simulation toolkits on the market. CLIPS provides an object-oriented knowledge representation language, as well as a rule-based production system, both of which are easily accessible from C and C++.

THE VIRTUAL ENVIRONMENT ARCHITECTURE

VEA uses events for representing certain kinds of simulated events and communications between simulated objects. The VEA kernel uses UNIX system facilities to implement its event-handling mechanisms. Figure 1 provides an overview of the event mechanism in the kernel. Events and objects are initialized from a configuration file. Events are placed in a priority queue, ordered by time. Objects are placed in the object list. When an object is initialized, it subscribes to the types of events that it is capable of accepting. The subscriptions are placed in the distribution lists, which are organized by event type. Objects may subscribe, or drop a subscription, at any time during the simulation.

Events

Events have properties that distinguish them from the more general concept of messages. Object-oriented languages such as C++ support messages as *member functions,* or *methods.* VEA contains additional facilities that support events. Events are characterized by the following attributes:

The primary purpose of an event is to effect a global state change.

Events occur in time. Conceptually, all objects receive events simultaneously.

An event can be processed by any object that registers to receive it. Each object interprets an event as appropriate to its function. A sender does not know a priori who will receive its event.

C-2.

Events may be initiated by the users of the system, by simulated objects, or by external influences.

Events are not elements in a communication dialog or protocol.

Event types include: button, gesture, graphic, locator, pointer, position, posture, sound, speech, text, valuator. The intention is to keep the number of events small, and to keep their meanings as flexible as possible. There may be some overlap between event types. For example, a posture event could be represented by a kind of button event. For convenience, however, a separate posture event allows a manager object to monitor sequences of postures for possible gestures. This design realizes the layering of device abstraction.

Clock

VEA contains a real-time clock, which is implemented with the UNIX interval timer. A time expansion factor is provided, which allows the simulation to run faster or slower than real-time. Most objects will not need to access the clock itself. The objects usually work with the time stamp reported in the events that they receive.

Multiprocessing

When the simulation clock reaches the time indicated by the event at the top of the priority queue, the event is dispatched for distribution to the objects. All objects on the distribution list for that event type receive a pointer to the event. Each object can then process the event, according to its own interpretation. Each object is run in its own process. In principle, all objects may proceed simultaneously. In practice, parallelism is limited by the number of processors available, as well as by other resources shared by all processes. A separate "lightweight" process is created for each event. These processes remain in existence only as long as necessary to process each event. Access to common data structures is synchronized using UNIX system semaphores, for which we created a C++ class interface.

As each object computes, it updates its internal state to reflect the consequences of the event. This may involve a message dialog with the sender of the event. The result of the processing might be new events added to the priority queue, or it might simply update an object's internal state. When an object completes its processing, the object becomes free to process another event, if another one is waiting. The kernel deletes an event when all recipients have released it.

Software Bus

In our prototypes, there is a need to use previously developed, stable tools. Such modules include planners, schedulers and simulation models. We called these tools *external modules* and our intent is to provide a mechanism, termed a *software bus*, by which such modules can easily be integrated within VEA. The first example of a external module we integrated is a CLIPS knowledge base. It is anticipated that the soft[ware bus will support a distributed interface for a wide variety of simulation models.

CURRENT EFFORT

Our current application is aimed at battle management. This application is intended to provide situation awareness to commanders, who must make critical decisions quickly. In addition, the system should facilitate the decision-making process directly. That is, the system should provide tools that enable the decision-maker to probe the current situation, obtain needed information, and assess the implications of various options. Someday, such an environment might host intelligent agents, which could suggest alternative courses of action, and once a decision is made, actually begin to implement those decisions in the real world.

A central aspect of this application is a detailed terrain scene. The terrain model will represent not only the terrain relief, but also cultural and natural features, such as buildings, roads, crop lands, rivers, and forests. Some of the stationary features will be militarily significant: supply depots, defense installations, etc. The scene will be populated with numerous moving vehicles, such as trucks, tanks, and aircraft. The vehicles could represent objects derived from real-time tactical data links, or they might be simulation models. The decision-maker will have the ability to move through the scene, to place himself at any viewing position on the ground, or in the air. The viewing position could be attached to any moving vehicle, so that the viewpoint tracks the vehicle as it moves.

The user will be able to designate objects, and to query them for identification, history, and any intelligence information that might be available. In a simulated battle scenario, the information will be dynamic, as new reports come in about each object. For simulated scenarios, it is important that the simulated objects move and behave as they would in reality. For example, trucks stay on roads, tanks avoid lakes, aircraft avoid the ground, and enemy forces engage or avoid each other, depending on their rules of engagement, and the current situation. The user may wish to directly reconfigure various assets, then have mobility and cost models advise whether the new arrangement is feasible, how long it would take to reach the new configuration, and what the costs would be.

Another aspect of this application is the *what if* scenario. In this case, it will be desirable to set-up simulations with several different initial conditions, then to view a simulation at faster-than real-time speeds. Conversely, if too many things are happening at once, the user may wish to slow down the scenario, so that all the object interactions can be observed.

The simulation and database systems needed to support such a scenario do not currently exist as integrated systems, but parts of these capabilities do exist throughout the military. Our job is to begin to pull the pieces together into a useful system.

The user of this application is quite different than envisioned for most virtual environments. A command officer will not be willing to suit-up in cumbersome, restrictive apparatus. There will be no opportunity to train users how to use the equipment. The devices used must be natural and easy to use, and the user interface must be intuitive.

Potential Enhancements

We have a number of areas in mind for expansion. Our immediate plans are to expand from a single-platform, multiprocessing system, to a networked, multi-platform, collaborative system for multiple users. Virtual environments represent a technology by which remote teams can work together on science and engineering problems. Following that, we envision adding intelligent agents. These agents could monitor the environment, and notify the user when certain situations arise, or carry out tasks on behalf of the user.

We would like to include support classes for physically based modeling. This would include basic Newtonian dynamics and collision models. This would allow us to create many interesting virtual environments, in which objects behave as one would expect. Many object behaviors are completely mechanical, so there is no need for the knowledge base to intervene in these cases.

Another potential area includes advanced artificial intelligence information systems. Many applications exist for advanced user interfaces for schedulers, and reactive and adversarial planners. MITRE has developed context dependent natural language parsers and our intention is to eventually integrate these tools as well.

FUTURE PROSPECTS

Worthy applications for VET are real, and are here today [10,11]. However, there is much engineering work needed to realize the potential. The immediate challenge is to construct an infrastructure that supports a wide variety of VET investigations. VEA does this by taking a modular, object-oriented programming approach. Input and output devices are embedded in filter and manager layers, which encourages device and application independence, and flexibility. Simulation and modeling are supported through the integral real-time clock, and the knowledge base. Rapid prototyping is supported through the use of flexible configuration files, which allow each VEA session to be tailored for a different use. High performance is obtained with built-in support for concurrent multiprocessing.

Power Utilities

MITRE is actively participating in the development of a simulated control panel for a fossil-fueled power plant control room. Power utility companies throughout the country must train and rehearse power plant operators in correct procedures. Currently, control room mockup trainers are constructed full-scale from the actual devices. Such trainers are very expensive to build, operate, and maintain. Virtual environments are a low-cost alternative that can be tailored to the needs of individual power plants. In this application, integration with a proven numerical simulation is critical.

Another application targets training for high-voltage switch and transformer yard repair procedures. Severe accidents, resulting in deaths, are currently a reality for many power utilities. Training and refresher courses in proper safety procedures are essential to reduce these occurrences. Virtual environments may offer a way to perform such training effectively, without exposing the operator to hazardous conditions.

Health

A recent conference focused on how VET could help persons with various disabilities [12]. For example, a person with Parkinson's disease often has an involuntary shaking in the hands, but is unaware of this motion, unless he looks at his hands. A glove device, however, can be programmed to filter out the involuntary motion, and present a stable representation of the hand. Thus, the presentation of the hand would match the users mental image. This could enable such persons to perform normal activities in the virtual environment, that would be difficult to perform in a real environment.

A related application for VET is as a flexible platform for psycho-physical experiments, on earth as well as in space. A well-designed and integrated suite of hardware and software could replace several specially-designed experimental apparati.

ACKNOWLEDGMENT

The work was sponsored in part by The MITRE Corporation, in addition to primary sponsorship by the Rome Laboratory, Griffiss AFB, NY, and the Electronic Systems Center, Hanscom AFB, MA, under contract #F19628-89-C-0001.

REFERENCES

[1] D. A. Southard (1992), Transformations for Stereoscopic Visual Simulation, *Computers & Graphics* 16(4) 401-410.

[2] R. B. Mitchell, J. L. Segal (1992), A Virtual Maintenance Trainer, *SID Digest 23* 906-908.

[3] P. J. Hezel, H. Veron (1993), Head Mounted Displays for Virtual Reality, *SID Digest 24*, Paper No. 41.3 (to appear).

[4] G. G. Grinstein, R. B. Mitchell, D. A. Southard (1993), Virtual Reality: An Interface Architecture for Interactive Simulations, *Proc. Soc. Computer Simulation* (to appear).

[5] D. A. Southard, J. P. Lee, R. B. Mitchell, G. G. Grinstein (1993), Case Study: A Virtual Environment Architecture (submitted to *IEEE Symposium on Research Frontiers in Virtual Reality*).

[6] H. C. Masterman, G. G. Grinstein (1993), Software Requirements for Virtual-Environment Applications, *SID Digest* 24, Paper No. 35.1 (to appear).

[7] G. Coco, *VEOS 2. 0 Tool Builders Manual,* Human Interface Technology Laboratory, University of Washington, May 1992.

[8] *Provision Software Overview,* Division Inc., Bristol, UK, 1991.

[9] P. A. Appino, J. B. Lewis, L. Koved, D. T. Ling, D. A. Rabenhorst, C. F. Codella (1992), "An Architecture for Virtual Worlds," *Presence: Teleoperators and Virtual Environments* 1(1).

[10] P. T. Breen (1992), *Near-Term Applications for Virtual Environment Technology,* M92B0000011, The MITRE Corp., Bedford, MA.

[11] P. T. Breen (1992), The Reality of Fantasy: Real Applications for Virtual Environments, *Information Display 8(1 1)* 15-18.

[12] *Proc. Birtual Reality and Persons with Disavilities Conf.,* California State University Northridge, 18-21 March 1992, Los Angles, CA. 10.

A Development System for Multi-User, Distributed Virtual Environments

C.F. Codella, R. Jalili, L. Koved, J.B. Lewis
Virtual Worlds Group, Comp. Science Dept.
IBM T.J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598
914-784-7511 914-784-7455

This paper reviews the design and operation of the VR Toolkit developed at IBM Research. Ease of use was emphasized both for the application builder and the toolkit extender. The system supports distributed processing, the building of multi-user shared environments, as well as a variety of specialized I/O devices such as gloves, 3-D position sensors, sound generation, speech recognition, and 3-D graphics displays, under an open and extensible architecture. Virtual world environments are created using a mixed object-oriented and event based paradigm for defining system behavior. Basic units, called modules, represent entities in the world such as objects, operations, functions or users. Modules communicate with each other by producing and consuming events, and are defined at a high-level using rules written in C++ that determine how events are handled. The VR Toolkit, designed to be run on one or more workstations, includes a development environment consisting of C++ class libraries for module construction, interprocess communication, device support, and hierarchical object-oriented graphics. The run-time environment includes an X Window System based control panel for dynamically constructing worlds from a collection of modules, allocating processes among hosts, editing modules, and monitoring operation, and a library of ready to use modules for various devices and common operations.

N96- 14985

Rapid Prototyping 3D Virtual World Interfaces within a Virtual Factory Environment

Charles Paul Kosta, Dr. Patrick D. Krolak
Center for Productivity Enhancement
University of Massachusetts Lowell

ABSTRACT

On going work into user requirements analysis using CLIPS (NASA/JSC) expert systems as an intelligent event simulator has led to research into three-dimensional (3D) interfaces. Previous work involved CLIPS and two-dimensional (2D) models. Integral to this work was the development of the University of Massachusetts Lowell parallel version of CLIPS, called PCLIPS. This allowed us to create both a Software Bus and a group problem-solving environment for expert systems development.

By shifting the PCLIPS paradigm to use the VEOS messaging protocol we have merged VEOS (HITL/Seattle) and CLIPS into a distributed virtual worlds prototyping environment (VCLIPS). VCLIPS uses the VEOS protocol layer to allow multiple experts to cooperate on a single problem.

We have begun to look at the control of a virtual factory. In the virtual factory there are actors and objects as found in our Lincoln Logs Factory of the Future project. In this artificial reality architecture there are three VCLIPS entities in action. One entity is responsible for display and user events in the 3D virtual world. Another is responsible for either simulating the virtual factory or communicating with the real factory. The third is a user interface expert. The interface expert maps user input events, within the current prototype, to control information for the factory. The interface to the virtual factory is based on a camera paradigm. The graphics subsystem generates camera views of the factory on standard X-Window displays. The camera allows for view control and object control. Control or the factory is accomplished by the user reaching into the camera views to perform object interactions. All communication between the separate VCLIPS expert systems is done through VEOS.

1. INTRODUCTION

This work is divided into three broad sections covering rapid prototyping, the virtual enterprise, and the VCLIPS architecture. Rapid prototyping is one example domain for VCLIPS development. In this paper we will examine a factory floor enterprise. VCLIPS will be used as both an event-based simulation and a control environment for the factory. The virtual factory section will explore the various aspect of' the physical factory and the identical virtual factory. The final section presents the VCLIPS architecture and related research topics.

2. RAPID PROTOTYPING

A prototype is an original type, form, or instance that serves as a model on which later stages are based or judged.

User interface prototyping has evolved into four distinguishable levels. The first level is that of painted or hand drawn diagrams representing static screens which the user views. The second Level is that of prototyping responses, where objects on the screen are no longer simply pictures; they are objects that react to the user. This is usually done though the use of' 'C' code or some script-like language. The next Levels is that of dynamics.

Dynamics includes both responsiveness to user events and simulation of user scenarios. Dynamic systems act as close to the real system as possible. The highest Levels of prototyping contains everything in the previous three levels plus the ability to capture and report usage metrics.

The function of prototyping is to demonstrate that a model serves a useful purpose. At the first Level, one tries to find out if the screens portray the correct meaning. The second level asks whether the prototype can respond in an intuitive manner. The third Level utilizes scenarios which in turn simulate events that the user will encounter on the real system. The highest level prototype would include measures (metrics) of the usefulness of the prototype. It is important to note that the first three levels also have metrics; but they are not integrated into the prototype - they are external: e.g. surveys, video taped sessions, subjective comments of the user community.

2.1 The Graphical Display

Today's prototypes not only deal with data models, but also with user models - an example is the use of icons. Icons must somehow depict a similar meaning for all users. Concern for user models inspire the larger role that windowing systems are playing in today's computing environments. Specifically, methods are being created to

specify how information is distributed into windows, such as the *Motif Style Guide.* These methods show how important icons can be for the users' understanding..

New techniques are still being developed [FB89], striving to go beyond the framework of windows of information into what has been termed widgets. A widget is an abstract graphical representation, in the form of an icon or glyph, that provides movement or actuation on some object. An example would be a sliding bar widget. It would look like the sliding bars used on stereo equipment. The user could select the slide bar with the mouse and move it along the axis to set or adjust some scalar value. Widget complexity is only limited by the creator's imagination. They can be as simple as a small radio knob dial or as complicated as the entire front panel of a virtual computer.

In general, prototyping systems are increasingly becoming object oriented [ZCW+91][FB89] Widget items acquire object properties. These properties can then be linked to widget functionality on the display. When an object value changes the corresponding widget can be updated. The objects can be displayed in two or three dimension.

2.2 Object Rendering

In general, a display can be viewed as having multiple objects expressed on the screen using some known output technique. This could be a simple table of integer values or a screen full of text. Each displayed objects rendered onto the display screen. For example consider rendering a *real* number. Doing so might produce the numerals on the screen that represent the real value, or, one might till in the picture of a thermometer with pixels that let the user read the real value from the thermometer [figure 1]. In particular, every object is rendered in some manner. Object rendering is usually based on an objects function - is it a number - is it a structure representing a workstation - is it a file - is it a disk drive'? one application development systems that allow for interactive creation of widgets is NASA's TAE+. Another example would be VAPS, a complete environment for designing real-time interfaces.

It is important to consider the user model as a guide to object rendering. Current windowing systems allow the designer to choose different techniques in the management of both windows and objects. The three main types of windows are *tiled, overlapping,* and *pop-up* windows. Tiled windows are those that split up the screen into smaller tiles such that no window ever covers up another. The tile model is based upon the user's ability to deal strictly with spatial relationships. overlapping windows allow for the possibility of data being covered up; however, it usually comes with the ability to resize, move and place the windows over one another. In user model terms, overlapping windows represent the "desktop" paradigm.

Pop-up windows are interesting in that they can represent a user model that goes beyond the "desktop" into models that are based on a virtual technical assistant. The assistant works with the user's "desktop." In particular, current pop up windows are used for: [1] displaying a message about the system that must be dealt with immediately (e.g. someone putting a high priority memo on your desktop), 121 presenting a menu that represents either local or global choices about the window below it, [3] creating computerized *post up notes* that are attached to objects until you peel them off and throw them away. These pop up examples represent items that an assistant might manage for you, the user.

2.3 The Camera Model

Individual windows within an artificial reality interfaces can be represented as a virtual camera. In a three-dimensional (3D) artificial world the one needs to be able to control the location or aspect in the world one is looking at. To perform this using today's two-dimensional (2D) screens, this work centers around the features of camera based controls. Most 3D graphics packages support some form of camera controls, such as HOOPS and PHIGS PLUS. The major advantage being that the user can also manipulate the objects **inside the** views of the virtual cameras.

In a virtual factory world where multiple objects will be changing and moving without regard for the user's view or attention, the interface paradigm becomes that of a camera-person or director. Monitoring of factory processes is achieved using these virtual cameras. The user chooses where the cameras will be pointed and how many will be present.

At the top Levels, a single window on the screen is a window into an artificial three dimensional world. Inside this world you can create objects that represent information and data flow. Further, you can create additional camera views which let you watch what is happening in different places within the artificial world.

3 THE VIRTUAL FACTORY ENTERPRISE

3.1 Historical perspective on Factory Controls

In the early days of the industrial revolution it was possible, if not required, for machinists to work in close proximity of the machines they were to control. Machines produced their own information in the form of sounds, odors, and "the feel" of the working unit.

Later, the central control room came into being. Here one could find whole rooms full of read-outs, charts, dials, and warning bells. For some people, it was difficult to be away from their machines, even these few yards. No longer were there noises and odors to be had. Often it took a new generation of employees to learn to use the control room gadgets in a productive manner.

As the number of automated machines increased, fewer controls could be kept in a single control room. In today's factories controls are distributed in a clustering manner. These machine clusters then report in a "control room" like manner to centralized monitors and strip charts which allow for recording and monitoring. Programmable controllers handle most of this reporting function. IBM PCs and clones are being used as front ends to these distributed control sites. Graphs and visual programming languages (ladder logic and flow diagrams) are being used to control these machines. Control information can be downloaded from the remote control rooms as well as at the local machine.
In this work we propose that 3D graphics can be used to recreate gadgets such as toggle buttons, numeric readouts, slider controls, and other controls. one can now take the control room to the person, instead of the person going to the control room. In fact, many people can manipulate and view the same control panel at the same time.

In addition to creating the control panels for the factory, an artificial reality environment can also reproduce the physical machines and objects. one such example is the ARKola Simulated Bottling Plant developed by Gaver, et. al [GS091]. In this artificial world multiple people manage different parts of the factory and interact with one another.

3.2 The Physical Factory

The original concept of the Lincoln Log Factory of the Future was that it be highly autonomous, ideally, needing minimal help from the user. The goal was to use multiple expert systems in a cooperative communication environment to develop an intelligent manufacturing environment. The system controls multiple robots, parts feeders, vision requirements, and a material handling subsystem [figure 2]. In figure 2, each of the rounded boxes represents an actor within the factory floor. The robot expert systems each control a single robot and the cell experts control a single workcell. The dashed lines, with arrows, are meant to show the one to one relationship between the VCLIPS experts and the physical objects.

The current model of the factory of the future utilizes an integrated Computer Aided Design (CAD) package which has knowledge of structural requirements and part constraints. It requires the user to select parts which can actually be placed. The intelligent CAD system creates a work order, represented by structured English sentences. It sends these instructions to the factory scheduling software. The scheduler is then responsible for creating and monitoring workcell and robot processes which actually built the product.

Past years of research has suggested that future factories will not be completely autonomous but will require man and machines to work together in a cooperative manner to produce quality goods.

3.3 The Virtual Factory

The artificial reality version of the factory consists of artificial entities which share a portion of their knowledge base. The knowledge base is used to render a virtual world. Rendering is done by one or more of the entities using a 3D object oriented graphics system called HOOPS. HOOPS is both a rendering and input system developed by Ithaca Software, Inc.. HOOPS allows for both presentation and mouse based input. HOOPS is a trademark of Ithaca Software, Inc.. We use the mouse mainly for picking and menu options. However, you could create any imaginable widget under mouse control.

The artificial world will contain 3D objects .These objects will be placed in the artificial world in a similar arrangement for each person in the environment. This allows the spatial relationships to be shared with others. However, the views of the world are controlled by the individual tailoring the camera objects.

4 EVENT BASED SIMULATION AND CONTROL ENVIRONMENT

The CLIPS expert system shell provides for production system based inferencing rules. The CLIPS rules react to each of the users requests as they are presented through the artificial reality graphics engine. These messages are sent via the VEOS message layer to the Interface expert.
The rules are designed with both preconditions that must be true and postconditions that will be propagated within the knowledge base. The work is similar to that reported by Daniel Gieskens and James Foley in their SIGCHI 92 paper titled "Controlling User Interface objects through pre- and postconditions.

The University of Massachusetts Lowell has developed a coarse-grain parallelism version of CLIPS, called Parallel CLIPS (PCLIPS). We used portions of the existing PCLIPS architecture and the VEOS messaging layer to create a new test bed which we call Virtual CLIPS (VCLIPS). To accomplish this merger of both PCLIPS and VEOS, we removed the XLisp layer from the distributed VEOS code. Simply using the "talk" layer of the VEOS environment we have been able to send VCLIPS have similar features: being entity based and having multi-platform capabilities. We chose VEOS because it has the potential to become a de facto standard with the research community

4.1 Rapid Prototyping Control Panels

Artificial reality based control objects such as toggles, sliders, and read-outs can be placed on a virtual control panel. These panels can then be combined in a module-like manner to create larger control stations [figure 3]. To create new iterations of the control panels the developer can arrange the objects differently on the control station. Figure 3 shows three different widgets are placed on the viewstation interface. Each of the three smaller monitors can be patched into the larger monitor. The *arrow* widget provides coarse camera control for the selected monitor.. The graphical objects (widgets) use a message passing scheme to inform the interface expert which event took place [figure 41-

To get around in the factory (A.R.) we are exploring different models. At present we are using the monitoring camera paradigm. The viewstation that we generate on the screen contains one main simulated monitor and three smaller monitors. The camera that is "patched-in" to the main monitor location can be manipulated by the controls on the viewstation including: Pan, Orbit and Dolly camera options. The three cameras can be looking anywhere in the artificial world, they do not need to be different views of the same work area.

The multiple cameras create a feeling of presence for the user. Similar work has been reported by [TYT+92] in their work with object oriented video where they have real cameras at real sites with graphic overlays. In our work, we have simulated cameras looking at virtual worlds where the objects seen through the cameras are controllable.

4.2 Shared Virtual Spaces

In the virtual factory of the future there will be teams of professionals. Each participant will share, from separate locations, the controls of the factory floor. Factories in one part of the world can be monitored and controlled from another part of the world. It will even be possible to meet at the same (synchronous) time, and jointly solve an engineering or manufacturing problem. The virtual factory of the future will still contain workers. There will be local technical repair teams who will be coordinating with others via Artificial Reality, Video Conference, or some other high bandwidth communication link.

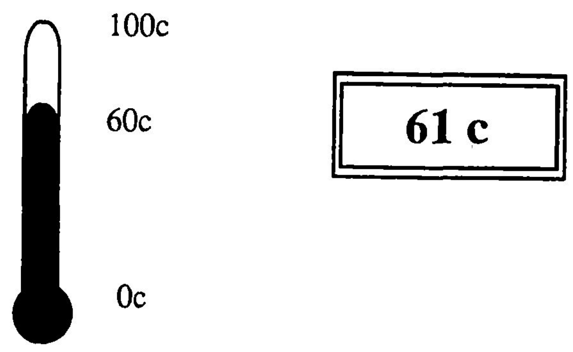


Figure 1. Digital vs. pictorial temperature widgets

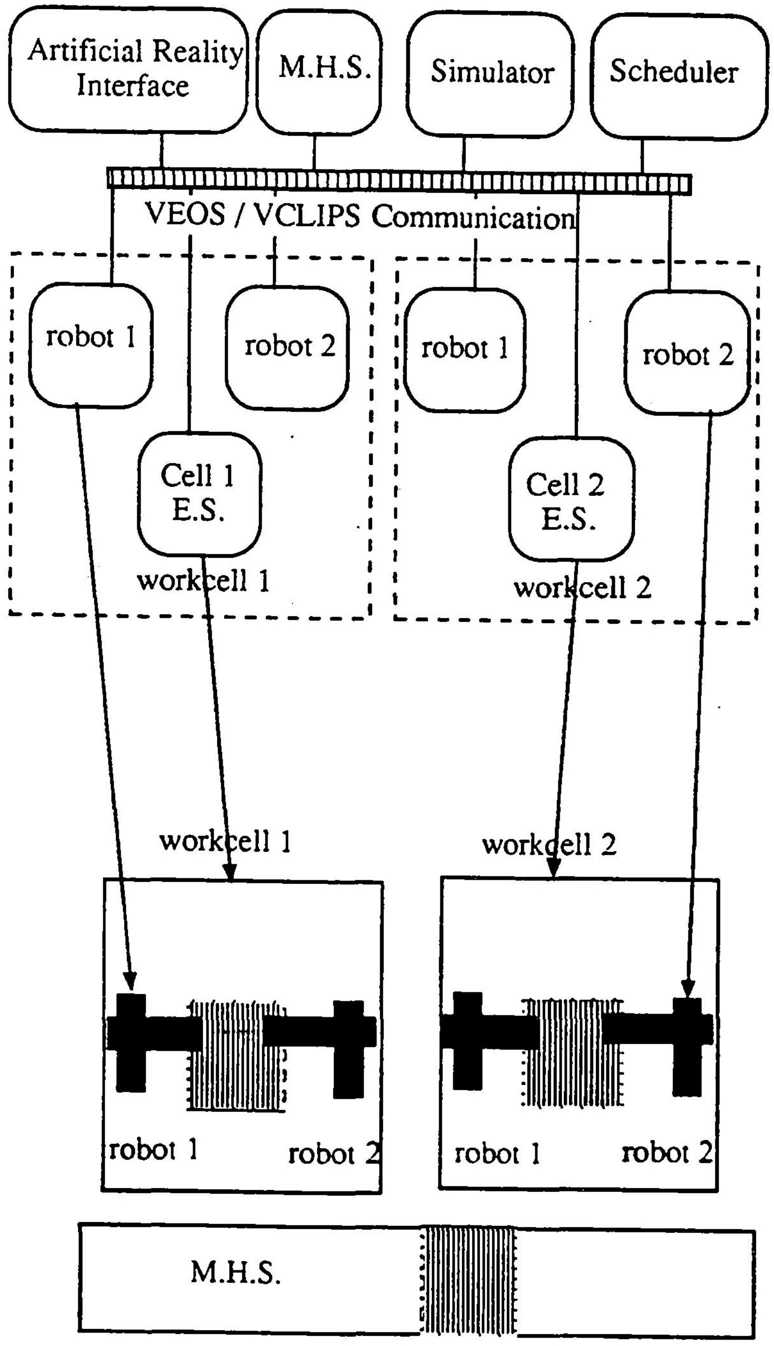


Figure 2. Factory controlled by VCLIPS expert systems.

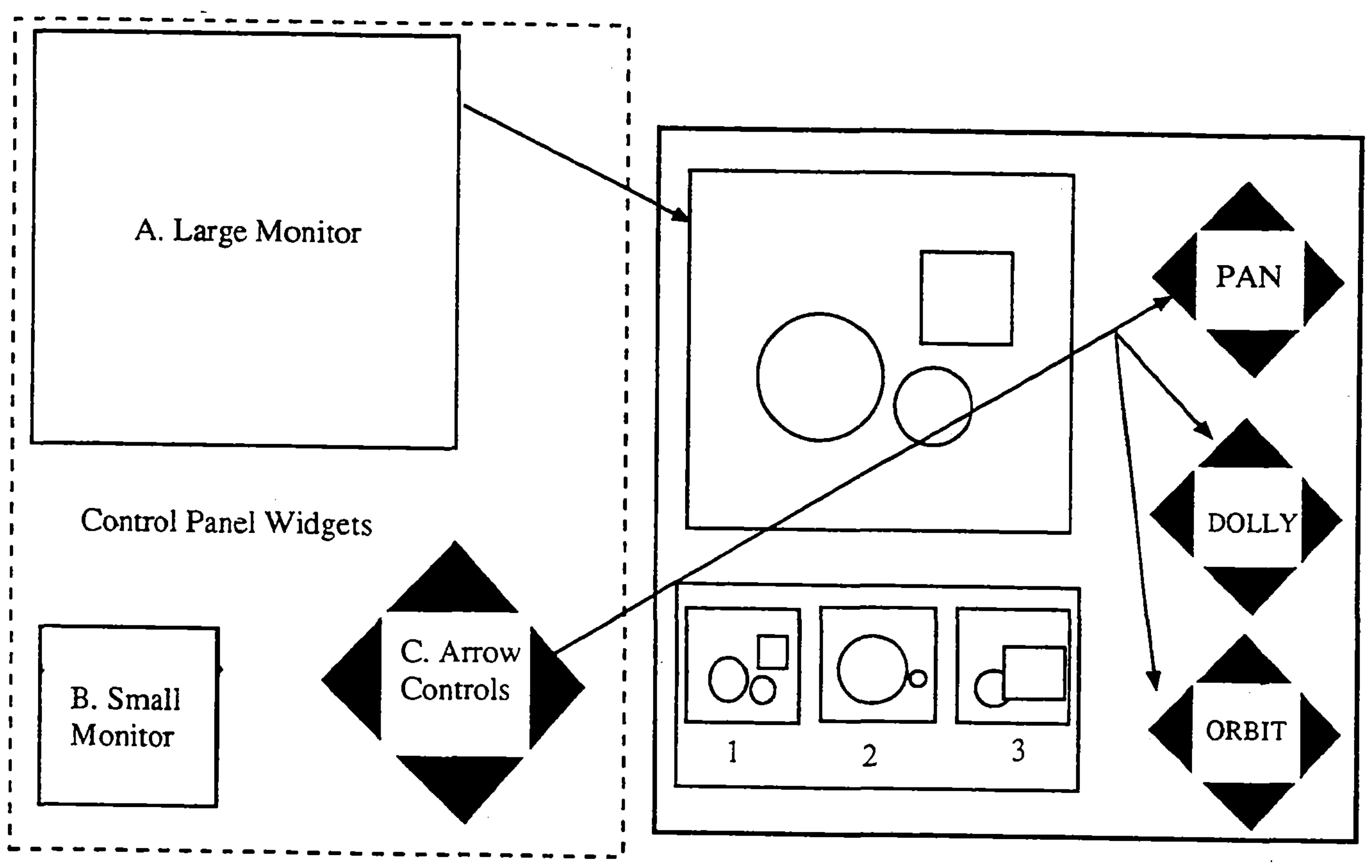


Figure 3. Viewstation Example

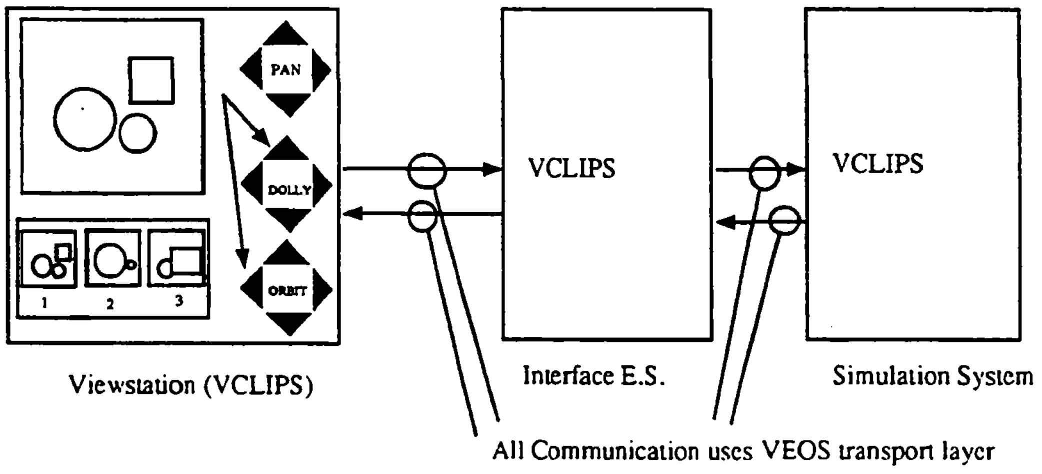


Figure 4. Artificial Reality based on VCLIPS model.

5 SUMMARY

Further research will be done in the areas of artificial reality-based user interfaces, virtual vehicles which can be used to move around in the artificial worlds and real-time control of physical objects from within the artificial reality. Additionally, we are seeking industrial and research partners who are interested in experimenting with artificial reality based monitoring of an actual factory floor. The next version of the control panels will provide camera creation and minimal task-based control. By task-based, we mean that commands such as "follow object 123" will keep the camera targeted at object-123 wherever it may move. We will also deal will more long distance instruction and multi-user interaction.

References

[Bro91] Rodney A. Brooks. A *robot that walks. emergent behaviors from a carefully evolved network,* pages 99-108. Morgan-Kaufmann, 1991. Editor(s): Badler, Norman I.; Barsky, Brian A.; Zeltzer, David.

[CJK+92] Christopher Codella, Reza Jalili, Lawrence Koved, J. Bryan Lewis, Daniel T. Ling, James S. Lipscomb, David Rabenhorst, Chu P. Wang, and Greg Turk. Interactive simulation in a multi-person virtual world. In *Proceedings ACM SIGCHI '92,* pages 329-334, 1992. (IBM Watson R. C., Veridical User Envs.).

[CRM91] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. The information visualizer, an information workspace. In *Proceedings ACM SIGCHI '91,* pages 181-188, 1991. (Card.PARC@Xerox.com).

[DB92] Paul Dourish and Sara Bly. Portholes: Supporting awareness in a distributed work group. In *Proceedings ACM SIGCHI '92,* pages 541-548, 1992.

[ES90] Margaret A. Ellis and Bjarne Stroustrup. *The Annotated C++ Reference Manual.* Addison-Wesley, Reading, MA, 1990.

[FB89] Jay Fenton and Kent Beck. "playground": An object-oriented simulation system with agents rules for children of all ages. In *Proceedings OOPSLA '89,* pages 123-136, October 1989. (Apple Vivarium Project) .

[GA90] Michael Girard and Susan Amkraut. "eurhythmy": Concept and process. *Journal of Visualization and Computer Animation, Vol. 1,* No. 1, August 1990.

[GF92] Daniel F. Gieskins and James D. Foley. Controlling user interface objects through pre- and postconditions. In *Proceedings* A *EM SIGCHI '92,* pages 189-194, 1992. (foley@cc.gatech.edu).

[Gre91] Mark Green. Using dynamics in computer animation: control and solution issues. In *Making them move. mechanics, control, and animation of articulated figures.,* pages 281-314. Morgan-Kaufmann, 1991. Editor(s): Badler, Norman I.; Barsky, Brian A.; Zeltzer, David.

[GSO91] William W. Gaver, Randall B. Smith, and Tim O'Shea. Effective sounds in complex systems: The arkola simulation. In *Proceedings of HCI '91,* pages 85-90, ,1991.

[Hou92] Stephanie Houde. Iterative design of an interface for easy 3d direct manipulation. In *Proceedings ACM SIGEHI '92,* pages 135-142, 1992.

[KL90] Dennis Kafura and Keung Hae Lee. "act++": Building a concurrent c++ with actors. *Journal of Object Oriented Programming,* pages 25-37, May/June 1990.

[KWN90] Dennis Kafura, Doug Washabaugh, and Jeff Nelson. Garbage collection of actors. In *ECOOP/OOPSLA '90 Proceedings,* pages 126-134. ACM, August 1990.

[LKL91] J. Bryan Lewis, Lawrence Loved, and Daniel T. Ling. Dialogue structures for virtual worlds. In *Proceedings ACM SIGCHI '9l,* pages 131-136, 1991. (IBM Watson R.C., Veridical User Envs.).

[Luc87] S.E. Lucco. Parallel programming in a virtual object space. In *OOPSLA '87: Conference on Object Orientated Programming, Systems, Languages and Applications*, pages 26-34. AEM, Oct 1987.

[McF91] Tim McFadden. *Cyberspace. first steps*, chapter Notes on the structure of cyberspace and the ballistic actors model, pages 334-362. MIT Press, 1991. Editor: Benedikt, Michael.

[Mes90] Jose Meseguer. A logical theory of concurrent objects. In *Proceedings ECOOP/OOPSLA '90*, pages 101-115, October 1990. (SRI International) .

[MF91] Chip Morningstar and Randall Farmer. *Cyberspace. first steps*, chapter Habitat, pages ?-?? MIT Press, 1991. Editor: Benedikt, Michael.

[MR89] Naftaly H. Minsky and David Rozenshtein. "controllable delegation": An exercise in law-governed systems. In *Proceedings OOPSLA '89*, pages 371-380, October 1989. (minsky@aramis.rutgers.edu).

[Rub90] Kenneth S. Rubin. Reuse in software engineering: An object oriented perspective. *IEEE Publication Num. CH2343-1/90*, pages 340-346, 1990.

[SLGS92] Chris Shaw, Jiandong Liang, Mark Green, and Yunqi Sun. The decoupled simulation model for virtual reality systems. In *Proceedings AEM SIGEHI '92*, pages 321-328, 1992. (Univ. of Alberta).

[TYT+92] Masayuki Tani, Kimiya Yamaashi, Koiehiro Tanikoshi, Masayasu Futakawa, and Shinya Tanifuji. Object-oriented video: Interaction with real-world objects through live video. In *Proceedings AEM SIGEHI '92*, pages 593-598, 1992.

[WH91] Jakub Wejchert and David Haumann. Animation aerodynamics. *Computer Graphics, Vol.* 25, No. 4:19-22, July 1991. ().

[ZCW+91] Robert E. Zeleznik, D. Brookshire Conner, Matthias M. Wloka, Daniel G. Aliaga, Nathan T. Huang, Philip M. Hubbard, Brian Knep, Henry Kaufman, John F. Hughes, and Andries van Dam. An object-oriented framework for the integration of interactive animation techniques. *Computer Graphics, Vol.* 25, No. 4:105-111, July 1991 .

Application of Knowledge Space Theory for Student Modeling

Rose Chu and Michael Villano
Honeywell Sensor & System Development
3360 Technology Drive
Minneapolis, MN 55418
612-951-7438
chu@ssdc.honeywell.com

In the field of intelligent tutoring systems (ITS), student modeling remains a difficult research problem. Probabilistic models such as Knowledge Space Theory and Bayesian Belief Network have been suggested as good candidates for representing the element of uncertainty due to a student's lucky guess and careless errors, for instance. As a first attempt to test the utility of probabilistic student modeling, we explore the application Knowledge Space Theory for student modeling in an existing ITS called GT-VITA (Georgia Tech-Visual Inspectable Tutor and Assistant). While subsequent versions of GT-VITA have been successfully fielded at NASA Goddard Space Center, the original GT-VITA prototype remains an excellent testbed to study various avenues in ITS such as student modeling and curriculum transitioning.

We will report on our "exercise" in applying Knowledge Space Theory and on the lessons learned in the process. In theory, KST provides many opportunities in item selection, curriculum advancement and student feedback. In practice, the task of building a knowledge space is nontrivial and the task of implementing it raises many more issues that are still unanswered.

N96-14986

Impasse-Driven Tutoring for Reactive Skill Acquisition

Randall W. Hill, Jr.
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, M/S 525-3631
Pasadena, CA 91109, USA
hill@gobi.jpl.nasa.gov

W. Lewis Johnson
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, USA
johnson@isi.edu

1. Introduction

We are interested in developing effective performance-oriented training for the operation of systems that are used for monitor and control purposes. We have focused on one such system, the communications Link Monitor and Control (LMC) system used in NASA's Deep Space Network (DSN), which is a worldwide system for navigating, tracking and communicating with unmanned interplanetary spacecraft. The tasks in this domain are procedural in nature and require reactive, goal-oriented skills; we have previously described a cognitive model for problem solving that accounts for both novice and expert levels of behavior as well as how skill is acquired [Hill and Johnson, 1993]. Our cognitive modeling work in this task domain led us to make a number of predictions about tutoring that have influenced the design of the system described in this paper.

Tutoring in our training system is *impasse-driven.* Unlike other tutoring techniques such as *model tracing* and *plan recognition,* which decide to intervene on the basis of understanding every student action, our approach depends instead on *impasse recognition* to drive the tutoring intervention. When the tutor recognizes that the student has reached a procedural impasse, it intervenes with advice that it generates with its expert cognitive model, which is used to understand and resolve the current impasse in situ.

The tutor uses a method called *situated plan attribution* to recognize and explicate students' procedural impasses in the LMC task domain. We use the term *plan attribution* instead of *plan recognition* because it does not assume that the problem solver's actions are controlled by plans. Rather, we take the situated action view of plans: plans are resources that guide and orient action, and actions are ultimately contingent on the state of the world [Suchman, 1987].

The tutor recognizes potential impasse situations by using a number of different resources. Plans are attributed to the student based on a description of the task. The attributed plans, in turn, generate expectations about the student's actions and goals. Each action is evaluated with respect to the student's attributed plans, its actual effects on the training device and the goals associated with these plans. Impasses related to specific actions are often indicated by feedback from the device simulator, e.g., a command is rejected or a warning is given. This type of impasse is easy to detect, both for the student and the tutor, though it is not always easy to resolve. Other impasses, are not as obvious, since the student's action may result from a misconception about a plan or goal rather than violating a constraint of the system, and the error may not manifest itself for some time. For instance, the student may complete a procedure without achieving it's goals or take an action that is not relevant to the current task. The tutor recognizes this type of error as a *potential* impasse since the device itself may not show any sign of malfunctioning and the student may be unaware of a problem until a much later stage in the task. When the tutor detects a potential impasse it intervenes, thereby forcing the impasse in order to resolve the misconception near the point at which it showed itself.

We have implemented our tutor in Soar [Laird et al., 1987], an integrated problem solving and learning architecture. Our tutor uses the Soar chunking mechanism to learn and is thus often able to recognize an impasse and its resolution in the task context without searching through a set of problem spaces to re-derive a solution.

2. Task Domain: Monitor and Control Systems

Monitor and control systems are ubiquitous in our increasingly automated society: power plants, factories, environmental control systems and operations centers all use computers to control other machines and to monitor their status and health. Typically only a portion of the work to be done in the problem domain is automated by the monitor and control system, leaving tasks requiring judgment and specialized knowledge to a human Operator.[1] In this paper we describe an approach to training Operators that has been developed for tasks in one such domain, namely, the communications Link Monitor and Control (LMC) system used in NASA's Deep Space Network.

The DSN is a worldwide system for navigating, tracking and communicating with all of NASA's unmanned interplanetary spacecraft. As the name suggests, the LMC system is used by operations personnel to monitor and control a DSN communications link during an interaction with a spacecraft. A communications link is formed by assigning a collection of equipment to an LMC system for a particular mission; the link typically includes a large antenna dish (34 or 70 meters in diameter), its electromechanical controllers and subsystems, a receiver/exciter, a digital spectral processor (DSP), a precision power monitor (PPM), a hydrogen maser, frequency and timing subsystem (FTS), phase calibration generators, digital tone extractor (DTE), and numerous other devices.

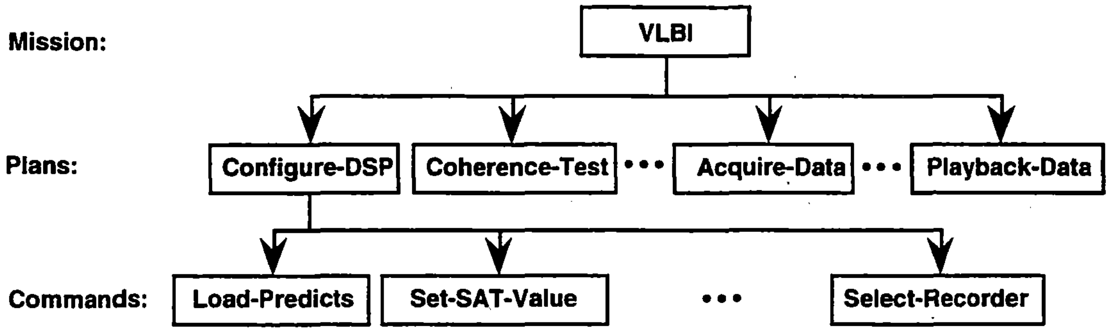


Figure 1 The VLBI task with a representative set of plans and operators

The tasks performed by LMC Operators involve configuring and calibrating the communications link, testing the configuration for coherency, acquiring data from the spacecraft and playing back the data to the network operations control center. Each mission consists of a number of other similar tasks, and the way to perform each task is described by a procedure[2] manual, which provides the Operator with command sequences for each of the various subsystems. Figure 1 shows a portion of the task hierarchy for a VLBI[3] mission. Note that the figure shows three basic levels in the hierarchy: *mission, plan,* and *command.* What Figure 1 does not convey is that the order in which the plans and commands are executed is not completely specified; though some partial orderings exist among plans and commands, there is a lot of room for variability from one mission to the next (or from the way one Operator performs the tasks to the next.)

To accomplish a VLBI mission, the Operator performs the plans that are shown in Figure 1: Configure-DSP, Coherence-Test, and so on. The Configure-DSP plan has operators[4] to load the mission-specific prediction data file (Load-Predicts), set the attenuation values on the Intermediate Frequency Video Down Converter (Set-SAT-Values), ..., and select a recording device to capture the mission data (Select-Recorder). Applying an operator involves issuing a command (e.g. the Load-Predicts directive is NLOAD *predicts-file*).

[1] The term *Operator* will be used two different ways in this paper. To disambiguate its usage, we capitalize *Operator* when referring to a human being, and we use lower case to refer to a Soar *operator.*

[2] Henceforth we refer to these written procedures as plans.

[3] VLBI stands for Very Long Baseline Interferometry.

[4] An operator with a small "o" denotes a function that is selected and applied to a state to get a result. This is in keeping with the Soar notion of an operator [Laird et al., 1987].

It would appear that the tasks in the LMC domain are straightforward and would require little or no training -- just follow the plans given in the mission procedure manuals. We have found, however, that this is not the approach taken by domain experts. Through extensive interviews with expert operators and system engineers, we determined that the procedure manuals only provide a subset of the knowledge needed to successfully perform the tasks associated with a mission. What is generally lacking in the procedure manuals is a complete description of the required device state conditions before and after a command is issued. Expert Operators possess a knowledge of the preconditions and postconditions for each command and verify these conditions are satisfied before and after issuing the commands. Operators who lack this knowledge may find it difficult to complete even simple plans, since the commands may be rejected, or worse, they may put the device into an incorrect state for the current plan or mission. For example, one of the preconditions for the Load-Predicts operator is that the predicts-file being loaded must be present on the system. If the Load-Predicts command is issued for the predicts-file named "JK" (i.e., the Operator issues NLOAD JK), it will be rejected if a file by that name is not present in the predicts file directory.

To complicate matters, during a two hour mission an Operator may interact with five major subsystems, comprised of fifty different devices, for which more than 250 unique attributes must be monitored and 500 event notice messages processed. The sheer quantity of monitor data accentuates the difficulty of executing the control procedures. Unexpected device state changes, device failures, and the slow reaction time of certain devices can cause procedural impasse and Operator confusion. If a command is issued when one of its preconditions is not satisfied, then it is likely to be rejected, or worse, it puts the device into an undesirable state. When Operators observe that a precondition is not satisfied, they have to know how to react. Consequently, procedurally-defined command sequences are not sufficient to accomplish most task goals. The plan acts as a guideline, but the Operator must bring other knowledge to bear on the performance of a task.

3. Domain Problem Solving: Implications for Plan Recognition

In this section we begin to motivate our approach to tutoring by first describing the nature of problem solving and skill acquisition in the LMC domain. In order to better understand the nature of these skills we implemented a detailed cognitive model that accounts for the behavior of both novice and expert LMC Operators. Developed in Soar, a problem solving architecture with a learning capability [Laird et al., 1987], the model enabled us to make a number of predictions about how Operators acquire knowledge and skill in this domain; it improves its behavior over time, acquires new knowledge and is able to recover from incorrect knowledge [Hill and Johnson, 1993]. We describe how these predictions about skill acquisition affected the design of our intelligent tutor. Ultimately, we desired to capitalize on the insights gained from the cognitive model, so we start by revisiting some of the key issues raised by the cognitive model that motivated our approach to tutoring using *situated plan attribution.*

3.1 Plan Execution and Situated Action

Problem solving in the LMC domain involves both plan execution and situated action. By plan execution we mean that Operators issue commands according to a written procedure. Actions are situated, on the other hand, in that the context in which commands are issued must be taken into account, resulting in behavior that does not always resemble the original plan. A plan may specify a command to be executed, but the situation may warrant taking a different action in order to achieve the underlying goals of the plan or task. To illustrate the effect of the device situation on the Operator's actions, consider the case where there are two plans that need to be executed as a part of the VLBI task: first the Configure-DSP plan and then the Coherence-Test plan. The tables in Figure 2 show the default command sequences for these two plans, where the operators are shown in the left column and the corresponding commands are shown in the right column. The Utility Commands table contains commands that are not typically used in any particular plan, but are useful for changing a device's state.

Configure-DSP	
Operator	Command
Load-Predicts	V NLOAD <set-id>
Set-S-Attenuation	V SAT <sat>
Set-X-Attenuation	V XAT <xat>
System-Temperature	V NTOP <s><x>
Select-Recorder	V NRMED <nrmed>
Set-Offset	V OFST <ofst>

Legend:
V = Subsystem ID, <> = parameter variable

Coherence-Test	
Operator	Command
Phase-Calibration-Gen	V NPCG <v1>
Run-NCB-Program	V NRUN <v2>
Run-FFT-Program	V NFFT <v3>
Digital-Tone-Extractor	V NDTE <v4>

Utility Commands	
Operator	Command
Idle-NCB-Program	V NIDLE REC

Figure 2 Plan Descriptions for Configure-DSP and Coherence-Test. A utility command is also shown that is not associated with a particular plan.

```
> V NLOAD JK
> COMPLETED. LOADING NCB PREDICTS ...
> V SAT 25
> COMPLETED.  S-BAND  ATTENUTATION=25
DB
> V XAT 20
> COMPLETED. X-BAND ATTENUATION=20DB
> V NTOP 17.3 19.4
> COMPLETED. SYSTEM TEMP: S 17.3 X 19.4
> V NRMED LD0
> COMPLETED. NRMED: LD0
> V NPCG MAN
> COMPLETED. NPCG MODE: MANUAL
> V NRUN COLD
> COMPLETED.  NCB MODE: NRUN
> V NFFT E

> ...
```

Case 1: Two errors: (1) wrong SAT value (should be 30 instead of 25) and (2) missing the OFST command.

```
> V NLOAD MK
> REJECTED. NOT ALLOWED IN NRUN MODE
> V NIDLE REC
> COMPLETED.  NIDLE REC INITIATED

> V NLOAD MK
> REJECTED. INVALID SET NAME
> ...
```

Case 2: Multiple constraint errors.

```
> V NIDLE REC
> COMPLETED.  NIDLE REC INITIATED
> V NLOAD JK
> COMPLETED. LOADING NCB PREDICTS
> V SAT 30
> COMPLETED.  S-BAND  ATTENUTATION=30
DB
> ...
```

Case 3: Avoiding the Case 2 errors.

Figure 3 Operator Logs: Examples of Errors and Situated Action

Figure 3 provides three different traces of an Operator executing these plans. The traces show the commands in the sequence they were issued by the Operator, where each command is followed by a message from the device telling whether the command was accepted or rejected.[5] Case 1 contains two typical novice errors that could easily go undetected. The first error was that the Operator mis-parameterized the SAT command with a value of 25 instead of 30. Normally the Operator would obtain the SAT value from a calibration table that contains this information for each antenna system, so the error may have been due to mis-reading the table. The second error was the omission of the OFST command, which is the last step of the Configure-DSP plan; this mistake will not manifest itself to the Operator, rather, it will affect the data correlation later, after the mission is complete.

Case 2 illustrates two examples of how the device simulator responds to constraint violations associated with the NLOAD command. We see that after NLOAD was rejected the first time the Operator recognized the nature of the

[5]The COMPLETED response means that the command was accepted by the device, but it does not imply that the intended action was successful. The REJECTED response means that the device will not attempt to execute the command because it violates a system constraint.

problem and corrected it with the NIDLE REC command. But when NLOAD was re-issued, it was again rejected due to the fact that the predict set specified by the NLOAD parameter, MK, did not exist.

Case 3 shows how a skilled Operator responds to the same situation as Case 2. Instead of rotely following the Configure-DSP plan, the Operator recognized that the situation called for issuing the NIDLE REC command first, and then the NLOAD JK command.

3.2 Cognitive Model

These three cases show the typical situations faced by LMC Operators, where it is not possible or feasible to just execute the plans described by the procedure manuals. Our Soar-based cognitive model was developed to learn how to handle situations of this type, where it executes the known plans until it recognizes that the situation requires a different action than the prescribed one. In cases where its actions failed, it learned from the failure and acquired new knowledge to help it avoid the same error in the future. The details of how the cognitive model performed are described in [Hill and Johnson, 1993], and in the following paragraphs we summarize the conclusions from that effort that had an influence on the design of the tutor described in this paper.

(1) *Learn by doing*. Though this is not a new insight in the field of education, it is one that is strongly supported by the cognitive model and is a direct result of the way that the Soar chunking mechanism works [Newell, 1990]. This result agrees with Anderson's account of proceduralization and skill acquisition [Anderson, 1983]. It is not sufficient to acquire declarative knowledge about a task, rather, knowledge must be brought directly to bear on solving problems. Our cognitive model's task performance improves only as it compiles its knowledge about how to perform the domain task. As a consequence of this result, we have designed our training system to maximize the Operator's experience in performing the target task skills; the Operator performs tasks on a Link Monitor and Control system simulator and the tutor strategically intervenes in a manner that will be described later.

NCB-Program Device	
Attribute	Value
MODE	RUN

VLBI Predicts Set Device	
Attribute	Value
NAME	JK
RECEIVED?	YES
QUALITY?	OK

Figure 4 Partial Set of Devices and their Attributes

Operator	Command	Precondition Name	Device	Attribute	Value
Load-Predicts	V NLOAD <id>	Load-Predicts-PC1	NCB-PROGRAM	MODE	IDLE
		Load-Predicts-PC2	VLBI-PREDICTS	RECEIVED?	YES
		Load-Predicts-PC3	VLBI-PREDICTS	QUALITY	OK

Figure 5 Preconditions for Load-Predicts Operator

(2) *Expert problem solvers do not rotely execute plans*. Rather, they evaluate the appropriateness of executing each command in the plan with respect to the device situation, and they issue a command only when all of its preconditions are met. Moreover, expert problem solvers recognize when it is necessary to reactively plan in order to overcome unsatisfied command preconditions. To illustrate these points, consider the device model in Figure 4, which reflects the state of the NCB-program and VLBI Predicts Set devices at the time that the Operator performed the actions shown in Figure 3. The state of a device is represented with a set of attribute-value pairs. In this case, the NCB-Program device's MODE attribute has a RUN value, and the VLBI Predicts Set is named JK, has been received and its quality is acceptable.

The Load-Predicts operator in Figure 5 has three preconditions, of which only two are satisfied with respect to the current device situation. The Load-Predicts-PC1 precondition is not satisfied since it requires the NCB-Program to be in the IDLE mode rather than the RUN mode. According to our cognitive model, an expert Operator will notice that this precondition is unsatisfied and will act to correct the situation, as we observe in Case 3, by issuing the NIDLE REC command to put the NCB-Program into the IDLE mode.

(3) *Novices acquire skill by learning operator preconditions.* In addition to describing expert behavior, our cognitive model also accounts for novice behavior and the process by which skill is acquired. It predicts that a novice Operator who does not know about a precondition that is unsatisfied will make the errors shown in Case 2, issuing the command in the wrong situation, resulting in a rejection. In our cognitive model, the novice acquires skill by recognizing there was an error (i.e., the rejection notice), understanding the meaning of the rejection message, searching for and applying an operator to repair the situation, and adding the newly acquired precondition to its knowledge; these steps correspond to the goals that a student would have in trying to recover from an error or an impasse, and they are also opportunities for a tutor to give assistance. In addition to making errors due to missing preconditions, our model also addresses the situation where the Operator has a misconception about a precondition and needs to learn the correct knowledge.

(4) *Learning occurs in impasse situations.* The cognitive model performs the task until it reaches a point where it needs additional knowledge to cope with the situation. It acquires knowledge at the impasse point from an artificial tutor, which provides information about the missing or incorrect operator precondition. These impasse points were a natural place to tutor the novice with the needed situational knowledge since the Soar chunking mechanism could immediately transform the declarative tutoring knowledge into procedural knowledge. In addition to the instruction about preconditions, the tutor also gave the novice corrective steps for resolving the impasse.

(5) *Learning occurs in a goal context.* This observation is related to the previous one and it influences the decision to tutor at the impasse point rather than before or after the training session. The Soar chunking mechanism works by summarizing the results of a subgoal in new operators that can be applied under similar circumstances to obtain the same results without needing to search the subgoal problem space again. All learning in the Soar architecture occurs in a goal context and the contents of what is learned also depends on the goal context. These observations led us to predict that the most effective way of tutoring human students is to intervene in the right goal context, that is, when the student has the goal to resolve an impasse. The challenge for the tutor is to recognize when the student is at an impasse point and then provide the appropriate instruction for resolving the impasse.

(6) *Some knowledge gaps are hidden.* If the student only learns about preconditions by violating them and reaching an impasse, then some preconditions may not be learned. The action sequence order may eliminate the need to verify that a precondition is satisfied since the results of one action satisfy the preconditions of the next; as long as the actions are always executed in the same order the precondition will be satisfied and the precondition will remain hidden in the sense that problem solver will not act to verify that it is satisfied. It is sometimes difficult to create device situations that will force the student into a failure impasse for certain preconditions, nevertheless the tutor needs to detect hidden knowledge gaps, perhaps by forcing the student out of a rote action sequence and into situations where the actions are executed in a different order.

3.3 The Tutoring Intervention Problem

We draw two conclusions about tutoring in the LMC domain from the preceding results. First, the model indicates that impasse situations are strategic opportunities for learning. In [Hill and Johnson, 1993] we describe how our cognitive model acquires new knowledge while searching for operators to resolve the impasse. The key to learning in the model was to provide the pertinent information to resolve the impasse when the student was in a problem space that could make use of it. The implication of these observations is that tutors must be capable of recognizing when the student has committed an error or potential error in order to intervene effectively. This conclusion is supported by [Galdes, 1990], which provides evidence that human tutors intervene during performance-oriented training only after they have identified definite or potential errors.

The second conclusion from our modeling work is that once the tutor chooses to intervene, it must situate its explanation with respect to the circumstances that created the impasse. The tutor must be able to apply its expert cognitive model at the impasse point, and it must do it in a way that will generate a situation-specific explanation of the impasse as well as a means of resolving it. We call these two issues the *tutoring intervention problem*, and the approach we describe in the next section attempts to address this problem as it pertains to reactive problem solving domains.

4. Situated Plan Attribution

Our approach to addressing the tutoring intervention problem is based on a method we call *situated plan attribution.* We describe in detail how this method is used by our tutor, which is integrated with an LMC system simulator. This section, which is organized into three parts, makes the following points: it (1) presents some assumptions about plans and actions and how they influenced our decision to use impasse recognition to drive the tutoring intervention; (2) describes how situated plan attribution creates expectations about behavior and provides a context for intervention; and (3) describes how the tutor recognizes impasses.

4.1 Assumptions about Plans and Action

Our approach to tutoring makes a key assumption: plans do not determine behavior [Suchman, 1987]. As we have shown in our cognitive modeling work, plans influence behavior, but they act primarily as a resource to help guide the performance of the task. Consequently, plans are useful for creating expectations about behavior, but they cannot always be used to understand and explain it. For instance, some situations call for reactive planning, resulting in actions that, when observed by a tutor, do not fit into the default plan. The tutor needs to be able to recognize whether the student's actions are appropriate in these situations, but it may be very difficult to do so if the action has to be recognized and explained in terms of a known plan. A strict plan recognition approach to understanding situated behavior would require a library of all plans for all situations.

For the link monitor and control domain, we have a complete set of default plans for performing a mission, but we believe it would be impractical to represent all of the variations of the plans that would be needed to account for situational differences that arise from the dynamic nature of the problem domain. For instance, in the situation that was described for Cases 1-3 in Figure 3, a variation of the Configure-DSP plan would have to include the NIDLE REC command to cover the situations where the Load-Predicts-PC1 precondition (Figure 5) is not satisfied. The same would hold true for every precondition of every operator in the plan: the situation could potentially force the Operator to deviate from the standard plan every time a precondition was not satisfied, meaning that there would potentially have to be a plan variant for each such situation.

This viewpoint led us to view plans as a way of providing only partial understanding of the student's behavior; more specifically, plans provide a framework for recognizing when the student has reached a potential impasse. Since impasses may result from a combination of student misconceptions and anomalous device states, we seek to understand the impasse using an attributed plan as a starting point, but not as the only means of understanding student behavior. Besides plans, the tutor also has access to several other resources for interpreting student behavior: it sees the state of the devices (i.e., situation in which the action is taken), it tracks the state of the goals associated with the attributed plans, and it sees the student's actions and the device's response to them. This approach shifts the computational burden away from giving a plan-based account of a student's behavior and toward using plans as one of several resources for the task of recognizing and explaining an impasse. Plans are a convenient way of organizing knowledge about goals and actions related to the task; they are a declarative description of how to perform the task as opposed to being an executable model, and they are meant to orient the tutor rather than giving a complete account of how to behave in the current situation, which is consistent with Suchman's view of plans and situated action.

An additional resource for detecting certain types of impasses is the simulator itself. The simulator rejects directives[6] whenever they violate a system-imposed constraint.[7] Directive rejections are cues used by the tutor in deciding whether to intervene: they eliminate the need to guess whether there is an impasse in these instances. Instead, the tutor only has to note that the directive was rejected and then determine the causes for the impasse

[6] Directives are synonymous with commands.

[7] The simulator is faithful to the actual link monitor and control system in the way that it accepts or rejects directives.

and resolve it. This eliminates the need to understand every student action, which is normally a requirement for both plan recognition and model tracing approaches to tutoring.

4.2 Situated Plan Attribution

The primary purpose of situated plan attribution is to generate expectations about Operator behavior. These expectations guide the impasse recognition process as well as the tutorial intervention. We begin by describing how plans are represented and used for generating expectations and recognizing impasses.

For each task there is a set of plans, each of which is based on a written procedure, that will accomplish the goals of the task. Examples of two plans are shown in Figure 2. A partial order among the plans for a mission is represented in a structure called a Temporal Dependency Network (TDN) [Fayyad and Cooper, 1992; Hill and Lee, 1992]. The goal of the TDN is to express an order among the plans that maximizes the concurrency of plan execution. A portion of the VLBI TDN is shown below in Figure 7. Note that the plans, Configure-Precision-Power-Monitor and Configure-Receiver-Exciter, can be executed concurrently, while Configure-DSP is executed only after these two have both been completed. If two plans can be executed concurrently, it means that their commands may be interleaved in the command sequence without harmful interaction. Plans that have dependencies are placed in succession to one another using the Before and After relations shown in Figure 7.

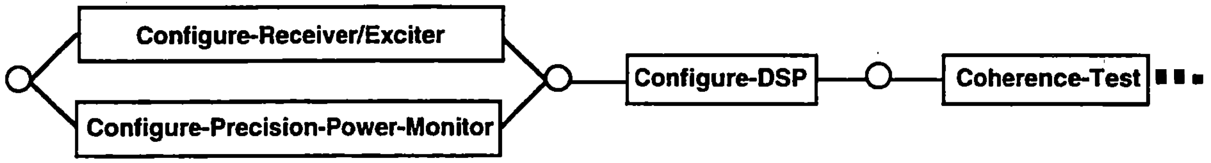


Figure 7 Partial Temporal Dependency Network for VLBI Task

The TDN resembles a procedure net: from a task perspective it describes a class of plans that would theoretically achieve the task goals. But unlike the procedure net approach to tutoring [Chen et al., 1991; Rickel, 1988; Warriner, 1990], we do not use the TDN as the sole basis for deciding when to intervene and provide tutoring. One difference is the granularity of the descriptions: the TDN specifies the relationship among plans, while the procedure net focuses on the relationships and constraints at the action level. We also loosen the procedure net assumption that actions are plan-based. Whereas the procedure net provides an action grammar for deciding on whether an action is appropriate or not, we use the TDN description to orient the impasse recognition process, which is described in the next section.

Given a mission, the tutor uses the mission's TDN to determine which plans are eligible for execution. These plans, known as *active plans* are the ones that we expect the Operator to be executing. As previously mentioned, multiple plans may be concurrently active, and we expect the Operator to interleave actions from different active plans while performing the various mission tasks. The tutor adds a plan to the active plan set when the plan's predecessors are satisfied and completed. Likewise, when a plan has been evaluated as satisfied and completed, the plan is removed from the active set and placed in the inactive set. Once a plan is placed in the active set, it creates expectations not only about what actions will be taken but also which goals will be active. The tutor marks successfully completed actions on the plan, and when all of the expected actions have been observed, the plan is marked "completed". A completed plan is not removed from the active set, however, unless its goals are also satisfied.

The plan's goals, like its actions, are continually monitored by the tutor, beginning at the time that the plan is placed in the active set. The tutor observes all device state changes and it recognizes when each of the plan's disjunctive goals are achieved. Once the conjunction of all the plan's goals are satisfied, the plan is considered to be satisfied. Figure 8 shows some of the Configure-DSP plan's goals.

Configure-DSP		
Device	Attribute	Expected Value
VLBI-PREDICTS	LOADED?	YES
CONFIGURATION-TABLE	SYNTHESIZER-FREQ-R1	300.21
CONGIGURATION-TABLE	SAT-VALUE	30
CONFIGURATION-TABLE	RECORDER	LD0
CONFIGURATION-TABLE	OFST-VALUE	2.7
...	...	...

Figure 8 Plan Goals for Configure-DSP

4.3 Impasse Recognition

The notion of an impasse is not new: it has been used in *repair theory* to help explain procedural errors in subtraction [Brown and VanLehn, 1980] and as a motivation for subgoaling during problem solving (e.g., the Soar architecture [Laird et al., 1987].) Our definition of an impasse is consistent with these uses of the term: impasses are obstacles to successfully performing a procedure, where the obstacle is a lack of knowledge or misconception about what to do next in a task situation.

The impasses recognized by our tutor fall into three categories: *action-constraint violations*, *goal failure*, and *plan dependency violations*. Impasses in the first category, *action-constraint violations*, are usually easy to recognize, both by the student and the tutor, because the device simulator gives an indication when a system constraint has been violated. Impasses in the other two categories are different from the *action-constraint violations* in that they are *potential* rather than *actual* impasses. They do not become actual impasses until the student notices that there is a problem, which may not be for a significant amount of time in these instances. Since we wish the tutor to intervene as near to the commission of an error as possible, the tutor forces an impasse when it detects a *goal failure* or *plan dependency violation*.

One of the significant aspects of our approach is that it does not attempt to recognize the mental state that led to the impasse. Instead, it depends on the device to detect many of the action constraint violations, while it uses it knowledge about plans, goals and the situation to detect the other impasses. In this section we will describe how the tutor recognizes impasses of each type.

4.3.1 Action Constraint Violations

The tutor recognizes *action constraint violations* by watching the communications link simulator. Examples of this type of impasse are shown in Figure 3: the NLOAD command was rejected in Case 2 because the NCB-program was in the wrong mode. In addition, Case 2 also shows the NLOAD directive subsequently being rejected because the parameter, MK, specified a non-existent predict set. The simulator is a faithful representation of the link monitor and control system: it rejects commands when system constraints are violated. These constraints constitute a subset of the operator preconditions.

Sometimes a rejection message provides a brief message of explanation with the rejection, but it never provides a means of working around the precondition failure. When the tutor observes that the command was rejected, it uses this fact to trigger its cognitive model, which determines the reason for the rejection and finds a way of resolving the impasse. The impasse explication process will be discussed in more detail in the next section, but it worth noting that the simulator recognizes the rejection conditions for the input commands and generates the appropriate rejection message. The tutor reads the same rejection message that the student receives, and it determines the reasons for the rejection without consulting the internal mechanisms of the simulator. Thus, recognizing this type of impasse does not depend on a detailed cognitive model of the student or of the student's plans. Rather, it depends on the simulator to reject inappropriate actions, removing some of the burden of impasse recognition from the tutor and placing it on the simulator. At the same time it does not place the burden of impasse diagnosis and recovery on the simulator, which only needs to recognize whether an action can be taken given the current state of the devices being simulated.

4.3.2 Goal Failure

The idea behind a *goal failure impasse* is that students may rotely follow a procedure without understanding the goals associated with it. A goal failure violation occurs when the student completes a plan (i.e., all of the expected actions have been observed for an active plan) but does not satisfy the plan's goals prior to beginning a successor plan. This type of error may not manifest itself in an obvious form, such as with a directive rejection, therefore, a student may not recognize the impasse immediately. Instead, a goal failure impasse would likely show up later as another type of impasse in a subsequent procedure. Allowed to pass without tutorial intervention, a goal failure impasse could be very difficult for the tutor to diagnose and resolve since the original context of the impasse may have been lost. Our intuition is that it is better to deal with these types of errors as they are recognized by the tutor rather than wait until it becomes an actual impasse at the action or task goal level (i.e., task failure).

Goal failure impasses are recognized using the expectations about actions, plans and goals, generated by situated plan attribution. The tutor is initially cued to this type of impasse when the student takes an action belonging to an inactive plan. If the inactive plan is a successor of an active plan that is complete but whose goals are not satisfied, then the tutor flags the action as a goal failure impasse, which triggers the tutor's cognitive model to diagnose and recover from the impasse in the current situation.

Case 1 in Figure 3 gives an example of a goal failure impasse. The Configure-DSP plan is assumed to be active during this trace. The student issued the SAT 25, which noted earlier should have been SAT 30, so we know that the goal associated with setting the SAT value properly was never satisfied. The Operator issued all of the other Configure-DSP commands (let's assume that the OFST command was properly issued) and moved on to the Coherence-Test plan, which begins with the V NPCG MAN command. The tutor notes that all of the expected commands from Configure-DSP have been issued and marks the plan COMPLETE. At the same time, it knows that the plan's goals are not satisfied, so it marks the plan UNSATISFIED. Once it observes the NPCG command, it notes that this command belongs to an inactive plan that follows the still active, albeit unsatisfied Configure-DSP plan. The tutor recognizes this situation as a goal failure impasse and decides to intervene. This is a case where the student's error was a result of mis-parameterizing the directive, which can only be detected as a goal failure since it does not violate any action constraints, hence it is detected after the student is apparently finished executing the plan.

4.3.3 Plan Dependency Violations

The reason for classifying certain behaviors as plan dependency violations is to aid the student who is confused about which plans are applicable for a given situation. The TDN describes the precedence constraints among plans. Thus, if the tutor observes the student executing a plan that is clearly inappropriate for the situation, it will intervene with tutorial advice. For example, consider once again Case 1 in Figure 3. The student omitted the OFST command and issued the NPCG MAN command, instead. The tutor matches this action with Coherence-Test plan, which is inactive. Clearly, this action was taken at the wrong time. This is a case where the tutor treats the command as a plan dependency impasse since resolving the error entails doing the Configure-DSP plan and it is not necessary to have the cognitive model figure this out when it is clearly a violation of the TDN constraints.

Due to the situated nature of tasks in this domain, however, the tutor does not automatically assume a command that does not match an active plan is a TDN constraint violation. The tutor evaluates the command with respect to the command preconditions of and plan goals of active plans. If it appears that the student was attempting to correct an anomaly or an unsatisfied precondition, then the tutor does not intervene. A good example of this is found in Case 3 of Figure 3. In this instance, the student issued the NIDLE REC command at the beginning of the sequence. The tutor recognizes that this command does not belong to the Configure-DSP plan, which is currently active, but that it does satisfy one of the NLOAD preconditions (i.e., it puts the NCB-program into the IDLE mode.) Consequently, the tutor does not intervene because the command clearly served a purpose for the current situation and the Operator was able to avoid a rejection of the NLOAD command.

This is the weakest category of impasse because it is the most difficult to correctly recognize. It requires being able to understand actions that do not fit into an expected behavior. Our next extension to the tutor will be to recognize and use severe device anomalies or failures to drive the model of expectation and situated plan attribution. The current effort focuses more on normal operations where the range of anomalies can be accommodated by fairly simple reactive plans or knowledge.

5. Impasse Explication

Once an impasse is recognized, the next step is to decide what to say to the student about it. The process of determining what to say is called impasse explication, which has two basic goals: first, explain the nature of the impasse in the context of the current situation, and second, determine how to resolve the impasse so that the tutor can teach the Operator the actions required by the situation. Our method for generating explanations for action constraint violations and goal failure impasses uses an executable expert cognitive model of the domain. The model is sensitive to the state of the devices at the point of the impasse; it reactively performs the task, identifies and repairs unsatisfied preconditions while concurrently generating an explanation for the student. Plan failure impasses are treated somewhat differently. Errors of this type do not require a detailed account of how to solve the problem, rather, we consider them to involve mis-executing a default plan.

5.1 Action Constraint Violations

The strategy for generating an explanation for an action constraint violation begins by applying the expert cognitive model at the impasse point; the explanation is constructed as the cognitive model resolves the impasse. We assume that the impasse was caused by a precondition violation that was overlooked by the student. The reason for the oversight could be due to not knowing the precondition, having a misconception about it, or not attending closely enough to the situation, which may have unexpectedly changed. In any of these cases, our cognitive modeling work (described in section 3) predicts that the impasse is a strategic intervention point and that the tutorial content is crucial to helping the student acquire the new skill.

The tutorial content is constructed by first recognizing the situational factors that led to the impasse and then reasoning about the steps necessary to resolve it. The tutor does not simply solve the problem at the impasse point; rather, it internally simulates applying the operator associated with the student's command. The student's command will normally have one or more parameter values (e.g., the NLOAD command requires a parameter value specifying the predicts set, such as JK). The tutor binds the student's command parameter value(s) to the operator precondition variables and proceeds with the task of attempting to apply the operator.

Figure 9 shows a portion of the problem space hierarchy that the Soar-based tutor searches in order to apply the Load-Predicts operator. Each of the preconditions is evaluated in the Verify-Operator-Preconditions problem space. If a precondition is not satisfied, the tutor searches the Repair-Unsatisfied-Precondition problem space for a way to resolve the impasse created by the unsatisfied precondition. Explanations are generated in both of these problem spaces when there is an unsatisfied precondition, detailing the source and nature of the problem and how to resolve it. Sometimes the unsatisfied precondition can be repaired by merely changing the command parameter, while in other cases it may be necessary to select and apply one or more command operators that will change the state of the device that caused the original precondition to be unsatisfied. The tutor internally simulates applying the commands to the devices, and includes these command applications in its explanation to the student.

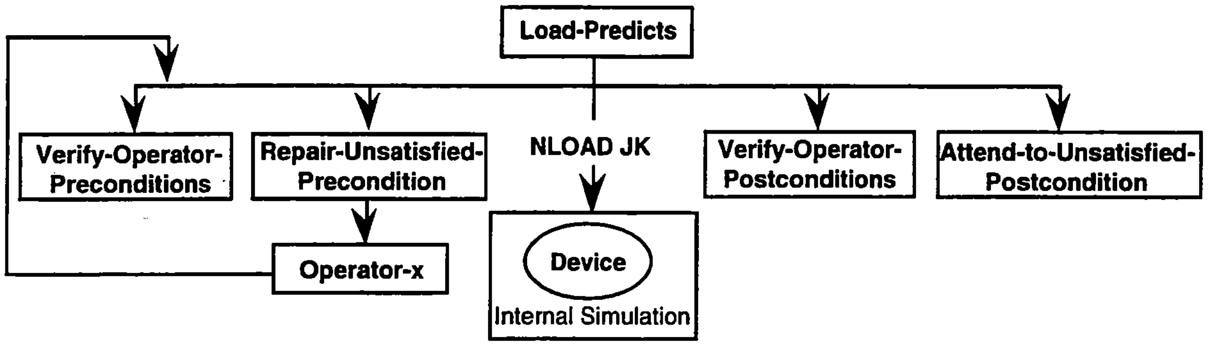

Load-Predicts
Verify-Operator-Preconditions
Repair-Unsatisfied-Precondition
NLOAD JK
Verify-Operator-Postconditions
Attend-to-Unsatisfied-Postcondition
Operator-x
Device
Internal Simulation

Figure 9 Problem Space Hierarchy of Expert Cognitive Model

To illustrate the explication process we have just summarized, let us return to one of the previous examples, shown as Case 2 in Figure 3. In this example the student issues the command, V NLOAD JK, and it is rejected, i.e., the simulator issues a message saying, REJECTED. NOT ALLOWED IN NRUN MODE. This interaction is observed by the tutor, and it recognizes the rejection as an action constraint violation. Knowing that the NLOAD command is associated with the Load-Predicts operator, the tutor selects this operator and subgoals into the Load-Predicts problem space.

As previously described, once the Soar-based tutor is in the Load-Predicts problem space, it selects an operator called Verify-Operator-Preconditions which creates another subgoal into the problem space for this operator. In the Verify-Operator-Preconditions space each of the Load-Predicts operator's preconditions (shown in Figure 5) is evaluated with respect to the current situation (i.e., with respect to the state of devices such as those shown in Figure 4.) In this particular case, the precondition named Load-Predicts-PC1 is unsatisfied because the NCB-Program device is in the RUN mode instead of the IDLE mode. Once detected, the tutor adds the information about the unsatisfied precondition to the explanation for the impasse (see Figure 10.1.)

Impasse Explanation			
Device	Attribute	Expected Value	Actual Value
NCB-Program	MODE	IDLE	RUN

Figure 10.1 Explanation of why the NLOAD command failed

Impasse Resolution			
Issue Command:	Affected Device	Affected Attribute	Value
NIDLE REC	NCB-PROGRAM	MODE	IDLE
NLOAD JK	VLBI-PREDICTS	LOADED?	YES
	CONFIG-TABLE	NCOMB	100.0
	...	...	...
	CONFIG-TABLE	TYPE	DOR

Figure 10.2 Explanation of how to resolve the impasse

The tutor must now find a way of satisfying the Load-Predicts-PC1 precondition, so it selects another operator in the Load-Predicts problem space called Repair-Unsatisfied-Preconditions. Again, the Soar-based tutor subgoals into a problem space for this operator, where it searches for a command operator that will change the NCB-Program from the RUN mode to the IDLE mode. The tutor finds an operator called Idle-NCB-Program which when applied will have the desired effects on the NCB-Program device. The tutor selects the Idle-NCB-Program operator and subgoals into a problem space that contains the same kinds of operators as the Load-Predicts problem space shown in Figure 9, and the process of verifying preconditions for the Idle-NCB-Program is performed. Given that the preconditions for this operator are satisfied, the Soar-based tutor internally simulates issuing the Idle-NCB-Program operator's command, NIDLE REC, and changes its internal model of the NCB-

Program device mode from RUN to IDLE. The NIDLE REC command is added to the explanation as shown in Figure 10.2.

Once this is accomplished, the Idle-NCB-Program subgoal terminates, and the tutor continues its problem solving in the Load-Predicts problem space. It begins by re-evaluating the previously unsatisfied Load-Predicts-PC1 precondition with respect to the revised device model. Since this precondition is now satisfied (and assuming that all of the other preconditions are also satisfied), the tutor applies the NLOAD command to its internal device model. This command is also added to the explanation for resolving the impasse (see Figure 10.2). The tutor terminates the Load-Predicts operator once it verifies that all of the postconditions have been satisfied, and the explanation is complete.

According to our model of skill acquisition [Hill and Johnson, 1993], the information contained in the explanation is precisely what the student should learn about the situational constraints of the failed command in order to avoid the same impasse in the future. In our modeling work, this new knowledge is initially acquired declaratively, and it is chunked into the form of a procedural skill as the student applies it in a problem solving context. The proceduralization of skill in the model is a direct result of the Soar architecture's learning capability provided by the chunking mechanism [Rosenbloom and Newell, 1986; Laird et al., 1987]. Moreover, our model of skill acquisition closely resembles the account given by Anderson [Anderson, 1983; Anderson, 1989; Anderson et al., 1990].

5.2 Goal Failure Impasse

Goal failure impasses are handled in much the same manner as action constraint violations; the primary difference is the level at which the tutor enters the expert cognitive model's problem space hierarchy. Instead of selecting a command operator, which was the case in example described in the last section, the tutor selects the plan-level operator (see Figure 1) corresponding to the plan whose goals were not satisfied. The tutor's goal in the plan operator's problem space is to select and apply command operators that will satisfy the plan's failed goals.

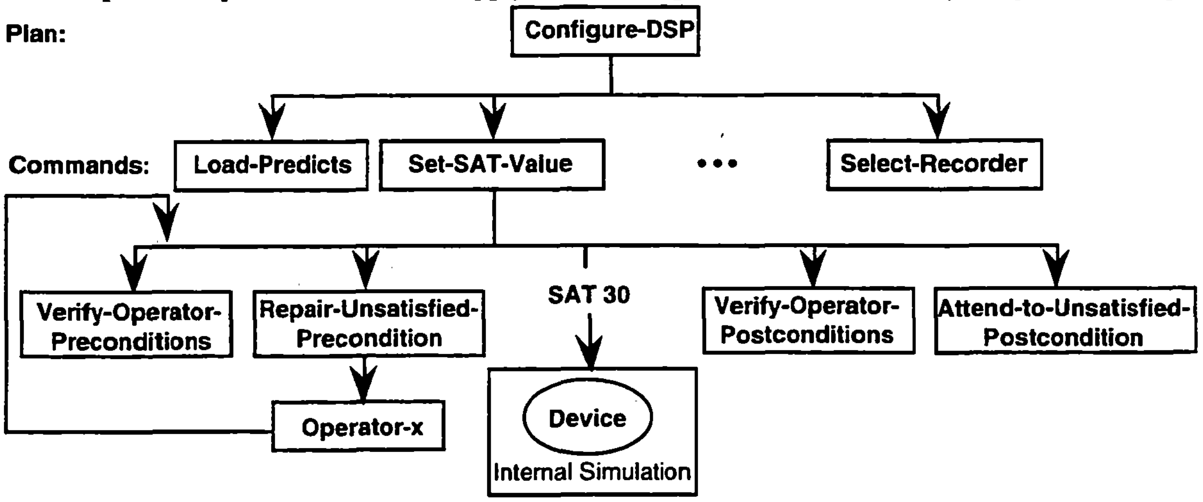

Plan:
Configure-DSP
Commands:
Load-Predicts
Set-SAT-Value
•••
Select-Recorder
Verify-Operator-Preconditions
Repair-Unsatisfied-Precondition
SAT 30
Verify-Operator-Postconditions
Attend-to-Unsatisfied-Postcondition
Operator-x
Device
Internal Simulation

Figure 11 Resolving a goal failure impasse related to the S-band attenuation (SAT) level setting.

A simple example of how the tutor explicates a goal failure impasse is shown in Figure 11. After the student finished the Configure-DSP plan, the tutor notices that the S-Band Attenuation (SAT) value was incorrectly set. The tutor subgoals into the Configure-DSP plan problem space and resolves the impasse by selecting and applying the Set-SAT-Value operator; this is a simple case where it was only necessary to correct one command of the Configure-DSP plan. One can imagine instances, however, where the plan's unsatisfied goals are not corrected so easily. In such cases, the tutor will search for and apply command operators in the plan's problem space until all of the goals are satisfied, and the explanation will reflect all of the commands that it selected to satisfy the goals.

5.3 Plan Failure Violations

This type of error appears to originate from a plan-level misconception or knowledge gap rather than at the command level. Either the student does not know which plan applies in the current situation or else does not know the plan's commands. Consequently, the tutor explains plan failure violations in terms of the plans that should be active in the current situation. The active plan's command operators are taught, but the details about their preconditions are omitted. In this way the student first becomes familiar with the default plan and its commands without being overwhelmed by the situational details of applying the commands.

6. The Tutor Improves over Time

The tutor was implemented to take advantage of Soar's learning capability. The Soar architecture embodies the idea that problem solving is a goal-oriented activity involving the search for and application of operators to a state in order to attain some desired results [Laird et al., 1987]. Search takes place in a hierarchy of problem spaces, where a problem space contains a set of operators and an initial state. The problem space hierarchy is traversed via subgoaling, which takes place whenever the problem solver cannot make any more progress toward a goal in the current problem space. Learning occurs when a subgoal yields a result; the result is stored in a production that summarizes the conditions for subgoaling and the result of the problem space search [Rosenbloom and Newell, 1986]. The chunk can be applied the next time a similar situation arises and the same results can be achieved without searching a subgoal problem space.

There are two consequences of having a tutor that learns: (1) the tutor's performance improves with experience, and (2) the nature of the tutor's problem solving changes as it learns. The first consequence has a bearing on the efficiency of the implementation. As the tutor gains experience with different student impasses, the knowledge of how to recognize and explicate these impasses is chunked into new productions. The chunks improve the tutor's performance significantly, since they limit the amount of search that is required to solve a familiar problem again. The second consequence affects the way we characterize our approach to tutoring, since the tutor's knowledge becomes more procedural as it gains experience.

6.1 Impasse Recognition Chunks

To illustrate how the tutor learns to recognize impasses, consider what happens when the tutor sees an action-response pair (i.e., a student command and device response); it subgoals into a problem space called Analyze-Action-Response, where it searches for a plan that contains the student's command. When the tutor finds a match, it marks the command in the plan as "matched". The tutor then decides whether there is a potential impasse, depending on whether the plan containing the command was active or not and whether the command was accepted or rejected by the device. Two types of potential impasses are identified in this problem space: plan-dependency-violation and action-constraint-violation. The Analyze-Action-Response problem space terminates once the tutor finishes analyzing the action-response pair.

The chunks built while searching in the Analyze-Action-Response problem space automatically recognize whether a specific action-response pair is a potential impasse for a given situation. Hence, the next time that the action-response pair is observed, the tutor will recognize whether there is a potential impasse or not without any further subgoaling or search.

6.2 Impasse Explication Chunks

Chunks are also built while the expert cognitive model explicates an impasse. Once the tutor has resolved a particular impasse, the resulting explication chunks are general enough to solve the same problem again even though the parameter values may be different. In addition, there is a transfer of knowledge about how to perform tasks such as verifying preconditions and postconditions, so as new impasses arise some of the previously learned chunks will be applied.

6.3 Improvement

There is a significant improvement in the tutor's performance after it has chunked the problem space hierarchies used to recognize and explicate student impasses. Figure 12 shows the amount of time it takes the tutor to handle action constraint violations for the NLOAD and SAT commands before and after learning.

Command	Recognize Impasse		Explicate Impasse		Total		Ratio: Before/After (Total)
	Before	After	Before	After	Before	After	
NLOAD	0.20	0.08	8.70	0.61	8.90	0.69	13::1
SAT	0.19	0.08	2.70	0.32	2.92	0.40	7::1

Figure 12 Tutor's performance on action constraint impasse before and after learning, measured in seconds

Note that the time it takes to recognize the impasse is constant among these commands, before and after learning; it takes roughly 0.20 seconds to recognize an impasse involving either command before learning, and it improves to 0.08 seconds after learning. On the other hand, the amount of time it takes to explicate an action constraint violation varies, depending on how many preconditions and postconditions must be checked for the command. We chose the NLOAD and SAT commands for this example because they provide upper and lower bounds for this type of impasse. The NLOAD command has the greatest number of preconditions and postconditions and the SAT command has the fewest.

7. Comparison to Model Tracing and Plan Recognition Approaches

We compare our tutoring method to two approaches that have been used in a number of intelligent tutoring systems, namely, model tracing [Anderson, 1990; Reiser et al., 1985; Ward, 1991] and plan recognition [Johnson, 1990; Warriner, 1989; Rickel, 1988]. Our tutor resembles elements of both model tracing and plan recognition, though it cannot be completely characterized as one or the other according to current definitions. In fact, we would suggest that the current conception of model tracing should be extended to allow for model tracing at different levels of abstraction. To clarify what this means, consider how model tracing and plan recognition address the tutoring intervention problem.

7.1 The Intervention Decision

Model tracing recognizes errors using an executable performance model of the student to search for production rule paths that account for the student's behavior. If an action can only be accounted for via a mal-rule application, then the tutor concludes that the student made an error and passes this interpretation to the pedagogical model [Anderson et al., 1990]. This differs from our approach in that our tutor does not detect errors by running a cognitive simulation of the student. Rather, our tutor focuses on recognizing impasses, and more closely resembles plan recognition than model tracing when deciding whether to intervene.

Whereas model tracing uses procedural knowledge to detect student errors, plan recognition approaches to student analysis typically match the observed behavior to a declarative description of action. This either involves matching and interpreting the student's action sequence using a library of plans [Johnson, 1990; Calistri, 1990], or else interpreting the action with an action grammar or procedure net description of the task [Burton, 1982; Rickel, 1988; Warriner et al., 1990]. In a plan matching approach, errors are detected with mal-plans or difference rules, while in the action grammar case an error is any action that can't be parsed by the grammar, or which is only parsed with a model that includes buggy rules. Our method of impasse recognition bears some resemblance to these approaches in that the tutor initially matches student actions to declarative plan descriptions and recognizes potential impasses when actions do not fit the expectations generated by the plans and their goals. It differs in that the tutor does not use manually encoded mal-plans or difference rules to recognize the impasse; as the tutor gains experience it effectively generates its own mal-plan rules in the form of recognition chunks. Hence, our tutor learns when to intervene. At an abstract level it traces a performance model, but the difference from standard model tracing is that it does not attempt to generate mental states to do so. It traces the student's progress in enough detail to make predictions of what plans the student might be following, and no more.

7.2 Deciding What to Say

One of the strengths of the model tracing is its ability to give a reasoned explanation for an error, once it is detected. In this regard, our tutor resembles a model tracer; it generates explanations using an executable cognitive model that diagnoses possible causes for the error and suggests how to resolve the current impasse. The difference between the two approaches is that model tracing relies partly on its knowledge of mal-rules to generate the explanation, while our tutor learns to recognize the causes for errors while applying an ideal performance model to the impasse. The declarative representations of operator preconditions used by the cognitive model end up being proceduralized, to a large extent, as the tutor gains experience. These chunks have the same effect as a mal-rule in that they can be used to explicate an impasse.

7.3 Generality of the Approach

The simplicity of our approach is partly due to the constrained nature of the task being tutored. We assume at the outset that the mission that the student is performing is known. This allows us to make predictions of what plans the students are likely to be following. All model tracing and most plan recognition systems make similar assumptions.

However, it should be possible to weaken this assumption and retain the same basic approach. Device failures and anomalies can lead the Operator to carry out different or additional plans. Operators sometimes get confused about which plans are appropriate for which missions, and carry out inappropriate plans. These cases can simply be treated as alternative sources of expectations for plans. Tutorial intervention will continue to focus on the appropriateness of attributed plans to the situation at hand.

8. Results

We have made a number of claims about how students learn, and based on this, how an intelligent tutor can effectively teach in a monitor and control domain. In order to assess the veracity of these claims, we plan to evaluate the tutoring system in two ways: first, by testing it with novice operators, and second, by soliciting the reactions of expert operators and system engineers to the method and accuracy of the advice given by the tutor.

The tests involving novice operators will divide the participants into two groups: a control group and a test group. Both groups will be given identical tasks to perform on the link monitor and control training simulator, and the simulator devices will also start in the same state for both groups. The primary difference between the two groups will be that the test group will receive advice from the tutor, while the control group will not. We will measure the task performance of each group over a number of trials on the same task; data will be collected on how the student performed on each trial in terms of (1) time elapsed, (2) number of commands needed to complete the task, and (3) the number of impasses or errors committed. This data will help us make a formative evaluation of how helpful the tutor is with respect to just using the simulator.

9. Conclusions

We have introduced a new approach to tutoring that focuses on recognizing student impasses through the use of situated plan attribution. Unlike plan recognition, we assume that behavior is situated rather than plan-based and therefore cannot necessarily be recognized as plans. Plans serve as resources for action which must ultimately be situated in the world. Likewise, we do not attempt to understand every student action as is the case with both model tracing and plan recognition. Instead, our tutor focuses on recognizing impasse situations. From the tutor's perspective, the task of recognizing certain impasses is simplified by listening to the device (i.e., directive rejections) rather than trying to generatively recognize the action as an error. Once an impasse is recognized, our expert cognitive model explicates the situation at the impasse point, thereby incorporating one of the strengths of model tracing.

10. References

[Anderson, 1983] John R. Anderson. The architecture of cognition. Harvard University Press, 1983.

[Anderson, 1988] John R. Anderson. "The Expert Module". Foundations of Intelligent Tutoring Systems, edited by Martha C. Polson and J. Jeffrey Richardson. Lawrence Erlbaum Associates, Inc, 1988.

[Anderson, 1989] John R. Anderson. "Use of analogy in a production system architecture," in Similarity and Analogical Reasoning, edited by Stella Vosniadou and Andrew Ortony. Cambridge University Press, New York, 1989.

[Anderson et al., 1990] John R. Anderson, C. Franklin Boyle, Albert T. Corbett and Matthew W. Lewis. "Cognitive modeling and intelligent tutoring," Artificial Intelligence, 42, pp 7-49, 1990.

[Brown and VanLehn, 1980] John Seely Brown and Kurt VanLehn. "Repair theory: a generative theory of bugs in procedural skills," Cognitive Science (4), pp. 379-426, 1980.

[Burton, 1982] Richard R. Burton. "Diagnosing bugs in a simple procedural skill," Intelligent Tutoring Systems, edited by D. Sleeman and J.S. Brown. Academic Press, Inc., 1982.

[Calistri, 1990] Randall J. Calistri. "Classifying and detecting plan-based misconceptions for robust plan recognition." Ph.D. Dissertation, Technical Report No. CS-90-11, Department of Computer Science, Brown University.

[Chen et al., 1991] Thomas T. Chen and Diann Barbee, "Integrating an intelligent tutoring system with an existing simulator," Proceeding of the 1991 Conference on Intelligent Computer-Aided Training, NASA/Johnson Space Center, Houston, Texas, November 20-22, 1991.

[Fayyad and Cooper, 1992] Kristina Fayyad and Lynne Cooper, "Representing Operations Procedures Using Temporal Dependency Networks," Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations, SPACEOPS-92, Pasadena, CA, 16-20 November, 1992.

[Galdes, 1990] Deborah K. Galdes. An empirical study of human tutors: the implications for intelligent tutoring systems. Ph.D. Dissertation, The Ohio State University, 1990. Order Number 9031068, UMI Dissertation Services, Ann Arbor, Michigan.

[Hill and Johnson, 1992] Randall W. Hill, Jr. and W. Lewis Johnson. "Designing an intelligent tutoring system based on a reactive model of skill acquisition," World Conference on Artificial Intelligence in Education (AI-ED 93), Edinburgh, Scotland, 1993.

[Hill and Lee, 1992] Randall W. Hill, Jr. and Lorrine Lee, "Situation Management in the Link Monitor and Control Operator Assistant," Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations, SPACEOPS-92, Pasadena, CA, 16-20 November, 1992.

[Johnson, 1990] W. Lewis Johnson. Understanding and Debugging Novice Programs. Artificial Intelligence, 42, pp. 51-97, 1990.

[Laird et al., 1987] John E. Laird, Allen Newell and Paul S. Rosenbloom. "Soar: An architecture for general intelligence," Artificial Intelligence, 33(3), 1987.

[Newell, 1990] Allen Newell. Unified Theories of Cognition. Harvard University Press, 1990.

[Reiser et al., 1985] Brian J. Reiser, John R. Anderson and Robert G. Farrell. "Dynamic student modelling in an intelligent tutor for lisp programming," Proceedings of IJCAI-85, Los Angeles, CA, 1985.

[Rickel, 1988] Jeff Rickel. "An intelligent tutoring framework for task-oriented domains," Proceedings of the International Conference on Intelligent Tutoring Systems, Montreal, June 1-3, 1988.

[Rosenbloom and Newell, 1986] Paul S. Rosenbloom and Allen Newell. "The chunking of goal hierarchies: a generalized model of practice," Machine Learning, Volume II, pp. 247- 288, edited by Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986.

[Suchman, 1987] Lucy A. Suchman. Plans and situated actions. Cambridge University Press, New York, 1987.

[Ward, 1991] Blake Ward. ET-Soar: Toward an ITS for Theory-Based Representations. Ph.D. Dissertation, CMU-CS-91-146, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

[Warinner et al., 1990] Andrew Warinner, Diann Barbee, Larry Brandt, Tom Chen, and John Maguire. "Building an intelligent tutoring system for procedural domains," Proceedings of the First CLIPS Conference, pages 881-892, NASA/Johnson Space Center, Houston, TX, 1990.

11. Acknowledgements

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Dr. Johnson was supported in part by the Advanced Research Projects Agency under contract number N00013-92-K-2015. Views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of the U.S. Government or any agency thereof.

We thank Paul Rosenbloom and the members of the Soar group at USC-ISI for their helpful comments on this paper. We also thank Lynne Cooper and members of the Link Monitor and Control Operator Assistant team at JPL for their input.

N96-14987

The Effects of a Dynamic Graphical Model During Simulation-Based Training of Console Operation Skill

John D. Farquhar
Department of Instructional Technology
University of Georgia

J. Wesley Regian
Technical Training Research Division
Air Force Armstrong Laboratory

ABSTRACT

LOADER is a Windows-based simulation of a complex procedural task. The task requires subjects to execute long sequences of console-operation actions (e.g., button presses, switch actuations, dial rotations) to accomplish specific goals. The LOADER interface is a graphical computer-simulated console which controls railroad cars, tracks, and cranes in a fictitious railroad yard. We hypothesized that acquisition of LOADER performance skill would be supported by the representation of a dynamic graphical model linking console actions to goal and goal states in the "railroad yard." Twenty-nine subjects were randomly assigned to one of two treatments (i.e., dynamic model or no model). During training, both groups received identical text-based instruction in an instructional-window above the LOADER interface. One group, however, additionally saw a dynamic version of the bird's-eye view of the railroad yard. After training, both groups were tested under identical conditions. They were asked to perform the complete procedure without guidance and without access to either type of railroad yard representation. Results indicate that rather than becoming dependent on the animated rail yard model, subjects in the dynamic model condition apparently internalized the model, as evidenced by their performance after the model was removed.

INTRODUCTION

This paper is concerned with the communication and representation of knowledge that aid in the acquisition of console operation skill. Console operations, which are common in industrial and defense settings, require the execution of sequences of control-panel actions such as the pressing of a button or the rotation of a dial. Console operation skill is arguably constructed of various knowledge types or forms of knowledge. While general theories of cognition and learning have asserted that all knowledge can be described as either declarative or procedural (Anderson, 1983), theories specific to console-operation knowledge have made additional distinctions (Polson & Kieras, 1984; Card, Moran, & Newell, 1980).

One central component in the theories of console-operation knowledge is an element describing the goal of the operation. The study reported in this paper examined the effects of communicating various goal states of a console-operation task through a dynamic graphical model. The console-operation task performed by subjects was a computer-simulation called LOADER. The LOADER interface is a graphical computer-simulated console which controls railroad cars, tracks, and cranes in a fictitious railroad yard. LOADER is designed to be a laboratory analog of procedural console operations and process control tasks. The advantage of using the LOADER simulation for this study was the capability to represent goal states in a dynamic visual display.

CONSOLE OPERATION KNOWLEDGE

The theory of console-operation knowledge described by Polson and Kieras (1984) classifies that knowledge into two components: the *job-task representation,* and *how-it-works knowledge.* How-it-works knowledge is knowledge of the underlying structure or processes of a device. The job-task representation includes *job-situation knowledge* (knowledge of tasks that the device can perform) and *how-to-do-it knowledge* (knowledge of how to carry out those tasks).

Overall, the job-task representation describes the relationship between the capabilities of the device and the processes for carrying out those tasks. A specific job goal (or capability) will identify an appropriate job-selection rule (or process) that should be performed. Additionally, how-it-works knowledge will aid the selection of appropriate rules when those rules can be inferred from the internal structure of a device (Kieras & Bovair, 1984).

Using the LOADER simulation as an example, a specific goal might be the positioning of a rail car to the appropriate dock. This goal identifies a sequence of control panel operations such as enabling control of the rail lines and switching the correct track. The LOADER interface displays the outcomes of the action with a bird's-eye view of the rail yard.

The GOMS model (Card, Moran, & Newell, 1980) asserts four categories of device-operation knowledge: (1) Goals, (2) Operations, (3) Methods, and (4) Selection rules. Goals and subgoals define the outcomes of a task. Operations are the individual actions that are performed by the user and the device to reach the goals and subgoals. Methods are the procedures, or sets of operations, performed in achieving each subgoal. Selection rules assert appropriate methods as determined by the subgoal.

Using the LOADER example again, the goal of loading a particular rail car will be made up of various subgoals (e.g. positioning rail cars, lifting and loading canisters). Each action performed on the way to achieving the goal, such as an individual switch setting, is an operation. A set of operations, that when performed sequentially achieve a goal or subgoal, is a method. Selection rules include information about goals and methods in order to match appropriate methods with the goals they achieve. These "rules of selection" determine methods from asserted goals.

Selection rules can be formalized as a *production rule.* Production rules, or productions, are condition-action pairs often represented in the form: **if** condition, **then** action (Anderson, 1983). Within LOADER, the goal of positioning a rail car from rail line #1 to rail dock #3 would assert the set of operations that would include Enabling Rail Car Control, Setting Rail Dock #3, and Engaging Rail Car South. The LOADER interface displays the initial and final arrangement of the rail cars through a dynamic graphical model.

The use of a graphical model in the acquisition of console and device operation has been found to be beneficial when the model addresses knowledge of the internal operations of the device (or how-it-works knowledge). Kieras and Bovair (1984) found that a *device model,* affected the acquisition of device operation skills by increasing the speed of acquisition, accuracy of retention, and speed of performance. While this model described the **internal** representation of device mechanisms, the study reported here investigated the use of an **external** representation of device goals and goal states.

Additional research in the area of knowledge communication and representation of console and procedural skill has found that some types of knowledge are less beneficial. In a study by Reder, Charney, and Morgan (1986), certain types of instructional elaborations facilitated the acquisition of procedural computer skills while others did not. Syntactic elaborations (i.e. how-its-done knowledge) improved learning outcomes, whereas conceptual elaborations (i.e. what-it-is knowledge) produced little effect.

Conclusions from previous research indicates that the type of knowledge communicated affects the performance outcomes in the acquisition of console operation skill. Additionally, the presentation of a device model which represents internal device mechanism aids the acquisition of procedural skill associated with console operation. The study described below was concerned with the effects of providing a meaningful model of goals and goal states within the context of interactive practice sessions.

METHOD

Design

The independent variable under investigation was the availability of a dynamic graphical model during acquisition of the task. The dynamic graphical model display, centered at the top of the screen, provided a birds-eye view of the loading task which dynamically responded to control panel actions. Animated sequences demonstrated realistic responses of crane-console interactions. During the practice trials of the task, the model was displayed to one of the two treatment groups. The immediately following retention test, however, required that both groups perform the task without access to the model. Subjects were randomly assigned to one of the two treatments (model and no-model groups).

Subjects

Twenty-nine subjects supplied through a temporary employment agency were paid for their participation in the study. Selection was limited to high-school graduates between the ages of 18 and 27 years. One subject was excluded from failure to complete the final test trial. Subjects completing the experiment early were given additional assignments.

Materials

A console operations task was designed to loosely represent a real-world task in remote crane control arm operation. The task (LOADER) involves the operation of a console to lift, transport, and load canisters from storage bins into rail cars. While skilled console operators follow an interacting set of proscribed procedures, a single 51-step procedure was selected for this experiment.

To provide instruction, practice, and testing for the task, a computer simulation was developed. The simulation displays a complete control panel consisting of a set of 29 interrelated buttons, knobs, and switches. The controls respond to clicks of a mouse with numerous switch and knob settings, meter readings, indicator lights, and an occasional beep. All panel components are appropriately labeled and organized by function on the lower half of the computer screen (see figure 1).

The upper half of the computer screen is reserved for on-screen messages, additional user options (i.e. "Assistance"), and the dynamic graphical display. On-screen messages include the practice or test trial number, a description of the step to perform (e.g. "Set Crane Control to 'Enable'"), and feedback messages for incorrect performance (e.g. "Incorrect. Click as indicated by the arrow.").

Selecting the "Assistance" option reveals a large red arrow indicating the location of the next button or switch, or knob position. An incorrect action in the procedure would automatically select the assistance option and display the red arrow.

The simulation software was developed using the ToolBook programming environment and was delivered via a laboratory of 22 individual computer stations. The stations were equipped with Compaq 486/33L microprocessors, VGA-quality monitors, keyboards, and right-handed, three-button mice.

Procedure

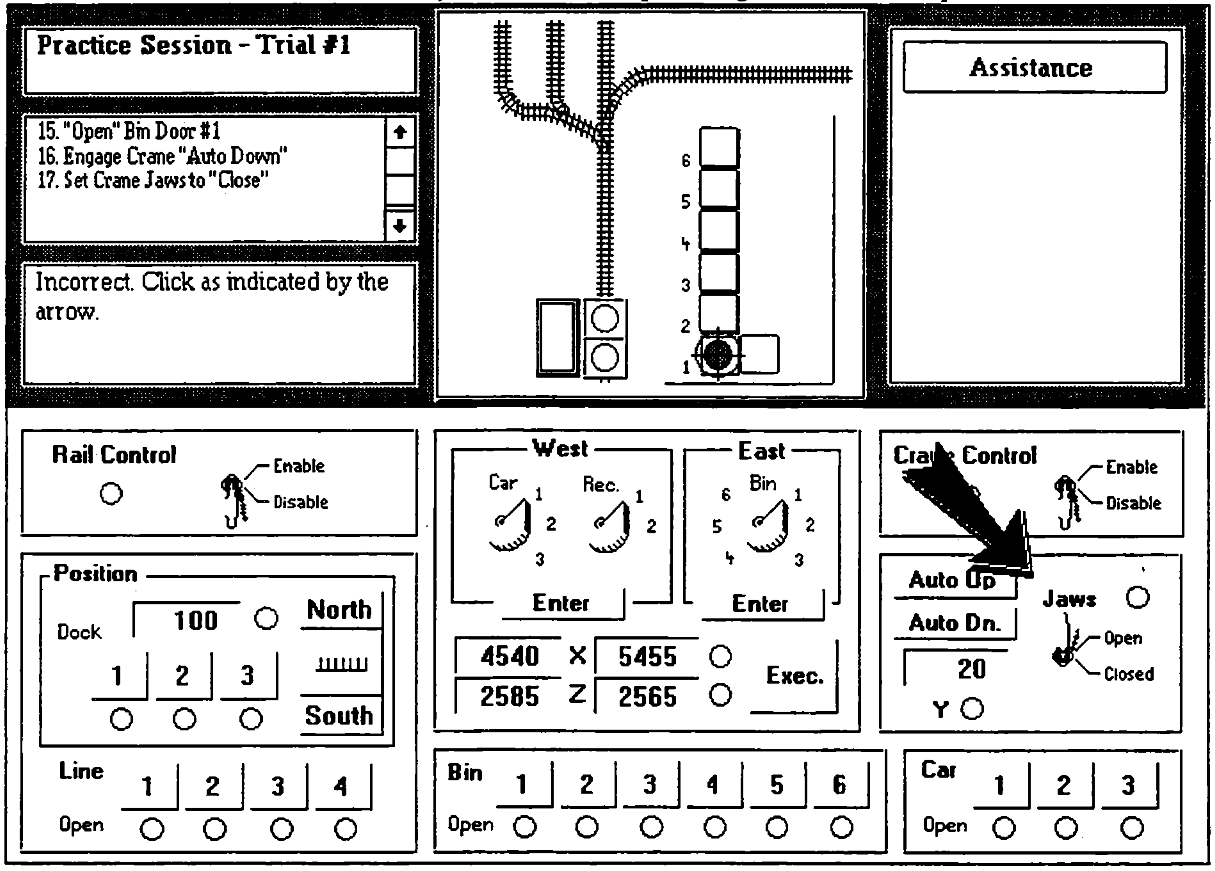
Subjects were brought into the laboratory in groups of 15 individuals. One of two treatments was randomly selected for the group. After a brief orientation in use of the equipment, the subjects were instructed to proceed through the program at their own pace. The program included three phases: an instructional phase, a practice phase, and a testing phase.

The instructional phase provided an introduction to the task of operating the crane control arm. This brief tutorial introduced and explained the task of crane control operations through a dynamic graphical model or display. The graphical model illustrated how the actions of the crane might appear as if the situation was viewed through a window. The instruction did not provide any specific instructions in crane operation, nor did it show or mention the related control panel. Finally, this phase concluded with directions of how the user should interact with the system during the practice and test phases.

During the practice phase, subjects completed 6 trials of the crane control procedure. A text prompt appeared on the screen to direct the subject in performing the individual steps. An on-screen "Assistance" option was available that when selected indicated the location of the next button or switch with a large red arrow.

In this practice phase, one treatment group had available the dynamic graphical model, an on-screen display identical to the graphical model presented in the instruction phase. During practice, however, this model represented crane arm-control panel interactions through dynamic changes of the display. The dynamic model was not present for the no-model group.

Following six practice trials, subjects entered the test phase. In this phase, subjects made three attempts at performing the crane control procedure without the aid of text prompts or assistance. Neither treatment group had available the dynamic graphical model during testing trials. If an error was made during testing, the red assistance arrow would direct the subject to the next step, forcing ultimate correct performance.


Practice Session - Trial #1

15. "Open" Bin Door #1
16. Engage Crane "Auto Down"
17. Set Crane Jaws to "Close"

Incorrect. Click as indicated by the arrow.

Figure 1 LOADER Interface.

RESULTS

Multiple analysis of variance on repeated measures were performed on completion times and errors. Mean completion times and errors for the nine trials (six practice, three test) are shown in Figures 2 and 3 respectively. Nearly equivalent completion times and errors for the model and no-model groups were recorded for each of the six practice trials. Therefore, differences between times [F(1,27) = .018, p = .8929] and errors [F(1,27) = .708, p = .4074] were insignificant. Test trials, however, yielded significant effects between groups for both completion times, [F(1,27) = 12.33, p = .0016] and errors [F(1,27) = 21.342, p = .0001].

These results indicate the following:

- the model and no-model groups performed equivalently during the 6 practice sessions,
- the model group performed the task faster than the no-model group during testing,
- the model group performed fewer errors than the no-model group during testing.

DISCUSSION AND SUMMARY

The presence of a dynamic model representing the goals of the procedural task during the acquisition phase of a procedural skill appears not to slow the learning process. In addition, the presence of the model significantly improves the performance of the skill with regard to the time and errors made during a testing phase even though the testing phase consisted of performance without the presence of the graphical model. That is, in the process of learning to carry out a specific sequence of actions to accomplish a goal, the operator came to imagine the corresponding events in the railroad yard, even if the operator could not actually see the yard while operating the console.

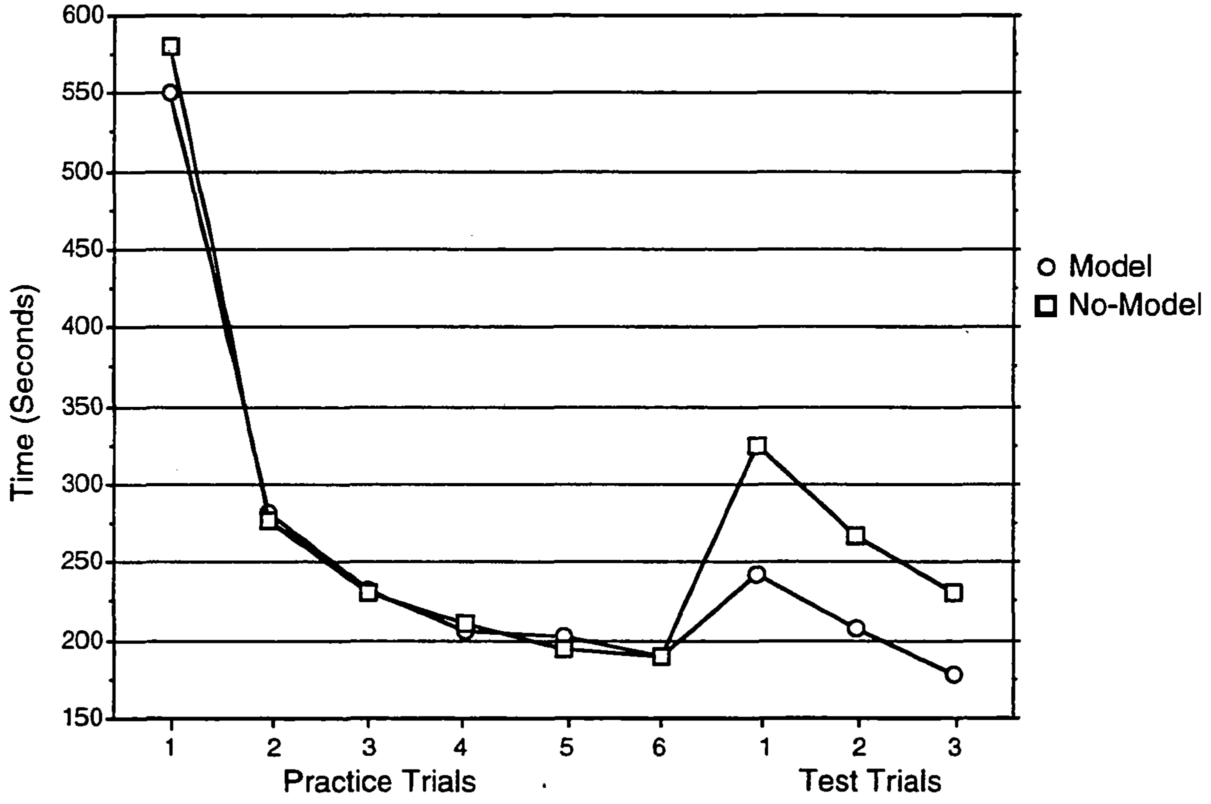


Figure 2 Time per trial.

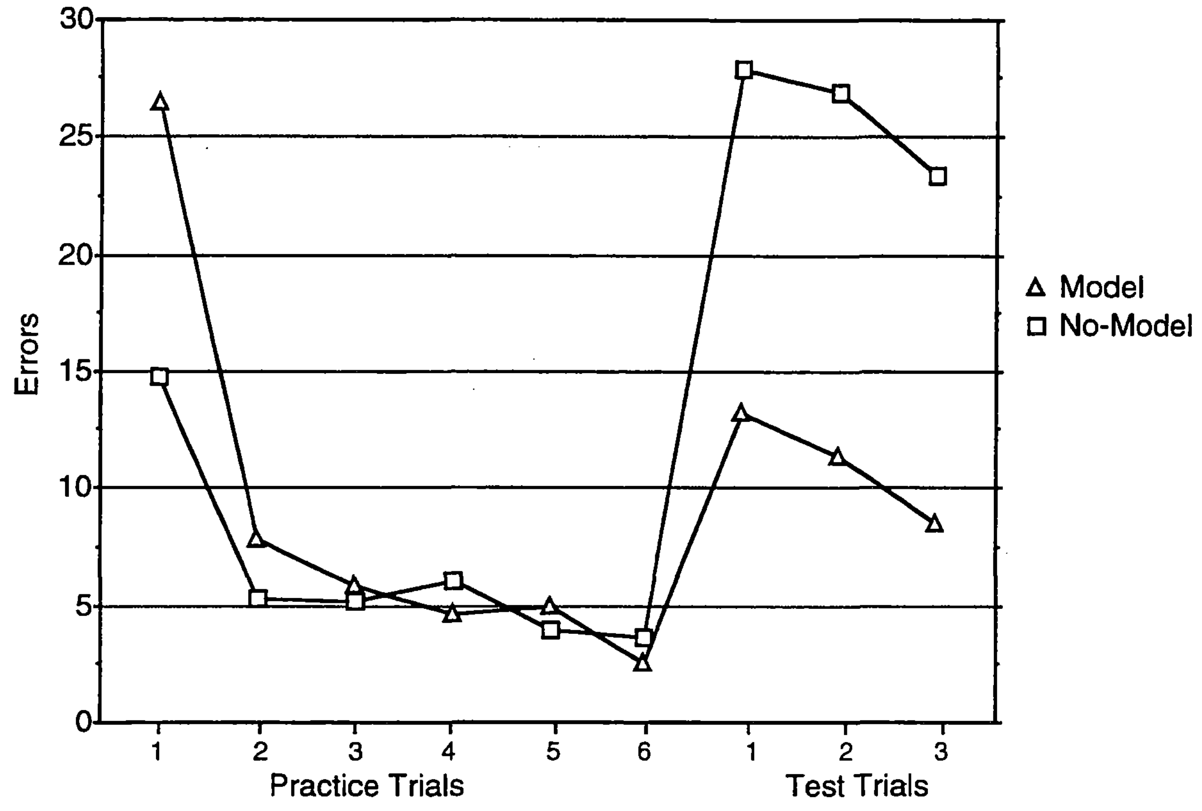


Figure 3 Errors per trial.

REFERENCES

Anderson, J.R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Card, S.K., Moran, T.P., & Newell, A. (1980). Computer text editing: An information-processing analysis of a routine cognitive skill. *Cognitive Psychology,* 12, 32-74

Kieras, D.E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science, 8*(3), 255-273

Polson, P.G. & Kieras, D.E. (1984). A formal description of users' knowledge of how to operate a device and user complexity. *Behavior Research Methods, Instruments, & Computers. 16*(2), 249-255

Reder, L.M., Charney, D.H., & Morgan, K.I. (1986). The role of elaborations in learning a skill from an instructional text. *Memory and Cognition, 14*(1), 64-78

AUTHORS

John D. Farquhar is a doctoral student of Instructional Technology at the University of Georgia.

J. Wesley Regian is Senior Scientist with the Technical Training Research Division at the Air Force Armstrong Laboratory. Address correspondence to: John D. Farquhar, Dept. of Instructional Technology, 630 Aderhold, University of Georgia, Athens, Georgia, 30602, or e-mail: johnf@moe.coe.uga.edu.

Funds for this research were provided by the U.S. Air Force, Technical Training Research Division, Brooks Air Force Base, San Antonio, Texas and the Air Force Office of Scientific Research, Bolling Air Force Base, Washington, D.C.

Advanced Technology Training System on Motor-Operated Valves

Bradley J. Wiederholt
T. Kiki Widjaja
Galaxy Scientific Corporation
Atlanta, GA

Joseph Y. Yasutake
Electric Power Research Institute
Palo Alto, California

Hachiro Isoda
Central Research Institute of Electric Power Industry
Tokyo, Japan

This paper describes how features from the field of Intelligent Tutoring Systems are applied to the Motor-Operated Valve (MOV) Advanced Technology Training System (ATTS). The MOV ATTS is a training system developed at Galaxy Scientific Corporation for the Central Research Institute of Electric Power Industry in Japan and the Electric Power Research Institute in the United States. The MOV ATTS combines traditional computer-based training approaches with system simulation, integrated expert systems, and student and expert modeling.

The primary goal of the MOV ATTS is to reduce human errors that occur during MOV overhaul and repair. The MOV ATTS addresses this goal by providing basic operational information of the MOV, simulating MOV operation, providing troubleshooting practice of MOV failures, and tailoring this training to the needs of each individual student.

The MOV ATTS integrates multiple expert models (functional and procedural) to provide advice and feedback to students. The integration also provides expert model validation support to developers. Student modeling is supported by two separate student models: one model registers and updates the student's current knowledge of basic MOV information, while another model logs the student's actions and errors during troubleshooting exercises. These two models are used to provide tailored feedback to the student during the MOV course.

INTRODUCTION

In February, 1989, the Electric Power Research Institute (EPRI) and the Central Research Institute of Electric Power Industry (CRIEPI) in Japan initiated a joint research program to investigate various interventions to reduce personnel errors and inefficiencies in the maintenance of nuclear power plants. One maintenance task identified as being particularly susceptible to human errors was the overhaul of Motor-Operated Valves (MOVs). MOVs are electro-mechanical devices used throughout nuclear power plants in many different systems, both safety-related and balance-of-plant. Because these valves are so numerous, systemic problems with any part of an MOV assembly can have negative effects on plant performance, including increases in (1) person-hours required for repair, (2) personnel radiation exposure, and (3) outage length (ANACAPA report).

A study of the MOV actuator overhaul task showed that MOV maintenance personnel must deal with a technically complex system, have limited access to the real equipment for either on-the-job or classroom training, and rely on experience (either their own or a group leader's) rather that written procedures for troubleshooting and diagnosis of the MOV actuator. In addition, requirements for the application of the personnel's troubleshooting knowledge are infrequent, requiring retraining or practice to maintain proficiency in the diagnostic-intensive parts of their job. The study also indicated that two separate occupational groups perform maintenance on MOV actuators: mechanics and electricians. Poor understanding of one another's skills and knowledge further increases the chance of on-the-job maintenance errors.

To train personnel to perform a task exhibiting the above characteristics, nuclear plant instructors currently augment stand-up classroom training with hands-on exercises. However, the lack of availability of both instructor time and real-world equipment limits the amount of effective hands-on training that instructors can provide. Informal interviews with instructors pointed to a need to provide students with training that could 1) increase student's exposure to MOV actuator troubleshooting tasks, 2) reduce the amount of subjectivity in evaluating a student's troubleshooting skills and knowledge, and 3) reduce the amount of instructor-led MOV retraining.

As a result of these needs and task analyses, CRIEPI and EPRI initiated the development of a computer-based training system to train nuclear power plant personnel in the diagnosis of MOV actuator problems. Initial review identified simulation-oriented tutoring as a training method most suited to help utility maintenance personnel gain a better understanding of MOV actuator operations and diagnosis (Widjaja analysis, 1992). For adult learners, various forms computer-based simulations of equipment for the purpose of teaching diagnostic skills have been repeatedly demonstrated to be effective within military and industrial environments. A previous EPRI project demonstrated the effectiveness of computer simulation to provide diagnostic training in the area of diesel generators (Johnson, et al., 1986, Maddox, et al., 1986).

Often these simulations are augmented with training and feedback that are tailored to the individual student's needs and understanding of the system. This adaptability of the training to the student is important to eliminate unnecessary training, focus students on their areas of weakness, and increase the acceptance of the training by the student. This simulation-oriented, "intelligent" feedback approach is particularly effective for teaching troubleshooting in systems that are 1) technically complex, 2) have varied instrumentation and test points, 3) require practice on the part of maintenance personnel to develop and maintain diagnostic proficiency, 4) have either formal or informal troubleshooting procedures, and 5) have limited access to the real equipment for training purposes (Norton, et al., 1991).

The MOV Advanced Technology Training System (ATTS) is the result of this development effort. The MOV ATTS provides basic operational information of the MOV, simulates MOV operation, provides 17 troubleshooting practice problems allowing student to diagnose MOV failures, and tailors this training to the needs of each individual student. The MOV ATTS is designed to be used by maintenance personnel of MOV actuator overhaul as a supplement to current initial training and refresher courses, and is not intended to replace these courses.

The remainder of this paper describes the internal features of the MOV ATTS, focusing on those aspects of the training that are based in the field of Intelligent Tutoring Systems.

GENERAL ARCHITECTURE

To support the goal of providing training that is tailored to the needs of each individual student, we have employed concepts and techniques from the field of Intelligent Tutoring Systems (ITS)[1] in the development of the MOV ATTS. A widely accepted representation of the architecture for an ITS is given in Figure 1 (Burns & Capps, 1988). The major components of an intelligent tutoring system are: 1) the *student*, 2) the computer *interface*, 3) the *instructional environment*, and 4) the *tutoring component*, which is in turn made up of models of the student, expert and instructor.

For purposes of the MOV ATTS, the computer interface consists simply of the physical interface (a computer monitor, mouse, keyboard and optional touch screen). The instructional environment, on the other hand, is the set of actions taken by the student and the feedback generated by the system during which the student performs the learning task. In the case of the MOV ATTS, there are two distinct instructional environments.

[1] General information on Intelligent Tutoring Systems can be found in Massey, et al., 1988 and Polsen and Richardson, 1988.

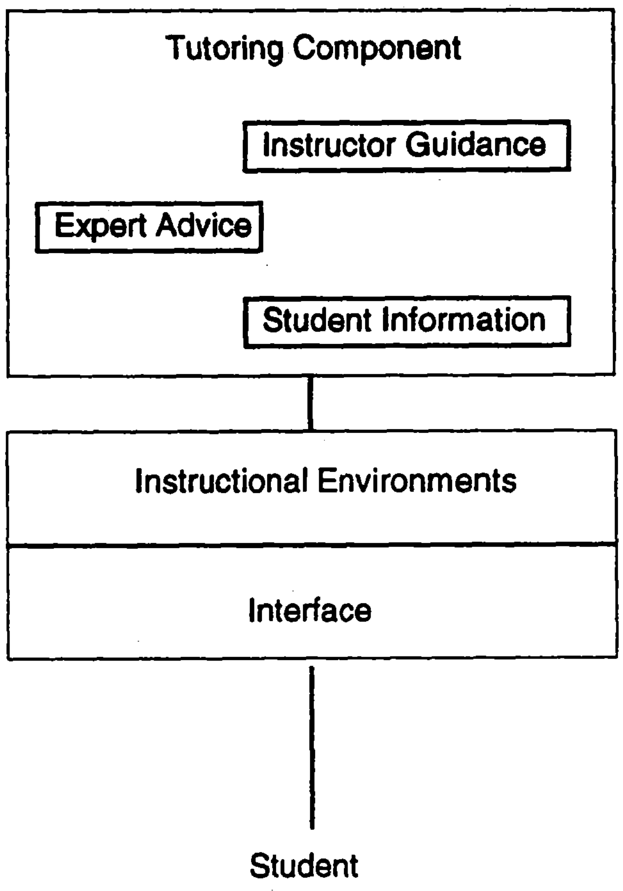


Figure 1. Simplified ITS Architecture.

In the first instructional environment, the student receives basic information on the MOV actuator, its mechanical components, and its electrical components. The student is occasionally called upon to answer question regarding various aspects of these topics. This self-paced material is presented to refresh the student's knowledge of the background material and to prepare them for the troubleshooting exercises to follow.

The MOV ATTS's second instructional environment is the system simulation (the troubleshooting module). Within the system simulation (see Figure 2), the student is actively involved in applying their knowledge to diagnose and locate failed components on a simulated MOV actuator. Within this environment, the student is testing components, receiving feedback when mistakes are made, and is trying to locate MOV failures as quickly as possible without making errors.

Note that according to Figure 1, both the basic information and system simulation instructional environments can be supported by the tutoring component. In fact, various portions of the MOV ATTS's tutoring component (the student, expert, and instructional models) are distributed among both instructional environments. The remainder of this paper will examine these components, where they are used in the MOV ATTS, how they interact with one another, and the benefits derived from their application.

STUDENT INFORMATION

In an ITS, the student information model is used to capture information on the student's current understanding of the subject matter being taught. The MOV ATTS actually uses two different student models: the declarative information model and the action history model.

The MOV ATTS declarative information student model contains 20 basic learning objectives similar to the example given above. These low level objectives were derived from an earlier analysis phase of the MOV ATTS project. Presently the declarative model is a linear representation of basic learning objectives. Extensions could be made in future applications of this approach to arrange them hierarchically, based on formal training needs analyses. Updating the higher order objectives would be based on propagating changes made to lower order objectives upward through the hierarchy, yielding a general approximation of the student's knowledge for the prescribed learning objectives.

Action History Student Model
The second student model attempts to create a history of actions and errors made by the student during troubleshooting exercises. Example actions that are logged to this action history model are troubleshooting tests performed, display screens seen, number and types of errors made, and time to solve a problem or perform an action. In short, all actions during a troubleshooting exercises are logged to the action history model. This action history is used by the expert and instructor models to provide advice as deemed appropriate. At the start of each new troubleshooting problem, the action history model from the previous problem is deleted and a new one is created.

The expert model uses the information contained in the action history model during the advice generation process. When the student solicits expert advice, the expert model will compare the suggested diagnostic procedure to the student's action history. When a discrepancy is found between what the student has done, and what the student should do, then the expert model uses this information to generate advice (see Expert Advice).

The instructor model uses the action history model to determine if the student is using the full power of the training system, or if the student is straying too far from common sense types of troubleshooting (e.g., making numerous redundant actions). More information on how this model is used in the Instructional Advice section.

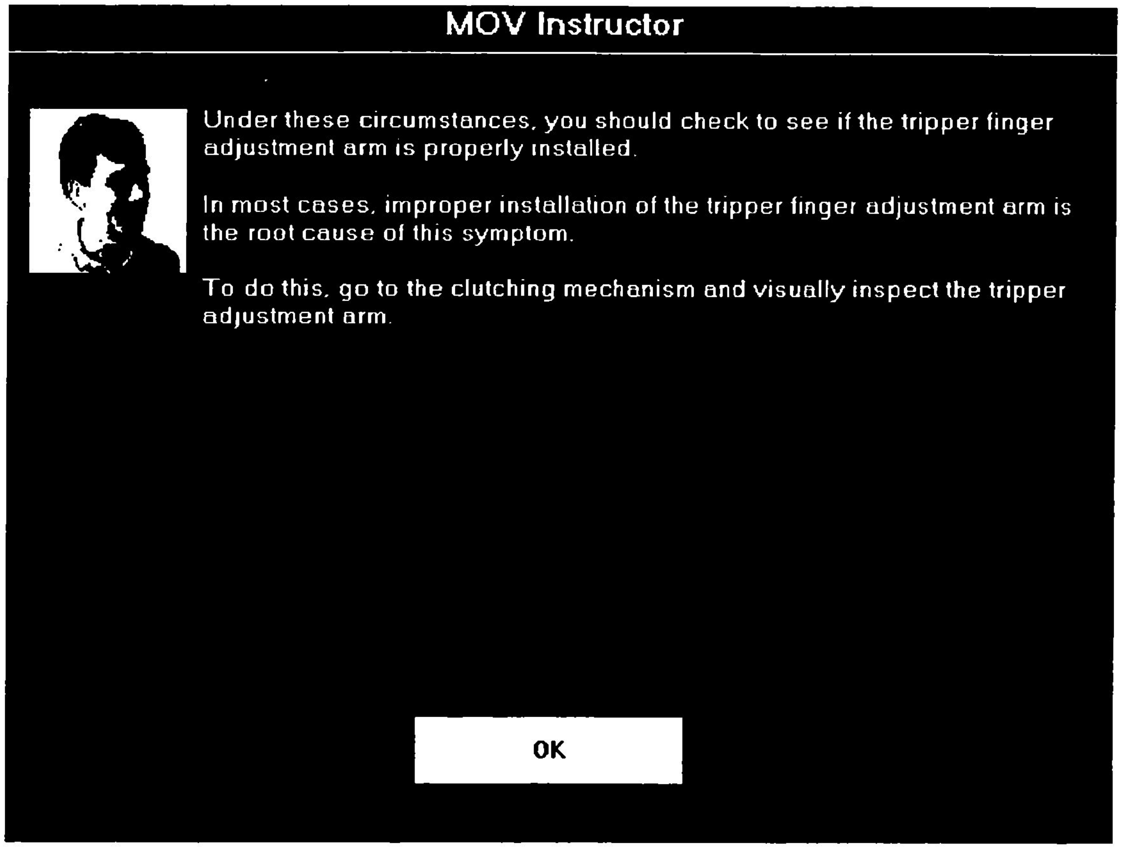
This action history list is relatively simple when compared to the declarative information model, but yields sufficient information to support both the expert and instructional advice systems. The use of this type of student model dates back to 1985 for use in the diesel generator simulation project discussed earlier, and has evolved over the years through the Microcomputer Intelligence for Technical Training (MITT) projects performed for the National Aeronautics and Space Administration and the U.S. Air Force (Norton, et al., 1991).

EXPERT ADVICE

The expert model in an ITS is used to capture the expertise and knowledge about a domain that is to be transferred to the student. The information is captured and represented in such a way, though, as to support subsequent computation upon the information. This is in contrast to the way that conventional computer-based training systems capture, store, and process knowledge. Often these systems simply collect a set of facts, assign this material to a particular screen, and then present this static information to the student without any further processing.

The MOV ATTS contains two separate representations of expert knowledge: a rule-based procedural expert model, and a functional model of the system. Both of these models are used during the troubleshooting training exercises.

The Procedural Expert Model
The first expert model is a procedural model of how an effective troubleshooter would diagnose and repair an MOV actuator. This model is used to determine the correct diagnosis procedure for a given problem. As input, the procedural expert model reviews the current state of the simulated system, actions taken by the student so far, and specific real-world troubleshooting procedures. The procedural expert model then processes this information to generate the next step the student should take in the troubleshooting procedure (see Figure 3).



MOV Instructor

Under these circumstances, you should check to see if the tripper finger adjustment arm is properly installed.

In most cases, improper installation of the tripper finger adjustment arm is the root cause of this symptom.

To do this, go to the clutching mechanism and visually inspect the tripper adjustment arm.

Figure 3. Sample Procedural Advice.

The real-world troubleshooting procedures used in the MOV ATTS were derived from informal interviews with MOV actuator subject-matter experts. The procedures were also based upon written rules and procedures in nuclear utility training manuals and the manufacturer's manuals.

The procedural expert model is put into action when a student requests advice during a troubleshooting exercise. The procedural expert first initializes itself with facts that it can gather about the system (e.g., facts about the state of components, the valve, indicator lights, etc., as well as information about what the student has done to this point). The procedural expert then scans its set of procedures to find the one that matches the initial conditions of the valve.

The procedural expert will then scan the set of actions and tests of the retrieved procedure, looking for an action that is appropriate at this stage in the troubleshooting process. This choice of an appropriate action is based on the results of various tests performed by the student and conditions of the simulated MOV actuator. Once an action is found, it is checked against the student's action history model to see if this action has already been performed. If it has not, it is presented to the student as the suggested next action to take. If the action has already been performed, the procedural expert will find the next appropriate action.

For example, an action that was suggested by the MOV instructors at Duke Power company was to always check the handwheel and declutch lever first when there is a problem switching the actuator from manual to remote. If there is still a problem even when these components are operating normally, then the problem is likely to involve either the tripper adjustment arm or the declutch fork. In the procedural expert, this knowledge is represented as:

```
if      problem-symptom is inability to switch from manual to remote and
        handwheel is feasible and
        clutching-subsystem is feasible
```

then

set advice to "Hold down the declutch lever and see if manual operation is possible in this position. To do this, click on the button labeled "Declutch Lever."

set explanation to "If manual operation is not possible even when you hold down the declutch lever then the problem is probably with the tripper adjustment arm or the declutch fork."

The procedural expert for the MOV ATTS contains approximately 75 of these procedural rules to cover the troubleshooting problems in the tutor. There is some economy of scale with the number of rules, i.e., as the number of problems increase, number of rules to be added for a new problem tend to decrease due to the existence of applicable rules from other procedures.

Functional Expert Model

The second expert model that is contained in the MOV ATTS is a functional model of the MOV actuator. This model is based upon individual parts in the system, and the influences that these parts have upon one another. This model evolved from previous experimental evaluations for support of diagnostic learning (Johnson, 1981; Rouse and Hunt, 1984) and have formed a core technology in many of our training systems since then.

Building an accurate functional expert model of a technical system requires detailed knowledge of the system, as well as a sharp understanding of the dependencies within this system. Even though our functional model of the MOV actuator (see Figure 4) only contains approximately 50 parts, it was a time consuming task to develop and refine this model and involved the participation of three Duke Power personnel and two Galaxy Scientific personnel.

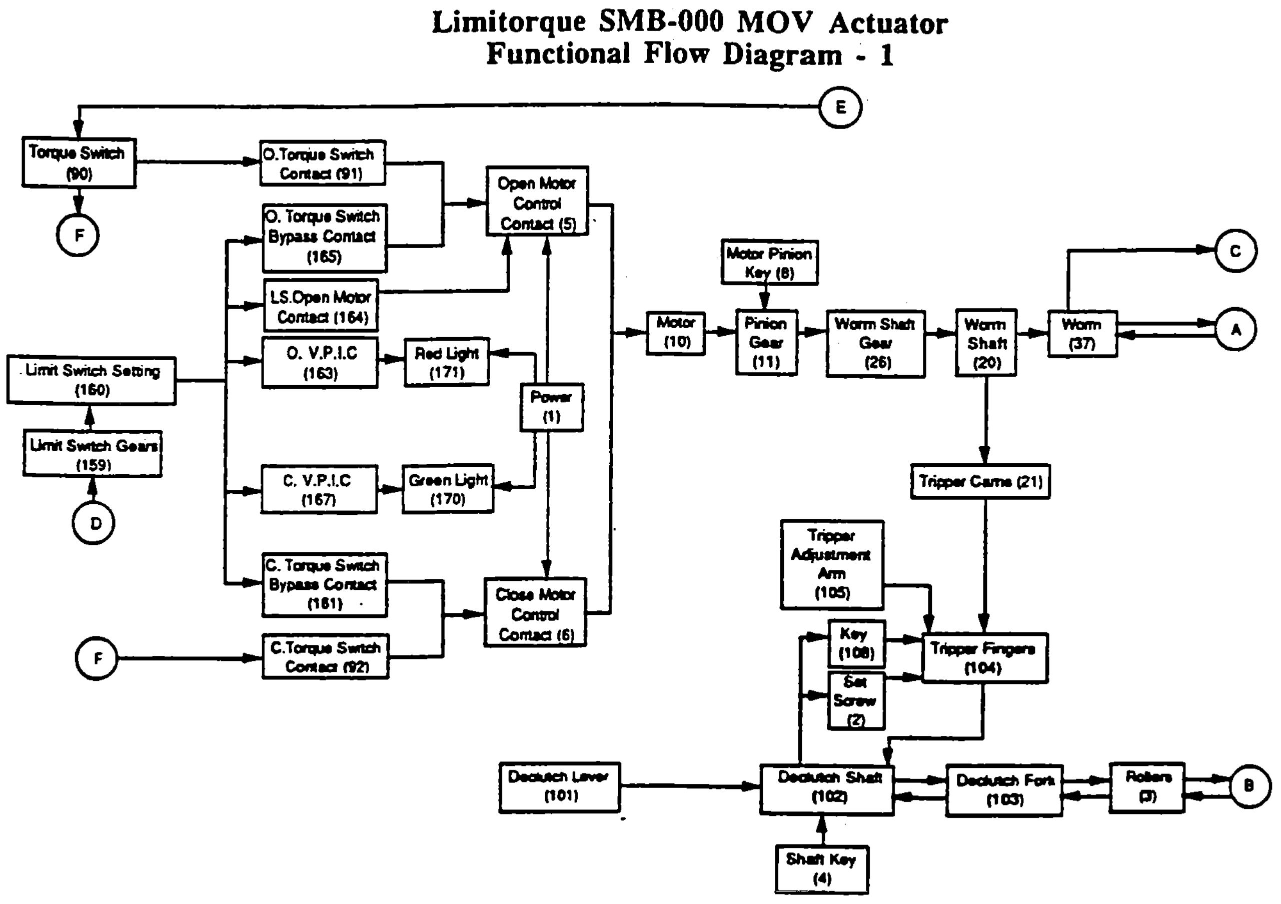


Figure 4. Sample section from MOV Functional Flow Diagram.

The functional expert model works on the concept of *functional dependency*. Functional dependency states that for a component in a system to function properly, all components that either directly or indirectly influence that part must also be functioning properly. For example, in Figure 4, in order for the Worm Shaft Gear (26) to function properly, this part and all parts that influence it must be functioning properly. This include the Pinion Gear (11), Motor Pinion Key, Motor (10), Open Motor Contact, Torque Switch (90), etc.

The MOV ATTS uses this functional expert model of the system in three ways. First, the functional expert model of the system is used to verify the procedural advice generated by the procedural expert model. Before the procedural expert presents advice to the student, it first checks the functional model to see if this piece of advice is logical. Note that the procedural expert is based upon human rules of thumb and advice, and is subject to contain some misleading advice or to overlook items that occur rarely in the real world. The functional model of the system, on the other hand, is completely logical in its representation of the system. By checking the procedural advice against this functional representation, we can avoid giving advice to the student based upon faulty expert logic (see Figure 5).

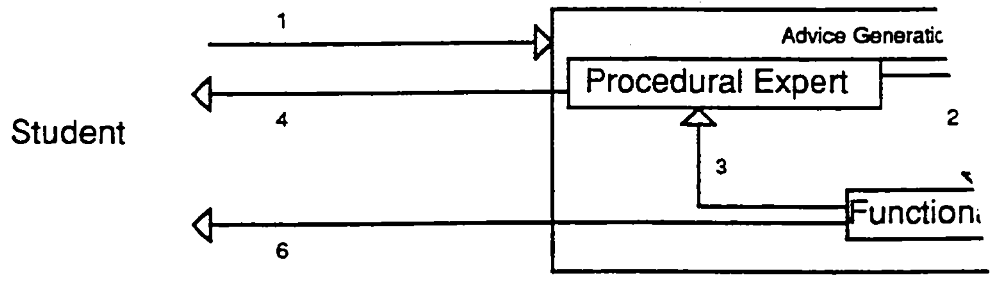


Figure 5. Advice generation by integrating functional and procedural models. *1) Student requests advice. 2) Procedural expert takes first attempt to generates advice. This is passed along to functional expert. 3) Functional expert either confirms or rejects advice. 4) If advice is confirmed, procedural expert formats advice and passes along to student. 5) If not confirmed, procedural advisor may generate more advice (via path 2) or may exhaust all procedural possibilities. When no more procedural advice is found, advice generation responsibilities are surrendered to functional expert via path 5. 6) Functional expert formats functional advice and passes along to student.*

One of the benefits of integrating both models is the ability to capture real-world shortcuts and procedures to teach the student (via the expert based procedural model), while maintaining the validity of these shortcuts (via the functional model of the system) before they become misconceptions. Even though there have been a few other ITSs that use dual expert models, the MOV ATTS represents a new step toward the integration of multiple experts in an ITS. In addition, the self-validation process provides a check-and-balance feature capable of aiding developers in the validation and debugging of rule-based expert models.

The second use of the functional expert model by the MOV ATTS is to provide a backup advice generator when the procedural expert cannot produce any further advice for a troubleshooting situation. The procedural advisor will only generate advice when it meets situations that it is programmed for. An example situation is when the procedural advisor has led the student to a point in which there are just a handful of remaining parts, and there is no predetermined procedures to further eliminate additional parts. The functional advisor is capable of taking over at this point and generating advice for the student.

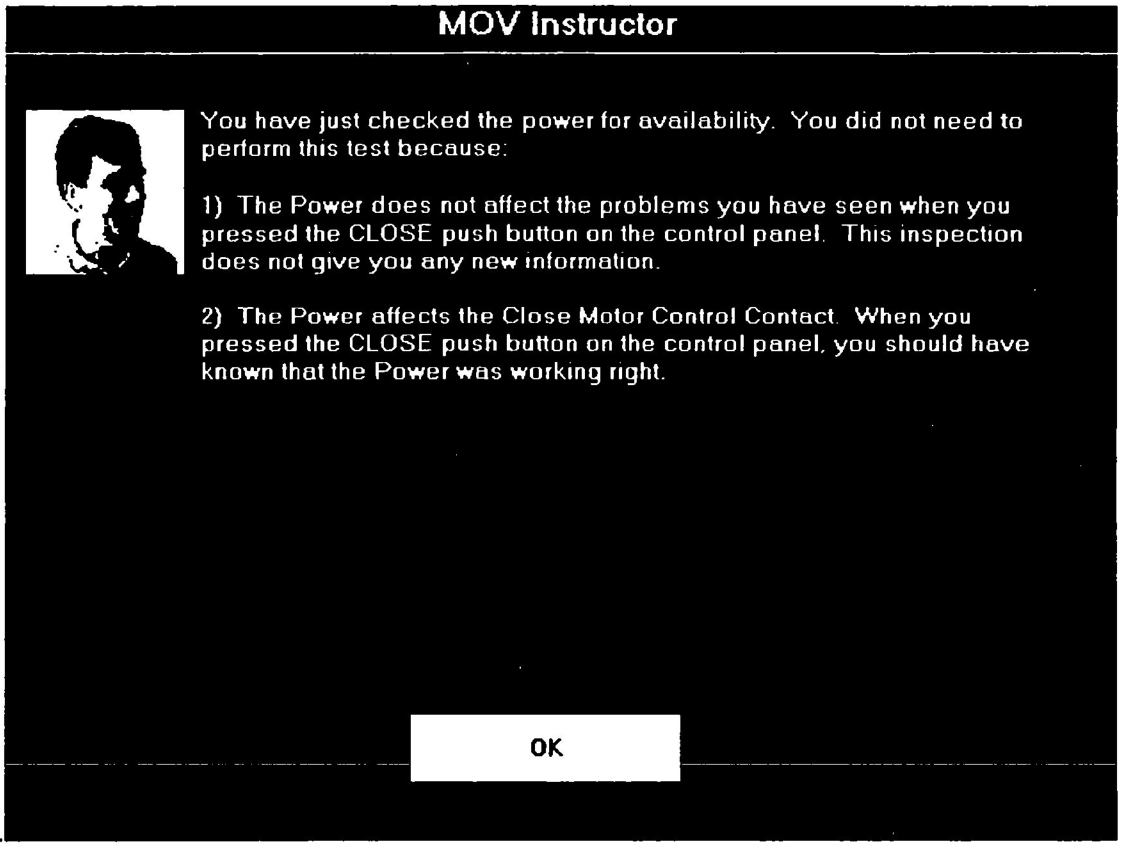
The functional advisor can generate advice at this point based upon the functional dependency of parts that have not been eliminated as a root cause of the problem. Generating this advice is accomplished basically by "half-splitting" the remaining parts, and determining the test or action that will eliminate the most parts from the feasible set. For example, if you had 4 parts in a system connected serially, the most logical test is to test the connection parts 2 and 3. If the results of the test are negative, you know that either part 1 or 2 has failed. If the results of the test are positive, you know that either part 3 or 4 is the failed part. The functional advisor is able to perform this half-splitting technique upon complex networks of parts, such as the MOV actuator representation, yielding logical advice that will produce the minimum number of tests to locate the failed component.

Note that if the system did not contain any procedural rules at all, then the functional advisor would produce advice based on this simple half-splitting technique. However, the procedural advice is important in order to capture and pass along troubleshooting techniques that are equipment and site specific, and that can offer a time and cost savings during the troubleshooting tasks.

The third manner in which the functional expert is used is to call the student's attention to simple diagnostic errors being made. This feedback manifests itself as unsolicited advice during a troubleshooting exercise (see Figure 6). As the student performs tests and actions in the troubleshooting exercise, a comparison is made between these actions and the functional expert. If a student's test is illogical according to the model, one of three types of responses are generated: 1) the test is declared incorrect because this test or a logically equivalent test was performed earlier, 2) the test is declared incorrect because the tested component functionally influences a component or subsystem that is known to be functioning properly from previous tests, 3) the test is declared incorrect because the tested component does not functionally influence a component or subsystem that is known to be functioning improperly.

Instructional Guidance

Instructional guidance is used in the MOV ATTS to select appropriate lessons for the student, and to guide the student through the troubleshooting exercises. The first use of the instructor model determines the amount of the basic MOV, mechanical, and electrical material that is to be presented to the student. Most of the work for this use is done by the declarative information student model, who has calculated an estimate of the student's understanding of the individual learning objectives.



You have just checked the power for availability. You did not need to perform this test because:

1) The Power does not affect the problems you have seen when you pressed the CLOSE push button on the control panel. This inspection does not give you any new information.

2) The Power affects the Close Motor Control Contact. When you pressed the CLOSE push button on the control panel, you should have known that the Power was working right.

on.

Figure 6. Unsolicited advice from the functional expert.

The instructor model at this point simply looks for learning objectives that exceed a predetermined threshold (in the case of the MOV ATTS, this threshold is 0.80) and exempts students from any modules related to these learning objectives. Note that the students are given the option to review the exempted modules if they so desire.

As mentioned earlier, the instructor model can also review the declarative information model and suggest to the student to review particular modules in which the student has demonstrated weak understanding. This is done at the end of each major module and before the student is allowed to practice troubleshooting. Note that the student, if he or she desires, can ignore the advice of the instructor and continue

The second use of the instructional guidance comes into effect during the troubleshooting exercises. Within this module, the function of instructional guidance is to call the student's attention to simple diagnostic errors being made. This guidance manifests itself as unsolicited advice during a troubleshooting exercise. The instructor model, using the action history student model, records the number of times advice is requested, the particular section of the MOV actuator being diagnosed, the number of redundant or unnecessary actions (as determined by the functional expert), and the number of diagnostic procedural errors (as determined by the procedural expert). The instructor model periodically checks these counts during a troubleshooting exercise and if they exceed a predetermined amount (e.g., if the student makes 3 unnecessary actions), the instructor model will offer unsolicited feedback to the student based upon these rules. The benefit of this approach is to direct the student's attention to mistakes at the time they are made, rather than waiting to the end of the troubleshooting exercise to recap and summarize these points.

Summary

The primary goal of the MOV Advanced Technology Training System (ATTS) is to reduce human errors that have occurred during MOV overhaul and repair. The MOV ATTS addresses this goal by providing basic operational information of the MOV, by simulating the MOV operation, and by providing troubleshooting practice on the diagnosis of MOV failures, and by tailoring this training to the needs of each individual student.

An pilot effectiveness study of the MOV ATTS was conducted at the Maintenance Training Facility of the Tennessee Valley Authority's Bellefonte Nuclear Station. Six mechanical and six electrical maintenance personnel participated in the study which was conducted at the site during November 3 - 6, 1992. All twelve subjects were first trained on the declarative knowledge portion of the MOV ATTS. The control group was then trained on troubleshooting techniques on actual MOV equipment, while the experimental group received troubleshooting training via the MOV ATTS troubleshooting tutor. Both groups were then tested on their troubleshooting skills using a modified version of the MOV ATTS troubleshooting simulation.

Results of the effectiveness study show the trend of the experimental group (i.e., those trained with the MOV ATTS Troubleshooting module) inspected more components before attempting to correct the failure than their counterparts who were trained on the real equipment. However, a significant difference was found in the correctness of their answers. The control group made a significantly higher number of premature answer attempts than those trained with the MOV ATTS. In addition, the control group made more errors than the MOV ATTS group. The results of this evaluation suggest that the subjects trained with the MOV ATTS Troubleshooting module collected more information about a particular failure before moving ahead and correcting the problem. The control group tended to collect insufficient data before attempting to correct the perceived problem, leading to more incidence of error. Due to the small sample population in the effectiveness study, care should be taken in applying these trends to other situations.

The current phase of the ATTS project is producing an *authoring tool* to help utility instructors construct troubleshooting training systems similar to the Troubleshooting module of the MOV ATTS. This tool is the first step toward providing utilities with the means to produce ATTSs for other troubleshooting training tasks.

References

Anacapa Report from Phase I EPRI/CRIEPI

Burns, H.L. and Capps, C.G. (1988). *Foundations of Intelligent Tutoring Systems: An Introduction.* In Foundations of Intelligent Tutoring Systems. Hillsdale, NJ: Lawrence Earlbaum Associates, Inc.

Johnson, W.B. (1981). Computer simulations for fault diagnosis training: An empirical study of learning from simulation to live system performance (doctoral Dissertation, University of Illinois, 1980), (*Dissertation Abstracts International, 4111*). 4652-A. (University Microfilms No. 8108555).

Johnson, W.B., Wiederholt, B.J., and Maddox, M.E. (1986). *Diagnostic training demonstration: Instructor and student manuals and sample diskettes* (EPRI NP-4493P). Palo Alto, CA: Electric Power Research Institute.

Maddox, M.E., Johnson, W.B., and Frey, P.R. (1986). *Diagnostic training for nuclear power plants personnel, volume 2: Implementation and evaluation* (EPRI NP-3829-II). Palo Alto, CA: Electric Power Research Institute.

Massey, L.D., Psotka, J., and Mutter, S.A. (Eds.) (1988). *Intelligent Tutoring Systems: Lessons Learned.* Hillsdale, NJ: Lawrence Earlbaum Associates, Inc.

Norton, J.E., Wiederholt, B.J., & Johnson, W.B. (1991). *Microcomputer Intelligence for Technical Training (MITT): The evolution of an Intelligent Tutoring System,* in Proceedings of the Contributed Sessions of the 1991 Conference on Intelligent Computer-Aided Training. Houston, TX: Software Technology Branch of the Johnson Space Center.

Polsen, M.C. and Richardson, J.J. (Eds.) (1988). *Foundations of Intelligent Tutoring Systems.* Hillsdale, NJ: Lawrence Earlbaum Associates, Inc.

Rouse, W.B. and Hunt, R.M. (1984). Human problem solving in fault diagnosis tasks. In W.B. Rouse (Ed.), *Advances in Man-Machine Systems Research: Vol. 1.* Greenwich, CT: JAI Press, 195-222.

Widjaja, T.,K. & Wiederholt, B.J. (1992). *MOV Advanced Technology Training System Specifications.* Atlanta: Galaxy Scientific Corporation.

The Learn, Explore and practice (LEAP) Intelligent Tutoring Systems Platform

Charles Bloom, Brigham Bell, Frank Linton & Edwin Norton
U S WEST Advanced Technologies
4001 Discovery Drive, Suite 270
Boulder, CO 80303
cbloom@advtech.uswest.com

Abstract

This paper describes the prototype LEAP (Learn Explore and Practice) intelligent tutoring systems (ITS) platform being developed at U S WEST Advanced Technologies (AT) to train U S WEST customer service representatives (CSRs) how to sell telecommunications services and products. . Domains of this type require the CSR to simultaneously carry on a conversation with the caller, manipulate database applications to find out information about the caller and enter information regarding service registration, and look up information about availability and service capabilities from reference documentation located on their desks. These services and products that the CSRs sell (and the information about them that the CSR must be facile with) are updated frequently, producing an ongoing learning problem in order to "stay current." In addition, telecommunications companies are highly regulated, and CSRs must be knowledgeable about and comply with all federal and state regulations under penalty of law. LEAP employs two interrelated, instructional modes: Procedural instruction, where students can exercise their customer interaction skills, and declarative instruction, where students can review services information. In procedural training mode, students work through typical customer interactions in a tutoring environment that simulates their actual working environment. Their goal is to handle some customer request regarding VMS, or transition a non-VMS call to a VMS sales opportunity. In declarative training mode, students can study multimedia lessons (e.g., animations, video segments, etc.) on information prerequisite to specific types of customer interaction skills. In addition, students can navigate between modes during instruction based on the topic currently being studied or practiced. Instruction in LEAP is student initiated. However, LEAP does assess student performance and uses that assessment to: Make recommendations about what to study or practice next, determine how to apply its different instructional methods, initiate interventions during procedural training sessions, and provide student performance feedback at the end of a procedural session. In addition, LEAP allows instructional designers to adjust LEAP's instructional and student modeling parameters to further individualize the delivered instruction. The first application of the LEAP ITS Platform is teaching CSRs how to operate and sell residential Voice Messaging Service (VMS).

INTRODUCTION

Intelligent tutoring systems are dynamically organized "intelligent" instructional programs that employ independent representations of domain, instructional and student knowledge to enable them to provide individualized instruction much like that provided by a personal human tutor. The goal of an Intelligent Tutoring System (ITS) should be to provide real-time, context-appropriate and cost-effective training that enables learners to perform appropriate domain tasks in the right manner and at the proper time. An ITS that is able to achieve this goal should produce a decrease in the time required to migrate learners from novice to expert, while increasing the number of trained personnel successfully reaching a more knowledgeable level. Unlike conventional computer-based training (CBT) programs that deliver instruction statically and uniformly, an ITS can be:

- Proactive — actively teach difficult and abstract concepts and skills
- Reactive — guide students in exploratory learning in a simulated environment or microworld
- Adaptive — tailor training scenarios to the student's learning progress

An ITS accomplishes this by employing a basic architecture that consists of the four interacting components depicted in Figure 1:

- Domain Model — a representation of the knowledge to be tutored to the student, also used as the standard for evaluating the student's performance (Anderson, 1988; Wenger, 1987).
- Instructional Model — a representation of the knowledge of how to tutor the student (i.e., the instructional methods to be employed in the tutor and how they are employed) (Halff, 1988).
- Student Model — a dynamic representation of the student's knowledge state (VanLehn, 1988).
- User Interface — the communication channel between tutor and student (Burns & Capps, 1988).

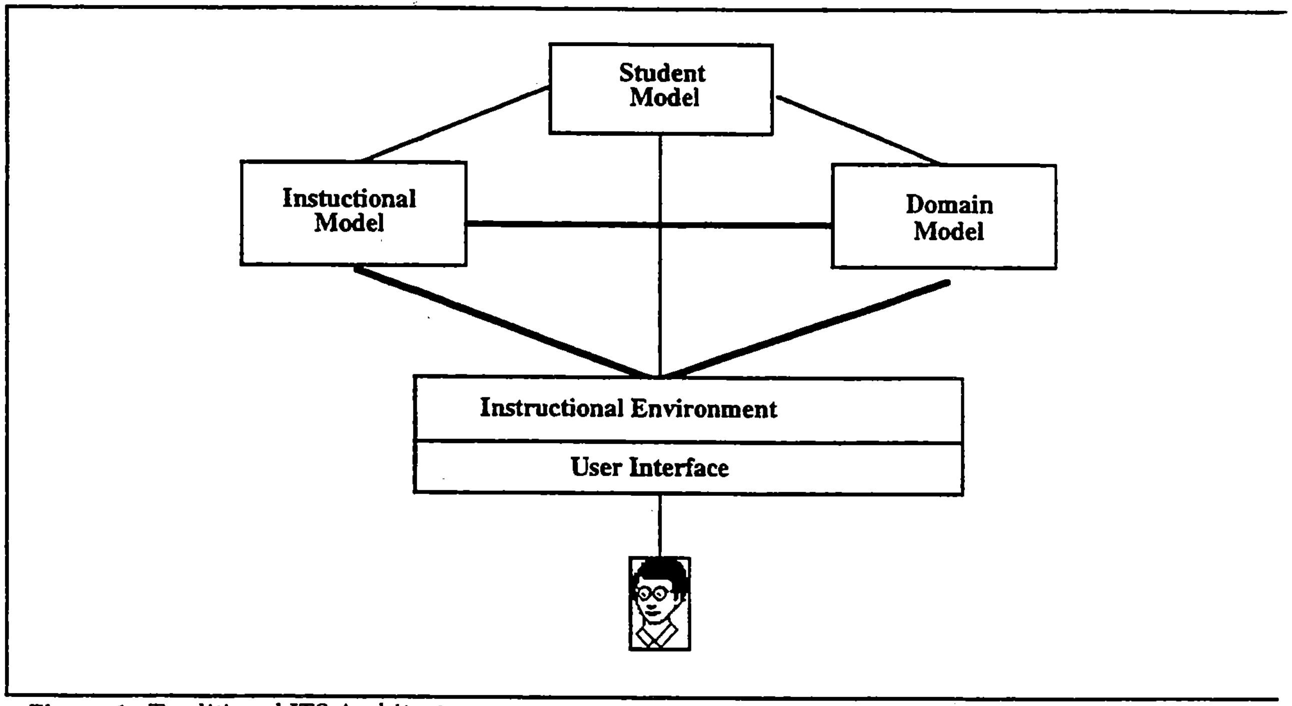


Figure 1. Traditional ITS Architecture

The domain selected for the first application of the LEAP ITS Platform is teaching U S WEST Home and Personal Services (H&PS) customer service representatives (CSRs) how to operate, sell and register customers for residential Voice Messaging Service (VMS). This domain was selected because it is representative of a broader class of domains that includes the sales and service of US WEST products across business units. This choice of domains makes it possible to identify portions of the ITS that are reusable both for extending its scope to cover additional business units, and for developing LEAP-based ITSs for other products and services across business units.

VMS is an Enhanced Service Product offered by U S WEST that answers incoming calls placed to the subscriber when the called number is busy or if the called number does not answer. (Enhanced Service Products are products that are optional, competitive, and not regulated.) VMS allows subscribers to record their own personal greeting or else use a prerecorded greeting. When messages are waiting, the subscriber is alerted by a special "stutter" dial tone. Messages can be retrieved either from home or away from home by calling a special access number using a push-button phone and entering a personalized security code. A representation of the high-level tasks related to VMS that a CSR performs is presented in Figure 2.

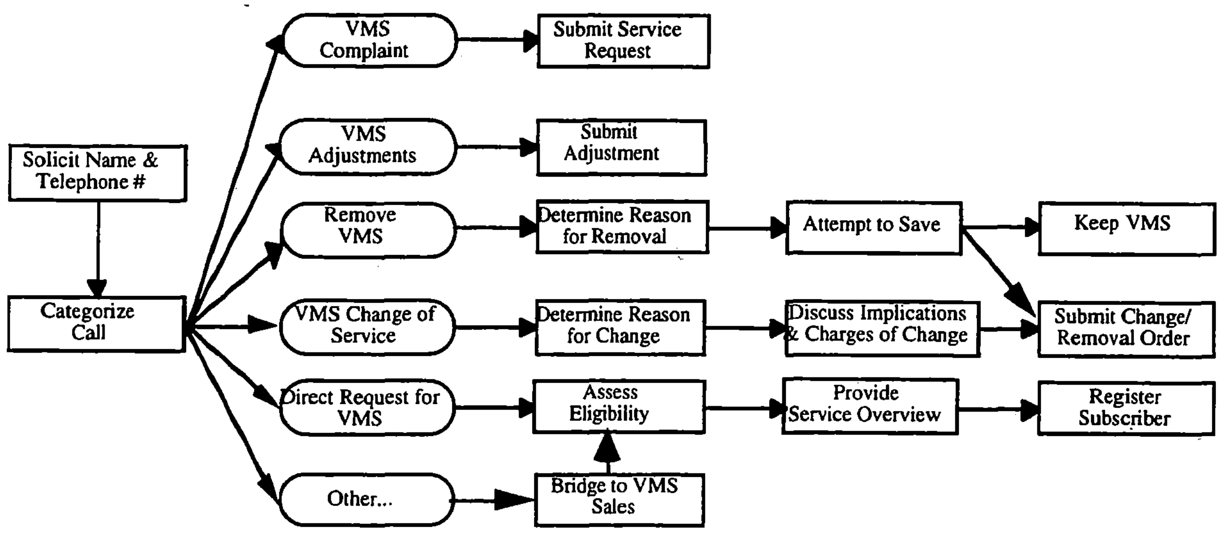


Figure 2. VMS Related Tasks

The VMS sales process begins with the CSR asking the caller's name and telephone number, which is then entered into a database (via a computer workstation) to pull up the customer's account information. At this point, the experienced service representative will engage in a short conversation with the customer to determine the nature of the call. In the VMS domain, calls can be categorized into one of three types: (1) Calls from a customer who already has VMS, (2) calls infolving a direct request for VMS, or (3) calls that have nothing to do with VMS.

There are basically two types of calls a CSR can receive from a customer who already has VMS: Calls in which the customer is complaining about VMS service or billing, or calls in which the customer is requesting some change to the service. For the first type of call, the CSR's task is to initiate the appropriate work order. For the second type of call, the CSR's task is more complicated involving:

- Determining the customer's reasons for wanting a change or removal of service
- Discussing with the customer the implications of these changes
- In the case of a request for removal of service, attempting to save the sale
- Initiating the change or removal if the customer still wants it.

In the case of calls that have nothing to do with VMS, the CSR must look for or make an opportunity to transition the conversation into a discussion of VMS (or some other product or service) capabilities and benefits. To do this, the CSR needs to have skills in the areas of:

- Evaluating customer data (either from credit information, the current conversation, or information volunteered by the customer) for VMS selling clues
- Transitioning the conversation to VMS sales
- Making selling points (i.e., explaining benefits and features)
- Countering specific customer responses.

Once the caller becomes interested in the service, or in the case where a caller is making a direct request for VMS, the CSR then assesses the compatibility with the subscribers other services and equipment, and discusses the legalities of selling VMS. Following that, the CSR might go over the specifics of VMS, and finally, go through the process of registering the subscriber for VMS.

During the process, the CSR has to simultaneously (1) carry on a conversation with the caller, (2) manipulate certain databases and applications on their computer terminal to find out information about the subscriber and enter information regarding VMS registration, (3) look up certain information about availability and service capabilities from reference documentation located on their desks, and (4) prepare mailings of VMS information for the caller.

An integral part of the CSR's job in selling and registering customers for VMS is interacting with the Service Order Negotiation and Retrieval (SONAR) application. SONAR is the primary system used by CSRs for collecting customer information and initiating orders for phone service. SONAR is a mainframe application that CSRs access over an internal U S WEST ethernet local area network (LAN) via terminal emulation software on their desktop workstations.

In an attempt to conceptualize and represent these real-world conversations in a manner that is usable in the LEAP Tutor, we developed an abstract representation of these conversations in terms of a task grammar. The basic unit of the task grammar is the situation-action (SA) pair. Situations can be either customer statements, requests or questions, or database application output or configurations. Actions can be either responses by the CSR to customer statements or to application information, commands or data entered into the application by the CSR, or actions focused around information processing and decision making. All conversations are made up of sequences of SA pairs. Conversations are grouped together to reflect different types of scenarios the could occur between caller and CSR (i.e., "New Connect," "Change of Service," etc.). Branches within conversations are based on customer situations (e.g., callers equipment or type of service, etc.). SA pairs that are conceptually related are referred to as sub-topics. Sub-topics are reusable portions of conversations that can appear in several different conversation scenarios. Figure 3 contains an example of a task grammar representation.

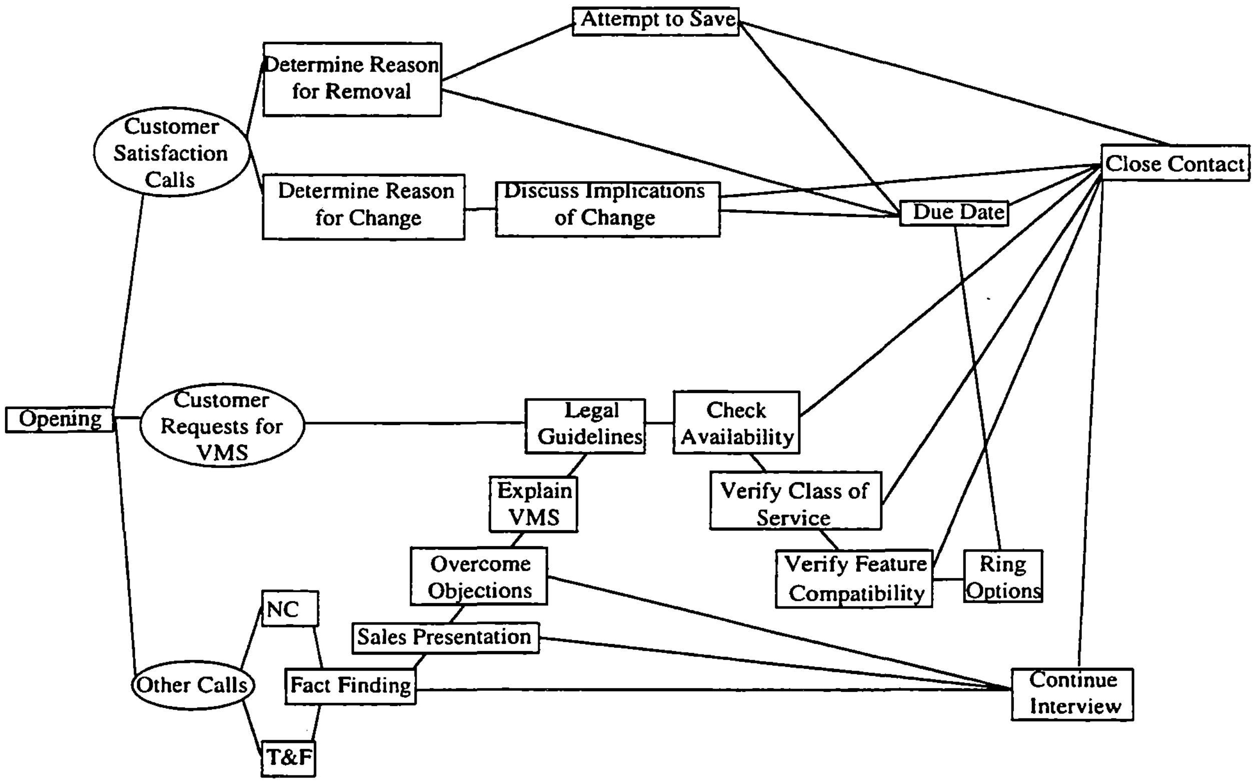


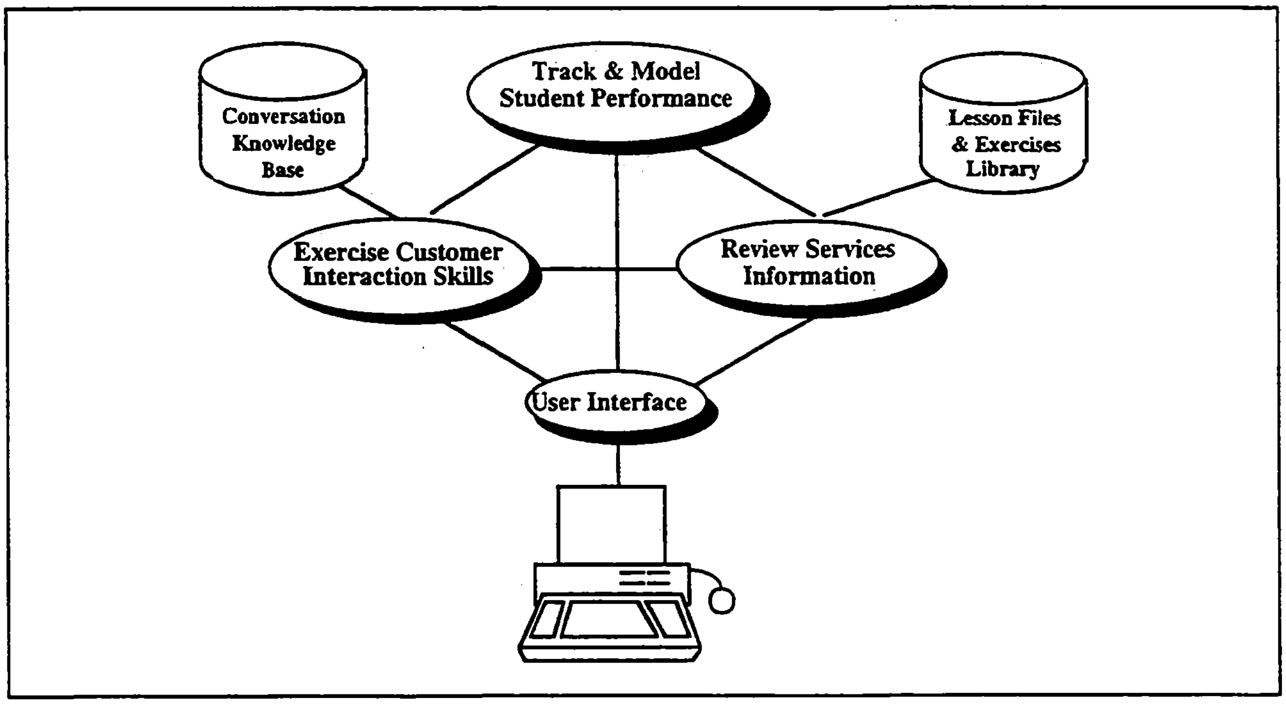
Figure 3. Example Task Grammar Space

The Task Grammar Space can be conceived of as a network representation of all of conversation sub-topics, and the interconnections between those sub-topics observed in the operational environment. In other words, the task grammar is a representation of the complete collection of possible customer-CSR conversations that LEAP will support. Each conversation scenario (i.e., VMS Change of service conversations, Direct request for VMS on new connect, etc.) is made up of several sub-topics (and their associated SA pairs) of a customer-CSR conversation. Each sub-topic contains a specification of all allowable SA pairs (including identified conversation errors).

Conversations or scenarios can be constructed by combining all of the individual sub-topics along any one branch (from left to right) of the Task Grammar Space. For example, the scenario of servicing a customer's direct request for VMS could involve some or all of the following: Opening, Legal Guidelines, Check Availability, Verify Class of Service, Verify Feature Compatibility, Ring Options, Dues Dates and Close Connection. The specific sub-topics employed would depend upon "customer" responses or constraints (i.e., if the service is not available in the customer's area, the scenario would contain only Opening, Legal Guidelines, Check Availability and Close Connection).

LEAP Tutor functional architecture

This section describes the instructional modes that the LEAP Tutor employs to teach CSR's how to perform the VMS sales and service related tasks discussed previously. Figure 4 contains a conceptual diagram of LEAP Tutor.



Conversation Knowledge Base
Track & Model Student Performance
Lesson Files & Exercises Library
Exercise Customer Interaction Skills
Review Services Information
User Interface

Figure 4. LEAP Tutor Conceptual Diagram

The LEAP Tutor employs two major instructional modes: Review Services Information (RSI) and Exercise Customer Interaction Skills (ECIS). In RSI mode the student can study interactive multimedia lessons on information prerequisite to VMS, or else perform study exercises on the topic selected. In ECIS mode, the student works through typical conversations (or else individual parts of conversations) where the goal is to handle some customer request regarding VMS, or else transition a non-VMS call to a VMS sales opportunity. Instruction in the LEAP Tutor is student initiated (i.e., the student selects both the mode, and the topic, grammar or scenario to study or practice). However, the LEAP Tutor will have the capability for tutor initiated recommendations (i.e., the student can request LEAP to recommend a topic, grammar or scenario for the student to study or practice) should the student want LEAP to take on that responsibility.

The LEAP Tutor also has the capability to assess the student's performance, in both instructional modes, and to use those assessment results to:

- Provide an introspective evaluation of the student's skill level on each topic and sub-topic (as listed in the LEAP Tutor's top-level topic list) on a five point scale ("Untried," "Needs Practice," "Almost," "Good," and "Excellent")
- Make recommendations about topics or scenarios to study or practice
- Initiate instructional interventions

- Provide student performance feedback at the end of the session.

The LEAP Tutor's top-level topic list provides the organizational structure for both RSI lessons and ECIS conversations. Therefore, the topic list provides explicit links between information in the two modes enabling the LEAP Tutor to:

- Recommend conversations to practice based on lessons just studied or exercises just practiced (and the student's performance on those exercises)
- Recommend lessons to study based on conversations just rehearsed (and the student's performance on those conversations)
- Support dynamic transitions between conversations (or parts of conversations) and appropriate lesson materials during ECIS practice (and vice versa for RSI study)

The LEAP Tutor recommends topics to study or practice based on a consideration of three factors: Topic sequence, relation to last topic studied, and user's topic-level proficiency. After recommending a topic, the LEAP Tutor then recommends a method of studying the topic: Either "Review Services Information," or else one of the "Exercise Customer Interaction Skills" sub-methods. The algorithm the LEAP Tutor uses for selecting a study method to recommend is as follows: First, see a demonstration of the topic in use in ECIS mode. Next, study the topic in RSI mode. Then, practice applying that knowledge in simulated conversations in ECIS mode (the types of simulated conversations will discussed in the next section). Finally, take a test on that topic (i.e., perform a conversation without the tutor's assistance). In addition, students can elect to perform drill and practice exercises on the declarative materials, or examine the flow of particular customer/CSR contacts.

The instructional objectives of the LEAP Tutor are consistent with several features of the minimalist approach to training and learning (Carroll, 1990):

- Teach people what they need to learn in order to perform their jobs. In the LEAP Tutor, most training takes place in an environment that closely emulates their real world job environment, and the tasks they perform on the tutor are representative of tasks they will perform on the job.
- Allow learners to start immediately on meaningful and realistic job tasks, and allow them to work on t hose tasks in any order. In the LEAP Tutor, the student can decide what they want to do and when to do it. Although the LEAP Tutor has the capability to recommend instruction, the student has the choice of whether to go along with that recommendation.
- Keep the amount of passive instruction (i.e., reading) to a minimum. Even though there are close to twenty topics covered in the LEAP VMS Tutor, only three of those topics, those covering prerequisite information necessary to support task performance, have declarative lessons. In addition, the declarative lessons are presented in interactive, multimedia formats to increase the level of activity or involvement of the student in the lesson.
- Training materials should explicitly support the recognition and recovery from error, both to make the learning materials more robust and complete, and to train learners in error recovery skills. The LEAP Tutor contains explicit training on errors and error recovery.

In the subsections that follow, we describe each of the LEAP Tutor's major functions.

Review Services Information

The Review Services Information (RSI) mode of the LEAP Tutor supports the delivery of declarative study materials in interactive multimedia formats, and drill and practice exercises on the topics identified as prerequisite to operating, selling and registering customers for VMS.

In RSI mode, the student selects a topic from the LEAP Tutor's top level list of topics. Not all topics have multimedia lessons or exercises. However, all topics do have an introduction and an overview. In those topics that do have multimedia lessons, those lesson materials are developed using multiple media (i.e., text, audio, graphics, photographs, animations and videos) as appropriate.

The lessons have been produced to a grain size that supports entry into the entire lesson at a topic level, or entry into parts of the lesson as they apply to parts of conversations. This second capability is particularly useful in enabling students to review information from part of a lesson to support their conversation rehearsals.

For example, prior to commencing any practice sessions, a student is advised to view the three prerequisite declarative lessons. One of those lessons deals with legal issues. The sub topics of that lesson include: Enhanced Service Product Guidelines, Consumer Protection Law Requirements, Anti-Trust Requirements, FCC Requirements, and U S WEST Full Disclosure Requirements. When viewing the lesson in RSI mode, the student is able to interactively view the lesson, selecting what to review and in what order to review them. However, the tutor does recommend they review ALL lesson materials.

On the other hand, suppose a student is practicing a conversation and gets to a point where they are to discuss the legal ramifications of selling competitive enhanced services products. If that student is unsure of exactly what these legal ramifications are, they may selectively review that portion of the "Legal" lesson that deals with this topic by pressing the related information button. Once they have reviewed as much of this information as they want or need, they can return to the conversation and continue where they left off.

The Study Exercises function presents students with one of four types of drill and practice exercise on the topic selected or studied. These drill and practice exercises are integrated with the LEAP Tutor's student model to help in the selection of items and for updating the model in terms of correct or incorrect responses. The types of drill and practice exercises are as follows:

- *Checklist Flashcards* — The student is presented with a series "clues" with accompanying questions and potential answer checklists about that clue. At the end of the series the student reviews their performance as compared to the tutor's "experts."
- *Audio Flashcards* — The student is presented with a series of auditory "cues" (i.e., prerecorded customer statements) to which they record a reply. At the end of the series the student reviews their performance as compared to the tutor's "experts."
- *Application Flashcards* — The student is presented with a series of application (i.e., SONAR) screens specifically configured to contain pertinent account information. The student must respond to specific "Yes-No" questions about that account information, as well as indicated additional account characteristics interactively on the application screen.
- *Question and Answer Flashcards* — The student is presented with a series of questions to which they must type a response into a specific input field. At the end of the series the student reviews their performance as compared to the tutor's "expert."

Exercise Customer Interaction Skills

The ECIS mode of the LEAP Tutor employs two different ways of exercising customer interaction skills: Rehearsing Conversations or Examining the Contact Flow.

Rehearse Conversation

The LEAP Tutor's Rehearse Conversation mode supports students working through conversations representative of problems or tasks faced on the job. As students work through the conversations , the LEAP Tutor will provide feedback and hints when students have difficulties. At the end of each scenario, the LEAP Tutor will provide summary feedback.

Rehearse Conversations is the most complex part of the LEAP Tutor's instructional process. The Rehearse Conversation instructional environment and philosophy is similar to that employed in the SHERLOCK ITS developed at the University of Pittsburgh's Learning Research and Development Center (Lesgold, Lajoie, Bunzo, & Eggan 1992). In SHERLOCK, students interact with an accurate emulation of their real-world environment. The objective is to produce a situated learning environment in which students can acquire and exercise their knowledge and skills. However, SHERLOCK is a non-directive tutor -- it does no proactive tutoring. However, the domains for which the LEAP Tutor was developed does require some proactive tutoring.

As a model of proactive tutoring for the LEAP Tutor's Rehearse Conversation mode, we have chose to employ an apprenticeship approach to teaching the skills and knowledge necessary for competent domain performance. The apprenticeship approach involves using methods like observation, coaching and successive approximation rather than didactic teaching (Collins, Brown & Newman, 1989). In addition, apprenticeship approaches embed the learning of skills and knowledge in their social and functional context.

There are four distinct study methods employed in the Rehearse Conversation mode: Demonstrate, Critique, Accompany and Practice. However, since Practice is the most complex method, with many of its features employed in the other methods, we will discuss Practice first.

Practice. The Practice method is the primary learning method employed in the LEAP Tutor. In practice, the LEAP Tutor proceeds through an entire conversation, with the tutor playing the role of customer, and the student playing the role of CSR. Conversations can be characterized by sets of situation-action pairs.

- Situations:
  - Customer statements, requests or questions (Verbal Situations)
  - Database Application (SONAR) output or configurations (Operational Situations)
- Actions:
  - Responses by the CSR to customer statements, requests of questions, or to SONAR information (Verbal Actions)
  - Commands or data entered into SONAR by the CSR (Operational Actions)
  - Actions focused around information gathering, information processing and decision making (Cognitive Actions)

Within a Practice session, the LEAP Tutor employs four specific study methods as defined by Collins et al. (1989) to facilitate learning: Skimming, Scaffolding, Fading, and Feedback (there is also a feedback method employed at the completion of a conversation).

Skimming takes place when the LEAP Tutor has determined that the student already knows a specific situation-action pair. To skim means that the tutor presents both the situation and its appropriate action, so that the student need only click on the action to continue the conversation. The application of skimming is probabilistic: When the situation-action pair is new to the student, he or she will always have to perform it. When the situation-action pair is known to the student, most of the time it will be skimmed. However, occasionally, the student will be asked to respond. This element of unpredictability contributes to maintaining the student's attention to the task as a whole.

Scaffolding takes place when some material is unknown to the student and not part of the current focus of instruction. In scaffolding (as in skimming and demonstrate), the tutor presents both the situation and its appropriate action and the student needs only click on the action to continue the conversation. Scaffolding also affords the student the opportunity to observe the tutor perform activities beyond their current capabilities and begin forming a model of that part of the task.

Skimming and scaffolding make it possible to support a situated learning model: Students can practice specific skills or parts of conversations in the context of entire conversations. In addition, the tutor does not waste time making the student either re-do tasks that are already well known, or else have difficulty with tasks that they are not yet capable of performing.

The third within practice study method employed in the LEAP Tutor is Fading. Fading consists of the gradual removal of "supports" until the students are on their own (Collins et al., 1989). In the LEAP Tutor, fading is used to support learning of situation-action pairs where the situation has no explicit features. For example, before selling VMS to a customer, CSRs are expected to remember to tell the customer that this is a competitive service available from other providers. There is no explicit situation to stimulate this action - the CSR must remember to make this "competitive service disclosure" on their own (a legal requirement). To support learning of responses to implicit situations, the LEAP Tutor will initially present them an explicit situation, which it gradually "fades," ultimately requiring the student to perform this action from memory as they would have to in actual conversations.

The fourth within practice study method employed in the LEAP Tutor is feedback. As the student works their way through a conversation, if their action is correct, the conversation continues. If, on the other hand, their action is incorrect, the LEAP Tutor provides them with a hint and allows the user a second try before supplying the student with the correct action and continuing the conversation.

As an example of a practice session, conversations begin with LEAP initiating a customer call which the student hears over a headset similar to the kind used on the job. The student then responds, prompting another customer statement and so on until the conversation is complete. Verbal responses are input as follows: First the student responds verbally (the LEAP Tutor records their responses for playback/feedback purposes). After indicating to LEAP that their verbal response is complete, the student is then presented with a menu of pre-defined responses and instructed to select the one that is closest to their verbal response. These responses are abstract representations of possible CSR responses (e.g., "Ask caller's name").

When the response selected by the student is incorrect, the LEAP Tutor provides them with an immediate intervention (a hint) and then allows them to attempt to select the correct response. If their second response is also incorrect, the tutor supplies them with the correct response and then continues with the practice session.
At times, instead of a verbal response, the student is expected to take some action in the database application (SONAR). This they do directly, with the LEAP Tutor monitoring, evaluating and responding to their actions. When the student's action is correct, the application monitoring software passes their input to the host application and continues the practice. When the student's action is incorrect, the application monitoring software notifies the student that their answer was incorrect, provides them with a hint, and recommends they try again. After a second incorrect action, the application monitoring software provides by the student and the host application with the correct action and continues the conversation.

There are four important points to note about practicing conversations in the LEAP Tutor:

- Customer statements are heard rather than read (emulates the "job" environment)
- CSRs verbal responses are recorded for use in end-of-conversation feedback/reviews
- Menu selection is employed as a work around our inability to parse spoken natural language
- The application students interact with in LEAP training IS the same application they will use on the job.

Demonstrate. The Demonstrate method is similar to Collins et al.'s "Modeling" method. The tutor shows the student how a task is performed, demonstrating both sides (customer and CSR) of a VMS-related contact. This includes all verbal statements made by either the customer or the CSR, the cognitive actions (strategic decisions) made by the CSR, and all database application actions (SONAR). The student observes the demonstration and builds a conceptual model of the task, in this case, a telephone contact with application use.

When a student observes a conversation using the Demonstrate method, they receive "credit" with the student model even though they did not do any of the conversation (or application) interactions.

Critique. The Critique method (still under development) is envisioned to be similar to Collins et al.'s "Articulate" method. The goal is to get the student to "use" their knowledge by articulating reasons for specific actions or decisions. The tutor presents a conversation (as in Demonstrate), and the student is assigned the task of monitoring that conversation for specific activities (thereby articulating their knowledge). These activities could be errors made during the conversation, or specific strategies used, or even specific actions taken.

During the Critique exercise, if the student correctly notes the occurrence of some item, the tutor provides them with positive feedback and continues with the exercise. If the student fails to note the occurrence of some item, or if the student notes an incorrect item, the tutor provides them with an appropriate hint and lets them try and identify the correct item. If the student gets the item correct on this second try, the receive positive feedback and the exercise continues. If the student fails to note the correct item on this second try, the tutor provides them with the correct answer before continuing the exercise. It is important to note that the student receives feedback on the critique items, not conversation items in Critique exercises.

At the end of a critique exercise, the student goes through a feedback session in which the conversation and their observations are played back for them, with errors and correct responses highlighted.
Critique exercises are designed to focus on the sub-topic (and in some cases, the topic) level. Therefore, the assessments provided to the student model reflect those levels. In addition, since students "observe" a conversation, they also receive credit with the student model for having seen the conversation and application situation-action pairs employed in that exercise.

Accompany. The accompany method is based on the principles of part-task training and reduction of cognitive load (Sweller, 1988). In accompany mode, the tutor presents both sides (customer and CSR) of the conversation, while the student is expected to perform the database application actions. Thus, the student "accompanies" the conversation on SONAR.

In an Accompany exercise, the student's receive "credit" with the student model for having seen the exercise's conversation situation-action pairs, and receive student model assessments for their performance on the exercise's application situation-action pairs.

Session Feedback. At the end of each rehearse conversation session, the LEAP Tutor evaluates the student's performance and prepares and presents feedback. The LEAP Tutor provides for several different types of feedback, selectable by the student. The types of feedback available are patterned after those implemented in SHERLOCK (A. Lesgold, personal communication, December 1992), and will include:

- *Summarize progress* — The student is presented with a graphical representation of the student model indicating LEAP's assessment of the student's status with regard to the various topics.
- *Session Playback* — The student is presented with a step-by-step walk-through of their rehearse conversation session, with expert responses available upon request.
- *Selective Review (Summary and Playback)* — The student can select a portion of the previous rehearse conversation session to review (this includes both a summary assessment and a step-by-step playback with expert responses available). The selective review can be either by conversation component (i.e., legal issues, verify feature compatibility, etc.) or by student response type (i.e., verbal, operations or implicit).
- *Global Facet Review* — The student is presented with a checklist in which they will be asked to rate their performance on a number of areas (e.g., conversational tone, conciseness and clarity of explanations, sales effectiveness, effectiveness of handling customer problem, etc.). This will prompt the student to keep these factors in consideration during subsequent rehearsals, and could also be used during instructor reviews.

Following completion of the rehearse conversation feedback, the student returns to the LEAP Tutor's top level screen set where the student's proficiency levels for the course topics have been updated and presented. From this point, the student proceeds as before, either selecting the next topic and method, or else asking the tutor for a recommendation.

Test. Test mode is the LEAP Tutor's means for measuring the student's knowledge and skill levels under as realistic conditions as possible. In Test mode, the student must perform entire conversations -- there is no skimming or scaffolding, all implicit situation cues are completely faded, access to the multimedia lesson materials is blocked, and no coaching hints are provided by the tutor.

Students can reach test model by one of two methods. First, when the student reaches some criterial level of performance in practice sessions, the LEAP Tutor recommends they take a "test." Second, students can access test mode at any time as a means of self-evaluating their readiness.

Examine Contact Flow

In Examine Contract Flow mode, the LEAP Tutor supports students' explorations of the underlying conversation structure (i.e., the task grammar network) by allowing them to navigate the network, select which branch of the network to take, and back up and try different alternatives. Student's can examine the flow of customer/CSR contacts in either of two ways: Explore or Browse. Browse enables students to see the tutor's response to conversational situation they (the student) select. Explore enables students to not only select conversational situations, but also select the response to those situations and have the tutor evaluate their selection.

In a Browse session, the students receive "credit" with the student model for those situation-action pairs they examine. In an Explore session, the students receive student model assessments for the situation-action pairs they interact with.

User Interface

The user interface is the means by which individuals interact with a system. Further, it is the user interface, more than any other function, that serves to define for the user what that system does, and how well it does it. This is particularly important in an intelligent tutoring system because users of ITSs are by definition working with concepts they do not understand very well (Miller, 1988). A bad user interface generally results in a system that is not only difficult to use, but one that users do not want to use. In an ITS, in addition do determining how students interact with the system, the user interface also defines how students interact with the domain being tutored. Keeping these issues in mind, the user interface was designed to provide an intuitive window into the LEAP system, and the domains in which it tutors. The remainder of this section discusses the design principles to which the LEAP Tutor user interface adheres.

The first principle is that all interactions in the LEAP Tutor user interface will take place using a first-person (i.e., direct manipulation), point and click implementation. In other words, all activities are carried out by single-action manipulation of graphic objects that map directly onto the task and domain of study. The second principle is that all objects in the LEAP Tutor user interface that can be manipulated or used in any way for information conveyance given a current system state should be visible to the user (i.e., no pull down menus or keystroke-mouse combinations to see additional options). The third principle is that interface objects should be grouped according to functionality. For instance, objects concerned with system navigation are kept together and kept separate from objects concerned with dialog control (which themselves are kept together). The fourth principle is neither require nor allow the user to control the presentation or appearance of any of the interface objects. Students need to expend their mental energies on learning the domain in question, not on customizing the user interface. The final principle is that system functions should be broken down into screen-sets, with all activities possible for a given function supported within the screen-set's objects.

For example, when a student rehearses a conversation, all of the interface objects required to support that rehearsal are contained within its screen-set (i.e., dialog objects, dialog control objects, feedback objects, session trace objects, and system navigation objects). If during that rehearsal the student chooses to review an associated declarative lesson, transitioning to the lesson results in their exiting the rehearsal screen-set (via a navigation button) and entering the study topic screen-set. While in the study topic screen-set the student would neither see nor have access to any of the rehearse conversation screen-set objects, unless they were to return to that function (via another navigation button). This type of conceptually focused organization minimizes the amount of external processing a student must commit to learning and using the interface, and focuses their processing on the task at hand - learning the tutor's domain. Figure 5 presents a hierarchical view of the LEAP Tutor's screen-sets.

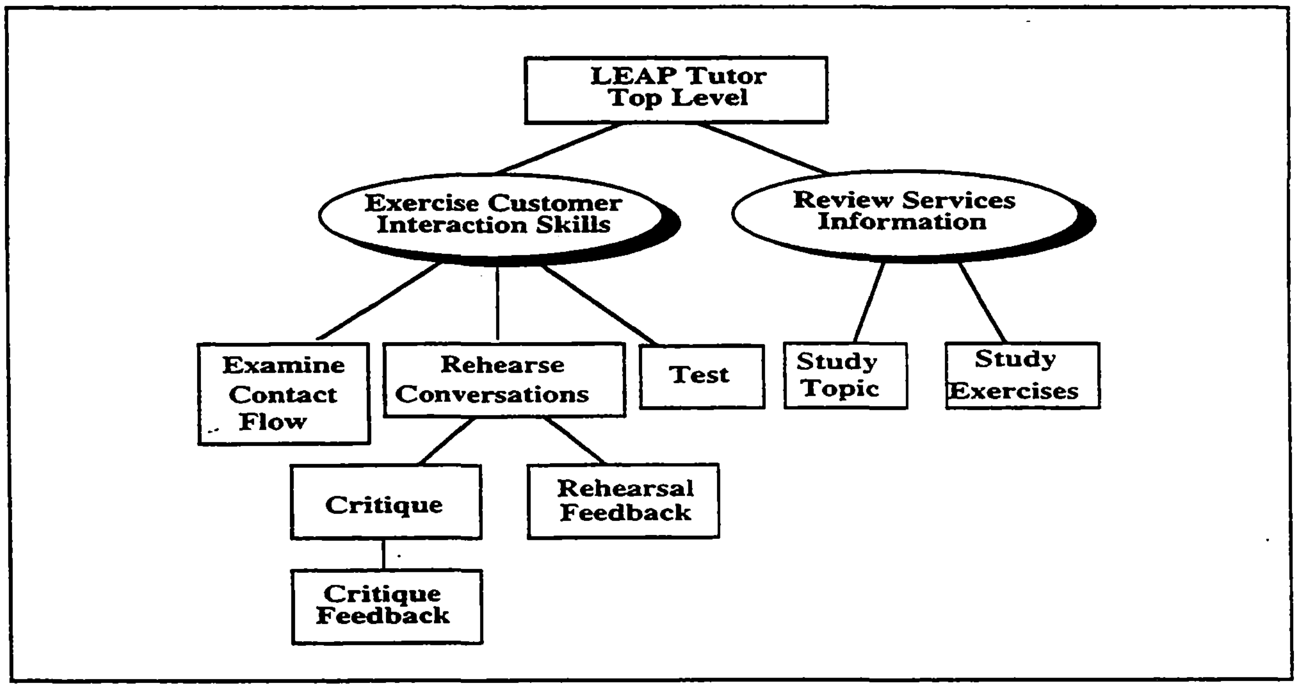


Figure 5. LEAP Tutor Screen-Set Hierarchy

As depicted by the rectangles in Figure 5, there are nine distinct screen-sets within the LEAP Tutor:

- *LEAP Tutor Top Level* — This screen-set provides the student with one window containing the set of control objects for accessing various tutor functions (i.e., navigation and recommendation), and a second window containing a selectable scrolling, hierarchical list of topics available for study and their proficiency with those topics, and a set of selectable objects representing the methods available for studying or practicing those topics.
- *Rehearse Conversation* — (used in the Demonstration, Accompany and Practice methods) This screen-set provides the student with one window containing the set of rehearse conversation control and navigation objects for accessing available tutor functions (i.e., Related Information, Help, Exit), another window containing SONAR, a third window where all dialog control activities take place, a fourth window for the coach to provide feedback, hints, and context information, and an optional fifth window containing a scrollable trace of the current session.
- *Critique* — This screen set is a super-set of the Rehearse Conversation screen set. In addition to all of the objects from the Rehearse Conversation screen set, the Critique screen set will also contain a checklist window containing lists of selectable objects (e.g., check buttons).
- *Rehearsal Feedback* — This screen set is a super-set of the Rehearse Conversation screen set. In addition to the Rehearse Conversation screen set objects, the Rehearsal Feedback screen set will contain an expanded feedback window.
- *Critique Feedback* — This screen set is a super-set of the Critique screen set. In addition to the Critique screen set objects, the Critique Feedback screen set will contain an expanded feedback window.
- *Test* — This screen set is a subset of the Rehearse Conversation screen set in which the feedback and context windows are removed, and the navigation object for viewing related information is disabled and grayed out.
- *Examine Contact Flow* — This screen-set provides the student with one window containing the set of Examine Contact Flow control and navigation objects, a second window where all dialog control activities take place, a third window for the coach to provide feedback, hints, and context information, and an optional fourth window containing a scrollable trace of the current session.
- *Study Topic* — This screen set provides the student with one window resembling an "interactive" book. The book contains "index markers" which are buttons providing interactive access to the various lessons. Each page of the book represents a particular sub-topic of a lesson and can contain text, graphics, animations and/or video sequences. In addition to continual access to the books index markers, each page also contains navigation buttons for "paging" forward or backward, for exiting the lesson and

returning to the LEAP Tutor's top-level screen set, or for exiting to a specific Rehearse conversation session on the topic being studied.

- *Study Exercises* — This screen set provides the student with one window where all exercise interactions t ake place. At the top of the window is an area for control buttons (i.e., Next, Record, Playback, Exit). Below that is an area where the specific exercise instructions are presented to the student. Below the instructions is a "Clue" area, and below the clue area is an area for student responding (i.e., checklists, areas for typed input, interactive application screens). In the special case of an audio study exercise, the student sees only the control area and instructions.

Student Model

The student model is essentially a dynamic data structure that maintains a "model" of each student within and across training sessions. Consistent with previous research (Villano & Bloom, 1992), the LEAP Tutor uses the student model to:

- Adaptively assess the student's mastery of the learning materials
- Represent the student's progress through the learning materials and recommend topics or conversations the student needs to study or practice
- Select and present instructional interventions at appropriate levels of understanding
- Provide the student with performance feedback

To accomplish these objectives, the student model employed in the LEAP Tutor is comprised of four major components:

- *Student Model Data Structure* — A dynamic representation of the knowledge and skills to be learned with associated confidence estimates
- *Student Model Performance Modeling Function* — Mechanism for tracking all student activities within the tutor and mapping them to the student model data structure
- *Student Model Updating Function* — Statistical techniques for going from a student's individual actions to estimates of how well the student knows each identified item of knowledge or skill
- *Student Model Diagnostic Algorithm* — Techniques for interpreting the student model data structure and applying the various learning techniques employed in the tutor

The LEAP Tutor's student model data structure is modeled after an Operator Function Model - OFM (Mitchell, 1987). Consistent with an OFM, the LEAP student model represents scenarios in a hierarchical network, with scenarios and major topics at the highest levels, and individual conversation activities at the lowest level. Actions can be verbal (i.e., responses by the CSR to the customer or to SONAR output), operational (i.e., a commands or data entered into SONAR by the CSR), or cognitive (i.e., information gathering, information processing, and decision making). Figure 6 presents an example of a subset of the LEAP student model representation.

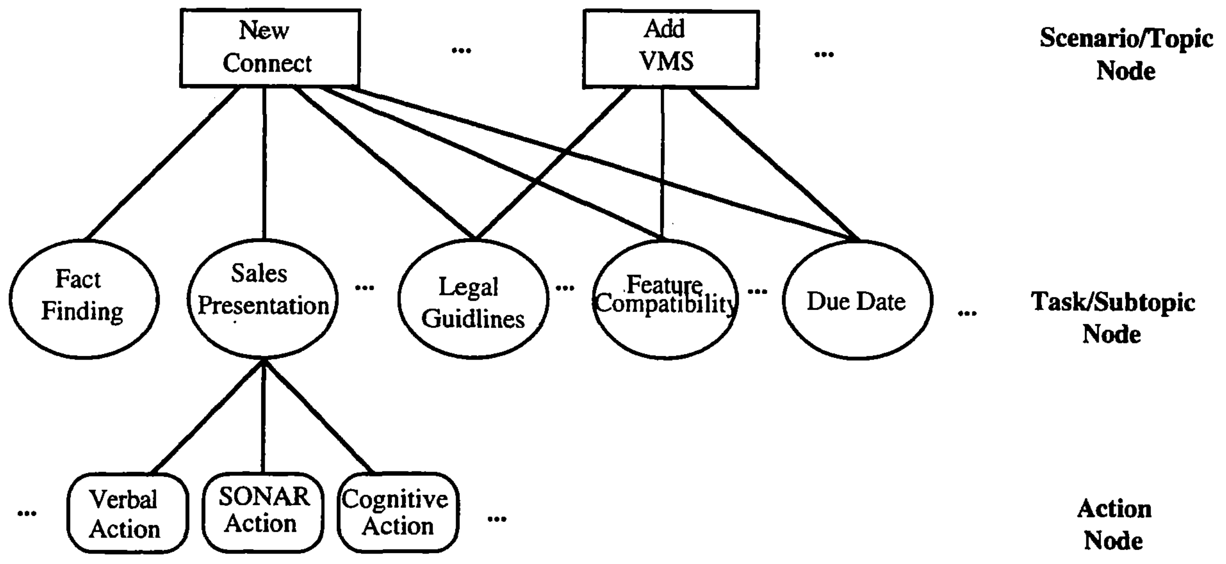


Figure 6. LEAP Student Model Representation

The top level, referred to in Figure 6 as the Scenario or Topic Node level, represents all of the possible conversational scenarios that the CSR can engage in (and supported by the LEAP Task Grammar). The second level, referred to in Figure 6 as the Task or Sub-Topic Node level, represents all of the major tasks or parts of conversations within each scenario or topic. The lowest level represents the individual actions (described previously).

In ECIS mode, for each student, the LEAP Tutor records three pieces of data for each situation-action pair: An average correct score, the number of consecutive correct or incorrect responses, and the number of times a pair is seen but not practiced. These data points are combined using a weighting function (the individual weights for each factor are interrelated and adjustable as described in the section on Parameter Editing). The computed value is a measure of the LEAP Tutor's confidence that the student can perform the activity associated with the situation. The value is updated each time a student sees or responds to a situation-action pair. The values from the situation-action pairs are propagated up to parts of conversations (i.e., sub-topics), and ultimately up to the conversation or topic level.

In the case of a Critique conversation, the LEAP Tutor records two pieces of data at the sub-topic and topic level: An average score on each critique exercise, and the number of consecutive correct or incorrect responses on each Critique exercise. These data points are combined using an adjustable weighting function and combined with the other ECIS sub-topic and topic estimates.

In RSI mode, the LEAP Tutor records three pieces of data for each topic and sub-topic: The number of times a lesson is reviewed, an average score on each Study Exercise, and the number of consecutive correct or incorrect responses on each Study Exercise. These data points are also combined using an adjustable weighting function, and also combined with the ECIS estimates propagated to the sub-topic and topic levels to provide the tutor's overall assessments. These combined estimates are then converted into one of five categorical labels (i.e., "Untried," "Needs Practice," "Almost," "Good," or "Excellent"). The categorical conversation is also adjustable, so that instructional designers or field instructors can vary the "scores" a student needs to move from one level to the next.

Given a lack of consensus in the student modeling literature on methods for updating confidence estimates based on observable data (Bloom, Villano, VanLehn, Jones, Watson & O'Bannon, 1992), the parameters used in the LEAP Tutor's student model updating function were made to be adjustable.

The diagnostic algorithm of the LEAP Tutor's student model is based on two assumptions: (1) The effect of practice on learning skills and knowledge can be fitted to a traditional, negatively accelerating learning curve, such that the more one practices, the greater that person's odds of responding correctly. (2) The need for practice can be fitted to a negatively decelerating learning curve, such that as skills increase, the need for practice decreases.

As stated previously, each student has their own student model representation that is saved across training sessions, enabling the student to resume their instruction based on their performance on pervious sessions. In addition, student models will only be viewable by their owners. The intent of LEAP is to provide CSRs with a coached practice environment that allows the student to monitor their own learning progress, it is not intended to allow supervisors to use LEAP's findings to grade or assess the CSRs. To protect each student, the student model will also contain a user-model component. This user model component will contain information about the user (i.e., user name, user preferences, etc.) as well as a private password for security protection.
Parameter Adjusting

One of the more powerful features of the LEAP ITS Platform is its Parameter Adjusting function. The LEAP Tutor makes its instructional decisions on what topic to recommend, what conversation to execute, how to evaluate a student's responses and how to apply its within-practice instructional methods (i.e., skim & scaffold) based on sets of parameters that can be adjusted by course developers and instructors.

The rationale for this approach is as follows: (1) We were able to identify what factors influenced these decision making strategies (each has been used in some previous instructional system or method), but not the relative contributions of each when taken together. And, (2) parameterization of these factors would provide us with the means of conducting systematic empirical investigations into this issue.

The LEAP Parameter Adjuster has been designed so that the weights given to each factor can be easily modified. The weights (and hence the contributions) of each factor can range on a continuous scale from zero (no contribution) to 100 (full contribution). In addition, these factors are interrelated on all but the determination of the application of within-practice instructional methods, so that increasing the weight of one factor will produce uniform decreases to the weights of the other factors (and vice versa).

Topic choice is based on three interrelated factors: (1) Sequential ordering of topics (as represented in the topic/sub-topic structure), (2) relationship to the last topic/sub-topic studied, and (3) the student's proficiency on each topic/sub-topic. The sequential order of topics represents the sequence determined in accord with current instructional design practices. In other words, if an instructor were going to build the "best and only" route through the course, it would be represented by this sequential order. However, different students learn best in different orders, so we have the two other factors that can produce a "resequencing" of the topics as recommended to the student. Relation to the last topic influences recommendations on a conceptual relatedness rather than sequential level. With regard to proficiency, at present the author can select from among three predefined prioritization settings that differ with regard to the way the tutor recommends student progress through topics in the "Untried" to "Almost" categories:

- *Prefer "Almost" to "Needs Practice" to "Untried"* — Elevate students proficiencies on individual topics/sub-topics to good before proceeding to other topics.
- *Prefer "Needs Practice" to "Untried" to "Almost"* — Elevate students proficiencies on all individual topics/sub-topics to "Almost" before proceeding to "Good."
- *Prefer "Untried" to "Needs Practice" to "Almost"* — Elevate students proficiencies on all topics/sub-topics one level at a time.

Conversation choice is based on six factors (5 of which are interrelated): (1) relation to selected topic, (2) complexity of conversation (in terms of the number of SA pairs in the conversation), (3) the number of times the conversation has already been seen or practiced by the student, (4) the students current skill level on the conversation, (5) an independent teacher preference rating, and (6) the student's preference for hard vs. easy conversations (the only independent factor).

At present, student response evaluation is only implemented for ECIS' mode. Plans are in progress to extend this capability to RSI mode and to increase the number of parameters employed in ECIS mode to include response latencies and times. Student response evaluation is based on three interrelated factors: (1) Average correct score when asked, (2) the number of consecutive correct (or incorrect) when asked, and (3) the number if times the item is seen but not asked.

The application of within-practice instructional methods (i.e., skim and scaffold) is the only Parameter Adjusting function in which the parameters are not interrelated. In this function, the author can set the parameters for skim (In Topic) and scaffold (Out of Topic) separately. However, each uses identical parameters: (1) Question the student when not known, (2) question the student when known, and (3) Decrease questioning when known.

With regard to skimming information that is "In Topic," the question when not known parameter is set to always as a default -- the objective is to get students to practice that information within the topic they are studying that they do not yet know. The question when known parameter is set at the midpoint as a default and refers to how often do you want the student to get prompted to respond to something they already know -- just enough to keep the information active. The decrease questioning when known factor refers to the how gradually you want the system to stop asking the student something they have demonstrated they know.

With regard to scaffolding information that is "Out of Topic," the question when not know parameter is set to never as a default -- we do not want students to get frustrated trying to answer items the tutor has no reason to believe they know. However, the more times a student sees and item that is unknown and out of topic, the tutor's assessment is updated to the point where is reflects some level of knowing for those items not practiced by seen. This is where the question when known-out of topic factor comes in -- we begin to ask the student to respond to items that are out of topic, but they should have some knowledge of. The default setting for this parameter is at the midpoint. The decrease questioning when known is the same for Out of Topic scaffolding as for In Topic skimming.

Hardware and Software Requirements

The LEAP Tutor is envisioned to ultimately migrate towards delivery on U S WEST Information Technologies' Intelligent WorkStation (IWS). The IWS is a platform composed of front-end UNIX X-terminals with 19 inch color monitors supporting 8 bit color displays, that is connected to several application and communication serves via a high speed (10 megabits per second) ethernet local area network (LAN). The IWS is proposed to be capable of supporting installations varying in size from one to 160 CSRs. The IWS provides access to the database application the LEAP Tutor will need to interact with, the Service Order Negotiation and Retrieval (SONAR) system. The SONAR application itself resides on an IBM 3270 mainframe accessed through the IWS application/communication server and displayed on the front-end X-terminal using a 3270 emulator.

For the purpose of evaluating the LEAP tutor's instructional effectiveness, we will be using a Sun Sparc 10 Model 41 with 128 MB of RAM as a server, and an external 2 GB disk drive for multimedia file storage. The Model 41 is a dual processor model, with one processor to be used as a server for the LEAP software and the second to be used for multimedia file serving. The server will be connected to the IWS LAN to enable access to SONAR. Connected to the LEAP/multimedia server via a dedicated LAN will be a series of Sun Sparc 2s and IPXs running in stand-alone mode. This configuration provides us with the ability to test the tutor without adversely impacting the IWS LAN, and provides us with a work-around the IWS limitations in the areas of audio and video processing. The LEAP system is built in common LISP, using a separate user interface code written in C++. In addition, the LEAP system uses two other pieces of software: (1) A standard 3270 terminal emulator for access to SONAR, and (2) a Bellcore application called XFUR, used to interact with the terminal emulator to allow the LEAP Tutor to control and monitor student actions taking place in SONAR (Lefkowitz & Farrell, 1991).

References

Anderson, J. R. (1988). The expert module. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Bloom, C. P., Villano, M., VanLehn, K., Jones, J., Watson, P. K., & O'Bannon, M. (1992). *Application of Artificial Intelligence Technologies to Training Systems: Computer-Based Diagnostic Testing System.* (Research Rep. No.

AL-TR-1992-0072). Brooks Air Force Base, TX: Air Force Human Resources Directorate, Technical Training Research Division.

Burns, H. L., & Capps, C. G. (1988). Foundations of intelligent tutoring systems: An introduction. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Carroll, J. M. (1990). *The Nuremberg funnel*. Cambridge, MA: The MIT Press.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive Apprenticeship: Teaching the crafts of reading, writing and mathematics. In L. B. Resnick (Ed.), *Knowing, learning and instruction*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 453-494.

Halff, H. M. (1988). Curriculum and instruction in automated tutors. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Lefkowitz, L. & Farrell, R. (1991). *Assessing the applicability, utility and feasibility of WITS intelligent tutoring for your operations system*. (Technical Memorandum No. TM-STS-019544). Piscataway, NJ: Bellcore.

Lesgold, A., Lajoie, S., Bunzo, M., & Eggan, G. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin & R. Chabay (eds.), *Computer-assisted instruction and intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 201-238.

Miller, J. R. (1988). The role of human-computer interaction in intelligent tutoring systems. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Mitchell, C. M. (1987) "GT-MSOCC: A domain for research on human-computer interaction and decision aiding in supervisory control systems," *IEEE Transactions of Systems, Man, and Cybernetics*, vol. 17, pp. 553-570.

Sweller, J. (1988). Cognitive load during problem solving: Effects of learning, *Cognitive Science, 12*, pp. 257-285.

VanLehn, K. (1988). Student modeling. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associated.

Villano, M. & Bloom, C. P. (1992). *Probabilistic student modeling with knowledge space theory*. (Research Rep. No. AL-TP-1992-0046). Brooks Air Force Base, TX: Air Force Human Resources Directorate, Technical Training Research Division.

Wenger, E. (1987). *Artificial intelligence and tutoring systems*. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

N96- 14989

An Authoring System for Creating a Practice Environment in the Network Service Field

Minoru Kiyama and Yoshimi Fukuhara
NTT Network Information Systems Laboratories
1-2356, Take, Yokosuka, Kanagawa, JAPAN
E-mail: kiyama@nttkb.ntt.jp
FAX: +81-468-59-3633

Abstract.

This paper describes an authoring system whose main purpose is to reduce the cost of developing and maintaining courseware which contains procedural knowledge used in the network service field. This aim can be achieved by considering the characteristics of this field. Material knowledge is divided into two parts, behavioral knowledge and procedural knowledge. We show that both of these parts are constructed by an easy authoring methods and efficient modification algorithms. This authoring system has been used to build several types of courseware, and the development costs have been reduced.

1. Introduction

The work presented here concerns a practical authoring system for training in the network service field, which is different from the authoring system used in schools. In our company, which is a supplier of network communication services, there are more than 200,000 employees engaged in maintenance, troubleshooting and sales of network services. Therefore expert training is very important. Network services personnel need many types of useful and interesting courseware for studying a lot of equipment and a variety of businesses, but the authors have limited time to build courseware. Why aren't intelligent tutors, by which students can give effective and useful instructions, being used in industry? One of the main reasons is the lack of practical intelligent development tools[1]. We developed an authoring system for the scene-based intelligent Computer Assisted Instruction (CAI) system in order to learn declarative knowledge, such as, what are the concepts and architectures of the switching systems[2,3,4]. The usefulness of the courseware built by this system was confirmed. It is also important for people who work with network equipment to be trained in procedural knowledge, such as how to operate digital switching systems, as well as declarative knowledge. It is known that a practice environment, based on direct manipulation, including interactive digital video for the sake of reality, helps people to effectively acquire procedural skills. We are particularly interested in a practical authoring system for tutoring procedural knowledge of many kinds of domain.

Another aspect of our work concerns the differences between the authoring system for the industrial field and the one for schools. Authoring systems must reflect the characteristics of the field that is applied. Physics laws, in general, are stable domains. The contents of such courseware are not changed, once the courseware is well established. On the other hand, many kinds of network equipment described in courseware are continually undergoing slight changes, and therefore the courseware itself needs to be frequently changed. In most cases, it is not necessary to set a deadline when building courseware concerning a stable domain, such as physics. The usefulness of that material will not decrease with a passing of time. In the network field, setting a time limit is important, because a courseware's usefulness might reach its peak before and after installing new equipment. The courseware's value decreases sharply after that time. It is useful for authors to have an authoring system that can modify courseware very simply and easily, and can develop courseware rapidly.

This paper is organized as follows. Section 2 presents the requirements for authoring systems in the network service field. Sections 3 and 4 explain how to build behavioral knowledge and procedural knowledge, respectively. Then in section 5, the development costs of our system are evaluated. Finally, we conclude with current status of the system.

2. Overview of Our System

2.1 Characteristics of network service field

It is important to make clear the characteristics of the network service field from the viewpoint of training when the authoring system is designed. These characteristics should be considered from the supplier side, not the users side.

a) Large scale - many kinds of complex equipment -
Supplying network services requires providing hundreds of kinds of complex communications equipment, such as digital switching systems, automobile telephone systems, ISDN systems and so on. The kinds of equipment are increasing as we begin to provide new services and old equipment still remains for a long time, until it is depreciated. In the case of switching systems, crossbar switching systems have been installed for more than a decade and more than ten kinds of digital switching systems are now being used.

b) Procedural knowledge
It is very important for employees to learn how to operate the equipment, as well as know its concept and how it is constructed, for the sake of troubleshooting and daily operations. Most equipment is operated by turning on/off switches on the panels, pointing to the abstract figure drawn on the display, and typing a command like a character-based command of UNIX.

c) Non-stop
Network systems are the infrastructure of a nation. Therefore, human error must not be the reason for a system being down. Novice operators can not use the actual machines in order to learn the operations.

d) Area configuration
Network services cover the nation, but equipment is installed distributively in several local areas. Different environments, such as the number of customers and the amount of communications traffic, cause different system configurations. Although the basic operations of the devices are the same in each region, some detail procedures are slightly different.

e) Frequent changes in equipment and operation
Many kinds of network systems are in general use for a long time, but they are continually undergoing slight changes in order to improve the systems. Part of the equipment or some of its software may change, therefore the appearance of the equipment and the order of the operation sequences may also change. The changes in the environment may also affect the operations.

f) Timely training
The training of equipment operations should be undertaken before and after introducing new equipment. The life cycles of equipment and training are depicted in Figure 1. The value of the training drops sharply after the peak.

g) Who is author
This characteristic is related to the organization of the company, and not with the equipment itself. Most courseware is built by operators and instructors rather than designers of the equipment. These authors do not have detail knowledge of the equipment and do not usually have any programming skills.

2.2 Requirements for the authoring system

We identify some requirements for the authoring system, which reflect the above characteristics in the network service field. Some characteristics may cause other requirements. These will be handled by setting the priority and trade-off of the conditions. Figure 2 shows the relationship between these characteristics and the requirements.

1) Practice environment
It is very important for people who work with network equipment to learn procedural knowledge, as well as declarative knowledge. The authoring system needs to build a practice environment for people

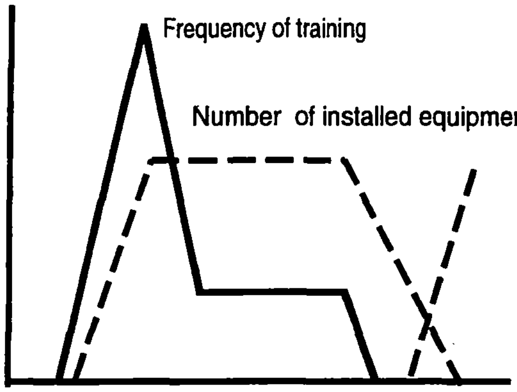


Figure 1. Lifecycle of equipment and training

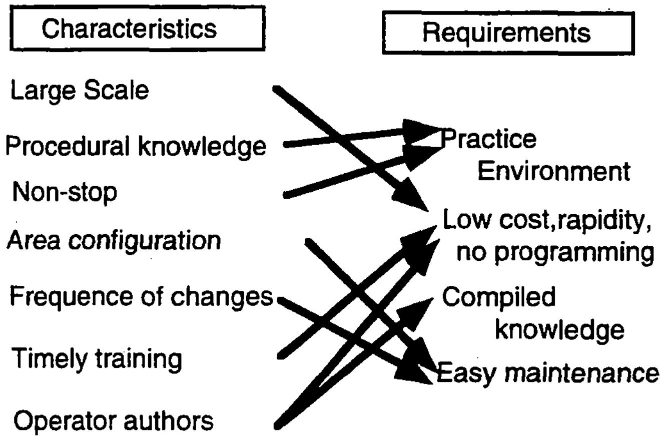


Figure 2. Relationship between field characteristics *and requirements for authoring*

to be trained in procedural knowledge. We have developed another authoring system for the scene-oriented intelligent CAI tutor for learning declarative knowledge, because these two types of knowledge require different types of instruction.

2) Low cost, rapidity and no programming
Courseware is built by employees who work with equipment rather than design the equipment. Thus, a practice environment should be created without programming. Many kinds of equipment requires spending extra time building different courseware and providing the courseware before the peak requires a rapid authoring system.

3) Compiled knowledge
Courseware authors do not have detail knowledge of equipment, like designers. Our system's aim is that logical domain knowledge, except for real-domain-knowledge, can be constructed by using the knowledge described in a users manual or an operations guidebook. Authors can supplement the lack of logical knowledge by using real-domain-knowledge, such as graphics, voice, motion pictures and so on, because they can see the behavior of the equipment very well. So the quality of the courseware will not be so bad.

4) Courseware maintenance
Many kinds of equipment described in the courseware are continually undergoing slight changes, and therefore the courseware itself needs to be frequently changed. The authoring system must have the capability of modifying courseware very simply and easily. The area configuration and the growth of that configuration determines the requirements for courseware maintenance.

2.3 Outline of system architecture

First, we show the operation model of the equipment in the real world, depicted in Figure 3. When the equipment receives a stimulus, such as trouble or manipulation, from other subsystems that are related to the equipment, the operation goal is fixed and the operator manipulates the equipment for the goal. An expert operator might assist them depending on the circumstance. Figure 4 shows the architecture of our system, which reflects the operation model in the real world. The material knowledge consists of three parts; simulation knowledge, training knowledge and real-domain-knowledge. Simulation knowledge represents the behavior of the equipment and enables the system to act in the same way as equipment in the real world. The tutor having that knowledge, will respond by either correcting or not correcting operations blindly. Training knowledge represents the procedural knowledge of the operations and manages the operation goal and stores the correct sequence of the operation to be learned. Real-domain-knowledge is the information to be presented to students and includes text, graphics, voice and motion pictures. Simulation knowledge, training knowledge and real-domain-knowledge reflect the equipment itself, the external stimulus and the appearance of the equipment, respectively, in the operation model in the real world. Our authoring system consists of three parts for each domain knowledge. The tutoring system, which executes the simulation and training knowledge, is the interpreter for unary representation. Some knowledge is transformed with the instruction mode which is specified by the students. These architectures satisfy the requirements for the authoring system. The reasons and the details are shown in the following sections.

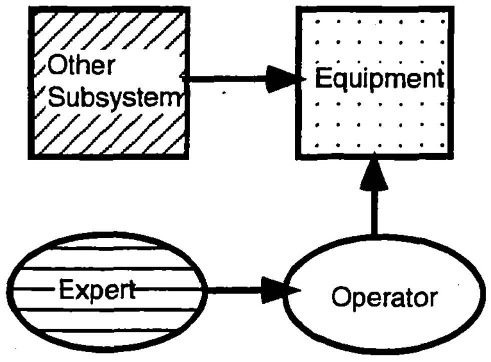


Figure 3. Operation model in the real world

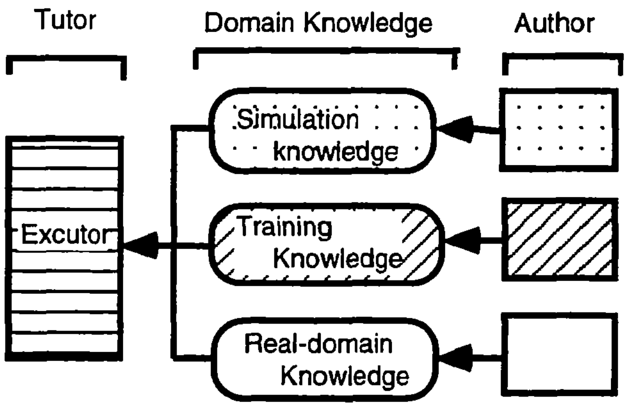


Figure 4. System architecuture

3. Authoring for Simulation Knowledge

3.1 Design concept

Simulation knowledge enables the system to behave in the same way as equipment in the real world, i.e. to blindly respond to either correct or incorrect operations. We have adopted the state transition model (STM) to represent simulation knowledge. This model is useful in describing the behavior of the network equipment. The steps of most operations are discussed below. Operators manipulate the equipment, then the equipment provides a response which the operators can see, and the internal state of the equipment changes to another state. These operations are represented exactly in STM. STM is also useful in describing the behavioral knowledge in the users manual, in which the sentences are written in a set of short pieces of knowledge, such as itemized form. We can represent the compiled knowledge, because STM covers both detail knowledge and rough knowledge in its representation ability.

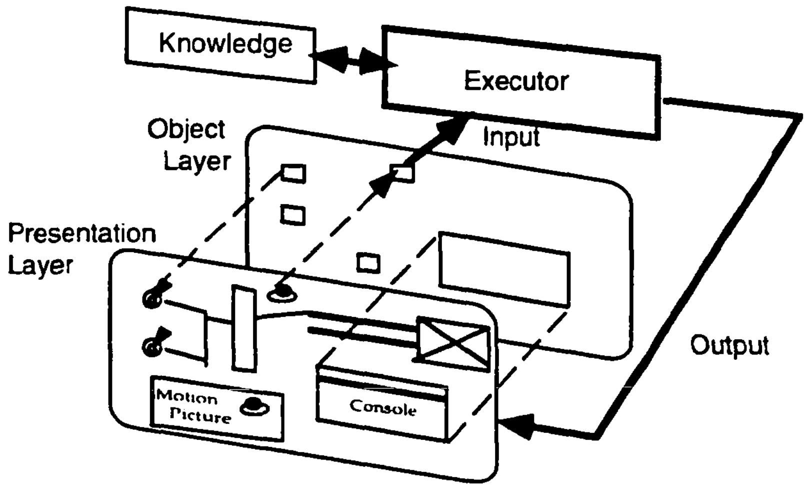


Figure 5. Layer of simulation knowledge

In the network service field, graphical user interfaces on the display are constructed by non-standard widgets, such as, network routes maps, as well as standard widgets such as a menu and dialog boxes. Two approaches exist for authoring the non-standard widgets: providing the object-base graphic editor and dividing the widgets into the transparent object layer and the presentation layer, like HyperCard (Figure 5). The object-based graphics editor in STEAMER has large sets of icons specifically designed to depict objects and represent behavior of objects in the propulsion domain[5]. It is clear that this editor will reduce the cost of authoring and provide easy modification to the courseware. However, we must implement an object-based editor for each domain. The cost is expensive and it takes a relatively long time to develop. We feel this approach is not suitable for rapid authoring and low-cost courseware development of many kinds of equipment.

3.2 Basic representation

We describe the simulation knowledge in the form of STM. The element, tuple is presented (P, I, O, Q), and is also visualized as follows;

$$\text{state P} \xrightarrow{\text{input I/ output O}} \text{state Q.}$$

This element means that the equipment receives input I at state P, then responds to the operator showing output O, and finally changes to the next state Q. We explain the implemented function in our system as follows.

a) Input: consists of object and event. The object is the target for the input. We have implemented two types of objects; a button object and a window object, which represents the button on the panel or the display and the character-based monitor console, respectively. The event is the stimulus given by the operators. The mouse and keyboard are used for input events.
b) Output: means the response to the operator from the equipment. The main functions are presenting graphics, voice, motion pictures on the presentation layer and setting the activity of group states.
c) State: represents the logical state of the equipment.

Additionally, you can place the objects on a motion picture on the presentation layer. The life of the object should be specified in this case.

3.3 Extended representation

When we try to describe the behavior of complex equipment in the real world, a lot f states may have to be defined. The equipment consists of several subsystems, and compound states related to some subsystems, which cause a number of states. Compound states, p1-q1, p1-q2, p2-q1,p2-q2, are the combination state of each subsystem when state p1 and p2 belong to subsystem A and state q1 and q2 belong to subsystem B. We have extended STM by adding the concept of group. A group is formed from a set of elements that have closed state transition, and therefore the compound state can be reduced. As a result, we can represent the behavior economically. Any inputted events are distributed to each group equally and transition occurs in each group independently. Although each group acts independently in state transition, the objects can be shared among the group. An event for shared objects may cause several transitions in each group. The order of firing, such as firing all of them simultaneously or no firing or firing the latest active state and so on, can be defined as simulation knowledge.

3.4 Tools

We provide some tools for entering the elements of STM and debugging the behavior. Two kinds of user interfaces are implemented for inputting: a table-formed interface for inputting the tuple form and a graphical interface for inputting the connection of states. You can chose either of them depending on the situation. The debugging tools execute the static check of contradiction and duplication, such as removing the isolated loop without an end state, and dynamic check of the behavior, such as, the traces of transition and execution on the way to the transition sequence. Our tutor executes unary representation, extending STM.

4. Authoring for Training Knowledge

4.1 Design Concept

Because tutoring systems with only simulator knowledge will respond to either correct or incorrect operations blindly, students can not take pedagogical instruction. When operators manipulate the equipment in a situation, they have some goals such as solving some troubles and have the plans and procedures in order to reach the goals, such as what sequences to use. It is important for students to learn the plans and procedures. Our authoring system stores the procedures as training knowledge, which is separated from simulation knowledge, i.e both kinds of knowledge are managed independently. The outline of authoring for the training knowledge is discussed below. Training knowledge is represented as the input sequences of STM which describes the simulation knowledge. Authors can build training knowledge by executing the process shown in Figure 6. First of all, authors define simulation knowledge and construct real-domain-knowledge when they build the courseware. After this, they can manipulate the display in the same way as with real equipment. Next, they enter the correct sequences of a procedure on the display by means of executing STM represented as the simulation model. The authoring system records the log of input as training knowledge. Our executor interprets only unary representation STM, so training knowledge is transformed into STM before the student starts to learn.

The merits of dividing the material knowledge into two parts are described below. i) Building training knowledge by using simulation knowledge will reduce the cost of developing the courseware, ii) We can add the procedures to be trained step by step without affecting the simulation knowledge. This facilitates easy maintenance of the courseware. iii)When some parts of the courseware are modified, both types of knowledge can be modified independently in some cases.

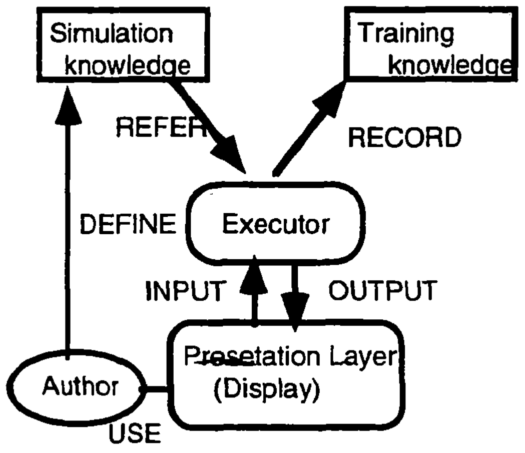


Figure 6. Building the training knowledge

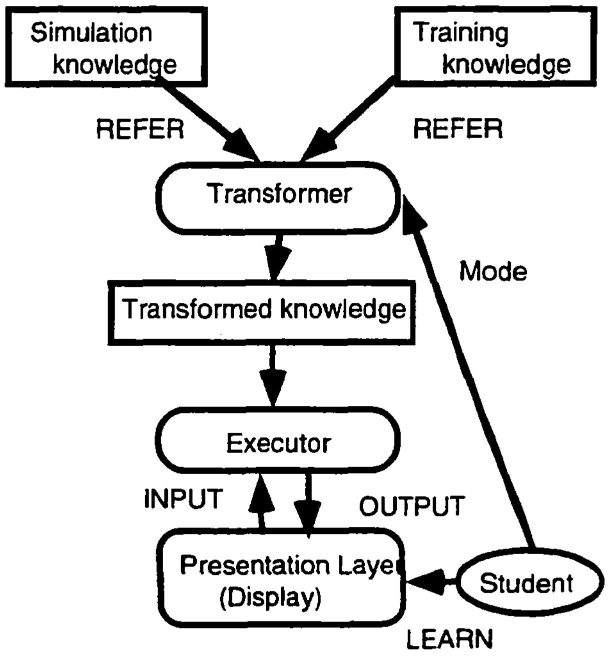


Figure 7. Transfoming the training knowledge to STM

4.2 Transformation algorithm

The transformation algorithms from the training knowledge into STM are shown below. When a student starts to learn, they choose an instruction mode: in the imitation mode the system shows the student the correct operations, in the guide mode the student follows the operation guided by the tutor, and in the practice mode the student manipulates freely. The transformations are accomplished for the mode which the student has chosen. We show a simple algorithm in the case of the imitation mode, although similar algorithms can be adapted in other cases.

The following is given; simulation knowledge T: {t1, t2, t3,...,tn}, tj = (pj, ij, oj, qj), training knowledge L: {l1, l2, l3, ... lm}, lk = (ik, pk), Confirmation input ic. For all elements lk of set L, find the tuple tj in T which satisfies the following condition, then rewrite the tuple of T. The set T' including rewritten tuples constructs the transformed training knowledge.

[condition] ik = ij of tj and pk = pj of tj.

[rewriting rule] tj is rewritten into (pj, ic, o'j, qj) , in which o'j is oj added to the two outputs, blinking the next object ok+1 and inactivating the group in case pk and pk+1 belong to a different group.

4.3 Maintenance method

When the position of a button is changed on the real equipment, the authors redraw the button on the presentation layer and simply move a button object on the object layer. It is not necessary to modify any STM elements and training knowledge. When changing an equipment's appearance, such as color or form of parts, no domain knowledge needs to be modified, only the presentation layer. When the order of the operations is undergone changes, such as reversing the sequences, only the training knowledge need to be rebuild. No modification to the simulation knowledge needs to be made. Additionally, training knowledge can be added without affecting the simulation knowledge. On the other hand, the authors must modify all types of knowledge when a button is removed. They must delete the button on the presentation and object layers, and modify the simulation and training knowledge. The system shows the authors some parts of the simulation and training knowledge that must be modified, but it is author's responsibility to correctly modify the knowledge.

5. Development Cost

The cost of developing the courseware using the authoring system has been compared with the cost of programming. Figure 8 shows the development cost of the automobile telephone system courseware. The man-hours for creating logical knowledge is 15 percent that of the programming cost, although the cost of creating real-domain-knowledge is the same in both cases. The total development cost using our system, which includes building real-domain-knowledge, is a quarter of the cost of building the courseware using programming.

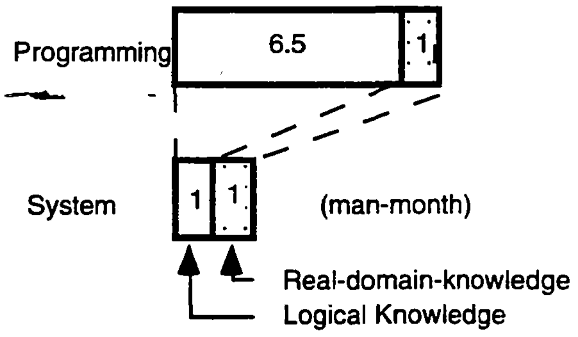


Figure 8. Authoring cost comparison

6. Conclusion

We have developed a practical authoring system for the network service field. The authoring system has several facilities which provide low-cost, rapid authoring. This system makes it easy to build and change procedural knowledge courseware. Simple and efficient methods are shown, that reduce the development and maintenance cost. The system was used to build large courseware, and the development cost was reduced to a quarter. This system is working on personal computers. Many operator authors are building more than a hundred types of courseware in a year. A tutoring system was introduced into our three hundred branches and five training centers. It is important to balance the tutoring ability and the authoring cost when a practical authoring system is developed.

Reference

1. Woolf,B.P., Soloway,E., Clancey,W.J., Van Lehn,K. and Suthers,D., "Knowledge-based Environments for Teaching and Learning," AI Magazine, Vol.11, No. 5, 1991.
2. Fukuhara,Y., Kiyama,M. and Nakata,K., "Multimedia Authoring System for Practical Scene-oriented ITS (CAIRNEY)," Int. Conf. on Multimedia in Education and Training, 1991.
3. Morihara,I., Ishida,T., and Furuya,H., "Rule-based Flexible Control of Tutoring Process in Scene-oriented CAI Systems," Int. Conf. on Artificial Intelligence Applications, 1987.
4. Ishida,T., Sasaki,Y. and Fukuhara,Y., "Use of Procedural Programming Languages for Controlling Production Systems," Int. Conf. on Artificial Intelligence Applications, 1991.
5. Hollan,J.D., Hutchins,E.L., Weitzman,L.M., "STEAMER: An Interactive, Inspectable, Simulation-Based Training System," in Artificial Intelligence and Instruction, Kearsley G., Adison & Wesley, 1987.

N96- 14990

Integrated Intelligent Training And Job Aiding For Combustion Turbine Engines

Clifford M. McKeithan Jr.
Galaxy Scientific Corporation
2310 Parklake Dr, Suite 325
Atlanta, Georgia, USA
George H. Quentin, Ph.D.
Electric Power Research Institute
3412 Hillview Ave
Palo Alto, California, USA

INTRODUCTION

Over the past decade, combustion turbines have become a prominent source of electric power generation in the United States and around the world. Advanced gas turbines have offered substantial gains in firing temperatures, thermal efficiency, and electrical output. Therefore, a family of expert system diagnostic tools has been introduced to monitor their operating performance and their maintenance condition.

One particularly important feature of gas turbines is their rapid startup feature that has made these engines valuable for peaking duty, i.e. to provide short-term supplemental power when all baseload sources are insufficient to meet electrical demand peaks. Because such windows of opportunity for peaking power are brief, failure-to-start incidents must be rapidly diagnosed. This has led to the development of an expert system job aid, or startup advisor, to help technicians detect and overcome the impediment.

This paper describes an ongoing program to augment such an expert system gas turbine startup advisor, known as the EPRI SA°VANT^{tm} System, by including an intelligent training package. It will give a brief background on the SA°VANT development and an overview of its evolution into a full-blown Gas Turbine Information System (GTIS) for rapid access of on-line documentation, diagnostics, and training.

In particular, the paper will address:

(1) The conversion of the knowledge base used by the SA°VANT startup advisor so that it can be used for both training and job aiding; and
(2) The hypertext-oriented user manuals being incorporated into the system for rapidly accessing on-line documentation at the job site.

BACKGROUND OF SA°VANT SYSTEM DEVELOPMENT

EPRI began developing the SA°VANT system in 1984, initially to establish the viability of expert system job aids for field use by maintenance technicians. A simple pilot system enabled novice technicians to locate short circuits in turbine control systems in about sixty-five minutes, whereas experienced technicians without the job aid required about sixty minutes on average. Novices could not locate faults without the job aid or other support. The original computerized job aid was cumbersome, with built-in delays, and did not improve performance times for experienced technicians.

This led EPRI to develop a compact portable computerized job aid to eliminate built-in time delays and thereby expedite troubleshooting. Applying the job aid with this new SA°VANT system, the experienced technicians cut their problem solving time in half (to about twenty-five minutes). Surprisingly, the novice technicians made similar gains, solving the same problems in about twenty-six minutes.

From this baseline experience, EPRI chose to apply the SA°VANT know-how for helping technicians to diagnose failure-to-start incidents on large gas turbines in peaking service. Of the existing power generation fleet in the USA of roughly two thousand gas turbines generating 50,000 megawatts, approximately 90 percent of those engines are operated as peaking units. Generally, without special attention, starting reliability of peaking units has been at a level of 85% or lower due to a variety of reasons. By contrast, the power industry has set starting reliability goals of 95% or better.

By developing an effective startup advisor job aid, EPRI could help the power industry attain such startup reliability goals by avoiding aborted startups. This also avoids possible engine damage due to faulty restarts with its attendant hazards, as well as costly delays at a critical time.

As noted above, peaking gas turbine units supply additional power to meed peak demands, when normal baseload power plants are already operating at full capacity. Such gas turbines are often located at remote sites, far apart, not manned at all times, and are used infrequently. When such engines are needed, they must respond on short notice. This does not leave much time for troubleshooting of any problems which may occur. Since many of the problems involved in starting the turbines are not visible prior to starting the unit, and the units are not usually started until they are needed, operators and technicians must be able to rapidly troubleshoot and correct any impediments.

To compile the original knowledge base for the gas turbine startup advisor expert system, EPRI contractors gathered several gas turbine experts familiar with particular engine models, including utility troubleshooters. Using a roundtable format, the procedural expert system was generated, leading technicians carefully step-by-step from symptom to cause.

While the knowledge base was being finalized, the computer-interface though which technicians would access the expert system job aid was evolved. EPRI chose to incorporate several multimedia concepts in an audiovisual-oriented system, using drawings, pictures and video resources, as well as voice I/O. A block diagram of the system appears in Figure 1. The basic SA°VANT hardware was originally an MS-DOS compatible system based on dual 80386SX processors. Once central processing unit (CPU) controls a graphics display, while the other runs the expert system. As an alternative to the traditional keyboard/video display interface, the EPRI system included a speech synthesis/voice recognition interface to allow "hands-free" operation of the unit.

Once the expert system was developed, the SA°VANT startup advisor was field tested to determine its effectiveness. This system was tested by a broad range of technicians, ranging in experience for six months to sixteen years. The use of SA°VANT resulted in a 25% decrease in time required to troubleshoot the turbine, with an 81% decrease in calls for assistance and an 89% decrease in the number of aborted restart attempts (Quentin, 1991).

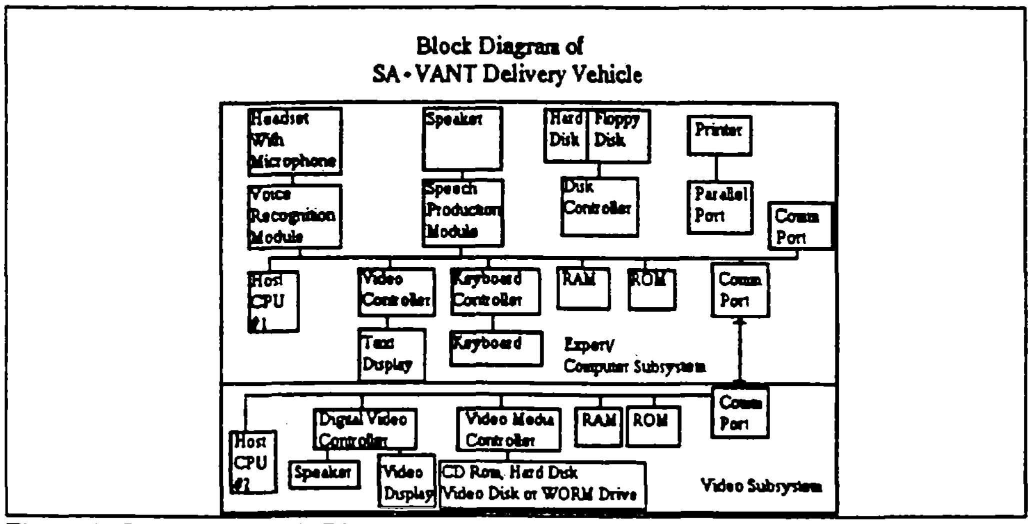


Figure 1: SA•VANT Block Diagram

THE GTIS: AN OVERVIEW

GTIS is a system composed of three major parts: an intelligent tutoring system (ITS) (Polson, 1988), a job aid, and a hypertext information manager. All three functions have been linked into a single, consistent, user environment. The GTIS runs on the same basic hardware as the SA°VANT system, with a few enhancements to both the hardware and the operating system.

Part 1: The Intelligent Tutoring System
The first major change to the SA°VANT system was the addition of the ITS. The goal of this addition to SA°VANT was to enable the technicians to practice solving troubleshooting problems during the time that they have available when the gas turbines are not being run. The hypothesis was that by practicing on a simulation-based training system, the technicians would be better prepared to troubleshoot failures within the real system when they occurred.

The simulation used by the GTIS is a surface level, static simulation. Most simulation elements maintain the same value throughout the diagnosis of a problem, with the exception of those elements that determine the ability to restart the turbine. When the trainer is started, the operator is presented with a description of the simulated failure. The operator is then free to troubleshoot the system, with the help of the intelligent tutor.

The ITS is made up of five components: the student interface, the instructional environment, the instructor model, the student model, and the expert model. These components are depicted in Figure 2.

The Student Interface

The student interacts with the ITS using a graphical user interface, depicted in Figure 3. The display consists of images and/or text on the screen. The operator can select items by pointing at the object of interest and clicking the mouse button.

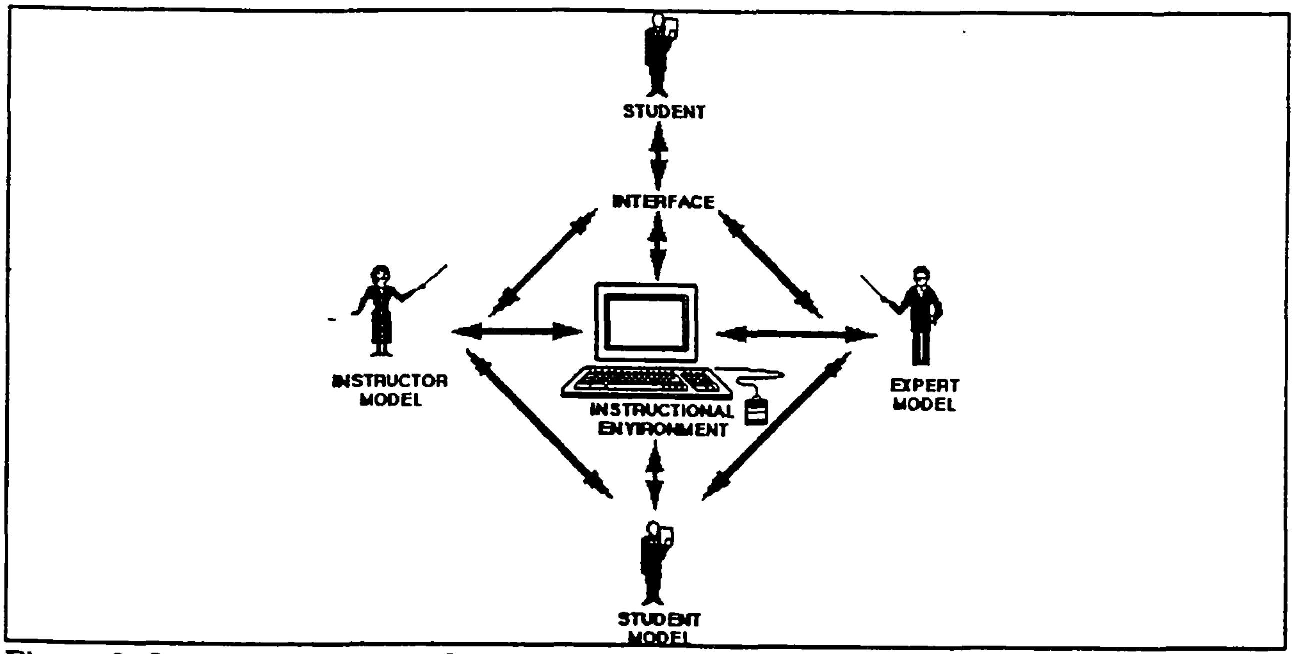


Figure 2: Intelligent Tutoring System Diagram

The Instructional Environment

As stated above, the ITS uses a static simulation as the instructional environment. In complex domains such as gas turbine engines, a combination of simulation and an ITS has proven to be effective in teaching troubleshooting skills (Johnson, 1988a). The images used in the simulation are high resolution, digitized images taken from a video recording of one of the gas turbine plants. By selecting areas on the screen that have been highlighted, the student can move from one area of the simulation to another. Within each area, the operator is able to get part descriptions, read gauges, test parts, and repair or replace parts. Each area also includes context sensitive help. The Student, Instructor, and Expert models all have access to the state of the simulation.

The Student Model

The student model tracks the actions taken by the student. This model keeps track of the number of errors made and the number of times the student has requested advice from the expert model. Access to the simulation enables it to determine which parts of the turbine have been examined, tested, or replaced. This information is used by the instructor model and the expert model to generate advice.

The Instructor Model

This model examines each action taken by the student to determine whether that action is an error. For example, if the operator attempts to replace a part that has been tested and determined to be fully functional, the instructor model would inform the student that the action is inappropriate. Any errors of this kind are logged in the student model. In addition, the Instructor Model compares each action the student takes with the recommended action from the Expert Model. This comparison allows the Instructor Model to determine if the student is progressing toward the solution. If no progress is being made, the Instructor Model advises the student to consult with the Expert Model.

The Expert Model

The expert model generates advice for the student when the student requests it. This model is passive, in that it does not give advice unless the student specifically asks for it. When advice is requested, the expert model checks the student model to see what information the student has acquired and the actions that the student has taken. It then makes use of the SA°VANT job aid knowledge base to determine the next action that the student should take to solve the problem.

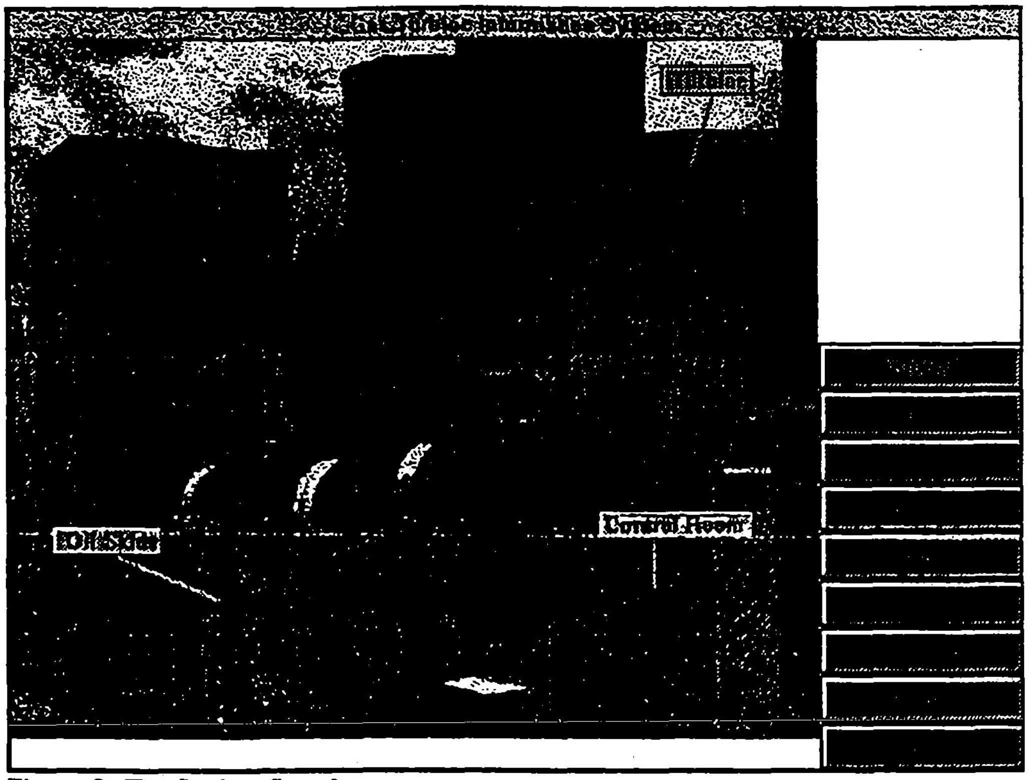


Figure 3: The Student Interface

The second part of the GTIS is the modified job aid. This consists of the SA°VANT system converted to function in the Windows environment. As stated above, the original SA°VANT interface used two CPUs. The first CPU handled the direct communication with the operator - receiving data, giving advice, supervising the speech synthesis, and recognizing voice commands. The second CPU displayed graphic images of the particular turbine part being examined. By moving the system to Windows, the need for the second CPU was eliminated. Graphic images can be displayed in another window on the same monitor that is displaying the advice, instead of using two monitors as was previously required.

Part 3: The Information Manager

The third part of the GTIS system is the information manager. The main element of this component is an on-line manual containing troubleshooting information used by the technicians when working on the job. This manual contains items such as alarm descriptions and computer access codes, and can be used both by the student when troubleshooting the simulated turbine, or by the technician in a real-time environment.

The operator also has access to information describing the parts of the turbine that make up the simulation. When the operator is examining a part, an "info" button is provided to gain access to a description of the part and its function. All the parts in the system are connected by hypertext links to enable the operator to learn of their relationships.

The final tool currently used by the GTIS information manager is digital video. Sequences of experienced technicians demonstrating troubleshooting techniques were videotaped. These sequences were then captured in digital format and incorporated into the system so that when an operator desires to test a part, he/she can view the video sequence describing that test.

It is intended that other information sources, such as operations operator manuals, will be available in addition to the troubleshooting manual. We are working on getting permission from the makers of the turbines to place their manuals on-line. When this occurs, the operator will have a powerful resource for information about the system.

REUSE OF THE SA°VANT KNOWLEDGE BASE

In developing the original SA°VANT system, a great deal of effort was invested in acquiring the knowledge to be used by the expert system. A panel of experts from several gas turbine sites was assembled to develop the rules to be used in the system. The first SA°VANT system had approximately forty-five-hundred rules compiled by this "round table" of experts. The development of these rules was both time-consuming and expensive (Quentin, 1991). Rather than trying to build a training system from scratch, EPRI decided the reuse the knowledge from SA°VANT as the basis for an ITS.

The Prototype

The approach to reusing of the knowledge base for the prototype GTIS was a simple one. The job aid knowledge was parsed "by hand" to extract sample problems which were then hard-coded into the software for the ITS. The simulation was then programmed to provide appropriate gauge readings and system settings for the gas turbine generator. Once this was completed, it was repeated for each additional problem added to the prototype. This process proved to be time-consuming in itself, as the knowledge base being parsed had several thousand rules combined into one ASCII file over two megabytes in length. It was clear that manual conversion of the knowledge base would not be appropriate.

Conversion of PML code.

The original knowledge base was written in PML, a high level, rule based description language developed by Honeywell (Cochran, 1991). This structure provides several classes of rules for building a rule based system. The classes of rules are as follows:

Tell Rule:	Gives a list of other rules which will be executed in order.

Announce Rule:	Provides for text output to the screen.
Record Rule:	Provides for text output to a disk file.
Ask Rule:	Displays a question and a list of possible answers to the screen, and then inputs the selection made into a variable.
If Rule:	Compares the value of one or more variables to constant values and then executes another rule based on that comparison.
Set Rule:	Sets a variable to a constant value.
When Rule:	Sequentially compares one or more variables to constant values and then executes another rule based on which of the comparisons yielded a "true" result.

Each rule involving textual display is able to display two text fields and a list of selections (if appropriate).

A sample "ask" rule appears in Figure 4. Note that the rule is broken down into several sections. The "action" line gives the name for the rule. The "heading" lines allow for the insertion of comments into the code. The "ask" keyword indicates that this rule is an "ask rule", and that the following symbol, "?FLAME-SD-MACH-SPD," is the variable name into which the response is to be placed. The text enclosed in the quotes is the information to be displayed to the operator. The keyword "explain" allows for additional information to be presented to assist the operator in understanding why the question is being asked. Following the text field is a list of choices to be shown to the operator and the corresponding values to be stored in the variable.

```
ACTION- CHECK-FLAME-SD-UNIT-SPD
 DOCUMENTATION- "created by JU at 19:14:38 4/30/89
edited by BULLEMER at 22:57:51 6/5/89"
 HEADING- "5111 CHECK-FLAME-SD-UNIT-SPD *#@*"
 ASK- ?FLAME-SD-MACH-SPD
(FORMAT NIL "
What was the speed of the unit at ~
the time of the flame shutdown?
")
 (EXPLAIN
(FORMAT NIL
 "
If you did not reach 1200 rpms, ~
the unit probably tripped after the ~
ignition sequence timer expired.

Note: 1200 rpm is 30% of full unit ~
speed. ~
"
 ))
 ("Unit speed < or = 1200 rpm" :Unit-speed—or---1200-rpm
 "Unit speed > 1200 rpm"  :Unit-speed---1200-rpm
 "Don't know unit speed"  :Don-t-know-unit-speed)
```

: **Sample "Ask" Rule**

Accessing a specific rule requires either a time consuming textual search, or an indexing scheme that indicates the rule's position in the file. If the latter scheme is used, any changes to the knowledge base would require recomputing the values in the index. It was decided that in the interest of data access and modification, the knowledge base would be converted to a database. This would allow for direct access to the rules based on the rule name.

To facilitate this conversion, the tools LEX (a lexical analyzer) and YACC (Yet Another Compiler Compiler) were used along with a C++ program written for this purpose. LEX was used to analyze the PML code and break each rule down into its respective parts. YACC was used to take the values for these parts and send them to the C++ code. This code in turn organized the rules and saved them into the database. For example, the LEX generated code would locate the keyword "ACTION" in the sample above. YACC would then parse the action name "CHECK_FLAME_SD_UNIT_SPD" and call a procedure in the C++ program which would make a new record in the database for the new action. The remaining fields in the database record would be filled as the rest of the rule is parsed.

Designing the database records involved tradeoffs between complexity, space, and operator requirements. The simplest design for the records would have been to create one record with a field for every possible kind of rule. However, this would have wasted a great deal of space, as most of the records have only a small subset of the total possible fields. A second option involved a more complex database structure which greatly reduced the disk space required to store the record. As this system was being developed, an additional operator requirement forced another change in the database structure. The operators responsible for maintaining and updating the PML code wanted to be able to edit the rules in their original form. As a result of this need, the current form of the database was developed. Each rule is stored separately in the database as a single field, which then has to be parsed using LEX and YACC at runtime to break it into its component parts.

In order to navigate through the rule base, the calling program only needs to initially know the name of the starting rule. It can then recall this rule from the database and begin to parse it. The starting rule is generally a "tell rule" which yields a stack of additional rules to be interpreted. The rule currently on the top of the stack is then loaded from the database and is parsed and executed in its turn.

Once the rules were converted to the database, a tool was written in C++ to allow for the editing of the text fields in the database. This tool allows the operators to maintain the knowledge base without having to manipulate the database directly, and provides both a graphical interface and a text-based interface to edit the PML code.

CONTEXT SWITCHING

The GTIS provides the operator the ability to switch at any time between the job aid and the tutor. This allows the operator to access the on-line manuals, parts descriptions, equipment descriptions and diagrams, and any other information that the system provides. Thus the operator can get any additional information about the gas turbine that may be required for troubleshooting. It also furnishes a way for the operator to investigate why the job aid has asked for the particular data that it needs. Such context switching allows the operator to learn about the gas turbine without interrupting the job aid.

CURRENT DEVELOPMENTS

There are several extensions and enhancements which are being added to the GTIS as a result of the latest round of developments.

The first extension to the system is increasing the coverage of the ITS. At this time, only a few of the failures covered by the SA°VANT system are handled by the ITS. Now that the knowledge base has been converted to the database format, additional failures are being incorporated into the GTIS. In addition, more digital video sequences are begin incorporated into the system.

As the system continues to develop, we will be adding tools for creating and editing new turbine information systems. EPRI has generated knowledge bases on several different styles of gas turbine generators. The GTIS only addresses one of these knowledge bases at this time. With these conversion tools, all of the knowledge bases can be converted to database format and added to the GTIS.

A number of hardware improvements have been made to the system. The system has been modified so that it will function when running on a pen-based computer platform. The current SA°VANT system is a large unit, which cannot be easily carried from one location in the plant to another. This isä a significant problem if the unit is being used as a job aid. The technician must go to the console to get instructions, then leave the console to carry out the instructions. Once the operator has performed the required task, he/she must then return to the console and repeat the process. The current pen-based computer models are much smaller - about the size of a paper notebook. At less than four pounds, they are also relatively light. Porting the GTIS to a pen-based computer will allow the operator to carry the computer while performing maintenance. He/she will no longer have to go continually back and forth between the turbine and the computer, thus reducing the amount of time required to complete each task.

The enhanced portability of the pen based system has benefits for training as well. By bringing the tutor out to the turbine site, the student will be able to more easily relate the task being shown on the computer to the task required on the turbine.

CONCLUSIONS

The GTIS employs the same knowledge for use in both training and job aiding. As tools for knowledge base management improve, the ability to revise existing job aiding knowledge for other tasks becomes feasible. By converting these knowledge bases instead of recreating them, developers will realize a significant savings both in cost and time in the development of training systems. By consolidating information into a common knowledge base source for the GTIS, it should be feasible to create an overall system that appears "seamless" to the user. That is, the user should be able to move freely among the three elements in the system (i.e. documentation, diagnostics, and training) as needed, without necessarily crossing "hard system boundaries."

REFERENCES

Quentin, G.H., and Dolbec, A.C. (1991). "Expert Systems for Peaking Combustion Turbine Plants," ASME, Budapest, Hungary

Polson, M.C., and Richardson, J.J. (Eds.) (1988). Foundations of Intelligent Tutoring Systems. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Johnson, W.B. (1988a). Developing expert system knowledge bases for technical training. In L.D. Massey, J. Psotka, and S.A. Mutter (Eds.), Intelligent Tutoring Systems: Lessons Learned (pp 83-92). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Cochran, T., Bloom, C., and Bullemer, P. (1991). Increasing End User Acceptance of Expert Systems by Using Multiple Experts. In K. McGraw and C. Westphal (Eds.), Readings in Knowledge Acquisition (pp 73-89). London, England: Ellis Horwood

N96-14991

FTDD973: A Multimedia Knowledge-Based System & Methodology for Operator Training & Diagnostics

Amir Hekmatpour, Gary Brown*, Randy Brault+, Greg Bowen#, Larry Grant&, Eric Millham
Knowledge Based Systems Development, *Hot Process, +Ion Implant, #Thin Films, &Multimedia Lab.
B21/975-1, IBM, Essex Junction, Vermont 05452
AMIR@BTVLABVM.VNET.IBM.COM

Abstract

FTDD973 (973 Fabricator Training, Documentation, and Diagnostics) is an interactive multimedia knowledge based system and methodology for computer-aided training and certification of operators, as well as tool and process diagnostics in IBM's CMOS SGP fabrication line (building 973). FTDD973 is an example of what can be achieved with modern multimedia workstations. Knowledge-based systems, hypertext, hypergraphics, high resolution images, audio, motion video, and animation are technologies that in synergy can be far more useful than each by itself. FTDD973's modular and object-oriented architecture is also an example of how improvements in software engineering are finally making it possible to combine many software modules into one application. FTDD973 is developed in ExperMedia/2; an OS/2 multimedia expert system shell for domain experts.

1) Introduction

Existing mass production approaches to workplace training no longer meet the new international standards for industrial competitiveness. The standards of the "new economy" include quality, variety, customization, convenience, timeliness, and continuous innovation. With multimedia KBS technology application, we can raise the level of manufacturing operator expertise across the board and enhance the operator's job to include more diagnostics. This will pay off in increasing productivity (proficiency across multiple operations), decreased scrap, and reduced reaction time. One of the biggest challenges in semiconductor manufacturing is to cut our "mean time off line" and improve our productivity per person. FTDD973 is directly addressing and benefiting both.

A major departure of our approach from previously reported multimedia systems for semiconductor manufacturing is the seamless integration of training, certification, and on-line documentation with knowledge based diagnostics. In addition, existing multimedia training methodologies usually require large initial course preparation investment and are suitable for relatively stable subject matters. Such applications usually require specialized and often expensive hardware and software. FTDD973 is developed in ExperMedia/2 and resides entirely on hard disk and can be used in stand-alone or network configuration. It provides modular, easy to upgrade, customizable training and diagnostic modules on standard hardware (386PC, 6Meg RAM, 70 Meg HD) and software (OS/2 2.0). ExperMedia/2 is an OS/2 multimedia expert system shell for domain experts. It is based on standard OS/2 features and a set of novel utilities.

1.1) Motivation

The IBM Vermont plant manufactures a wide range of Integrated Circuit (IC) products (RAM, ROM, ASIC, Processors, Logic, EPROM, ...) fabricated on three full scale production lines. The 973 fabricator is one of IBM's advanced CMOS manufacturing lines. It is comprised of over 40 tool groups, 175 processes across 7 different technologies and is serviced by over 200 operators, production technicians (PT), and maintenance technicians (MT). FTDD973 started as a diagnostic assistant for the oxide growth process, one of the major tasks in the Hot Process area (one of the areas currently supported by FTDD973). A closer study of the oxide growth operation in the scope of CIM and IBM's six sigma manufacturing concepts and requirements (zero defect, paperless workplace, improved cost, quality, and turn around time) revealed that the biggest challenge was to reduce "mean time off line" and improve operator productivity. In order to reduce "mean time off line", we needed to put first call diagnostics in the hands of the individuals closest to the process (operators). This cuts down on reaction time because the operator is physically located in the area. On the other hand, to increase operator proficiency across multiple operations and decrease scrap, it is essential to increase operator expertise across the board (efficient training and information dissemination).

Our initial feasibility and requirements analysis, and interviews with manufacturing personnel, revealed that the four major components of the operator's workplace: *training, certification, diagnostics, and documentation* needed to be integrated taking advantage of the multimedia capability of the workstations (PS2), multi-tasking capabilities of the operating system (OS/2) and AI techniques proved successful in our previous diagnostic applications [Hekmatpour91] [Hekmatpour93]. A decision was made to extend FTDD973's architecture and initial intent to provide a complete intelligent and efficient workplace for all aspects of the 973 fabricator operation (Figure 1).

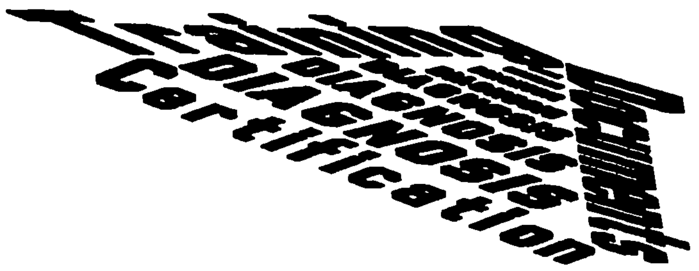


Figure 1. FTDD973's Integrated Workplace Methodology

1.2) Architecture

FTDD973 is developed based on an object–oriented modular knowledge structure, comprised of a collection of interacting and cooperating modules arranged around the control and management knowledge bases (CMKB), as shown in Figure 2. Each module represents an autonomous and self sufficient utility/task. Training modules provide interactive multimedia training for the 973 fabricator tool and process operation, as well as orientation for the area. Diagnostic modules provide interactive tool and process diagnosis. Documentation modules provide on–line hypermedia documents covering all necessary information on tool, process, process parameters, logistics, and various other applications and databases which the operator comes in contact with. Certification modules provide an interactive test, evaluation, and feedback environment. The user profile database includes various information on the manufacturing personnel in each area. For example, it includes employee number, name, department, tools and processes for which they are certified, date certified, level certified (beginner, novice, intermediate, expert), any special expertise, and authority level (system administrator, production technician, maintenance technician, operator, ...). Any module can query the user profile database directly, but the updates and modifications are managed by CMKB or directly via GUI (Graphical User Interface) by authorized personnel (System administrator).

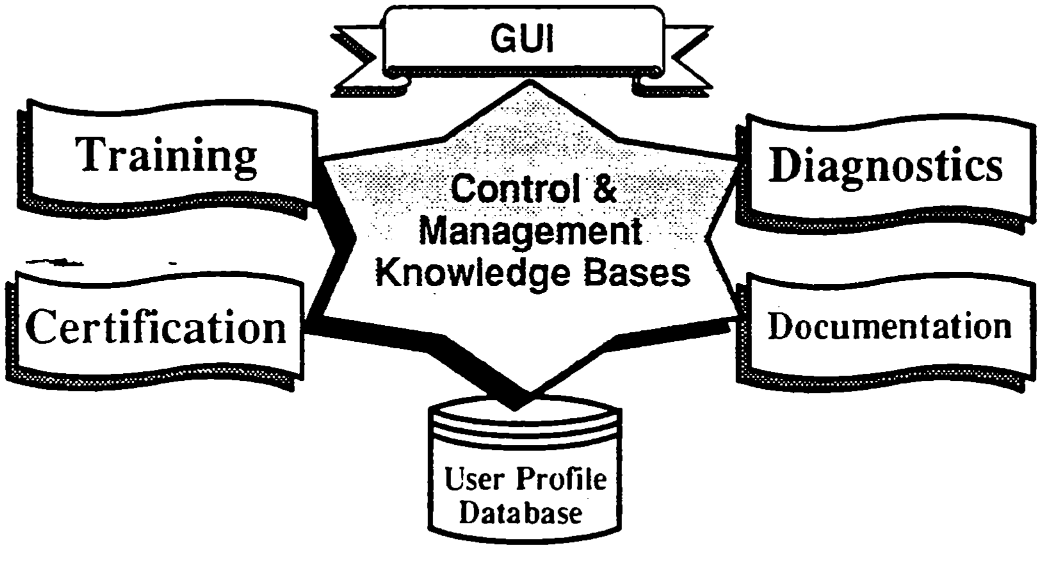


Figure 2. FTDD973's Architecture

1.3) System Hierarchy

Knowledge bases and multimedia utilities are hierarchically organized according to the expert's mental model (how the expert conceptualizes the diagnostic task hierarchy and its relationship to training, certification, and documentations). FTDD973 currently consists of 10 diagnostic knowledge bases, 3 hypertext documents, 45 Multimedia/Hypermedia training documents, and 60 manufacturing procedure documents (Figure 3).

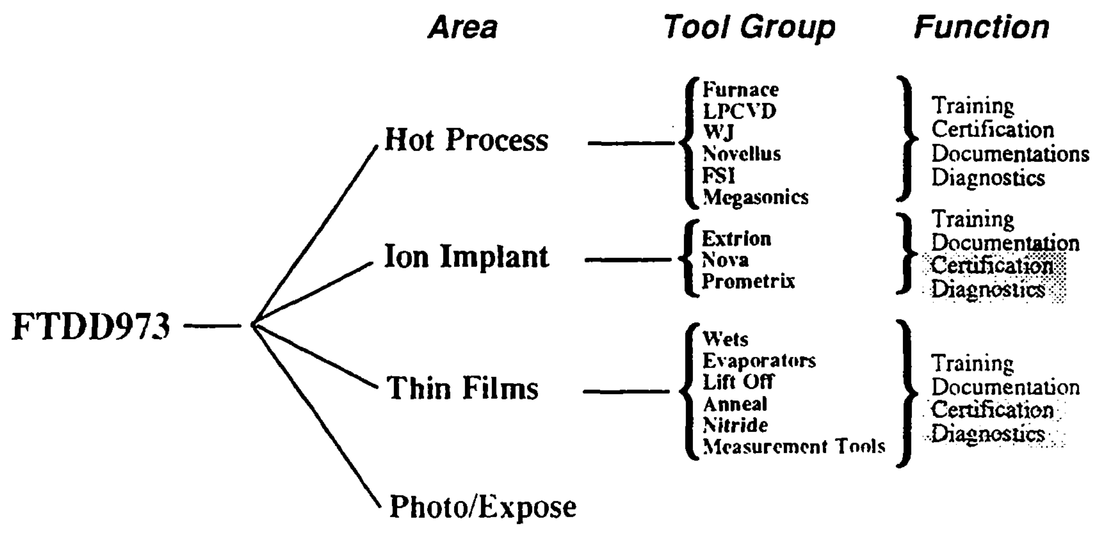


Figure 3. FTDD973 System Hierarchy

2) Interactive Multimedia–Based Training and Education

Traditional Computer Based Training (CBT) and computer–assisted instruction (CAI) teach a subject by offering corrective feedback based on a large, pre–stored set of problems, solutions and associated remedial advice. The most noticeable problem with CBT and CAI is the amount of time it takes to produce effective courseware as all possibilities need to be anticipated in advance by the author. Some Intelligent Tutoring Systems (ITS) have addressed this problem by providing an environment where problems are generated automatically. Such "generative" systems need only be given general teaching strategies and they could produce a large number of interactions without all of them being explicitly programmed in advanced. However, most of these systems lack a clear representation of the knowledge that they are teaching. In such systems there is a mismatch between the program's internal process and those of the trainee's cognitive processes. The trainee using CBT or ITS is usually denied any chance of using his/her initiative in guiding the learning process. The CBT and CAI strategy is "learning by being told". However, research has shown that "learning by discovery" and "learning–by–doing" is in many cases a more effective strategy [Kass91] [Yazdani88]. The main goal in our training methodology has been to reduce intimidation and to put the trainee in control of learning, where it belongs. Instead of sitting in a classroom listening to an instructor, or following an operator on the manufacturing floor (who is usually fighting fires and the last thing on his/her mind is training), the trainees work individually on computers that combine text, high resolution color images, graphics, audio, animation, and motion video. This environment is characterized by student sequencing through course material in a self–paced manner, while being monitored by the system. The goals in developing FTDD973's training methodology can be summarized as:

- *Shorten the training process.*
- *Improve the training quality.*
- *Provide consistent training across the board.*
- *Produce modular and portable training courseware. Since IBM is a member of the Interactive Multimedia Association (IMA), we have adapted the general principle of the IMA's practices for Multimedia Portability [IMA1] which is also known as MIL–STD–1379D Appendix D (for portable courseware).*
- *Effectively deliver and monitor the training objectives.*
- *Enable domain experts to easily develop and modify multimedia training courseware by providing them with architecture, methodology, and necessary tools and techniques*

2.1) Why Multimedia Training

The IBM Vermont plant manufactures a wide range of Integrated Circuit (IC) products (RAM, ROM, ASIC, Processors, Logic, EPROM, ...) fabricated on three full scale production lines (different technologies). The 973 fabricator is one of IBM's advanced CMOS manufacturing lines. It is serviced by several hundred operators, production technicians (PT), and maintenance technicians (MT). Currently, a multitude of methods are used for educating and assisting the manufacturing personnel. These methods include informal, unstructured training sessions, printed "in–house" manuals, sit–down classroom courses, and walk–thru orientations, to name a few. On the other hand, in our manufacturing environment, the student population grows with every group of new employees (temporary or permanent). This student group is large and decentralized. They cannot all be released from the manufacturing line simultaneously to take a standard training course.

A majority of the process engineers, process technicians, and senior operators interviewed at the IBM Vermont plant, emphasized the fact that their major concern was the heterogeneous work environment they had to deal with. In addition, it was emphasized that the current training and certification methodology was inefficient and time consuming due to usual operator turn around and frequent changes in the process and tools. In most cases the operators are trained by current operators and process technicians who have their own styles and biases. In addition, most of the information required for their regular activities are textual documents (softcopy or hardcopy).

Interactive multimedia training and certification offers consistent presentation of the subject matter, on a flexible 24 hour per day seven day a week schedule. While, admittedly, the use of interactive multimedia training and certification will never totally replace conventional methods of instruction, our experience has shown tremendous benefits in quality and the overall cost of training. The subject matter delivered to the employee is guaranteed consistent when such a methodology is used. We avoid the "Monday Morning Syndrome" when some of the subject matter presented by a line technician to new hires might be a little sketchy or missing altogether. Using such an on-line computer-based training methodology also eliminates the back-level problem. Currently, using printed documents, we're never sure if the line operator is using the most current revision or whether an operator's training had covered the latest version of process or tool upgrades. By making the information available "on-line", and integrating training, certification, diagnostics, and documentation, we have better control over what information is being used and what is needed to be disseminated.

Most importantly, interactive multimedia training works. This technology goes beyond the point of being user-friendly to being what Sullivan [Sullivan 91] calls USER-SEDUCTIVE. People like to use this technology. This has been shown in numerous studies and we see it everyday from our contacts with existing and new users here at the Vermont manufacturing plant. Thus, to use the words of Ben Franklin: "You tell me and I forget, You teach me and I remember, You involve me and I learn." We can certainly involve our operators through the use of interactive multimedia training and certification and take one step closer to being a Six Sigma, World-Class semiconductor manufacturing.

2.2) Multi-level Student Model (Beginner, Novice, Intermediate, Expert)

Training and certification methodologies implemented in manufacturing enterprises are usually based on a single and simple student model. Regardless of the familiarity or lack of familiarity of the students with the business process and equipment, all go through the same training and certification process. In addition, the certification procedure is usually conducted orally in a walk-thru show and tell fashion and is subject to the trainer's judgment and biases. A new hire, who has never worked in the semiconductor manufacturing environment, is given the same training material as an operator who has many years of experience in semiconductor manufacturing and has been transferred to this new assignment from another line. In addition, there is no formal methodology for increasing the responsibility of an operator as the training proceeds. To address this problem, we developed a new multi-level student model. This new model advances the student rank as the training progresses and as the student improves his/her certification test score.

Expert PTs-MEs and operators were interviewed to identify efficient methodologies for training the manufacturing personnel in their area. In other words, how should we present and manage the training material to students of varying background and expertise? Four levels were identified; *Beginner*, *Novice*, *Intermediate*, and *Expert* (Figure 4). An operator rated as a *Beginner*, would require the most assistance and step by step guidance, Whereas an *Expert* would only need to be able to locate the information rapidly and efficiently as the need arises. A user is assigned an initial level by the area PT. The user advances to the next level (increasing responsibility and access to information) as he/she passes the certification procedure for that level. As the user level is advanced, the detailed procedural information is reduced, but the conceptual and deep knowledge of the process and the tool is made available. A *Novice* student knows the steps involved in the operation, an *Intermediate* student should know the cardinality and temporal relationship and importance of the steps. An *Expert* user would also understand the taxonomy of the process steps and tasks.

Such a multi-level student model reduces the danger of operators getting involved in potentially dangerous or destructive tasks which they may not have had adequate training and preparation for. In fact, a large percentage of "wafer scrap and reworks" is traced back to inaccurate or incomplete processing. In the previous operator training methodology, once an operator was certified, that operator could have been assigned to process products, although he/she may not have had any training in process or tool trouble-shooting. In most cases, ignoring initial signs of problems, or not being able to respond to problems quickly, could be very costly and result in the shut down of a tool or a process. On the other hand, in some situations, initiatives taken by operators who have not been adequately trained and informed of their responsibilities could result in scrap, rework, damaged tools (e.g. broken furnace tube) or an out of spec process. In our multi-level methodology, CMKB keeps track of the user level and last certification date, then decides what tasks could be performed by the operator, and what new training or certification is required at each stage.

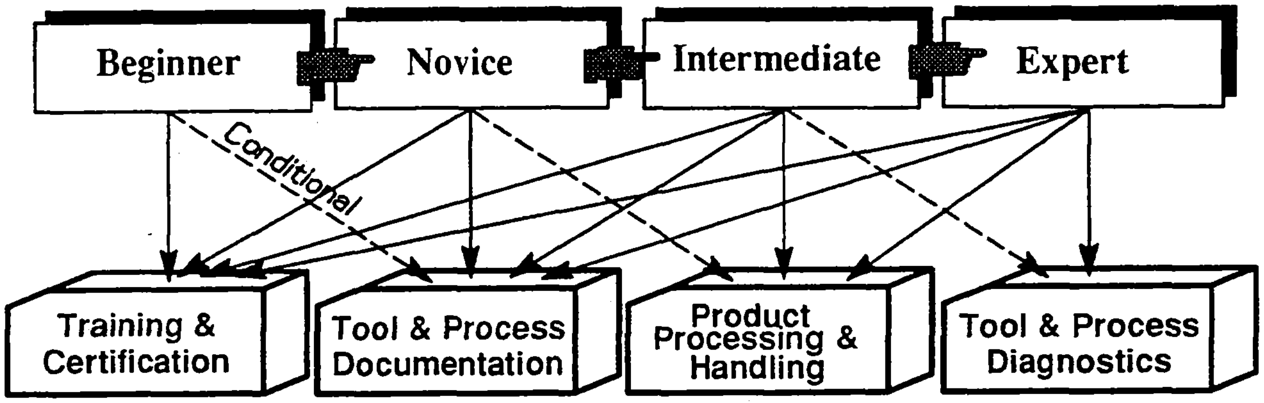

Beginner
Novice
Intermediate
Expert
Conditional
Training & Certification
Tool & Process Documentation
Product Processing & Handling
Tool & Process Diagnostics

Figure 4. Multi-level Training and Certification Model

2.2.1) User Profile Management

User name, department, passwords, user level, process and tools certified for, certification dates, authorities, and specialized expertise are maintained in the user profile database for each user. User level is identified and assigned to the global environment variable USER_LEVEL by FTDD973 at logon time, but may be changed at anytime during the session (e.g. after the operator takes a certification test). Global access to training, documentation, logistics, and diagnostics is controlled by FTDD973, whereas local access to specific sections of training modules or on-line documents are controlled via local environment variables which are maintained by the corresponding module. The value of these local environment variables determines which sections of a module can be accessed by a user and which sections should be hidden.

2.3) Active vs. Passive Training Modules

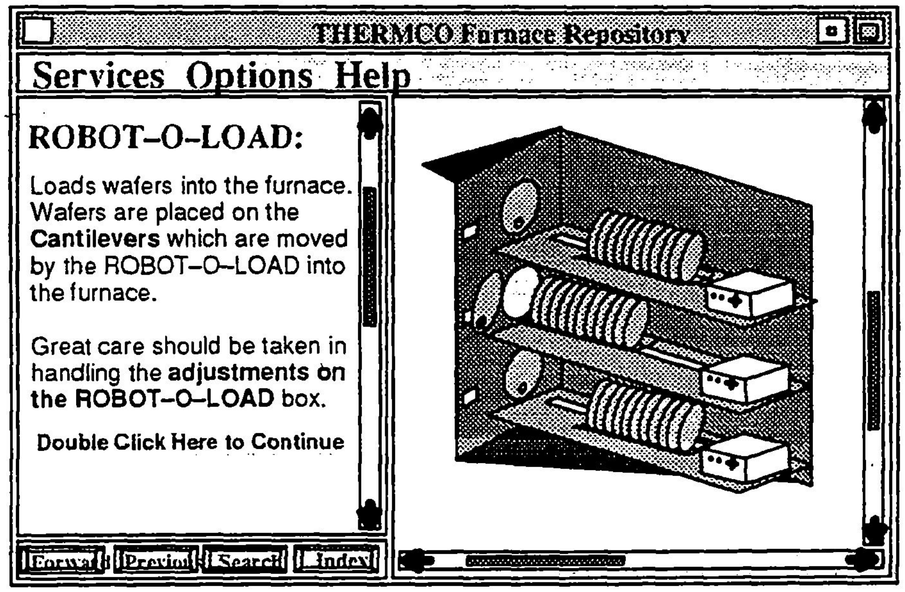
Active training modules provide interactive step by step guidance and monitor user responses and adjust/react accordingly. These modules usually cover the more complex scenarios, where the system serves as an active participant in the process, analyzes user responses and formulates a corresponding action based on the response and the overall goal. Active training modules could ask trainees to perform specific tasks, examine tool components, review process procedures or make specific tool/process/product measurements before continuing to the next step. These modules are very similar to FTDD973's diagnostic modules and are implemented as CATs (an acyclic directed decision graph). Active training modules are usually used for beginner and novice training, where the user needs step by step guidance, monitoring, and feedback.

Passive training modules can also provide step by step instructions and procedures, but they neither enforce it, nor monitor the progress. Such modules are suitable for intermediate and expert users who may not require step by step control and guidance. Whenever multimedia and hypermedia modules are accessed directly, they provide passive training, whereas if these are presented via diagnostic knowledge bases or the CMKB, then active training is performed.

2.4) Partitioning of Multimedia Modules into Logical Pages

Coordinating text, graphics, and images in a multimedia environment is very important in effective presentation and management of the subject matter [Hekmatpour92][Feiner90]. Some researchers have investigated the automatic generation of coordinated multimedia [Feiner91]. Our methodology is based on a pre-defined presentation scenario and is implemented in one of the pre-defined templates. Multimedia modules are partitioned into "logical pages" (Figure 5 & 6). A logical page (Figure 5) consists of a set of images (still, video, animation, graphics)) and all their associated description (text and audio). In other words, a logical page is the pre-defined collection of all related information (text, graphics, images, audio instruction, animation, video clips) which the trainee should/could review when studying the subject matter covered in that page. The appearance, format, and access to logical pages are fixed and consistent throughout the system. Logical pages are related to each other via hypertext and hypergraphic links.

ORIGINAL PAGE IS
OF POOR QUALITY



ROBOT-O-LOAD:

Loads wafers into the furnace. Wafers are placed on the **Cantilevers** which are moved by the ROBOT-O-LOAD into the furnace.

Great care should be taken in handling the **adjustments on the ROBOT-O-LOAD** box.

Double Click Here to Continue

Figure 5. An Example of Multimedia Logical Page

2.5) Logical Page Templates

To accommodate various situations and to maintain consistency among modules, a set of "Logical Page Templates" are defined (Figure 6) and adhered to throughout FTDD973. These templates provide consistent facilities for embedding various multimedia and hypermedia utilities and functions in applications. The logical page templates facilitate sharing of multimedia courseware between the areas covered by FTDD973 and between other systems planned for other manufacturing lines. In addition, multimedia documents can be shared among projects, maintaining a consistent user interaction and presentation. For example, the "safety and hazardous material handling" multimedia training document developed in Ion Implantation area can be easily used by other areas within the 973 fabricator.

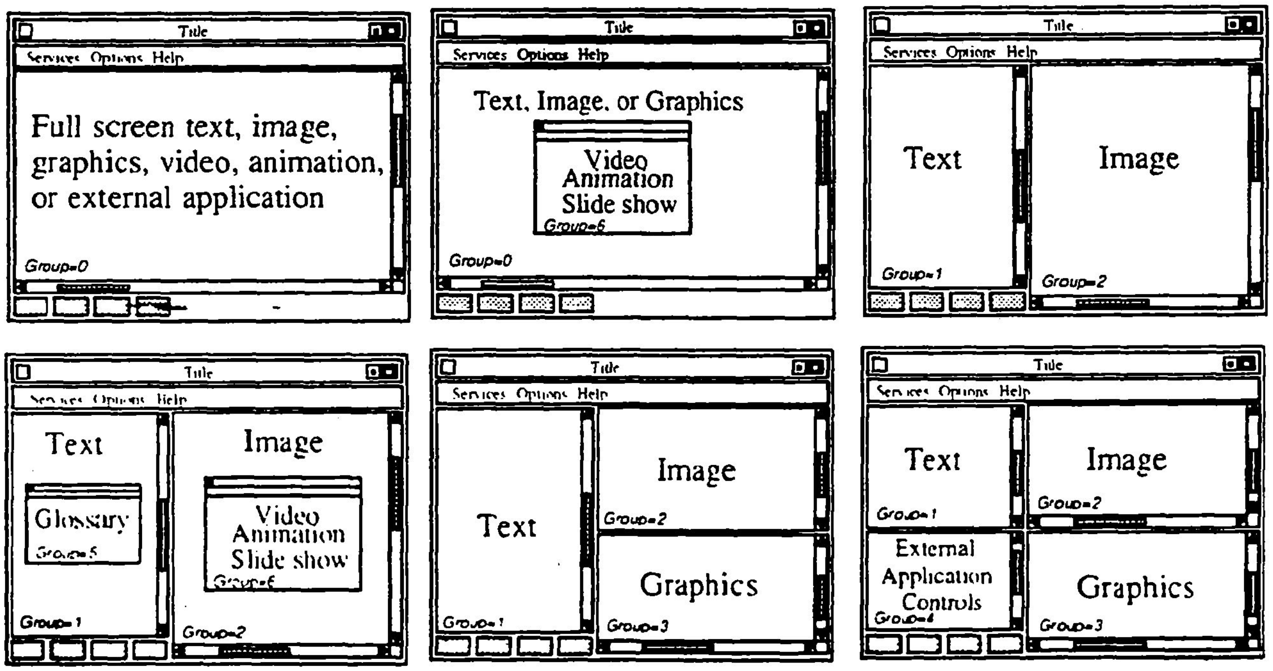


Figure 6. A Partial List of Multimedia Logical Page Templates

2.6) Customizing the Logical Page

In addition to a variety of logical page templates, most of the major attributes of a logical page can be customized. For example, page background color, highlight color, page size and position, partition ratios, text style and font size, pull down menus and menu items, system menu items, push buttons, vertical and horizontal scrolls, min/max icons and window titles. Some of these are controlled via the tagging language used to define the logical page templates (e.g. *:font facename=Courier size=13x8.*) and some are controlled by utilities provided by ExperMedia/2.

3) Hypermedia Documentation

3.1) Area and Tool Orientation

The first time a user logs in, he/she is presented with an introduction to the system and a tutorial on hypermedia and multimedia. Once the user has finished reviewing the introduction, he/she is then presented with the list of orientations to be reviewed. These introduce the user to functions performed in the area, the tools, terminologies, documents, computer systems, data collection procedures and the overall view of how all the pieces fit together in that area, as well as how the area fits into the overall semiconductor manufacturing process in general and that specific fabrication line (i.e. 973 fabricator) in particular. Orientation modules utilize hypertext, hypegraphic, audio, motion video, and animation. On average, it takes new operators about 2 hours to review all orientations for their area. Most of the systems developed so far do not enforce any time limitations but they keep track of documents reviewed and time spent on each. An operator may review a document as many times as necessary and return to document at any time.

3.2) Hypergraphic Repository

Manuals (maintenance, diagnostic, and operation) are usually comprised of textual descriptions, pictures of components and schematic diagrams. The quality of these pictures (black & white) is often not good. In some cases these documents are outdated or the pictures don't match the component on the tool (either due to local or vendor upgrades). Technicians and operators complain about having to go back and forth between several manuals to get all the information about a component. In addition, they complain that such manuals are hard to comprehend and are boring. To address this dilemma, we have developed hypergraphic repositories. The repositories utilize hypertext and hypergraphic to provide an efficient and interesting media for presenting various operation and technical information about tools and processes. Coordinated text and graphics has proved valuable in training and explanation-based systems [Feiner91][Maybury91]. The user can query the repository based on component name, operation, component location or function. The repositories utilize the capabilities of hypertext (i.e. links, table of contents, topic and keyword search and margin notes) as well as hypergraphic. Hypergraphic allows the user to obtain specific type of information about a component by clicking on the component to bring up its hot-spot pop-up menu. Then he/she may review the component's textual description (name, symbol, function, part #), view a close-up picture, listen to any associated audio instruction or description, or view an animation or video by selecting the appropriate hot-spot function (Figure 6). Hot-spots can be linked to any other part of the repository (text, images, graphics) or external applications such as audio, animation, motion video, diagnostic modules, certification, or other multimedia documents (Figure 7). Hypergraphic capability also allows navigation into composite components (black boxes). For example, a user can click on a panel door to open the panel and get a view of what is inside. The user may then click on a component inside the panel to get a close-up view, or open up a complex component to review its sub-components.

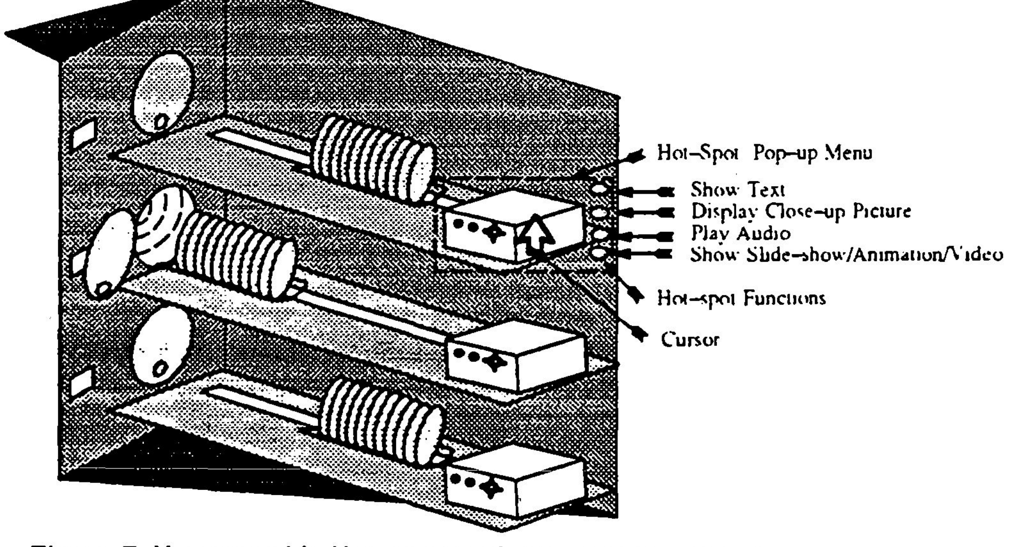


Figure 7. Hypergraphic Hot-spot and Hot-spot Functions

Such repositories have proven to be very useful and are frequently used by *Beginner* and *Novice* operators and technicians during training. It is an efficient, effective, and safe medium for providing orientation on complex semiconductor manufacturing tools without exposing the trainee to hazardous materials, heat, radiation, or poisonous gases. It enables learning by discovery and navigation without being watched by or bothering the trainer. It also eliminates the need for shutting down a tool to show the internal components to trainees. It has allowed the trainees to perform what-if simulations and investigate the internal structure of the tools at their own pace. In addition, the repository serves as a central place for documenting upgrades, component changes, and alternatives for a component.

4) Certification

The certification process is interactive and cooperative and is geared towards process and tool operator certification in general, and semiconductor manufacturing operator certification in particular. The certification procedures test the trainee's progress, track his/her performance and help the trainee to concentrate on the problem area. The feedback and tracking was developed in response to the concerns with the existing certification and training procedures which lacked a meaningful and unintimidating feedback mechanism. In addition, to reduce intimidation, enhance learning and increase the trainee involvement, we expanded the certification beyond a simple pass or fail. Certification modules can be considered an extension of the training by providing meaningful feedback, suggesting topics to be reviewed, and providing general comments as well constructive criticism.

A new user needs a password and a valid IBM employee serial number to take a certification test. New users are assigned USER_LEVEL="Beginner", unless a different level is assigned to them by the PT or MT responsible for the area. Once a trainee has reviewed all the required materials and procedures for his/her area, tool group, and level, he/she will be given an on-line certification test. The test is usually comprised of about 20-50 questions (multiple choice and True/False) which may be answered in any order. The trainee can review an up-to-the -point status summary which shows the user responses to questions and whether a question has not been visited yet (Not Answered) or no answer has been selected for it yet (skip). Once the test is turned in, the CMKB evaluates the responses and generates a report. For each question answered incorrectly, the system lists topics, documents, or training modules to be reviewed.

5) Knowledge-Based Diagnosis

CIM (Computer-Integrated Manufacturing) is viewed as an emerging technology in the domain of manufacturing. The concept of CIM is that the whole performs better than the sum of individuals [Vail88]. In order to make the system run smoothly with minimum delay (continuous flow manufacturing), it is necessary to have a diagnostic system for discovering any cause of a system failure. It is desirable that this diagnostic system is capable of performing its task as fast as possible, since the duration of the system's breakdown is very closely related to productivity and cost. Therefore, there is a need for intelligent diagnostic systems. Diagnosis is also recognized as one of the major tasks in CIM [Alexander89]. Furthermore, in AI and expert systems, diagnosis has been given more attention in recent years, including trouble-shooting in electronic circuits [de Kleer87] and medical diagnosis [Gordon85][Jamieson90].

There have ~~been~~ a variety of approaches taken by various researchers in an attempt to understand methods for creating intelligent diagnostic systems. These range from shallow reasoning using compiled rules [Shortliffe76] to model-based ("deep") systems that reason by exploiting causal, structural, and functional relationships [de Kleer87]. Some acknowledge combining shallow and deep knowledge [Smith89][Sticklen87]. It is hypothesized by some [Gomez81] that the use of "deep" representations of entities to be diagnosed is superior to using empirical knowledge about associations between malfunctioning parts of an entity and symptoms. The rationale is that one cannot exhaustively catalog all such associations; without such a catalog, a heuristic-based diagnostic system becomes brittle and fails when presented with a case that it does not understand. On the other hand, deep knowledge representation is based on models that are difficult to construct-especially models that exhibit the technological intent of the designer. Further, it is unlikely that models will mirror the failure behavior of entities of any complexity, particularly with regard to providing information about multiple perspectives [Bourne91]. Moreover, models are likely to be domain-specific and only some fraction of knowledge will be transferable from one diagnostic system's knowledge base to another. It is also recognized that rule-based systems become increasingly difficult to understand and maintain, as the number of rules grow. While a reasonable rule-based expert system shell can assist a domain expert in formulating cause-and-effect rules, a collection of such rules typically will not function as an expert system, except for the most simple cases. To overcome these limitations, some expert system shells allow the encoding of strategic and object-level knowledge as meta-rules. This however requires extensive knowledge of the programming paradigm and the development environment. Another class of tools provide search algorithm for a flat problem-space representation. Although the problem representation is simplified, the search complexity for a problem of solution length L, and search space branching factor of B, has the worst-case complexity $O(B^L)$. Given the above arguments, what is the solution?

Bourne et al. [Bourne91] suggest that it is actually more fruitful in certain domains to concentrate on constructing models of belief organization for diagnosis than on models of physical entities. The concept of belief potentially has a wider scope than explicitly defined knowledge [Rapaport86]. Bourne et al. propose a method of organization of beliefs about diagnostic problems that provides explicit belief organization with implicit organization of knowledge about physical device characteristics, functionality, and behavior. The method is claimed to provide reasoning about belief among alternatives, is extensible, and can be easily scaled up to large problems. Further, it is asserted that belief manipulation coupled with information about fault history, and symptoms is sufficient to secure good diagnostic results.

5.1) Knowledge Base Architecture

Our knowledge-base architecture is an extension of the Bourne et al. proposal whereby, the expert's knowledge about the behavior of the environment is represented as a Behavioral Hierarchy Knowledge Base (BHKB). Next, a number of knowledge bases representing the expert's knowledge about the physical and structural hierarchy of the environment, called Structural Hierarchy Knowledge Base (SHKB) are attached to each terminal node of BHKB. Finally, the expert's knowledge about the association between symptoms and the causes and the functional characteristics of the environment are represented as branches of the SHKB, implemented as Condition Action Trees (CAT), as shown in Figure 8. Representation of knowledge as a hierarchy of CATs, rather than rules, has a three fold advantage. First of all, system performance and incremental expansion is improved, by eliminating the rule-base maintenance and rule interference. Secondly, decomposing the problem into functional CATs provides access to intermediate states, thereby reducing the search space [Minsky63][Newell62]. Thirdly, such a knowledge representation based on the cause-and-effect networks of association is consistent with the experts mental model of the environment and the reasoning process. Overall, the development and maintenance effort, as well as the predictability of the system behavior, independent of the amount of knowledge added into the system, is improved.

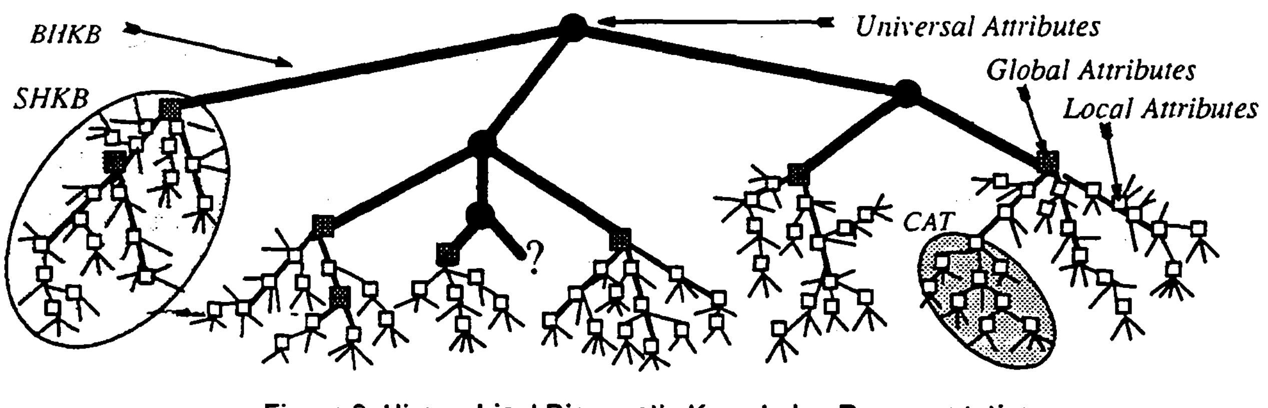


Figure 8. Hierarchical Diagnostic Knowledge Representation

Diagnostic flowcharts are generated for problems according to our knowledge engineering methodology and guidelines. Each flowchart is mapped to a knowledge base, implemented as a CAT. Each node in a CAT may have one or many possible branches associated with it. Each node has three explicit attributes assigned to it. Node strength (NS) represents the expert's belief that the node would be the proper diagnosis (option, symptom) for the parent node ($\sum NS = 1$, for all children of a node). Node cost (NC) represents the relative cost (dollar, time, man-power, resources) of performing, testing, and/or verifying the conditions/tests/actions described at the node ($0 \leq NC \leq 1$). And finally node rank (NR), which reflects the most natural, useful, and meaningful ordering of all nodes in a group, when presenting them to the user (NR=1,2, ... ,99).

To assure the independence and self sufficiency of each knowledge base, a set of universal attributes, common to all knowledge modules are dynamically loaded at the root node of the BHKB. These include tool id, directories, libraries, maintenance personnel, application support personnel, user preferences and system configurations. Universal attributes are available to all knowledge modules, CATs, menus, screens, and windows throughout the session. Attributes specific to a CAT (global) are loaded when it is put on the agenda and are therefore available to all services and diagnostic modules called by this module, as long as the module is active. Global attributes are updated whenever a CAT is added to or deleted from the agenda. Local attributes are valid as long as the node to which they are assigned is active.

6) Summary and Future Plans

Although FTDD973 is only a few months into its operation, its impact on the training and certification business practices are very encouraging. The preliminary results show that a technical and dynamic environment such as 973 fabricator that involves a great amounts of specification and procedure is conductive to multimedia and hypermedia technology. In addition, intelligent management and control of the training, certification, and diagnostics has proved to be very productive and have been highly praised. We view the current FTDD973 as an opportunity for further introduction of ICAT and VET technology into the manufacturing workplace. We are currently investigating use of VR for tool diagnosis, as well as 3D SEM display for defect analysis and classification.

7) Acknowledgments

The authors acknowledge the continued support from the 973 fabricator management (Scott Kramer, Scott Fullerton, Gary Moore, Mike Renaudette, and Delores Pavlik) and KBSD group management (Mike Potter).

8) References

[Alexander89] Alexander S. M., Graham J. H. and Vaidya C. M., "Issues in the diagnosis of manufacturing systems," AAAI Spring Symposium Series: AI in Manufacturing, Stanford University, Mar. 28–30, pp.1–5, 1989.

[Bourne91] Bourne J. R., et al, "Organizing and Understanding Beliefs in Advice-Giving Diagnostic Systems," IEEE Trans. on Knowledge and data Engineering, Vol. 3, No. 3, pp. 269–280, Sept. 1991.

[de Kleer87] de Kleer J. and Williams B.C., "Diagnosing multiple faults," Artificial Intelligence 32, pp. 97–130, 1987.

[Feiner90] Feiner S. and McKeown K., "Coordinating Text and Graphics in Explanation Generation," Proc. AAAI-90, pp. 442–449, Boston, MA, July 29–August 3, 1990.

[Feiner91] Feiner S. and McKeown K., "Automating the Generation of Coordinated Multimedia Explanation," IEEE Computer 24(10), pp. 33–41, October 91.

[Gomez81] Gomez F. and Chandrasekaran B, "Knowledge organization and distribution for medical diagnosis," IEEE Trans. Syst., Man, Cybern., vol. SMC-11, Jan. 1981.

[Gordon85] Gordon and Shortliffe E. H., "A method for managing evidential reckoning in a hierarchical hypothesis space," Artificial Intelligence, 26, pp. 323–357, 1985.

[Hekmatpour91] A. Hekmatpour, A. Orailoglu, and Paul Chau, "Hierarchical Modeling of the VLSI Design Process," IEEE EXPERT, 6(2):56–70, April 1991.

[Hekmatpour92] A. Hekmatpour, C. Elkan, "A Multimedia Expert System for Wafer Polisher Maintenance," Technical Report CS-92-257, Dept. of Computer Science and Engineering, University of California, San Diego, August 1992.

[Hekmatpour93] A. Hekmatpour, C. Elkan, "Categorization-based Diagnostic Problem Solving in the VLSI design Domain," Proc. of IEEE Ninth Conference on Artificial Intelligence for Applications (CAIA93), pp. 121–127, March 1993, Orlando, Florida.

[IMA1] Interactive Multimedia Association, "Recommended Practices for Multimedia Portability," Version 1.1, 800 K Street, N.W., Suite 240, Washington, D.C., 20001..

[Jamieson90] Jamieson P. W., "A New Paradigm for Explaining and Linking Knowledge in Diagnostic Problem Solving," J. of Clinical Engineering, Vol. 15, No 5, pp. 371–380, Sept/Oct. 1990.

[Kass91] Kass A. and Blevis E., "Learning Through Experience: An Intelligent Learning-by-Doing Environment for Business Consultants," Proc. of Conf. on Intelligent Computer-Aided Training (ICAT91), NASA Publication 10100, pp.289–302, 1991.

[Maybury91] Maybury M. T., "Planning Multimedia Explanation Using Communicative Acts," Proc. of AAAI-91, pp.61–66.

[McKeown92] McKeown K., Feiner S., Robin J., Seligmann D. D. and Tanenblatt M., "Generating Cross-References for Multimedia Explanation," Proc. AAAI-92, pp. 9–16, San Jose, CA, July 12–16, 1992.

[Minsky63] Minsky M., "Steps toward artificial intelligence," in Edward A. Feigenbaum, editor, Computers and Thought, pp. 406–450, McBraw-hill, New York, NY, 1963.

[Newell62] Newell A., Shaw J. C., and Simon H. A., "The process of creative thinking," In comtemporary Approaches to Creative Thinking, pp. 63–119, Atherton Press, New York, 1962.

[Rapaport86] Rapaport W. J., "Logical foundations for belief representation," Cognitive Science, Vol. 10, pp.371–422, Dec., 1986.

[Sullivan91] Sullivan C. A and Relyen R. J., "M3TE: Manufacturing Multimedia Training and Education," Proc. of IBM Multimedia ITL Conf., pp. 19-26, Boca Raton, Fl., Feb. 1991.

[Shortliffe76] Shortliffe E. H., "Computer-Based Medical Consultation: MYCIN," New York, American Elsevier, 1976.

[Smith89] Smith J. A., Biswas G., "Multilevel models for diagnosis of complex electro-mechanical systems," proc. Conf. Appl. Artificial Intelligence, M. M. Trevedi, Ed., Vol 7, Orlando, Fl, pp. 346–356, March 1989.

[Sticklen87] Sticklen J. H., "MDX2: An integrated medical diagnostic system," Ph.D. dissertation, Ohio State Univ., 1987.

[Vail88] Vail P. S., "Computer Integrated Manufacturing," PWS-KENT, Boston, MA, 1988.

[Yazdani88] Yazdani Masoud, "Expert Tutoring Systems," Expert Systems, Vol. 5, No. 4, pp. 270–271, November 1988.

N96- 14992

Embedding Speech into Virtual Realities

Christian-Arved Bohn and Wolfgang Krueger
German National Research Center for Computer Science (GMD)
Supercomputer Center (HLRZ)
Dept. *Scientific Visualization*
P.O. Box 1316
D-5205 Sankt Augustin 1, Germany
Tel. +49(2241)14-2230
Fax. +49(2241)14-2040
E-mail: {bohn, krueger}∂viswiz.gmd.de

Abstract

In this work a speaker-independent speech recognition system is presented, which is suitable for implementation in *Virtual Reality* applications.
The use of an artificial neural network in connection with a special compression of the acoustic input leads to a system, which is robust, fast, easy to use and needs no additional hardware, beside a common VR-equipment.

2 Introduction

Interaction has been proven to be one of the most important challenges for the future of nearly all computational tasks. Virtual Reality systems summarizes these under one subject. It may be a test for developed techniques, but also a mainspring for new research in this field.
Not leaving the user alone in a 'Virtual World' is the main intention and delivers the question: What are the best, the most intuitive, the most efficient interfaces between user and data-space ?
Thinking of the most often used 'real-world devices' automatically leads to speech recognition. To enable the human just to say, what he wants, is evidently a step away from the current user-interfaces, towards a more realistic man-machine interaction [14].
To get a practical speech recognition system, usable inside any VR-equipment, several conditions must be satisfied:

- Good recognition rate;
- Short delay time,
- Easy to use (easy adaptable to various speakers or configurable for any speaker),
- Avoidance of extensive resources (need to run in parallel with a VR hardware-software environment),
- Sufficient word capacity.

Therefore, emphasis was laid on *applicability* .
The descripted speech recognition system consists of the following moduls:

1. Built-in sound recording hardware of an *SiliconGraphics Indigo* workstation,
2. Fourier analysis module (software) for preprocessing speech input,
3. Compression module (software) for compression, generalization and for making the task robust,
4. *Artificial-neural-network* (software, *backpropagation*-network) for classification.

3 Speech Recognition with an Artificial Neural Network

3.1 Introducion

The ability to identify spoken words is desirable in a variety of application areas, such as manufacturing, telecommunication and medicine [15], but high-quality speech recognition systems are not easy to built. The challenging computational problems associated with speech recognition and the limited success of the conventional pattern matching techniques, proposed to solve them, have fostered the development of artificial neural network (*ANN*) approaches to speech recognition tasks.

Most speech recognition systems are designed in a similar manner. Given is a continous data-flow, the sampled speech data, which is divided into parts, which stand for closed units of speech, like phonems, syllables, whole words or higher structures. This separation makes the data-flow suitable for classification, and can, in case of recognizing whole words for example, be done by detection of short pauses between the words.
The separation follows a preprocessing sequence, which converts the sampled acoustic data in a compressed form.

The classification then tries to assign the preprocessed input to certain classes, for example to a class of utterances of one certain word. The number of classes is limited, that means, increasing the number of recognizable words for example either decreases the recognition probability or needs additional structuring methods, which extends the classification.
So, speech recognition systems can be structured as follows.

- Preprocessing
  Mostly a Fourier transformation is used, in addition a cepstrum analysis can be used. The resulting spectrum is subsampled, leading to *melscale frequencies* . Finally, a certain kind of normalizing leads to so called *speech vectors* , which are of low dimension (10 - 20 elements).

- Classification
  The classification part determines from speech vectors certain symbols like phonems, words or other parts of the speech. There exist various neural methods, like multi-layer-perceptrons, recurrent-, time-delay- neural networks or feature-maps [8, 10, 20].

- Postprocessing
  In the simplest case, this may be just the assignment of the classification to a certain event, for example to words for word recognition. Some systems build higher levels of speech by the use of Hidden-Markov-Models or by combining several neural networks. This allows to recognize a arbitrary number of words by building higher structures of speech, like words from phonems or sentences from words or syllables.

In the most cases there is no need of a postprocessing part. Speech recognition systems are often used in very special applications, where only a limited number of words have to be recognized. The recognition of an arbitrary number of words is needed in *phonetic-typewriter* systems.

3.2. Realization

This paper presents a system that is one of the 'word-recognition-type'. It recognizes a limited vocabulary of spoken words. The implementation can be tuned to speaker independent speech-recognition or to one-speaker application. The system requires no additional hardware support for acoustic preprocessing.

3.2.1 Preprocessing

The sound were sampled with 16000 Hz in ambient noise conditions. Only the hardware of an standard *SGI-Indigo* were used.

In the first step, the preprocessing software automatically extracts the individual words from the speech signal under consideration. The developed extraction algorithm succeeds in finding the word boundaries with an error rate of less than 5%, which is quite satisfactory considering that no provisions have been taken to avoid environmental noise.

A 512-point fast Fourier transform analysis, computed every 10.9 milliseconds using a Hamming window [16] (2/3 overlap between successive windows), is then performed to obtain the short-time frequency spectrums of each extracted word.

Each spectrum is subsequently transformed into a 15-dimensional speech vector by integrating the (logarithmically scaled) spectral amplitudes centered at the following frequencies in Hz (taken from [6]; bandwith indicated in parentheses): 130 (30), 164 (38), 206 (48), 260 (60), 327 (76), 412 (96), 520 (121), 655 (152), 828 (192), 1040 (242, 1310 (305), 1650 (384).

n a final step, the sequence of speech vectors of each word is compressed in time by accumulating and averaging them until the sum of their distances exceeds a threshold, as proposed in [2]. When this threshold is reached, the sequence of speech vectors is replaced by the average value, leading to 10-17 of such 15-dimensional speech vectors for each word considered. These are used as the neural network input.

The use of a compression algorithm is one of the major differences of this approach to other word recognition endeavours [1, 4, 6, 11, 12, 19]. which seem to be based on the assumption that as much as possible of the speech information should be kept to achieve high recognition rates. However, compression does not only reduce the dimensionality of the neural network inputs, which enables the network to learn faster, but also provides a more uniform representation of the utterances, which seems to be beneficial for improving the generalization ability of the network.

3.2.2 Classification/Postprocessing

Used is an feed-forward, multi-layer perceptron. Several network architectures were studied with different numbers of hidden units/layers and parameter settings for the single speaker word recognition task. The net which gave the best performance results is used in all experiments. The resulting network architecture is a 3-layer feed-forward network with 240 input units, 18 hidden units and 45 output units. The input layer receives the speech vectors of one word ordered into a linear array and the output layer uses a simple 1-out-of-45 coding, where the output unit with the highest activation corresponds to the word recognized by the network.

The learning rule is the standard backpropagation algorithm with the following properties: a) the usual quadratic error function, as described in [8], is used; b) errors are accumulated after each input in the training set; c) the learning rate lies between 0.4 and 0.9; d) the momentum term is 0.7; e) the weights are initialized to random values in the range between -0.3-0.3; and f) the input vectors are always presented in the same order.

4 Artificial Neural Networks

4.1 Network-architecture

Artificial neural networks were developed to overcome the disadvantages of common algorithmical solutions of computational problems. Because of the excellent facilities of the human brain in solving almost all challenges, that the today's computers in a life time never could, people tried to simulate the functionality of human's neural network. The computer should be enabled to think more intuitively.

So, small programs raised from this thought, which simulates one neuron (*unit*) put together by a network of simulated connections.

The task, one neuron has to do, is to sum up a number (*k*) of weighted (*w*) inputs (*x*) and to deliver the result *y* to the input of connected neurons (1).

$$y_i = g\left(\sum_j w_{ij}x_j - \mu_i\right), j = 1\ldots k \tag{1}$$

g is called *activation-* or *gain function* with its activation-threshold μ. The most implementations use (2).

$$g(h) = \frac{1}{1+e^{-2\beta h}} \tag{2}$$

It is a *sigmoid -function. ß* determines their steepness.
Putting *m* of such units together leads to a network, whose overall function can be described by (3). The actual output values of the units (*y*) at a certain time (*t*) are used to accumlate the values at the next time-step (*t+1*).

$$y_i(t+1) = g\left(\sum_j w_{ij}y_i(t) - \mu_i\right), i, j = 1\ldots m \tag{3}$$

So an ANN is defined by its topology (*w*)) and the kind and parameters of the activation function.

In this work a *multi-layer-perceptron* [8] is used. It is an feed-forward network with a certain number of input- and output-units.

4.2 Network-programming

The implementation of an ANN-simulation can be seen as program that simulates a stupid brain. To make it executing a wished function, it has to learn.

This learning is managed by presenting an event to the net and telling him what it is - especially, giving an input and the associated output, the network should try to adapt his own output (*supervised learning* [8]).

So, beside the net simulation program, there exists a second learning program (*Backpropagation*-algorithm (*BP*) ,[8, 5]). It does the learning by modifying the ANN's topology (*w*) and the parameters of the gain-function. A sequence of input-output pairs is presented to the ANN. BP calculates an error from the real and the wished output and the net parameters are corrected to a decreasing error.

This is done for the whole set of learning pairs several times, hoping that the net error sinks into a global minimum.

5 Application

5.1 Example

To give an impression of the problem let's see a short example. Figure 1 shows the sampled raw data of an utterance of the german word /quadrat/. This transformed in uncompressed speech vectors looks like figure 2. After compression there is just figure 3 left.

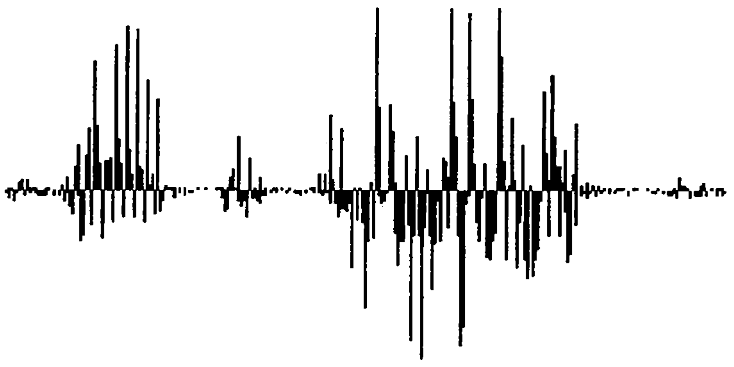


Figure 1 Sampled word /quadrat/.

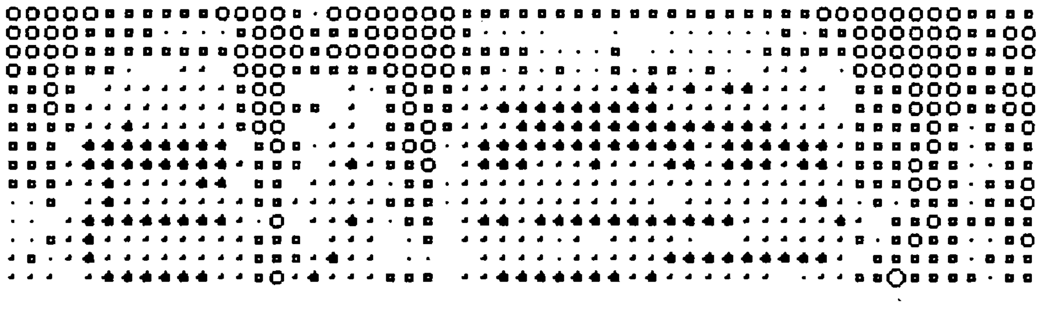


Figure 2 Word converted to speech vectors.

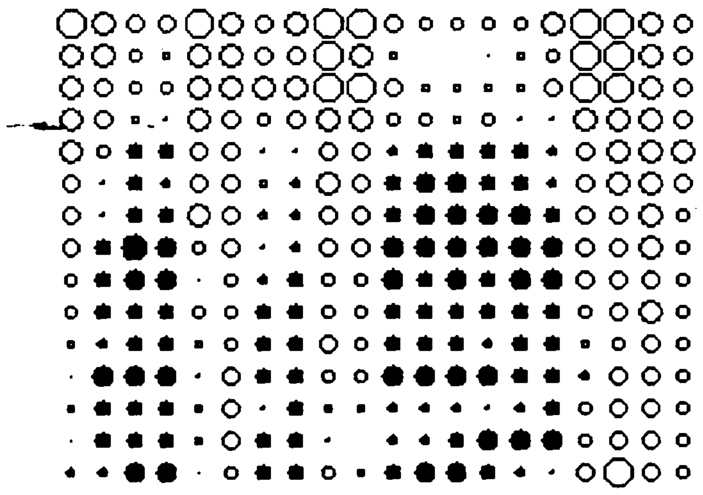


Figure 3 Speech vectors after compression.

5.2 Tests, Results

Training the net always happens in the same manner. There is a limited number of words, that the network is trained with, the *learn-set*. A second *test-set*, containing the same words but different utterances, is only applied, without any effects to the net parameters. The recognition rate of this foreign, unknown words is taken as a convergence criterion; it shows the generalization ability of the network, and may be seen similar to an application of the trained net after learning.

The net converges fast, at most in less than one minute. To learn a number of words, the speaker has to speak the application-specific vocabulary about five times - waits a few minutes, and then can use his trained network in an arbitrary implementation. Recognition time after learning is less then one second.
In contrast to other speech-recognition approaches, the system showed good robustness and applicability. Convergence appears in almost all cases, there is no need of support of specialists or people, familiar with the implementation. The following tests prooved usability of the system.

5.2.1 Single-Speaker Word Recognition

Goal was to train the network to 'understand' the vocabulary of one speaker, to have a higher recognition rate, than in the multi-speaker case. Each word was spoken four times. The net learned three different utteranes of 45 words. As test of generalization ability the last word-set were applied. The net learned all words from the learn-set correctly. The never-heard words of the test-set were correct identified in 96% of the cases.

5.2.2 Speaker-Independent Word Recognition

In contrast to trimming the net to the speech of one certain speaker, the training time when changing the user can be cancelled out by training the network to arbitrary users. The lower recognition rate can be neglected, looking to the advantages of a speaker(user)-independent application.

The utterances of 45 words of 16 speakers were used. 8 sets were defined as learn sets, the other 8 as test sets. The ANN learned again all utterances. The generalization ability converged to 72%. A similar test with only male speakers increased recognition-rate to 91%.

5.2.3 Speaker Recognition

The last test should show the typical flexibility of an ANN-system. It should learn speaker-recognition. Learn sets of each 30 words of five speakers were created. The test sets contained 10 other words of the same five speakers. After learning, hearing these unknown words, the system should say, which of the speakers this utterances did. The net could recognize the person with an exactness of 82%.

5.3 Observations

The typical behaviour, which are exspected if using ANN's could be seen.

- The hidden layer creates an internal coding of the net input, that means their output-values mostly have certain levels, similar to binary values.

- There is a better generalization-ability with *not exact learning* by either a) using less hidden units or b) stopping learning before final convergence. In most cases the best generalization is reached if 80% of the input words are learned.

- The learning behaviour is robuster for a smaller net. That means less input units and so less weights in the first layer accelerate learning. The convergence curve was more smooth and the convergence faster - a further reason for using the above mentioned compression-algorithm.

6 Conclusion

Our work demonstrates the practical feasibility of building a high quality speaker-independent speech recognition system, which can easily be trained to recognize the desired words with high accuracy and does not require the assistance of somebody, who is deeply familiar with the issues involved in speech processing.

The system has purely been implemented in software and is quite competitive to other approaches. Through the special design by using a certain compression method of the input data and an artificial neural network, a system has arised, which is well suited for applications where a limited vocabulary needs to be recognized and where using has to be fast, robust and uncomplicated, like in a Virtual Reality application. The time for running the preprocessing software and training the network with the backpropagation algorithm is pretty short (about one minute), and the recognition delay time lies under one second.

The system could also be used for speaker-recognition by simply training the network to learn the mapping between a set of words and a number of speakers. This shows a great flexibility, which is also an important feature for Virtual Reality applications, where one can imagine alternative noise-steering methods.

7 Discussion, Future

Actually the system has been implemented into several VR-applications, that includes a standard VR-system, but also an application with the *BOOM2* . Of course, it can be combined with every subset of VR-equipment. Almost steering tasks in addition or in case of gesture-recognition is tested and gives at most a better feeling in moving through the data-space.

It reveals that gestures, used for moving, are very far beyond the human intuitiveness. Simply saying 'back' for example seems to be easier than stretching or bending some middle (which ?) finger.

A current VR-project called *Responsive Environment* [9] will be a very attractive application. Subject of this project is a workbench, where many people can in parallel interact, work together. The project simulates the ordinary working-environment of architects designers and surgery crews. Such a cooperative work can greatly be assisted by tools like speech recognition. Even this case is, because of its overlapping acoustical events, a great challenge for a speech recognition system.

8 Bibliography

[1] Behme H: *A Neural Net for Recognition and Storing of Spoken Words,* In: Parallel Processing in Neural Systems and Computers, pp. 379-382, Elsevier Science Publishers, 1990.

[2] Bengio Y, Cardin R, and De Mori R: *Speaker Independent Speech Recognition with Neural Networks and Speech Knowledge* In: Advances in Neural Information Processing Systems , Vol. 2, pp. 218-225, Morgan Kaufman Publishers, 1990.

[3] Bourlard H, and Morgan N: *A Continuous Speech Recognition System Embedding MLP into HMM,* In: Advances in Neural Information Processing Systems , Vol. 2, pp. 186-193, Morgan Kaufman Publishers, 1990.

[4] Franzini M A: *Learning to Recognize Spoken Words: A study in Connectionist Speech Recognition ,* In: Proceedings of the 1988 Connectionist Models Summer School , pp. 407-416, Morgan Kaufman Publishers, 1988.

[5] Freisleben B, Bohn C-A: *Speaker-Independent Word Recognition with Backpropagation Networks* In: Artificial Neural Nets and Genetic Algorithms , pp. 243-248, Springer-Verlag Wien New York, 1993.

[6] Grajski K A, Witmer D P, and Chen C,: *A Preliminary Note on Static and Recurrent Neural Networks for Word-Level Speech Recognition,* In: Proceedings of the 1990 International Joint Conference on Neural Networks , Vol.~2, pp. 245-248, Lawrence Erlbaum Publishers, 1990.

[7] Hampshire II J B, and Waibel A: *Connectionist Architectures for Multi-Speaker Phoneme Recognition,* In: Advances in Neural Information Processing Systems , Vol. 2, pp. 203-210, Morgan Kaufman Publishers, 1990.

[8] Hertz J A, Krogh A, and Palmer R, *Introduction to the Theory of Neural Computation* , Addison-Wesley, Reading, Massachusetts, 1991.

[9] Krueger M: *Artificial Reality II* Addison-Wesley, Reading, Massachusetts, 1991.

[10] Kohonen T: *The Neural Phonetic Typewriter,,* IEEE Computer , 3:11-22, 1988.

[11] Kowalewski F, and Strube H: *Word Recognition with a Recurrent Neural Network,* In: Parallel Processing in Neural Systems and Computers , pp. 390-394, Elsevier Publishers, 1990.

[12] Lee K: *Context-Dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition,* IEEE Transactions on Acoustics, Speech, and Signal Processing , 38(4), 1990.

[13] Lee Y, and Lippmann R P: *Practical Characteristics of Neural Network and Conventional Pattern Classifiers on Artificial and Speech Problems,* In: Advances in Neural Information Processing Systems , Vol.~2, pp. 168-177, Morgan Kaufman Publishers, 1990.

[14] Nielson J: *Noncommand User Interfacer,* In: Communications of the ACM , Vol. 36, No. 4, pp. 82-99, ACM, New York, 1993.

[15] Peacocke R D, and Graf D H: *An Introduction to Speech and Speaker Recognition',* IEEE Computer , 8:26-33, 1990.

[16] Rabiner L R, and Gold B: *Theory and Applications of Digital Signal Processing,* Prentice-Hall , 1975.

[17] Rigoll G: *Neural Network Based Continous Speech Recognition by Combining Self Organizing Maps and Hidden Markov Modelling,* In: Lecture Notes in Computer Science , Vol. 134, pp. 58-65, Springer-Verlag, Berlin, 1990.

[18] Rumelhart, D E, Hinton, G, and Williams, R E: *Learning Internal Representations by Error Propagation,* In: Parallel Distributed Processing: Explorations in the Microstructures of Cognition , Vol. 1, 318-362, MIT Press.

[19] Sung C, and Jones W C: *A Speech Recognition System Featuring Neural Network Processing of Global Lexical Features,* In: Proceedings of the 1990 International Joint Conference on Neural Networks , Vol.~2, pp. 437-440, Lawrence Erlbaum Publishers, 1990.

[20] Waibel A, Sawai H, and Shikano K:, *Modularity and Scaling in Large Phonemic Neural Networks,* IEEE Transactions on Acoustics, Speech, and Signal Processing , 37(12):1888-1889, 1989.

[21] Waibel A, Hanazawa T, Hinton G, Shikano K, and Lang K: *Phoneme Recognition Using Time-Delay Neural Networks,* IEEE Transactions on Acoustics, Speech, and Signal Processing , 37(3):328-339, 1989.

Virtual Instrument Technology

Major Donald Pryor and James Larsen
US Army, HQ TRADOC
DCST
Attn: ATTG-CI
Ft. Monroe, VA 23651
804-728-5506

Virtual Instrument technology is a computer-based software development that could revolutionize field maintenance performance and training. The object-based developmental software is a true graphics language where the developer only manipulates software "objects" on the screen, not text. The resulting control panel (ex. a multimeter) is both a working, functionally capable multimeter and also a 2D or 3D simulator.

VI Capabilities: The VI world uses a standard bus structure which allows for the following functions:

- IEEE 488 external control of 3D instruments

- Software-generated signals and fault insertion

- Statistical data collection within the instrument

- Embedded training/help functions within the instrument

- Rapid prototyping/human engineering of instruments

- Tenfold decrease in simulator authoring for CBT

VI Military Applications: VI technology can essentially replace existing Test, Measurement, and Diagnostic Equipment (TMDE) and panel trainers at dramatically lower costs with increases in mission performance.

VI Benefits: Combining TMDE and training simulations into one computer-based system will increase mission capability, increase soldier performance, decrease costs, and make CBT affordable and deployable.

N96-14993

Spatial Considerations for Instructional Development in a Virtual Environment

Laurie McCarthy
Michael Pontecorvo
Frances Grant
Randy Stiles

Lockheed Palo Alto Research Laboratories*

Abstract

In this paper we discuss spatial considerations for instructional development in a virtual environment. For both the instructional developer and the student, the important spatial criteria are perspective, orientation, scale, level of visual detail, and granularity of simulation. Developing a representation that allows an instructional developer to specify spatial criteria and enables intelligent agents to reason about a given instructional problem is of paramount importance to the success of instruction delivered in a virtual environment, especially one that supports dynamic exploration or spans more than one scale of operation.

1.0 Introduction

The motivation for integrating an instructional system with a simulation is usually to support exploratory learning. In the case of a virtual environment for instruction, the main purpose is to give the students three-dimensional visual feedback during exploration. Getting the proper mix of exploration and visual presentation can be difficult. In an exploratory setting, an instructional developer normally does know ahead of time if the student will be able to witness certain events in the simulation. Our goal is to support the developer's task of explanation by providing a representational system that can be used to support reasoning about the appropriate spatial presentation of events, and a set of visual abstractions for manipulating the underlying representation.

The spatial issues presented in this paper are an outgrowth of the research conducted for the design of Lockheed's virtual environment instructional system. We are working on a visual programming system to support the instructional developer in the seamless visual development of lesson plans, objects, and simulations situated in a virtual environment (Stiles, 1992). The underlying metaphor for the Lockheed system is that of a television studio, with a studio control booth, stage, and audience section (see Figure 1). The control booth serves as the developer's information workspace (Card, 1991), providing all the tools required for courseware development. The visual simulation and interactions with the system are carried out on the studio stage, where the trainee may participate and affect the outcome of a given instructional simulation. The audience metaphor allows passive observation, and if the instructional developer allows it, provides the trainee the freedom of movement within the virtual environment without affecting the simulation (Grant, 1991).

In the following sections, we cover related work, review some of the spatial problems that are present when carrying out instructional development using a virtual environment, discuss our representation system, called *tscript*, for dealing with these problems, and then cover the approach for reasoning in an intelligent tutoring system using this representation.

* contact mccarthy@aic.lockheed.com ph 415.354.5257 or stiles@aic.lockheed.com ph 415.354.5256

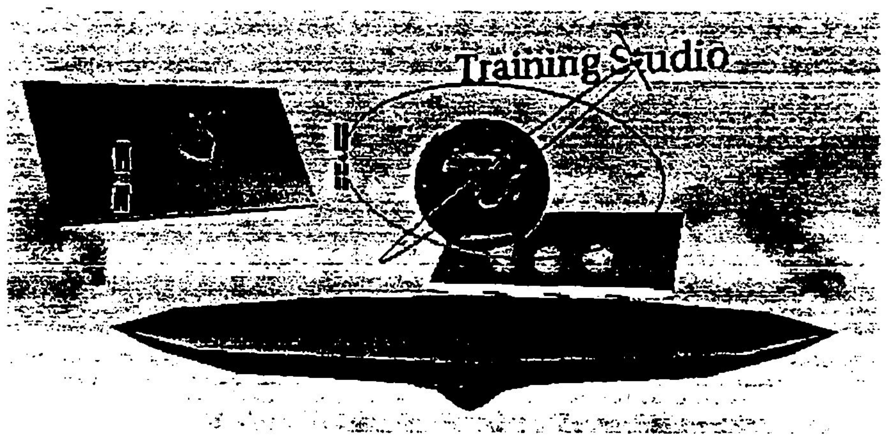


Figure 1. Training studio viewed from control room, with stage as initial reference

2.0 Background

The problem of ensuring effective understanding of three-dimensional phenomena has been addressed from a number of directions. There have been efforts to understand what attributes can enhance or degrade communication in a three-dimensional setting (Sedgewick 1991, Goldstein 1991), and others that have characterized the nature of pictorial communication and arrived at specific solutions for rescaling to achieve understanding (Ellis 1991, Eyles 1991). Then there have been attempts at automatic generation of explanatory three-dimensional scenes in response to dynamic changes in viewpoint and changing communication goals. We feel that one of the most outstanding efforts in this area is by Seligmann and Feiner, in their Intent-Based Illustration System (IBIS) (Seligmann 1991).

The IBIS system uses a collection of CLIPS-based style rules that create possible 3D illustrations and then test them for suitability. This process is handled by bodies of rules called illustrators and drafters. Illustrators are given a set of communication goals, and these goals trigger design rules that could possibly satisfy the goals. The results of the illustrator design rules, called style strategies, are passed on to the drafter. Drafters have associated with them style rules that invoke style methods. In this way, a possible 3D illustration is generated. Then the evaluation process begins. Both the design rules and style rules also have evaluation constraints associated with them, and the resulting illustration is evaluated first by the drafter's style evaluators, and then by the illustrator's design evaluators. If a given illustration strategy meets the evaluation criteria, it is then used.

Our efforts are concerned with extending this approach to address training triggered by a given locale, transitions from one locale and/or scale to another for a given training purpose, and allowing the interactive, visual specification of constraints and scenes for the delivery of instruction in a virtual environment. We do not make a distinction between illustrator rules and drafter rules, but we do adopt an generate and test approach to structure the search for an appropriate presentation.

3.0 Spatial Considerations

Many three-dimensional graphical representations, and accompanying simulations, are based on implicit assumptions about the scale they operate at, and the abstract representations that can be shown with them. These scale categories are item-based (CAD) system-based (ships, airplanes), locale-based (DIS/SIMNET battlefields) and macro-based (Planetary Systems, Orbital Representations, Galaxies, etc.). When presentation or interaction across these traditional categories is required, it is necessary to provide selected elements on several scales. For instance, training students in orbital mechanics on the distinctions between geodetic and geocentric coordinate systems may perhaps be best accomplished by providing a ground perspective for a local patch of terrain overlaid with a fine latitude and longitude grid, and drawing orientation lines from the person out to satellites above. This can be followed by launching the person's perspective smoothly out into the same orbit, at all times providing the same orientation lines from launch point to moving satellite, and then showing a transparent earth with the geocentric coordinate system underlying.

A key feature of virtual environments is the ability to change perspectives not only by manipulation of an isolated object within an environment but also by changing viewers coordinates with respect to a system. In the first case, the object's location or orientation changes, but the environment within which it exists remains constant, thus maintaining the orientation of and frame of reference for the viewer. The second condition, however, changing the coordinates of the viewer, involves a new frame of reference. Drastic changes can be disorienting, for example, watching a satellite orbit the earth at some distant point then suddenly moving to a location on earth observing the same satellite. Unless a link between the two viewpoints is established, by providing a common point of reference or some type of observable mapping, the significance of the lesson may be lost.

Changing to a completely different view can create similar orientation problems. The trainee may be required to view a location currently off-screen. If the two coordinates are distant, the two views may have nothing in common and the trainees may have difficulty placing the new location in context with respect to the previous.

Establishing a link between two seemingly disparate scenes can be accomplished before, during, or after the transition. "You are here" type maps that diagram the two locations can be displayed on either or both before and after displays. It may be useful to allow trainees to observe the transition itself; i.e., to sequence the trainee's view points over time, similar to the way a film camera pans a scene. Other effective techniques borrowed from the film industry are the concepts of zooming in (showing general scenes then moving closer to observe a particular area in detail) and zooming out (show close up views then pulling back to view the entire scene) for providing context and establishing frames of reference.

Current technology allows the representation of real life objects and processes to a great level of detail; far greater than one might need or want in an instructional sequence. Over-complexity can be detrimental to the learning conditions of the lesson since it can be difficult for learners to recognize the pertinent or critical information. This becomes especially evident when introducing complex processes. Given that trainees automatically attempt to formulate meaningful wholes from a presentation, an over-complex display can increase the probability of erroneous interpretations (Gagne, 1987). In fact, perception of critical information can be increased by limiting details (Fleming & Levie, 1978) and three-dimensional representations may not always be the most desirable for introducing concepts. Instead it may prove fruitful to introduce complex concepts or scenes as two-dimensional projections, allowing three-dimensional viewing and examination as the trainee becomes more sophisticated with respect to the domain. Some of the physics presented in "The Mechanical Universe," for example, were simplified to two-dimensional simulations to present a more understandable view of complex concepts (Blinn, 1991). There are also cases when details may be purposely blurred. For example, the earth, may be used as a link between two views. A detailed representation of the earth may appear in one scene but may be just a distant object in the current display. By providing a less detailed representation of the earth in the distance, this object becomes a reminder of earth's location without distracting the learners attention.

On the other hand, providing too little detail can have a detrimental effect since learners tend to disregard material that is too simplistic (Fleming and Levie, 1978). Additionally, the simple-to-complex characteristic of most instructional sequencing strategies suggests that levels of details should be dynamic, changing with respect to the current state of the student model. The developer, then, must have the capability of providing varying levels of detail depending upon the needs/abilities of the individual learner. Additionally, the developer must be able to specify conditions under which certain details are added or detracted.

Due to the exploratory nature of the system, a trainee's path, orientation and location just prior to a particular lesson cannot always be predicted. To provide guidance to the trainee, the developer must be able to control when certain events will take place with respect to the instructional maturity of the trainee. Strict temporal sequencing is not possible since neither the trainee's path nor time on a task is predictable. Spatial coordinates are important since certain events can only be viewed form specific locations; however, they cannot be the only triggers. The hierarchical nature of the domain content must also be considered, as well as the learning history of the trainee, and there are a variety of approaches that can be implemented to address these concerns. For example, a simulation can be activated by a spatial trigger, with granularity and details adjusted with respect to the trainee's history, or the simulation can involve a specific level of complexity and be triggered only when the learner has acquired the appropriate skills and knowledge. In either case, most instructional developers would not have the time or resources to develop separate lessons to adequately address all possible scenarios.

As discussed above, explorations and manipulations within a virtual environment involve special spatial considerations, such as orientation, level of visual detail, and granularity of simulations. To develop instructional sequences in a three-dimensional system, the developer requires a suite of tools, in addition to those relating to pedagogical and domain-related issues, that provide the following:

- Objects representing viewpoints that can be physically positioned in the virtual development. The developer should also have to ability to designate temporal attributes to these view objects to produce guided instructional sequences or "tours."
- A series of camera techniques, such as pans and zooms, to allow the developer to control or guide perspective and focus in an instructional sequence or lesson. These techniques must be represented in a form that will allow easy specification by the developer; for example, providing icons that can be attached to a view-object.
- A mapping ability to show start and end coordinates, as well as the path between. The capability to display such maps in a variety of methods (e.g., pop-up windows, overlays) is also required.
- The ability to transition to and from two-dimensional projections.
- A method of adding and subtracting detail for existing objects and processes. All objects must be scalable both individually and within a specified context.
- A method for describing domain, learner, and spatial criteria as triggers for presenting simulations and lessons. The criteria would then be placed at the appropriate coordinates in the environment and would activate specified levels of instructional sequences only when the trainee fulfilled all criteria.

All the capabilities discussed above must be supplied in forms easily recognized, manipulated, and implemented by a non-programmer. Figure 2 illustrates some of these issues using a satellite operations scenario. Eye objects are placed to guide the student's view at critical parts of the simulation. Audio can be added, and a text window provides additional information, in this case, perhaps to display telemetry corresponding to the satellite location or a two-dimensional map of the earth region covered by the satellite. Objects are exaggerated out of scale (here, the satellite is nearly the same size as earth), and realistic but unnecessary details, such as the planets between the earth and sun, have been eliminated.

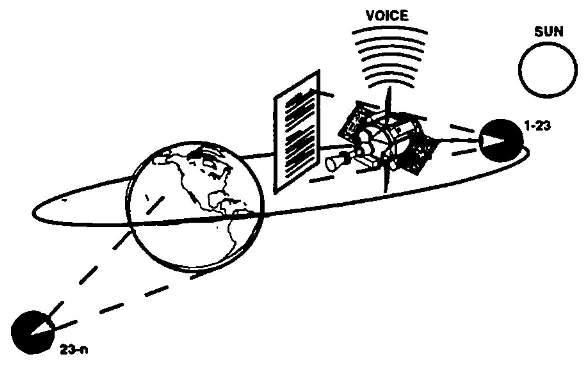


Figure 2. Sample scenario involving exaggerations of scale, two-dimensional displays, and viewpoints.

4.0 Representational System

In this section we discuss the representational system necessary for illustrating concepts in a spatially oriented, dynamic system, and how they relate to instructional development. For our purposes, an instructional unit is defined by a single learning objective. In the case of a virtual environment, the objective of an instructional unit is accomplished by presenting the student with a set of visual objects that are arranged spatially from one time step to the next in such a manner as to illustrate a concept. This scenario may be accompanied by other effects, such as sound, but that is not our emphasis.

4.1 Objects and Actors

Our framework adopts the useful object/actor breakdown for modeling a domain. An object is a model of a real or abstract entity, characterized by a state, behavior, and appearance. Objects may be simple or complex. Simple objects cannot be subdivided into sub-objects at the granularity of the model representation. A complex object is a grouping of component objects. Objects that possess intention or internal state can be considered actors. Additionally, objects can be considered either domain objects or instructional objects. Domain objects model the subject matter of the instruction. Instructional objects serve as tools for presentation and manipulation of the domain objects.

An object's state consists of attribute-value pairs which define that object's particular model of itself and the environment in which it is situated. This can be subdivided into internal and external state elements. Another useful characterization of an object's state is to partition the state into public and private elements. The publicly accessible portion of an object's state includes its appearance which is given a special distinction in the area of virtual environment modeling.

Domain objects in an instructional unit setting may serve different roles. Some objects merely provide context or cues of scale and scope. Other objects are more central to the current instructional unit. Other attributes of an object must be assigned measures of appropriateness or applicability for a given context. These attributes include rendering effects such as fixed vs. dynamic orientation shifts, transparency, material properties (surface smoothness), lighting (shadows and reflection), texturing, spatio-temporal and orientation distortion, spatial marking (highlighting) level of detail, and realism (photo-realistic to 3D and 2D stylizations). Behavioral characteristics in this category include model fidelity (high-resolution model vs. primitive lookup-tables), and temporal distortions.

At the most fundamental level an object's behavior is characterized as a collection of stimuli-response pairs. A stimulus is represented as a set of events, which are changes in the internal or external state of a given object. A response is a sequence of primitive actions; i.e., operations which cause changes in the state or appearance of an object or its external environment. In this model, when an object perceives that a specific set of events have taken place, the associated sequence of actions associated with that set of events is carried out. Environment modifying actions are realized by the sending of messages between objects either in a point to point or broadcast fashion.

An object in a virtual environment has a spatial extent or envelope. In the context of an instructional unit an object exists in 'some' state at a location or in a region (spatial envelope) for the duration of the instructional unit. More specifically, the object's location and appearance is dictated primarily by the objectives of the instructional unit in conjunction with the orientation of the viewer and objects relevant to the instructional unit.

Object Elements	**Description**
(scale <reference-object> <object>)	sets the scale property for a given object
(attr-relevance <object> <context>)	
(reference <geometry> <origin>)	defines a reference system using a point, geometry or object as origin
(locale <object> <viewpoint> <detail>)	defines a locale based on view, level of detail, and relevant object

4.2 Envelope Conditionals

Spatial envelopes provide a convenient means of characterizing object and inter-object behaviors. A behavior or action can be characterized by its extent in the virtual space. The triggering of a behavior is represented as the crossing of the boundary surface in a space consisting of the states of the system and space and time. For many practical situations the triggering of a behavior is simply a matter of crossing the threshold of a particular spatial or temporal envelope when a particular object state is present.

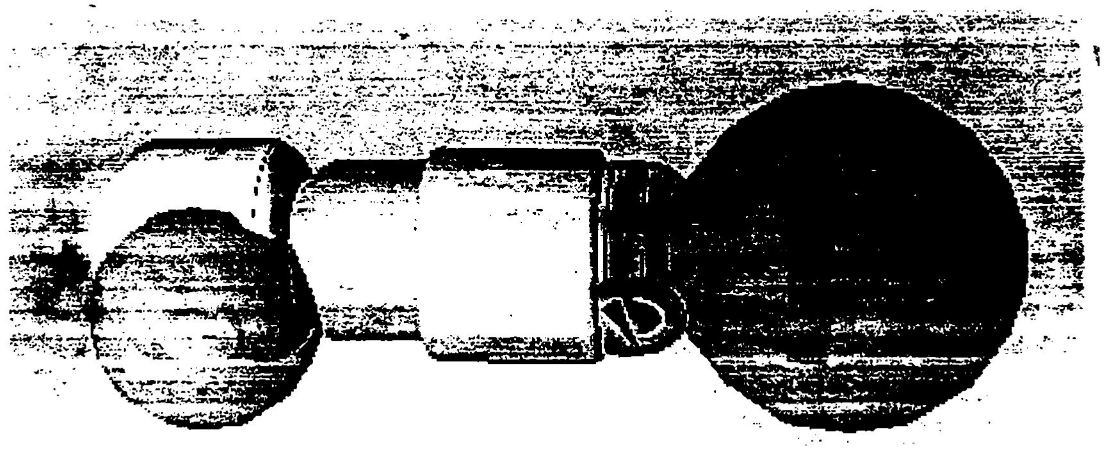


Figure 3. Spatial envelopes associated with rocket engine nozzle and inlet

An example would be the envelopment of an object in a nested spherical enveloping-regions with graduated sizes, with a different appearance rendering behavior attached to each sphere. The viewer' visual focus (eye) crossing an envelope relevant to an instructional unit would cause the rendering behavior associated with that envelope to be used to render the object. Moving closer to the object could cause renditions of increasing level of detail to be used for viewing the object. This is often done in visual simulations.

Envelope conditionals can be modified to be applicable only at a given scale by adding another conjunctive conditional wherever they appear that explicitly references scale of operation. Alternatively, they can be made applicable across scales by utilizing the fact that each envelope associated with an object is attached as a child in the transformation hierarchy, and will be appropriately scaled and translated whenever its associated object is.

Envelope Elements	Description
(envSphere <object> <radius> <opt-scale>)	specifies spherical envelope around and object or point
(envRect <object> <x-extent> <y-extent> <z-extent> <opt-scale>)	specifies bounding box envelope
(envGeometry <object> <scale-factor>)	specifies more detailed envelope, using a new scaling factor applied to the geometry

4.3 Viewing

Viepoints onto a given scene are used during the specification of transitions, and by themselves in the context of an instructional unit. These viewing elements can be used as conditionals that trigger actions associated with an instructional unit, including level of simulation granularity, relevant explanation actions, or display actions. In display actions, the viewing elements of *tscript* are used on the left-hand-side of generative rules serve to constrain the kinds of display actions or transitions that are generated. When they appear on the left-hand-side of evaluation rules, they are used as tests, determining whether or not the display objective has been met. The elements of our representation that describe constraints on views follow.

Viewing Constraints	Description
(viewPos <viewer> <object> x y z)	specifies view position relative to a given object's center, commonly known as tethering
(viewAnchor <viewer> <object>)	specifies orientation of viewer relative to another object
(viewGlobalPos <viewer> x y z)	specifies position of viewer in global coordinates always overrides any conflicting view
(viewReference <viewer> <object> <reference>)	specifies that reference system should be present in scene
(viewNormal <viewer> <object>)	view object normal to surface
(viewVisible <viewer> <object>)	ensure object is visible in any resulting scene
(viewEntire <viewer> <object>)	ensure all of an object's bounding box is visible in any chosen scene
(viewContain <viewer> <object1> <object2>)	make sure view contains both objects in relation to each other, independent of scale

4.4 Transitions

The envelopes of objects relevant to an instructional-unit may include vast areas of transition (dead zones) between regions of interaction. This motivates the need for a representation of these transitions allowing manipulation and reasoning about transitions. Methods for handling spatial or temporal transitions include:

- path-following - travel along a given path to impart a sense of continuity.
- inset transition - travel out to destination showing inset of original location (see Figure 4).
- teleporting - abrupt discontinuous movement from one place or time to another.
- warping - Constitutes a transition where compression or expansion of time or space when traveling between two locales is used. An excellent example of this is logarithmic travel where an observer travels the same relative percentage toward a given object with each time step, initially traveling very fast, and gradually slowing down as the target locale is reached (Mack 1990).
- shift exaggeration - A modification of teleporting where cinematic style transitions such as wipes, fades, zooms, pans are used.

Shift exaggerations do not conform to physical laws, but have been shown to be quite effective cues to the observer. These techniques can be used as a subtle reinforcement to highlight a transition when applied in combination with the other more physically consistent transition methods.

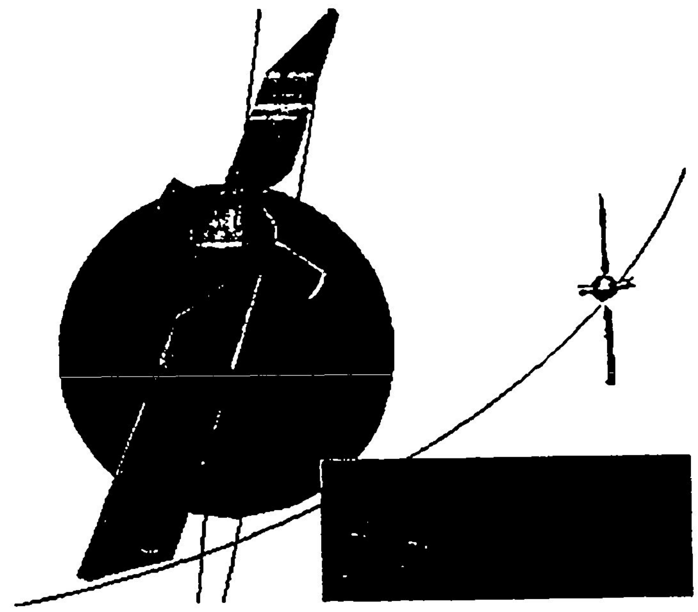


Figure 4. Inset transition from locale scale to macro-scale in orbit

Transition Element	Description
(tranInset <view1> <view2>)	specifies transition using inset of original vie prior to start of transition
(tranWarp <view1> <view2> <percentage>)	specifies a warp transition type, with percentag being the amount of remaining distance covered a each step of the transition
(tranTeleport <view1> <view2> <overlay>)	specifies a teleport transition type, with optiona overlay
(tranPath <view1> <view2> <path-list>)	specifies a path transition over a defined path
(tranRoadMap <view1> <view2> <reference>)	specifies transition with roadmap associated wit different scale inserted between
(tranSurface <view1> <view2> <object>)	specifies transition across surface
(tranDirect <view1> <view2>)	specifies shortest line transition

4.5 Level of Detail

Displaying geometric objects at different levels of detail based on the distance from a viewer is a practical approach to supporting interactive rates in virtual environments. When visible objects are distant, they can be replaced with less polygonally detailed models, and thus the frame rate for rendering these objects can be increased. This principle can be useful for instructional purposes as well. As Fleming & Levie (1978) note, the learner does not always know what is the most critical information in a complex display. To minimize the discrepancy between the nominal stimulus (what the designer meant by the display) and the effective stimulus (what learner attends to), the essential part must be salient, dominant, and noticeable (Gagne, 1987). There are cases when instructional material must be presented in less detail, and then further detail is introduced when it becomes relevant.

The easiest qualification on detail is a simple number, with 1 being the highest level of detail available in the system, and increasing numbers indicating lessening detail. This is easily supported under tree-based graphics systems such as IRIS Inventor (Strauss 1992) by testing distance and then selectively displaying nodes by switching between them, or IRIS Performer (McLendon 1992) by using built-in level-of-detail testing. Qualitative constraints on detail are important as well. For example, in the case where a student is being familiarized with an engine assembly, detail including individual bolts is not necessary. But as instruction progresses, and stripping down the engine is the focus of the instructional unit, this level of detail becomes appropriate.

Detail Element	Description
(detail <object> <detail-level>)	specifies detail using numerical detail level, ca override default checks based on distance fro object center
(detail <object> <detail-object>)	specifies detail level that has a given sub-objec recognizable and included in scene

4.6 Visual Specification

We are working on the visual specification of *tscript* elements. Transition types and spatial envelopes will be represented by iconic geometric structures. In the case of envelopes, a simple system of attaching an icon to a given domain object and then stretching it out to the appropriate size will be supported. For *tscript* elements that require object arguments, direct selection of each object is appropriate, initially using a mouse, and eventually using a glove interface. Transitions pose a larger problem because they can involve radically different settings. This can be handled by reducing locales themselves to small icons in the training studio control room.

5.0 Instructional Approach

This section relates the *tscript* representational system to the characteristic intelligent tutoring system (ITS) framework. Most intelligent tutoring system include an expert module, a student module, an instructional module, an interface, and a simulation (Regian 1991). The *tscript* reasoning system should be considered part of the instructional module. One might at first assume *tscript* is a natural part of the interface system, but *tscript* allows specification of, and reasoning across, a spatial domain. For virtual environments, the domain of instruction is mapped onto a spatial representation. Pedagogy expressed in *tscript,* and concerned with the appropriate way to present spatial and geometrical information, is essential.

5.1 Limiting Evaluation

Through the application spatial envelopes, the search for appropriate presentation of objects in the training environment can be limited to those objects which are relevant to the student at a given locale. Of course, the first and primary criteria for limiting evaluation is the student model (see Figure 5). At the point where a set of actions are possible using the student model, they also are matched conjunctively against any applicable spatial envelope conditions, using viewer coordinates. At this point, a smaller set of instructional strategies are generated. These are then evaluated against viewing and detail criteria to result in a decision on the appropriate instructional diaplay.

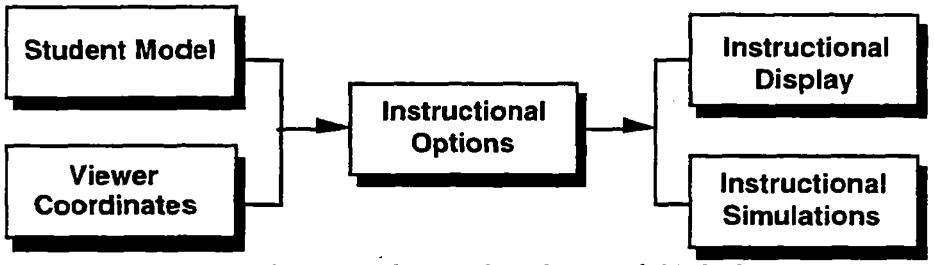


Figure 5. Viewer coordinates and normal student model info determine instructional options

5.2 Prioritizing Spatial Responses

Under the dynamic conditions supported by a domain simulation, it is possible for viewing constraints stated by the instructional developer to be in conflict. To resolve conflicts, we associate a dynamic measure of importance with each object. This measure ranks the objects in priority for computing resources and the attention of the student. The priority can't be a simple quantity because it must be able to express the priority of an object along multiple dimensions. These dimensions include: behavioral fidelity, rendering fidelity, and relevance to the instructional unit. This priority will determine which of several competing instructional actions will be used.

6.0 Summary and Future Work

Virtual environments couple visual, spatial representations with one or more simulated aspects of reality. When they are used in support of training, there must be support for visual presentation of instructional concepts, which may not always be subordinate to realistic display and interaction. We have highlighted a set of visual effects that are necessary for effective instruction in a virtual environment, and presented a representational system for defining and reasoning about these visual effects in the context of a dynamic, simulated environment.

Our future work will focus on visually specifying the elements of *tscript*, building a comprehensive set of heuristic 3D display rules, and on refining the interaction between the *tscript* presentation rules and the student module.

References

Blinn, J.F. (1991). The making of The Mechanical Universe. In S. Ellis, M. Kaiser & A. Grunwald (Eds.) *Pictorial Communication in Virtual and Real Environments* (pp. 138-155). New York, NY: Taylor & Francis, Ltd.

Ellis, S. R. (1991). Pictorial communication: pictures and the synthetic universe. *Pictorial Communication in Virtual and Real Environments* (pp. 22-40). New York, NY: Taylor & Francis, Ltd.

Eyles, D. E. (1991). A computer graphics system for visualizing spacecraft in orbit. In S. Ellis, M. Kaiser & A. Grunwald (Eds.) *Pictorial Communication in Virtual and Real Environments* (pp. 196-206). New York, NY: Taylor & Francis, Ltd.

Fleming, M., & Levie, W. H. (1978). *Instructional message design.* Englewood Cliffs, NJ: Educational Technology Publications.

Gagne, R. M. (1987), *Instructional Technology: Foundations.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Gagne, R. M., & Driscoll, M. F. (1988), *The essentials of learning for instruction, 3rd edition.* Fort Worth, TX: Holt, Rinehart and Winston.

Grant, F., McCarthy, L., Pontecorvo, M., Stiles, R., (Nov. 1991). Training in virtual environments. *Conference on Intelligent Computer-Aided Training,* Houston, TX.

Mackinlay, J. D., Card, S. D., Robertson, G. G. (1990) Rapid controlled movement through a virtual 3D workspace. In *SIGGRAPH Conference Proceedings,* (pp. 171-176). Dallas TX: ACM Press.

McLendon, P., Phillips, C., Helman, J. et. al. (1992), *IRIS performer programming guide,* Mountain View CA: Silicon Graphics, Inc.

Regian, J. W., (1991). Representing and teaching high performance tasks within intelligent tutoring systems. In H. Burns, J. Parlett & C. Redfield (Eds.) *Intelligent Tutoring Systems: Evolutions in Design* (pp. 225-241). Hillsdale N. J.: Lawrence Earlbaum & Assoc, Inc.

Seligmann, D. D., Feiner, S., (1991), Automated generation of intent-based 3D illustrations. In *SIGGRAPH Conference Proceedings,* (pp. 123-131). Las Vegas NV: ACM Press.

Stiles, R., Pontecorvo, M., (Sept. 1992) Lingua graphica: a visual language for virtual environments. *Proceedings of the IEEE International Workshop on Visual Languages.* (pp. 225-228). Seattle WA: IEEE Press.

Strauss, P. S., Carey, R. (July 1992), An object-oriented 3D graphics toolkit. In *SIGGRAPH Conference Proceedings,* (pp. 341-350). Chicago, IL: ACM Press.

Virtual Environments for Science Education: A Virtual Physics Laboratory'

R. Bowen Loftin (UH-Downtown), Mark Engelberg (LinCom), and Robin Benedetti (USC)
University of Houston-Downtown
One Main Street
Houston, TX 77002

Students usually enter beginning courses in physics with many misconceptions about the physical world. The "real" physics laboratory does not normally provide the means to alter "reality" in such a way as to remove impediments to acquiring a conceptually correct view of many physical phenomena. A "virtual" physics laboratory has been constructed that provides students with the ability to control the laboratory environment as well as the physical properties of objects in that laboratory. Those environmental factors that can be controlled include gravity (both magnitude and direction), surface friction, and atmospheric drag. The coefficients of restitution of elastic bodies can also be altered. Trajectories of objects can be "traced" to facilitate measurements. The laboratory allows students to measure both displacements and elapsed time. Time may be "frozen" to allow for precise observation of time-varying phenomena. A number of issues related to the observation of physical events and the interaction of students with a virtual environment have been addressed in this project. In addition to its role in allowing students to observe and control the kinematic and dynamic behavior of physical objects, this laboratory will ultimately be extended into the macroscopic and microscopic domains—giving students access to direct observations that were heretofore impossible. This same approach can also be applied to laboratories for chemistry, life sciences, and planetary sciences. This new application of computer graphics in education has the potential to augment or replace traditional laboratory instruction with approaches that offer superior motivation, retention, and intellectual stimulation.

N96- 14994

Sounds of Silence - How to Animate Virtual Worlds with Sound

Peter Astheimer
Fraunhofer-Institute for Computer Graphics (IGD)
Wilhelminenstr. 7
6100 Darmstadt
Germany
phone: ++49 6151 155 121
fax: ++49 6151 155 199
email: astheime@igd.fhg.de

Keywords

Sonification, Auralization, Acoustic Rendering, Virtual Reality

Abstract

Sounds are an integral and sometimes annoying part of our daily life. Virtual worlds which imitate natural environments gain a lot of authenticity from fast, high quality visualization combined with sound effects. Sounds help to increase the degree of immersion for human dwellers in imaginary worlds significantly.

The virtual reality toolkit of IGD features a broad range of standard visual and advanced realtime audio components which interprete an object-oriented definition of the scene. The virtual reality system "Virtual Design" realized with the toolkit enables the designer of virtual worlds to create a true audiovisual environment. Several examples on video demonstrate the usage of the audio features in Virtual Design.

1 INTRODUCTION

Virtual reality applications are mainly characterized by realtime processing capabilities in order to react immediately to user actions and by close integration of humans by novel output and interaction devices. Realized applications either focus on the imitation of real worlds (interior design, architecture, landscapes) or the creation of imaginary, abstract worlds with even different behaviour (experimental research, art).

Today many virtual reality applications concentrate on the output side on the generation of a sufficiently high image frame rate. This approach was motivated by the availability of high performance graphics workstations, adequate rendering techniques and the stimulation of one primar human sense - our visual perception capability.

Within the last years low-cost personal computer and mid-size workstations have been equipped with audio devices originally for multimedia purposes. On the other hand electronic music devices like synthesizers have made rapid progress in sound generation and usability especially with the introduction of digital components and the midi programming and controlling interface.

Although audio hardware and software is readily available today, few systems use acoustic sensations to enhance the man-machine communication in virtual reality. Acoustical simulations - in certain aspects similar to visual light simulations - contribute a great deal to a realistic impression of virtual worlds. The algorithms presented can either be applied in realtime or they can precompute acoustical characteristics which will be evaluated later in realtime.

The virtual reality system "Virtual Design" of IGD concentrates on the imitation of natural environments, i.e. offices, buildings, cities, landscapes. It is completely data-driven, i.e. the scene with all visual and aural objects and associated attributes is defined in a datafile-interface.

This paper introduces the facilities at the VR demonstration centre of IGD in Darmstadt and the architecture of the VR system "Virtual Design". The available qualities of acoustical simulations are explained and an overview over the specification of worlds with the datafile-interface is given. Furthermore the interpretation and processing of the audiovisual objects is outlined. Several examples demonstrate the usage of the system and the achievable audiovisual effects.

2 THE VIRTUAL REALITY DEMONSTRATION CENTRE OF IGD - HARDWARE FACILITIES

Since the beginning of 1993 the Fraunhofer Society has established a demonstration centre for virtual reality partly located in Darmstadt. The intention was to provide and establish a facility where different systems can be demonstrated and evaluated, where consulting, know-how-transfer and training is offered.

The demonstration centre in Darmstadt provides a high quality environment of virtual reality hardware (figure 1) to run different applications. Central node is a high performance graphics workstation (SGI RE), to which various input devices (dataglove, flying joystick, spaceball, spacemouse, 3d tracking system) and output devices (head-mounted-display, large screen stereoscopic projection, video) are connected.

The graphics workstation is linked to a multimedia-workstation and a personal computer, which controls the midi-equipment (synthesizer and effect processor), the 3d audio processing card and a power unit control card. The audio output signals of both machines are combined in a mixer, which also includes the signals of a compact disk player, a digital audio tape, a microphone and a video player. The mixer drives speakers as well as the headphones in the head-mounted-display via the 3d audio card.

There are many different processing tasks to solve in virtual reality applications: the simulation of world behaviour, the computation of the visual and acoustic representation, the management and access of peripheral devices and the interpretation of user actions. As there is no integrated machine available which satisfies all needs, specialized computing nodes distributed in a net offer optimal use of processing power. Besides the high performance graphics workstation a compute server (Convex) for simulation purposes and a multimedia workstation (SGI Indigo), 3d audio processing hardware (focal point) and midi-equipment (Akai S1000, Alesis QuadraVerb) for sound rendering are available.

To manage and utilize the full processing power of a heterogenous net a distributed system is required. In Virtual Design a sound server addresses multimedia workstations as well as midi-devices via a common programming interface. The server stores a table of digitally sampled sounds, which can be manipulated.

The multimedia-workstation permits calculation (e.g. the convolution of impulse responses with dry sound samples) and subsequent rendering of digital sounds plus basic manipulation (volume). The midi-equipment consists of a synthesizer and an effect processor. The synthesizer permits the realtime manipulation of volume, frequency and stereo level. The effect processor has a number of controllers to simulate the reverberation of a sound in an enclosure. Adjustable parameters of the effect processor are:

- enclosure type (room, hall, echo chamber),
- decay time,
- degree of diffusion (number of reflections),
- reverberation density (density of impulse response),
- low frequency decay,

- high frequency decay.

The devices installed at the demonstration centre allow the presentation of virtual worlds to a large audience via stereoscopic large screen projection and speakers as well as an individual self-experience via head-mounted-display and headphones (also simultaneous).

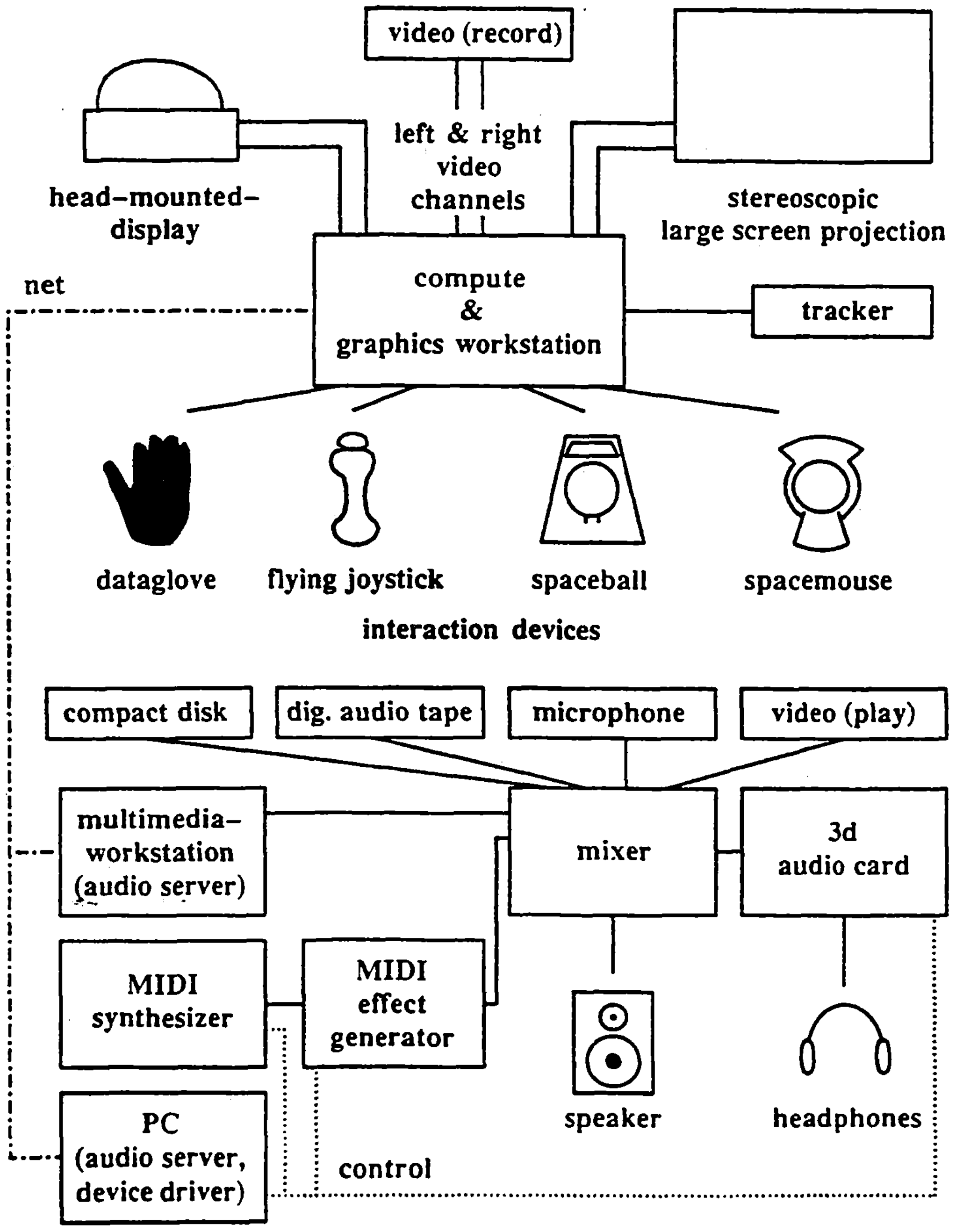


Figure 1 Facilities at IGD's demonstration centre in Darmstadt

3 QUALITIES OF ACOUSTICAL RENDERING

To animate virtual worlds with sound which have been designed as models of our real world, realistic sounds of our environment have to be integrated. Common environmental sounds are discrete (steps on the floor, door closing etc.) and continuous sounds (teapot boiling, drilling etc.) caused by material interaction (sound events), vibrating objects (machine parts), liquids (splashes), aerodynamic sounds (exploding balloon) or sounds from loudspeakers (radio, tv). Sounds are also used for alarms (fire alarm), warnings (r.p.m. of engine getting too high) and messages (engine going). Sounds are damped through objects (windows), they can "crawl" around objects (pillar, especially low frequencies) and the reflections are mostly specular (due to wavelength). A frequency shift can be noticed from passing objects (police car with siren). Humans can detect the position of a sound source in space (azimuth better than elevation).

To satisfy at least some of the above mentioned acoustic effects Virtual Design offers several acoustic rendering qualities. The room acoustical simulation algorithms have many parallels to visual simulation algorithms, the main differences caused by the different propagation speed and wavelength of sound and light and the time resolution.

Sound events: An object is assigned an acoustic material which has a reference to a sound source. Ideally a matrix of sounds should be stored containing all possible collision sounds between moving objects. In the toolkit this is reduced to a list for the cursor (fingertip of dataglove echo) and all collision-sensitive objects. A sound source indexes a sound stored in the sound server. The toolkit cycles an action loop to update image frames; in each cycle collisions are tested and a callback function is eventually evoked. When an object has an audio event attribute the referenced sound source is triggered. Examples show a drum kit which can be played with the dataglove and an assembly/disassembly system (3d puzzle).

Direct propagation: In our daily life three acoustic parameters can be noticed when sound propagates directly from a sound source to a receiver: volume damping (caused by distance and medium), frequency shift for moving objects (doppler effect) and position in space. If a sound source, a receiver and an object have been assigned direct propagation attributes, these objects are passed for computation of the audio parameters to the algorithm. Then the parameters are passed for auralization to the sound server. An example shows a beetle proceeding over a plain.

Statistical approach: This approach is based on assumptions stated by W.C. Sabine. The algorithm computes an average reverberation time independent of receiver and sound source position. Thus it is possible to characterize a room type, e.g. a living room, a conference room or a church hall. The computed parameter value configures the effect processor, which performs the reverberation in realtime. Whenever an alternative acoustic model is chosen, the reverberation value is recalculated. An example shows a conference room with different reverberation characteristics.

Room Acoustics: Room acoustical simulations compute the energy distribution of an energy pulse emitted from a sound source and detected by a receiver within an enclosure. The simulation considers the exact geometry with reflections, absorption, dispersion and air turbulences (wind). The receiver detects the so-called impulse response (figure 2) which can be convoluted with any dry sound sample.

The image source algorithm mirrors actual sound sources at enclosing faces, which can be executed quite easily for rectangular rooms. In Virtual Design the user navigates to the desired listening position, then the impulse response is calculated at this position, visualized and sent to the audio server for convolution. The original, dry sound and the convoluted sound can both be rendered to compare the results.

Ray tracing algorithms trace the propagation of sound particles or rays in an arbitrary world. They are similar to visual simulation algorithms in respect to ray propagation and intersection calculation.

Database: The acoustic renderer require additional objects like sound sources, receivers and acoustic material attributes. Visual and acoustic renderer are in many respects very similar; they both operate on geometrically defined scenes. It is the different propagation speed of light and sound which causes algorithmic differences. The renderer compute parameters which are used by an audio server to manipulate digitally sampled sounds.

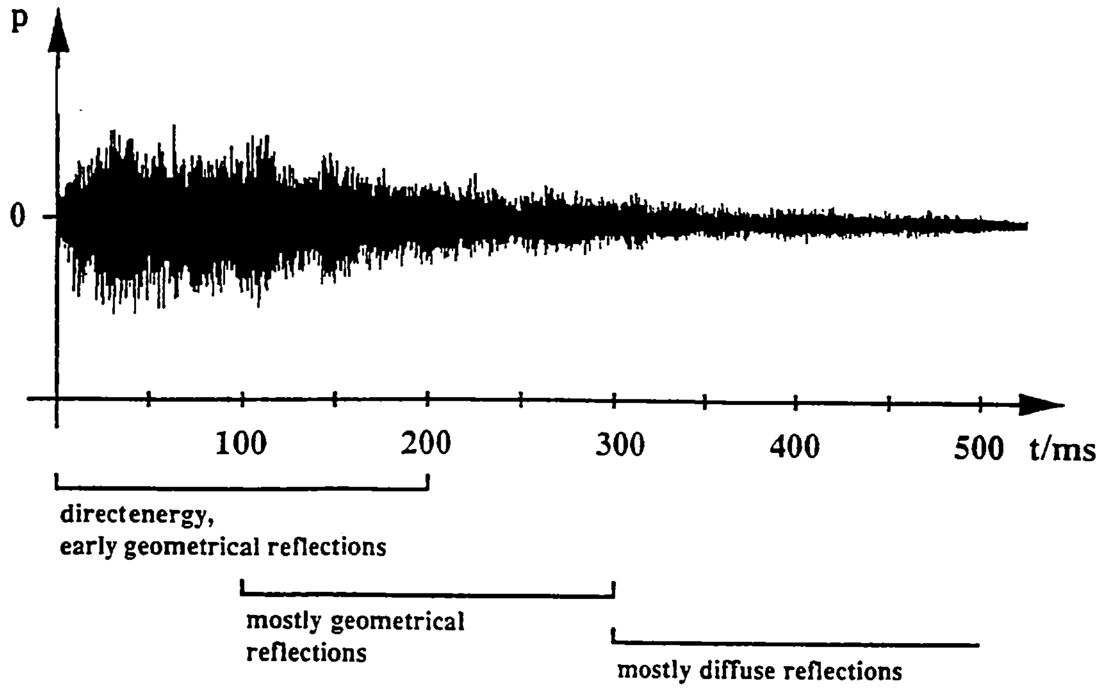


Figure 2 Impulse response

The visual model comprises polygonal objects, lights, camera and a scene description with multiple object references. Additional acoustic objects supplement the visual model:

- medium (propagation speed of sound, damping),
- acoustic material (local reflections, sound),
- sound source,
- listener (can be coupled with the observer),
- impulse response,
- sound sample format.

The medium and material data are held in a table with some predefined entries like air medium or carpet material. Objects have visual as well as acoustic attributes; they can be classified to be processed by the desired renderer. A datafile-interface (see chapter 4) provides a convenient definition of the visual as well as the acoustic scene model. When all model description data is loaded into the database, a cyclic rendering process (see chapter 5) is started. In a single cycle each object is inspected, the attributes are evaluated and the object is passed to the specified visual or acoustic renderer for further processing. Typically all visual objects are rendered and acoustic parameters are computed which will alter e.g. a continuous sound.

4 THE DATAFILE INTERFACE

An application processed by Virtual Design is defined by a set of environment variables and data files. The environment variables specify device ports, output screen size and device configuration. The model and interaction modes are defined in various data files:

Control file: References all other data files (root file). Moreover a lot of parameters can be specified to define the coordinate system, background colour, logo, dataglove representation and tracking parameters, alternative object representations, collision-sensitive objects, images, menu configuration and appearance. Functions can be assigned to menu entries like fly mode, converter level, automatic camera recording, toggling of system states, triggering of sound sources and acoustic rendering algorithms.

Scene file: References and places all objects in the scene, sets objects attributes and transformations, camera and lights.

Visual definition files: Defines the object geometries (set of polygons), attributes, colours and textures.

Acoustic definition file: Defines receivers and sound sources with attributes, acoustic object properties and configures the audio server.

Camera file: Defines a set of specifications for the camera (observer), can be used to load and replay prerecorded walkthroughs.

Animation file: Defines a set of transformation matrices for objects. With these data objects can be animated in a world (the beetle for instance).

To integrate geometric models into Virtual Design a number of CAD formats of various modelers including Autodesk's AutoCad are supported.

5 VIRTUAL DESIGN - A FRAMEWORK FOR VIRTUAL REALITY APPLICATIONS

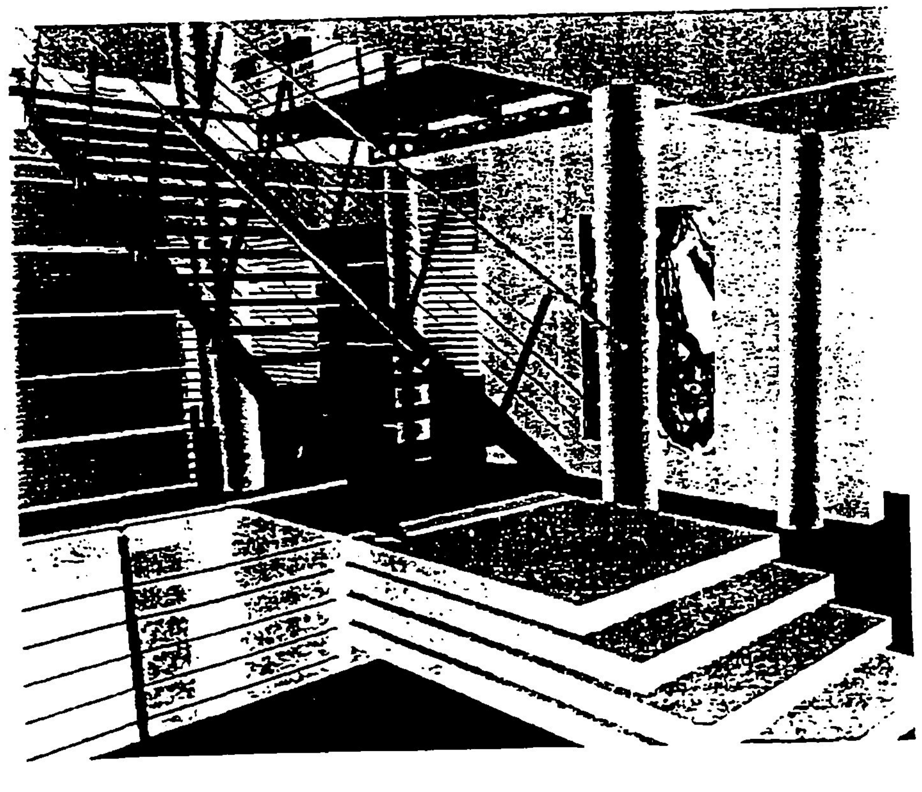


Figure 3 Snapshot of a walkthrough in interior design (CAD data by Asplan, postprocessing by CAD & Art)

Virtual Design provides a convenient and powerful development platform for virtual reality applications (figure 3).

Virtual Design is an example for a system with abilites corresponding to aspects specified near the end of the axis of the DIP-model [EAFF-93]. At the heart of the virtual reality toolkit of IGD a central database is located. Many modules with different tasks operate on the database and manipulate database objects (figure 4).

Objects can be attributed as collision-sensitive.These objects are tested each frame against a cursor object (forefinger of dataglove object for instance). Whenever a collision is detected, a callback function is evaluated to perform a predefined action [Zach-92].

The tracking system transforms device-specific position and orientation data in space into a world coordinate system, which can be interpreted by the navigation module. This module computes the transformation matrices for the observer object and interaction objects driven by devices like the dataglove [FeFG-93].

Human hand gestures detected by dataglove sensors are compared against a table of predefined gestures. Upon recognition the appropriate action function is evoked [Wirt-92].

A radiosity system can be used to precalculate colour values of polygon patches. An integrated ray tracer computes photorealistic images, which can be shown as still frames during a walkthrough as an example for high quality rendering [MüUG-93]. The radiosity system features also level-of-detail object hierarchies, visibility culling, context-sensitive object transformation and time management of frame display rates. It serves as a testbed for further enhancements of the toolkit.

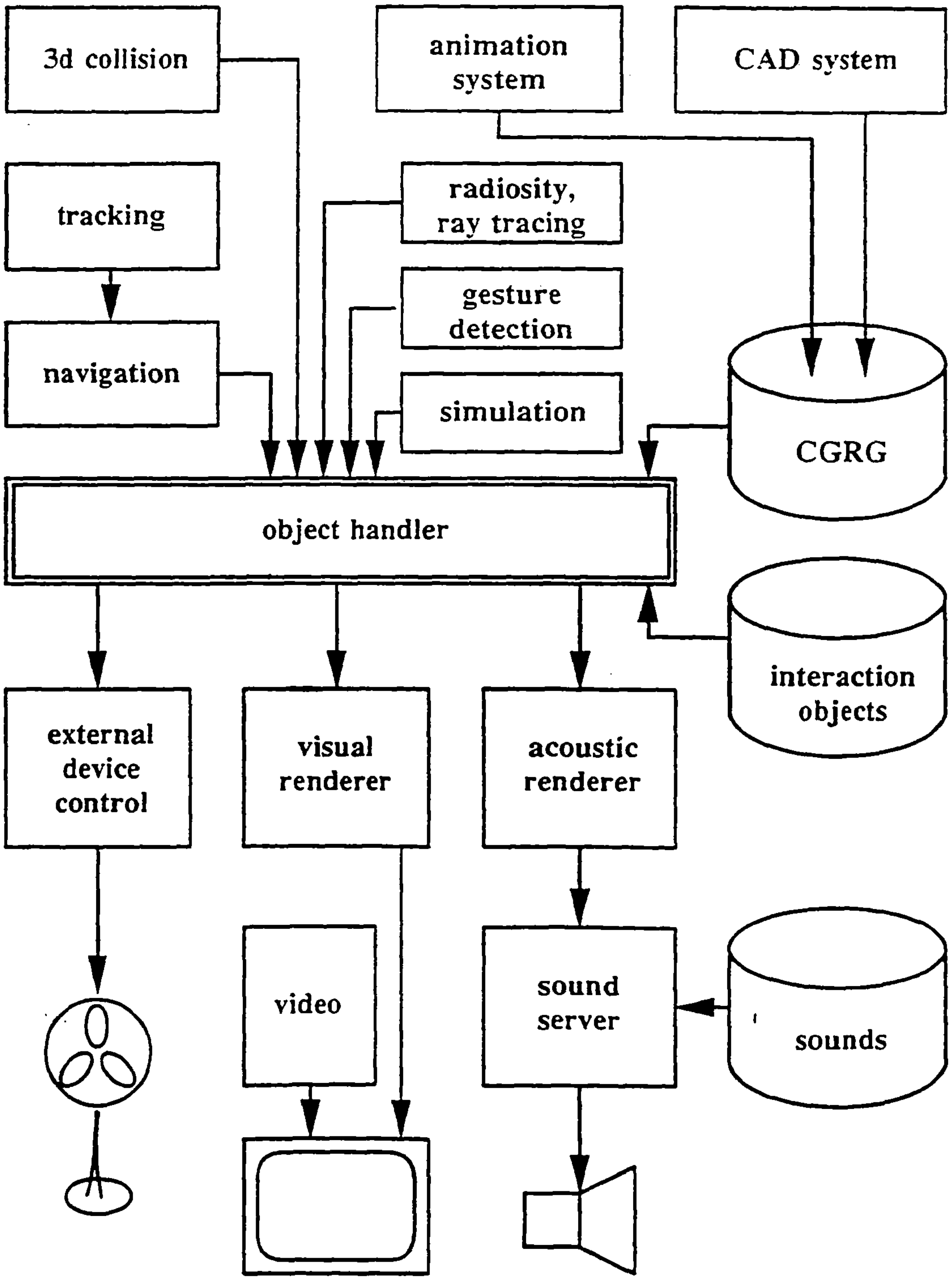
Standard modelers like AutoCad can be used to create and update a scene model. A number of CAD object formats can be imported and loaded into the database to define the scene geometry and the visual objects with their attributes. Additional objects like a hand model and a 3d menu object are included by default for visual feedback of interaction.

A sophisticated rendering system provides an object database interpreted by several visual and acoustic renderer with realtime processing capabilities. The object database comprises up to now for visualization a two level hierarchy: For modelling purposes the objects are grouped into sets, for rendering purposes the objects are transformed into a flat, referenceless but fast interpretable cache structure. To master very large and complex scenes visibility culling, level-of-detail scene analysis and a hierarchical database is currently under development. The visual renderer uses SGI's GL to achieve the necessary performance.

The acoustic renderer require additional objects like sound sources, receivers and acoustic material attributes. The rendering system currently comprises four different acoustic renderer [AsGö-93]. The renderer compute parameters which are used by an audio server to manipulate digitally sampled sounds.

A programmable video-recorder integrates motion pictures into virtual reality applications. A peripheral device controller steers AC-powered electric devices like fans to create the illusion of movement.

With these modules a sample virtual reality system "Virtual Design" has been realized, which permits the processing of highly differing applications. Virtual Design is extremely flexible and can be configured through a data interface. Possible interactions are specified with a configurable 3d menu.



F

Figure 4 Module architecture of Virtual Design

The basic structure of Virtual Design in respect of acoustic rendering is as follows:

```
main:

    load system configuration
    load scene
    initialize peripheral devices (interaction device, tracker, audio server)

    if (any object with statistic attribute)
        evoke statistic renderer
        send reverberation value to audio server

    navigation loop

        get navigation data
        render next frame

        if (any objects with direct attribute)
            send play (object->source, volume, frequency, 3d position) to audio server

        if (menu evoked)
            disable navigation
            evaluate selected menu item:

                image source renderer:
                    evoke image source renderer
                    visualize impulse response
                    send impulse response to audio server

                trigger sound source:
                    send play (source) to audio server

                trigger convoluted sound:
                    send play convoluted sound to audio server

            enable navigation

        if (collision detected)
            if (detected object has event attribute)
                send play (object->source) to audio server
```

6 Examples

The acoustic rendering qualities described in chapter 3 are demonstrated in examplary virtual world applications, which can be seen on the supplemental video tape. Some snapshots from these virtual world explorations are shown below. Each application is generated by loading and interpreting file data for the visual and acoustic scene.

In figure 5 an example for the application of sound events is pictured: a drum set placed in the middle of a virtual room can be played with the dataglove.

Figure 6 shows an example of an assembly/disassembly system with sound feedback. Notifying sounds are releasen when a part is grabbed, when two parts collide and when they fit together and finally an applause rewards the lucky user.

Figure 7 pictures a beetle standing in the middle of a plain. When the simulation is switched on, the beetle passes several times the observer (equals listener). Change of volume, frequency shift and stereo position can be noticed.

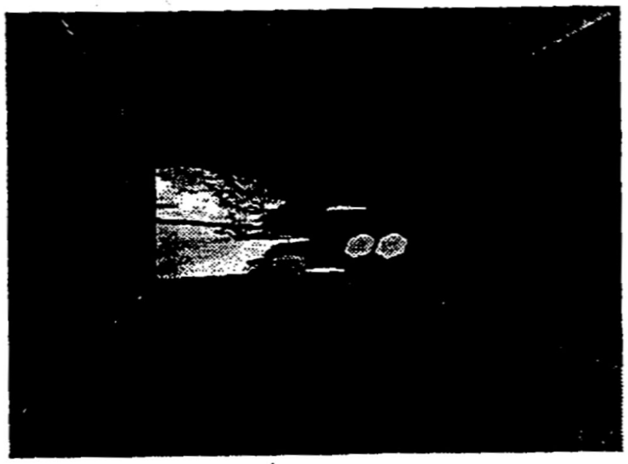


Figure 5 Drum kit

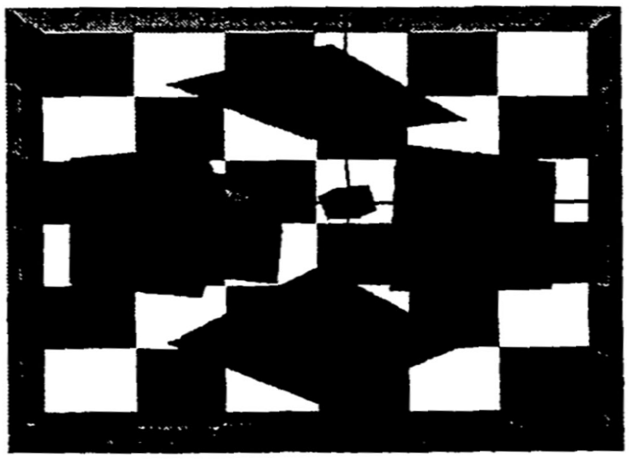


Figure 6 Puzzle

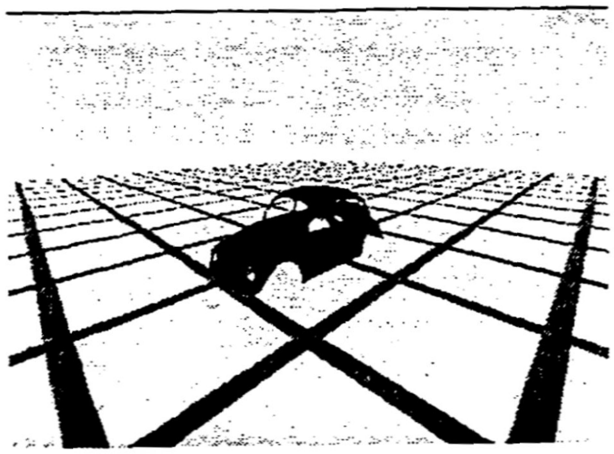


Figure 7 Passing beetle

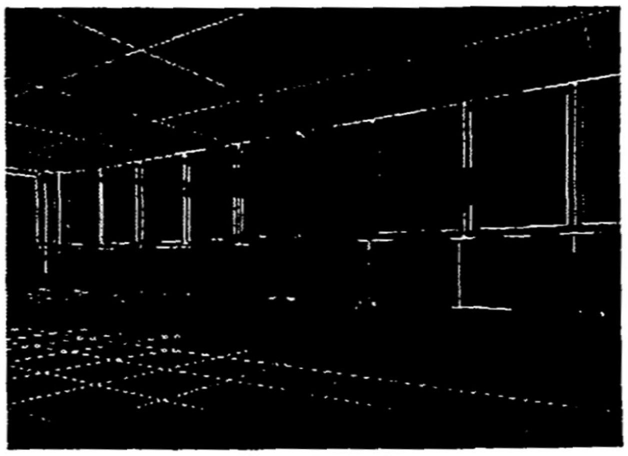


Figure 8 Image source algorithm

Figure 8 shows a snapshot from an audiovisual virtual world, where the visual model is shown in wire frame quality, the acoustic model in solid representation with absorption attribute mapped onto a blue colour scale. In the center of the model the calculated impulse response is visualized; the vectors depict direction and detected energy of a sound source, the colour depicts the arrival time on a red (early) to blue (late) colour range.

All examples have been realized with predefined scenes modeled with standard CAD packages. Acoustic objects with attributes (receivers, sound sources) and geometry-related attributes as absorption coefficients have been added in the acoustic datafile.

7 Conclusion and Future Work

The integration of audio features in virtual reality applications contributes a great deal to a natural simulation of imaginary worlds. It leads a step further to the ultimate goal to stimulate all human senses and let humans totally immerse in virtual-worlds. With the Virtual Design system of IGD a variety of audiovisual worlds are easily defined and presented for exploration to the user.

Standard CAD packages and animation systems allow the specification of scenes with objects, geometry, visual attributes, camera parameters and light sources; they do not support the interactive modeling of acoustic object attributes, receivers and sound sources. The extension of available modelers with acoustic related functions will be one area of research in future. Furthermore the integration of a particle tracer, a geometry simplification module and the connection and application of appropriate signal processing hardware for binaural convolution of sounds in realtime will be evaluated.

8 References

[AsFe-91] Astheimer, P., Felger, W.: *Application of Animation and Audio in Visualization Systems*, in: Second Eurographics Workshop on Visualization in Scientific Computing. Delft, Netherlands, 22-24 April 1991.

[AEFF-92] Astheimer, P., Encarnacao, J.L, Felger, W., Frühauf, M., Göbel, M., Karlsson, K.: *Interactive Modeling in High Performance Scientific Visualization*, Special Issue of Computers in Industry "Modeling in Computer Graphics", Vol 19, No 2, pp. 213 - 225

[AsGö-93] Astheimer, P., Göbel, M.: *Integration akustischer Effekte und Simulationen in VR-Entwicklungsumgebungen*, Proceedings VR '93, Stuttgart, Februar 1993

[Asth-92a] Astheimer, P.: *Sonification Tools to supplement Dataflow Visualization*, in: Third Eurographics Workshop on Visualization in Scientific Computing, Viareggio, Italy, April 1992 (also in: Scientific Visualization - Advanced Software Techniques, Patrizia Palamidese (ed.), Ellis Horwood Workshop Series, London, 1993, pp. 15 - 36)

[Asth-92b] Astheimer, P.: *Sonification in Scientific Visualization and Virtual Reality Applications*, in: Visualisierung, Rolle von Echtzeit und Interaktivität, GI Workshop, St. Augustin, Juni 1992

[Asth-93] Astheimer, P.: *Realtime Sonification to enhance the Human-Computer-Interaction in Virtual Worlds*, Fourth Eurographics Workshop on Visualization in Scientific Computing, Abingdon, England, April 1993

[Broo-90] Brooks, F.P., Ouh-Young, M., Batter, J.J., Kilpatrick, P.J.: *Project GROPE - Haptic Displays for Scientific Visualization*, in: ACM Computer Graphics, Vol. 24, No. 4, August 1990, pp. 177-185

[Carr-90] Carrabine, L.: *Plugging into the Computer to Sense*, Computer-Aided Engineering, June 1990, pp. 16-26

[Cric-93] The Cricket, Product Information, Digital Design Incorporated, New York, 1993

[EAFF-93] Encarnacao, J.L., Astheimer, P., Felger, W., Frühauf, T., Göbel, M., Müller, S.: *Graphics and Visualization: The Essential Features for the Classification of Systems*, Proceedings ICCG, Bombay, February 1993

[Felg-92] Felger, W.: *How interactive visualization can benefit from multi-dimensional input devices*, in: Alexander, J.R., (Ed.), Visual Data Interpretation, SPIE 1992

[FeFG-93] Felger, W., Fröhlich, T., Göbel, M.: *Techniken zur Navigation durch virtuelle Welten*, Proceedings VR '93, Stuttgart, Februar 1993

[Fish-90] Fisher, S.S., Foster, S.H., Stone, P.K., Wenzel, E.M.: *A System for three-Dimensional Acoustic "Visualization" in a Virtual Environment Workstation*, in: Kaufman, A. (ed): Proceedings of Visualization '90,IEEE Computer Society Press, Los Alamitos, 1990

[Frys-84] Frysinger, S. et.al.: *Dynamic Representation of multivariate time series data*, Journal of the American Statistical Association, 1984

[Kutt-73] Kuttruff, H.: *Room Acoustics*, Applied Science, London 1973

[MüUG-93] Müller, S., Unbescheiden, M., Göbel, M.: *Genesis - Eine interaktive Forschungsumgebung zur Parallelisierung des Radiosity-Verfahrens für die virtuelle Welt*, Proceedings VR '93, Stuttgart, Februar 1993

[ScCr-91] Scaletti, C., Craig, A.B.: *Using sound to extract meaning from complex data*, in: Farrell, E.J. (Ed.): Extracting Meaning from Complex Data: Processing, Display, Interaction II, Proc. SPIE 1459, (1991)

[Shi-91] Shi, J.: *On Integration of 3D Visual and Acoustical Rendering*, internal report, FAGD-91i013, 1991

[StGr-89] Stettner, A., Greenberg, D.P.: *Computer Graphics Visualization For Acoustic Simulation*, Computer Graphics, Vol 23, No 3, July 1989, pp. 195 - 206

[Smit-91] Smith,S. et.al.: *Global geometric, sound and color controls for iconographic displays of scientific data*, in: Farrell, E.J. (Ed.): Extracting Meaning from Complex Data: Processing, Display, Interaction II, Proc. SPIE 1459, 1991

[Wirt-92] Wirth, H.: *Integration von multidimensionalen Eingabegeräten in 3D-Graphiksysteme*, Diplomarbeit, TH Darmstadt, Januar 1992

[Zach-92] Zachmann, G.: *Drei-Dimensionale Buttons*, Technical Report, FhG-IGD, Juni 1992

Biography:

Peter Astheimer is a staff member of the Fraunhofer-Institute for Computer Graphics (IGD) since 1987. He received the diploma degree in 1987 from the Technical University of Darmstadt. He currently works on the integration of audio features in visualization and virtual reality systems.

N96- 14995

LIVING COLOR FRAME SYSTEM: PC GRAPHICS TOOL FOR DATA VISUALIZATION

Long V. Truong

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

Living Color Frame System (LCFS) is a personal computer software tool for generating real-time graphics applications. It is highly applicable for a wide range of data visualization in virtual environment applications. Engineers often use computer graphics to enhance the interpretation of data under observation. These graphics become more complicated when "run time" animations are required, such as found in many typical modern artificial intelligence and expert systems. Living Color Frame System solves many of these real-time graphics problems.

INTRODUCTION

Lewis Research Center is NASA's lead center in space power technology. Our technologies include the use of artificial intelligence (AI) for power management and distribution. Specific projects in AI are power scheduling, fault detection and isolation using expert systems [1,2], neural networks, and fuzzy logic.

Power AI projects typically require real-time computer graphics that involve graphics image animation, such as alternating video colors and intensities and moving objects during run time. In some of our applications these images display the system hardware diagram and its real-time status (Fig. 1). The system process control variables must be displayed along with these graphical "frames." It is also desirable to be able to zoom in and out at any level of detail of these object images (hardware components) and to have the personal computer (PC) buzzer sound when operator response is urgently requested.

Because considerable effort has been required to generate these real-time graphics applications, the need for an easy graphics generation and management tool became apparent. Although there are many sophisticated commercial graphics tools, such as Freelance (trademark of Lotus Development Corp.) and Quattro Pro (trademark of Borland International Inc.), none support or provide adequate run-time software for graphics manipulation and/or disclose their graphics file formats for user implementation of real-time applications. Because such a tool was not available for PC use, we designed a new tool to eliminate custom graphics programming and to simplify the development of our application software. It is called the Living Color Frame System (LCFS).

This paper introduces the LCFS and presents an example of real-time graphics applications in the area of monitoring and diagnostics for an electromechanical actuator system (Fig. 1).

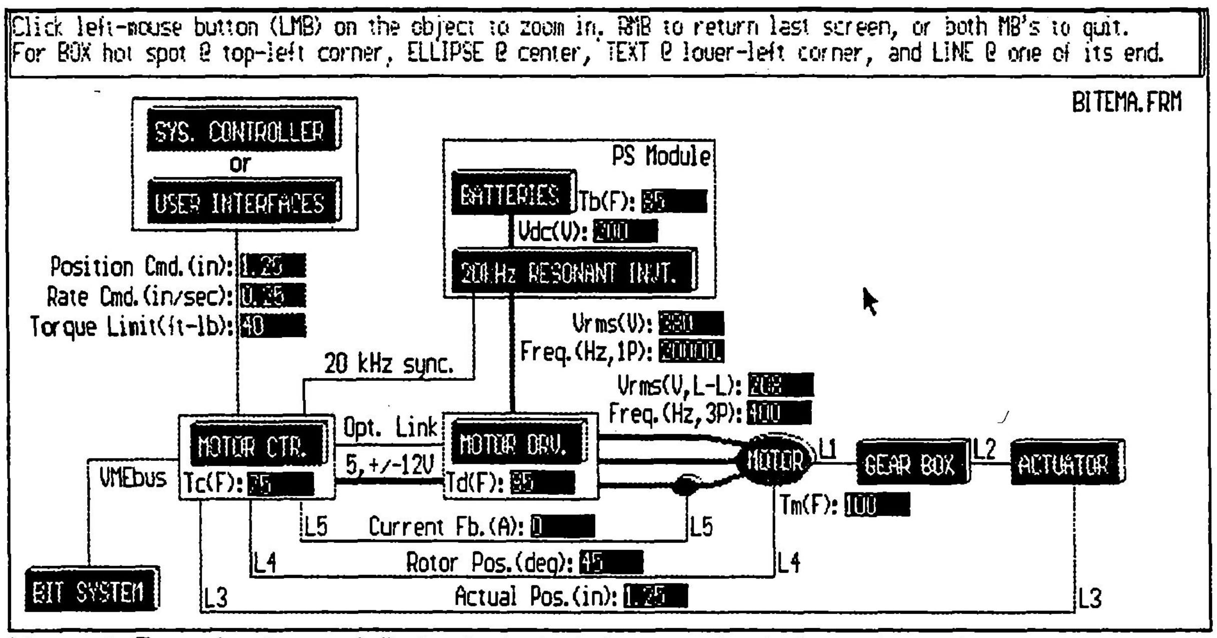

Click left-mouse button (LMB) on the object to zoom in, RMB to return last screen, or both MB's to quit.
For BOX hot spot @ top-left corner, ELLIPSE @ center, TEXT @ lower-left corner, and LINE @ one of its end.

Figure 1 Example of normal display for typical electromechanical actuator system.

LIVING COLOR FRAME SYSTEM

I. ABOUT THE SYSTEM

Living Color Frame System, broadly speaking, is a PC software tool for real-time graphics applications. It is highly applicable for data visualization in virtual environment applications. This new graphics tool, LCFS, is user friendly, having a graphical interface, mouse-driven controls, and on-line instructions. Computer graphics screens or "frames" can be conveniently drawn, logically linked, and then dynamically recalled for display in real time by using mouse commands. Run-time software for custom video and sound effects on these frames are provided, such as display of monitored data, manipulation of graphics images, and/or sounding of a PC buzzer. In some of our applications these frames can be visual aids for managing systems. For monitoring, diagnosing, and/or controlling purposes circuit or system diagrams can be brought to *life* by using designated video colors and intensities to symbolize the status (feedback from sensors) of hardware environments. With LCFS, custom graphics programming is largely eliminated, allowing software developers to focus fully on their applications' contents. Thus, LCFS is suitable for a wide range of real-time graphics applications.

II. ITS SOFTWARE STRUCTURE

Figure 2 shows a simplified block diagram of LCFS's software structure. It basically describes the integration of the software modules (oval shapes) and the flow of information (rectangular shapes) throughout the system. In the next three sections LCFS's software structure is discussed in terms of user application software, LCFS's built-in software, and frame data bases, respectively.

1. User Application Software

As shown in Figure 2, user application software is divided into two modules: data communication and data processing. They are described here.

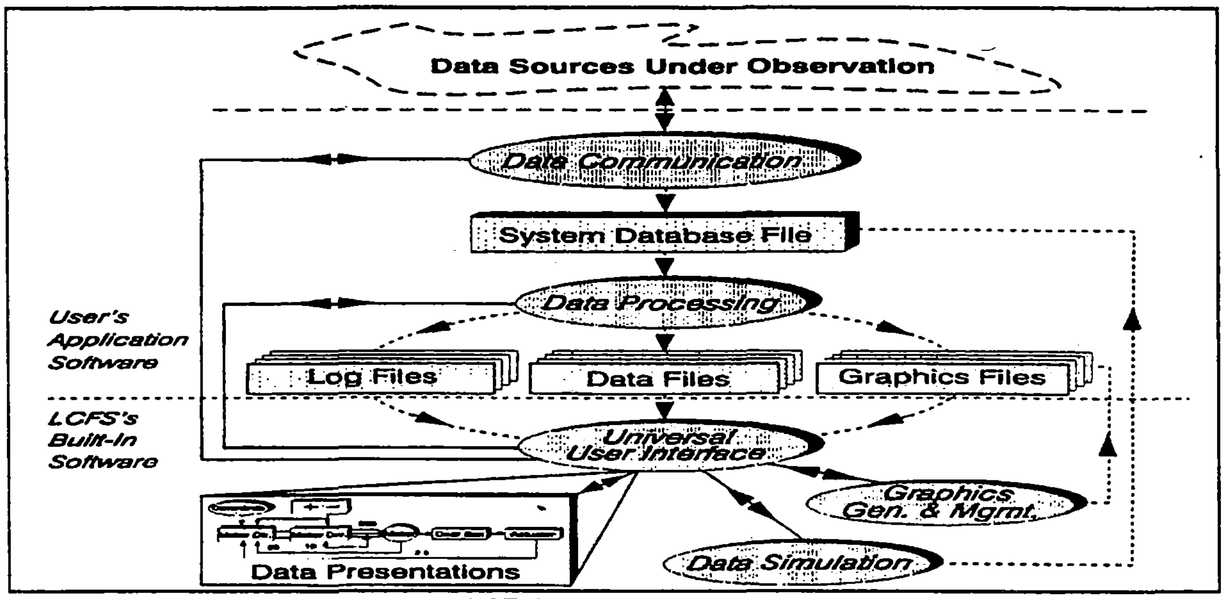


Figure 2 Simplified block diagram of LCFS's software structure.

1.1. Data communication (DC) module.- Because each application is unique, users are responsible for the function of the DC module (Fig. 2). Its primary task is to transfer raw monitored data from their sources (Fig. 2) to the system data base file (see section 3.1 for its format and creation). This executable module must be named "DATACOMM.EXE" in order for the system to recognize it as a "child" process during the data updating procedure. Optional routines for hardware control should also be implemented here.

1.2. Data processing (DP) module.- For each "static" graphics screen or file generated by the graphics generation and management module (Fig. 2), users have an option to develop a run-time interfaced program for linking and displaying additional information. Most of the software routines for this option are provided, such as superimposing numerical monitored data, manipulating graphics images to reflect the current data, and sounding the PC buzzer to get the operator's attention. These interface programs (data processing module, Fig. 2) must have the same names as their "related" graphics files for the system to acknowledge them. They can be added or deleted without interfering with LCFS's built-in software. Their typical tasks would be

- To read (routines are provided) data from the system data base file
- To analyze (user application rules) the data
- To select and prepare (routines are provided) frame information for display. Thus, run-time graphics, numerical data, and/or text messages are dynamically constructed for display in this step. See section 3 for implementations.

2. LCFS's Built-In Software

As shown in Figure 2, this permanent part of the system software is divided into four modules: universal user interface, data presentations, graphics generation and management, and data simulation. They are discussed here.

2.1. Universal user interface (UUI) module.- This unique user interface module (Fig. 2) controls the entire system operations, manually or automatically. In automatic operation mode it continuously updates frame information by repeatedly executing three responsible tasks: data communication, data processing, and data presentation. On the other hand, in manual operation mode (default), it allows users random access to many convenient built-in options. At present these options are as follows:

- Display UUI's instructions (click the right-mouse button).
- Zoom in or out (click the left-mouse button on or off the frame objects).
- Set automatic operation mode (press the A key).
- Set manual operation mode (press the M key).
- Turn off the PC buzzer's sound (press the S key); see section 3.3 for activating the PC buzzer.
- List run-time messages (press the L key); see section 3.2 for implementations.
- Generate or modify graphics screens or files (press the F key); see section 2.3 for usage.
- Simulate data (press the D key); see section 2.4 for usage.
- Exit LCFS (press the Q key).

2.2. **Data presentations (DPRE) module.-** This module (Fig. 2) is responsible for refreshing frame information. Its functions are as follows:

- Spawn (executes as a child process) the data communication module to update monitored data in the system data base file (see section 3.1).
- Spawn the data processing module to prepare frame information for display.
- Display frame information.
- Return to UUI module.

2.3. Graphics generation and management (GG&M) module.- This module is a graphics utility [3,4] for custom generation and management of computer graphics screens. Drawing options are fully displayed on a single menu and are executed by using mouse commands for fast and easy operation of the tool. Once an option is selected (click the left-mouse button on the desired menu item), users will be assisted with the on-line instructions to complete the process. Figure 3 shows some of the typical drawing elements. Its custom graphics file format (see section 3.4) is opened for user applications.

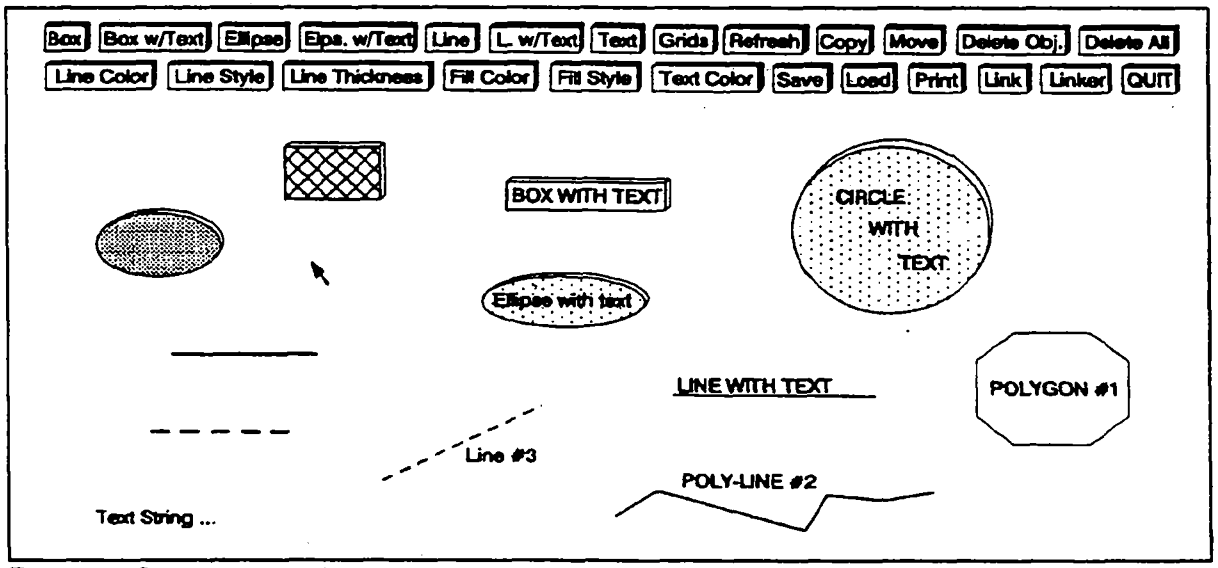


Figure 3 Samples of basic drawing elements.

2.4. Data simulation (DS) module.- This module (Fig. 2) allows users to modify the contents of the system data base file (see section 3.1) from the keyboard without programming. Thus, raw data can be conveniently simulated. Users can use this advantage to develop and test their application software independently from the target system.

3. Frame Data Bases

As mentioned in section 1.2, frame data bases or files (Fig. 2; can be printed out by any text editor) are needed for holding frame information. Four types of data files were designed for the purpose. They are discussed here.

3.1. System data base file.- This "master" data file (Fig. 2), named "SYSDBASE.DAT," was designed for holding any numerical data that users desire to access during the data processing procedure (see section 1.2). Routines for run-time creation of this file (mentioned in section 1.1) are provided. Figure 4 shows a typical listing of the file.

```
2.25
5.00
125
...
45.75
```

Figure 4 Typical listing of system data base file.

3.2. Frame log files.- By design, frame log files (Fig. 2) are created (routines are provided) during data processing procedures (see section 1.2) for holding run-time messages. These messages are optionally implemented by users to instruct operators in difficult situations, such as in an alarm situation. These files are designated with a common file extension, .LOG, and have the same names as their related frame graphics files. An example of a typical frame log file (refer to Fig. 8 for better understanding of the text) is shown in Figure 5.

Fault Detection:
ACTUATOR is not at its commanded position within the given time.

Fault Isolation:
A possible failure in one of the mechanical links between MOTOR and GEARBOX, or GEARBOX and ACTUATOR results in an immobilized actuator.

Corrective Suggestions:
1. Shut down the power immediately.
2. Check for visible damage between the interconnections of MOTOR, GEARBOX, and ACTUATOR. Repair the damage, if found. If the interconnections appear normal, proceed to the next step.
3. Disconnect the couplings between MOTOR, GEARBOX, and ACTUATOR. Then, independently check for proper operations of MOTOR, GEARBOX, ACTUATOR, and the mechanical links (L1 and L2) between them.
...

Figure 5 Typical listing of frame log file.

3.3. Frame data files.- Being the same as frame log files, frame data files (Fig. 2) are also generated (routines are provided) during the data processing procedure (see section 1.2) and named after their related frame graphics files with a common file extension, .DAT. They are designed for holding run-time numerical data to be displayed along with color and sound instructions for highlighting purposes.

Figure 6 shows a typical listing of the file. Each record (112 bytes max.) includes seven fields and spaces as field separators.

```
021 132 15 0 4 1.250000 Position Cmd.(in.):
010 162 15 0 4 40.00000 Torque Limit(ft-lb):
...
190 263 12 1 4 55.00000 Current Fb.(Amps):
```

Figure 6 Typical listing of frame data file.

Individual fields (from left to right) are explained as follows:

- Fields 1 and 2 contain the position (x-y screen coordinate) of the text label (field 7) for the numerical data (field 6) to be displayed. Since the label is created by the GG&M module, this information (fields 1, 2, and 7) can be extracted directly from the related frame graphics file.
- Field 3 contains the background color (2 bytes max.; computer standard color codes (0-15)) of the numerical data (field 6) to be displayed. Designated background colors can be used for highlighting ranges of data.
- Field 4 contains the sound flag (1 byte, 0=disable, 1=enable). This feature is excellent in getting the operator's attention by sounding the PC buzzer. It works nicely with the color option in field 3.
- Field 5 contains the number (2 bytes max.) of digits of the numerical data (field 6) to be displayed, for sizing of the display area.
- Field 6 contains the numerical data (any real number) to be displayed.
- Field 7 contains the label (mentioned in fields 1 and 2; 80 bytes max.) of the numerical data (field 6) to be displayed.

3.4. Frame graphics files.- A custom frame graphics file format (designated file extension, .FRM) and run-time supported software for graphics manipulation were designed and developed for solving real-time graphics problems. Unknown graphics file format and the lack of software support in many commercial graphics tools (e.g., Lotus Freelance Plus and Quattro Pro) had made it difficult or impossible to access the graphics knowledge base from user programs in many typical real-time graphics applications.

Figure 7 shows a typical listing of a frame graphics file that allows easy access and tremendous saving of computer memory in both run time (random access memory) and storage (hard disk). Each drawing object is saved as a record (125 bytes max.), including 13 fields and 12 bytes of spaces as field separators.

```
T    noname  21 132  1  1 10  0  1  1 10 15 Position Cmd.(in):
T    noname  10 162  0  0 10  0  1  1 10 15 Torque Limit(ft-lb):
T    noname 190 263  0  0 10  0  1  1 10 15 Current Fb.(A):
...
L      L1 429 227 451 227 10  0  1  1 10 15 L1
```

Figure 7 Typical listing of frame graphics file.

To start the tool, simply type the command LCFS at the DOS prompt [5] and hit the Enter key. After successful completion of the command, first-time users can click the right-mouse button for instructions. Thus, users are assisted with the custom on-line instructions throughout the operations. Other usage instructions (installing software, using supported software, etc.) are provided with the software package.

REAL-TIME GRAPHICS APPLICATION EXAMPLE

As previously mentioned, the application example given pertains to the area of monitoring and diagnosis (M&D) of an electromechanical actuator (EMA) system. A simulated top-level diagram of such a system was shown in Figure 1.

Because of the nature of the electrical power system, such as shown in Figure 1, electricians normally use hardware schematics as "roadmaps" for troubleshooting problems. What could be better than having "live" schematics on the computer screen for M&D? An essentially live schematic can be achieved by using computer images with designated colors, video intensity, and sound to symbolize the component hardware status (real-time feedback from sensors), and therefore the status of the system itself. Table 1 defines these image representations or models for the example below. Note that only black-and-white shadings are shown to represent the actual colors in illustrated figures.

Table 1 Designated Image Representations of Colors and Sound Denoting System Status.

Image Representations	Status of Hardware Components/Modules
Green	NORMAL, no failure diagnosed (requires no attention)
Yellow	WARNING, a "side effect" or minor failure, temporary or permanent, diagnosed (usually requires attention)
Red and beeping	ALARM, a "hard" or serious failure diagnosed (requires immediate attention)

For the application example an alarm situation for the EMA system was simulated and is shown in Figure 8. Note the differences in "colors" (different shadings in black and white) from the normal status display of the system in Figure 1. By looking at this color graphics presentation (Fig. 8) and having the knowledge from Table 1, *operators are quickly aware of the alarm situation without losing time to analyze the data (numerical numbers), especially when minutes and seconds count:*

A fault was detected and isolated. Suspected failure components are highlighted in red (solid shading) and beeping, namely the MOTOR, L1, GEARBOX, L2, and ACTUATOR.

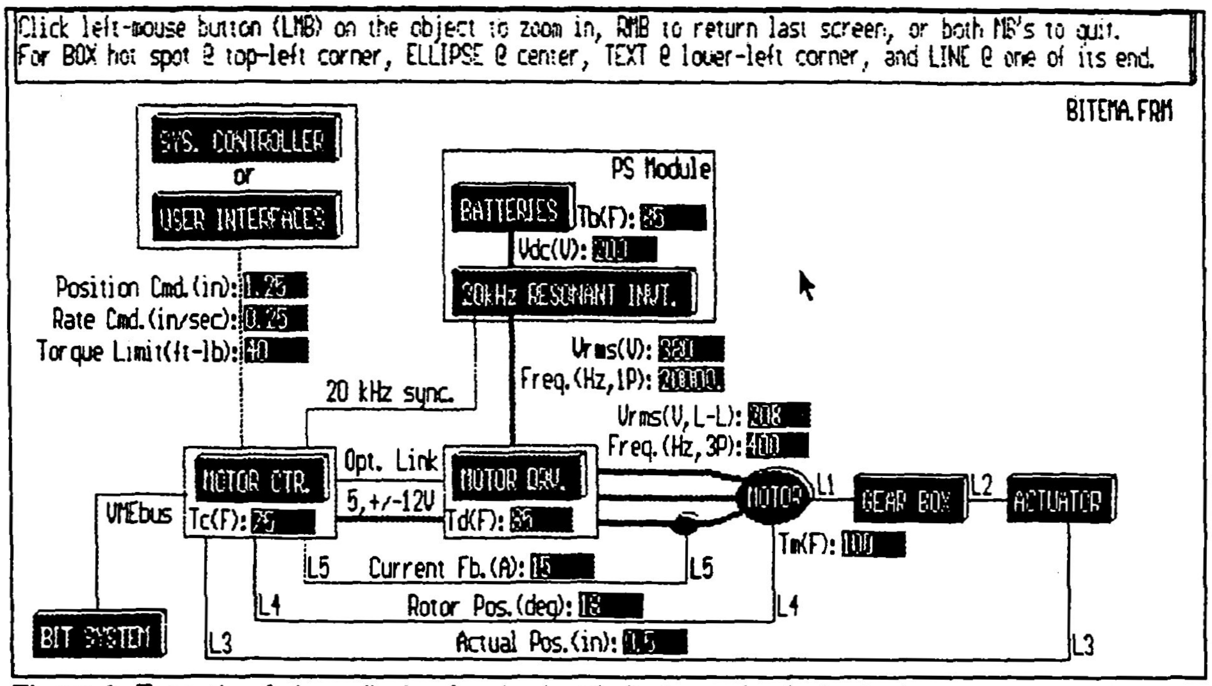

Click left-mouse button (LMB) on the object to zoom in, RMB to return last screen, or both MB's to quit.
For BOX hot spot @ top-left corner, ELLIPSE @ center, TEXT @ lower-left corner, and LINE @ one of its end.

Figure 8 Example of alarm display for simulated electromechanical actuator system.

Diagnostic text messages for further explanations of the situation can also be implemented as shown earlier in Figure 5. To illustrate the zooming (in/out) capability, a second-level detail of the MOTOR DRV. module was implemented (created and linked by the GG&M module) and is shown in Figure 9 (obtained by clicking the left-mouse button on the MOTOR DRV. image (Fig. 8)).

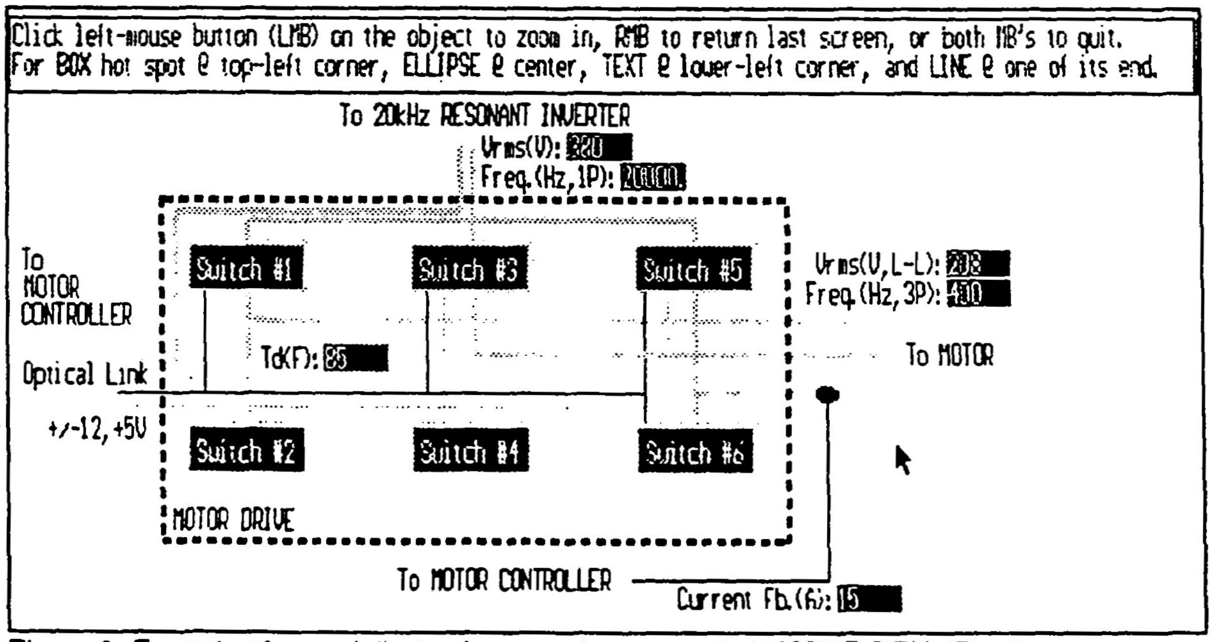

Click left-mouse button (LMB) on the object to zoom in, RMB to return last screen, or both MB's to quit.
For BOX hot spot @ top-left corner, ELLIPSE @ center, TEXT @ lower-left corner, and LINE @ one of its end.

Figure 9 Example of normal display for second-level detail of MOTOR DRV. (Fig. 8).

As you can see by comparing Figures 1 and 8, *colors* are essential for adding information to the images in such dynamic applications. Thus, real-time data visualization and highlighting are key contributions of these new graphics tools.

SUMMARY AND CONCLUSIONS

LCFS is an excellent tool for a wide range of real-time graphics applications, especially in data visualization for system health monitoring and management. The tool is user friendly, having a graphical interface, mouse-driven controls, and on-line instructions. A virtually unlimited (or limited only by the hard disk) number of frames can be easily created, logically wired, and then dynamically recalled for display in real time by using mouse commands. Its convenient built-in graphics tools, data simulation utility, and support of run-time software truly allow users to focus fully on their applications' contents. The system is easy to maintain and modify because of its software structure: modular and independent from the users' data bases. And finally, with the open (versus proprietary for most commercial graphics tools) format of the frame graphics file, users have the opportunity to customize their own application software. Thus, this new approach for real-time graphics applications suggests new opportunities for commercial markets.

REFERENCES

1. Truong, L., et al.: Autonomous Power Expert Fault Diagnostic for Space Station *Freedom* Electrical Power Testbed. Proceedings of Third Annual Workshop on Space Operations Automation and Robotics (SOAR 1989), NASA CP-309, 1989, pp. 181–186.

2. Walters, J., et al.: Autonomous Power Expert System. The 1990 Goddard Conference on Space Applications of Artificial Intelligence, J.L. Rash, ed., NASA CP-3068, 1990, pp. 147–156.

3. Truong, L.: PC Graphics Generation and Management Tool for Real-Time Applications. NASA TM-105749, 1992.

4. Truong, L.: LCFM—Living Color Frame Maker: PC Graphics Generation and Management Tool for Real-Time Applications. NASA/COSMIC, The University of Georgia, Athens, GA 30602-4272, (706) 542-3265, 1992.

5. Microsoft MS-DOS Version 5.0, User's Guide and Reference. Microsoft Press, Redmond, WA, 1991.

N96- 14996

Design Strategies and Functionality of the Visual Interface for Virtual Interaction Development (VIVID) Tool

Lac Nguyen I-Net Corporation

Patrick J. Kenney LinCom Corporation

Software Technology Branch
NASA / Johnson Space Center

ABSTRACT

Development of interactive virtual environments (VE) has typically consisted of three primary activities: model (object) development, model relationship tree development, and environment behavior definition and coding. The model and relationship tree development activities are accomplished with a variety of well-established graphic library (GL) based programs - most utilizing graphical user interfaces (GUI) with point-and-click interactions. Because of this GUI format, little programming expertise on the part of the developer is necessary to create the 3D graphical models or to establish interrelationships between the models. However, the third VE development activity, environment behavior definition and coding, has generally required the greatest amount of time and programmer expertise. Behaviors, characteristics, and interactions between objects and the user within a VE must be defined via command line C coding prior to rendering the environment scenes. In an effort to simplify this environment behavior definition phase for non-programmers, and to provide easy access to model and tree tools, a graphical interface and development tool has been created. The principal thrust of this research is to effect rapid development and prototyping of virtual environments. This presentation will discuss the "Visual Interface for Virtual Interaction Development" (VIVID) tool; an X-Windows based system employing drop-down menus for user selection of program access, models and trees, behavior editing, and code generation. Examples of these selections will be highlighted in this presentation, as will the currently available program interfaces. The functionality of this tool allows non-programming users access to all facets of VE development while providing experienced programmers with a collection of pre-coded behaviors. In conjunction with its existing interfaces and predefined suite of behaviors, future development plans for VIVID will be described. These include incorporation of dual user virtual environment enhancements, tool expansion, and additional behaviors.

INTRODUCTION

The creation of interactive virtual environments (VE) generally requires three developmental steps: model creation, model relationship definition, and environment/object behavior definition. Of these three, the behavior definition necessitates the greatest amount of time and expertise for coding on the part of a programmer. To alleviate much of this dependence on a dedicated programmer, and to create a rapid prototyping tool for VE demonstrations, the Visual Interface for Virtual Interaction Development (VIVID) was developed at the Software Technology Branch (STB) of the NASA/Johnson Space Center. VIVID not only provides a common interface between existing graphical library development programs being used, but also generates the code associated with behaviors and actions in a virtual environment. This paper discusses the design and interface considerations addressed while developing VIVID, describes examples of its functional execution, and outlines existing capabilities and directions for future enhancements.

BACKGROUND

Development of interactive virtual environments (VE) has typically consisted of three primary activities: model (object) development, model relationship tree development, and environment behavior definition and coding. The model and relationship tree development activities are accomplished with a variety of well-established graphic library (GL) based programs - most utilizing graphical user interfaces (GUI) with point-and-click interactions.

The GL development programs currently in use for VE development at the STB Lab are the Solid Surface Modeler[1] (SSM) and the Tree Display Manager1 (TDM). SSM is a graphics application for solid shaded and wireframe 3D geometric modeling and is used to develop the models (objects) that comprise a VE. TDM is a graphics visualization tool which uses a hierarchical representation (relationship tree) of the 3D models created with SSM to give structure to a VE. It provides an interface for generating and manipulating the relationship tree of models needed to portray a visual scene. For example, in TDM the user will specify which models to include in the environment, the position and intensity of light sources, and the visible perspectives of the environment (e.g. eyes and cameras). Both the SSM and TDM programs were developed on Silicon Graphics Incorporated (SGI) 4D series workstations under the IRIX (SGI UNIX) operating system.

Other, commercially available programs under evaluation at the STB Lab are Inventor[2] and Performer[2]. They serve essentially the same functions as SSM and TDM, respectively. Both programs were developed on SGI IRIS computers under the IRIX operating system, are portable and platform independent. In addition to having similar capabilities to SSM, Inventor includes a library of graphics objects and manipulations based on its object-oriented software library. With its database of scenes, Inventor not only renders, but provides other operations such as picking, printing, event handling, and input/output. Performer, comparable to the TDM program, provides for rapid development and functionality of generated environments. Also, users are given real-time control of the visual system without many of the constraints imposed by traditional image generation systems. The most attractive aspect of programs such as Performer is the availability of numerous and unique special effects that can be incorporated into an environment. Two of particular interest in developing virtual environments are point lights and weather effects. Although many other comparable model and tree development programs exist and are not mentioned here, they will be given similar consideration as they are obtained or made available for evaluation.

Because of the GUI format of these programs, little programming expertise on the part of the developer is necessary to create the 3D graphical models or to establish interrelationships between the models. However, the third VE development activity, environment behavior definition and coding, has generally required the greatest amount of time and programmer expertise. Behaviors, characteristics, and interactions between objects and the user within a VE must be defined via command line C coding prior to rendering the environment scenes. In an effort to simplify this environment behavior definition phase for non-programmers, and to provide for easy access (i.e. common interface) to the model and tree development tools, a graphical interface and development tool has been created. The Visual Interactive Virtual Interface Developer (VIVID) is an X-Windows based system utilizing the Motif Tool Kit and employing a drop-down menu system for user selections. The thrust of this research is to effect rapid development and prototyping of virtual environments.

VIVID FUNCTIONALITY

At this stage in its development VIVID has three primary functions; program access, behavior editing, and code generation. These functions are accessed via point-and-click activation of drop-down menus. As displayed in Figure 1, the VIVID screen format is common with other graphical user interfaces (GUI) such as those found on Macintosh and PC compatible computers - with some exceptions. Referring to the figure, the FILE, EDIT, VIEW, and HELP menu selections meet standard GUI design specifications. However, the PROPERTIES, CODE, and PROGRAMS choices are specific to the functionality of VIVID mentioned above. A description of each of these menu selections is presented in the following paragraphs. The large window on the left of the depicted screen in Figure 1 contains a list of all objects associated with a specific environment. The environment, and in turn models, is selected as a TDM "tree" via a FILE menu item selection. The large window on the right of the screen lists those objects from the list on the left which have already had properties assigned to them. Adding and deleting models from this is accomplished with the EDIT menu, and properties are associated with models via the PROPERTIES menu.

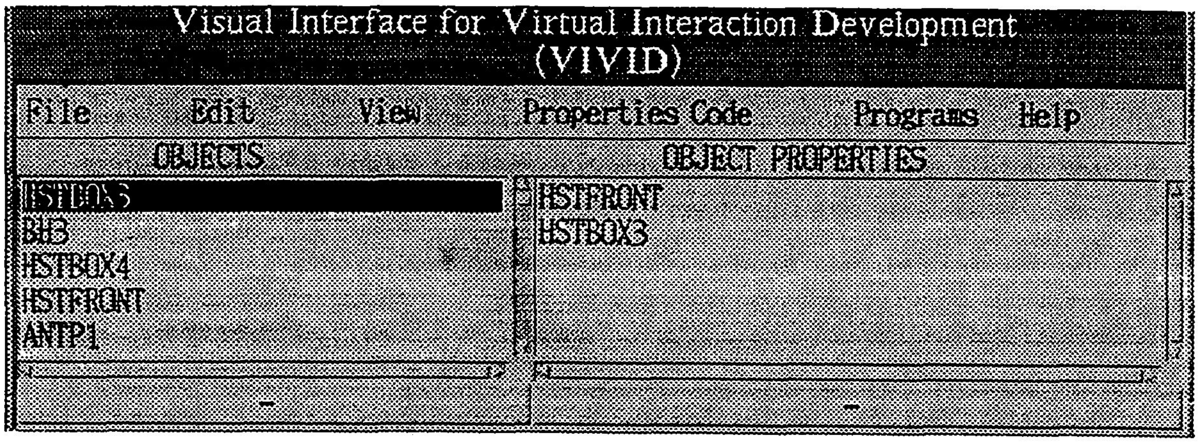


Figure 1 VIVID user interface and top-level menu items.

Initially, the FILE menu item is selected to display file operations commands. The available selections are shown in Figure 2. Currently, the NEW menu item is not functional. At a later date this will be used to create a new executable virtual environment file from a program "template". This template will include the C code necessary to direct the computer software to: load the appropriate model and tree software; render, sinc, and display an environment; and, interface with the data input/output devices. Other function code such as that for interfacing with the sound generating computer will also be included in the template. The LOAD item is presently configured to list for selection, only TDM trees in the configured path. As more development programs are evaluated and used (e.g. Performer), the LOAD item will be expanded with a secondary menu listing. Functionality of the other FILE menu selections; CLOSE, SAVE, and QUIT, is comparable to those of other GUIs.

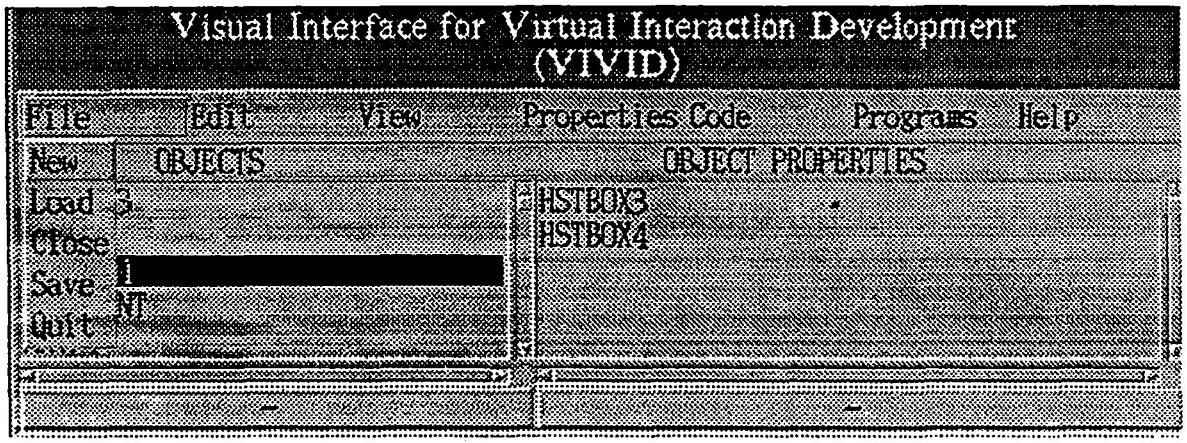


Figure 2 FILE menu item selections.

As shown in the EDIT menu screen, Figure 3, all of the available selections are standard with the exception of TREE and MODEL. The CUT, COPY, PASTE, ADD, and DELETE functions are used to manipulate the items listed in the "Objects" and "Object Properties" windows. An item is first selected (highlighted) with a single mouse click, and the desired operation is initiated. As mentioned, the TREE and MODEL operations are somewhat different from the others. The MODEL option will load an SSM model set and display it's content in the left-hand window. Any models with pre-existing properties will appear in the right-hand window. Selecting TREE will cause a TDM tree - associated models, to be loaded and its corresponding models to appear similarly in the windows. Once models are loaded and listed in either fashion, they can be edited with the desired EDIT menu selection.

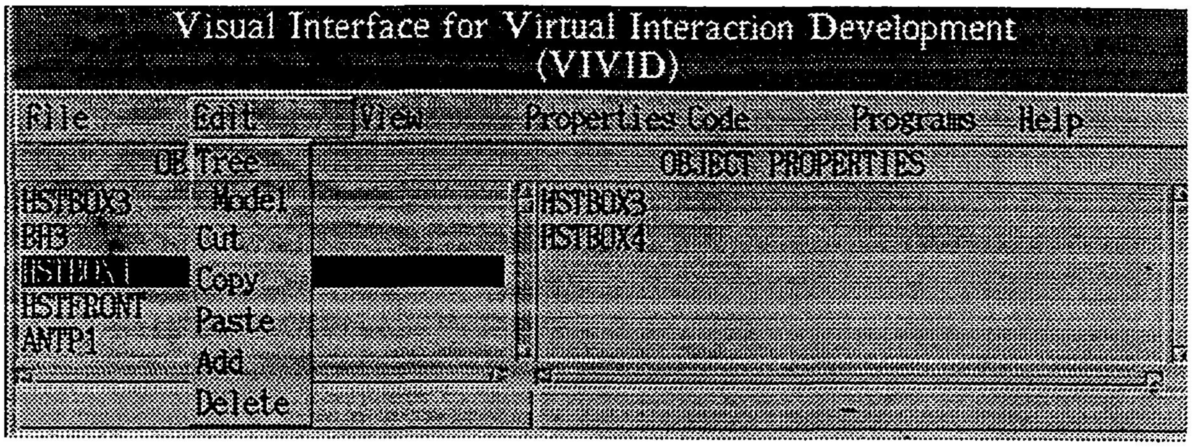


Figure 3 EDIT menu item selections.

To view either a model or a tree that has been loaded via the EDIT menu selections, the TREE or MODEL selections can be activated from the VIEW drop-down menu listing, Figure 4. This function is useful when trying to organize and decide which objects (models) to associate properties with.

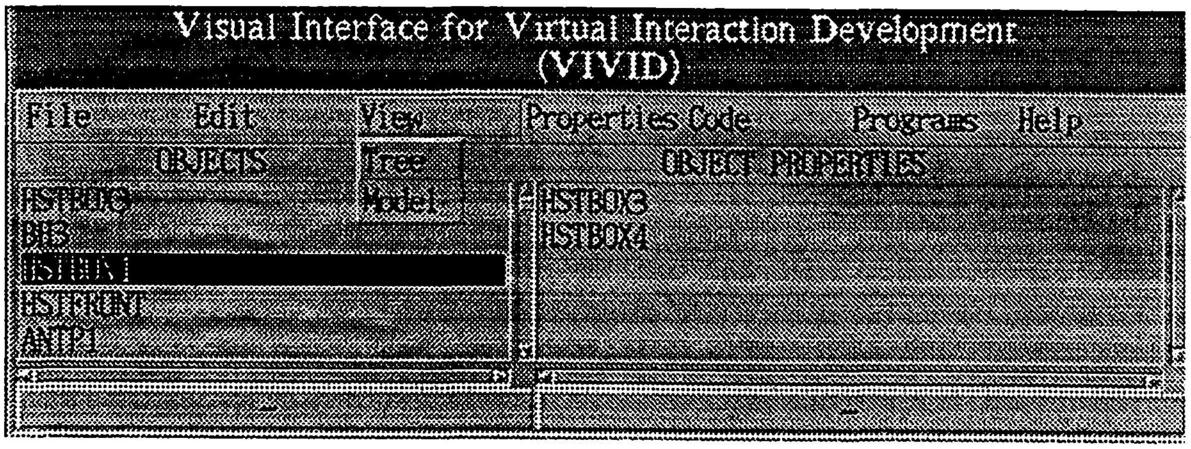


Figure 4 VIEW menu item selections.

Once objects (models) have been added to the Object Properties list the user can select the desired properties to associate with them. These are selected from the drop-down property list of the PROPERTIES menu item, Figure 5. Currently only "grabability", GRASP, and contact sounds, SOUND, are available options on this list. As more behaviors, properties, and "functionalities" are defined and coded for these environments they will be included.

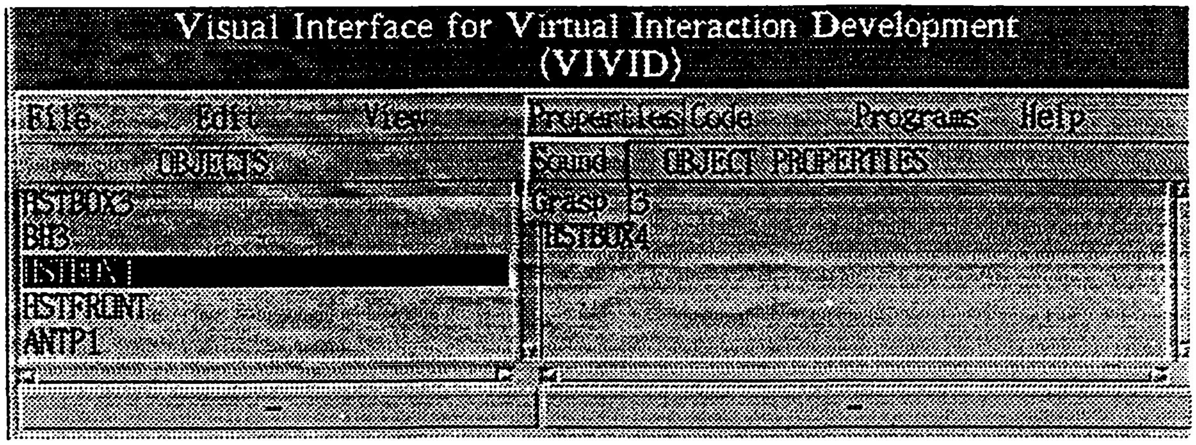


Figure 5 PROPERTIES menu item selections.

After models have been added to the Object Properties list with the EDIT menu, and properties linked to these models with the PROPERTIES menu, it is necessary to update the executable environment program. This is done with the CODE menu items in Figure 6. By simply selecting GENERATE, the C code associated with each of the selected models' properties will be created. After the code is generated it is inserted at the appropriate position in the existing program code. This is the step of the virtual environment development process that takes so much time, but is nearly eliminated with the use of VIVID. The new and existing code must then be compiled into executable form. This is done with the MAKE menu item selection. Finally, the new environment program is tested and verified to be operationally correct (i.e. correct maneuvering, object orientation, and object properties) by selecting the RUN menu item. This will run the program and display it on a screen window alongside the VIVID window. If there are problems or additional modifications needed, the necessary changes can be made by again calling VIVID.

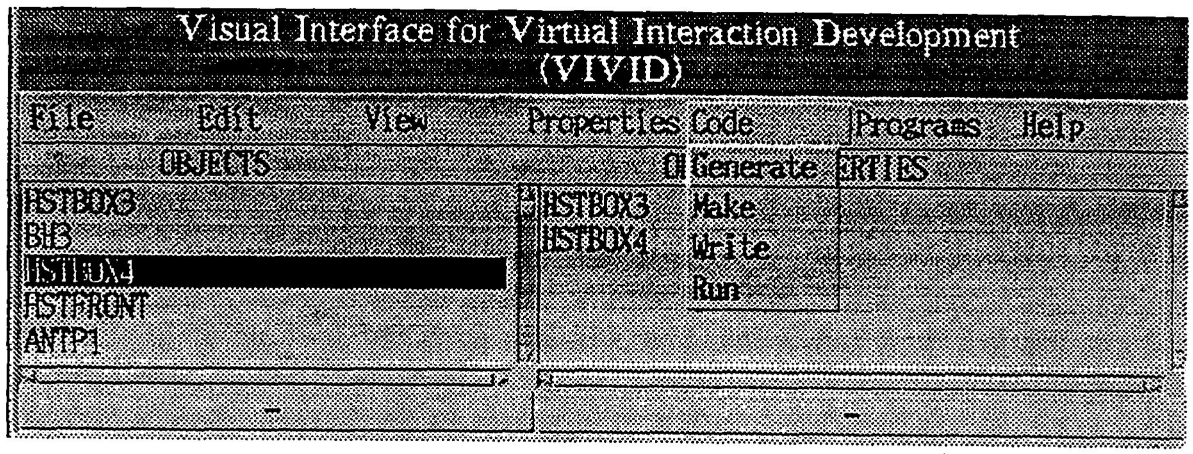


Figure 6 CODE menu item selections.

The PROGRAMS item is basically the interface to other virtual reality development tools. The drop-down options associated with this item can be seen in Figure 7. The SSM selection will run and overlay the Solid Surface Modeler program over the current screen, while selecting TDM will concurrently display it's interface window with the VIVID screen. At this point, either the SSM and TDM program is running and can be used in it's normal mode of operation. SSM can be used to edit or create models, while hierarchical tree relationships can be modified with TDM. VIVID is still in memory and running, but is in the background until SSM or TDM are closed normally. As other virtual environment tools are installed and used, similar single-click access will be provided as menu items.

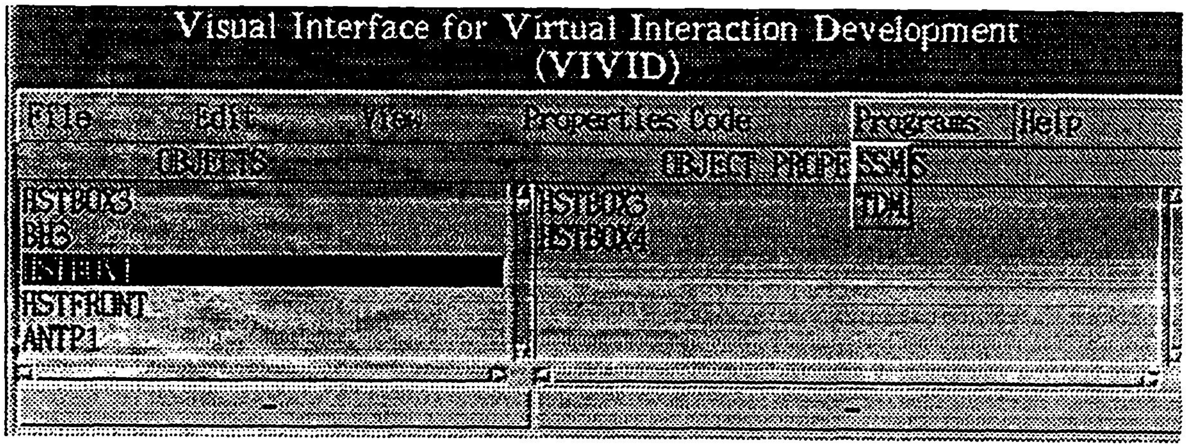


Figure 7 PROGRAMS menu item selections.

FUTURE DIRECTIONS

In conjunction with its existing interfaces and predefined suite of behaviors, future development plans for VIVID include incorporation of multi-user VE capability enhancements, tool expansion, additional behaviors, and interfacing with other graphics library and development programs.

The first user capability to be added is a graphical behavior editor. This consists of a tree diagram of the models in an environment - similar to that found in TDM - and a menu window of available behaviors. Selecting a model will display a pop-up listing of its existing behaviors. These can be added to by "dragging-and-dropping" a new behavior from a menu listing of those available. Another part of this feature will be an editor to add and delete objects within a tree. The second feature to be added is a user capability to create object behaviors. A user can modify their individual menu of available behaviors by adding it to the menu and writing the associated C code. This will be useful as new virtual environments are created and unique behaviors are required. A tool expansion feature will also be incorporated with VIVID. As new peripheral I/O devices are attached to the VE system, associated behaviors can be added to the VIVID menu listing. For example, when a tactile feedback glove is attached at least two new behaviors are possible within a VE; force applied to the user from an object and force applied to an object from the user. Finally, as new, more versatile graphic library and development programs are available VIVID will be modified to interface with them. This will involve data I/O between VIVID and these programs, as well as updating the properties and behaviors available within VIVID for implementation in virtual environments.

1. **Programs developed by the NASA/JSC Integrated Graphics and Operations Analysis Laboratory (IGOAL).**

2. **Programs developed by Silicon Graphics, Incorporated, Mountain View, CA.**