Research Report 1496

From Semantics of Procedures to Instructional Design: Project Review

Stanley J. Kostyla

Technologies for Skill Acquisition and Retention

Training Research Laboratory

U.S. Army
Research Institute for the Behavioral and Social Sciences

September 1988

U.S. ARMY RESEARCH INSTITUTE

FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction
of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON Technical Director	JON W. BLADES COL, IN Commanding

Technical review by

Richard P. Kern
Mark A. Sabol

NOTICES

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS --
2a. SECURITY CLASSIFICATION AUTHORITY --	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE --	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ARI Research Report 1496	5. MONITORING ORGANIZATION REPORT NUMBER(S) --
6a. NAME OF PERFORMING ORGANIZATION U.S. Army Research Institute; 6b. OFFICE SYMBOL (If applicable) PERI-IC	7a. NAME OF MONITORING ORGANIZATION --
6c. ADDRESS (City, State, and ZIP Code) 5001 Eisenhower Avenue Alexandria, VA 22333-5600	7b. ADDRESS (City, State, and ZIP Code) --
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Army Research Institute for the Behavioral & Social Sciences; 8b. OFFICE SYMBOL (If applicable) --	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER --
8c. ADDRESS (City, State, and ZIP Code) 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 62785; PROJECT NO. A790; TASK NO. 337; WORK UNIT ACCESSION NO. H1

11. TITLE (Include Security Classification)
From Semantics of Procedures to Instructional Design: Project Review

12. PERSONAL AUTHOR(S)
Kostyla, Stanley J.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 4/88 TO 6/88	1988, September	32

16. SUPPLEMENTARY NOTATION
This report summarizes research as described in ARI RN 88-11, Semantics of Procedures: A Cognitive Basis for Maintenance Training Competency.

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Troubleshooting; Intelligent tutoring system; Instructional design; Diagnostics; Qualitative modelling; Maintenance training

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In 1984, the U.S. Army Research Institute (ARI) initiated a 3-year project to study, design, and develop instructional environments to enhance procedural troubleshooting skills for maintaining complex machines. The goal of the project was to identify artificial intelligence technologies that could be used to increase the technical proficiency of maintenance personnel. Initially the effort focused on the role of conceptual and procedural knowledge in troubleshooting and the ways in which procedural skills can be learned as meaningful structures. Various types of computational tools were used to extract, analyze, and represent the structure of diagnostic procedures, field troubleshooting expertise, and the nature of mental models of complex machines and their role in causal reasoning. Simulation and qualitative modelling studies were conducted to determine the role of mental modelling in instruction and to investigate how simulation of machine behavior and repair strategies can provide maintenance personnel with a means

(Continued)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Stanley J. Kostyla	22b. TELEPHONE (Include Area Code) (202) 274-5540; 22c. OFFICE SYMBOL PERI-IC

DD Form 1473, JUN 86 — Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ARI Research Report 1496

19. ABSTRACT (Continued)

for understanding machine components, functions, and troubleshooting procedures. The investigation of instructio al strategies for teaching diagnostic skills led to the development of an interactive design and development system, Instructional Design Environment (IDE). IDE is a prototype interactive design and development system that assists instructional designers with the process of creating complex instruction. IDE is essentially a knowledge structuring system where the knowledge is course content, structure, and instructional method. IDE accepts knowledge describing course goals as input, and assists the designer in creating a course as output. The system implements a way of articulating the design and development process by fostering the creation of a structure that explains why curriculum design and delivery decisions were made. IDE can help create course design, structure course content, and create instructional sequences for standard as well as adaptive delivery.

Keywords: Army training, Maintenance training, Fault diagnosis, Systems maintenance, Maintenance technicians. (SDW)

Research Report 1496

From Semantics of Procedures to Instructional Design: Project Review

Stanley J. Kostyla

Technologies for Skill Acquisition and Retention
Zita M. Simutis, Chief

Training Research Laboratory
Jack H. Hiller, Director

U.S. Army Research Institute for the Behavioral and Social Sciences
5001 Eisenhower Avenue, Alexandria, Virginia 22333-5600

Office, Deputy Chief of Staff for Personnel
Department of the Army

September 1988

Army Project Number
2Q162785A790

Human Performance Effectiveness
and Simulation

FOREWORD

The Technologies for Skill Acquisition and Retention Technical Area (TSARTA) of the U.S. Army Research Institute for the Behavioral and Social Sciences (ARI) performs research and development to improve the quality of Army training.

With the increased complexity of technologically sophisticated equipment, the Army requires highly skilled personnel for systems maintenance, fault diagnosis, and repair. This report describes research efforts to develop effective instruction for teaching the complex skills and knowledge basic to maintenance troubleshooting tasks. The research examines factors underlying the learning of complex procedural skills. It also examines the instructional design and development process and describes the development of an interactive computer-based instructional design environment (IDE) to assist the instructional designer in managing and creating instruction. IDE is being used as a research tool in the development of intelligent tutoring systems. Its capabilities are being enhanced to incorporate built-in intelligent assistance for instruction management.

This work has been briefed to personnel at the U.S. Army Intelligence Center and School (USAICS). Research using IDE to develop intelligent tutoring systems for foreign language sustainment training is currently in progress with USAICS under MOA "Coordination of Efforts on Research and Development of Language Sustainment Training in Support of Military Intelligence," 27 July 1987.

[illegible]

EDGAR M. JOHNSON
Technical Director

(3) 6 blank

FROM SEMANTICS OF PROCEDURES TO INSTRUCTIONAL DESIGN: PROJECT REVIEW

EXECUTIVE SUMMARY

Requirement:

Technologically sophisticated equipment, often consisting of complex, interrelated mechanical, electronic, and optical subsystems, requires highly skilled personnel for systems maintenance, fault diagnosis, and repair. The effective diagnosis and repair of system malfunctions is dependent on skilled troubleshooting behavior that is acquired through extensive classroom and on-the-job training. As Army systems increase in complexity, the burden on the training system to produce the skilled maintenance personnel to meet the Army's needs will also increase.

The "Semantics of Procedures" project was initiated to investigate the process of creating effective instruction for teaching complex maintenance skills. This report describes the project from the initial analysis of troubleshooting behavior to the development of an interactive computer-based design tool, IDE (Instruction Design Environment), which assists the instructional designer in managing and creating instruction for teaching complex skills and information.

Procedure:

Troubleshooting behavior of maintenance technicians was analyzed to determine how procedural diagnostic skills can be learned as meaningful structures within the troubleshooting process. A variety of computational tools were used to extract, analyze, and represent the structure of diagnostic procedures, domain knowledge, and mental models of complex machines and their role in causal reasoning. Based on a model of troubleshooting behavior and qualitative modeling studies of complex machines, a variety of strategies for teaching troubleshooting skills were developed.

The process of creating effective instruction for teaching troubleshooting skills led to the development of a variety of tools to support domain content analysis and course design. The relationship between instructional design science and intelligent tutoring systems research was considered in the development of a conceptual framework for the instructional design and development process. This framework served as the basis for the development of the Instructional Design Environment (IDE), an interactive computer-based instructional design system developed to assist in the management and creation of instruction.

Findings:

The results of the studies support cognitive science research that suggests that, in designing instruction for complex skills such as troubleshooting, designers should consider human reasoning and problem solving skills and provide for an adequate level of understanding of machine functioning to support troubleshooting behavior. Researchers found that the development of mental models that support causal reasoning is facilitated by incorporating qualitative models of machine functioning in instruction.

The complexity involved in designing and developing instruction for teaching troubleshooting skills resulted in the development of IDE. IDE provides a representation for the domain knowledge and structure of a course, a mechanism for representing the principles and rationale underlying course design, and a structure for guiding the transformation of instruction to a selected delivery medium.

Utilization of Findings:

The research in how procedural troubleshooting skills are learned led to the examination of those aspects of instruction that proved most effective in training maintenance technicians in troubleshooting tasks. The complexity involved in the process of creating complex, multimedia-based instructional programs resulted in the development of IDE, which provides the instructional designer with an interactive on-line environment and tools to fully specify and rationalize the decisions made in developing content, structure, and instructional strategy. IDE is being used as a research tool in the development of intelligent tutoring systems. Its capabilities are being enhanced to incorporate built-in intelligent assistance to manage and create instruction.

FROM SEMANTICS OF PROCEDURES TO INSTRUCTIONAL DESIGN: PROJECT REVIEW

CONTENTS

	Page
INTRODUCTION	1
INVESTIGATING THE TROUBLESHOOTING PROCESS	2
The Role of Conceptual and Procedural Knowledge in Troubleshooting	2
Analysis of Fault Isolation Procedures	3
RESEARCH IN MODELLING REASONING AND EXPERTISE	3
Rationalizing Troubleshooting Procedures: DARN	3
Troubleshooting Models	11
Qualitative Simulation	13
INVESTIGATING INSTRUCTIONAL DESIGN	14
Issues in Instructional Design	14
Identifying the Instructional Design Space	14
THE INSTRUCTIONAL DESIGN ENVIRONMENT (IDE)	16
REFERENCES	23

LIST OF FIGURES

Figure 1.	Representation of part of a Fault Isolation Procedure (FIP) in DARN	4
2.	Representation of the FIP goal structure and standard Fault Isolation Procedure	6
3.	FIP goal structure hierarchy linking tests and conditions constraining each action	7
4.	Portion of a FAST graph showing a decomposition of the function: replenish developer	9
5.	A fault card representation in a FMEA analysis showing an hierarchical organization of possible failure modes and effects	10

Page

Figure 6. The ESF (Evidence-Symptoms-Faults) model showing the transition from the characterization of the initial evidence of a fault to the fix 11

7. The ESC (Evidence-Symptoms-Causes) elaboration (of the ESF Model) which captures the cognitive processes and knowledge of the technician in the troubleshooting process . . 12

8. The problem space of instructional design 15

9. Knowledge areas of IDE 16

10. An IDE display showing the organization and development of a basic course in xerography 17

11. An IDE display of a rationale trace of a fault prediction decision 18

12. A checker showing the rationale links between principles and instruction 18

13. The knowledge structure (KS) within IDE 19

14. Rules for defining instructional strategy and tactics 20

15. A notecard representing an instructional unit 21

FROM SEMANTICS OF PROCEDURES TO INSTRUCTIONAL DESIGN: PROJECT REVIEW

INTRODUCTION

This report reviews research sponsored by ARI to develop instructional design environments to assist the instructional designer in the process of creating, designing and delivering instruction for teaching complex skills and knowledge. The research covers a period of three years beginning with initial investigations of troubleshooting behavior, through the development, test and refinement of various knowledge representation and qualitative modelling approaches to the development of interactive instruction. The final phase of the project resulted in the design of an interactive, on-line instructional design environment for managing the creation of complex courses of instruction. The work considers recent advances in artificial intelligence, computational technology, cognitive science, and instructional systems design. The research assesses how a synthesis of this knowledge can be used to design and develop effective systems for teaching complex maintenance skills.

The initial investigations focused on an analysis of troubleshooting behavior of Xerox copier service representatives. Three lines of study were pursued in the investigation of troubleshooting skills. The first concerned the question of how procedural skills can be learned as meaningful structures by analyzing various types of diagnostic procedures as tools, as well as analyzing troubleshooting protocols and investigating the role of domain models in troubleshooting. This work focused on the use of computational tools to extract, analyze, and represent the structure of existing diagnostic procedures, field expertise and the nature of mental models of complex machines and their role in causal reasoning. The second line of study resulted in a model of the troubleshooting process based on field observation of troubleshooting behavior of maintenance technicians. The third line of investigation involved a series of qualitative modelling studies conducted to determine the role of mental modelling in instruction.

The results of the studies of troubleshooting behavior and the strategies for teaching troubleshooting skills, led to the exploration of the use of qualitative simulations in the development of instructional programs. The complexity involved in developing instruction resulted in further investigations of how to approach domain content and instructional strategies in technical training. This effort led to the development of tools to support domain content analysis and course design based on cognitive science and instructional design research.

The final phase of the research project resulted in the development of an Instructional Design Environment (IDE), an on-line instructional design system built as a tool to aid in the management of course creation. IDE provides a representation for the knowledge and structure of a course, as well as a representation mechanism for the principles and rationale underlying course design. A key feature of the system, the IDE interpreter, uses the knowledge structures created during course design as a knowledge source to guide the automatic creation of instruction. The interpreter synthesizes a course plan based on the description of the instructional method and the domain knowledge.

It then creates a plan of instructional units to satisfy previously defined instructional goals.

INVESTIGATING THE TROUBLESHOOTING PROCESS

The Role of Conceptual and Procedural Knowledge in Troubleshooting

Increased complexity inherent in the design of technologically sophisticated equipment requires highly skilled technical personnel for systems maintenance, fault diagnosis, and repair. The effective diagnosis and repair of system malfunctions is dependent on skilled troubleshooting behavior, which is most often the result of extensive classroom and on-the-job training. A variety of strategies have been used in the training of troubleshooting skills. Some approaches to training have emphasized the rote learning of procedures, often supplemented with detailed procedural guides or job aids. Other approaches have attempted to reduce reliance on procedural training by providing instruction in basic engineering theory underlying system design. Neither approach has been found to be completely satisfactory for effective teaching of troubleshooting skills. The problem with rote learning strategies, which rely on fully specified and directive procedures, is that they are inflexible and may not cover all potential machine malfunctions. The approach relying on instruction in basic engineering theory has been criticized because basic theory is not often directly relevant to the skills and knowledge required for troubleshooting. Research has suggested that for the design of effective instruction for troubleshooting, training should consider human reasoning and problem solving skills and also provide for an adequate level of understanding of machine functioning to support troubleshooting behavior.

The initial stage of the investigation for developing strategies for effective teaching of troubleshooting skills examined classroom materials, training of service personnel, and observations of diagnostic and repair activities of technicians in the field. The purpose of this aspect of the research was to explore ways in which the procedural skills identified for troubleshooting complex machines could be learned as meaningful structures within the procedural context of the troubleshooting process and mental models of machine functioning, rather than as a set of rote sequences of actions. The effort was aimed at developing and exploring forms of rationalized procedures and documentation for teaching troubleshooting skills. The work focused on using a variety of computational tools to extract, analyze, and represent the structure of existing diagnostic procedures and field expertise, and to study the role of mental models in causal reasoning. A variety of methodologies were used to investigate how to apply computational tools to the instruction of troubleshooting skills. These methodologies included qualitative modelling and knowledge representation approaches, psychological modelling, anthropological field studies, instructional design, and interactive system design.

In assessing the strengths and weaknesses of various procedural troubleshooting tools, the need for an underlying rationale and clear mental models in procedural training became evident. Diagnostic activity requires some understanding of machine structure and functioning in order for the technician to interpret the range of available information from service documentation to evidence of machine faults. As a basis for developing intelligent instructional techniques, a range of forms of rationalized documentation, models, and simulations of machine operations and diagnostic and repair strategies were investigated (Moran, Jordan, Newman, Orr, Russel, Rypa, and Shrager, 1985).

Analysis of Fault Isolation Procedures

It was determined that a large part of the skill required for following appropriate procedures in troubleshooting malfunctions depends upon knowing how to use technical documentation effectively. Part of the documentation provided to Xerox service technicians are the Fault Isolation Procedures (FIPs); block schematic diagrams of the subsystems and interconnections; and the service data consisting of a parts listing and supplementary information on machine repairs and adjustments. The FIPs represent detailed and highly directive troubleshooting procedures organized in a decision-tree structure. They are designed to reduce the requirement for a conceptual understanding of machine functioning. However, in practice it was found that the FIPs and supporting technical documentation are not used as intended. They are not followed blindly but tend to be used to support reasoning about the machine. The results of observation and protocol analysis of service activities of personnel in the field suggested that, while directed procedures may be useful for inexperienced service personnel, they do not effectively support the causal reasoning and strategic thinking demonstrated by more experienced field technicians.

RESEARCH IN MODELLING REASONING AND EXPERTISE

Rationalizing Troubleshooting Procedures: DARN

The analysis of the structure and use of the FIPS and supporting technical documentation resulted in efforts to develop training materials and job performance aids to represent the gradual transition of skill level from detailed procedure following to expert troubleshooting. The first step in this process was to analyze the underlying structure of the FIPS using a variety of computational tools for representation, analysis, and structuring of domain knowledge. Two of the standard FIPS for the 1075 copier were embedded into the Diagnostic and Repair Network (DARN), which is a computer-based job aid for fault diagnosis and repair (Mittal, Bobrow, and de Kleer, 1987). DARN encodes diagnostic procedures in a directed graph format, and classifies the steps in the procedures as net nodes in the graph (Figure 1).

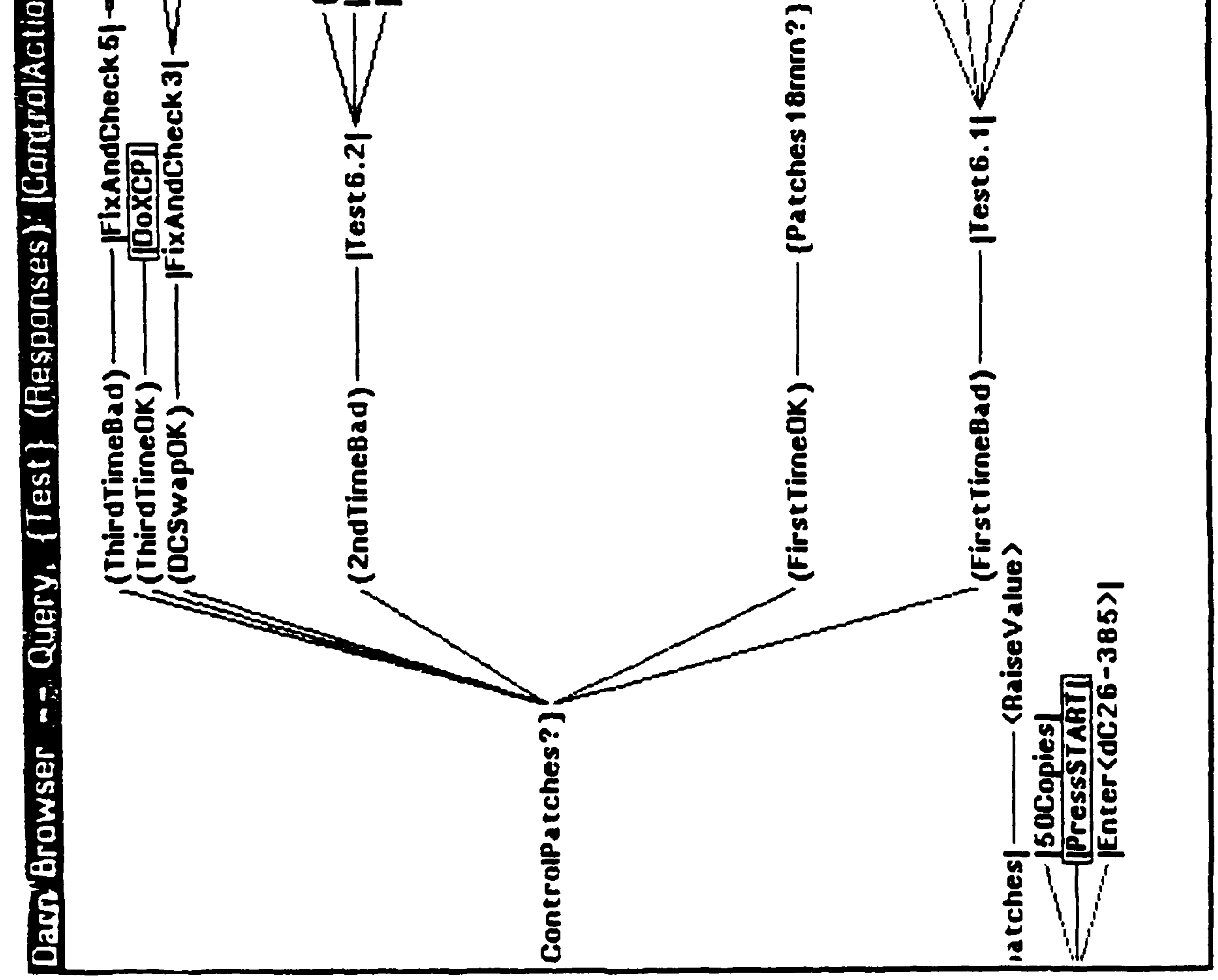


Figure 1. Representation of Part of a Fault Isolation Procedure (FIP) in Darn

Each node may represent one of four basic types of procedural steps: test, response, control action, and corrective action. During the encoding of a procedure, DARN creates a link between each node and an annotated explanation window. The system is interactive, providing menu commands and query and prompt facilities. DARN was used to examine more closely the structure of a FIP and to consider whether an interactive procedure-representation system would be useful as an on-line job performance aid. DARN was also evaluated as a basis for possible use as an on-line community diagnostic memory, due to its ability to record and display a history of the diagnostic actions of a number of technicians. The experience with DARN revealed a number of limitations in its use for the analysis and rationalization of FIPs. Its primary limitation in rationalizing procedures was that it did not provide a rich enough language for explanation. The annotations provided were limited to text, although much relevant information about machine structure and function may be best represented through diagrams and illustrations. Annotations were also linked to specific actions and did not address the significance of a set of actions or their relationships. In addition, the structural representation of the procedural actions did not convey a clear understanding of how a series of steps relates to machine structure or function, or to the overall troubleshooting process. As a result of these limitations in the rationalization of procedures within DARN, attempts were made to represent the FIPS in relation to a specific type of rationale based on a goal structure hierarchy. The intent was to provide a machine-specific functional context for each step in the fault isolation procedure and to determine the necessary and unnecessary orderings of procedural actions in the troubleshooting process.

For this analysis the Notecards system (Halasz, Moran, and Trigg, 1987) was used. Notecards consists of a linked network of electronic notecards which may contain text and graphic information (graphs, sketches, etc.). Fault isolation procedures were represented as a network of notecards. Each card represented one procedural action with a link to the rationale behind that action. Each procedural action or node in the goal structure was linked to a goal notecard which stated the goal of the action and the means to satisfy that goal (Figure 2).

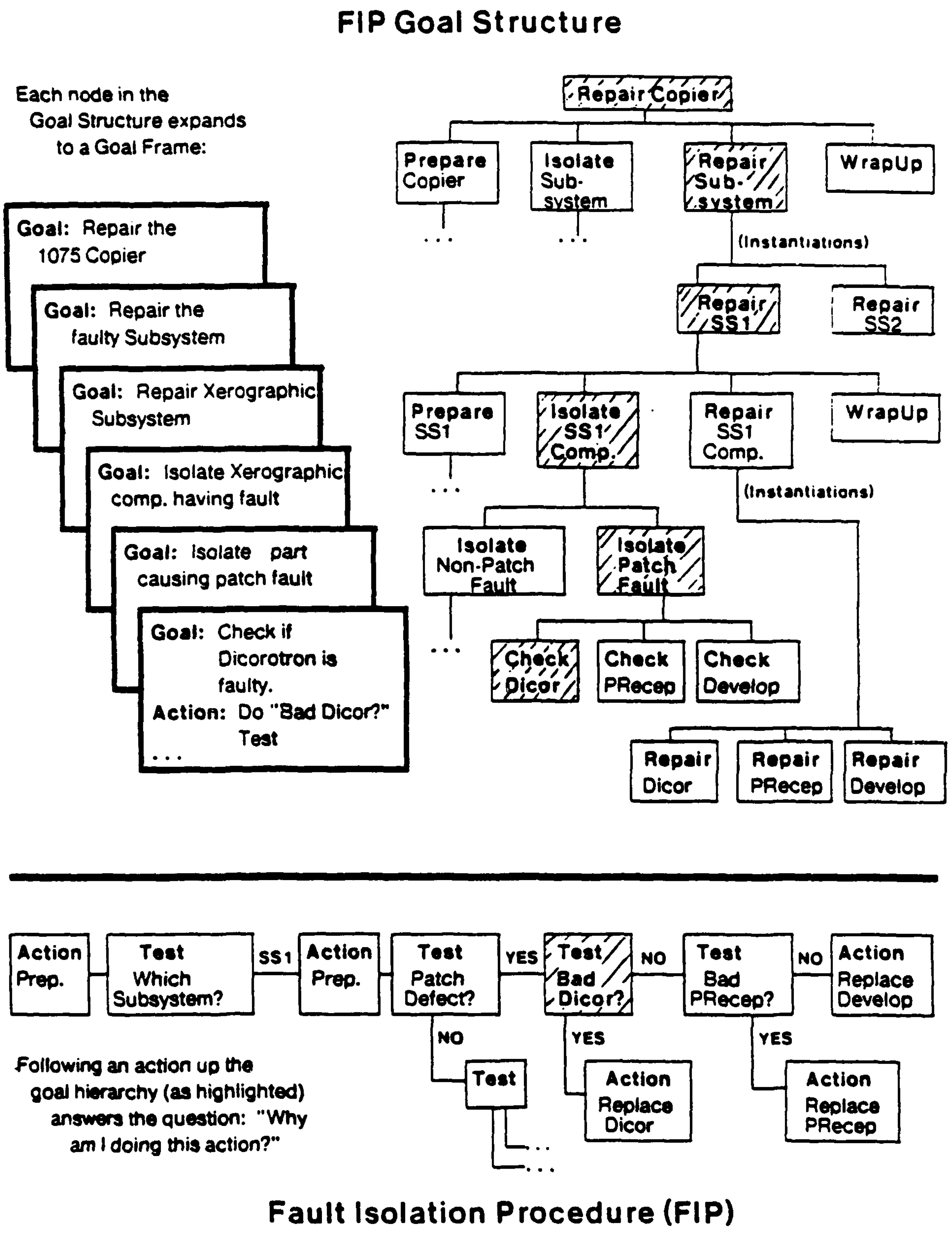
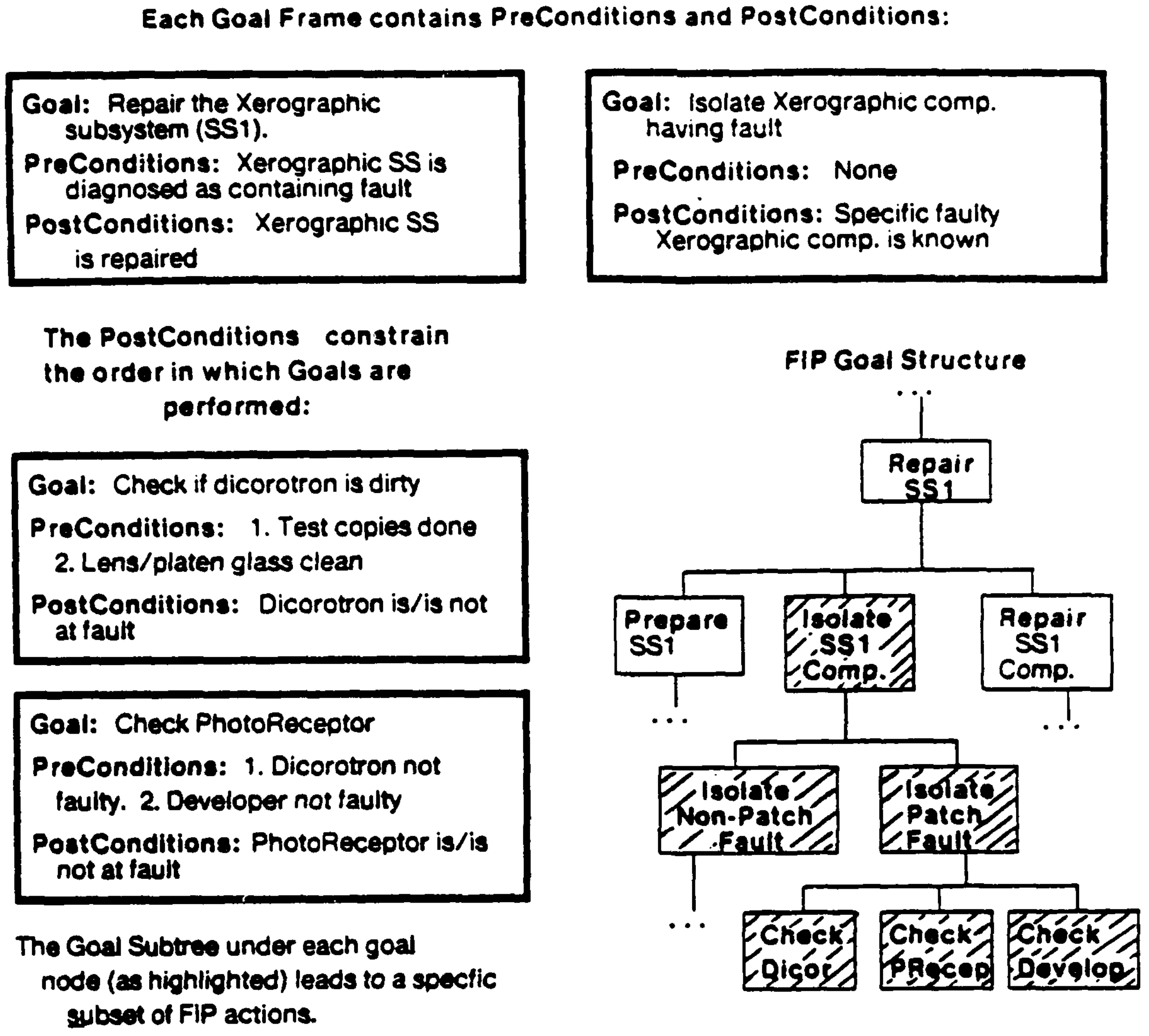


Figure 2. Representation of the FIP Goal Structure and Standard Fault Isolation Procedure.



Each Goal Frame contains PreConditions and PostConditions:

Goal: Repair the Xerographic subsystem (SS1).
PreConditions: Xerographic SS is diagnosed as containing fault
PostConditions: Xerographic SS is repaired

Goal: Isolate Xerographic comp. having fault
PreConditions: None
PostConditions: Specific faulty Xerographic comp. is known

The PostConditions constrain the order in which Goals are performed:

Goal: Check if dicorotron is dirty
PreConditions: 1. Test copies done 2. Lens/platen glass clean
PostConditions: Dicorotron is/is not at fault

Goal: Check PhotoReceptor
PreConditions: 1. Dicorotron not faulty. 2. Developer not faulty
PostConditions: PhotoReceptor is/is not at fault

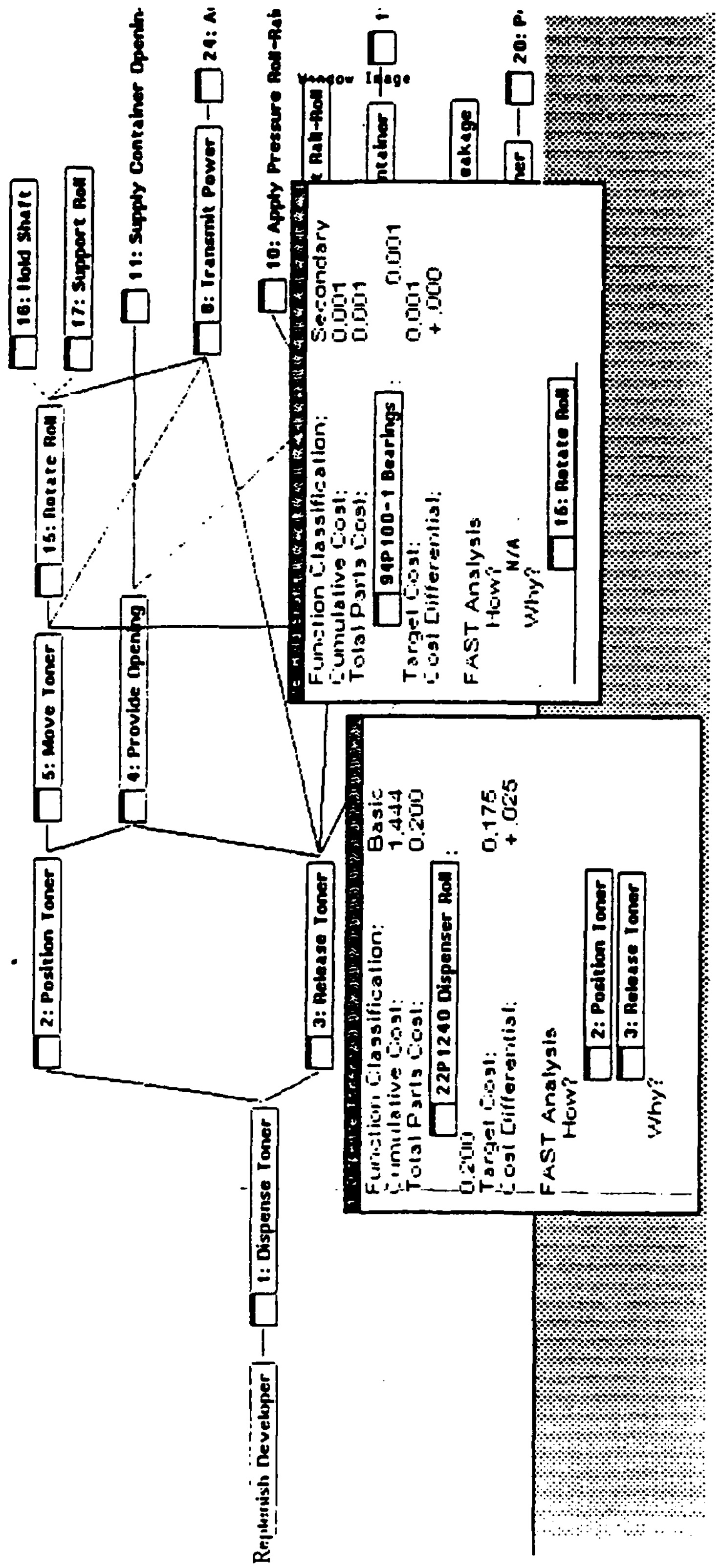
The Goal Subtree under each goal node (as highlighted) leads to a specfic subset of FIP actions.

Figure 3. FIP Goal Structure Hierarchy Linking Tests and Conditions Constraining Each Action.

In the goal structure hierarchy, the FIPS are represented as a series of linked actions and tests together with the preconditions and postconditions for carrying out those actions (Figure 3). The notecards representation allowed for an explicit rationale for individual procedural steps and also provided the user with an understanding of the relationship between current action, previous and subsequent actions, and the larger functional context.

Experience with this form of representation revealed some limitations. Not all goals could be represented by a standard taxonomy, and troubleshooting heuristics were not explicitly captured in the goal analysis structure. It was concluded that the FIPs, as currently structured and represented, were resistant to attempts at after the fact rationalization. Rather than pursue the rationalization of existing FIPS, efforts were directed towards developing rationalized procedures from an examination of the troubleshooting process based on field observations and causal machine models.

In addition to work on rationalizing existing procedures, one aspect of this research considered the task of creating the theoretical basi' -nd knowledge engineering tools for developing rationalized procedures from engineering design as a source of machine knowledge. Two existing engineering design techniques were examined: FAST (Functional Analysis System Technique) and FMEA (Failure Mode and Effects Analysis) (Moran, et al., 1985). FAST develops functional models of normal machine operation by explicitly representing the role of individual parts and how they interact with other components and subsystems (Figure 4). FMEA provides a hierarchical model of how the machine can fail (Figure 5). Using these engineering analysis techniques and the Notecards system, a FAST/FMEA system was created.

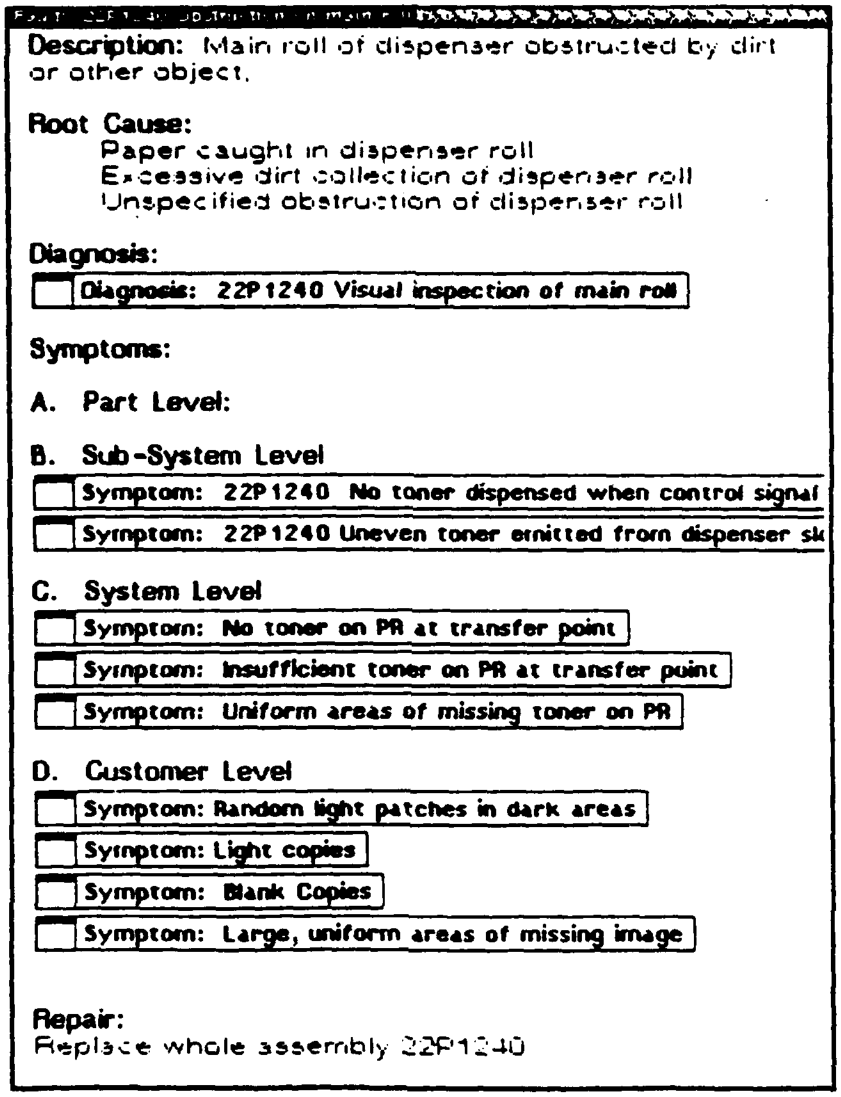


Function Classification: Secondary
Cumulative Cost: 0.001
Total Parts Cost: 0.001
94P100-1 Bearings : 0.001
Target Cost: 0.001
Cost Differential: + .000
FAST Analysis
How? N/A
Why? 15: Rotate Roll

Function Classification: Basic
Cumulative Cost: 1.444
Total Parts Cost: 0.200
22P1240 Dispenser Roll : 0.200
Target Cost: 0.175
Cost Differential: + .025
FAST Analysis
How? 2: Position Toner
3: Release Toner
Why?

Figure 4. Portion of a FAST Graph Showing a Decomposition of the Function: Replenish Developer.

The prototype system can be used to develop service documentation directly the machine's design documentation. Using the available design rationale, diagnostic steps can be explained in terms of underlying machine models which reflect a richer source of knowledge about machine processes. The inclusion of engineering design data on how machines can fail also suggests a priority order and sequence of diagnostic tests. In addition, design information is tied directly to the repair procedures, allowing easy modification of maintenance documentation when engineering changes occur.



Description: Main roll of dispenser obstructed by dirt or other object.

Root Cause:
Paper caught in dispenser roll
Excessive dirt collection of dispenser roll
Unspecified obstruction of dispenser roll

Diagnosis:
Diagnosis: 22P1240 Visual inspection of main roll

Symptoms:

A. Part Level:

B. Sub-System Level
Symptom: 22P1240 No toner dispensed when control signal
Symptom: 22P1240 Uneven toner emitted from dispenser sk

C. System Level
Symptom: No toner on PR at transfer point
Symptom: Insufficient toner on PR at transfer point
Symptom: Uniform areas of missing toner on PR

D. Customer Level
Symptom: Random light patches in dark areas
Symptom: Light copies
Symptom: Blank Copies
Symptom: Large, uniform areas of missing image

Repair:
Replace whole assembly 22P1240

Figure 5. A Fault Card Representation in a FMEA Analysis Showing an Hierarchical Organization of Possible Failure Modes and Effects.

Troubleshooting Models

One important aspect of the development of rationalized procedures was the need for a coherent notion of the task of diagnosing or troubleshooting malfunctions. It was important, therefore, in understanding the nature of the task of troubleshooting, to construct a realistic model of the troubleshooting process. The models developed were based on observed and recorded troubleshooting behavior as practiced in the field. This emphasis on observed behavior was chosen for several reasons. Principal among these was that instruction needs to teach skills and knowledge relevant to good troubleshooting such as that represented by the skill and knowledge possessed by expert troubleshooters. In addition, it was expected that the observation of troubleshooting behavior might lead to the characterization of individual differences in approaches to diagnostic problem solving. One of the goals of the observation was to identify the terms expert troubleshooters use to talk about their activities and to infer how they think about the troubleshooting process (Orr, 1987).

Two troubleshooting models were developed. The ESF (Evidence-Symptoms-Faults) model is a simplified, idealized data flow model which illustrates the basic types of information used in troubleshooting and the general cognitive activities that move the technician from one step to the next in the troubleshooting process (Figure 6).

$$\begin{tikzcd}
\{E\} \arrow[r, "\text{characterize}"'] & \{S\} \arrow[r, "\text{generate}"'] & \{F\} \arrow[r, "\text{isolate}"'] & F \arrow[r, "\text{fix}"'] & OK
\end{tikzcd}$$

Figure 6. The ESF (Evidence-Symptoms-Faults) Model Showing the Transition from the Characterization of the Initial Evidence of a Fault to the Fix.

The second model, ESC (Evidence-Symptoms-Causes), is a more elaborate version of the ESF model. It attempts to characterize in greater depth the types of knowledge and cognitive processes that technicians use in troubleshooting (Figure 7).

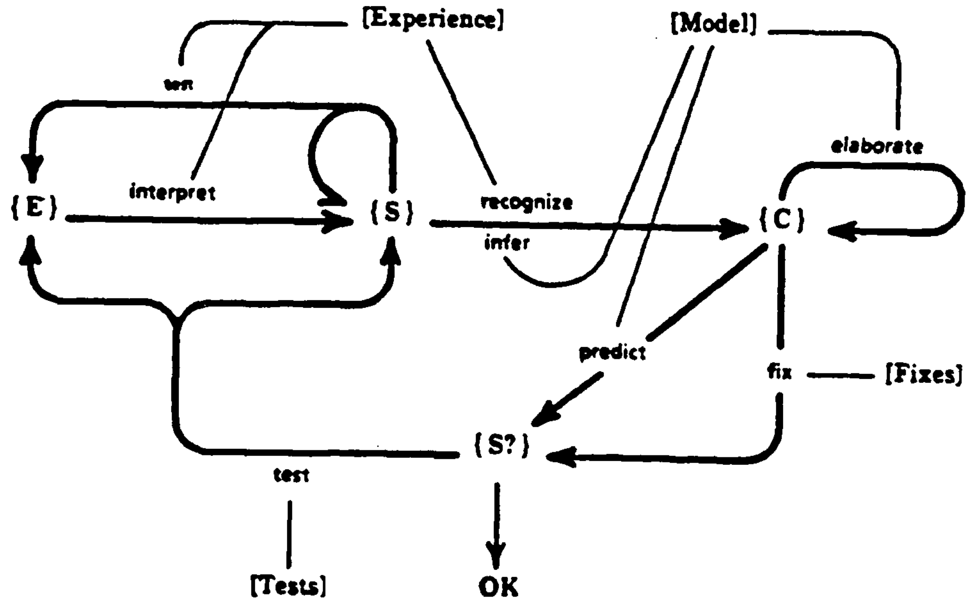


Figure 7. The ESC (Evidence-Symptoms-Causes) Elaboration (of the ESF model) which Captures the Cognitive Processes and Knowledge of the Technician in the Troubleshooting Process

The ESF data flow model shows that the troubleshooting process begins with a set of evidence about machine functioning, including normal operation and faults. The technician, in moving from evidence to symptoms, attempts to refine or characterize the existing evidence into a set of symptoms. Symptoms provide the basis for generating a set of possible faults and the procedures for isolating a particular fault. The ESF model distinguishes between evidence and symptoms. The identification of symptoms is seen as an intermediate step between the characterization of more general problems or evidence and the generation of a set of specific candidate faults. Because the standard language for talking about machine faults does not include a precise characterization of symptoms or describe symptoms in a way that emphasizes causality, research focused on developing a language that better relates symptoms to underlying faults. The language developed for describing machine faults provides cognitive support for reasoning from symptoms to causes and helps to focus attention on the causal relationship between component functions and interactions, component failures, and observable malfunctions.

The goal of the ESF model was to identify some of the basic elements of troubleshooting. The ESC model enhances the data flow notion of ESF by accounting for many more of the observable behaviors of technicians troubleshooting behavior in the field. It shows the role of a mental model of the machine, as well as the role of other knowledge, such as experience. It makes explicit the process of information gathering and shows the role of predicting and testing additional symptoms in order to verify diagnoses or generate new ones. ESC allows for the generation of alternative accounts of new evidence and recognizes the need for reinterpretation of prior evidence and how this process can serve to further refine the set of symptoms.

Qualitative Simulation

The observation of troubleshooting behavior of technicians supported cognitive science research in the area which indicates that mental models are a primary resource for understanding the behavior of complex systems and for causal reasoning involved in diagnosing machine faults. Efforts to develop qualitative simulations of machine functioning were aimed at providing models for use as instructional devices through which technician trainees could learn basic component functions and causal relationships.

The initial attempt at developing qualitative simulation models used an existing simulation, ARIA, which is a system for building qualitative models of machines (Shrager, Jordan, Moran, Russell, and Kiczales, 1986). ARIA provides a language in which the qualitative behavior of machine components, their states, and interactions can be specified. The development of the ARIA simulation raised the question of how qualitative models are to be used effectively in providing instruction for teaching novice technicians. The conclusions drawn from the ARIA studies suggested that the level of detail and understanding provided by the model to support effective troubleshooting need not be complete. The goal of troubleshooting instruction at the novice level should be to teach an adequate understanding of causality and propagation of effects between components. Trainees need to build a useful mental model of the system, which requires knowledge of the function and structure of system components and their causal relationships. The ARIA simulation model allows the trainee to examine and alter component parameters to explore how interacting components produce a variety of machine behaviors.

The effort to explore the range of what kinds of instructional tools can be built using qualitative models led to the development of ARIACore, an attempt to abstract the core of the ARIA simulation system and make it more easily usable by instructors with little or no computer programming experience. The ARIACore simulation allows an instructor to develop qualitative simulations by creating sets of objects and defining qualitative relationships that define the interactions between the objects.

Further research in qualitative modelling explored the use of multiple models to represent complex mechanical devices and to support efficient reasoning about these devices. Interactions among various machine models were investigated, and increasingly detailed versions of qualitative models were developed. A functional description, a component process description, and a qualitative physics description of three-dimensional shape and constrained motion were prepared as mental models of increasing levels of sophistication (Weld, 1985). Another aspect of the research was concerned with the proposed development of qualitative models to support automatic diagnostic reasoning (Farley, 1984), where the qualitative model is used to support the creation of causal explanations of abnormal machine behavior.

INVESTIGATING INSTRUCTIONAL DESIGN

Issues in Instructional Design

The initial efforts to design instruction based on ARIA attempted to explore a variety of instructional strategies based on qualitative simulations. A basic xerography course incorporating ARIA as a component for teaching conceptual understanding of the xerography system was developed for use at Xerox regional training centers (Xerox, 1985). The result of the experience in developing complex instruction prompted further study in methods to develop quality multi-media instructional materials for technically complex subject matter. It was found that imparting a global understanding of a complex system could be accomplished most effectively by presenting multiple conceptual structures affording several views of the same knowledge, for example, component-centered, process, and functional views. Although no particular order of presenting such conceptual structures in instruction proved to be more effective, it appears that a more coherent and complete picture of the subject matter is provided by presenting multiple, interacting, and complementary views. The appreciation and understanding of the complexity of the course design process led to the investigation of how to approach domain content and instructional strategy in technical training, and to the study and development of tools to support domain content analysis, course design, and course development based on instructional design and cognitive science research.

Identifying the Instructional Design Space

In investigating the design of effective instruction, developments in the fields of instructional design science and intelligent tutoring systems and the relationship between .hem were considered. In an analysis by Pirolli and Greeno (1987), instructional design is characterized as a problem solving task in which the instructional design process determines a selection among various design alternatives, within certain constraints of the instructional situation, to produce instruction which satisfies specified goals. They propose the notion of instructional design space which consists of a problem space of instructional design issues that provides a framework for organizing the various approaches found in instructional design and intelligent tutoring systems research. The objectives of the analysis were 1) to contribute to the formulation of general principles in the field of instructional design and intelligent tutoring systems that may facilitate further systematic research and 2) to understand the relationship between a variety of approaches to instructional design including the design of intelligent tutoring systems.

In this framework the task of instructional design is considered to be a problem-solving process. The problem space of the task characterizes several levels or subspaces of the alternative decisions and methods that the instructional designer has available. Within the problem space, Pirolli and Greeno define levels and aspects of instruction (Figure 8).

The three levels include "global issues," which consider general decisions related to the content and goals of instruction; "intermediate issues," which are concerned with lesson design, presentation, organization and representation of subject matter, and cognitive models; and "local issues," which consider details of media design and the human-computer interaction. These levels correspond approximately to issues in developing courses, designing lessons, and designing specific presentations and instructional activities.

Levels of Design Issues	Goals and Constraints	Technological Resources	Theoretical Resources
Global	Forms and Content of Learning	Learning Environments	Principles of Epistemology
Intermediate	Lessons and Activities	Content Topics, Tasks, Cognitive Models, Diagnostic Systems, Systems of Representation	Methods of Task Analysis, Structure of Subject-Matter Disciplines
Local	Presentations and Specific Tasks	Texts, Lectures, Conversation, Graphics Design, Human-Computer Interface	Principles of Component Display, Theory of Communication

Figure 8. The Problem Space of Instructional Design

The issues at each level involve goals and constraints, technological resources, and theoretical resources. The three classes of issues are interrelated, with technological resources providing the means for achieving goals and satisfying constraints. Theoretical resources provide reasons and explanation for the decisions that are made in selecting from alternative goals and methods. In this framework, a design process searches through the instructional design space selecting instructional means that achieve desired outcomes under certain specified constraints and stated instructional goals.

The integration of the process and products of instructional design was a major motivating idea in the development of the Instructional Design Environment (IDE)(Russell, Moran, and Jordan, 1987). The instructional analysis categories proposed in the instructional problem space framework of Pirolli and Greeno map onto the IDE design space.

THE INSTRUCTIONAL DESIGN ENVIRONMENT (IDE)

IDE is a prototype, interactive, computer-based tool designed to aid the instructional designer with the process of managing and creating complex instruction. It provides a representation for the domain knowledge and structure of a course, as well as a mechanism for representing the instructional principles and rationale underlying course design. IDE was developed as an interactive software environment to assist the instructional designer in managing and organizing course content, presentation, and delivery, and in developing an explicit rationale for course design consistent with instructional design principles and goals of instruction. The design environment guides the process of creating instruction by providing a mechanism to structure and record decisions related to course content, structure, and instructional approach. By creating and documenting an explicit rationale for course design, IDE allows for rapid changes in instructional approach or course content based on changes in goals or instructional rationale underlying course design and content.

IDE is built on Notecards a hypertext structuring system written in Xerox LISP. The organization of the system, representing the flow of knowledge from the representation of instructional objectives and constraints through the design of course materials, is presented in Figure 9.

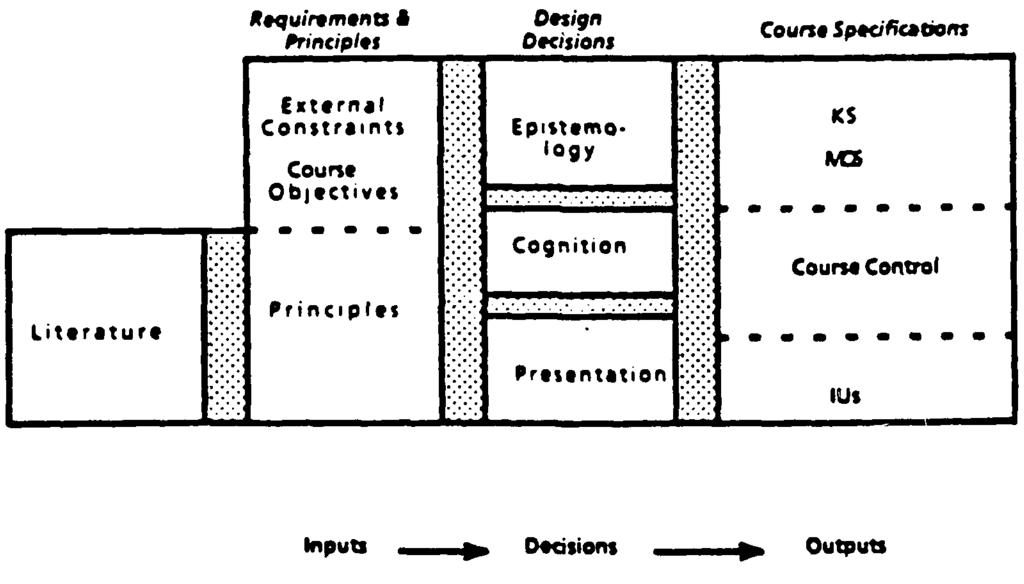
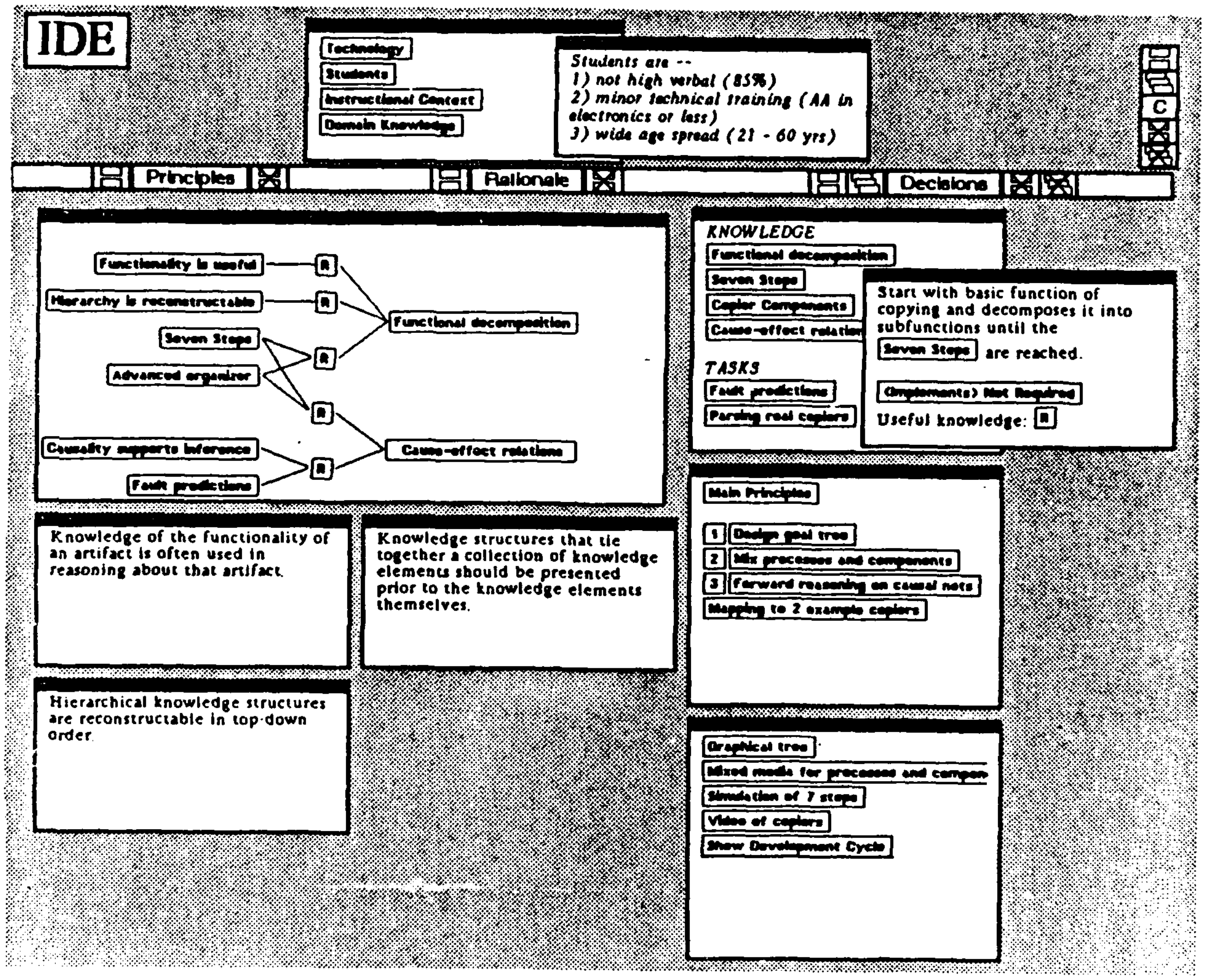


Figure 9. Knowledge Areas of IDE

The instructional designer creates a course design by structuring knowledge within the knowledge areas defined by IDE. The process begins with the definition of Requirements and Principles, which include information concerning course objectives, external constraints, and references to principles or theories of instruction. Design Decisions represent decisions about the domain knowledge to be taught, how the student will learn the skill or knowledge, and decisions which specify how the materials are to be presented. Course Specifications relate to the structuring of the domain knowledge representing course content, the model of the student, the knowledge required to structure and guide the instruction, and the Instructional Units which represent the course content derived from the domain knowledge structure (Figure 10).



Students are --
1) not high verbal (85%)
2) minor technical training (AA in electronics or less)
3) wide age spread (21 - 60 yrs)

Start with basic function of copying and decomposes it into subfunctions until the Seven Steps are reached.

(Implements) Not Required

Useful knowledge: R

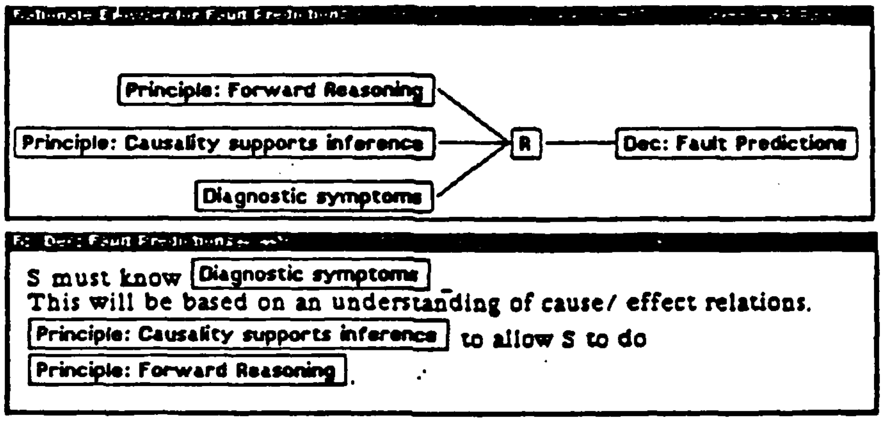
Knowledge of the functionality of an artifact is often used in reasoning about that artifact.

Knowledge structures that tie together a collection of knowledge elements should be presented prior to the knowledge elements themselves.

Hierarchical knowledge structures are reconstructable in top-down order.

Figure 10. An IDE Display Showing the Organization and Development of a Basic Course in Xerography

Knowledge or ideas are represented in IDE by a system of linked notecards which represent decisions about course content or structure rationalized by links to other cards which provide support for those decisions. The instructional designer has access to two analysis tools within IDE. A Rationale Tracer displays the rationale associated with each decision by tracing links from a decision back to its parent nodes (Figure 11).


Principle: Forward Reasoning
Principle: Causality supports inference
Diagnostic symptoms
R
Dec: Fault Predictions

S must know Diagnostic symptoms
This will be based on an understanding of cause/ effect relations.
Principle: Causality supports inference to allow S to do
Principle: Forward Reasoning.

Figure 11. An IDE Display of a Rationale Trace of a Fault Prediction Decision.

A Checker maps elements from one area to another based on specific relationships between the two areas (Figure 12). The tools provide a mechanism for reviewing design decisions and for checking redundancies, inconsistencies, and areas of incomplete information in the knowledge and relational structure.

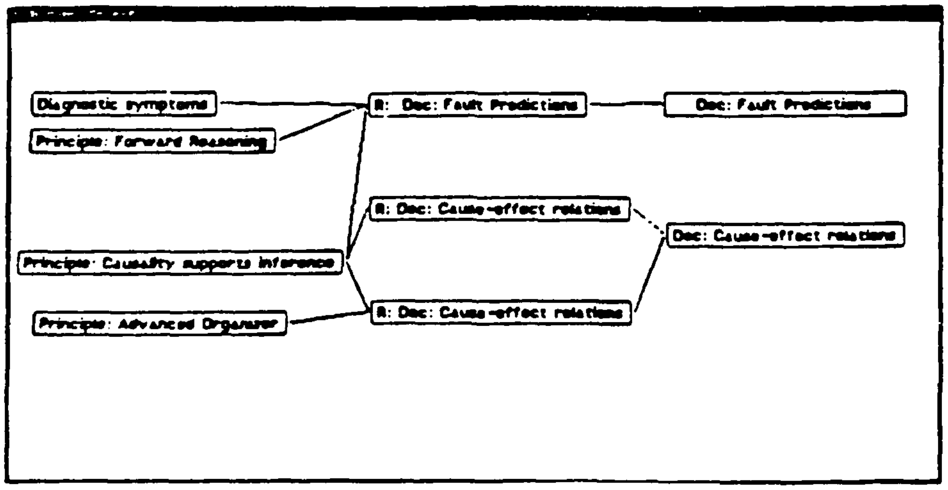

Diagnostic symptoms
Principle: Forward Reasoning
Principle: Causality supports inference
Principle: Advanced Organizer
R: Dec: Fault Predictions
R: Dec: Cause-effect relations
R: Dec: Cause-effect relations
Dec: Fault Predictions
Dec: Cause-effect relations

Figure 12. A Checker Showing the Rationale Links Between Principles and Instruction

The conceptual structure of the domain knowledge is represented in the Knowledge Structure (KS), which consists of a relational structure of the concepts of the domain (Figure 13). The content of the knowledge structure together with the instructional strategy fully defines the content of the course.

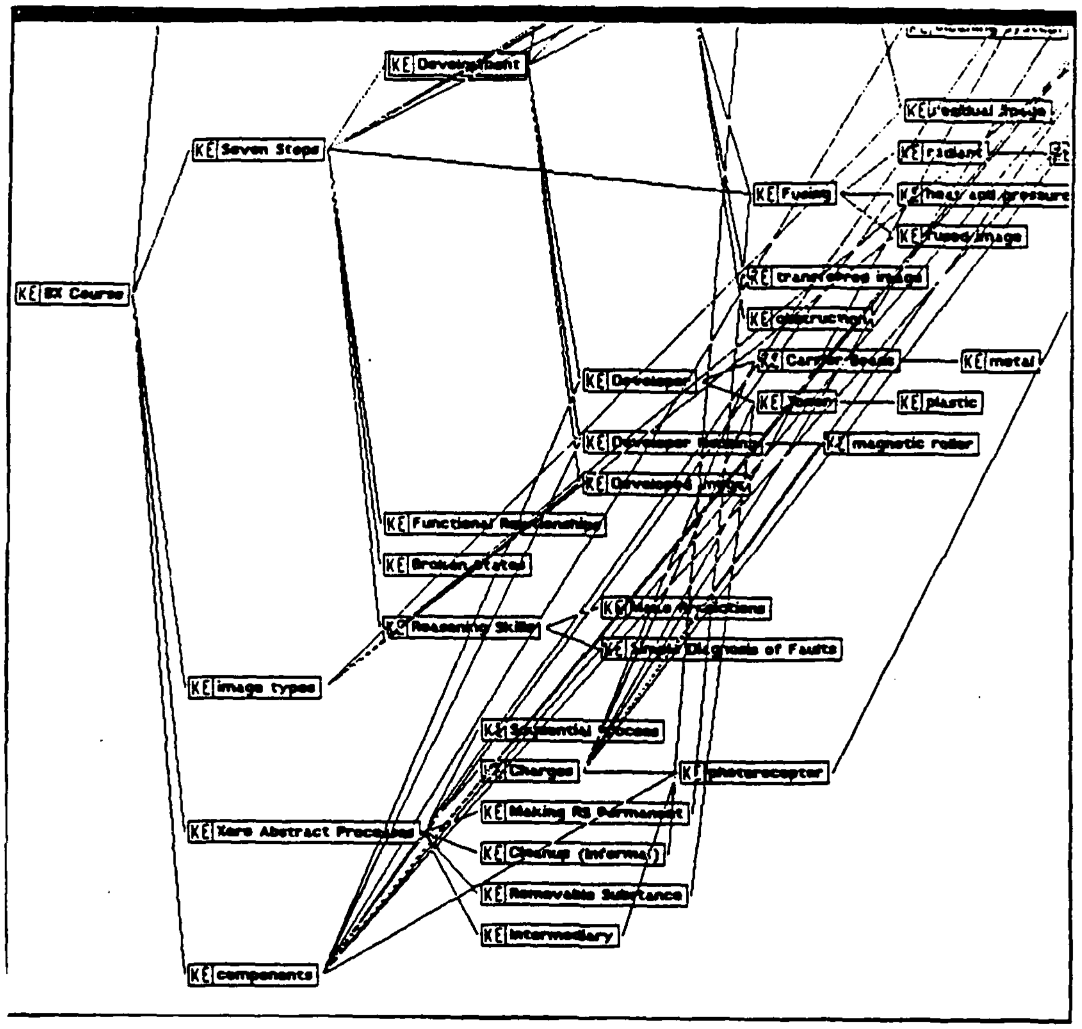
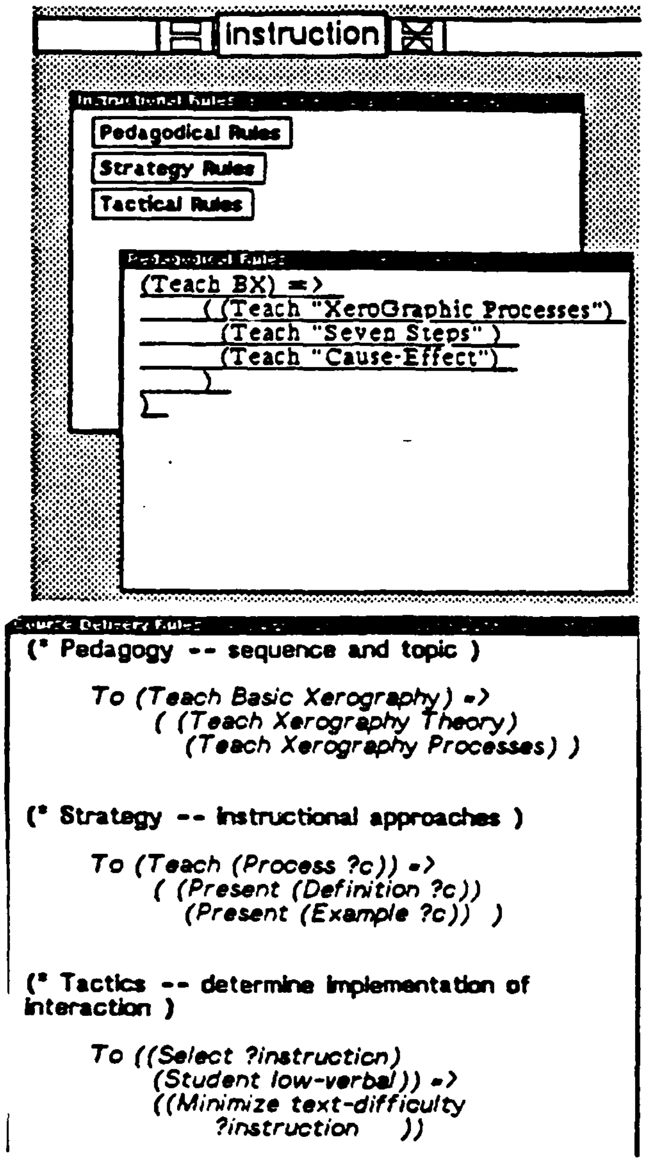


Figure 13. The Knowledge Structure (KS) within IDE. The nodes in the graph represent concepts or skills and the links the relationship between them.

The Knowledge Structure maps directly onto Instructional Units (IU). The structure is developed and rationalized by the instructional designer and the subject matter expert using a browser as a tool for defining and manipulating the Knowledge Structure. The browser is used to create nodes, links, and link types.

The instructional Strategy is developed through Rules and Constraints which create goals of instruction and describe how those goals are to be implemented. The knowledge base for designing Instructional Strategy consists of pedagogical rules and constraints, strategy rules, and tactical rules and constraints. These course control rules explicitly define the designer s approach to organizing and presenting the material (Figure 14).

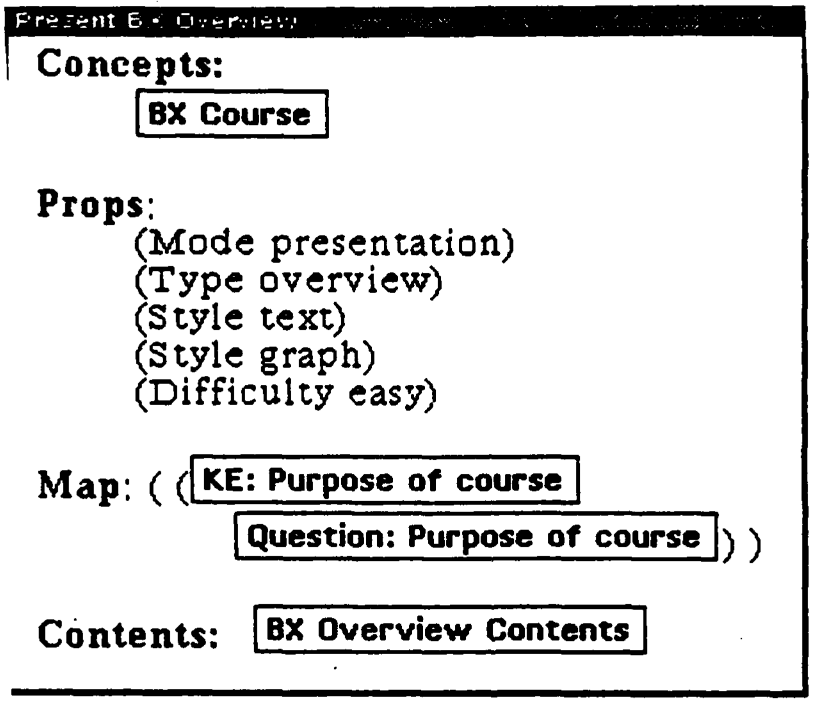


```
(Teach BX) =>
      ((Teach "XeroGraphic Processes")
        (Teach "Seven Steps" )
        (Teach "Cause-Effect")
      )
)
```

```
(* Pedagogy -- sequence and topic )

      To (Teach Basic Xerography) =>
           ( (Teach Xerography Theory)
              (Teach Xerography Processes) )

(* Strategy -- instructional approaches )

      To (Teach (Process ?c)) =>
           ( (Present (Definition ?c))
              (Present (Example ?c))  )

(* Tactics -- determine implementation of
interaction )

      To ((Select ?instruction)
           (Student low-verbal)) =>
           ((Minimize text-difficulty
                ?instruction     ))
```

Figure 14. Rules for Defining Instructional Strategy and Tactics. The course delivery rules will determine how the course will be taught.

The final output of the IDE design process is the creation of Instructional Units. An Instructional Unit defines the instructional material related to a specific concept. It is represented by a notecard that contains information on content, presentation mode, and questions and answers related to a particular concept (Figure 15).



Present BX Overview

Concepts:
BX Course

Props:
(Mode presentation)
(Type overview)
(Style text)
(Style graph)
(Difficulty easy)

Map: ((KE: Purpose of course
Question: Purpose of course))

Contents: BX Overview Contents

Figure 15. A Notecard Representing an Instructional Unit

At the completion of the design process, the course must be adapted to a particular delivery medium such as interactive videodisc, computer-based training, lecture, etc. The structure, representation and rules defining Instructional Units guide the transformation to whatever delivery medium is selected.

REFERENCES

Farley, A. M. (September 1984). Diagnostic mechanism modelling (Working Paper). Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.

Halasz, F. G., Moran, T. P., & Trigg, R. H. (1986). Notecards in a nutshell (Technical Publication). Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.

Mittal, S., Bobrow, D. G., & deKleer, J. (1987). "DARN: Towards a Community Memory for Diagnosis and Repair Tasks." In J. Hendler (Ed.), Expert Systems: The User Interface. Hillsdale, NJ: Ablex.

Moran, T. P., Jordan, D., Newman, S. E., Orr, J., Russell, D. M., Rypa, M., & Shrager, J. (April 1985). Early investigations of the role of conceptual and procedural knowledge in troubleshooting (Interim Progress Report, Contract MDA-903-83-C-0189). Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.

Orr, J. (January 1987). Talking about machines: Social aspects of expertise (Technical Publication, Contract MDA-903-83-C-0189). Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.

Pirolli, P., & Greeno, J. G. (August 1987). The problem of space of instructional design (Interim Progress Report, Contract MDA-903-83-C-0189). Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.

Russell, D. M., Moran, T. P., & Jordan, D. S. (August 1987). The instructional design environment (Technical Publication, Contract MDA-903-83-C-0189). Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.

Shrager, J., Jordan, D., Moran, T. P., Russell, D. M., & Kiczales, G. (April 1986). Pragmatic issues in qualitative modelling: Lessons learned from modelling xerography (Working Paper). Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.

Weld, D. S. (September 1985). Issues in the automatic description of complex mechanical devices (Working Paper). Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.

Xerox (June 1985). Basic xerography course. Palo Alto, CA: Intelligent Systems Laboratory, Xerox Palo Alto Research Center.