1

ARI Research Note 88-11

SEMANTICS OF PROCEDURES: A COGNITIVE BASIS FOR MAINTENANCE TRAINING COMPETENCY

Thomas Moran, Daniel M. Russell, Daniel Jordan,
Anne-Marie Jensen, Julian Orr, and Marikka Rypa

Xerox Special Information Systems

for

Contracting Officer's Representative
Joseph Psotka

Technologies for Skill Acquisition & Retention Technical Area
Zita M. Simutis, Chief

TRAINING RESEARCH LABORATORY
Jack H. Hiller, Director

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

April 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

	WM. DARRYL HENDERSON
EDGAR M. JOHNSON	COL, IN
Technical Director	Commanding

Research accomplished under contract
for the Department of the Army

Xerox Special Information Systems

Technical review by

J.C. Meiers
Stanley J. Kostyla

This report, as submitted by the contractor, has been cleared for release to Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or other reference services such as the National Technical Information Service (NTIS). The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.

THE ARI PROJECT

Xerox Palo Alto Research Center
Intelligent Systems Laboratory

July 1987

This work was partially sponsored by a research contract
MDA903-83-C-0189 from the Army Research Institute to Xerox

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA194403

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS
- -

2a. SECURITY CLASSIFICATION AUTHORITY
- -

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE
- -

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release;
distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
Xerox Special Information

5. MONITORING ORGANIZATION REPORT NUMBER(S)
ARI Research Note 88-11

6a. NAME OF PERFORMING ORGANIZATION
Xerox Special Information
Systems

6b. OFFICE SYMBOL *(If applicable)*
- -

7a. NAME OF MONITORING ORGANIZATION
U.S. Army Research Institute

6c. ADDRESS *(City, State, and ZIP Code)*
300 North Halstead Street
Pasadena, CA 91107

7b. ADDRESS *(City, State, and ZIP Code)*
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
U.S. Army Engineer School

8b. OFFICE SYMBOL *(If applicable)*
ATZA-A-TE-TT

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
MDA 903-83-C-0189

8c. ADDRESS *(City, State, and ZIP Code)*
Fort Belvoir,
Virginia 22060-5331

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
62717A	2Q1627 17A790	3.3.7	3.3.7.C.1

11. TITLE *(Include Security Classification)*
Semantics of Procedures: A Cognitive Basis for Maintenance Training Competency

12. PERSONAL AUTHOR(S)
T. Moran, D.M. Russell, D. Jordan, A. Jensen, and M. Rypa

13a. TYPE OF REPORT
Final Report

13b. TIME COVERED
FROM Nov 83 TO Dec 87

14. DATE OF REPORT *(Year, Month, Day)*
April 1988

15. PAGE COUNT
283

16. SUPPLEMENTARY NOTATION
Joseph Psotka, contracting officer's representative.

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Intelligent Tutoring System   Instructional Design
Qualitative Modelling   Diagnostics
Maintenance Training   Troubleshooting   (OVER)

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

In 1984, the Army Research Institute initiated a three year project to study, design, and develop instructional environments to enhance the learning of procedural troubleshooting skills for the maintenance of complex machines. The goal of the project was to identify how artificial intelligence technologies could be used to create better technical proficiency instruction for maintenance personnel.

Initially, the effort focused on the role of conceptual and procedural knowledge in troubleshooting, and on the ways that 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, expertise in troubleshooting in the field, and the nature of mental models of complex machines, and the role of such models 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 (OVER)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Joseph Psotka

22b. TELEPHONE *(Include Area Code)*
202/ 274-5540

22c. OFFICE SYMBOL
PERI-IC

Previous editions are obsolete.

ARI RESEARCH NOTE 88-11

18. Subject Terms (continued)

Artificial Intelligence (AI)
Instructional Design Environment (IDE)

19. Abstract (continued)

personnel with a means for understanding machine components, functions, and troubleshooting procedures.

The investigation of instructional strategies for teaching diagnostic skills led to the development of an interactive design and development system - the Instructional Design Environment (IDE).

IDE is a prototype interactive design and development system that assists instructional designers in the process of creating complex instruction. Put another way; it is essentially a knowledge structuring system, in which the knowledge is course content, structure, and instructional method. It accepts knowledge describing course goals as input, and then assists the designer in creating his output - courses.

The system thus implants a way of articulating the design and development process, by helping to create a structure which explains why curriculum design and delivery decisions were made. IDE can aid in creating course designs, structuring course content, and creating instructional sequences for standard, as well as adaptive, delivery.

I Project Overview

Understanding Instruction: From "Semantics of Procedures" to "Instructional Design"

Thomas P. Moran

Anne-Marie Jensen

Daniel Jordan

Julian Orr

Daniel M. Russell

Marikka Rypa

Xerox Palo Alto Research Center

Intelligent Systems Laboratory

May, 1987

This work was partially sponsored by research contract MDA903-83-C-0189 from the Army Research Institute.

Table of Contents

1.0 Introduction

This report documents our efforts to develop instructional environments incorporating qualitative knowledge and conceptual structures, as well as our work in exploring systems designed to help instructional designers manage the complexity of creating, designing and delivering instruction. In this work, we considered recent advances in artificial intelligence, computational technology, and cognitive science and how they can be used to build cost-effective systems for promoting the learning of procedural skills for maintaining complex machines.

We trace the project's evolution in terms of the original objectives and follow through our investigation of complex course design considerations. Our investigation initially focused on the determination of meaningful troubleshooting behavior and how to impart this in an instructional setting; this in turn led to our study of effective instructional design.

2.0 Investigating the Troubleshooting Process

In the following section we describe our investigations into troubleshooting behavior.

2.1 Identifying Troubleshooting Behavior

The first stages of our effort were devoted to the study of effective teaching in the area of troubleshooting. We focused on how procedural skills can be learned as meaningful structures by analyzing various types of procedures as tools as well as troubleshooting protocols and the role of domain models in troubleshooting. Much of our work at this stage focused on using computational tools with a range of techniques to extract, analyze, and represent the structure of a) existing diagnostic procedures, b) field expertise and c) the nature of mental models of a complex machine and their role in causal reasoning. We also experimented with computer-based systems that can aid in capturing and transmitting knowledge to service personnel [DARN]. Our aim was to identify how AI technologies might facilitate the building of machine-specific knowledge bases from which to create new learning materials and tools.

2.2 Adding Semantics to Procedures

In assessing the strengths and weaknesses of various procedural 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 function to interpret available information, from service documentation to evidence of machine faults. As a basis for developing intelligent instructional techniques, we investigated forms of rationalized documentation together with models and simulations of machine operations and repair strategies.

After building an on-line form of the directive documentation and analyzing the goal structure of these procedures, we concluded that rationalization of diagnostic and repair procedures cannot be easily added post-hoc. Rationalization in this manner is too localized, having lost the larger scope of principles underlying the procedure's design [DARN]. Furthermore, we discovered inherent limitations in the structure of standard repair procedure representations in supporting rationalization [NC-FIPs]. Even when a goal structure is created that allows a tech rep to view his position and path taken within that structure, it is difficult for to allow for the heuristic and unpredictable nature of troubleshooting, as well as a broad range of troubleshooting actions [DARN2].

We explored extending a repair procedure representation by using simplified documentation of the 3300 copier. Because diagnostic information is organized along system functional lines, this simplified representation facilitates heuristic behavior by providing an overview of machine parts, symptoms, and tests, which then can form the basis for reasoned choices by more experienced tech reps [Larson] (Also see [Minimal Manual]).

The limitations inherent in post-hoc rationalization of diagnostic procedures led us to build a prototype system from engineering design techniques that could be used to develop service documentation directly from the machine's design documentation. Using the available design rationale, diagnostic steps can be explained in terms of underlying machine models and consequently reflect a richer source of knowledge about machine processes. In addition, the inclusion of engineering analyses of data on how machines can fail suggests a priority order of diagnostic tests. Design information is tied directly to the repair procedures, allowing easy updates when engineering changes occur [FAST-FMEA].

2.3. Troubleshooting Models

Field observation of service personnel troubleshooting behavior provided the basis for a proposed data-flow model of this process. The ESF (Evidence-Symptoms-Faults) data-flow model pinpoints the distinction between evidence and symptoms and emphasizes the possibility of multiple faults. It is important to note that this model proposes no control structure, only the data flow in troubleshooting, allowing for individually varying styles. While working with this model of troubleshooting, we explored the feasibility of creating a descriptive language for copy quality defects to provide cognitive support for reasoning from symptoms to causes. The ESC (Evidence-Symptoms-Causes) model enhances the data-flow notion by accounting for more observable service personnel behavior. It shows, for example, the role of a mental model of the copier, making explicit the cyclic nature of processes such as information gathering and the interpretation of evidence [ESF/ESC].

3.0 Research in Modeling, Reasoning, and Expertise

The following sections are a discussion of the qualitative modeling studies conducted to determine their role of mental modeling in instruction.

3.1 ARIA

The ARIA simulation of the 1075 xerographics subsystem was built to investigate how on-line simulation of machine behavior can provide novice tech reps with a means for understanding machine components and function based on qualitative, causal mental models. ARIA was designed as an instructional device to study how to create instructional settings for trainees to incorporate causal models in their learning. Among the issues that emerged was the level of detail incorporated into model, and the level of understanding needed to support effective troubleshooting. We concluded that a less than total understanding is sufficient, and that the main goal of troubleshooting instruction should be to teach an adequate understanding of causality and the propogation of effects between various components[Causality]. ARIA represents a causal reasoning chain in machine function that provides a more precise characterization of copy quality defects relating symptoms to their underlying causes. One strength of this model lies in its differentation between qualitative inferences and spatial relationships derived from physical proximity. The effort to explore the range of what can be built using qualitative models led to the construction of ARIACore, an attempt to abstract the core of the ARIA simulation system and make it tailorable for non-programming instructors; ARIACore permits an instructor to create a qualitative simulation by creating a set of objects and defining qualitative relationships that define the interactions between these objects [ARIACore].

3.2 Further Qualitative Simulations

Further research into qualitative modeling 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 version of qualitative models were established. A functional description, a component process description, and a qualitative physics description of three-dimensional shape and constrained motion were proposed as mental models of increasing sophistication [Weld]. Another aspect of our study involved proposing qualitative models to support automatic diagnostic reasoning, where the qualitative model is used to support the creation of causal explanations of abnormal machine behavior [Farley].

4.0 Investigating Instructional Design

The following section describes our study of considerations in instrucional design.

4.1 Creating Courses

Our first efforts to build instructional settings based on our modeling work with ARIA were designed to explore instructional possibilities using qualitative simulation. These included devising student problems in the xerographic domain, such as "Find the Bug," and "Generate the Symptom."

As a result of our experimentation with these student problems, we determined the feasibility of implementation of various kinds of tasks within the kind of simulation we had built, and in turn we incorporated this knowledge into the construction of a course in xerography [Basic Xerography].

To gain experience in course creation and to further explore how to incorporate qualitative simulations into instruction, we developed a xerography course based on the Basic Xerography Module created internally for Xerox Regional Training Centers. This course incorporated ARIA as a component for teaching a conceptual understanding of the system and how system errors cause effects. The complexity of both the subject matter presentation and accompanying explanations and subsequent multi-media development caused us to investigate how to develop quality multi-media instructional material for subject matter that itself (independent of the medium) is complicated. We 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, e.g. component-centered, process, and functional views. Although no particular order of teaching such structures emerged as dominant, a more coherent picture of the subject matter is provided by presenting these multiple, interacting views. These conclusions are similar to the views of George Polya, who proposed the use of heuristic multiple approaches to problem solving as a formulation of an "appropriate view" of a problem, taking into account various motives and procedures for problem solving [Coherency].

As our understanding of the complexity of the course design process deepened, we began to consider how to approach domain content and instructional strategy in technical training. This led us to study and develop tools to support domain content analysis, course design, and course development based on instructional design and cognitive science research in this area.

4.2 Integrating Instructional Design Science with ITS Design

We first considered developments in the fields of instructional design and intelligent tutoring systems and the relationship between them. Although Instructional Design Science provides useful ways of characterizing a science of the design of intelligent tutoring systems, we argue that the two disciplines differ markedly in their representation of the task of instruction. Traditional instructional design science is a set of methods to achieve "optimal" results given specific task settings. However, in our study of building intelligent tutoring systems, we view instructional design as providing the designer with a set of operators in a space composed of student mental states. Designing instruction is, in essence, creating a plan to satisfy a goal in a state space of student mental models.

4.3 Identifying the Instructional Design Space

In [Instruction Design] Greeno and Pirolli propose a richer scheme of design process than is traditional, a scheme that rests on such distinctions as prescriptive vs. descriptive and course decision abstraction levels. They establish the notion of an instructional design space, i.e., a problem space of design issues that organizes the various approaches found in instructional design and intelligent tutoring system research. This framework of issues comprises three areas: (1) goals and environments of instruction (2) sequences of topics and systems of representation and (3) detailed sequences of instructional transactions and features of human-computer interaction. The first area involves clarifying metacognitive factors in learning, and conceptual structures and how they change; an expanding repertoire of instructional environments allows us to study these in greater depth. The second area considers representations of knowledge and distinctions among various forms of knowledge related to different tasks. One example is Riegeluth's organization of sets of concepts by different levels of abstraction and generality that elaborates the understanding of knowledge in ways related to theoretical cognitive analyses using semantic networks. The third area is concerned with decisions underlying specific types of presentation of ideas and examples. Each of these three areas maps directly onto a set of decisions to be made when building an ITS. As an example of the last area, decisions about specific types of presentations affect how the ITS displays will appear and interact.

This scheme contributed to our thinking in developing the Instructional Design Environment (IDE) For example, IDE categories correspond to Greeno and Pirolli's instructional analysis categories, mapping onto the IDE design space. The following table expresses this mapping:

The IDS / IDE Category Mapping:

IDS	IDE
Goals of instruction	Problem constraints/ course objectives
Topic sequence	Course Control
Systems of Representation	Epistemology
Instructional Transactions	Presentation

4.4 The Instructional Design Environment (IDE)

IDE (Instructional Design Environment) is an online instructional design system built as a tool to aid in the management of course creation. It provides a representation for the knowledge (content) and structure of a course, as well as a representation mechanism for principles and rationale underlying course design. Among the principles guiding the design of this environment were extensibility of the system in terms of interactive capabilities and the "open" nature of the environment, as opposed to the prescriptive nature of more traditional instructional design aids [IDE].

A hypertext structuring system allows users to create informal content representations within a formal relational structuring system. The capabilities of the system aid the designer in various ways. For example, the course designer can produce English text on cards and at the same time construct and represent inter-card relationships by building meaningful links between different types of cards for both design and creation of instruction. This structure maps onto the instruction development process we have observed in field studies. The management utilities of the system also aid in manipulating the structure of the underlying instructional design to respond to changing instructional assumptions, constraints, principles, etc. IDE also provides the user with a partially structured environment that directs attention to concerns appropriate to the process of course creation.

4.5 The IDE-Interpreter

The IDE Interpreter uses the knowledge structures produced during IDE use as a knowledge source to guide the automatic creation of instruction. Enough information is represented in the design to allow a planner-based Interpreter to create sequences of Instructional Units (i.e., atomic instruction sequences) that will satisfy course objectives. The Interpreter synthesizes a course plan based on the description of the instruction method (as specified in the Course Control rules, a part of the IDE design structure) and the domain knowledge (as represented in the IDE knowledge structure). It then creates a plan of Instructional Units to satisfy the stated instructional goals.

The IDE Interpreter may be used to deliver instruction interactively as well as aid during the instruction design phase. Delivering instruction to a student is equivalent to creating the instruction in real-time, displaying the result, and interpreting the student's interaction. If this can be done effectively, the instruction can be adaptive to the student's behavior by incorporating replanning during delivery in response to changes in a model of the student's knowledge. This requires modelling and tracking student behavior during instruction to provide useful feedback for the planner.

The Interpreter's responsive generation of instruction guided by an expressed instructional strategy allows the Interpreter to be used as a testbed of theories of instruction and learning. This distinctive separation of instruction knowledge and domain knowledge and the use of a planner to implement the instructional strategy make it possible to selectively change the style of instruction while holding the content of the instruction relatively constant over experimental trials.

As a system design, the IDE Interpreter is sufficiently general to act as a "meta-ITS;" many currently described ITSs may be translated into the IDE Interpreter's scheme. This encourages us in the belief that the Interpreter design is broad enough to encompass much of the work in ICAI; the abstractions forming the structure of the Interpreter (instruction planning, student modeling, instructional units and explicit instructional strategy) may serve as useful generalizations of essential ITS concepts [IDE-Interpreter].

5.0 Summary

The "Semantics of Procedures" project has covered a wide range of topics, spanning the spectrum from mental modelling through instruction design aids and interactive planner-based ITSs. Each step taken in our research path was based on our growing understanding that creating effective instruction lies at the core of increased technical performance by maintenance personnel. We have found that the greatest leverage is provided by improving the quality of the instructional materials and teaching systems. The models and tools described in this and previous reports were developed as a contribution to the original project goal.

This report takes a four-part form; following the section I Overview are sections detailing the project development over three phases spanning three years: II Early Investigations of the Role of Cenceptual and Procedural Knowledge in Troubleshooting; III Research in Modeling, Reasoning, and Expertise; and IV Instructional Design and Delivery.

References

[ARIA]

"Issues in the Pragmatics of Qualitative Modelling" Lessons Learned from a Xerographics Modelling Project"
J. Shrager, D.S. Jordan, T. Moran, G. Kiczales, D. M. Russell
To appear: CACM, ??, 1987
Also -- presented at ARI, Interim Progress Report
for Contract MDA-903-83-C-0189
December, 1984

[ARIACore]

"Artificial Intelligence Applications to Maintenance Training"
in G. Kearsley (ed.)- *Artificial Intelligence and Instruction*
Addison-Wesley, Reading, MA (1987)
Also -- presented at ARI, Interim Progress Report
for Contract MDA-903-83-C-0189
June, 1985

[BX Course]

"Basic Xerography Course"
Courseware created by ARI Project team
Intelligent Systems Laboratory
Xerox PARC, June, 1985

[Causality]

"Causal Reasoning for Diagnosis and Repair"
Presented at ARI, Interim Progress Report
for Contract MDA-903-83-C-0189
June, 1985

[Coherency]

"Building Coherent Mental Models by Teaching Multiple Views of a System"
Presented at ARI, Interim Progress Report
for Contract MDA-903-83-C-0189

[DARN]

"DARN: Towards a Community Memory for Diagnosis and Repair Tasks"
Mittal, S., Bobrow, D. G., deKleer, J.
to appear as a chapter in *Expert Systems: The User Interface*

J. Hendler (ed.), Ablex, 1987

[DARN2]
"Early Investigations of the Role of Conceptual and Procedural Knowledge in Troubleshooting"
ISL Technical Report
Xerox PARC, November, 1986

[FAST-FMEA]
"FAST-FMEA Prototype for Creating Fault Isolation Procedures from Engineering Design Data"
Presented at ARI, Interim Progress Report
for Contract MDA-903-83-C-0189
December, 1984

[ESF/ESC]
"Mental Models of Troubleshooting Behavior"
Presented at ARI, Interim Progress Report
for Contract MDA-903-83-C-0189
December, 1984

[Farley]
"Diagnostic Mechanism Modelling"
Working Paper, ISL Technical Report
Xerox PARC, September, 1984
(also in Volume 2, "Papers from the 'Semantics of Procedures' Project)

[Instruction Design]
"The Problem Space of Instructional Design"
Volume 3, "Papers from the 'Semantics of Procedures' Project", 1987
(Also presented at AI & Education Conference, Pittsburgh, PA, May, 1987)

[IDE]
"The Instructional Design Environment"
Volume 3, "Papers from the 'Semantics of Procedures' Project", 1987
(Also presented at First International Conference on Knowledge Engineering, Madrid, Spain, November, 1985)

[IDE-Interpreter]
"The IDE-Interpreter: Creating Instruction by Planning"
Volume 3, "Papers from the 'Semantics of Procedures' Project", 1987
(Also presented at AI & Education Conference, Pittsburgh, PA, May, 1987)

[Larson]

"Simplified Documentation for the 3300 Copier"
P. Larson
XIMTDC, Leesburg, VA, June, 1984

[Minimal Manual]

"Learning to Use a Word Processor: By doing, by thinking, and by Knowing"
in J.C. Thomas & M. Scheneider, (eds.)
Human Factors in Computing Systems
Norwood, NJ: Ablex, 1984

[NC-FIP]

"NC-FIP System for Rationalizing Procedures"
Presented at ARI, Interim Progress Report
for Contract MDA-903-83-C-0189
December, 1984

[Weld]

"Issues in the Automatic Description of Complex Mechanical Devices"
ISL Technical Report
Xerox PARC, September, 1985
(also in Volume 2, "Papers from the 'Semantics of Procedures' Project)
June, 1985

II Early Investigations of the Role of Conceptual and Procedural Knowledge in Troubleshooting

Thomas P. Moran
Daniel Jordan
Susan E. Newman
Julian Orr
Daniel M. Russell
Marikka Rypa
Jeff Shrager

Xerox Palo Alto Research Center
Intelligent Systems Laboratory

April 1985
Revised July 1987

This work was partially sponsored by a research contract MDA903-83-C-0189 from the Army Research Institute to Xerox.

Abstract

Our goal is to improve the troubleshooting abilities of service technicians by focussing on the training of troubleshooting skills. Our investigation incorporates three different lines of study: the structure and use of procedures as tools, how the troubleshooting process works, and the role of mental models in troubleshooting.

Field evidence indicates that directive procedural representations are not used by the technicians as intended. Analysis of current technical documentation reveals that these procedures do not support the user in either understanding or recalling the procedures while troubleshooting. These points suggest that it is important to give the technician the underlying rationale for a procedure, along with clear mental models of the machine and methods to use those models.

We began our analysis of current documentation by building and using an on-line form of the directive documentation (the DARN system), and by analyzing the goal structure of Fault Isolation Procedures (FIPS) used by Xerox technical repair personnel. It became clear that rationalization of such diagnostic and repair procedures cannot easily be added *post hoc.* We also learned that significant information is lost between engineering design and subsequent documentation and development of diagnosis and repair procedures. This led us to build a prototype system (FAST/FMEA) that could be used to develop service documentation directly from the machine's original design.

Based on field observations, we propose a data-flow model of the troubleshooting process derived from the experience of expert troubleshooters and a protocol analysis of troubleshooting behavior. This model shows the central role of a mental model of the machine in technical troubleshooting and illustrates the role of symptom characterization and causal stories in diagnosis.

An effective technique to teach coherent mentals model is to develop both understandable and manipulable representations of these models. In our approach, the student would interact with an animated computer simulation of an underlying

qualitative model. We discuss the current version of our simulation, which is based on qualitative physics. The role of qualitative simulation in instruction is outlined, with emphasis on encouraging the student to create and use causal representations in performing troubleshooting.

We conclude by analyzing the instructional requirements for teaching model-based troubleshooting.

Contents

1. INTRODUCTION

U.S. industry and the military face a growing problem in training technical personnel to operate, maintain, and repair complex machines. Increased sophistication in systems design means that a single piece of equipment often incorporates multiple technologies, making comprehension of the whole system difficult. In addition, accelerated technological development has created a need for more efficient methods for retraining, as technicians may be required to service many new machines over their service careers.

In the past, some technical training strategies and materials have emphasized rote learning of procedures; more recently, some corporate and military planners have suggested reducing the amount of formal training by providing technical personnel with detailed directive procedures to be used on the job. In contrast, various experimental attempts have been made to improve troubleshooting skills and to decrease reliance on procedures by teaching the engineering theory that underlies machine design.

We feel that none of these approaches is wholly effective. Research has shown that engineering theory is not easily assimilated into the intuitive understanding that technicians bring to training (Morris and Rouse, 1985), nor is it directly relevant to the skills and knowledge needed for troubleshooting. On the other hand, reliance on rotely learned or blindly followed procedures has a number of serious shortcomings, particularly in the face of increased machine complexity. The interaction of multiple subsystems in current technologies makes it difficult, if not impossible, for those who develop directive procedures to anticipate all the conditions under which machines may malfunction. Therefore, fully specified procedures are an inherently inadequate resource for troubleshooting. Diagnosis and repair procedures that are learned without an adequate understanding of the machine are difficult both to remember and to adapt when unusual problems arise. Moreover, the ability to apply procedures appropriately depends on accurately recognizing and categorizing failure symptoms. Without an adequate model of how the machine works, it is difficult for technicians to distinguish among symptoms that may differ only in subtle ways.

To be effective, training should take into account human cognitive strengths and weaknesses. Rote training methods place excessive demands on memory while failing to take advantage of human capacities for reasoning, problem-solving, and reconstructing forgotten procedural knowledge from conceptual understanding. Training should also be designed to provide a foundation for subsequent training; retraining should be able to "bootstrap" on previous instruction. A classic result in research on learning has shown that old rote knowledge structures can interfere with the acquisition of new ones (e.g., Poffenberger,1915). We hypothesize that relating troubleshooting procedures to causal models of how machines work will provide an improved basis for learning about new machines.

1.1. Procedure Following vs. Troubleshooting

Our analysis of materials and classroom training and observation of service personnel in the field shows that the skills for following procedures and for troubleshooting are distinct but not entirely unrelated. Beginning technicians may have little more than the basic ability to use technical documentation. Xerox service documentation relies on directed (rote) procedures but includes more qualitative descriptions (mainly illustrations) of machine components. The troubleshooting done by experts involves problem solving and reasoning from principles derived from causal mental models of machine operation and from basic troubleshooting strategies. Troubleshooting also makes use of other knowledge, such as immediate perceptual cues, data from customers, and experience. Experienced troubleshooters may resort to procedural documentation when needed. Our approach is to recognize this continuum of skill and to develop instructional strategies that help technicians move along the skill spectrum, integrating more qualitative knowledge and troubleshooting skills as they go.

1.2. Research Domain

Though our ultimate goal is to construct generalizable methodologies for improving the development of technical training, we begin by focussing our research on a particular machine. We are concentrating on the training of Xerox technical representatives ("tech reps") to diagnose and repair the Xerox 1075 copier, in particular its xerographic subsystem.

We chose xerography and the 1075 copier for a variety of reasons. First, diagnosis of the 1075 provides an excellent domain for understanding the cognitive demands of troubleshooting complex equipment since the copier incorporates electronic, mechanical, electrostatic, electrical, and optical subsystems. The xerographic process itself is the central function of the machine and, as such, is connected to many other subsystems. The 1075 xerographic subsystem also employs software that makes use of multiple forms of feedback control.

Skill in troubleshooting the xerographic subsystem has clear practical importance since xerographic faults are frequent and open-ended. Because xerography is not a precise domain, it provides an interesting problem space for causal reasoning. Finally, xerography is a generic process found in most copiers; understanding how it works involves knowledge that is transferrable across machines.

1.3. Research Goals and Methodologies

The purpose of our research is to explore ways in which the procedural skills for troubleshooting complex machines can be cost-effectively learned as meaningful structures rather than as rote sequences of actions. Our major thesis is that procedures can be *semantically rationalized* (explained) in terms of

(a) the characteristics of the task,

(b) the tech rep's mental model of the machine,
(c) the cognitive limits of the tech rep, and
(d) the procedural context.

Both training and troubleshooting occur in a rich cognitive and social context. Accordingly, we are employing multiple methodologies for investigating how best to apply computational technology to the teaching of procedural and troubleshooting skills. These methodologies include qualitative modelling and knowledge representation from AI, psychological modelling, anthropological field work, instructional design, and interactive system design. We will employ these methodologies in the following idealized project sequence:

(1) Develop on-line representations of rationalized copier-repair procedures and qualitative models of how copiers work. This work will make use of AI modelling representations as well as behavioral analysis of how technicians understand copiers and the troubleshooting process.
(2) Employ these representations in instructional systems, including on-the-job performance and learning aids. This work will require the application of instructional and interactive system design techniques as well as an understanding of communication within the tech rep community .
(3) Deploy these systems in Xerox training centers and other sites.
(4) Evaluate their effectiveness using protocol analysis and other evaluative techniques.

This approach leads to two main classes of research:

Skill Analysis. A basic insight of cognitive learning research is that much of the knowledge and skill that make up expertise in a domain is tacit (Brown and Burton, 1978, 1986), making it difficult to transmit to students through instruction.

Accordingly, much of our work has focussed on using computational tools and a range of techniques to extract, analyze, and represent the structure of existing diagnostic procedures (Section 2), the nature of troubleshooting as practiced by expert field service personnel (Section 3), and the nature of mental models of the machine and their role in causal reasoning (Section 4). These research efforts provide the underpinnings for the design of instruction aimed at helping students to integrate procedural knowledge more effectively with conceptual and strategic knowledge.

System Development. At the same time, a major focus for us is to experiment with computer-based systems that can aid in capturing and transmitting knowledge to service personnel. Our aim is to identify how AI technologies might facilitate the building of machine-specific knowledge bases from which to create new learning materials and tools. In addition, we have begun to consider how these knowledge bases might be transformed into community knowledge bases that can be accessed, modified, and refined by technicians in the field (Section 5).

2. PROCEDURES

2.1. Fault Isolation Procedures (FIPs)

A large part of the skill required for procedure following lies in knowing how to use documentation effectively. The service documentation for the Xerox 1075 copier consists of an interrelated set of parts. The first part is Call Management, which specifies a set of general maintenance procedures and initial diagnostic tests. These tests provide gross clues about the possible location of a fault and direct the tech rep to the second part, the Fault Isolation Procedures, or FIPs (Figure 1). The FIPs are cross-referenced to another part, the Block Schematic Diagrams (Figure 2), which shows the electrical, mechanical, and feedback pathways of each subsystem and provides input and output pointers to other subsystems. Specific steps in the FIPs are cross-referenced to yet another part, the Service Data, which contains parts lists and exploded illustrations showing how to make various repairs and adjustments.

The FIPs represent the core of the troubleshooting process for the 1075. They are intended to lead the tech rep from general clues about what is wrong to identification of a specific component that needs to be adjusted, repaired, or replaced. We analyzed the FIPs for the xerographic subsystem of the 1075 for information about their structure, the skills required to use them, and their relationship to other information sources in the documentation and training materials.

A FIP is actually a decision tree. For example, the FIP in Figure 1 is explicitly diagrammed as a decision tree in Figure 3, which more clearly shows its structure. At the end of the FIP are "drop outs," which direct the tech rep to other FIPs or to repair procedures. Each FIP assumes a single-fault malfunction and does not cover problems that may result from multiple faults or complex system interactions. The Call Management documentation uses error codes from the 1075's maintenance panel to direct the technician to a FIP; other than the FIP title, however, there is no indication of the connection between error codes and particular FIPs. There is also no rationale for those conditions which direct the tech rep to another FIP. Thus the tech rep is led from Call Management to FIP and from FIP to FIP without being given much sense of where he is going.

While instructions in FIPs often refer to the appropriate wires, circuits, or components in Block Schematic Diagrams, there is no attempt to relate the information in Block Schematic Diagrams to the diagnostic procedures or to particular procedural steps. Thus much of the information about functional interconnections within and between machine subsystems provided in the Block Schematic Diagrams serves only a tacit and ill-defined purpose in the documentation. There is a mismatch between the knowledge required to interpret the Block Schematic Diagrams and the lack of knowledge assumed in the language and structure of the FIPs. FIPs are highly directive; actions are specified in minute detail, leaving no room for the tech rep to use judgment (intuitions, experience, reasoning) about the procedure as a whole.

Although the FIPs are intended to circumvent the need for a conceptual understanding of the machine, it is clear that the use of strongly directive diagnostic procedures makes major cognitive demands of its own. Because of its complexity, a great deal of effort must be expended in order to learn to use the documentation effectively. This fact is recognized by curriculum designers. The principal objective of the service training courses is to teach the tech reps how to use the documentation. As students they are required to keep a written record of their use of the documentation as they solve training exercises.

It is also clear that at least some understanding of machine structure and function is required to interpret much of the information in the service documentation. Current training courses provides a supplementary text, the *Principles of Operation*, which gives a description of all operations of the machine, including the xerographic process. Although the *Principles of Operation* is surprisingly detailed, it contains no systematic discussion of how to use the information in troubleshooting. Moreover, conceptual understanding is not reinforced through testing, nor is it required in the training exercises. Thus neither the documentation nor curriculum design is successful in integrating diagnostic procedures with a conceptual or causal understanding of the machine.

In spite of the emphasis on FIPs in training and the effort devoted to their development, tech reps in the field do not use the FIPs as intended. Observation and protocol analysis reveal that, instead of following the steps blindly in order, experienced tech reps often scan the tests or the drop outs to identify candidate faults, which they then test. Alternately, they refer only to the Block Schematic Diagrams, suggesting that they use them to support reasoning about the machine or to supplement an internal model of the machine. There is also some indication that reliance on the FIPs carries the social stigma of naivete or incompetence within tech rep culture. Therefore, while directed procedures may be useful for inexperienced tech reps, it appears that they neither substitute for nor effectively support the causal reasoning and strategic thinking of tech reps in the field.

We have begun to look for a way to design and represent procedures in training materials and job performance aids in order to enable tech reps to move smoothly up the skill spectrum from detailed procedure following to expert troubleshooting. Our first step in this research is to analyze the underlying structure of the FIPs using computational tools for representation and for analysis and structuring.

2.2. DARN Procedure Net

As a first step in analysis, we embedded two of the standard 1075 FIPs into the knowledge-based "Diagnostic and Repair Network" (DARN) system (Mittal et al., fothcoming). We had three objectives in the DARN analysis: (1) to look more closely at the structure of a FIP, (2) to consider whether an interactive procedure-representation system would be useful as an online job performance aid, and (3) to assess DARN's effectiveness as the basis for an online community memory.

System Features. DARN was originally designed to aid technicians in the diagnosis and repair of a particular disk drive (see Bobrow et al., 1985, for further details about DARN). DARN encodes a procedure in a directed graph, classifying the procedure steps as net nodes (Figure 4). Each node represents one of four basic types of procedural steps: test, response, control action, and corrective action. DARN provides menu commands for a basic set of actions, as well as query and prompt facilities. During the coding of a procedure, it creates a link between each node and an explanation/annotation window.

DARN has a number of properties that we expected to be useful in representing the FIPs: DARN displays the procedure's branching structure graphically, thereby showing more clearly the relationship between steps. It also allows the user to query the system for additional information about particular procedural steps and highlights the user's position in the procedural net. In addition, it maintains a history of the user's actions during diagnosis, which is useful in monitoring one's own activities.

Problems with DARN. We discovered that DARN has a number of limitations, particularly for the analysis and rationalization of FIPs. As a tool for rationalizing the FIPs, its primary limitation is that it does not provide a rich enough language for explanation. Annotations are currently limited to text, although much relevant information about machine structure and function would be represented best through illustrations and diagrams. Annotations are also local, hanging off individual actions, making it difficult to convey the significance of a set of actions and their interrelationships. Because there is presently no clear theoretical framework for constraining the explanations, they are informal and nonsystematic, depending entirely on the knowledge and explanatory skill of the writer. As a tool for structural analysis, DARN adheres too closely to the surface structure of the existing FIPs. DARN's categorization and structuring of procedural actions show the dependencies and relations of particular types of individual steps to each other, but they do not readily show how a series of steps relates to machine structure and function or to the goals of the overall troubleshooting process.

DARN as a Job Performance Aid. As a procedure authoring environment, DARN has several distinctive and potentially useful properties. These include an ontology of diagnostic procedural steps, an interactive graphical network of dependencies and alternatives, annotation space and history mechanisms, and an underlying control structure for progressing through the network. Our experience with DARN has led to a number of insights about ways to extend its usefulness and to guide future attempts to build job performance aids.

First, in trying to embed FIPs in DARN, it became apparent that the ontology and control structure of procedural actions is closely tied to the characteristics of the failure modes in a particular system. For example, fault diagnosis of the particular disk drive for which DARN was built is conditioned by the fact that the disk drive reveals hard failure fairly quickly. Once a test has revealed a failure, DARN is structured to try fixes in sequence, looping back through the failed test for a close verification of successful diagnosis. In 1075 Xerographic failures, by contrast, observation of a high-level symptom is followed by a detailed search through intermediate symptoms for a component failure. A test revealing a component failure may result in moving to a new diagnostic procedure or to a remote repair or adjustment procedure before returning to the top level for verification. A more broadly useful tool would provide a representation language relating the characteristics of machine failure, testing, and verification to the procedural ontology and control structures.

One important characteristic of DARN as a learning tool and resource for collaborative troubleshooting is its maintenance of a record of the history of diagnostic actions in both graphical and textual form. The textual history collects user prompt messages, system answers, and remedial actions. The graphical history consists of a second browser graph with the technician's progress through the procedural network highlighted in inverse video. Such facilities would be useful both as learning tools, enabling novice technicians to reflect on their efforts in a real troubleshooting session, and as a comprehensible record for anyone coming to the aid of a stalled diagnosis.

Perhaps DARN's most interesting potential derives from its dual nature as an authoring environment and as an interactive troubleshooting guide. Because it provides a language and structure for representing procedural knowledge, it could be modified by a tech rep during a troubleshooting session, thereby making it possible to capture and share field knowledge about machine failures and troubleshooting strategies. This possibility raises some interesting research questions in its own right, such as how to provide tools for building an active community knowledge base that captures the expertise of field personnel while maintaining adequate control over a carefully constructed and rationalized procedure.

2.3. FIP Goal Analysis

In the DARN analysis, we developed an online representation of the FIPs as a procedural net and attached rationale to this. However, the rationale was informal, unconstrained, and limited to individual steps rather addressing the structure of the procedure as a whole. As one step in developing a general taxonomy of rationale, we attempted to represent the FIPs in relation to a specific type of rationale: goal structure. Our aim was to provide a functional context for each FIP step and to uncover the necessary and unnecessary orderings of procedural actions.

FIP Goal Structures. To perform this analysis, we used the NoteCards system (Halasz, Moran, and Trigg, 1987), which allows the user to define electronic notecards with multiple links between them. Linked networks of notecards can be graphically displayed in a browser for an overview. The nodes in a browser can be expanded to show the notecards, which can contain many types of information: text, sketches, graphs, etc. We first represented a standard Fault Isolation Procedure as a network of NoteCards, formatted as a flow chart much like a horizontal version of the FIP representation seen in Figure 3. Each card is one of the actions in the original FIP; however, each card is also a terminal node, or leaf, of a hierarchical goal tree designed to reveal the rationale behind each FIP action. This is illustrated in Figure 5a.

Examining the FIPs revealed that the standard goal is to REPAIR something, beginning at the top level with the copier itself. This goal breaks down into four standard subgoals: PREPARE (status-setting operations such as cleaning or setting parameters), ISOLATE (tests designed to find the faulted subsystem), REPAIR, and WRAP-UP (clean-up operations, logging repair session, etc.). As can be seen in Figure 5a, there is ı systematic recursion on the REPAIR goal, each instance initiating the next round of PREPARE, ISOLATE, REPAIR, and WRAP-UP goals. Representing the FIPs in this way shows level-by-level refinement that provides the basic structure for the FIPs. Actions to isolate the source of a fault lead to the goal of repairing the next level: system (e.g., xerography), subsystem (e.g. cleaning), component (e.g., dicorotron), and part (e.g., coronode).

Each node in the goal structure browser expands to a goal notecard, which states the goal and means to satisfy that goal. These are links to either other goal notecards(subgoals) or to action notecards within a representation of the FIPs as a series of linked actions and tests. Finally, in addition to specifying the action associated with a goal, each goal notecard states the preconditions and postconditions for carrying out that action (see Figure 5b).

Assessment. A primary resource for making sense of procedures is an understanding of their purpose or function within the overall context of the troubleshooting task. The linkages between individual goal frames and FIP actions enable a tech rep executing the procedural steps to find out at any point what goal his actions are satisfying. Through a modification of the graphical display characteristics of NoteCards, we also made it possible for a tech rep to view his position in the goal hierarchy as a whole by highlighting the path from that position "backward" to the top-level goal or "forward" through the subgoals, as shown in Figures 5a and 5b. As with DARN, these features suggest an explicit rationale for individual steps. They also help the user to see the relationship between the current action, previous and subsequent actions, and the larger functional context.

One of the primary constraints in following directed procedures such as the FIPs is that they do not allow, or provide information for, varying the order of procedural steps, even when the order is entirely arbitrary. This inhibits tech reps from using their experience to decide how to go about isolating and repairing faults. Showing explicitly the pre- and post-conditions for carrying out a goal gives the user the opportunity to go directly to testing for a suspected fault; he need only check the pre- and post-conditions to be sure that necessary constraints are satisfied. This allows tech reps to use procedures primarily as a set of ideas and resources rather than as a recipe, which our observations indicate they do anyway. This is one example of ways in which job performance aids can help support tech reps in the strategies they develop in the field.

However, we did find clear limitations in the attempt to represent the FIP goal structure. One characteristic that emerged is that the ISOLATE goal is not systematic. We were unable to identify a standard taxonomy for its subgoals, as we were in the case of the REPAIR goal. One sort of element that should be represented in such a taxonomy is the set of rules for deciding among tests for isolating faults at the next level of subsystem. Such troubleshooting heuristics as "split halves" tests or "do easy tests first" are not captured in the goal analysis. One obvious difficulty in overlaying these heuristics on existing FIPs is that applying them often requires additional knowledge. For example, deciding which test will split the set of candidate faults and isolation tests in half requires knowing something about the structure and function of the machine and about how these are related to the tests. In addition, the goal analysis did not account for all actions. For example, many procedural actions fall under both the isolate and repair goals; e.g., in the process of eliminating candidates for causes, a fault is often repaired.

In the final analysis, it appeared that the FIPs were resistant to post-facto rationalization because they had not been written with the need to rationalize them in mind. While it is possible that continuing a detailed analysis of pre- and post-conditions of the FIPs' procedural actions would lead to a better rationalization, it seemed more worthwhile to concentrate our efforts on developing rationalized procedures rather than rationalizing the existing ones. As a result, we have begun to explore the underpinnings for rationalized procedures -- i.e., a model of the troubleshooting process based on field observations and causal machine models.

2.4. Simplified Documentation

Another aspect of our investigation of procedures involved considering alternatives to the heavily directed, non-rationalized FIPs in current documentation. One hypothesis is that procedures need not be completely explicit if tech reps were trained appropriately in the principles of machine structure and function. Paul Larsen, a trainer at the Xerox International Center for Training and Management

Development in Leesburg, Virginia, provided a set of simplified documentation based on this principle, designed for the 3300 copier. Although the 3300 is a much smaller and simpler machine than the 1075, we examined this simplified documentation for possible relevance to rationalizing procedures for the 1075.

The 3300 documentation provides an extremely compact representation of the entire machine, consisting of just two large sheets. The first consists of three cross-referenced drawings: one shows the xerographic process and the paper path, the second shows all mechanical systems, and the third shows the electrical system. Figure 6a is a vastly reduced copy of this, and Figure 6b shows the representation of the developer in all three drawings. The representation of the xerographic process shows the succession of events on the photoreceptor and the paper, separated by sketches suggesting the components which perform this process. The juxtaposition of xerographics and paper path makes explicit the sequence and timing constraints of the two processes. The component sketches have tags giving real component labels and pointers to their appearance in the mechanical drawing or to their connections in the electrical drawings. The xerographic and paper path drawing also includes labels associated with either components or process steps which point to lists of known problems on the second sheet.

The second sheet of the simplified documentation is a diagnostic guide for trouble-shooting copy defects. Use of this guide, shown in Figure 6c, begins with a list of specific defects found on copies of standard test patterns. Each defect has a suggested flow, or sequence, for trouble-shooting, including a mix of additional tests and suggested known problem areas to investigate. One difference from the 1075 Fault Isolation procedures is that the entire sequence is visible at a glance, and a quick check of the problem lists provides the names of all the usual suspects. This accessibility permits the technician to modify the sequence according to experience or personal predilection; the compact representation permits one to try this improvisation while being able to keep one's place in the problem space. The compactness may also encourage the associative mappings between symptoms and causes that experienced tech reps exhibit. The system is like the 1075 FIPs, however, in that no rationale for the sequence or the association of suspect and fault is given.

The documentation does assume that the tech rep possesses certain skills and understanding about the machine and xerography. Procedures for testing suspected components are not specified; this assumes, for example, that the tech rep knows how to check the ground connection of the prefuser transport. This assumption of competence is in marked contrast to the apparent presumption of incompetence implicit in the present documentation.

Assessment. The simplified documentation extracts and economically represents the essential information from the source documentation, thereby reducing the cognitive overhead in the use of the documentation. Through its integration of some of the knowledge essential for troubleshooting (machine knowledge, symptom-cause mappings, diagnostic tests), it may provide a useful cognitive tool for learning and practicing troubleshooting. Finally, it affords flexibility in procedure following. Thus it can be used by tech reps at all levels of expertise, from those who follow procedures to those creating their own troubleshooting strategies.

2.5. FAST/FMEA Analysis

Having examined the issue of rationalizing existing procedures, we also consider the task of creating the theoretical basis and knowledge engineering tools for developing rationalized procedures. One step in this direction is to view data from engineering design as a source of the machine knowledge that must be represented in rationalized procedures.

Xerox uses two engineering design techniques to coordinate the knowledge of subsystem specialists and to arrive at cost-effective decisions on the design of new copiers. FAST (Functional Analysis System Technique) develops a functional model of normal machine operation by explicitly representing the role of individual parts and how they interact with other components and subsystems. FMEA (Failure Mode and Effects Analysis) yields a hierarchical model of how the machine can fail. It includes information about specific faults of parts, the symptoms they produce at several levels (from the customer's view to internal evidence), and methods for diagnosing these faults.

These engineering analyses contain information about machine design and construction that could be used to create FIPs. One of the difficulties in using them in their current form is that information about faults is not organized by symptoms. Thus, although a given symptom can have multiple causes, there is currently no attempt to bring that information together -- a tedious task without computer aids. Another limitation of the analyses in their current form is that they do not provide information about multiple or co-occuring faults, a flaw that is shared in the current FIPs.

Assessment. We examined these engineering analyses and began experimenting with ways of capturing the essential information on-line. By using the NoteCards system, we created the FAST/FMEA system (see Figures 7a and 7b). These figures illustrate the kinds of data that are captured in the design and development process. We have found that once this information is created, it is often discarded at the next level of machine development. This is apparently a common occurrence within large industrial development projects; information is constantly re-discovered, re-created, and re-rationalized. The FAST/FMEA system illustrates several significant points: (1) design and development information (including design rationale) need not be lost during a machine's evolution from initial design through manufacturing, but can be carried along with the machine; (2) design information can be an effective starting point for the creation of effective service troubleshooting documentation; (3) the availability of design rationale information can be valuable for the creators of subsequent service training materials; (4) easy access to design and development information can be used to relate design time decisions to information about design failures uncovered in the field.

3. TROUBLESHOOTING

3.1. Field Observation of the Troubleshooting Process

As stated earlier, one component in the development of rationalized procedures is a coherent notion of the task that the user is carrying out -- in this case, diagnosing or troubleshooting a complex copier. Consequently, it is important to construct a model of the troubleshooting process. Our emphasis has been on developing a realistic model based on troubleshooting as practiced in the field. We chose this emphasis for several reasons related to our instructional goals. First, effective instruction needs to teach the skills and knowledge relevant to good troubleshooting; one way to determine what those skills are is to analyze the knowledge that expert troubleshooters possess. Second, instructional strategies might be designed to take into account differing cognitive styles; observing experienced tech reps may lead to the characterization of some individual differences in approaching diagnostic problem-solving. Finally, in instruction as in any form of communication, it is important to address the audience in a manner that will make sense to them; one therefore needs to understand how tech reps think about machines and about the troubleshooting process.

In order to construct a descriptive model of real troubleshooting processes, we observed and recorded tech reps repairing copiers in the field. We used an ethnomethodological approach in the analysis of behavior in these troubleshooting sessions. Our goal was to observe what real tech reps do, to identify the terms they use to talk about their activities, and as far as possible, to infer how they think about the process.

Our present focus on developing a descriptive model of troubleshooting does not preclude developing a normative one. One of the open questions that needs to be addressed is whether instruction should teach trainees to do what current tech reps already do, or whether we should attempt to modify their present approach. Our hope is that field observations and analyses can help to characterize the strategies that tech reps typically employ and to see how these differ from "optimal" strategies. Field observations may also help to identify snags and difficulties -- places where tech reps need help in the form of job performance aids or new instructional approaches. Descriptive models are needed to reflect the variety of constraints and resources that tech reps find in the real world of troubleshooting.

In the following two sections we discuss two troubleshooting models that we have developed thus far. The first is a simplified and idealized data flow model, which illustrates the basic types of information used in troubleshooting and the general cognitive activities that move the tech rep from one step to the next. The second model is a more elaborate version of the first. It attempts to characterize the types of knowledge and cognitive processes that tech reps use in troubleshooting, as well as feedback loops and other relationships between parts of the process.

3.2. ESF Model of Troubleshooting

The ESF (Evidence-Symptoms-Faults) data-flow model (see Figure 8) shows that troubleshooting begins with a set of evidence {E} about what works and what doesn't work in the machine. This might include such easily accessible information as gross copy quality defects, data from customers, error codes, etc. An arrow in this model represents a general function -- i.e., a whole collection of activities that has the function of producing new data from existing data. In moving from evidence to symptoms {S}, the tech rep is attempting to refine or characterize existing evidence into a set of symptoms. Symptoms provide the basis for generating a set of possible faults {F}, from which there are activities to isolate a particular fault F.

Perhaps the most interesting aspect of the ESF model is the distinction between evidence (which we originally called "appearances") and symptoms. Identification of symptoms is the necessary intermediate step between characterization of some global problems or evidence (e.g.,the copies are unclear) and generation of a set of specific candidate faults. At present, however, the standard language for talking about machine faults does not include a precise characterization of symptoms, nor are symptoms described in a way that emphasizes their causality. One goal for future research, therefore, is to develop a language for symptoms that will better relate them to the underlying machine faults. Preliminary research has focussed on copy quality defects as an interesting set of symptoms resulting from faults in the xerographic subsystem.

Copy Quality Taxonomy and Language. The current language for describing machine faults and copy quality defects provides no cognitive support for reasoning from symptoms to causes; rather, it presupposes the underlying cause. For example, a typical label for a machine fault is "pinch roller defect." In such cases, the causal relationship between the external evidence provided by symptoms and the faulted component itself is left implicit. Rather than learning a language for describing defects that begins with the observable behavior of the copier and provides "hooks" for causal links to specific components, tech reps must construct their own causal links or associations between observable symptoms and defect descriptors that are rotely learned. Development of an appropriate language for copy quality defects and other symptoms could help focus attention on the causal relations between component functions and interactions, component failures, and observable defects.

As a first step toward developing a causal language for describing symptoms, we considered the characteristics of a set of observable symptoms that might provide clues to their underlying causes. A taxonomy of copy-quality defects developed by Paul Larsen provides a first level of distinctions among types of copy defects (see Figure 9). For example, lines on the copy may be either streaks or strobes; strobes in turn may be manifest as skips or smears.

We then specified a set of distinguishing characteristics for each defect that point to a faulted component or subsystem. For example, we know that a smear is caused by a difference in speed between the photoreceptor, which carries the toned image, and the copy paper. The peculiar characteristics of smears are causally related to the machine fault so as to help narrow down the field of possible faults.

Thus, if a tech rep observes a smear in a test pattern, he can examine the toned photoreceptor image. The fact that the smear does not appear on the photoreceptor indicates that it results from an interaction between the paper and the photoreceptor at the point of transfer, not from a disturbance of the toned image prior to transfer. A smeared image may also be stretched, which directly suggests a difference in speed between the paper and the photoreceptor. If the stretching varies in width from one side of the paper to the other, it indicates that the paper has slipped during transfer and therefore that the mechanism that brings the paper in contact with the photoreceptor may be at fault.

Other Characteristics of the ESF Model. Even with a good copy quality, symptoms are often ambiguous. Not only is it difficult to distinguish symptom characteristics at a fine enough level of detail, but symptoms often have multiple possible causes. One feature of the ESF model is that it emphasizes the possibility of a set of candidate faults, which are derived from initial symptoms and then refined by further evidence as the troubleshooting session progresses.

The ESF model does not attempt to differentiate the activities that move the tech rep through the troubleshooting process, nor to specify the order in which these activities occur. The work of isolating faults, for example, might include executing diagnostic tests and isolation procedures. Isolation activities are also likely to lead to looping back and reinterpreting evidence and symptoms and ultimately revising the set of fault candidates. Since this model characterizes only information flow (not control), it encompasses different troubleshooting styles.

Control Structure and Troubleshooting Styles. According to our observations of and discussions with expert troubleshooters, problem-solving style may differ considerably from one individual to the next. For example, we interviewed two of the 1075 trainers -- both expert troubleshooters -- at the Xerox training center in Leesburg and found very distinct approaches. Both of the experts had extensive field experience, deep understanding of how copiers work, and expertise in teaching copier repair. However, in terms of our troubleshooting style, one expert quickly moved from observing symptoms to generating and testing a set of faults, while the other concentrated effort on discovering and refining his knowledge of the symptoms before generating and testing a much smaller set of faults.

The control structure for a realistic troubleshooting model must therefore be a variable function of problem-solving style. It is not clear from empirical evidence that either of these general strategies is preferable; both are highly effective. We thus have chosen to focus first on finding ways to teach knowledge about the elements and processes involved in troubleshooting, leaving the choice of control structure or strategy for using this knowledge up to the individual tech rep.

Nevertheless, we shall not ignore the issue of troubleshooting strategy. It may be that explictly teaching alternative strategies will be helpful to trainees in integrating and using their knowledge. However, there is no evidence that a given strategy is superior and thus no evidence that movement within the troubleshooting process should be "hardwired" into instructional activities or models of troubleshooting. Expert strategy seems to involve keeping track of all the elements of troubleshooting and employing *reasonable* strategies to keep moving in the troubleshooting process, rather than following one routine.

3.3. ESC Model of Troubleshooting

The aim of the ESF model was to identify some of the basic elements of troubleshooting. In a second iteration, we attempted to describe more completely both the processes and the knowledge structures that interrelate knowledge components in troubleshooting.

Characteristics of the ESC Model. The ESC model (see Figure 10) includes the same basic elements of troubleshooting as the ESF model: troubleshooting moves from observable evidence {E} and symptoms {S} to faults. Underlying faults are called causes {C} to reflect greater emphasis on causal reasoning in this model.

Process arrows show more clearly the distinctions between types of cognitive activities that move tech reps from one type of data to another. Bracketed elements specify knowledge sources (past experience, mental model of the machine, results of tests and fixes) and lighter lines show how these relate to cognitive processes. With this scheme, one might recognize a symptom's cause from compiled knowledge or experience. One might also infer an intermediate or underlying cause by "running" a mental model of the machine or particular subsystem.

The ESC model accounts for many more of the observable behaviors of tech reps troubleshooting in the field. It shows the role of a mental model of the copier, as well as the role of other knowledge, such as experience. It makes explicit the cyclic process of information gathering and shows the role of predicting and testing additional symptoms {S?} in order to verify diagnoses or generate new ones. ESC also allows for the generation of alternative accounts of new evidence, including evidence from failed repairs -- that is, it recognizes the need for reinterpretation of prior evidence and how this process can refine the set of symptoms. Finally, ESC includes the notion of intermediate causes in the set of possible causes to be maintained by the tech rep.

Examples from Protocols. Figure 11 shows protocol segements gathered during actual troubleshooting sessions in the field and relates them to the ESC model. The verbal exchanges of two tech reps during a joint troubleshooting session were recorded on tape by an anthropologist, himself a former technician, who occasionally asked questions or participated in the troubleshooting process in order to clarify the meaning of events. Protocol analysis is still in progress, but we were able to extract specific instances that illustrate some of the processes we hypothesize in the ESC model.

The first example captures the problem of interpreting evidence (in this instance, error-log evidence) to discover the important symptoms. For this case, the tech reps must find some common symptom through interpreting multiple pieces of conflicting evidence. Example 2 takes place when the tech reps are in the middle of various diagnostic tests suggested by the initial set of symptoms and evidence. They have identified the 5-volt line {C} as faulted, but they are trying to specify the intermediate cause of the fault by recourse to symptoms derived from diagnostic tests.

Example 3 provides a view on the process of generating and testing predicted symptoms. The tech reps move from initial evidence {E} ("click, click, click") to a symptom {S} that provides enough information to hypothesize a general cause {C} (power problem). Elaboration of the cause (distribution of power) predicts that not only would "hits" be multiple, but they would occur across the board {S?}. The report of T1 confirms that this is in fact what happened (during an earlier troubleshooting session) and that these hits are random. The tech reps now have a refined set of clear symptoms (random, multiple hits across the board) and a general diagnosis of cause (power distribution). They are now ready to move on through further tests, elaborations, and revisions to isolate a particular fault.

Causal Stories. A key observation underlying the ESC model is that, among experienced tech reps, diagnosis of faults is embedded in what we call "causal stories." We conceive of these stories as elaborate knowledge structures that relate observable evidence {E} and symptoms {S} to causes {C} through various kinds of knowledge about the machine. Causal stories serve a number of important purposes in troubleshooting. First, they rationalize evidence and symptoms in terms of candidate causes. In the process of elaborating the causal story, the tech rep moves via processes such as inference, recognition, and prediction to symptoms and causes, with the aim of producing enough evidence to justify a repair. This can be especially important when the repair is an expensive or time-consuming one.

Having performed a fix, the tech rep can sometimes verify his diagnosis by testing for symptoms. However, in the case of intermittent problems, the completeness of the causal story is often the only "proof" the tech rep has that the underlying fault has been repaired. Thus the causal story serves a post facto purpose of assessing the probability that a given diagnosis was accurate. Furthermore, causal stories provide an accessible source of information and organization to support further troubleshooting if the first fix is unsuccessful. Finally, causal stories provide a means for explaining and sharing knowledge with other tech reps.

Understanding how causal stories are constructed and transmitted within the tech rep culture can provide important insights for designing instruction. One implication of the use of causal stories by successful tech reps is that our instructional goals must include helping trainees become articulate about the diagnosis of copier faults. Analysis of causal stories in protocols will provide insight into the types of reasoning employed by successful tech reps, their use of various knowledge resources, and the processes of revising a story in the face of new evidence. This research will feed back into the construction of troubleshooting models, and will provide a basis for possible development of an articulate expert as a component in an intelligent coaching system.

4. QUALITATIVE MODELS

Our observation of tech reps supports cognitive science research indicating that mental models are a primary resource for understanding the behavior of complex systems and for causal reasoning involved in diagnosing machine faults. Our interest therefore centers on investigating ways to build on-line simulation models of xerography. The purpose of our research is to to simulate machine behavior from which trainees can learn basic component functions and relationships by providing a basis for qualitative causal mental models that can be transmitted to novice tech reps.

4.1. ARIA Qualitative Simulation Model

Our first attempts to construct qualitative simulation models of xerographic systems began with ARIA. (See Figure 12 for the ARIA simulation of the 1075 xerographics subsystem.) ARIA is a system for building qualitative simulation models by providing a language in which the qualitiative behavior of system components can be specified, the component interactions specified, and the components and their states linked to graphic displays. It also allows the author to develop different abstractions of the same underlying qualitative model, and so can provide a basis for studying comparison issues such as cognitive versus physical fidelity and the understandability of visual versus symbolic representations.

System Structure. The basic building blocks of ARIA are *objects*; system behavior derives from a set of primitive *Q-functions* that bring about changes in the state of an object. Figure 13 shows the structure of an example ARIA object. (For the details of ARIA's construction, see Shrager et al., 1986.) Objects represent actual machine components; they usually have *spatial location*, although some objects move through space and so are not fixed in space. Objects also have *parameters*, which are local, typed values characterizing the state of the object. Value type specifies the dimension along which values vary. For example, charge on a coronode is a typed value; the type is voltage and the value is some qualitative value such as highly charged or weakly charged. Qualitative scales for parameters are of three kinds: nominal (the status of a component might be broken or fixed), ordinal, describing ordered values (such as a positive/zero/negative charge on the photoreceptor belt), and continuous. Continuous scales are closest to quantitative scales and describe such xerographic objects as toner supply level. In addition to spatial location and parameters, objects carry attachments to other objects via their parameters. An attachment may be *direct*, as when a wire or pipe connects two components in the machine, or it may be *spatial*, meaning that the state of one object has an effect on the state of another object when they have specified spatial relationships.

Object behavior derives from *Q-expressions*, which determine behavior in two ways. First, a Q-expression may change the location of an object. Thus, Q-expressions are responsible for moving the segments of the photoreceptor to different locations in the model, where they can be acted upon by other objects. In addition, a Q-expression specifies input-output value relations. An example of a Q-function is *Q-change*, which instructs the model to change a value on an object's parameter scale based on the conditions captured in the relevant Q-expressions.

Figure 14 is a graphic depiction of basic xerographic objects and attachments in ARIA-1075. Objects are boxed and attachments are labelled as direct or spatial. The arrows indicate directional flow through the system.

Characteristics of ARIA. ARIA is structured so that the object evocation sequence needs to be explicitly declared. That is, to run a model, one needs to evoke the behavior of each component. ARIA does not prescribe the method to do this.

In order to model the 1075 xerographic system, ARIA must solve some practical problems not commonly found in qualititative physical models. One problem is the need for a language to describe images on the photoreceptor as a function of the parameters of relevant objects. We have begun to develop a Q-language of parameters and their values that adequately describes images and their defects. The 1075-ARIA model also requires modelling the feedback behavior of the DSS, a system that measures test patches on the photoreceptor and uses the information to control the charge voltage. Finally, in order to model transfer (the movement of the toned image from the photoreceptor onto the copy paper), ARIA requires a fictitious object that causes transfer to occur. 1075-ARIA has a transfer process "object" that moves the toner to the paper when certain parameter values are satisfied.

4.2. Causal Reasoning in Troubleshooting

The development of qualitative simulation models for instruction raises the question of how they are to be used to teach trainees. How detailed a model of the device does a novice technician need to have in order to troubleshoot? We believe that a less than total understanding will suffice. The main goal of troubleshooting instruction should be to teach an adequate understanding of causality and the propagation of effects between various components.

One way to do this is to rely on trainees to infer causal relations simply by observing changes on the screen as the model is running. Alternately, we might provide an explicit causal model. The advantage of the latter approach lies not only in the causal connections that are made explicit in the model, but also in the necessary components of a knowledge-based troubleshooting tutor that are addressed.

For example, Figure 15 shows a causal reasoning chain that presents possible causes of a particular copy quality defect (see Figure 15). The causal chain provides a more precise characterization of the informal causal story that relates symptoms to their underlying causes. As an example, one characteristic of this causal model is that it differentiates causal reasoning requiring *qualitative inferences* from that based on *spatial chaining* derived from physical proximity (Farley, 1985).

5. INSTRUCTIONAL DEVELOPMENT

Research in this project has two primary purposes: 1) to specify the kinds of knowledge that are needed to diagnose and repair complex systems as a basis for instructional design; and 2) to construct computer tools and simulations that can serve as learning environments in a variety of settings. We have begun to experiment with ways of transmitting troubleshooting and other knowledge using simulation models and other computer-based tools.

At this point we have a general outline of the sorts of knowledge that tech reps require in the field, and generally specified learning goals. However, these already suggest some of the elements of instructional activities required to teach troubleshooting and to treat issues surrounding design of learning environments.

Mental Models. One important instructional goal is to help trainees build a useful mental model of the system. This requires knowledge of the function and structure of system components, including their causal relationships. In particular, trainees must learn how interacting components produce the varieties of machine behavior. At present, trainees may explore an existing simulation model (ARIA-1075). They can watch it run, examine component parameters to see their important structural characteristics, and alter component parameters and observe the effects. In the futıre, we expect to be able to allow trainees to look at the simulation in varying degrees of detail, "zooming in" on the workings of individual components and "zooming out" to see the effects of a component's behavior within the system as a whole. Our field observations suggest that causal reasoning in diagnosing complex systems can make use of multiple, partial, and even "distorted" mental models. The incomplete or erroneous models can be useful when the missing or wrong details are not relevant to the problem at hand. Further development of ARIA and other modelling techniques may enable us to build instructional environments that provide alternate causal views on the workings of the system.

Transfer of Knowledge. Strong mental models of a given xerographic system should aid tech reps who are training on new machine by facilitating the transfer of general technical understanding during retraining. Further development of the ARIA schema for building xerographic models will make it possible to build xerography construction kits. Such a construction kit would incorporate the system constraints that arise from the basic technology of xerography and from general structural considerations. By separating xerography-based constraints from machine-specific ones, we expect to provide a useful basis for the acquisition of a generic mental model of xerography. Such a model should not only aid tech reps in causal reasoning about xerographic faults, but also provide an important knowledge structure to which machine-specific knowledge can be efficiently attached. Other instructional activities designed to improve retraining methods center around providing facilities for trainees to transform copier models to represent different existing copiers. This instructional strategy attempts to build directly on the trainees' knowledge of the old machine. Rather than constructing a completely new mental model from interactions with a new simulation, the trainee will be able simply to alter an existing mapping at the points of difference.

Troubleshooting Skills. Another general instructional goal for training tech reps is the development of troubleshooting skills. Cur analysis of tech reps in the field suggests a number of specific cognitive skills that must be addressed in the design of learning activities. For example, trainees must be able to identify non-normal component states. This is sometimes done by direct examination of a malfunctioning component, but more often a component fault is manifested by the appearance of some symptom. Instructional activities might require the trainee to identify the whole set of symptomsassociated with a given component fault, at different levels of observable behavior.

Another important aspect of troubleshooting is the need to move from gross symptoms, such as a copy quality fault, to generation and testing of hypotheses about causes. We have already implemented an instructional activity in 1075-ARIA that requires the trainee to find the cause of a randomly generated symptom. As in a real troubleshooting sesssion, trainees must gather information in order to delineate symptoms more precisely as a basis for hypothesizing causes. Further information is gathered through means such as hypothesis testing or diagnostic tests. At present, trainees are able only to examine object parameters, but eventually they should be able to simulate various diagnostic tests or examine component models at finer levels of detail. As in field troubleshooting, information gathering is costly; trainees are required to pay tokens for peeking at parameters, making a guess about the cause, or activating the model. As we develop a better understanding of generic troubleshooting strategies in the xerographic domain, it should be possible to develop a more sophisticated system of rewards and penalties for strategic moves.

A complementary instructional activity currently in prototype in 1075-ARIA requires trainees to generate a symptom, rather than find a cause. The trainee is instructed by the system to produce a particular copy quality fault by adjusting component parameters, thus exercising the specific skill of constructing and testing hypotheses. The activities of finding a fault and generating a symptom focus on different kinds of causal reasoning. In finding a fault, trainees must construct one or more backward causal chains from observable symptoms to possible underlying causes. The constraint of paying for information implicitly encourages students to mentally test their hypotheses or guesses by reasoning forward again, but the initial task focusses on backward chaining. Generating symptoms, on the other hand, focusses the student's attention explicitly on reasoning forward through a causal net toward some goal. Both kinds of reasoning are employed in troubleshooting, although different troubleshooting styles suggest that people may vary in their facility with one or the other. Providing focussed practice on both types of reasoning skills should improve the tech rep's problem solving facility.

Composite Skills vs. Realistic Troubleshooting. Decomposition of troubleshooting skills makes it possible to provide practice in specific aspects of the overall process without distraction by the numerous considerations that attend a real troubleshooting session. However, it is important to keep in mind that the skills are practiced as an integrated whole and that management of the entire process is one of the requisite abilities for successful troubleshooting. We need to know more about what guides tech reps in the use of composite skills within a realistic troubleshooting session, how they keep track of multiple hypotheses and assess their changing plausibility, and how they integrate the identification of new symptoms into an ongoing "causal story."

In the field, one purpose of causal stories is to make explicit various chains of causal reasoning and to provide some organization for the tech rep's emerging knowledge about the state of the particular machine. If we can improve the trainee's ability to articulate hypotheses, assess symptoms, and engage in causal reasoning, we may provide an important advance in current training methods. Computer techniques for recording and organizing traces of a user's problem-solving activities may encourage trainees to reflect more consciously and coherently on their troubleshooting processes.

Rationalized Procedures. A key goal of this project is to devise methods of teaching complex troubleshooting procedures based on understanding. Procedures are designed to capture an ideal model of troubleshooting by providing an efficient strategic path through the problem space. One way to help tech reps understand the structure of procedures is to encourage the practice of deriving rationalized procedures by modifying the history of their own or another's problem-solving activity. Computer traces of problem-solving actions can be represented and edited to create an ideal, or proceduralized, troubleshooting path. A prerequisite to building such a learning environment is the development of appropriate representations for procedural actions.

A second method of teaching rationalized procedures is the development of an *articulate expert* -- a knowledge-based expert system capable of explaining procedural steps (or coherent blocks of steps) in terms of troubleshooting strategies and causal models of the copier. This instructional strategy rests on a successful outcome for most of the research vectors for this project. Explicitly rationalized procedures require a clear model of troubleshooting strategies and processes, the development of causal networks linking symptoms, intermediate causes and underlying causes, and a schema relating mental models of systems to diagnostic actions.

BIBLIOGRAPHY

1) Morris, N.M. and W. B. Rouse, "Review and Evaluation of Empirical Research in Troubleshooting". Human Factors 27:503-530, 1985.

2) Poffenberger, A. T., "The influence of improvement in one simple mental process upon other related processes." Journal of Educational Psychology 6, 1915.

3) a. Brown, John S. and Richard R. Burton, "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills", Cognitive Science 2:155-192, 1978.

b. Brown, John S. and Richard R. Burton, "Reactive Learning Environments for Teaching Electronic Troubleshooting", to appear in Rouse, W.B., (ed.) Advances in Man-Machine Systems Research, Vol. 3, Greenwich, Ct.: JAI Press, 1986.

4) Mittal, Sanjay., Daniel G. Bobrow, and Johann De Kleer, "DARN: A Community Memory for a Diagnosis and Repair Task", forthcoming.

5) Halasz, Frank G., Thomas P. Moran, and Randy Trigg, "Notecards in a Nutshell", submitted to CHI and GI, April, 1987.

6) Orr, Julian E.,"Narratives at Work: Story Telling as Cooperative Diagnostic Activity," to be published in the Proceedings of the Conference on Computer-Supported Cooperative Work, Austin, Texas, December 3-5, 1986

7) Bobrow, Daniel G.(ed.), Qualitative Reasoning about Physical Systems, Cambridge, MA: MIT Press, 1985.

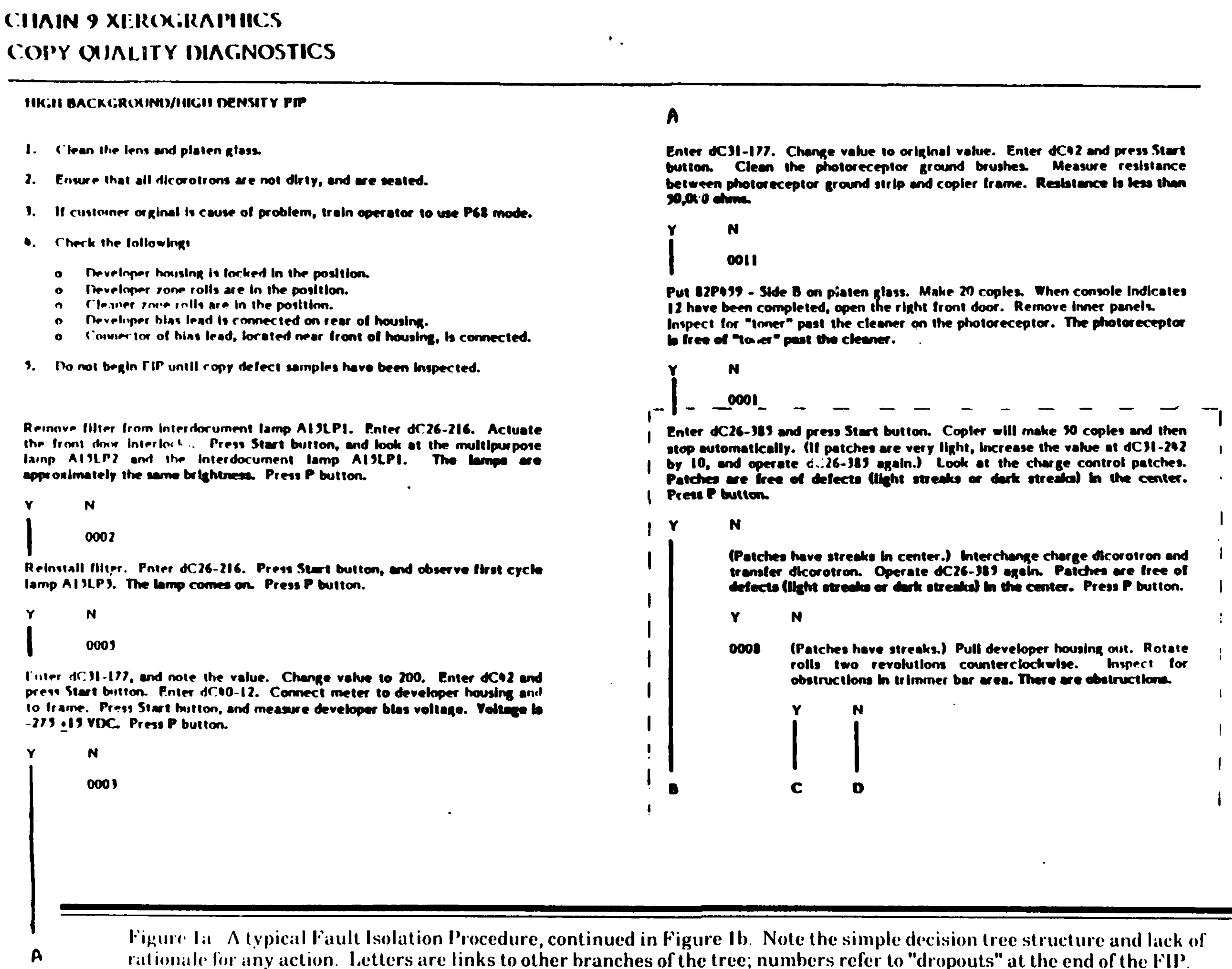
8)Gentner, D., and Stevens, A., (Eds.), Mental Models, Erlbaum, Hillsdale, N.J., 1983.

9) Farley, Arthur M., "Diagnostic Mechanism Modeling". Unpublished manuscript, Palo Alto, CA: Xerox Palo Alto Research Center, 1985.

LIST OF FIGURES:

Figure 14 -- All ARIA functional connections

Figure 15 -- Causality chain for Copy Quality problem



CHAIN 9 XEROGRAPHICS
COPY QUALITY DIAGNOSTICS

HIGH BACKGROUND/HIGH DENSITY FIP

1. Clean the lens and platen glass.

2. Ensure that all dicorotrons are not dirty, and are seated.

3. If customer orginal is cause of problem, train operator to use P68 mode.

4. Check the following:

   - o Developer housing is locked in the position.
   - o Developer zone rolls are in the position.
   - o Cleaner zone rolls are in the position.
   - o Developer bias lead is connected on rear of housing.
   - o Connector of bias lead, located near front of housing, is connected.

5. Do not begin FIP until copy defect samples have been inspected.

Remove filter from interdocument lamp A15LP1. Enter dC26-216. Actuate the front door interlock.. Press Start button, and look at the multipurpose lamp A15LP2 and the interdocument lamp A15LP1. The lamps are approximately the same brightness. Press P button.

Y N

0002

Reinstall filter. Enter dC26-216. Press Start button, and observe first cycle lamp A15LP3. The lamp comes on. Press P button.

Y N

0005

Enter dC31-177, and note the value. Change value to 200. Enter dC02 and press Start button. Enter dC00-12. Connect meter to developer housing and to frame. Press Start button, and measure developer bias voltage. Voltage is -275 ±15 VDC. Press P button.

Y N

0003

A

A

Enter dC31-177. Change value to original value. Enter dC02 and press Start button. Clean the photoreceptor ground brushes. Measure resistance between photoreceptor ground strip and copier frame. Resistance is less than 50,000 ohms.

Y N

0011

Put 82P059 - Side B on platen glass. Make 20 copies. When console indicates 12 have been completed, open the right front door. Remove inner panels. Inspect for "toner" past the cleaner on the photoreceptor. The photoreceptor is free of "toner" past the cleaner.

Y N

0001

Enter dC26-385 and press Start button. Copier will make 50 copies and then stop automatically. (If patches are very light, increase the value at dC31-202 by 10, and operate dC26-385 again.) Look at the charge control patches. Patches are free of defects (light streaks or dark streaks) in the center. Press P button.

Y N

(Patches have streaks in center.) Interchange charge dicorotron and transfer dicorotron. Operate dC26-385 again. Patches are free of defects (light streaks or dark streaks) in the center. Press P button.

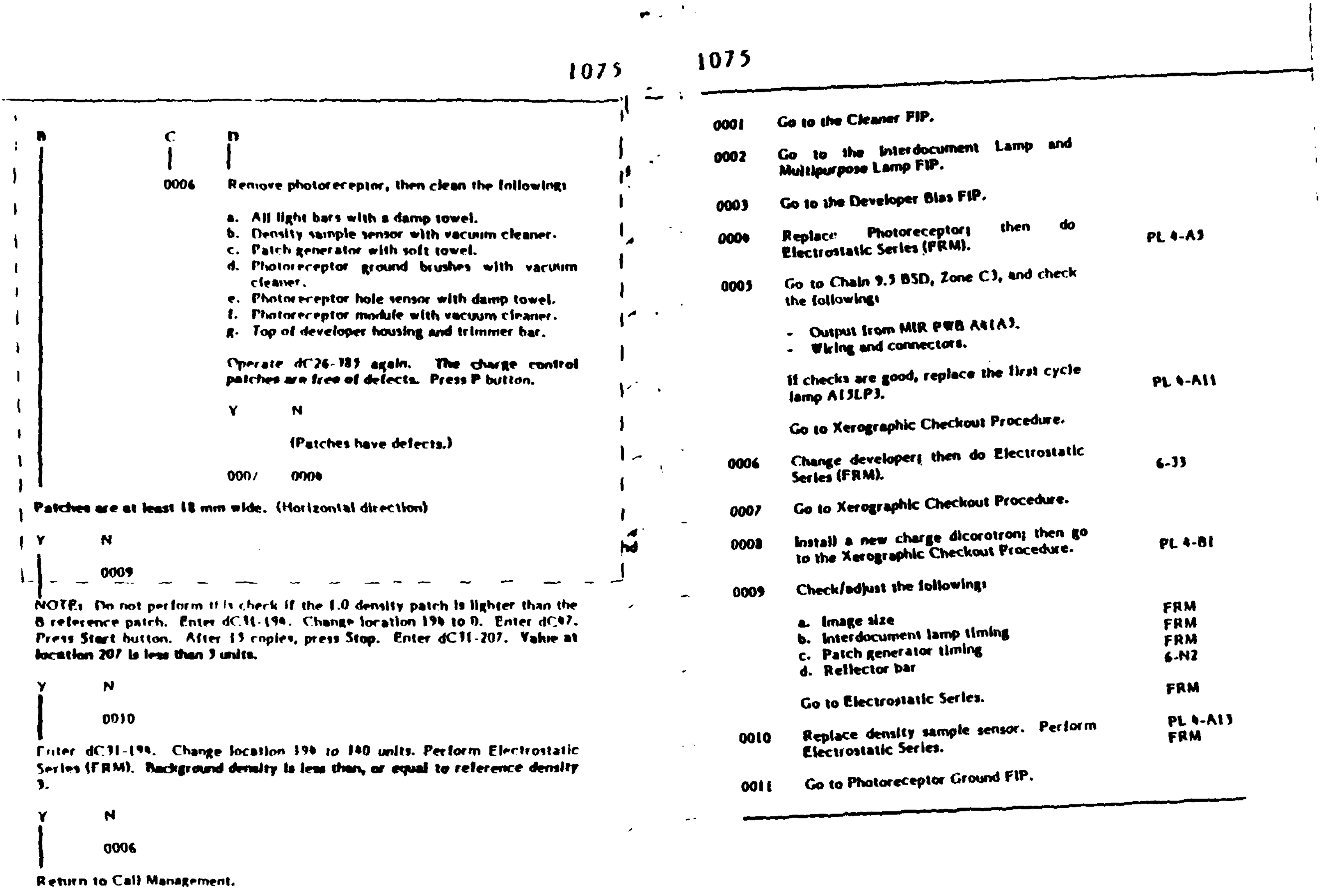
Y N

0008 (Patches have streaks.) Pull developer housing out. Rotate rolls two revolutions counterclockwise. Inspect for obstructions in trimmer bar area. There are obstructions.

Y N

B C D

Figure 1a A typical Fault Isolation Procedure, continued in Figure 1b. Note the simple decision tree structure and lack of rationale for any action. Letters are links to other branches of the tree; numbers refer to "dropouts" at the end of the FIP. Dropouts point to other FIPs or to repair actions. Figures 3 and 4 show this FIP from other perspectives.
Section enclosed in dashed line is that shown in Figure 4.



1075

B C D

0006 Remove photoreceptor, then clean the following:

a. All light bars with a damp towel.
b. Density sample sensor with vacuum cleaner.
c. Patch generator with soft towel.
d. Photoreceptor ground brushes with vacuum cleaner.
e. Photoreceptor hole sensor with damp towel.
f. Photoreceptor module with vacuum cleaner.
g. Top of developer housing and trimmer bar.

Operate dC26-385 again. The charge control patches are free of defects. Press P button.

Y N

(Patches have defects.)

0007 0008

Patches are at least 18 mm wide. (Horizontal direction)

Y N

0009

NOTE: Do not perform this check if the 1.0 density patch is lighter than the B reference patch. Enter dC31-190. Change location 190 to 0. Enter dC87. Press Start button. After 15 copies, press Stop. Enter dC31-207. Value at location 207 is less than 3 units.

Y N

0010

Enter dC31-190. Change location 190 to 140 units. Perform Electrostatic Series (FRM). Background density is less than, or equal to reference density 3.

Y N

0006

Return to Call Management.

1075

0001	Go to the Cleaner FIP.	
0002	Go to the Interdocument Lamp and Multipurpose Lamp FIP.	
0003	Go to the Developer Bias FIP.	
0004	Replace Photoreceptor; then do Electrostatic Series (FRM).	PL 4-A3
0005	Go to Chain 9.5 BSD, Zone C3, and check the following: - Output from MIR PWB A4(A3). - Wiring and connectors.	
	If checks are good, replace the first cycle lamp A15LP3.	PL 4-A11
	Go to Xerographic Checkout Procedure.	
0006	Change developer; then do Electrostatic Series (FRM).	6-33
0007	Go to Xerographic Checkout Procedure.	
0008	Install a new charge dicorotron; then go to the Xerographic Checkout Procedure.	PL 4-B1
0009	Check/adjust the following:	
	a. Image size	FRM
	b. Interdocument lamp timing	FRM
	c. Patch generator timing	FRM
	d. Reflector bar	6-N2
	Go to Electrostatic Series.	FRM
0010	Replace density sample sensor. Perform Electrostatic Series.	PL 4-A13 FRM
0011	Go to Photoreceptor Ground FIP.	

Figure 1b. Continuation of FIP from Figure 1a. The bottom of the left column, Return to Call Management, represents the failure of this procedure. The numbered items on the right are the dropouts which either indicate repair procedures or point to other FIPs. Section enclosed in dashed line is that included in Figure 4.

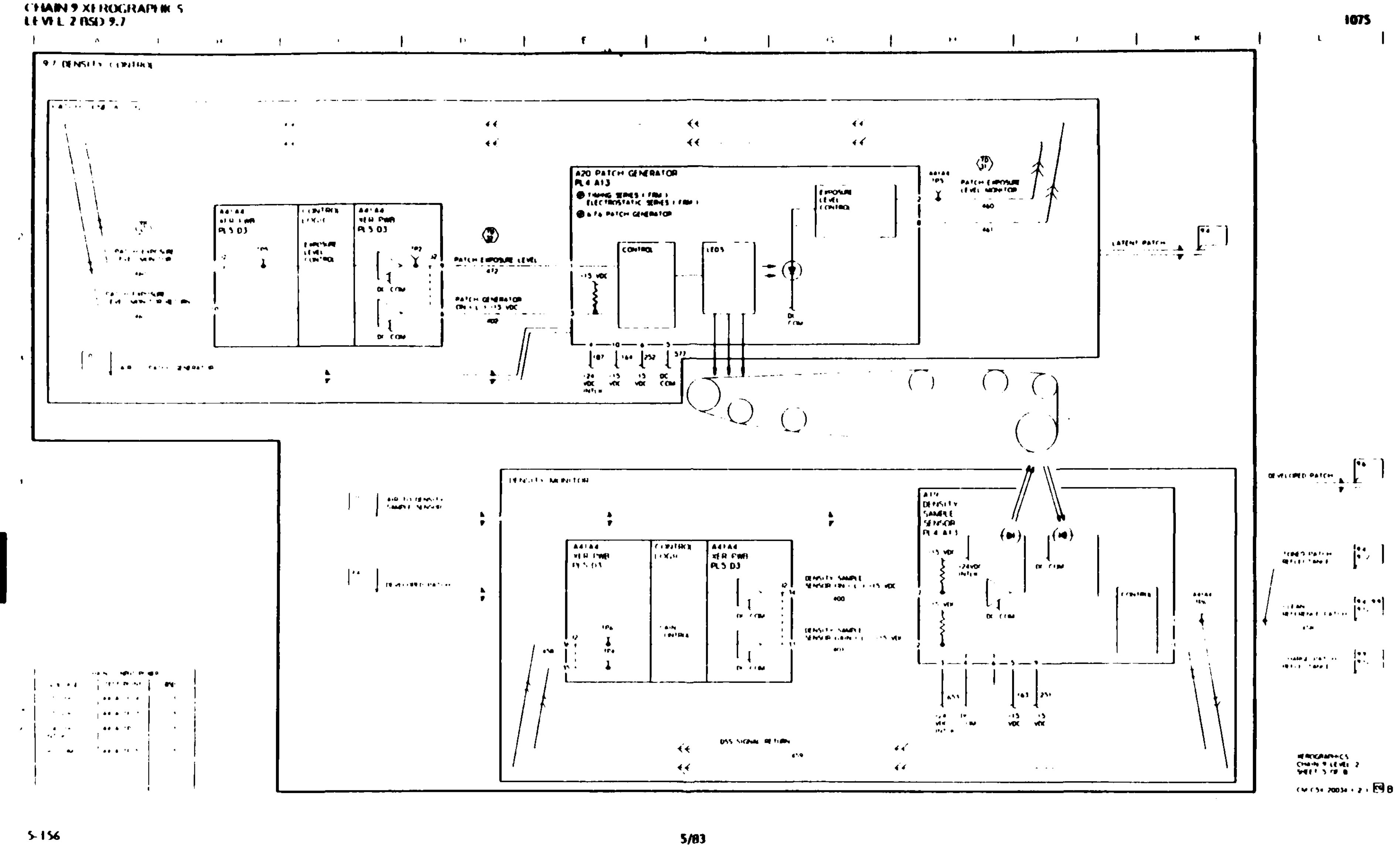


Figure 2 The Block Schematic Diagram [BSD] for the Density Control Circuit. Note that the information is principally interconnections, signal labels, and sometimes signal levels. Mechanical movement of photoreceptor and compressed air is also shown, but virtually no information about the electronic circuitry is presented.

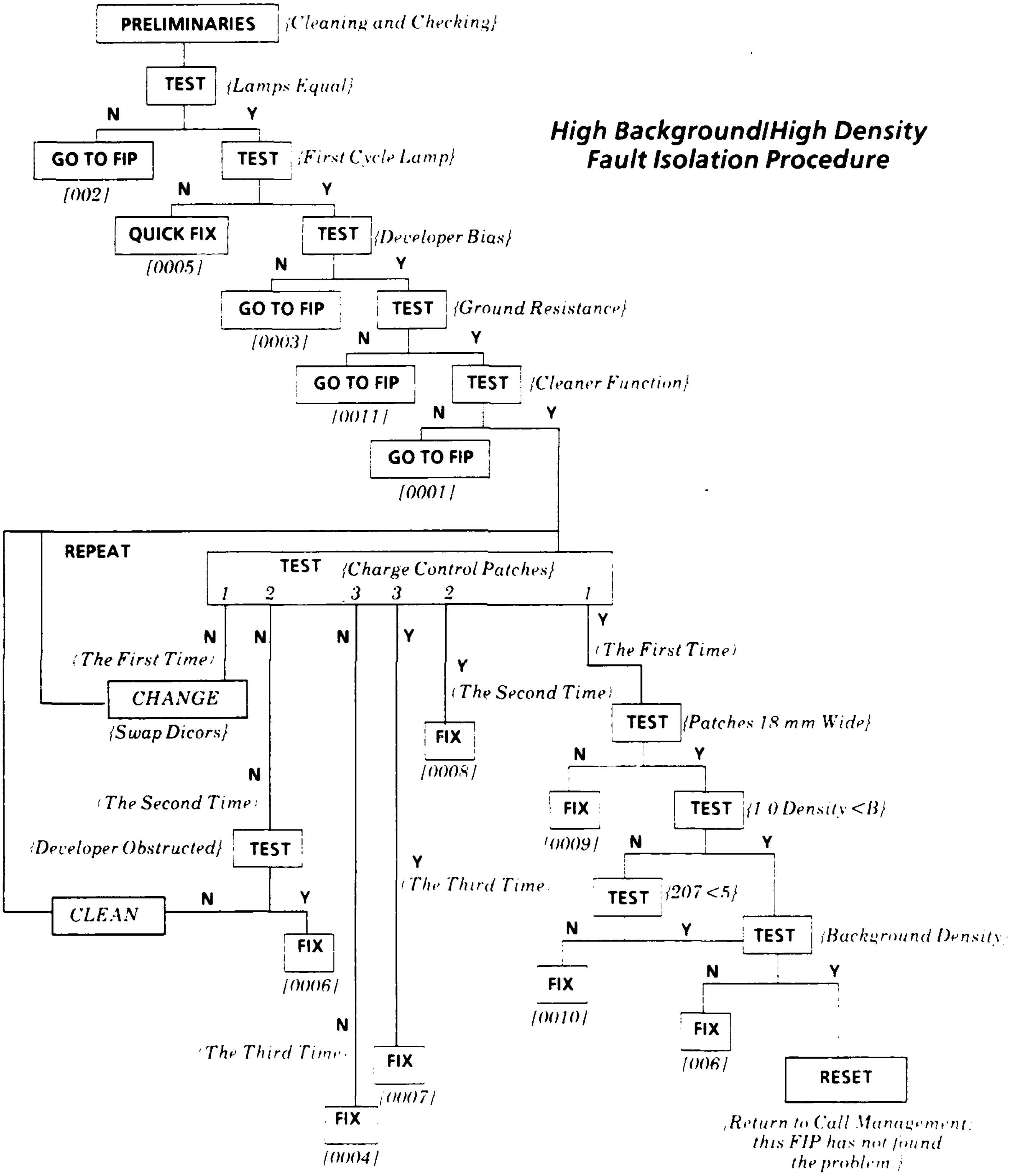


Figure 3 This is a diagram of the Fault Isolation Procedure shown in Figures 1a, 1b, and 4. Notice that the first half of the FIP is primarily branching conditions to direct the technician to other FIPs. The second half has one principal test which may be repeated up to three times for different failures. If the problem is not found, the technician is directed back to the top level of the documentation. The principal test of the second half, {Charge Control Patches}, and its dependents appear in Figure 4.

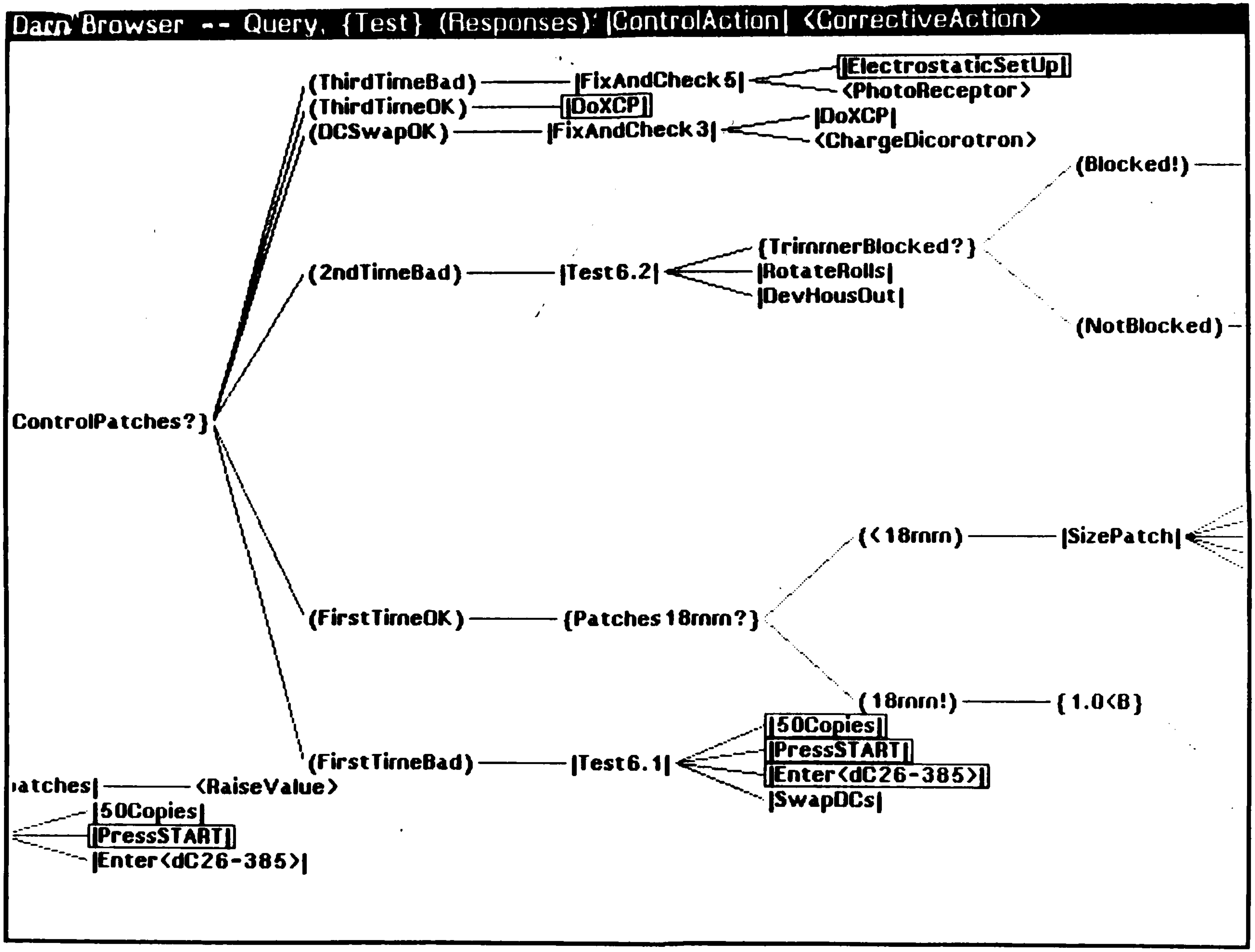
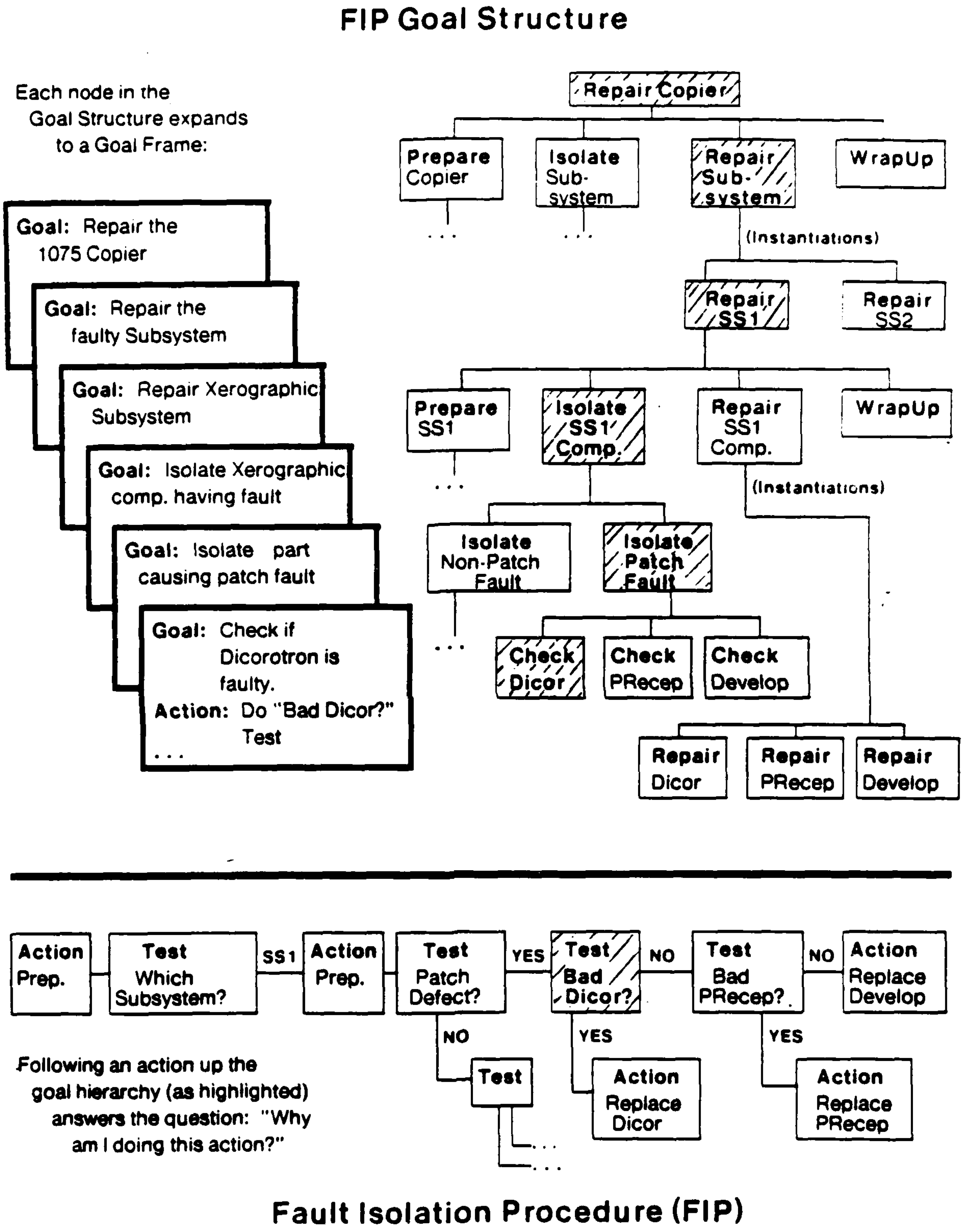


Figure 4. This DARN browser shows part of the representation in DARN of the FIP also seen in Figures 1a, 1b, and 3. The node at the left, ControlPatches?, is a test which may be repeated three times. The results of the test and consequent actions in diagnosis or repair are the branches from this node.


FIP Goal Structure

Each node in the Goal Structure expands to a Goal Frame:

Goal: Repair the 1075 Copier

Goal: Repair the faulty Subsystem

Goal: Repair Xerographic Subsystem

Goal: Isolate Xerographic comp. having fault

Goal: Isolate part causing patch fault

Goal: Check if Dicorotron is faulty.
Action: Do "Bad Dicor?" Test
. . .

Following an action up the goal hierarchy (as highlighted) answers the question: "Why am I doing this action?"

Fault Isolation Procedure (FIP)

Figure 5a. The FIP Goal Structure shows the goals addressed by the various steps of the FIP. The lower half of the figure shows the FIP seen in Figures 1a, 1b, and 3, represented as in Figure 3. The shaded blocks in the tree in the upper half represent the hierarchy of goals being met by the shaded block "Test Bad Dicor?" in the lower portion. Further detail is shown in Fig. 5b.

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

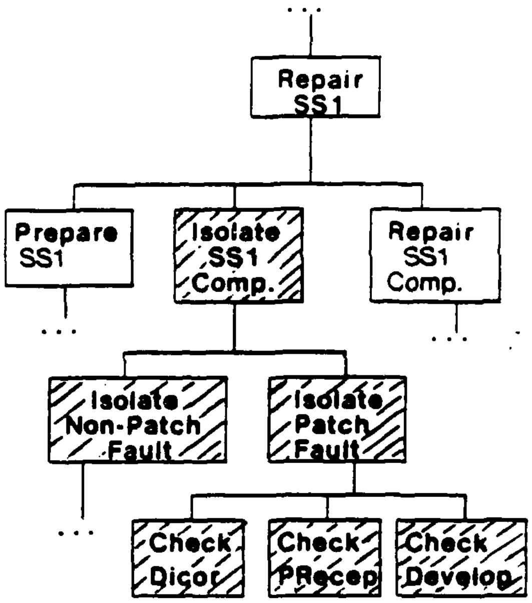
The PostConditions constrain the order in which Goals are performed:

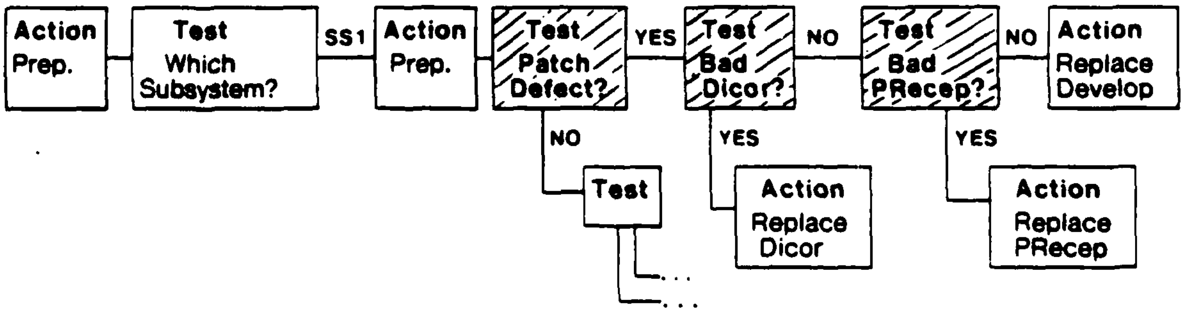
Goal: Check if dicorotron is dirty
PreConditions: 1. Test copies done 2. Lens/platen glass clean
PostConditions: Dicorotron is/is not at fault

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.

FIP Goal Structure





Fault Isolation Procedure (FIP)

Figure 5b This figure shows in shaded blocks the various tests which combine to meet the goal of isolating a faulty xerographic component In addition, the upper half shows the pre-conditions and post-conditions constraining each test, while the lower half shows how the same tests appear in the standard FIP structure

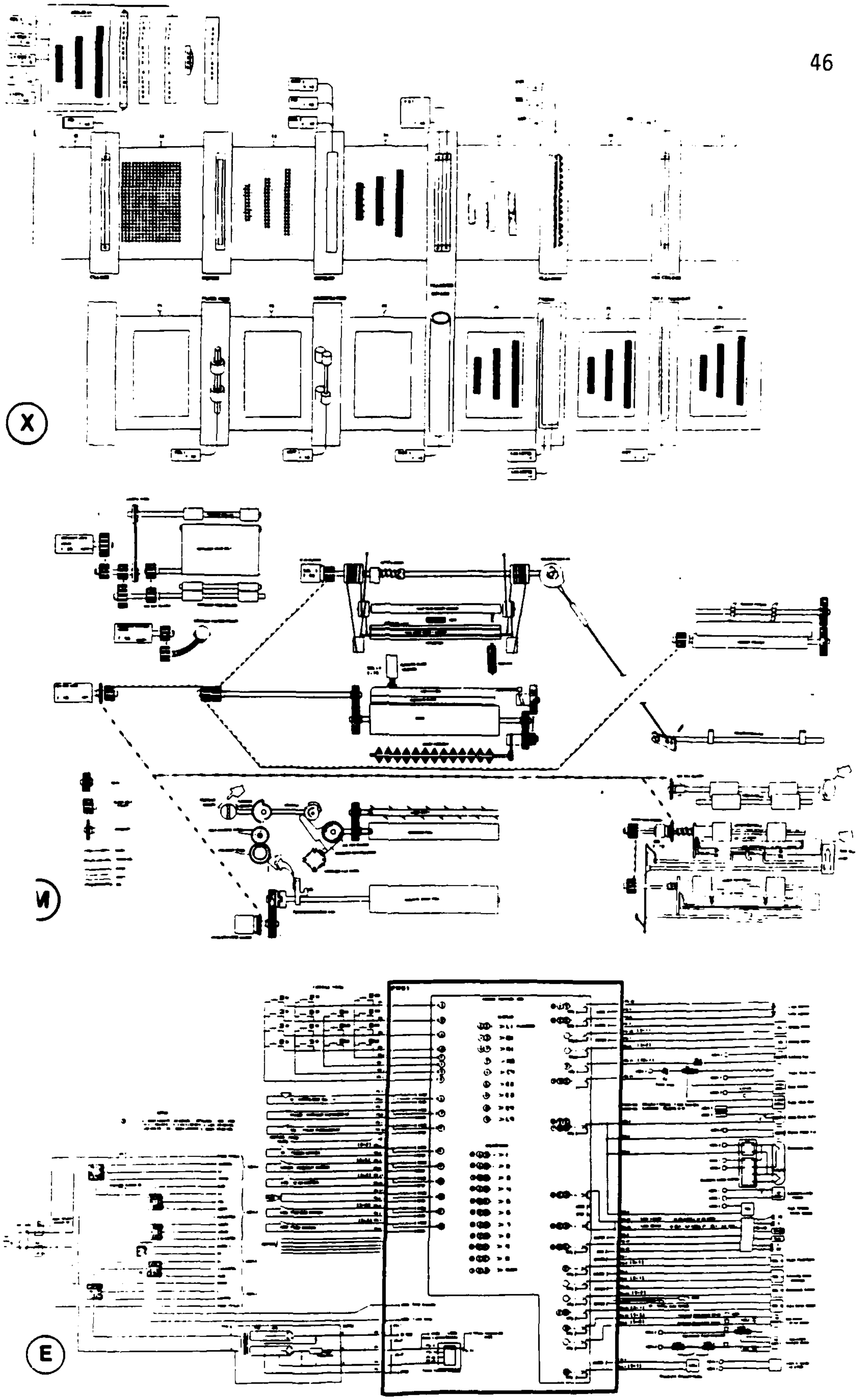


Figure 6a. This figure is a much reduced version of the first page of the Simplified 3300 Documentation. The three drawings depict the entire machine, divided into the xerographic and paper feed system, the mechanical system, and the electrical system. Reference numbers point to lists of known failures associated either with specific components or with specific steps in the process. Component labels include both the part numbers and references to their appearance in either or both of the other drawings.

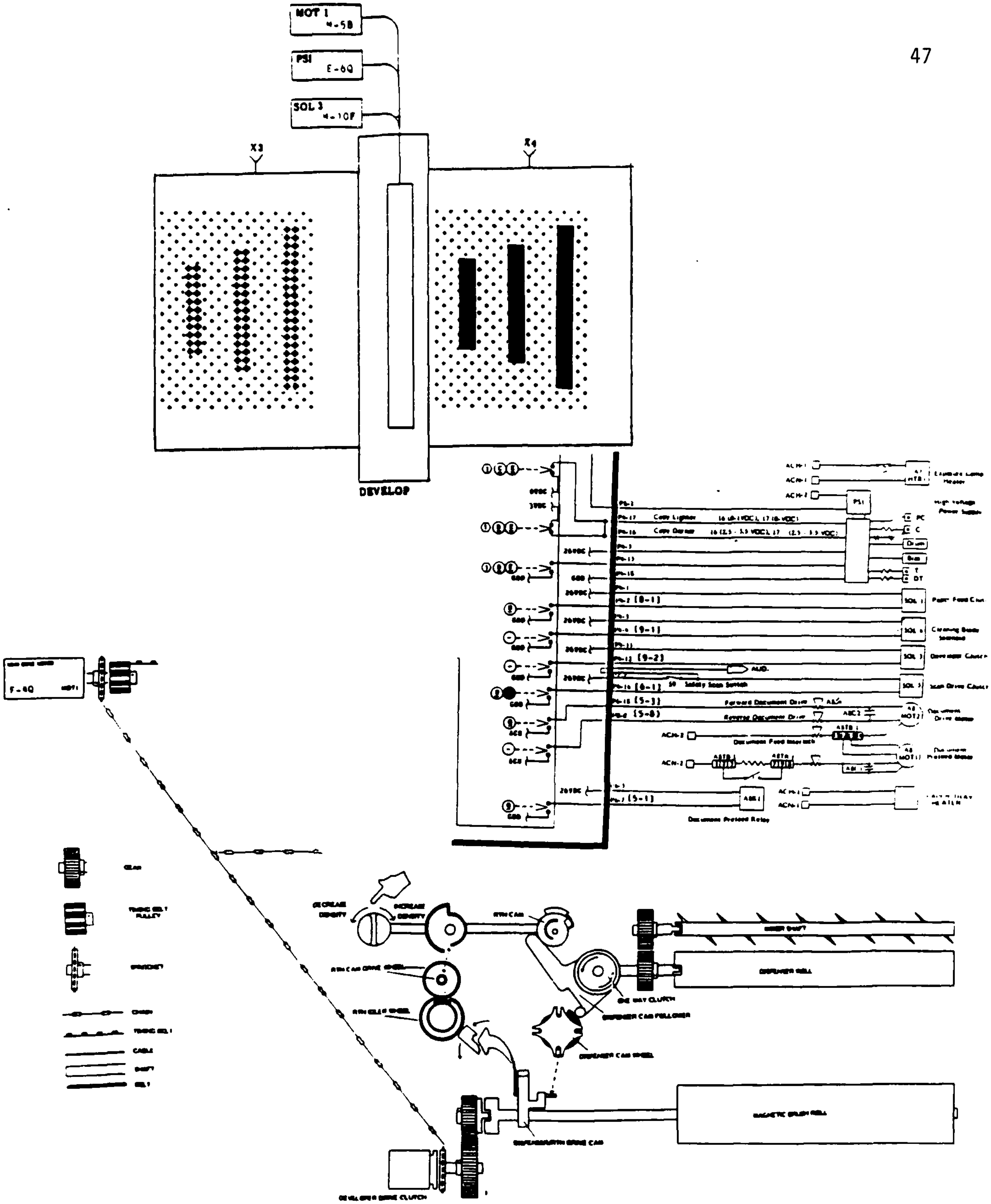


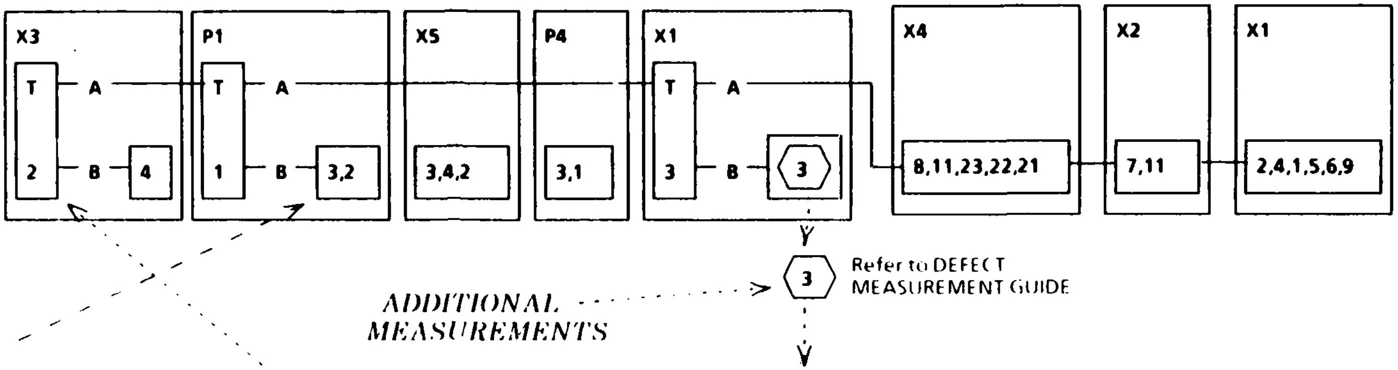
Figure 6b This illustration shows development as shown in all three drawings. In the first, it appears as a process step. X3 and X4 are pointers to problem lists on the other sheet. The tags MOT1, PS1, and SOL3 refer to the Main Drive Motor, the High Voltage Power Supply, and the Developer Drive Clutch, respectively, while the coordinates point to their locations on the mechanical or electrical drawings. Similarly, the depiction of the Main Drive Motor on the mechanical sheet includes a reference to the electrical sheet for the power to the motor.

One starts with the appearance of a copy quality defect on a copy of a test pattern:

UNIVERSAL TEST PATTERN (SIDE A)

DEFECTS **ADDITIONAL CHARACTERISTICS** **FLOW** *{A recommended but not mandatory sequence for diagnosis.}*

DELETIONS *[none]*



PROBLEM LIST

PAPER PATH

P1 -- PAPER TRAY
2. Paper - excessively damp
3. Paper - damaged / defective

P4 -- TRANSFER / DETACK
1. Prefuser Transport Latch misadjusted
3. Prefuser Transport not connected to ground

XEROGRAPHICS

X1 -- DRUM
1. Drum - Light shocked
2. Drum - contaminated
4. Drum - crystallized
5. Drum - defective
6. Drum - damaged
9. Drum - not centered on end bells

X2 -- CHARGE
7. Charge Corotron - not seated properly
11. Charge Cord - switched with Pre-Charge Cord

X3 -- EXPOSURE
4. Light leaks

X4 -- DEVELOPMENT
8. Developer Housing - missing drive
11. Developer - low amount
21. Developer Module - out of adjustment
22. Magnetic Roll - worn smooth or damaged
23. Brush height - uneven

X5 -- TRANSFER / DETACK
2. Transfer Corotron - not grounded
3. Transfer Corotron - obstructed by foreign material
4. Transfer Corotron - dirty

DEFECT MEASUREMENT GUIDE

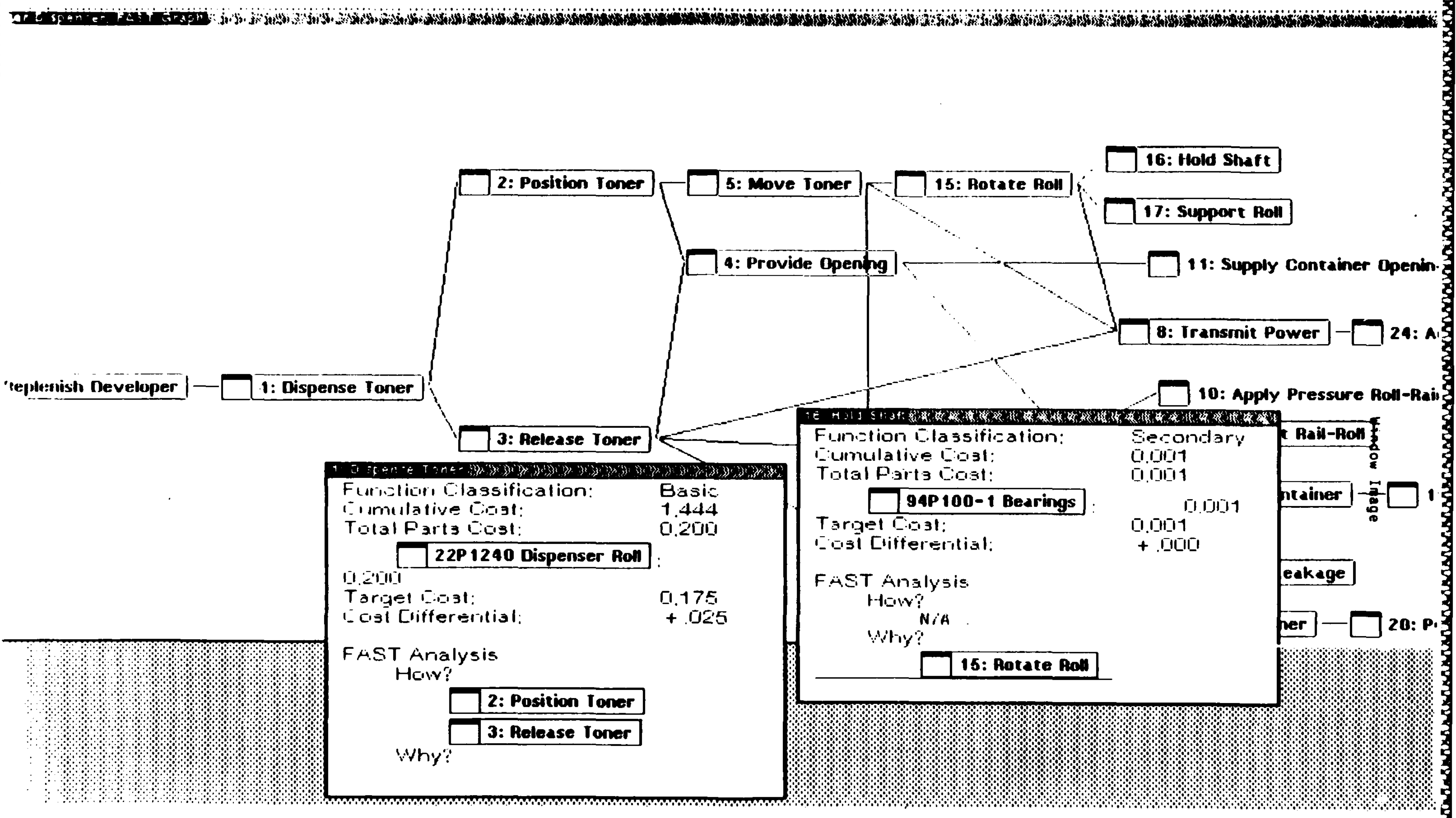
DELETIONS

DEFECT DIMENSION	REFERENCE	PROBLEM NUMBER
1 3/4 INCHES	BETWEEN SUCCESSIVE COPIES	X1 6,5,2,4,1,9
1/4 INCH OR LESS	DEFECT REPEATS	X2 4,8
SAME PLACE	EACH COPY	P1 3,2 X2 6,7,14 X4 13

DIAGNOSTIC TESTS

TEST NUMBER	TESTS	RESULTS
1	Make one UNIVERSAL (SIDE A) copy. Create a jam by restricting the paper feed. Examine the image on the drum. Is the image on the drum similar to the defect on the copy?	A. YES B. NO
2	Disconnect P/J 12. Select COPY LIGHTER. Make one copy. Is the defect still present?	A. YES B. NO
3	Make five copies. Lay them side by side and examine them. Do the defects a ppear in a specific pattern?	A. NO B. YES

Figure 6c. The diagnostic guide of the simplified documentation begins with defects on copies of test patterns. This leads to a recommended series of tests, the results of which point either to possible problems, listed in order of probability, or to additional tests or measurements.



1: Dispense Toner

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?

16: Hold Shaft

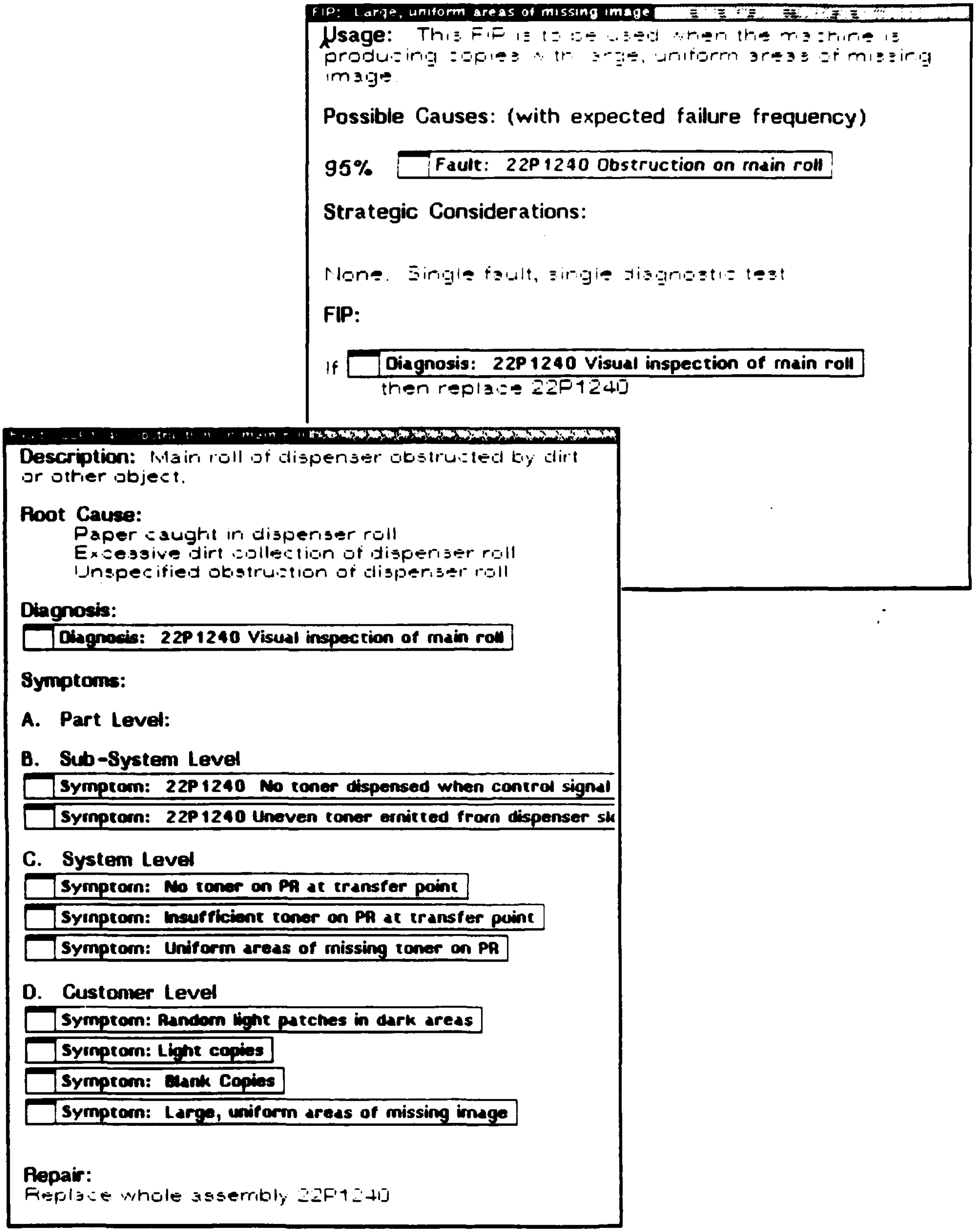
Function Classification: Secondary
Cumulative Cost: 0.001
Total Parts Cost: 0.001

□ 94P100-1 Bearings : 0.001

Target Cost: 0.001
Cost Differential: + .000

FAST Analysis
How?
N/A
Why?
□ 15: Rotate Roll

Figure 7a This portion of a FAST graph shows the decomposition of the function "Replenish Developer"; the subordinate cards show the parts used for a specific sub-function, information about those parts, and information about the relationship of that sub function to antecedent or subsequent sub functions.



FIP: Large, uniform areas of missing image

Usage: This FIP is to be used when the machine is producing copies with large, uniform areas of missing image.

Possible Causes: (with expected failure frequency)

95% □ **Fault: 22P1240 Obstruction on main roll**

Strategic Considerations:

None. Single fault, single diagnostic test

FIP:

If □ **Diagnosis: 22P1240 Visual inspection of main roll**
then replace 22P1240

Fault: 22P1240 Obstruction on main roll

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**

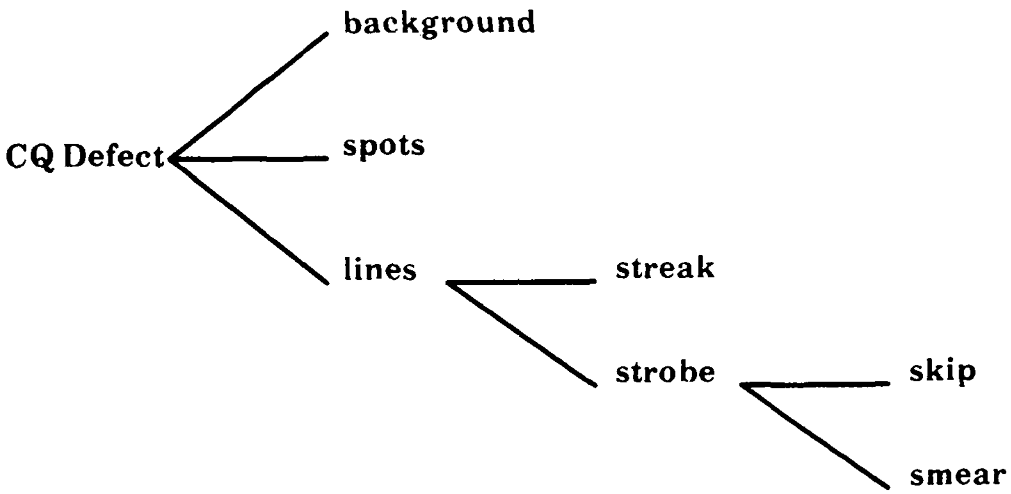
□ **Symptom: Large, uniform areas of missing image**

Repair:
Replace whole assembly 22P1240

Figure 7b. The Fault card represents a traditional FMEA analysis. The FIP card shows how that information might be used in the production of diagnostic documentation.

Figure. 8. The ESF (Evidence-Symptoms-Faults) model of troubleshooting shows how a technician moves from a problem to a fix. The technician progresses by characterizing the initial Evidence as a better-defined set of Symptoms. From these, she generates a set of possible Faults, which she isolates to the real Fault and fixes.

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



SMEAR

Cause: difference in speed between drum and paper

Characteristics:

- perpendicular to paper travel
- can differ in width across paper
- can stretch image
- will not show on PR image
- test pattern

STREAK

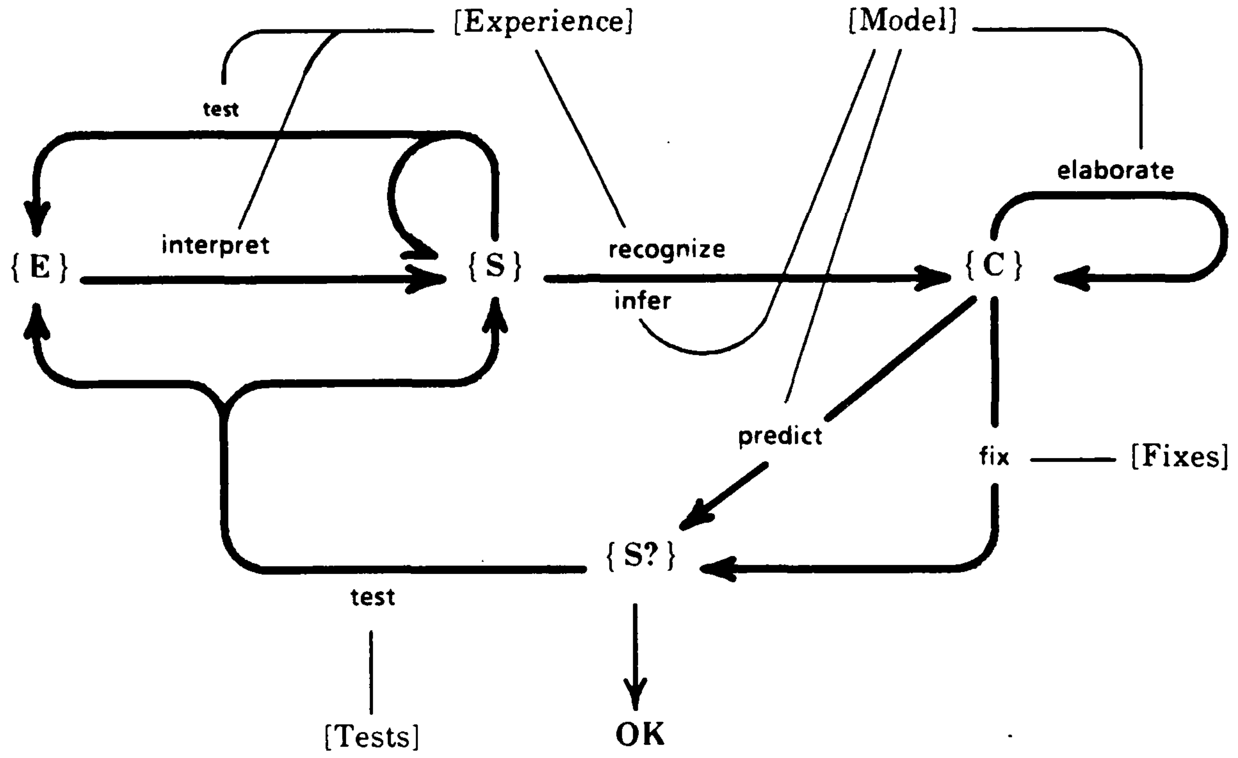
Cause: a fixed point along travel path

Characteristics:

- parallel to paper travel
- may show on PR image
- constant width
- test pattern

Figure 9. A taxonomy of Copy Quality defects, showing how differences in linear defects reveal different causes and point the troubleshooter in different directions.

Figure 10. The ESC (Evidence-Symptoms-Causes) model of troubleshooting elaborates the ESF model, seen in Figure 8, by acknowledging both the cyclic nature of the diagnostic process and the contributions of mental models, experience, tests, and attempted fixes to the process.



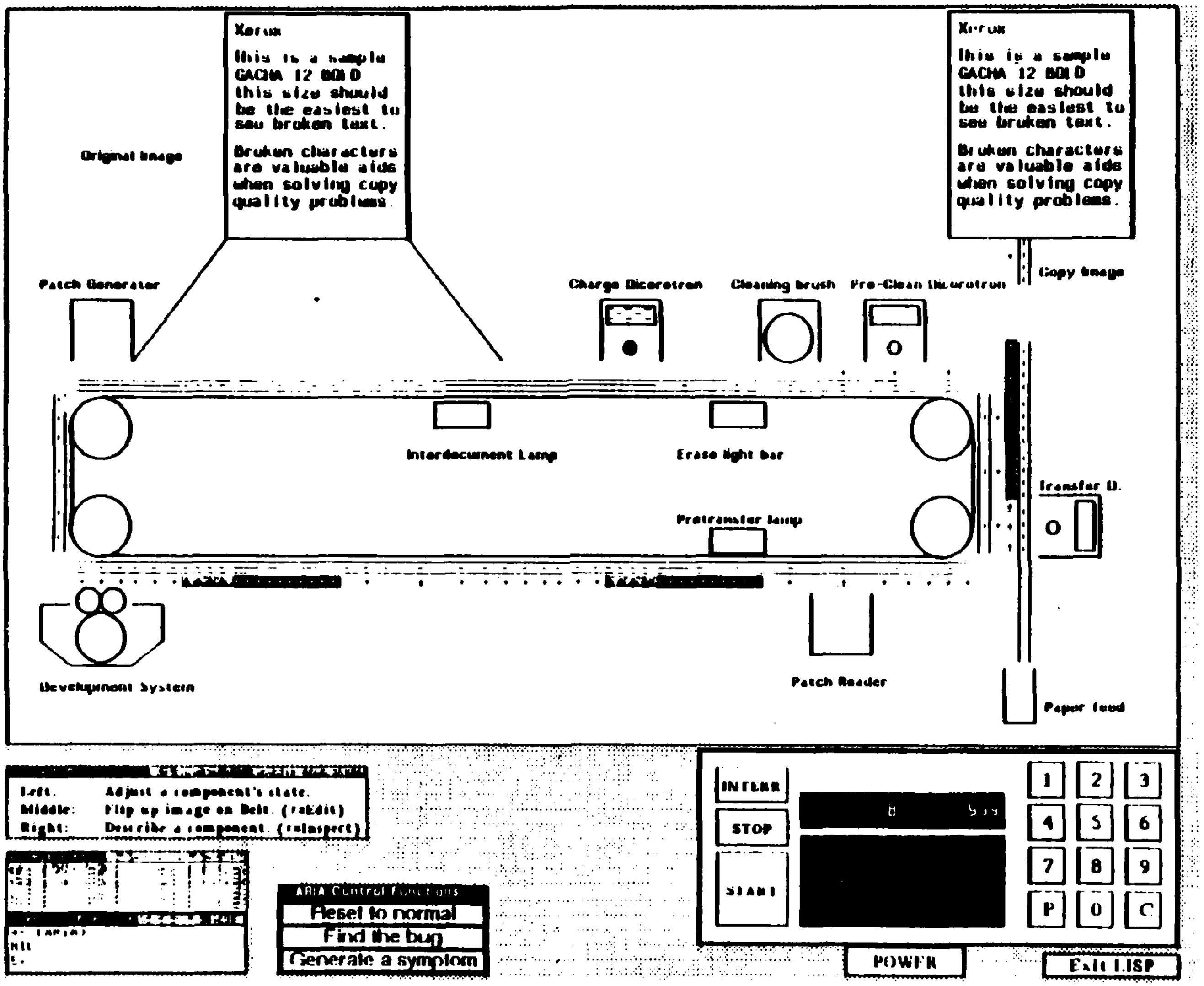
Protocols as Evidence for the ESC Model

1. Interpreting Error Log Evidence.	
T1: "The hard part about this problem is we don't know exactly what the symptom is. If it was a random H017 all the time, that would give us an idea of where to start. But, since it's a random H017, 18, 19, or 23, what's in common?"	{E} to {S} *The desire here is to interpret the evidence to reveal a symptom, which equates to movement from {E} to {S} in the model.*
2. Elaborating an Intermediate Cause	*The 5 volts may be the cause.*
T1: "But what would . . . sitting right here, and blup? drop out. I mean, doing nothing and just drop out. What about the 5-volt line would do that? . . . "	{S} to {C} *The question is how would such a cause produce these symptoms.*
T2: "Bad connector? Bad pin connection? Not so much a short as just a bad connection some place."	{C} to {C} *The question is how could the 5 volts cause the problem.*
3. Generating and Testing Predicted Symptoms	
T1: "It goes click, click, click. It puts 20 or 30 hits . . . "	{E} to {S} *Noise is evidence; error log entries are symptoms.*

T2: "That tells me I've got a power problem and because of the distribution of that power you'll get a multitude of hits clear across the board."	{S} to {C}
	The symptoms indicate a cause which suggests other symptoms.
	{C} to {S?}

T1: "Right. That's what's happening. And, it's random."	{S?} to {S} *The suggested symptoms do exist.*

Figure 11. These protocol segments show movement through portions of the ESC model of troubleshooting shown in Figure 10.



Xerox

This is a sample GACHA 12 BOLD this size should be the easiest to see broken text.

Broken characters are valuable aids when solving copy quality problems.

Xerox

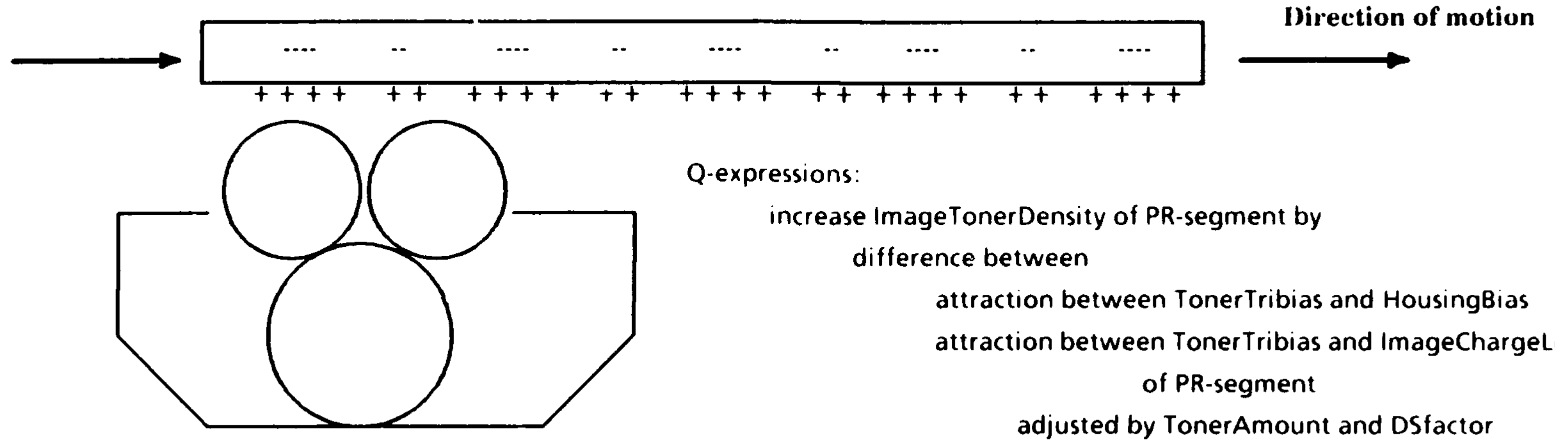
This is a sample GACHA 12 BOLD this size should be the easiest to see broken text.

Broken characters are valuable aids when solving copy quality problems.

Left: Adjust a component's state.
Middle: Flip up image on Belt. (rabEdit)
Right: Describe a component. (rabInspect)

Figure 12. The ARIA simulation of the 1075 xerographic subsystems displays this screen, showing a schematic representation of the xerographic module, original and copy images, the 1075 control panel, and some controls for manipulating the ARIA model.

```
PR-segment object
    located next to DevelopmentSystem
    parameters:
        Image (BitmapName)
        ImageChargeLevel (ChargeLevel)
        ImageTonerDensity (TonerLevel)
        ImageFaults (FaultList)
    attachments: none
    Q-expressions:  none
```



```
DevelopmentSystem object
    parameters:
        HousingBias (ChargeLevel)
        TonerTribias (ChargeLevel)
        TonerAmount (Amount)
        DeveloperStatus (DeveloperFaults)
    attachments:
        spatial to ImageTonerDensity of PR-segment
        spatial to ImageFaults of PR-segment
```

```
Q-expressions:
    increase ImageTonerDensity of PR-segment by
        difference between
            attraction between TonerTribias and HousingBias
            attraction between TonerTribias and ImageChargeLevel
                    of PR-segment
                adjusted by TonerAmount and DSfactor
    add to ImageFaults of PR-segment:
        if DeveloperStatus = DevoBiasShorted
            then DarkVerticalStreak
        if DeveloperStatus = HousingShorted
            then HighBackground
```

Figure 13 ARIA simulates the development system of the 1075 with two objects, shown here both in their visual representation and in pseudo-code, and two Q expressions [which are actually part of the Development System object] which govern the interaction between the two objects when either of the spatial attachments of the stationary object are valid.

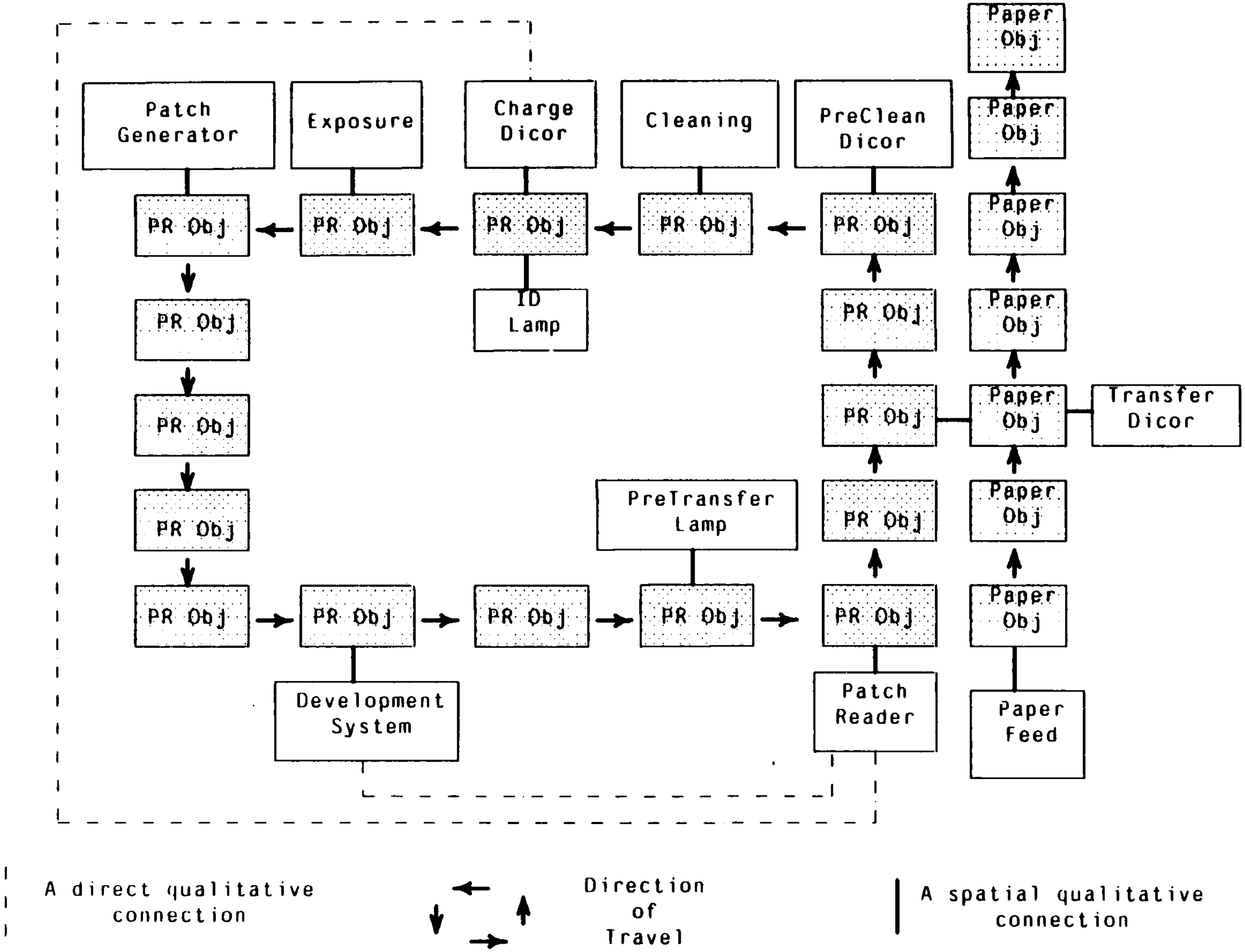


Figure 14 This illustrates the basic objects in the ARIA 1075 simulation and the different qualitative connections between them. Arrows showing direction of travel are provided for those objects which move

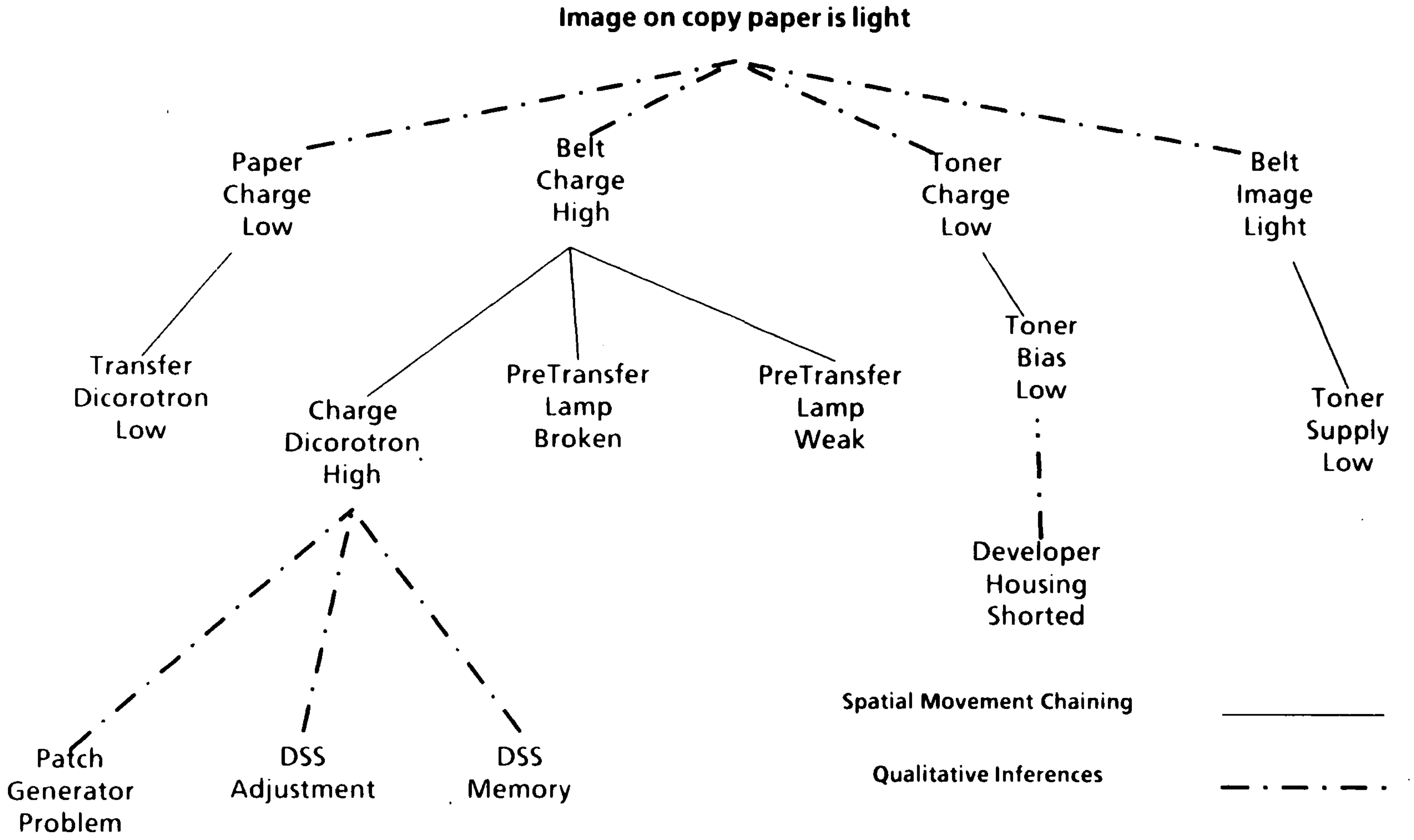


Figure 15 A causal net for a light copy problem. Links in the chain are differentiated between those requiring qualitative inferences and spatial chaining based on physical proximity, either in the machine or in the ARIA simulator

III Research in Modeling, Reasoning, and Expertise:

Four Papers from Research into the Semantics of Procedures

Thomas P. Moran
Arthur Farley
Daniel S. Jordan
Gregor Kiczales
Julian E. Orr
Daniel M. Russell
Marikka Rypa
Jeff Shrager
Daniel Weld

Xerox Palo Alto Research Center
Intelligent Systems Laboratory

November 1986

This work was partially sponsored by a research contract MDA903-83-C-0189 from the Army Research Institute to Xerox.

1.0 Introduction

This report documents the second year of work on the "Semantics of Procedures" project. During this period we continued to focus on the teaching of diagnostic reasoning skills for use in troubleshooting. We investigated the representations of qualitative models of complex mechanical systems and the use of such qualitative models to support diagnostic reasoning about those systems. We also conducted field investigations into present expertise to help set goals for training technicians.

2.0 Papers

2.1 Qualitative Modelling

The first paper in this collection, "Pragmatic Issues in Qualitative Modelling," is a discussion of our attempts to design and build a qualitative simulation model of xerography for use in an intelligent instructional environment with the aim of effectively training technicians in copier repair. We evaluated the xerographic simulation model, called ARIA, with respect to five criteria: behavioral correctness, explanatory accessibilty, usability, efficiency and flexibilty. Overall, ARIA proved successful in simulating the desired xerographic behaviors for our purposes. However, its shortcomings include limited represention of spatial relations that result in decreased model flexibility and range of simulated abnormal copier behaviors. A redesign of ARIA, called DUET, proposes a different qualitative model structure that overcomes this and other shortcomings of ARIA.

2.2 Automatic Description of Complex Mechanical Devices

The paper entitled "Issues in the Automatic Description of Complex Mechanical Devices" focuses on the role of automated diagnostic reasoning in the generation of explanations about the behavior of complex mechanical devices such as xerographic copiers. It is established that complex devices require multiple models for efficient diagnostic reasoning, and problems with switching between models and sharing information across models are examined. Three views of a hypothetical copier are presented in increasing detail: the first model describes the copier functionally, the second describes the copier in terms of component processes, and the third uses a qualitative physics of three dimensional shape and constrained motion to describe a single component of the copier. Interactions between these models and their implications for automated reasoning about the physical system are discussed.

2.3 Diagnostic Modeling

The third paper, "Diagnostic Mechanism Modeling," proposes another method for constructing a model of complex mechanical and electronic systems that is designed to support diagnostic reasoning. This qualitative model is used to support the creation of causal explanations of abnormal machine behavior. Two principles of diagnostic reasoning are investigated: a "normality principle", where model variables assume values relative to normal levels, and a "single product principle," where model variable values are propagated to produce a single description of the fault.

2.4 Social Aspects of Diagnostic Expertise

The fourth paper in this collection, "Talking About Machines: Social Aspects of Expertise," is a report on an ethnographic investigation of Xerox's field service force. The purpose was to characterize the expertise of experienced technicians and to try to determine how this was acquired in order to develop training methods conducive to this expertise. The conclusion presented in this paper suggests that in this context expertise is primarily an integrative ability, an ability to order numerous facts, results of tests, and chains of events into a coherent understanding of the state of the machine and how it came to be. The process of integration appears to be primarily verbal, a fragmented narration of the testing, which relates what has been done, what the results were, and what this might mean about the state of the machine. This narration often includes anecdotes of personal or second-hand *experience*, used as references, sources of inspiration, or validation of an interpretation. The integration of the understanding occurs with reference to a mental model that has been constructed from fairly weak theoretical understanding, personal experience, and the experience of the community as represented in anecdotes. The conclusion is that the anecdote is vital to the ability of the community to cope with unanticipated service problems, and that the situated nature of the anecdote, the encapsulation of context with machine information, makes it an ideal medium for circulating information about machines among people of varying levels of expertise and concomitant variation in their models of the machine.

3.0 Summary

The first three papers are a contribution to the study of the development of intelligent instructional environments for teaching effective diagnostic reasoning skills. Together, these papers explore various issues and techniques in the design of automated diagnostic reasoning systems to be used in instructional environments, and the qualitative modelling that underlies these systems. The fourth is intended to ground these efforts by providing an understanding of the real expertise that is the goal of technical education.

Pragmatic Issues in Qualitative Modelling: Lessons Learned from Modelling Xerography

Jeff Shrager, Daniel Jordan, Thomas P. Moran, Daniel M. Russell, and Gregor Kiczales

Intelligent Systems Laboratory
Xerox Palo Alto Research Center

Working Paper - April 1986

The project was funded, in part, by the Army Research Institute under contract number MDA903-83-C-0189.

Abstract

Qualitative reasoning has proved useful in a variety of AI systems for dealing with the design, diagnosis, and explanation of physical devices. This paper discusses the design of a training simulator, based on a qualitative model, for the xerographic subsystem of the Xerox 1075 copier. The process of xerography is very complex, and our modelling efforts reveal a number of important issues.

We describe our xerographic simulator, called ARIA, and assess it with respect to various modelling goals: its ability to correctly model both abnormal and normal behaviors of the copier, its utility for the automatic construction of explanations, its useability by students and teachers, its flexibility to simulate other copiers, and its computational efficiency. Analysis of ARIA reveals several representational limitations. We describe how these problems can be circumvented by sketching the design of a new simulator called DUET. Some of the features of DUET for overcoming ARIA's limitations include: more modular representations of devices and physical processes, quantitative parameter values with qualitative overlays, spatially-distributed arrays of parameter values, hierarchic structure that allows simulation at different levels of detail, and processes that run at different time scales to model effects at different simulation speeds.

The various technical points in this paper shed light on where research is needed in qualitative modelling, and our experiences should be of use to others with similar endeavors.

Introduction

Xerographic copiers are among the most complex devices in common use today. Numerous people depend upon their correct operation and a vast support network has grown up within Xerox, and other companies, to maintain these products. The goal of our research is to teach a conceptual understanding of the workings of xerographic copiers which will enable service personnel to understand, remember, and, when necessary, modify and create repair procedures for effective servicing of copiers. A part of this research has led to the development of a computational simulation of the primary components of the xerographic copier. The present paper discusses this simulation and its subsequent redesign in order to elucidate the problems that arise in modelling a very complex physical system.

We begin with an outline of the project goals to motivate the use of computer simulation. In particular, we have chosen to use qualitative simulation technology (de Kleer & Brown, 1984, Forbus, 1984) in order to derive explanations of the behavior of the system for the student, and in order to permit various types of reasoning over the model. Next we describe the Xerox 1075 copier's xerographics machinery from which most of the examples in the paper will be dervied. We will try to support the assertion that xerographics is a strikingly complex physical process. The next section describes ARIA -- an instantiation of a qualitative model for xerographics. ARIA is an implemented and running system. ARIA is then critiqued on several grounds, and a redesign of the model, which we refer to as DUET, is proposed.

We feel that there is much to be learned by an analysis of this project as the use of qualitative modelling in a real complex domain with actual training goals has rarely been attempted although there exist quantitative quantitative models (Brown, J.S., Burton, R.R., & de Kleer, J., 1982; Hollan, J.D., Hutchins, E.L., & Weitzman, L., 1984). The various technical points in this paper will shed light on where research is needed in qualitative modelling, and may enable others in similar positions to avoid some of the problems that we have encountered.

Goals of Modelling

The overall goal of the present project is to develop a training technology that will enable Xerox service personnel to repair copiers more efficiently. We assume (for this paper) that if we can teach the service person to reason causally about the xerographics subsystem of the copier, then he or she will be in a better position to repair it. Unfortunately, the technology of xerography makes it very difficult to provide hands-on experience for the student as a means of achieving this goal. A major problem is that the xerography process is light-sensitive. Furthermore, many of the phenomena which lead to problems take place at the level of electrical fields and other microscopic effects that are impossible to see. Many of these phenomena are also time critical -- one cannot stop the machine and start it again where it left off. Some of the problems also interact with states in the copier's computerized "copy-quality monitoring system". These facts make it difficult to use a real xerographics system as a training model.

A computerized simulation circumvents these difficulties. In particular, a simulation can make light-sensitive processes visible for student observation and exploration. It can enable the student to stop the action, scale up microscopic phenomena, and make microprocessor state observable. A further attraction is that a simulation features explicit representation of machine state and activity that can be used by a tutoring system to reason about state and behavior for purposes such as explanation. These features, among others, provide a basis for a powerful and, hopefully, intelligent instructional environment. Of course, a simulation is not without instructional shortcomings. Being an approximation to the real device it cannot provide completely accurate device behavior, making it necessary to carefully chose the set of desired behaviors and to live within those boundaries. A simulation also does not model a number of real-world features, such as kinesthetic feedback. Nonetheless, the advantages of computerized simulation are sufficiently attractive to have led us to undertake the project.

These pedagogical motivations led to two primary goals of the modelling project. These goals directed our efforts and are those by which we assess the adequacy of the resulting simulators:

> **Behavioral Correctness**: The model should simulate selected normal and abnormal behaviors of the device over its possible parameterizations. That is, the model must exhibit the "correct" behavior for all possible settings of its parameters, regardless of whether or not the behavior is among the selected normal and abnormal behaviors.
>
> It is important to note the distinction between "normal" and "abnormal" device behavior and "correct" and "incorrect" simulator behavior. The terms "normal" and "abnormal" describe copier behaviors. A good photocopy is called "normal" and a photocopy that is too dark, for instance, is called "abnormal". This is distinct from "correct" and "incorrect" performance of the simulator. If a particular fault in the real copier would cause the real copy to come out too dark, and the simulator also produces this result, then the simulator correctly models an abnormal behavior of the copier. If, however, a particular fault in the real copier would cause the real copy to come out too dark, and the simulator produces lighter simulated copies, then the simulator is incorrectly modelling the copier.
>
> **Explanatory Accessibility**: The simulator's internal representations should be usable by a reasoning system to construct causal explanations of the device's normal and abnormal operations.

Three less important goals played a role in the progress of this project. They are (in decreasing order of importance):

> **Useability**: The simulation should be usable by students and teachers for understanding the causal underpinnings of xerography. That is, if the simulation is not an instructional explanation device per se

(and in general they are not), then it should provide the means for interfacing to a such a mode. Furthermore, the instructional designer should be able to change the model by providing an explanation, approximately in his or her own terms.

Efficiency: The model should run at a reasonable speed on commonly available equipment.

Flexibility: The system should permit instructional designers to change the underlying model in accordance with new copier designs, with minimal effort required to maintain *behavior correctness.*

These goals strongly suggest the use of qualitative modelling technologies (over, e.g., numerical modelling technologies). Qualitative simulation and reasoning has proved useful in a variety of AI technologies that are closely related to these goals: design of devices (Williams, 1985), causal explanation (de Kleer, 1984), fault diagnosis (de Kleer & Williams, 1986), etc. (See Bobrow & Hayes, 1984 for various other papers.) In fact, the purpose of the original qualitative simulation work was spurred by educational goals very similar to our own (Brown, J.S., Burton, R.R., & de Kleer, J., 1982). The ability to construct reasoning devices such as those requested by the *explanatory accessibility* goal depends crucially on the simulation's representation of the device -- that is, its access to symbolic terms that are appropriate for meeting the explanatory goals. Qualitative modelling provides appropriate device representation for constructing such a reasoner (de Kleer, 1984). Furthermore, qualitative modelling features great flexibility in the level of detail at which a device and its components can be represented, allowing us to reason about the device at a granularity that is appropriate for the construction of explanations for students; this partly satisfies the *useability* goal. Some combination of the *useability* and *flexibility* goals suggests a qualitative programming language that can be used by instructional designers to change around the model without their having to know about either its internals or its implementation language.

For these reasons, constructing a qualitative simulation of the copier's xerographics subsystem seemed a logical choice. However, the area of qualitative modelling still does contain much unchartered terrority, and we anticipated that our application might lead us into such areas. In particular, the theory of qualitative modelling is well worked out only in simple cases, and a device as complicated as a xerographics subsystem has not been tackled before. Topological dynamics, which are important in devices such as a copier, are neglected in most qualitative technologies (but see Forbus, 1984 and Weld, 1985). We will return to this point a number of times in this paper and explain how our simulator and its underlying qualitative model deals with these limitations.

In the next section we describe the heart of a copier: the steps of xerography. We also discuss a number of copier problems that arise, their impact on the quality of copies, and the consequent impact upon the difficulty of our project. We will see that xerography is not a simple process. The implementation and performance of our model, ARIA, and its subsequent redesign, referred to as DUET, raises interesting issues about the application of qualitative modelling technology to real systems of great complexity.

The Steps of Xerography.

How does xerography work? Before diving into this dicussion it is useful to point out some potentially comfusing terminology. We will use the term "component" to refer to a particular interchangable part of the device, as generally referred to by the service personel and copier engineers. Components recursively decompose into other components. A "step" will be used to refer to a logical phase of the entire operation of producing a copy. Steps also decompose, into smaller steps, in the obvious way. Steps are specific to the operation (copying) under investigation. "Processes" are like steps, but are not purpose specific. Several components and processes take part in each step and the edge between steps is not crisply defined. The term "chain" is used to refer to a number of components which implement steps to carry out some complete operation. All of these terms are abstractions for explanation purposes. We will see in this section how these terms are instantiated in the model of the xerographics chain of a particular photocopier.

For the copying process, plain paper does not have the right properties to effect direct image transfer so a complex intermediary mechanism is used. There are six steps involved in the copying operation. Although the instantiation of each step in components is machine specific, these steps are common to almost all xerographic copying. The present discussion refers to the Xerox 1075 copier.

Step 1: Charging: A light sensitive insulating "photoreceptor" is negatively charged by a component called a "dicorotron". The photoreceptor is the intermediary mechanism needed to effect image transfer to plain paper. The dicorotron has a high voltage AC wire passing through a grounded housing in front of a negatively charged plate. The wire creates positive and negative ions. The positive ones are attracted to the negative plate, and the negative ones are repelled toward the photoreceptor. Figure 1 is a schematic of a dicorotron. These devices are used in three of the steps.

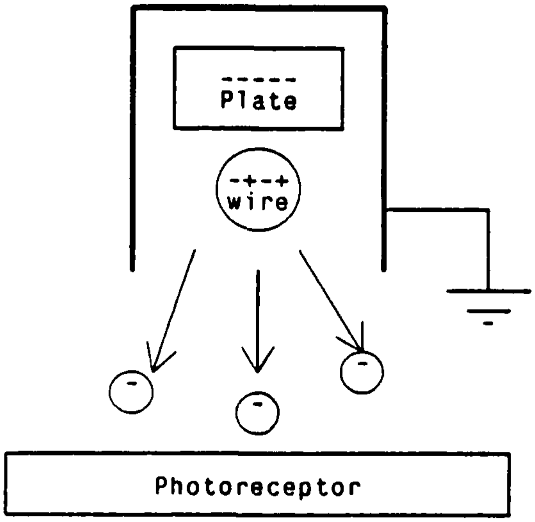


Figure 1: The Dicorotron.

Step 2: Imaging and Exposure: The charged photoreceptor is exposed to a projection of the image of the original onto the photoreceptor's uniform field of negative charge. The photoreceptor is made of a light senseitive material such that areas that are illuminated become electrically conductive. Since the projection has light only where there is white in the original, the illuminated part of the negative field of charge is grounded, leaving a pattern of high negative charge where there are dark lines on the incoming image, and low or neutral charge where there is white. Thus a latent, charged image of the original is "cookie-cut" onto the photoreceptor.

Step 3: Development: Particles of toner (microscopic balls of black plastic) are brushed onto the photoreceptor. These have a slightly positive charge as a result of their own friction. They are attracted to the places where the photoreceptor is highly charged, moving from the wheels to the surface of the photoreceptor. This translates the charge into a visible toner image. Figure 2 depicts the development system. Note that the developer wheels, which brush toner onto the photoreceptor, are also negatively charged; this helps lift the toner out of the well onto the photoreceptor. There is a delicate balance of charges between the wheels and photoreceptor so that just the right amount of toner is transferred to the charged parts of the photoreceptor and kept off of the neutral or lesser charged parts.

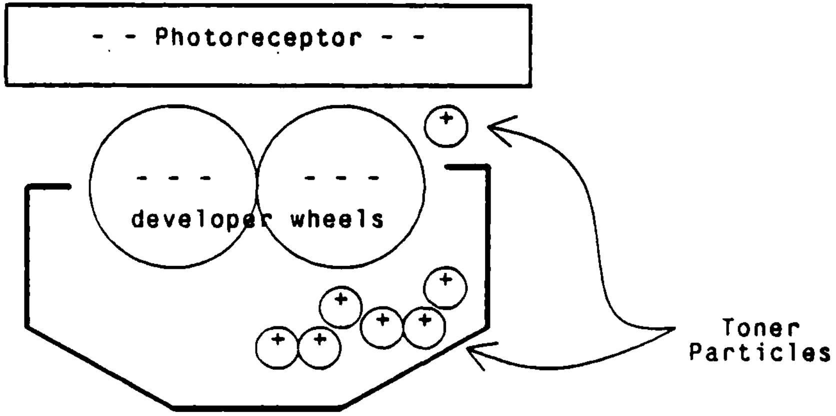


Figure 2: The development system

Step 4: Transfer: A piece of paper with a dicorotron behind it is pressed against the photoreceptor. The charge holding the toner to the photoreceptor is less now than the charge behind the paper so the toner particles move to the paper.

Step 5: Fusing: The paper is moved to a hot roller which melts the plastic toner particles into the page.

Step 6: Cleaning: The photoreceptor (and whatever toner is left on it) is charged positively by another dicorotron and then brought into contact with a negatively charged brush. Left over toner is attracted to the brush. This serves to clean the photoreceptor, which is then reused in step 1.

Figure 3 depicts the entire copying process, approximately as it is laid out in the Xerox 1075 copier. Recall that various components compose each step. (The 1075 photoreceptor is a belt which circulates in the machine. The belt has five distinct page areas on it. Therefore, five copies are pipelined through this process.)

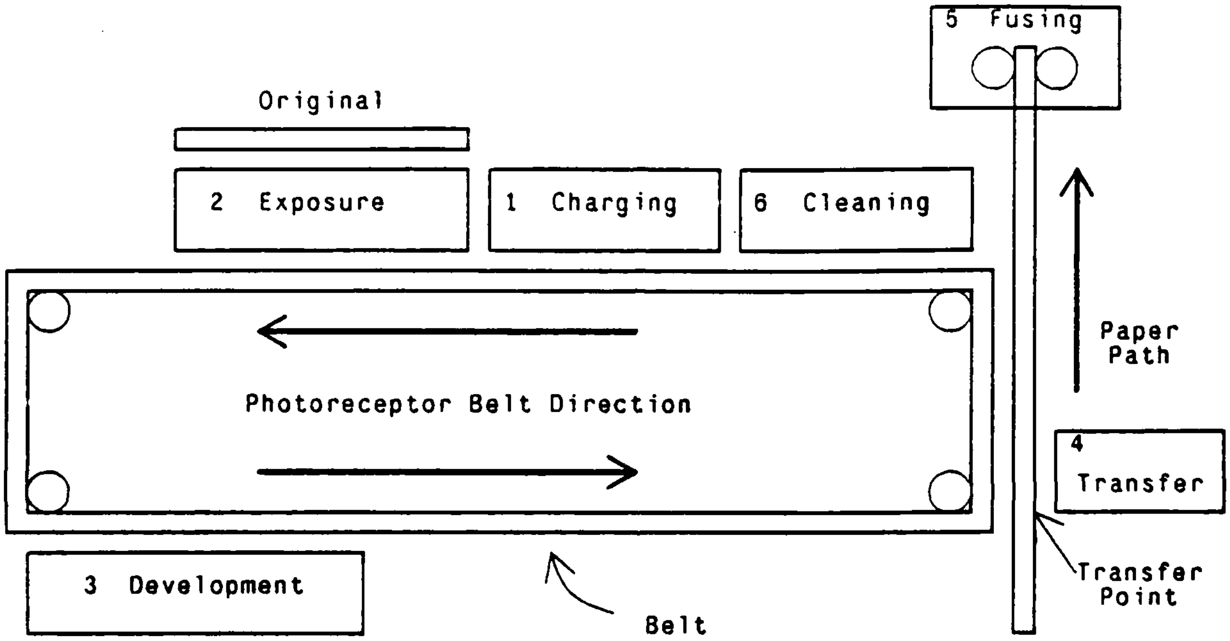


Figure 3: The Steps of Xerography (approximate side view).

What Can Go Wrong in 1075 Xerography.

The Xerox 1075 has a few peculiarities that add complexity to the above description. In the following descriptions we point out some of the peculiarities, leading to problems that often have subtle causes.

Cleaning and Development: Recall that the belt has a pattern of negative charge when it reaches the development step. Although there are high and low negative charges at that time, there is no part that is completely without charge, or that has a positive charge. Why isn't the background of all copies grey? It would seem reasonable that if you simply spread out the positively charged toner particles on a pattern of high and low negative charges, you should get grey and black, but not white. Recall that the developer wheels (which form the brushes -- see figure 2) are also negatively charged, higher than the lows on the belt, but lower than the highs on the belt. Thus, white (or at least very light) development takes place when the wheels attract the toner more than the belt, and dark takes place where the belt attracts more than the wheels. Unfortunately, if these charges come out of range, then the developer can do the wrong thing. In fact, developing and cleaning are done by almost precisely the same mechanism, except that the charges are different. Thus, if the developer wheels are charged more negatively than the high negative charge on the belt, the developer will *clean toner off the belt instead of putting it on.*

Another obscure bug that can take place in this step is caused by loading the wrong toner into the machine. Toner (for other machines) has a negative charge instead of a positive charge. The 1075's development wheels cannot properly pick up this toner.

Detacking: Ambient charge on the paper, belt, and toner are removed by yet another dicorotron (which, in this case, exudes both positive and negative ions) just after the transfer point, and just before the belt bends to go under the cleaner. This is referred to as "detacking" (a station not shown in figure 3). If the paper detack process fails, the page may not be stripped off of the belt as it rounds the bend to be cleaned, thus dragging the page into the cleaning mechanism. This is a common cause of copier jams. Differences in environmental humidity can change the capacitance of the paper, leading to this type of problem.

Microprocessor quality monitoring: In addition to the copy areas, the belt has between-copy regions, one of which is a "patch" area. Each time around (once every five copies) this patch area is exposed and developed as if there were a piece of grey paper being copied there. However, instead of being transferred to paper, the exposed and developed patch is read by a photocell which determines how well the belt is attracting toner. Every ten patch reads (= every 50 copies) a microprocessor in the copier determines how well the copier is doing its various jobs, as evidenced by the jobs done on the patch. The toner level in the developer and various charging levels can be adjusted by the microprocessor if it determines that some changes have to be made. (If this isn't already complicated enough, the patch sequence that is put down is actually more complicated than just 10 grey trials. It is designed to reveal not only copy quality problems, but also cleaning problems.) This self adjusting mechanism also permits the copier to set its own initial parameters both when it first warms up, and whenever the service person makes adjustments. However, the mechanism can also get in the way of repairs. For example, service persons who are not used to the self-adjustment of the 1075 will try to adjust charging levels manually in order to correct various problems. The microprocessor has an unfortunate habit of changing things back without notice, thus making this sort of fix unworkable.

Summary.

Copiers have a very simple criterion of correct operation: they make good copies. About 90% of copy quality problems are due to dirt in the system, or an electrical imbalance that can be repaired by cleaning the machine and running it through its automatic electrostatic setup procedure. Problems due to massive mechanical, electrical or optical defects resulting in blank copy can be tracked down by exhaustive search of the few possible broken components. Some knowledge of causal ordering in the device, and some clever detective work may save time in tracking down these bugs. Various geometric problems in the copy such as streaks and smears can be broken down into two classes: horizontal streaks which are generally due to a permanent problem in one place, with the belt moving under it, and vertical streaks which usually indicate some synchronous problem, e.g., a roller which has been left too long in one position that becomes flattened on that side. Here some reasoning about the various physical structures of the machine may be useful. It is the obscure problems -- the other 10% -- which take up most of the time in copier repair and to which we hope to provide insight by constructing the ARIA model.

ARIA

Our first attempt to model the 1075 xerographics subsystem resulted in a system called ARIA. We intended ARIA to be a stand-alone copier simulation with nominal graphics and interface capabilities so that a student would be able to use it with the help of an instructor even before the tutorial components were built onto it. More importantly, ARIA was the test bed for qualitative modelling of xerographics, so we tried to do something with each of the parts of the subsystem discussed above.

A set of six copy quality problems were chosen as the target copier misbehaviors from those covered in the service manual for the Xerox 1075. Thus, we had the service community's causal analysis for these particular problems. The simulation goal of ARIA was to model all of these problems by appropriate fault settings of the models of various devices, and to be able to simulate the correct behavior of the device when everything was appropriately set -- the *behavioral correctness* goal. We also tried to satisfy the *explanatory accessibility* goal by using an explicit modelling language. The use of a modelling language which makes the components in the system explicit also supports the *flexibility* goal, although this was not a part of our plans at the outset.

We initially ignored issues of *useability* and *efficiency*, although they were of concern at a later stage in the modelling effort, when it became clear that ARIA was actually going to be used as-is. It was assumed that these problems would be addressed in later work if the qualitative modelling was successful.

Ontology.

The ARIA qualitative model consists of a collection of *components*. These components implement the six steps, plus the photoreceptor belt, patch writer, reader, and micro-processor. Each component has associated with it qualitative variables, or *QVariables*, which describe its own internal (local) state, and qualitative expressions, or *QExpressions*, which use and change both local QVariables and QVariables in other components. A *neighboring component* is one that is connected to a given component by virtue of QExpressions in the one component referring to QVariables in the other.

The QVariables which constitute a component's state are each instances of a qualitative type, or *QType*. Each QType defines a total order of qualitative values, or *QValues*, along some dimension. Each QVariable has as its QValue some atom from the range given by the corresponding QType. For instance, the Photoreceptor Belt segments have a QVariable: BELT.CHARGE. This is a QVariable of QType CHARGE, which may take on the QValues HIGH.NEGATIVE, NEGATIVE, NEUTRAL, POSITIVE, or HIGH.POSITIVE. The particular names are irrelevant to the qualitative computations. Only their relative order matters.

Component Interaction.

The QExpressions are written in a basic qualitative calculus and use these QValues to produce the component behaviors. During an ARIA cycle, each component's QExpressions are evaluated, changing that component's QVariable QValues and/or QValues in its neighboring components. For example, the Charge Dicorotron component has a CORONODE.VOLTAGE QVariable with possible QValues: HIGH, MEDIUM, and LOW. This QVariable is an instance of the QType AC-AMPLITUDE. An example of an QExpression for the interaction between the Charge Dicorotron and Photoreceptor Belt is given in Figure 4.

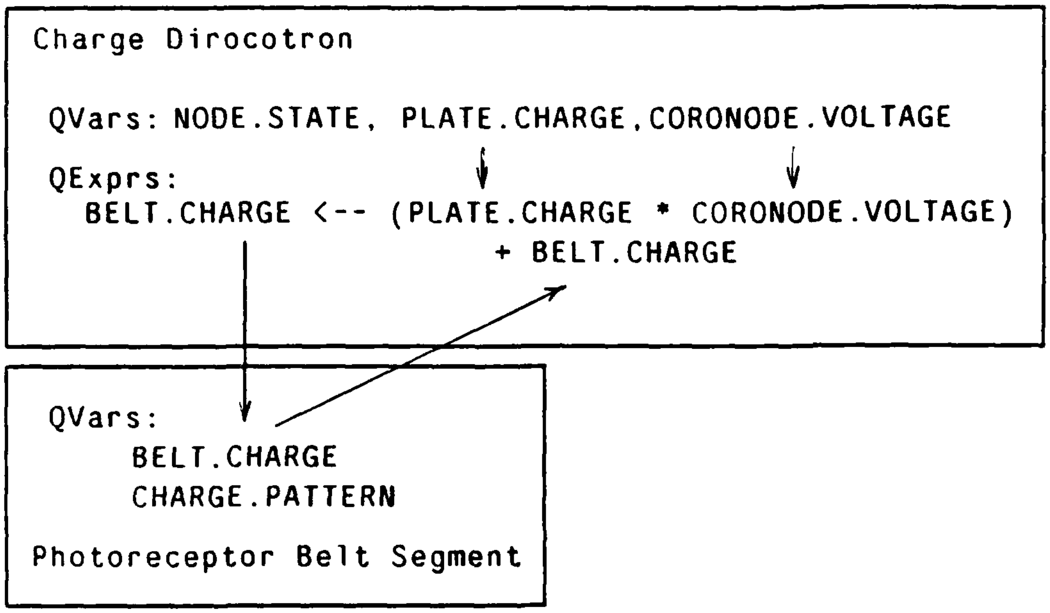

Charge Dirocotron
QVars: NODE.STATE, PLATE.CHARGE, CORONODE.VOLTAGE
QExprs:
BELT.CHARGE <-- (PLATE.CHARGE * CORONODE.VOLTAGE)
+ BELT.CHARGE
QVars:
BELT.CHARGE
CHARGE.PATTERN
Photoreceptor Belt Segment

Figure 4: Schematic of the Dicorotron/Belt interaction

In this example, the QVariable BELT.CHARGE is a variable in a segment of the Photoreceptor Belt component (as described above), which is a neighbor of the Charge Dicorotron. The qualitative normalized average of the CORONODE.VOLTAGE and the DICOROTRON.PLATE.CHARGE is computed according to the QValues' positions in their respective qualitative dimensions. The computations are actually implemented by special functions: QMOVE, which unplements "←" (assign) and "+" (add), and QCROSS, which implements "*" (multiply), for qualitative values.

Issues of Physical Space.

So far we have only described the QExpressions which take part in making a copy, and not, for example, in the process of paper transfer, which is where paper jams usually take place. However, other copy quality defects, such as streaks and smears on the copy, are generally due to physical damage or misadjustment rather than electrical, optical, or photoelectrical problems. There are two issues here: physical representation and movement.

There are machine defects that are spatial in nature, such as a location-specific crack in the corotron node (the high voltage source), as well as defects that also entail movement, such as a flattened out roller. These defects have to be represented in ARIA in the same language as other phenomena. For example, the corotron has a QVariable called NODE.STATE that has the possible QValues: SOLID and CRACKED. Such physical features are usually simply passed along in the form of another physical feature. For example, one of the QExpressions describing the charge dicorotron is:

```
CHARGE.PATTERN ← (QCASE NODE.STATE (SOLID 'SOLID)
                                        (CRACKED 'STREAKED))
```

A CRACKED dicorotron makes a STREAKED pattern on the belt which becomes a STREAKED pattern on the resulting copy. CHARGE.PATTERN is a QVariable associated with belt regions and holds QValues of SOLID or STREAKED. QCASE is a special version of a Lisp CASE function, and is understood by the reasoner. This does not interfere with explanation accessibility.

The only things which move in the ARIA paradigm are belt segments and paper segments. A *spatial region* representation supports association of fixed and movable components. Component QExpressions which change QVariables in moving neighbors (e.g., the charging QExpression changes the belt's charge) actually name a spatial region instead of a particular neighboring component. Spatial regions contain a pointer to the component that is currently occupying that region of space (Figure 5). References and assignments to QVariables in spatial regions are automatically redirected to whatever component happens to be in that region of space at the time. This indirection is hidden to both of the components involved.

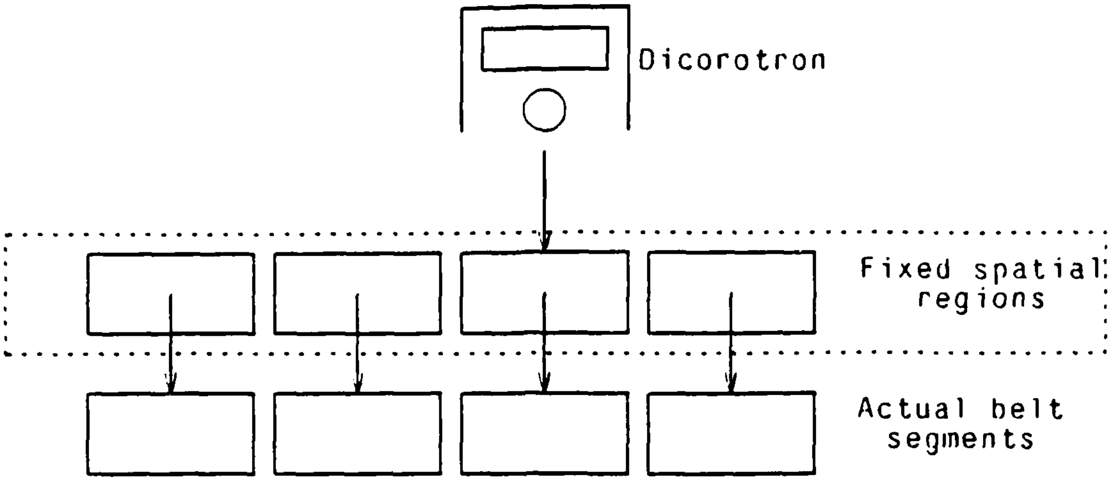


Figure 5: Topology of Spatial Indirection.
Contents of dashed box are hidden from the Dicorotron.

Any component is able to change the system by changing the bindings of spatial regions. This happens entirely by side-effect. Thus, for instance, there is a "motor" component which shifts around all of the belt and paper segments, when it is fired.

Control of the Model.

The ARIA model is normally driven by a stepwise global controller. This runs through each of the components in a prespecified order, including the belt component, which causes the belt to shift position. The belt must be either the first or last component invoked by the driver, or else it will move in the middle of a cycle. Invoking a component simply binds all of the relevant QVariables and then evaluates all of its QExpressions.

Assessment of ARIA

ARIA partly succeeded in satisfying the goals that we set out at the beginning of the project. In this section, we discuss its limitations and the reasons behind the failures. In the next section, we discuss various proposals to resolve these difficulties.

For the two primary modelling goals:

ARIA partly satifies the **Correctness** goal: It succeeds in simulating the normal operation of the Xerox 1075 copier and a number of faulty conditions, including some of those discussed above. It includes a representation of the contents of the microprocessor store and the actions of the patch writer and reader. Almost complete *behavioral correctness* was obtained in ARIA, but only by extensive trial-and-error tuning of parameter space divisions.

ARIA also partly satisfies the **Explanatory Accessibility** goal: We must distinguish the success of the system as a simulator from its success as a reasoning substrate. It proved difficult to construct a reasoner over the ARIA QExpressions even though ARIA is a successful simulator (the *behavioral correctness* goal). By following the paths of simulation we can obtain an explanation in reasonable qualitative terms of the behavior of the copier, and causal backchaining is possible with sufficient constraints and heuristics. However, various problems contributed to severe limitations in *explanatory accessibility*. We summarise them here and expand on some of them in the rest of this section:

Side effects: The movement of the belt segments by side effect of the belt (motor) component is not represented in a QExpression and is thus inaccessible to a reasoner.

Order reliance: The order of evaluation of the component QExpressions is important to analysis of the behavior of the system, but is not explicitly represented.

Historical ambiguity: The QExpressions are written as QValue-setters and hence lose the information of what the old value of a QVariable was. This makes it very difficult to use backward chaining.

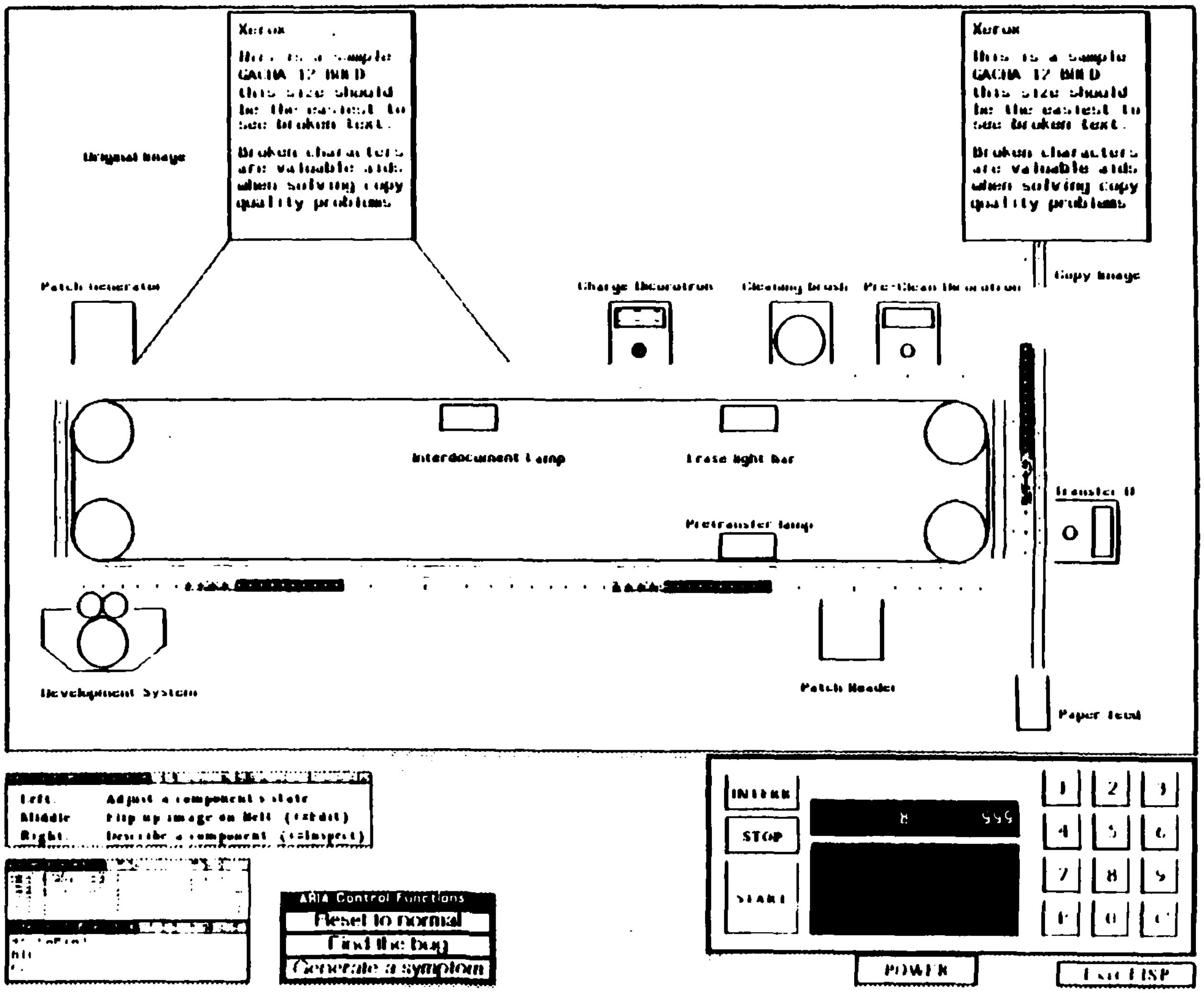
Evaluating ARIA's success at satisfying the three less important goals:

Useability: ARIA has an interactive graphical interface that supports changing the QValues and editing the QExpressions of all the components. Figure 6 is a snapshot from the screen of the actual ARIA model in operation. Figure 7 is a closer view in which the QVariable settings for the developer is opened. These menus can be used to set the QValues of the QVariables displayed.

Efficiency: ARIA runs quickly on a 4 megabyte Xerox 1132 and acceptably on a 3.5 megabyte Xerox 1109 . Once various caches have been filled one simulated copy takes about 10 real seconds. This is satisfactory for educational purposes.

ARIA fails in **Flexibility**: It proved difficult to change ARIA to represent another device, or even a slightly modified Xerox 1075.

Figure 6: The ARIA Interface



Xerox

This is a sample GACHA 12 BOLD this size should be the easiest to see broken text.

Broken characters are valuable aids when solving copy quality problems.

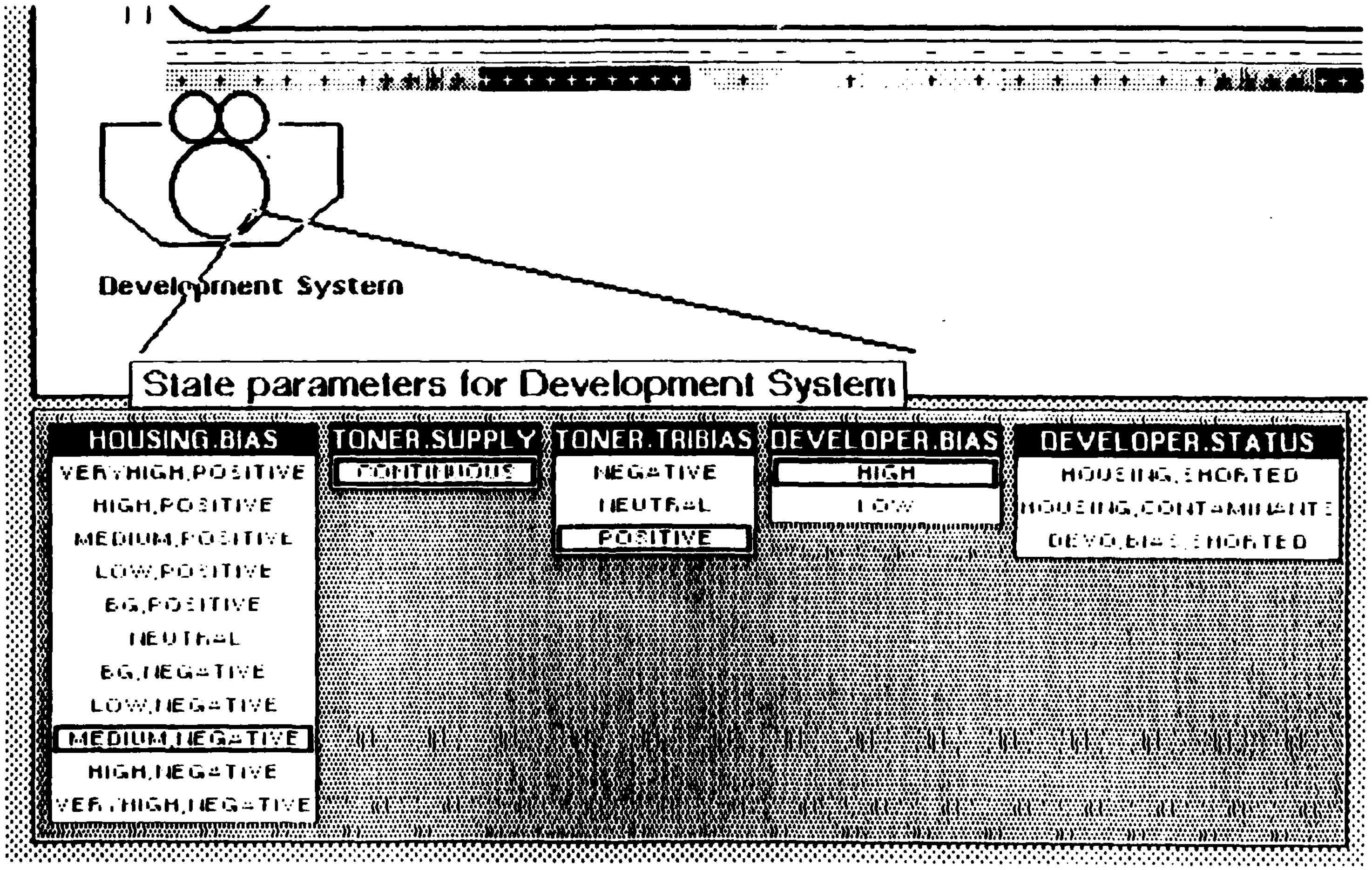
Xerox

This is a sample GACHA 12 BOLD this size should be the easiest to see broken text.

Broken characters are valuable aids when solving copy quality problems.

Left: Adjust a component's state
Middle: Flip up image on Belt (*Edit)
Right: Describe a component (*Inspect)

Figure 7: Setting Parameters for the Development System



State parameters for Development System

HOUSING.BIAS
VERYHIGH.POSITIVE
HIGH.POSITIVE
MEDIUM.POSITIVE
LOW.POSITIVE
BG.POSITIVE
NEUTRAL
BG.NEGATIVE
LOW.NEGATIVE
[✓ MEDIUM.NEGATIVE]
HIGH.NEGATIVE
VERY.HIGH.NEGATIVE

TONER.SUPPLY
[✓ CONTINUOUS]

TONER.TRIBIAS
NEGATIVE
NEUTRAL
[✓ POSITIVE]

DEVELOPER.BIAS
[✓ HIGH]
LOW

DEVELOPER.STATUS
HOUSING.SHORTED
HOUSING.CONTAMINANTS
DEVO.BIAS.SHORTED

Problems with ARIA's Use of Qualitative Calculus.

The qualitative calculus underlying ARIA presents a number of problems. By taking various representational shortcuts, we lost some desirable correctness, flexibility, and explanatory accessibility.

ARIA explicitly computes in terms of quantities (represented by the various QType values). Our QExpressions compute the change in a value relative to its old value. QMOVE is actually a value-setting function. For example, a part of the developer's QExpression is:

```
BELT.TONEDNESS = BELT.TONEDNESS + AmountOfTonerAdded (some computation).
```

The standard technology for problem solving with QExpressions is to turn them into constraint networks and then permit the constraint propagation algorithm to figure out what the various values must be in order to satisfy all the constraints. Thus, we can use a set of constraint equations to do backward chaining to decide what tests to perform in order to assess the cause of a problem in the machine (cite de Kleer's troubleshooting paper). We can likewise use the constraints to forward-chain from a disturbance introduced into the system, and construct a causal explanation of how the device normally operates (deKleer & Brown, 1984).

Unfortunately, these uses of constraint propagation require that each equation be expressible without reference to prior states of QVariables. It must be essentially time-independent. The above equation really says:

```
NEW.BELT.TONEDNESS = OLD.BELT.TONEDNESS + AmountOfTonerAdded.
```

OLD.BELT.TONEDNESS derives from a previous instantiation of this same QExpression (at which time it was NEW.BELT.TONEDNESS) and so we are caught in an infinite regress of belt tonedness values.

The de Kleer & Brown (1984) style of qualitative modelling is based upon equations that represent the derivatives of the ones used in ARIA. Thus, the equations describe the *direction* of change of the values of the devices, not their actual value. The terms in that calculus are restricted to be +, 0, and - (minus), which refer to "increasing", "not changing" and "decreasing". In the ontology of de Kleer & Brown, a component can be operating in one of various *states*. Different states of the component have different characteristic qualitative expressions. By using derivative equations, de Kleer and Brown are able to compute when the component will shift from one state to the next. Note that by reasoning entirely in terms of derivatives, de Kleer's qualitative calculus avoids this problem in local cases (feedback loops are handled differently) because they refer to how the toner level is *changing*, rather than how it is *changed.*

It is important to recognize that in simulation this is not a difficulty: forward-going propagation has no trouble carrying along historical values. However, backward chaining (or other reasoning) over the equations is complicated by this type of analysis. One option is to have the reasoner recognize a loop and post an assumption of normalcy (or not) at that point in the chain of reasoning. For instance, we can begin by assuming that OLD.BELT.TONEDNESS was correct, and proceed with the rest of the causal reasoning. If there is no other problem located, then this assumption of normalcy should be relaxed. The device heuristics in ENVISION (de Kleer & Brown, 1984) serve a similar function.

Qualitative Value Space Division.

In our model, components do not have states (in the way that the devices in de Kleer & Brown's device topology defines the state of a component). Thus, for instance, the corotron is always "ON" in ARIA. We saw that there are states implicit in ARIA, e.g., NODE.STATE = CRACKED, but these are propagated via explicit expressions. One result of this is that ARIA's (non-derivative) qualitative expressions require the use of a finely divided qualitative value space. Unfortunately, there is no guiding principle which can be used by the author of a model to divide up the QValue space into terms such as NEUTRAL, ZERO, HIGH, and so on. Thus, QValue spaces are not appropriately scaled with respect to one another. To simulate the system correctly, the author has to adjust QExpressions (sometimes by inserting arbitrary constants) and redivide the QValue scales, making up unreasonable names and divisions. For instance, there are eleven qualitative values for the developer housing supply bias: VERY.HIGH.POSITIVE, HIGH.POSITIVE, MEDIUM.POSITIVE, LOW.POSITIVE, BG.POSITIVE, NEUTRAL, BG.NEGATIVE, LOW.NEGATIVE, MEDIUM.NEGATIVE, HIGH.NEGATIVE, and VERY.HIGH.NEGATIVE. This large number of values is required in order to obtain sufficient differentiation of charges to obtain all of the incorrect toning phenomena that need to be exhibited in order to satisfy the correctness goal. Furthermore, if we were to divide the charge dicorotron's value space slightly differently, we might need to change the set of 11 above to include more or less terms, in order to again obtain the behaviors appropriately. Weld (1985) discusses this problem in greater detail.

This difficulty derives, in part, from forcing a *qualitative* model onto every aspect of the device. In principle this can be done. However, it would be simpler to do this modelling quantitatively. We are therefore faced with divergent goals: quantitative parameters would simplify the model tuning process but complicate explanation; qualitative parameters have the opposite properties. These goals should not be at odds, and in simpler systems (e.g., electronics) they are not. We will see what can be done about this in the discussion of new modelling principles, in a later section of the paper.

Limitations of ARIA's Spatial Representation.

Because each component refers directly to the QVariables in other components, and knows how to affect those QVariables, the model is effectively laid out in a geometrically arbitrary manner, while still behaving appropriately. (Recall that movement is a side-effect of evaluating the belt component.) Unfortunately, a copier is not a geometrically arbitrary device. Things move around within the copier -- most notably the photoreceptor belt and the copy paper -- in a manner which is predictable, and should be expressible in QExpressions rather than by side-effect. As it stands, it is impossible to write programs which reason about the system as a device which changes in time, since some of the crucial time-effected changes only happen as side-effects.

Another problem with ARIA's side-effect laden spatial representation is that the author of the model must be careful to put his component models in a particular order so that devices with spatial side-effects happen at the right time in the cycle. The ordering information is not explicit in the representation of the model so a reasoner cannot get to it, defeating our accessibility goal. Ordering could be made explicit but this defeats our flexibility goal since such ordering is not a relevant feature of a real copier and would have to be derived by consideration of the model, not the copier.

ARIA's representation of various normal and abnormal spatial statuses causes yet another problem. Above we saw that problems such as a cracked coronode are explicitly represented in the system as if they are measurable properties of the component. In ARIA there have to be explicit QExpressions to handle, for example, the translation of cracks to streaks. This makes the modelling languages particularly inflexible and severely constrains the sorts of bugs that can be represented. Several ongoing research efforts are working on this problem (e.g., Weld, 1985) and it appears to be extremely difficult to do correctly.

The Function in Component Structures.

The difficulty of modelling the paths of interaction in spatially laid out devices led us to embed references from the QExpressions in one component to the QVariables in its neighbor. This creates another significant problem which undermines *flexibility* and the ease of satisfying *behavioral correctness.*

de Kleer and Brown (1984) have emphasized the importance for qualitative modelling of the general "no function in structure" principle. As applied to qualitative modelling, this principle states that components are necessarily "generic" in the sense that they occur in the model as unmodified instances of generic component models. For example, a transistor in an amplifier must be exactly the same as a transistor that plays a role in a computer circuit. In the case of an ARIA model, the several uses of a dicorotron should be able to use a generic dicorotron, and simply provide its attachments and parameters. Adherence to this principle also constrains the author of the model from writing the function of a component into its QExpressions. The example above of the developing rollers functioning like cleaning rollers when the charges are out of balance demonstrates clearly why it is desirable not to write the function (e.g., "adding toner to the belt") into the QExpressions of the developer.

For the most part, ARIA adheres to this principle. However, it fails in one important way. Recall that a QExpression (e.g., the QMOVEs shown above) explicitly selects a target which is a QVariable value *inside some other component* (or appropriately indirected to some other device via a spatial region). Consider, for instance, that ARIA's dicorotron QExpression refers to QValues in the state of the belt segments. The actual photoreceptor belt is differentially conducting in different temperature ranges and in different illumination conditions. In ARIA, one would have to refer to the belt's illumination in the dicorotron's QExpression. This clearly causes problems for our flexibility goal, forcing special case code to be inserted in each device for each instance of its use.

This inability to use generic devices is not necessarily bad, and certainly does not prevent one from modeling other copier designs; however, it does make the task more difficult than we would like. Adherence to the no-function-in-structure principle also guarantees that components behave independently of the overall functionality of the system, allowing accurate predictions of system behavior when individual components are modified (satisfying *behavioral correctness* and *flexibility*).

Limitations in Modelling Abnormal Behaviors.

Not having full capability to model spatial relations between components also limits ARIA's ability to simulate a number of abnormal behaviors. In particular, we are not able to model situations where (a) things that aren't supposed to be in a particular place get there inadvertently; for example, when a piece of paper gets caught in the xerographics chain, or (b) the timing of the system is relevant to its operation; for example, when the photoreceptor belt and the paper path get out of synch.

Furthermore, it is somewhat difficult to add abnormal behaviors to the vocabulary of copier defects. Although the designer can change the model, we have lost the simple relationship between the independent description of a component and its behavior within the xerographic process. One should be able to correctly model every machine fault that is effected by the available parameters by exhaustively searching the component state space -- the *behavioral correctness* goal. However, because components contain local functionality, some settings of component states do not make the overall model behave the way the real-world copier would. The designer can experiment with the process descriptions and QType dimensions to obtain the appropriate behaviors, but extensive exploration by actually running the model is required to determine whether or not the other normal and abnormal behaviors of the model have been thrown awry.

These limitations in the ARIA model led us to consider a redesign of the model, which is discussed in the next section.

DUET: A Redesign of ARIA

In this section we discuss various changes in our modelling technology in response to the forgoing criticisms. We view these changes as a redesign of ARIA and refer to them collectively as DUET. Some of the details in this design are more speculative than others. We have tried to provide a fairly complete outline.

Values and Expressions.

Some of ARIA's shortcomings are caused by our insistence upon enforcing qualitative modelling throughout the model. The model underlying DUET is *not qualitative.* The task of simulation can been separated from the task of explanation by moving to *quantitative* simulation. The resulting values can be translated into qualitative terms for purposes of explanation. This resolves the difficulties caused by trying to use multiplication and addition on qualitative terms that are not divided up into the same qualitative value spaces (and are therefore not continuous). The STEAMER model (Hollan, J.D., Hutchins, E.L., & Weitzman, L. 1984) is quantitative, and Sophie-III (Brown, J.S., Burton, R.R., & de Kleer, J., 1982) divided quantitative values into ranges.

Another class of problems in ARIA is caused by the separation of the spatial character of various phenomena from their quantities. For instance, we saw that a term representing a cracked versus undamaged coronode was explictly converted into a streaked versus uniform charge pattern. In response to this we introduce the notion of *spatially distributed values.* The values used in ARIA are scalar. In DUET, values can also take on two dimensions; we do not deal with the three dimensional case. The value generated by the coronode is a vector which is normally the same all along its length. If the coronode is cracked there might be a dip, which is indicated by smaller quantities in various cells in the vector. A two-dimensional value represents the image (or, charge pattern) on the belt or paper (Figure 8).

Visualization	NODE.STATE =	
	ARIA [scalar]	DUET [spatial]
	SOLID	{+1 +1 +1 +1 +1}
	CRACKED	{+1 +1 0 +1 +1}

Figure 8: Scalar Qualitative (ARIA)
versus Spatially Distributed Numeric (DUET) Values.

We will return to dimensional values in a moment and see how they are changed during transmission.

Processes.

Recall that ARIA has QExpressions which refer directly to QVariables in the neighboring component. For example, we saw that the charge dicorotron changed the belt charge in the belt segments explicitly. In the DUET model, we divide the variables in a component model into two types: *internal variables* and *exported variables.*

We also introduce a class of objects called *processes* that act as intermediaries between components (for instance, the dicorotron and the belt segment). Processes transfer values between the exported variables of different components. For example, the "charging process" reads and sets the exported variables associated with the dicorotron and the belt segment (Figure 9). QExpressions in a particular component can only refer to the variables that are associated directly with that component. The notion of a process-based qualitative physics is due primarily to Hendrix (1973) & Forbus (1984).

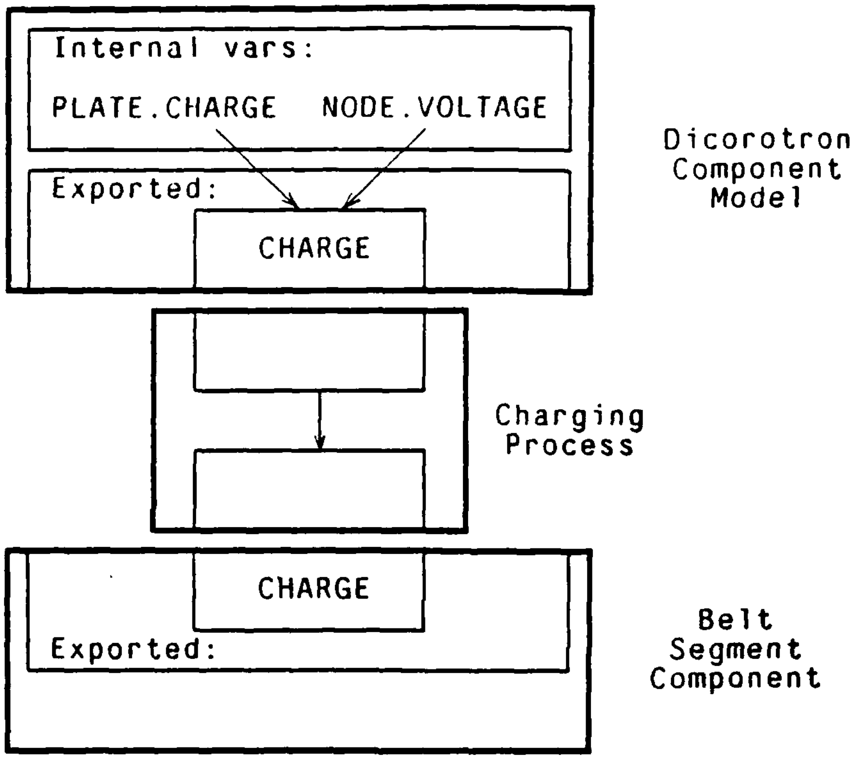


Figure 9: New Component/Process Topology.

Processes represent physical processes in the copier. They are *instantiated* when the appropriate conditions for their existence appear. For example, when an electrical field is put into contact with an object that has electrical capacitance, a process of electrical change (called "charging") is constructed between the two components. When the conditions cease to exist, the process is deactivated[1] This explicit intermediary does away with the notion of spatial region indirection that was used in ARIA to permit components like the dicorotron to communicate with moving components like belt segments. Processes can be instantiated (and are potentially deactivated) between various components as their spatial arrangement changes. Thus, the connectivity of components is explicit.

Processes help avoid the problem of function in structure. We saw in a previous example that in ARIA, for the charge dicorotron to work appropriately in the presence of illumination, the dicorotron's QExpressions refer to the belt's illumination. A possible repair to ARIA would be to have the belt be sensitive to incoming charge, and to have the belt's QExpressions access the illumination when it is charged. However, in the general case under this proposal, each device will be constantly communicating with its neighbors until a settling of values takes place. This is probably closer to the real world, but is not computationally viable. The process notion resolves this by putting an active intermediary into the model, which handles the required resolution of variables in neighboring components.

The rewritten expression for the charge dicorotron looks something like this:

```
Exported variable: CHARGE
Direction is given by tne value of: PLATE.CHARGE
Computation: NODE.VOLTAGE * PLATE.CHARGE
```

and the process of charging the belt reads (terms with "=" represent typed variables):

```
Expression:
      (COPY.VALUE (EXPORTED.VAR =POTENTIAL1)
                  (EXPORTED.VAR =POTENTIAL2))
From: DICOROTRON
To: BELT
```

This might have been instantiated from a more general charging expression reading something like:

```
Test for instantiation:
      (AND  (NEAR =COMPONENT1 =COMPONENT2)
            (GREATERP   (EXPORTED.VAR =COMPONENT1 =POTENTIAL1)
                        (EXPORTED.VAR =COMPONENT2 =POTENTIAL2)))

Process form:
      {Expression to instantiate:
            (COPY.VALUE (EXPORTED.VAR =POTENTIAL1)
                        (EXPORTED.VAR =POTENTIAL2))

        From: =COMPONENT1
        To: =COMPONENT2}
```

The charging process mediates the exported charge variables in the dicorotron and the belt, passing charge values as needed. The charging process is generic, as is the above dicorotron expression. In order for instantiation to take place the belt must export a charge-accepting value, as a result of, e.g., the ambient illumination. All that the charging process can do is mediate charges between charge-taking objects by looking at various required visible variables of those objects: their current charge, their capacitance, etc. All objects which can take a charge (e.g., a belt piece, a stray piece of paper) will take a charge from the dicorotron when a charge process is appropriately instantiated between them. Objects that cannot be charged are not affected by charging processes. (This is probably not the best example for this last point since it is difficult to imagine a real thing that cannot have some charge. However, processes such as illumination are more interesting.)

When do the processes run? Note that the dicorotron must express its charge value, and the belt must express its capacitance and charge values, and those are affected by the local illumination, which is a process acting on the belt, and so on. If we were to permit the various processes to settle over time, ordering decisions would not be necessary. The problem arises because the settling processes are condensed into a single step by the model. One can invent various heuristics for making ordering decisions. One heuristic is to recognize various special cases of conflicting processes and prescribe their order. This could become complicated very quicky for more than two processes. Another is to specify some sort of preceeding negotiation that the processes involved undertake before actually enacting the changes. For instance, if the charging process wants to use the capacitance value on the belt, but the illumination process wants to set it, the illumination will run first. If there are no loops or conflicting references (e.g., two sets to the same variable) then this ordering heuristic will deterministically select an order. In practice this situation very rarely arises, and it seems to be sufficient to simply restrict it for the present.

Spreading.

Processes can change the dimensional character of the values that are transmitted. For instance, the charging process (actually the COPY.VALUE function) will "spread" the one-dimensional vector value that is exported from the coronode, into a two-dimensional charge pattern on the belt. The above example of the instantiation of a CHARGE process was simplified to ignore this fact. Here is a more complete version of the instantiation predicate for charging:

```
Test for instantiation:
      (AND (NEAR =COMPONENT1 =COMPONENT2)
            (GREATERP (EXPORTED.VAR =COMPONENT1 =POTENTIAL1)
                       (EXPORTED.VAR =COMPONENT2 =POTENTIAL2))
            (ONE.D (EXPORTED.VAR =COMPONENT1 =POTENTIAL1))
            (TWO.D (EXPORTED.VAR =COMPONENT2 =POTENTIAL2))
      )
Process form:
      {Expression to instantiate:
            (COPY.AND.SPREAD.VALUE (EXPORTED.VAR =POTENTIAL1)
                          (EXPORTED.VAR =POTENTIAL2))

       From: =COMPONENT1
       To: =COMPONENT2}
```

Note that COPY.AND.SPREAD.VALUE assumes that the target object is moving. This fact is implicit in the fact that the ONE.D and TWO.D predicates were satisfied by the relevant components.

Hierarchical Models.

The introduction of processes in the representation and the additional computation required to process dimensional values leads to renewed concern for the *efficiency* goal. Processes are additional objects which must be included in the overall calculation of the copier's behavior, and processes which are dynamically instantiated between moving components are especially costly. In part to counteract these potential computational problems and improve the processing speed, DUET proposes a hierarchical representation of functionality.

In the discussion of the steps of xerography above, we mentioned that steps are decomposible into components, and those into subcomponents, etc. We can mirror this hierarchical decomposition by allowing components to be decomposed into sub-components and sub-processes. This expansion of detail will also permit us to compact small physical phenomena such as bent rollers or a cracked coronode in a more uniform way, by specifying arbitrary terms to represent the dilation of physical-level symptoms, and packaging the various component expressions to simply deal with these symptoms in a special-case manner.

Compression of components removes some, but not all, of the connecting processes and exported variables in the system. In particular, processes that join the ends of sibling components in the tree and the exported variables of these components remain when the compression takes place. Figure 10 shows this graphically.

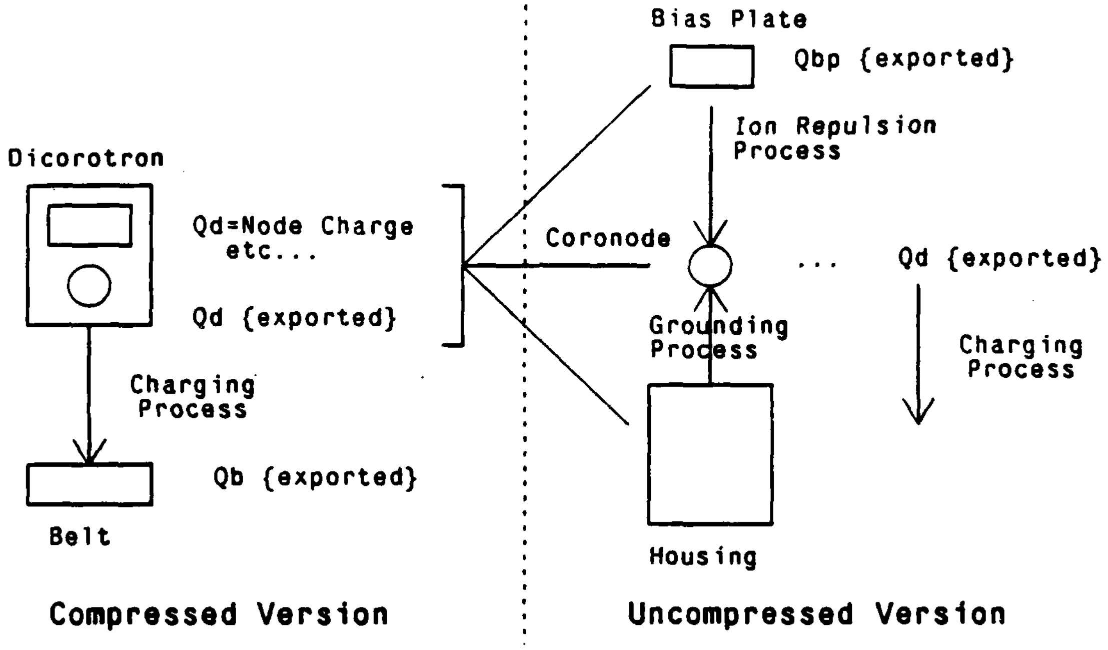


Figure 10: What happens in component compression.

The level of detail to which a particular component and its associated processes are analyzed constrains the level of detail to which explanations can be given by a reasoner. This directly affects the capabilities of the instructional mechanisms that depend on the reasoner. By controlling the level of detail of analysis, the instructor can focus the student's attention on certain aspects of the machine while pushing others into the background. A related benefit of the hierarchical representation is that it can be used to train and encourage the student to think about the functionality of the copier on many different levels. In these ways, this technology advances the *usability* and *explanatory accessibility* goals.

In the simulation model, higher level functions are less computationally costly than lower level functions because of the reduction in the number of components and processes involved in the calculations. The real-time effectiveness of the simulation can be chosen depending on the level of detail desired for a given simulation session: the user can decide the trade-off between processing speed and level of detailed analysis. In this way, this technology advances our efficiency goals[2]

Compression of Process Time Scale.

Part of the reason that ARIA was forced into the modelling difficulties that it has is that it tries to model effects that take place at various time scales, as well as at various levels of detail (see Hobbs, 1985 for a discussion). For example, included with the QExpressions that describe how the developer works are expressions that describe how toner runs out (in the toner supply). The component/process technique of DUET permits us to represent processes in the model at different time scales in a way similar to that used above for describing different component scales. Thus, the toner draining process would be executed, eventually causing the toner to run out, only when the user is running the model at a high speed setting.

This approach has all of the positive and negative features of the component compression: *useability* and *efficiency* are improved, but compressed functions have to be hand written. However, in this case it is possible to literally drop the action of certain processes at longer time scales without sacrificing *behavioral correctness*, because the grain size of what is considered to be correct behavior shifts accordingly. For example, if the student is interested in the time that it takes to cause belt fatigue (many thousands of copies), he or she is not going to be very concerned with the charging process over that period. Analysis of the charging process would be obtained by shifting down to a smaller time scale, or, alternatively, by virtue of long-term charging processes written into the model by its author. Similarly, when interested in the details of small time scale phenomena, the student can get along without the long-range phenomena of belt fatigue or toner drain effects. Information is lost in time scale shifts -- the model runs differently and different things happen. This need not be the case in component level shifts since the information available by the exported variables will presumably be the same as if the expanded component description were being evaluated. Of course, the author may choose to write the compressed model differently.

Conclusion

In this paper, we have discussed our research attempts at designing and building a qualitative simulation model of the xerographic process to be used in an intelligent instructional environment for effectively training Xerox service personnel in the ways of copier repair. Our pedagogical premise is that a good understanding of the xerographic process enhances the ability to troubleshoot effectively, and that a simulation of this process would help develop this understanding. The simulation portion of our project endeavor was directed by five goals: *behavioral correctness* and *expanatory accessibility*, and, to a lesser degree, *useability, efficiency* and *flexibility*. These goals led us to consider using qualitative modelling technologies for the simulation. However, qualitative modelling is not well developed for devices of great complexity, such as copiers, and our modelling attempts here raise interesting issues about the application of qualitative modelling to complex devices.

We constructed a qualitative simulation model of the xerographic process, called ARIA. ARIA consists of components representing real-world copier parts, where the function of a component is represented by qualitative variables (QVariables) and qualitative expressions (QExpressions). QVariables take on qualitative values (QValues) that are defined over a specified qualitative type (QType) space. ARIA satisfies the following of our goals: *useability*, *efficiency*, and, partially, *behavioral correctness* and *expanatory accessibility*. However, ARIA had several shortcomings, most notably in its use of a qualitative calculus and value space, resulting in limited success in achieving the *behavioral correctness* and *explanatory accessibility* goals, and in its structural spatial representations, resulting in a limitation of the model's flexibility and its range of simulated abnormal copier behaviors. The redesign of ARIA, refered to as DUET, overcomes many of these shortcomings. DUET proposes a quantitative model, on top of which ride qualitative terms for purposes of explanation, where *processes* as well as components are represented in the model. DUET is consequently more faithful to the principle of no-function-in-structure, althought the addition of processes make DUET much more computationally intensive. Hierarchical functional decomposition is used as a way of circumventing computational bottlenecks by trading off processing speed with the level of detail of analysis of the copier's functions. Time compression is also featured, alleviating the need to divide the value space up into infinitely fine-grained representation. DUET specifically addresses many of the problems encountered by the ARIA model, and promises to better satisfy our five modelling goals.

Beyond the details of the various modelling decisions made in ARIA and DUET, the principle lesson of this paper is that the design and construction of a simulation model for practical use entails the consideration of many competing goals, and that these goals, when properly weighted, determine where on a continuum of *simulation building difficulty* the modelling task falls. Trying to model selected behaviors of a complex machine is difficult, but not as difficult as also insisting on maintaining correctness without building the entire machine in infinite detail. Hoping to do all this within a system that can also model a wide class of such machines is even more challenging. Further hoping to satisfy pedagogical goals of ease of use and of reasoning over the model makes the problem formidable. We believe that somewhere along this continuum lie our modelling attempts, ARIA and DUET.

Acknowledgements

Steve Bagley, Danny Bobrow, John Seely Brown, Johan de Kleer, Art Farley, Julian Orr, Joe Psotka and Dan Weld read and discussed previous drafts of this report.

References

Bobrow, D., & Hayes, P. (1984). Special Volume on Qualitative Reasoning about Physical Systems. in Artificial Intelligence. vol. 24. #1-3. Also published as Bobrow, D. (1984) Qualitative Reasoning about Physical Systems. North Holland; New York.

Brown, J.S., Burton, R.R., & de Kleer, J. (1982) Pedagogical Natural Language and Knowledge Engineering Techniques in Sophie I, II, and III. in D. Sleeman & J.S. Brown (eds.) Intelligent Tutoring Systems. Ch. 11. Academic Press. London, England.

de Kleer, J. & Brown, J.S. (1984). A Qualitative Physics Based on Confluences. Artificial Intelligence. 24(1-3). pp. 7-83

de Kleer, J.S. (1984). How Circuits Work. Artificial Intelligence. 24(1-3). pp. 205-280.

de Kleer, J.S., & Williams, B.C. (1986) Diagnosing Multiple Faults. to appear in Artificial Intelligence.

Forbus, K.D. (1984) Qualitative Process Theory. Artificial Intelligence. 24(1-3). pp. 85-168.

Hendrix, G. (1973) Modelling Simulataneuos Actions and Continuous Processes. Artificial Intelligence. Vol. 4. pp. 145-180.

Hobbs, J.R. (August, 1985) Granularity. in Procedings of the Ninth International Joint Conference on Aritificial Intelligence. Los Angeles. pp. 432-435.

Hollan, J.D., Hutchins, E.L., & Weitzman, L. (1984). STEAMER: An Interactive Inspectable Simulation-Based Training System. AI Magazine. vol. 5, #2. pgs. 15-27.

Weld, D.S. (1984). Switching Between Discrete and Contionuous Process Models to Predict Genetic Activity. MIT-AI Technical Report #793. Also to appear in Artificial Intelligence, 1986.

Weld, D.S. (September, 1985). Issues in the Automatic Modelling of Complex Mechanical Devices. Xerox PARC working paper.

Williams, B.C. (1984) Qualitative Analysis of MOS Circuits. Artificial Intelligence. 24(1-3). pp. 281-346.

Notes

1. This use of "instantiation" mixes Forbus' instantiation and activation (checking the preconditions for application of an instantiated process) mechanisms.

2. In a perfect world, one would not have to author the higher-level subsystem models. These would be compiled in from the component QExpressions. We plan to have these written by the author of the model, just as if they were individual components. Weld (1984, and in press) discusses a method of automatically doing such compression by analysing the progress of the simulator on a number of cycles, and producing "aggregate" expressions.

Issues in the Automatic Description of Complex Mechanical Devices

Daniel S. Weld
Intelligent Systems Laboratory
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304

WORKING PAPER --- September, 1985

This paper attempts to illuminate the gap between the state of the art in automated reasoning about physical systems and the technology necessary to generate explanations about the behavior of large mechanical devices such as a xerographic copier. The first part of the paper presents the issues abstractly. The possible types of qualitative physics are discussed and distinguished in an effort to clarify when each type is most appropriate. Complex devices present special problems because they require multiple models for efficient reasoning. I discuss the problems with switching between models and sharing information across models. The second part of the paper analyzes a set of interrelated models of a hypothetical copier. Three models of increasing detail are presented. The first model describes the copier functionally. The second describes it in terms of component processes. The third model uses a qualitative physics of three dimensional shape and constrained motion to describe a single component of the copier more accurately.

This report describes research done at the Intelligent Systems Laboratory of the Xerox Palo Alto Research Center. Support for this research was provided in part by the Army Research Institute under contract number MDA903-83-C-0189.

Table of Contents

1. Introduction

The world abounds in complex machines. Consider how much easier it would be to keep them operating properly if the expert designer's knowledge of the devices could be encoded in such a way as to allow automatic generation of detailed descriptions of how they work and fail. This paper explores the issues. approaches. and possible solutions to the problems of explaining the operation of complex mechanical devices such as a xerographic copier.

Following the work of [28], I assume that each component in a device can be explained by recursively describing the component's role in the device, its behavior, its structure in terms of subcomponents, and its mechanism i.e., how the subcomponents achieve the behavior. I assume that the model of the device includes all but the mechanistic information which would be too voluminous to include for all combinations of functioning and failed components. The mechanistic information must be derived.

This paper concentrates on methods for the derivation of mechanistic descriptions by performing causal simulation of a device model. Considerable work has been done in this area [7, 9, 12, 31, 18, 22], but most of this research has focused on rather simple phenomena. This paper discusses extensions to current reasoning techniques addressing the following problems:

- **Rich Physics**
  Xerographic copiers rely on a diverse set of interacting technologies: chemistry, mechanics, magnetics, and electrostatics. The qualitative physics required to model a copier is much more complex than existing proposed physics. Considerable work is required to keep it consistent.

- **Spatial Reasoning**
  An added complication is the strong spatial aspect of reasoning about mechanical devices. The three dimensional shape of many components is critical to their function. Efficient and flexible representations for shapes and constrained motion are required to provide competence in causal simulation of mechanical devices.

- **Device Complexity**
  Copiers, like many other devices, are enormously complex. Explanations based on fully detailed simulations would be too choked with unnecessary information to understand. Although some research has addressed the use of multiple, consistent models at differing levels of abstraction [6, 21, 24], considerable work is required to mesh these ideas with a physics capable of representing a mechanical device. An additional question is which level model should be used at any particular time.

1.1 Disclaimer

This paper represents preliminary research, not a finished piece of work. It is highly speculative and should be taken as a proposal for further investigation. I have attempted to highlight the issues and problems involved with automated reasoning about complex mechanical devices. The solutions proposed are untested and unimplemented, but thought to be in the right direction.

1.2 Outline

The remainder of this paper is divided into two parts. The first part summarizes the global issues with modeling complex physical devices. Section two discusses criteria and tradeoffs for qualitative physics. Section three considers the possibilities and problems with using multiple models to simplify the task of reasoning about large systems.

The second part proposes a specific series of formalisms and models for reasoning about a xerographic copier. Section four presents two models of a copier: a functional model of a copier as a single process and a more detailed model of a copier as a set of interacting processes. Section five develops a qualitative physics of three dimensional shape and constrained movement, and Section six uses this physics to provide a very detailed model of a photoreceptor cleaner. Both operative and broken cleaners are analyzed. Section seven summarizes and proposes areas for future work.

PART I: Survey of Issues

The next two sections discuss the necessary decisions to be made when choosing a world-physics and the use of multiple models to simplify reasoning about complex systems.

2. Types of Models

To model a device, some sort of simplified system of dynamics (perhaps qualitative) is necessary. The different types of change must be described and their interactions formalized. I restrict consideration to representations capable of describing changes as continuous over intervals of time[1] since these [9, 31, 12, 19] support the powerful limit analysis technique.

Since our concern is the explanation of a device's behavior, I focus on the relationship of a causal physics to the task of predicting a device's possible behavior. There are two general ways to do prediction: simulation and mathematical analysis.[2] The difference between the two techniques *results from the different types of models that they utilize--mathematical analysis solves a system of* equations while simulation follows the step-by-step local interactions of a set of components connected in some topology. The simulator predicts the behavior of component through mathematical analysis since components are modeled with some form of differential equations. Thus the two techniques are related; the main difference being the simulation's handling of device descriptions that *are composed out of component descriptions and structural information. In this* section, I concentrate on simulation.

2.1 Desiderata for a Qualitative Physics

Before discussing the important criteria for a qualitative physics, the term should be discussed. What is a qualitative physics? *Because it is a physics, it must be a formalism for reasoning about* changes in the world. The term "qualitative" distinguishes it from traditional physics; it is simpler than full physics as a result of having smaller scope or using a simpler model of state variable than real numbers. Simplicity is critical because without it there would be no advantage over full physics. It must also be predictive, since otherwise it does not describe the world and thus does not qualify as a physics. *These two qualities are so tied to the name that I exclude them from the desiderata below.*

- **Composability**
  It should be possible to take two models that are built using the physics and compose them to form a new model. For example, combining models for a wheel and for string should result in a pulley. If the representation does not allow composition, then it is likely

[1] As opposed to the discrete representations of change that model actions as instantaneous. See [30] for more on this distinction.

[2] A third method for prediction, analysis by reformulation, is less well understood. The following example (from [22]) as well as the classic mutilated checkerboard problem are good examples. Suppose two trains are heading towards each other on the same track, each one moving at 75 miles per hour. When the trains are 150 miles apart, a bird starts flying from one train towards the other at 100 miles an hour. When the bird reaches the second train, it turns around and flies back to the first until it eventually gets smashed. *The question: how far does the bird fly before it gets crushed?* Simulation is not the best way to answer this question. Reformulating in terms of time is much cleaner.

to behave in a surprising manner and will certainly make the description of a complex system arduous. The no-function-in-structure and locality principles of de Kleer and Brown [9] may be viewed as a method for ensuring composable models. By enforcing the no-function-in-structure principle with the requirement for class wide assumptions, de Kleer and Brown have guaranteed that all instances of a class of components will have the same behavior. Component descriptions can thus be composed predictably. QP theory [12] also satisfies the class-wide assumptions criteria, since only processes (i.e. classes of process instances) can be specified.[3] If care is made not to violate the locality principle, then QP theory models will satisfy the no-function-in-structure principle.

- **Natural**
This criterion is difficult to formulate. The interactions between primitive components should mesh with people's expectations of how changes in one component cause the others to change. Since causal physics models are often used to produce explanations of behavior, a natural representation of change is required for the explanation to make sense. The locality principle of de Kleer and Brown [9] can be viewed as a restriction that increases naturalness, since people dislike "action at a distance."

- **Expressive**
The physics should be expressive, handling a variety of types of quantities (see Section 5.2.5) and numerous types of change.

- **Graceful Extension**
As pointed out by Forbus [13], it is advantageous for a qualitative physics to be extendable. Ideally, the physics should support simple reasoning with limited information and allow more powerful conclusions when more (presumably quantitative) information is available. If achieved, this criterion increases the utility of a qualitative physics, because it allows it to be the universal kernel for a wide variety of reasoning types.

2.2 Qualitative vs. Quantitative

Although purely qualitative representations of state variables, quantities, are simple and often convenient, they are not always adequate. When simulating systems with long chains of multiple-input, high-amplification changes, it is necessary to have detailed distinctions between possible values for a variable. Even if three values (high, normal, and low) suffice to describe the output variable for a system, more values will be necessary to describe input and intermediate values if ambiguity is to be minimized.

One approach is to simply add more qualitative values (e.g. very high, slightly high, sorta high, etc.), but this violates the principle that makes qualitative representations useful in the first place: they distinguish between *qualitatively different* regions in the space of values. The key point is that this qualitative difference is dependent on ones perspective--on the types of processes in which one is

[3] Assuming that the individuals field in a process definition does not reference specific individuals and the relations and influence fields only reference objects mentioned in the individuals field.

interested. When reasoning about the temperature of water in pipes, the regions (frozen, liquid, boiling) are important because of the activity of processes that are important to the pipes (e.g. pipes cracking due to the expansion of freezing liquid), but is is not useful to consider the exact temperature of water in liquid form. In the mind of a bather, however, a different partitioning is necessary (lukewarm, nice, scalding) because of the different processes (discomfort, shivering, pain, etc.) involved.

Assuming that one can usefully describe the output of a system with the quantity space (high, normal, low), and the mechanism of the system is too complex to correlate output value with any small subset of input values, then it is likely artificial to use a purely qualitative representation for the inputs. One might as well use the real numbers. There are three ways to do so:

- **Dual Models**
  Two models could be kept in synchrony. All computation would be done in the more detailed quantitative model, and the results would be summarized to the qualitative model by seeing what regions the variables were in after time steps. This approach is an idealization of the Steamer system [27]. Advantages include accuracy and easy interface to graphics since exact values would be known. The major disadvantages are the complexity of the more detailed model (it's not a qualitative physics), the problem of ambiguous or unknown input values, and the inability to keep useful dependency information.

- **Disambiguation**
  Another approach is to reason completely qualitatively until reaching an ambiguity [8], and then consult an oracle. This approach is being used in a MOS circuit simulator.[4] The oracle is a numerical simulator; upon ambiguity, a description is composed (using default values if necessary) and fed to the quantitative engine. The result is summarized qualitatively, and qualitative simulation continues. The advantage of this approach is its simplicity and almost pure qualitative nature. Disadvantages arise from the need to choose default values and the loss of dependency information when consulting the oracle.

- **Quantity Lattice**
  I recommend using a mixed qualitative/quantitative representation like the quantity lattice of Simmons [22]. This representation allows assertions about quantity values to be both qualitative (i.e. inequality information) and quantitative (i.e. a given quantity can be specified to be in an real valued interval). A constraint propagator uses all available information to narrow the bounds on each quantity as much as possible.

There are two other issues subissues to consider:

1. How should one interface a quantity lattice system to a graphics package? When one doesn't know the exact value of a quantity, does one just guess and use backtracking to

[4] Williams, personal communication.

resolve inconsistencies? John Mohammed is rumored to have such a system.

2. How do the different qualitative quantity space representations compare? Is the plus/zero/minus space of de Kleer et al. equivalent to Forbus' spaces which allow an arbitrary number of limit points?

2.3 Device v.s. Process Orientation

Proposed qualitative physics fit into two classes depending on the locus of change: device and process centered models. These two classes can be thought of as complements of each other.

QP theory [12] is the main process centered representation. All changes are caused by process instances, abstract forms of change that occur whenever conditions are right. Each process is defined by listing the individuals (objects) necessary to instantiate the process, the conditions (predicates) that must be true for the instance to be active, relations (definitions of quantities and qualitative differential equations between quantities) that are held true if the instance is active, and influences (tendencies that differentiated quantities have on other quantities) that are in effect if the process instance is active. Simulation proceeds by determining the process structure (set of process instances that are currently active), resolving influences (summing influences over all active instances to construct differential equations, then solving these equations and those from the relations fields to determine the derivatives of quantities) and performing limit analysis (determining what quantities will first change values to create a new process structure).

The work of de Kleer and Brown [9] is an example of a qualitative physics centered on device descriptions. All changes occur as the result of action by some component that is topologically connected by conduits in a network of components. Each component is defined in terms of a number of operating regions (inequality predicates on local quantities). The behavior of a component is described for each region by a series of confluences (qualitative differential equations) that must hold when the component is in that region. Ports on the components are connected by conduits which only convey information from one end to the other; they do not allow quantities to change. Prediction includes reasoning about intra- and interstate behavior, where intrastate behavior is the change of quantities that do not result in a transition of a component from one operating region to another.

2.3.1 Comparison

Restricting a QP model to a fixed process structure and the confluence physics to intrastate behavior, both physics are of comparable expressibility. The difference is that the device physics specifies behavior in a fixed set of confluences while the QP description includes some straight confluences (relations) and influences that need to be summed using a closed world assumption to produce the remaining confluences.

I suspect that the discrepancy in qualitative values (de Kleer and Brown allow only the values plus/zero/minus while Forbus allows an arbitrary range) is not significant. but I haven't show this.

Differences do appear when considering interstate behavior. because the device physics assumes a static topology of components connected with conduits while the topology is more dynamic in QP theory. In fact since certain types of individuals (liquids. say) can split to create more individuals. their could be an unbounded number of process instances resulting from a finite QP description. To match this a device physics would need to be able to create devices and modify the topology dynamically. It is not clear (and probably doubtful) that this distinction is important in practice.

2.3.2 How to Choose

Even if the two orientations are equally expressive, it does not mean that they are indistinguishable for any given representation task. Are there cases where one is more convenient than the other? Which is better for specifying complex mechanical devices?

For any given domain, the choice seems to depend on where the forces of change are localized relative to the objects. If the changes are internal to the objects (as is the case in digital and analog electronics) then a device centered physics is appropriate. If the changes occur in the connections or interactions between objects (as is the case in the domains of molecular genetics and naive models of liquids) then a *process model* may be more natural. This distinction is a natural result of the complementary nature of the two orientations. The choice will always be arbitrary since the distinction between object and connection is a matter of taste. The natural device centered model of a liquid pressure regulator [9] emphasizes this point. It also suggests the need for a unified physics that could consistently represent changes in both formats.

For the domain of complex, mechanical devices, I deem the process model more appropriate since the most natural choice of objects is the various geometrically shaped parts. and the changes seem to occur at the contacts between objects: rotation is change of position relative to an axle; rubbing, scraping, charging, all are changes that occur between objects.

2.4 Other Issues

Here are some other important issues which distinguish the different proposed physics. but that I haven't had time to research thoroughly.

- **Temporal Model**
  There are a variety of different temporal models: Allen's interval representation [1], McDermott's time logic [20] (with persistences), Tom Dean's work [11], Yoav Shoham[5] is a cleaned up and enhanced version of Allen's model, and Brian Williams' temporal constraint propagator (TeCP).

- **Handling Feedback**
  Williams' [31] simulator handles feedback explicitly by recognizing situations where the input to a sum is a function of an output. Rules are provided for recognizing both resistive feedback and cases with memory. Another rule determines which term is the cause and which is the effect in cases where bidirectional comparisons occur. Higher order derivatives can also be used [31, 10].

[5] PhD thesis, Yale University, In Publication.

3. Managing Complexity

The complexity of large systems makes it difficult for both people and programs to reason about them. The standard doctrine from engineering is to manage complexity through the use of abstraction techniques. This technique works best when reasoning about engineered systems, because these systems are usually built in layers with clean interfaces. Yet we use abstraction techniques when trying to understand other (perhaps less modular) systems simply because it is one of the only techniques that we have.

Work in naive physics [14] suggests that competent reasoning can result from a number of interacting locally-consistent, locally-connected models of the world. Since each model is simple, reasoning should be relatively easy assuming that the correct model is used. Utilization of these multiple overlapping models (some at each abstraction level) is another technique for managing complexity that is quite similar to shifting abstraction levels: similar selection and reformulation operations between models ("choosing the right viewpoint") are required.

For a causal simulator to benefit from techniques involving multiple models, it must get the model somehow, either from a database or by dynamic construction. If the device's mechanism is unknown, then the reasoner must create the model dynamically (e.g. the diagnosis program ABEL [21]), but if the mechanism is fixed (as is the case when performing simulation, test generation [24], etc.) the different models may be precomputed.

To use multiple models to reason effectively about a device, three questions must be resolved: which model should be used at any given time, how can existing information be reformulated to fit usefully into a new model, and where should the model boundaries be drawn in the first place? We discuss these issues after briefly summarizing previous work.

3.1 Previous Work

There have been two major AI programs which reason at multiple abstraction levels: the ABEL medical diagnosis program and the Saturn digital electronics test generation system. Work on formalizing clusters of knowledge [14] and reformulation between viewpoints through the use of cliches [5, 17] is more speculative.

3.1.1 ABEL

ABEL [21] is a medical diagnosis program that diagnoses acid-base electrolyte disorders. It distinguished a structural and a multilevel causal representation of a system. The structural representation included a part-of hierarchy for organs and parts with notion of spatial containment. Each level of the causal representation (there were five levels) consisted of a network of nodes (states) related with "causal links" which told how to compute the value of either node from the other. Causal links did not distinguish between probably empirical associations and actual guaranteed causality, so only weak transitivity was obeyed. At a given level a node could be composite (i.e. it had an elaboration network at a lower level) or primitive. If composite, there must be a focal node in the elaboration network which got mapped one to one with the higher level node. Three key operators were used to build different possible multilevel models of a diseased patient that were rated once complete. Aggregation summarized a network into a node or link at a higher level. Elaboration did the opposite: it built a lower level network to explain a link or node. Component summation and decomposition was used to sum low level causes to determine the magnitude and duration at a higher level (it also handled feedback).

The work was pioneering, but its weak causal model and restricted mappings between levels (one to one mapping between focal nodes) limit its usefulness for the domain of mechanical devices.

3.1.2 Saturn

SATURN is a test generation program for digital electronics [24]. It represents a model of a device at multiple levels of abstraction using a device independent language, but the descriptions are a (precomputed) fixed network of modules that are not modified during operation. Elaboration is relatively simple since ports do not gain structure and since single focal nodes are assumed. Quantities ranged from bit vectors to integers.

3.1.3 Other

One should also consider Korf's work on macro operators, David Chapman's Cognitive Cliche paper [5], Glenn Kramer's work on understanding devices with cliches [17], Jerry Hobbs' work on granularity [16], among others. Unfortunately, these papers can not be discussed here.

3.2 Choosing the Appropriate Model for Reasoning

Given a model with multiple levels of detail, which level should be used to solve a given problem? An obvious answer is to use the simplest level that will suffice. For causal simulation that means using the highest level that will answer a question or support reasoning about a particular (perhaps faulty) component.

Since each component in a device is modular, it makes as much sense to talk of the level of detail for a given component as it does for the device as a whole. The critical question, then, is whether the level of reasoning must be uniform or can it be heterogeneous.

- **Homogeneous**
  Requiring reasoning to be performed at one level of detail for the whole device simplifies the problem of consistency between levels, since there would be fewer possible connections between modules--connections such as the one shown in Figure 3-1 would be illegal. Unfortunately, the simplicity in implementation comes at the cost of increasing extraneous detail in simulation.

- **Heterogeneous** Just because one component must be simulated at a detailed level, doesn't mean that all components do. Most probably, only one component is responsible for a certain aspect of behavior, and only it need be simulated in a detailed fashion. Abstract descriptions of the other components should be used to simplify reasoning.

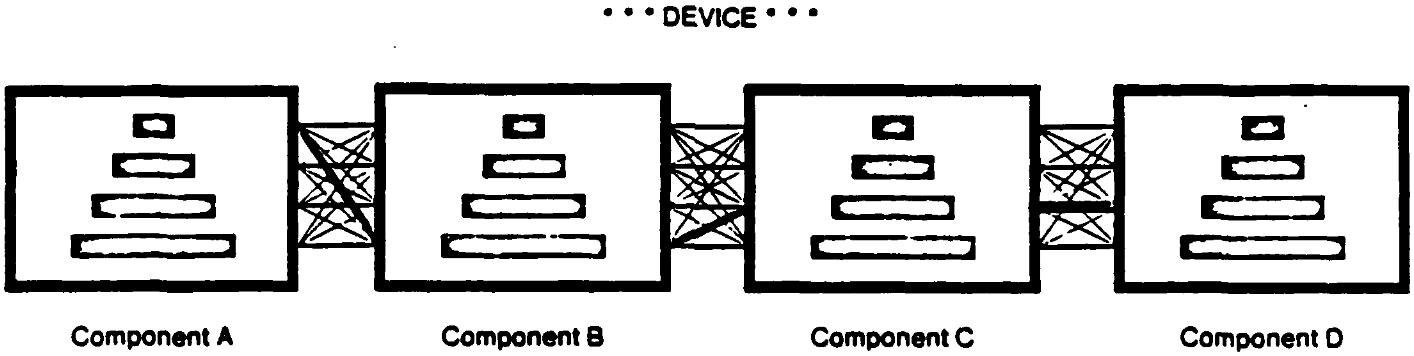


Figure 3-1: Heterogeneous Level Connections Between Components

It seems clear that heterogeneous level shifting is preferable if it is possible to keep all possible port connections consistent. Use of this technique results in the natural generation of the empirically observed mixed level explanations from human subjects.

3.3 Switching Models

Models can differ by abstraction level or by viewpoint. To simplify the issues involved, this section will only discuss the process of changing between models at different abstraction levels. Although the discussion is based on the concept of a single dimension, abstraction level, keep in mind that this is only reasonable if the models are extant; there are far too many possible elaborations of a component to be characterized by a single dimension.

Determining how to switch levels consistently is much harder for programs which do diagnosis: they must move down as well as up in the abstraction hierarchy. Although finding the best level to use to answer a question may involve the same problems. Moving down in the hierarchy, elaboration, is harder than moving up because information must be added to complete the new model. It is then necessary to choose between completions of the resulting ambiguous models.

What happens when levels shift? The set of valid components changes. The set of quantities changes as does their type. And the interface between components can change.

3.3.1 Component Decomposition

If the system decides to elaborate the description, i.e. shift to a more detailed model, then the current component can no longer be considered primitive. It is replaced by a group of subcomponents that are interrelated with some topology of connections. Since there is new detail, new quantities may be necessary to describe the state.

ABEL's mapping between different-leveled quantities is inadequate, however, because there is not always a homomorphism between high level terms and low level ones. Certain abstract terms may disappear at the elaborated level, replaced by a collection of terms. For example, it is reasonable to characterize a copier by its dirtiness at a very abstract level. But when reasoning with a detailed model, there is no one dirtiness; instead there is *photoreceptor fatigue*, *dicorotron dirtiness*, *smears on the optics*, etc. To deal with complex mechanical devices, a more sophisticated mapping scheme is necessary.

3.3.2 Reformulating Quantity Types

Quantities can also change type when the level of the model shifts. For example, at one level of model, it is reasonable to conceptualize the paper in the copier solely by its amount; the paper is described completely by the value of an ordinal quantity.[6] But when reasoning with a more detailed model (perhaps attempting to understand the cause of a jammed paper path) more structure is necessary. The paper is more usefully considered as a stack of discrete elements with a unique top page that moves about. The old ordinal quantity is mapped into the reformulated description as the depth of the stack.

Toner provides another example. At the topmost level, it too is best modeled by an ordinal

[6] In previous papers [30] I have used the term "linear quantity" to denote a quantity whose possible values are totally ordered, but I now use the term *ordinal quantity* for this concept.

quantity, amount of toner. When reasoning about the mechanism of a subcomponent (the cleaner say) the toner gains new structure (a liquid) to better model the types of behavior it exhibits (coating a region of surface, soaked up by the sponge-like action of the charged toner roller, spreading down the slanted surface of the cleaner blade, ...). In the even more detailed model required to explain the mechanism of the magnetic brush, the toner must be considered as a collection of particles that cling to carrier beads.

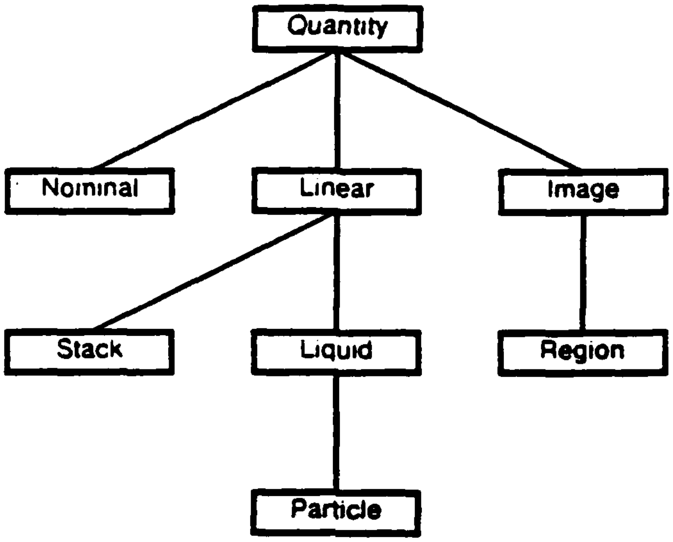


Figure 3-2: Reformulation Links for Quantities

By investigating the possible reformulations of different quantity types, I obtained the (incomplete) DAG displayed in Figure 3-2. Keep in mind that the edges in the graph are not ISA links, but rather display possible reformulations. The liquid box refers to the model of liquid as contained space [15]. The region box denotes the robust surface description representation language (a set of rectangles) described in Section 5.2.5, and the image box refers to a copy quality language (under construction) which has separate terms for streaks, strobes, etc.

3.3.3 Consistency

It is a possibly provable theorem that no simple model of the world can predict behavior correctly in all interesting cases. Since simple models have useful computational advantages, there is strong pressure to achieve a wider range of predictivity by considering systems with many simple models. But this implies that the different models will be inconsistent (since if they always agreed, then there would be no point to have more than one). For example, unlike the toner-as-linear-quantity "amount" model, the toner-as-property-of-surface model (Section 5.2.5.4) predicts the likely result of rubbing and scraping processes correctly. Yet the latter model doesn't support conservation of mass arguments even though the former does.

Our whole notion of model seems inadequate as it stands. Models need to be augmented with some representation of their limitations, their edges. Preferably these edges would be annotated with the name of another model that correctly deals with the limitation.[7] Possible approaches to this problem include:

- Specifying the region of applicability by a set of propositions that must be true for the model to be used. This probably subsumes all other approaches, but offers little guidance on the nature of the best predicates to use.

- Specifying a model's range of utility by describing the space of questions that it can be used to answer.

Much more work on this subject is needed.

3.3.4 Interface Decomposition

The standard assumptions about multiple models at different abstraction levels assumes that each component can be described in terms of subcomponents independent of the other components. In other words, the interface between components is always adequate when reasoning at the level of their subcomponents. Unfortunately, although this is often true for structured software and digital electronic systems (which have been designed with abstraction in mind), it is not true for mechanical devices. Often the interface between components expands as the components are elaborated.

For example, consider a gear connected to a chain.[8] At one level of detail, they are simply linked by an infinite friction connection; the interface is simple: transmission of force. When reasoning about the gear with a more detailed level (i.e. when considering the movement of its teeth) the interface is no longer adequate: the intermeshing of the teeth and holes is necessary.

One could object that the choice of gear as a separate component is the root of the problem. In fact, the problem would disappear if the gear and chain were together considered a single component. But this seems unnatural. Certainly, the model for some viewpoint will require them to be separated.

More work is necessary to determine how to handle this problem.

[7] This can be viewed as the brittleness problem in expert system work.

[8] Thanks to Dan Russell for this example.

3.4 Building a Model with Multiple Abstraction Levels

One key question is how to choose the levels and the decomposition into components when modeling a device. In the past this has been done mainly by art with few principles involved. Arguments have been made for modularity and locality. One chooses a decomposition into components by reducing the interactions between components into a simple interface.

One new hypothetical principle results from the reformulation DAG of Figure 3-2. Clearly, the DAG constrains the possible elaborations of a component by restricting the new types of various quantities. In addition, I hypothesize that the models that result from selectively reformulating quantities in the model, *moving only one step through the DAG at a time*, will probably all be useful.

For example, consider the model of the cleaner presented above. In this model, toner is represented by the very thin volume of space that it occupies directly above regions on surfaces. A more abstract representation would result from eliminating the height component and treating toner simply as a property of surfaces[9] (i.e. a surface could be toned) rather than a property of volume. A more detailed representation might refer to toner as a collection of particles. By varying the type of quantity used to represent toner, the physics is forced to change. Specifically the semantics of the tight and loose sliding connections and the scraping and rubbing processes would change. Section 5.4 defines the physics for the contained volume quantity type--if two surfaces are connected with a tight sliding then they touch, but if the are connected with a loose sliding then they don't quite touch, they only share a directed surface. If toner were a property of surfaces, then both of the sliding contacts would touch identically, and attraction would differentiate. If toner was modeled as particles, then the physics would need to include terms for substance instead of describing objects solely in terms of their shape.

If validated, this principle would still not totally constrain the choice of models, but the guidance provided by the DAG should be helpful.

[9] For simplicity this model was left out of the DAG of Figure 3-2.

PART II: Proposed Solution

The following sections describe one possible framework for reasoning about xerographic copiers. A hypothetical copier is defined at two levels: user's view and summary mechanistic view. To reason about the actual operation of the components of the copier, a physics of constrained motion is presented. This is used to explain the operation of copiers with broken and normal cleaners.

4. A Xerographic Copier

This section presents two models of a generic (hypothetical) copier: a functional model which describes the copier as a single component, and a more detailed description of the copier's internal process structure.

4.1 Copier as Process

The following functional model of the copier's normal operation is sufficient for most users of the machine. If the device malfunctions then a more detailed model is necessary. The status of the button and the position of sheets of paper are nominal quantities, while the amount of toner and paper, the copier's wear, and the image quality are all ordinal. Operation is described as a discrete process:

```
(def-dp copy
  (when (and (= button-status pushed)
             (< 0 (amount toner))
             (< 0 (amount paper)))
        (:= (contents (position output-tray))
            (paper-with-image
             (muckup (image original) (wear copier))))
        (increment (wear copier))
        (decrement (amount toner))
        (decrement (amount paper))))
```

The discrete process, copy, is active whenever the button is pushed as long as the copier has enough paper and toner. When active, a new piece of paper appears in the output tray with an image that is based on the image of the original but affected by the copier's wear. In addition, the wear is increased and both the paper and toner supplies are decreased.

There are two important facts to notice about this formalism.

- Mass is not conserved. The model doesn't say where the toner goes, nor that the sheets of paper really move. Instead quantities are created and destroyed in an ad hoc manner. This isn't necessarily an important drawback since conservation-enabled reasoning isn't probably too useful at this level. The added simplicity and relevance of the ad hoc description is a benefit.

- Abstracting the notion of copy quality to an ordinal quantity throws away a lot of information. If not in this model certainly the next should have some sort of qualitative copy quality language (e.g. describing defects in terms of streaks, strobes, high background, contrast, etc...). At my lowest level model, in Section 6, images (and thus defects) are described in a rectangular layout language. More work needs to be done to develop this intermediate vocabulary.

4.2 Internal Process Structure of Copier

A more informative model is achieved if the behavior of the copier is specified by multiple interacting processes. The interaction is centered around a circular photoreceptor (PR) belt.

4.2.1 Definition

```
(def-circular-quant (position pr-doc-area)
  (charge expose develop transfer clean))
(def-ordinal-quant (charge d) -high)
(def-ordinal-quant (wear d) zero)
(def-surface-quant (brightness original))
(def-surface-quant (charge pr-doc-area))
```

The pr-doc-area, a rectangular area which holds the latent document image, is described by its position along the circular path of the belt. The ordering among some initial distinguished points is given. The charge and wear of the dicorotron, d, are represented by ordinal quantities. Both the brightness of the original image and the charge of the pr-doc-area are defined as a surface, i.e. a collection of rectangles each with an associated ordinal quantity value.

```
(def-dp charge
  (when (= (position pr-doc-area) charge)
        (:= (charge pr) (charge d))
        (increment (wear d))))
```

The charge process is active whenever the pr-doc-area is in the charge position. Its effect is to charge the photoreceptor frame to the voltage on the dicorotron and to increase the dicorotron's wear.

```
(def-dp expose
  (when (= (position pr-doc-area) expose)
        (:= (charge pr-doc-area)
            (discharge-fn (charge pr-doc-area)
                          (brightness original)))))

(= (discharge-fn high black) high)
(= (discharge-fn high white) zero)
```

When the pr-doc-area moves to the expose position, its charge is reduced in the places where the original is light and unchanged where the original is dark. The function, discharge-fn, denotes qualitative minus which acts monotonically on its two qualitative arguments. When the arguments (as in this case) are surfaces, then a new surface is constructed by comparing the values of the rectangles of the arguments. The two equality statements serve to calibrate the function.

```
(def-ordinal-quant (amount toner))
(def-surface-quant (toner-density pr-doc-area))

(def-dp develop
  (when (and (= (position pr-doc-are) develop)
             (> (amount toner) zero))
        (:= (toner-density pr-doc-area)
          (charge pr-doc-area))
        (decrement (amount toner) (charge pr-doc-area))))
```

The amount of toner in the developer is represented as an ordinal quantity, and the toner density on the photoreceptor document area is another surface. The develop process is active whenever the document area is in the develop position, and the amount of toner is positive. When active, the document area gets coated with toner in direct proportion to its charge. This uses up toner proportional to the document area charge.

```
(def-ordinal-quant (position paper)
  (input-tray transfer fuser output-tray))
(def-ordinal-quantity (amount paper))
(def-surface-quant (toner-density pr-doc-area))

(def-dp transfer
  (when (and (= (position pr-doc-area) transfer)
             (> (amount paper) zero))
        (:= (toner-density paper)
            (part (toner-density pr-doc-area)))
        (:= (toner-density pr-doc-area)
            (part (toner-density pr-doc-area)))))

(=> (> ?amt zero) (> (part ?amt) zero))
(=> (= ?amt zero) (= (part ?amt) zero))
```

The position of a sheet of paper in the paperpath and the number of sheets of paper are modeled as ordinal quantities. The paper's surface, like the pr-doc-area, can be coated with toner. The transfer process is active when there is a sheet of paper and the pr-doc-area is in the correct position. When active, some nonzero part of the toner is transferred from the document area on the photoreceptor to the paper. Since this is a model of only the xerographic train of a copier, it ignores the movement of paper, attachment to and stripping from the photoreceptor, fusing, and the control issues involved in assuring that the paper arrives various places at the correct time.

```
(def-ordinal-quant (amount waste-toner) zero)

(def-dp cleaning
  (when (and (= (position pr-doc-area) clean)
             (< (amount waste-toner) full))
        (:= (toner-density pr-doc-area) zero)
        (increment (amount waste-toner) (toner-density pr-doc-area))))
```

When the document area reaches the clean position then all the toner is removed as long as waste receptacle is not full. If there was toner on the document area, then the waste receptacle becomes more full.

```
(= (position pr-doc-area) clean)
(def-cp rotate (increase (position pr-doc-area)))
```

Assuming that the document area is initially in the clean position, the simulation can be started by defining a continuously active continuous process that moves the document area around the belt.

4.2.2 Discussion

This model seems to be at the right level of detail for explaining xerography to novices. Like the ARIA model it could easily be interfaced to a graphics package for instructional purposes. Since it consists primarily of discrete processes, the only difficult implementation issues would be the handling of surface quantities and exploitation of the constraints on the qualitative functions (e.g. part, discharge-fn). Forbus' implementation techniques for qualitative proportionalities [13] would be applicable to this latter issue.

The model could be used for more than short term simulation. If long term conclusions were desired about the system's performance, aggregation [30] could be used to recognize the repeating cycle of processes and generate a continuous process abstraction that made explicit the net change in various ordinal quantities over time (e.g. (wear d), (amount toner), etc.).

The main problem with the model is its *implicit* treatment of structure, *essentially* a *violation of* the no-function-in-structure principle. This complicates modifying the model in order to reason about faulty copiers. It is easy to describe a fault as the absence of certain functionality (i.e. the cleaner isn't working), but more difficult to specify partially functional behavior. See Section 6.2 for an example. The real cause of the problem is the high abstraction level of the model. By making the model even more detailed it is possible to increase versatility.

5. A Qualitative Physics of Shape and Constrained Movement

Without a language for specifying three dimensional shapes and certain types of movement it would be impossible to describe the detailed interactions between subcomponents of a mechanical device. In this Section I discuss the goals for this language, the issues involved in satisfying the goals, the simplifications necessary to restrict the problem, and a possible solution. In the next Section, I demonstrate the language's expressibility by presenting a detailed description of a xerographic photoreceptor cleaner.

5.1 Goals

The first goal is to develop a framework which can support efficient spatial reasoning. In addition, the representations should be as domain independent as possible and interface cleanly with other types (e.g. electrostatic) of change. Supporting other types of reasoning (e.g. measurement interpretation, planning, troubleshooting, etc.) is a secondary goal.

An inherent premise which underlies this endeavor is the assumption that there exists a simplified, consistent model of the physical world that is useful for reasoning. I believe that there probably are models of the world that are simpler than modern physics and are still powerful. I believe that we all have these models in our heads. But it's highly unlikely that the models are consistent. Although they may be useful for solving certain problems, the simplified models (just like Newtonian mechanics) will have limitations, edges to applicability. An important goal is to find a way to describe these edges and to catalog how certain types of edges point towards different models with different strengths. These issues are discussed more in Section 3.

5.2 Issues

In pursuing these goals, numerous issues were raised.

5.2.1 Three Whole Dimensions?

Do the representations need to allow full three dimensional objects and movement? Clearly many devices can be described with only two dimensional slices. A functioning cleaner can, but how about one with a scratched toner roll or a bent cleaner blade? How about a differential gear? If two dimensional slices are adequate, how can they chosen?

5.2.2 No Function in Geometry

It is seductively easy to mix functional information with geometrical information, i.e. to have a separate vocabulary term for axle, lever, etc instead of one term for a long thin rod. But this is a serious mistake because it eliminates possible behavior, i.e. using an axle as a lever, or as both. Since all motion can be derived from descriptions of the shape of rigid members and the types of joints, it is best to describe this type of functionality completely geometrically. This requirement is analogous to the class wide assumptions and locality restrictions recommended in [8, 9].

5.2.3 Interfacing Kinematics and Dynamics

Since the mechanisms of many devices can be understood without considering forces, only a kinematic theory of movement is needed. Representing other types of changes (e.g. electrostatic), however, requires the full qualitative dynamics of QP theory. What is the best way to extend the kinematic theory to allow changes in motion (keeping the expressive range of the physics at the uniform level of dynamics) while minimizing complexity? There are at least three possibilities.

1. Bite the bullet and provide a full dynamics theory of motion. This would greatly complicate reasoning, but allow complete consistency.

2. Attempt to fix the mass of all objects to epsilon or unity so that force becomes equivalent to acceleration, thus blurring the distinction between the kinematic and dynamics theories. It could be difficult or impossible to get this to work correctly.

3. Provide a vastly simplified dynamics theory initially with the hope of eventually adding detail. This can be done by including discrete processes for changes in velocity. Since the dynamics processes are discrete, state is not defined during acceleration resulting in a considerably simpler theory. As detailed below, I adopt this approach.

5.2.4 Dynamically Changing Joints

When the joints between members change dynamically, reasoning becomes much more complicated. Consider Figure 5-1.

On the left end is a block which can slide to the right or left. On the right end is a wheel attached to a rod. If angle 2 is large (as shown) then the block can slide freely. If angle 2 is small, then the wheel will restrict the range of the block's motion. But if the block has slid far enough to the right before the wheel comes down, then the block could slide freely while the wheel rolled on its top.

One way to look at this problem is encapsulated in the following question: "When is it ok to assume that nothing will interfere with anticipated motion?" Consider Figure 5-2.

Since the dense object on the left weights more than the object on the right, the pulley will

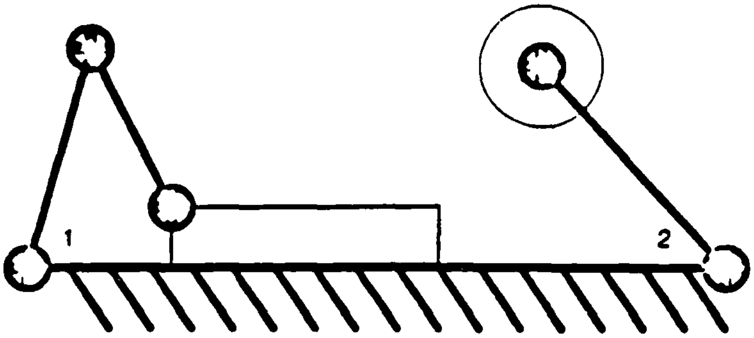


Figure 5-1: Device with Several Possible Joints

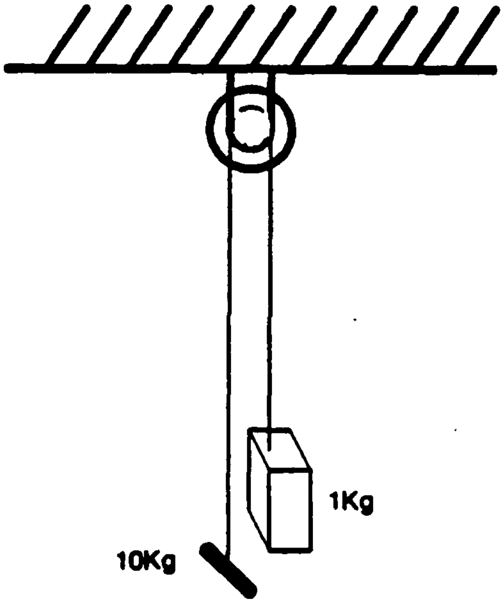


Figure 5-2: Pulley Problem

rotate counter clockwise, the 10Kg object will fall, and the 1Kg object will rise. But what will be the final situation? It is clear that the 1Kg object will go all the way up to the pulley, but what then? It requires much more detailed reasoning to determine whether the 1Kg object will fit through the pulley. Yet people know that they don't need to do that detailed reasoning before the object gets near the pulley. It seems as though they think about the starting situation and determine the initial motion, then do limit analysis (using some sort of spatial limit conditions) to conclude that they don't need to do more detailed reasoning until the object reaches the pulley.

This is interesting, but also potentially complicated. It would be nice to finesse the problem by avoiding devices where joints (i.e. between the 1Kg object and the pulley or between the wheel and block) can form dynamically. Unfortunately, these dynamic joints are present in the xerographic cleaner--at different times the toner is attached to the photoreceptor, the cleaner roll, the toner roll, and the cleaner blade. As this is the only place where joints change dynamically and since toner must be dealt with specially anyway, I'll attempt to handle the problem in this restricted case only. In general, joint types will be assumed static. In other words all contacts between rigid members (besides toner) must be specified as part of the device topology.

5.2.5 Quantities

Previous work in qualitative physics has dealt only with simple ordinal and nominal quantities. Ordinal quantities such as the volume of liquid in a container or voltage across a resistor have a total order relating the possible values. Nominal quantities such as the state of a light switch (on or off), however, have no meaningful relation between values.

The domain of mechanical devices (copiers in particular) is full of quantities which do not fit into either of these categories. New representations must be developed to deal with them.

5.2.5.1 Circular Quantity Spaces

Some quantities, such as the position on the rotating photoreceptor belt, have no total order because they are circular. Presumably the use of modular arithmetic will provide a simple way to extend limit analysis techniques to work with these quantities.

5.2.5.2 Regions on Surfaces

A good representation for regions on surfaces is especially important when modeling xerographic processes. During the different steps of xerography, an image gets passed about between device components in different forms: as a light pattern, as a distribution of charge, and as a density of toner on different substrates. In each of these phases, the image is best described as a set of regions on a surface with some sort of attributes (e.g. illumination, charge, toner density). Thus, a region specification language is quite essential. There are at least two possible ways to specify regions:

1. Each surface could be represented as a matrix of "pixels", each of which could hold a distinct value for the attribute of interest (charge, etc.). This scheme would be easy to implement and allow simple simulation. Unfortunately, the representation is at such a low level, that it would make meaningful descriptions of images or defects (i.e. "normal image except for a narrow stripe of black") difficult. Before explanations could be generated, the region representation would need to be summarized into a higher level language that had terms for the typical rectangular defects.

2. Since arbitrary regions need to be described as rectangles anyway, an obvious approach is to use them as the primitive in the specification language. The surface could be denoted as a list of rectangle, attribute value pairs. Presumably each type of attribute would be a nominal or ordinal quantity. The only phenomenon that would be difficult to capture with this representation are the splotch-like photoreceptor defects. As a result, this is the recommended way to describe regions on a surface.

5.2.5.3 Stacks

Sheets of paper, whether in the paper tray, recirculating document handler, or finisher, are arranged into stacks. Like common ordinal quantities, it is important to reason about the size of a stack; limit analysis can be used. Unlike ordinal quantities, stacks have discrete nature due to a distinguished member at the top. Only this distinguished top object is directly manipulated.

5.2.5.4 Toner

Toner is difficult to represent because one thinks about it differently at different abstraction levels. At the coarsest level, it is simply stuff: there is either enough toner to make a copy or not. At an intermediate level, toner is dry fluid that gets evenly spread over the photoreceptor. But toner doesn't stick everywhere, and this can only be explained by considering toner in a more refined way, as a bunch of particles that cling to carrier beads among other things. Aside from the issues of how to switch between these different models[10] there is the problem of choosing the appropriate representation for reasoning about the behavior of toner as it passes through the cleaner. There seem to be three possibilities:

- **Particles**
  Toner could be represented as a group of particles and discussed as a group. Since reasoning about the mechanism of the magnetic brush will require this model, an argument can be made for using it at the higher level too. Simulation could be performed by reasoning about the behavior of a single particle and then generalizing through the use of parallel aggregation [29]. Although simulation is easy, the resulting low level description is a disadvantage. People seem to use a higher level vocabulary when discussing mechanisms of this type, and explanations that continually refer to individual particles sound unnatural.

- **Fluid Individuals**
  Toner could be represented as a type of fluid that covers the various photoreceptor and roller surfaces. Axiomatization could be difficult due to the amorphous nature of liquid individuals, but the cases could be restricted since no notion of spraying, etc need be considered.

- **Property of Volumes**
  Hayes [15] advocates modeling fluids as contained substances. Rather than talking about liquids as things, he describes them in terms of the space they occupy; they are a

[10] Aggregation [30] could be used to generalize from the behavior of a single toner particle to the higher level models.

property of the space rather than individuals in their own right. Adopting this viewpoint would lead to modeling toner as a property of a surface: its toner density. This approach is wonderfully simple, but does pose problems: since toner would not be an individual, conservation of mass would not be obeyed. A different representation would be necessary to explain how toner gets used up.

It is difficult to choose between these options, but I suspect that it is best to represent toner coating as a property of the infinitesimally thick volume above surfaces.

Another issue raise by the presence of toner is the best way to represent the fact that only a limited amount of toner can stick to any finite area of surface. There seem to be two ways to model this:

- **Attraction**
  The saturation phenomenon could be derived from the definition of the electrostatic attraction process. The simplest version of this scheme would simply involve adding a precondition of the attraction process that specified that no attached toner could be present. Intuitively, the attached toner would be blocking the other toner from being attracted. A more sophisticated approach would be to define attraction as a function both of the voltage difference and the amount of toner already coating the surface.

- **Capacity**
  Instead, each unit square area of surface could have a capacity (perhaps a function of the voltage difference). The surface can hold only so before getting saturated. The capacity can be intuitively explained as the maximal thickness of the fluid layer on the surface. Saturation seems to be a natural way to explain many types of behavior, but the definition of this new parameter complicates the nature of surfaces somewhat.

Although the attraction viewpoint is more correct, it seems too sophisticated for this level of model. I choose to use the capacity model at this level and propose that it be linked to a more detailed model of attraction using particles at the level where the mechanism of the magnetic brush is explained.

5.3 Simplifications

Only components whose movement is constrained to one degree of freedom are allowed. Only planar movement is allowed. This does not eliminate devices such as differential gears, but does filter components with helical motion such as screws. Only geometrical kinematics will be considered--no dynamics. Joints between all objects except toner do not change type dynamically; in addition, no cammed joints are allowed. There is no friction, and all objects are rigid and massless.

5.4 Formalization of a Proposed Physics

Please bear with the morass of definitions in this section. Their utility will be demonstrated in the next. The physics describes *rigid members* geometrically, allows five types of contact between objects, and describes the different kinds of possible change with a process model.

5.4.1 Rigid Members

Rigid members are described geometrically for the reasons outlined in Section 5.2.2. Each member defines its own frame of reference and coordinate system.

- **Rods** Rods are long thin (zero width) sticks. They are specified by a name, their axis direction (in the parent frame of reference), and their length. Common examples include: axles, levers, and dicorotrons.

- **Cylinders** Cylinders are specified by listing a name, the name of a rod that acts as an axle (although they do not need to rotate about the axle), the distance from the end of the axle rod to the beginning of the cylinder, the radius, and the length of the cylinder. Common examples include rollers and gears. The system must know certain facts about cylinders to allow inferences that are critical to causal simulation. All points on the surface are equidistant from the axis. If the cylinder rotates, then contact (sliding and rolling processes) continues. If a cylinder rotates in contact with a fixed point, then the same points on the surface of the cylinder will continually pass under the fixed point.

- **Planes** Planes may be specified by a name, a normal axis, and the plane's length and width. Examples include scraper blades and the photoreceptor sheet.

It is necessary to refer to the surfaces of the objects mentioned above, especially in the case of planes and cylinders. One *surface description* is necessary for each attribute: charge, toner density, toner charge, etc. As described in Section 5.2.5.4 above, the surface can be described by listing a set of rectangular regions with the attribute value for the region. The reasoner must know how to wrap regions around if the space is circular and must collapse two rectangles into one if possible.

5.4.2 Contacts

The mechanical engineering theory of joints (pairs) is used as a guideline for the types of contacts allowed by this physics. Mechanical engineering recognizes four types of kinematic joints in planar machinery: hinged, sliding, rotating, and cammed contacts. I disallow cammed contacts because the motions induced become arbitrarily complex, and because they aren't necessary to describe a large class of devices. There are two types of sliding contact to distinguish between the flexible nature of materials without complicating the model. And two types of contact, which don't come from mechanical engineering but simplify description, are added: fixed contacts and adjacency.

Associated with each contact type (but not included in the definition) is certain information that is necessary for reasoning. The definition of the translation process, below, can require the knowledge that two contacts aren't moving relative to each other at a certain velocity. This can only be proved if rules about the possible degrees of freedom are provided somehow.

Each contact can be specified by listing the subsurfaces of the two objects that "touch".

- **Hinged Contact** Hinged contacts allow one degree of freedom: rotation about an axis. To specify a hinged contact, positions on the two objects must be given as well as the axis of rotation. To simplify the specification, one of the objects is restricted to be a rod and its axis is chosen as the axis of rotation. This doesn't reduce the class of devices that can be described because a (very short) dummy rod can be easily added.

- **Loose Sliding Contact** Loose sliding contacts are used whenever two objects rub gently. For example, consider the block sliding on the ground in Figure 5-1 or the joint between the TR and the CR in Figure 6-1. One view of the connection is that the two surfaces aren't really touching, but they are very close and sharing a directed face [15]. Although the two surfaces aren't touching each other, any liquid or toner that is on either surface (thus occupying an infinitesimally thin layer above the surface) must touch both.

- **Tight Sliding Contact** Tight sliding contacts are used whenever two objects scrape, i.e. rub so tightly that no attached material (e.g. toner) can make it through the joint. The distinction between the loose and tight sliding contacts is really a violation of the no-function-in-geometry principle that wouldn't be necessary if the physics allowed the specification of different substances as well as different object shapes. The distinction between the two sliding results from the bending of certain substances and the rigidity of others. Since no model is perfect or has flawless predictive power, this violation can be viewed as acceptable if it results in a simple and useful model. One would hope, however, that a reasoner would have another model, at a more detailed abstraction level, which would distinguish between substance and object, thus providing a more complete understanding of the difference between the loose and tight sliding contacts.

- **Meshed Contact**[11] When a cylinder rolls on a flat surface (or when two rotating gears mesh), there is no slippage at the point of contact. The contact is always in effect, yet the points that touch change over time--the contact is constantly being broken and reformed.

- **Fixed Contact** Fixed contacts aren't joints at all in the formal mechanical engineering sense of the word. I include them as a means for describing complex members with only a simple geometric vocabulary. The resulting complex is treated as a single rigid member.

- **Adjacency** Adjacency isn't really a type of physical contact at all, but it is necessary to describe how processes such as dicorotron charging behave. At some point it may be desirable to include some sort of distance information, but now the contact simply asserts that parts of two objects are close by.

[11] Mechanical engineers call this a rotating joint (or pair), but I adopt the term meshed contact to avoid confusion with hinged contacts or rotation processes.

5.4.3 Processes

The following informal characterization of processes in QP notation is meant to give the flavor of what is needed. I provide one filed. conditions, rather than Qp theory's preconditions and quantity conditions because of the unified treatment of nominal and linear quantities. The influences field contains not just monotonic tendencies of certain quantities, but also general inference rules that may help the system reason in manners unrelated to limit analysis.[12]

- **Rotation**
    - *Individuals:* A rod, R, and a member, A.
    - *Conditions:* R and A must be connected by a hinge joint. A point, P, on the surface of A is translating.
    - *Relations:* There is a quantity, called angular velocity, which is proportional to the speed of translation divided by the distance from R to P.
    - *Influences:* All points of A rotate. Their distance to R doesn't change. They return to their original location after time proportional to angular velocity.
- **Translation**
  The translation process really means relative motion.
    - *Individuals:* Two members, A and B.
    - *Conditions:* One of the following sentences must be true. A and B are connected by a *sliding contact (loose or tight), and at least one is rotating, and the ratio of* their rotations is not equal to the ratio of their radii (where the radius of a plane is assumed infinite). A is translating relative to C and C is not translating relative to B at the same rate.
    - *Relations:* There is a quantity, relative velocity.
    - *Influences:* The distance between points on A and B which touch at time t0 increases proportional to relative velocity. Some sort of continuity relation is also necessary.
- **Meshing**
  Intuitively, meshing is the process that happens when two interlocked gears rotate, or a wheel rolls along a surface with no slipping.
    - *Individuals:* A cylinder, C, and a member A.
    - *Conditions:* A and C are connected with a meshing joint, and A is rotating or there is relative translation between A and C.
    - *Relations:* Some combination of rotation and relative axle movement.

[12] These rules are probably not complete.

- **Scraping**
  Intuitively scraping happens when a hard blade slides tightly against a surface.

  - *Individuals:* Two members, A and B.

  - *Conditions:* A and B are joined by a tight sliding connection, and there is relative translation.

  - *Influences:* Any toner occupying the space above one of the surfaces will be transferred to the other surface..

- **Rubbing**
  Rubbing is a gentler interaction than scraping, because loose sliding is interpreted as almost touching (sharing a directed surface) rather than real contact. Toner transfer can happen if there is some attraction, but doesn't need to. Since there are no influences or relations, this process might be best considered an individual view [12].

  - *Individuals:* Two members, A and B.

  - *Conditions:* A and B are joined by a loose sliding connection, and there is relative translation.

- **Attracting**
  This happens when a new surface contacts toner. It is not necessary to keep toner stuck.

  - *Individuals:* Two members, A and B.

  - *Conditions:* A and B are rubbing. There is toner in the thin space above A. The capacity of the space above B (as a function of the amount of toner on A and B and the voltages of A and B) is not saturated.

  - *Influences:* A decrease (to zero?) in the amount of toner above A. An increase (to capacity?) in the amount of toner above B. There are deep issues here about probabilistic modeling.

- **Gravitation**
  This is weird since there is more than one degree of freedom possible when considering toner falling and sliding down planes. To be reasonable, the model of gravitation must work for rigid members as well as for toner. The intent is to model gravitation as a force that causes translation. A problem is the interaction between gravitation and the electrostatic attractive force holding toner onto a surface. Since the attractive force is not modeled as holding the toner to the surface[13], there is asymmetry in the model--the forces don't balance. Although I won't formalize the model of gravitation, it works as follows: if the surface is facing down and there is toner present and the toner isn't held on very well, then a translation down will happen.

- **Charging**
  For this to work, the system must know that if A is adjacent to B and a toner film is above

[13] Perhaps attraction should be modeled this way, but it seemed unintuitive and overly sophisticated.

B then A is adjacent to the toner.

- *Individuals:* A member, D, with a charging property, and another member, B.
- *Conditions:* D is charging. B and D are adjacent or otherwise connected.
- *Relations:* The charge of B equals that of max(B, D).

5.5 Reascning and Implementation

Unfortunately, the reasoning and implementation techniques for this language are not completely worked out. The next Section presents several unimplemented scenarios. I expect to base the implementation on Brian Williams' implementation of limit analysis (transition analysis) [31, 32, 33] with extensions to the new quantity types.

Inspiration for the geometrical reasoning will come from Stanfill's work on geometrical to behavioral (QP theory) representation transformation [25, 26] and also from work on robotics [2], specifically work on configuration space, trajectory planning, and the findpath problem [4, 3].

6. Describing a Cleaner

This Section attempts to apply the ideas of Section 5 by taking one part of a xerographic copier, the cleaner, and building a more detailed model. Section 6.1 describes a normal cleaner. Section 6.2 models a faulty cleaner with a bent cleaner blade. Section 6.3 discusses the issues with modeling other defects. Finally, Section 6.4 talks about the areas where even more detailed models could be useful.

6.1 Normal Cleaner

In normal operation a cleaner removes residual toner from the photoreceptor so that the next copy will be a clean reproduction of the new image. In Section 4.2.1 the behavior of a cleaner was described as follows.

```
(def-dp cleaning
  (when (and (= (position pr-doc-area) clean)
             (< (amount waste-toner) full))
        (:= (toner-density pr-doc-area) zero)
        (increment (amount waste-toner) (toner-density pr-doc-area))))
```

The way a cleaner implements this functionality is shown in Figure 6-1.

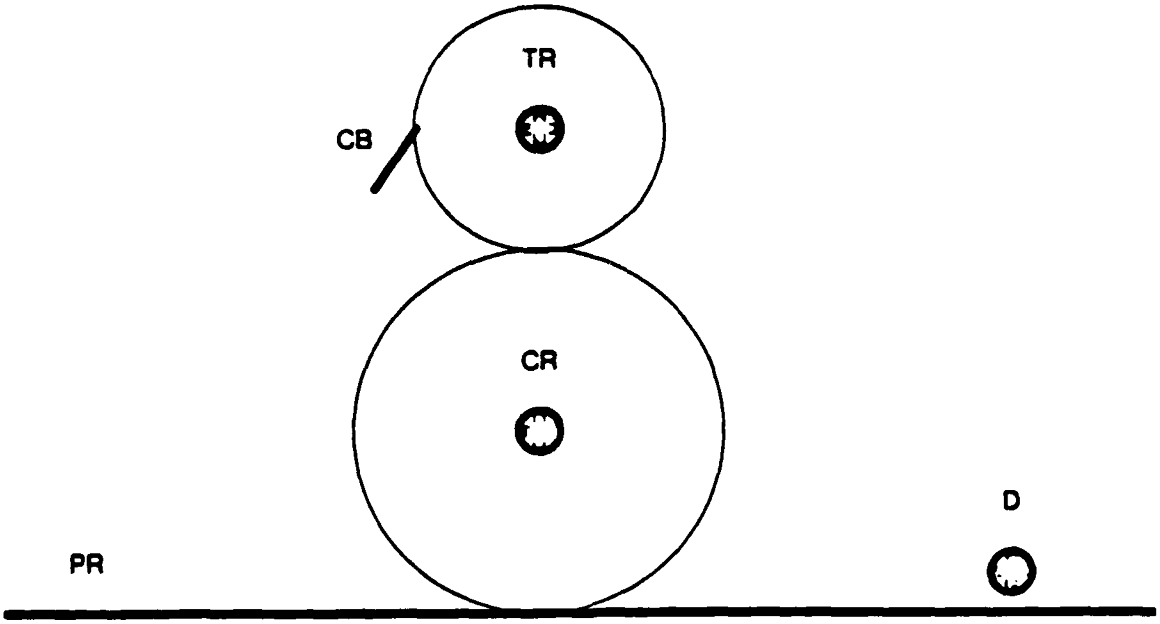


Figure 6-1: Xerographic Cleaner

Using the physics developed above, this device can be codified as follows:

```
(rod d Z len)
(rod cra Z len)
(rod tra Z len)
(cylinder cr Z large-rad len)
(cylinder tr Z small-rad len)
(plane cb (rot Z 45degrees X) small-rad len)
(plane pr Z infinite len)
```

The dicorotron, d, and the two axles, cra and tra, project into the paper along the Z axis and are of length, len. The cr (cleaner roller) and tr (toner roller) are both cylinders. The cb (cleaner blade) and pr (photoreceptor) are planar. To simplify the analysis, the pr is infinitely long rather than a circular belt as above.

```
(contact fixed (d 0) ground)
(contact fixed (cra 0) ground)
(contact fixed (tra 0) ground)
(contact hinge (cra ?z) (cr 0 0 ?z))
(contact hinge (tra ?z) (tr 0 0 ?z))
(contact loose-sliding (pr ?x ?y) ground)
(contact adjacent (d ?l) (pr point2 ?l))
(contact loose-sliding (cr 180degrees ?l) (pr point1 ?l))
(contact loose-sliding (cr 0degrees  ?l) (tr 180degrees ?l))
(contact tight-sliding (cb small-rad ?l) (tr 270degrees ?l))
(> point2 point1)
```

This just says that things are connected as the Figure shows. The cr and tr can rotate around their respective axles. The contacts between moving surfaces are all loose-slidings except for the cb/tr contact which is tight (and could thus induce a scraping process).

```
(translate pr ground X -speed1)
(rotate cr cra -omega1)
(rotate tr tra -omega2)
(voltage d +170)
(charge d)
(voltage cr -300)
(voltage tr -450)
(toned (pr (+ point2 ?pos) ?l))
```

The initial conditions specify that the pr is sliding to the left, and both the cr and tr are rotating counter clockwise. All the pr surface to the right of d is toned. This allows the following deductions:

```
pr and cr are rubbing
cr and tr are rubbing
cb and tr are scraping
line on pr under D, including toner, is charged
```

Limit analysis concludes that nothing will change until the pr line with toner hits the cr at the point of loose sliding contact. At this time the surfaces of toner and pr between points 1 and 2 are charged. There is now a rubbing between the front edge of toner and the cr as well as between the pr and cr. This allows the system to deduce that an attracting is also active in the same instant.

Limit analysis concludes that nothing will change until the toner coated cr surface reaches the line of loose sliding with the tr surface. At this time a rubbing and attracting process between the toner surface and tr surface will become active in addition to the previous processes which will continue to be active.

Limit analysis concludes that nothing will change until the toner on the tr hits the cb. At this time a scraping process will ensue, causing a transfer of the toner to the cb. Limit analysis concludes that this is the steady state.

6.2 Cleaner with Bent Cleaner Blade

Figure 6-2 shows a cleaner with a bent cb (cleaner blade).

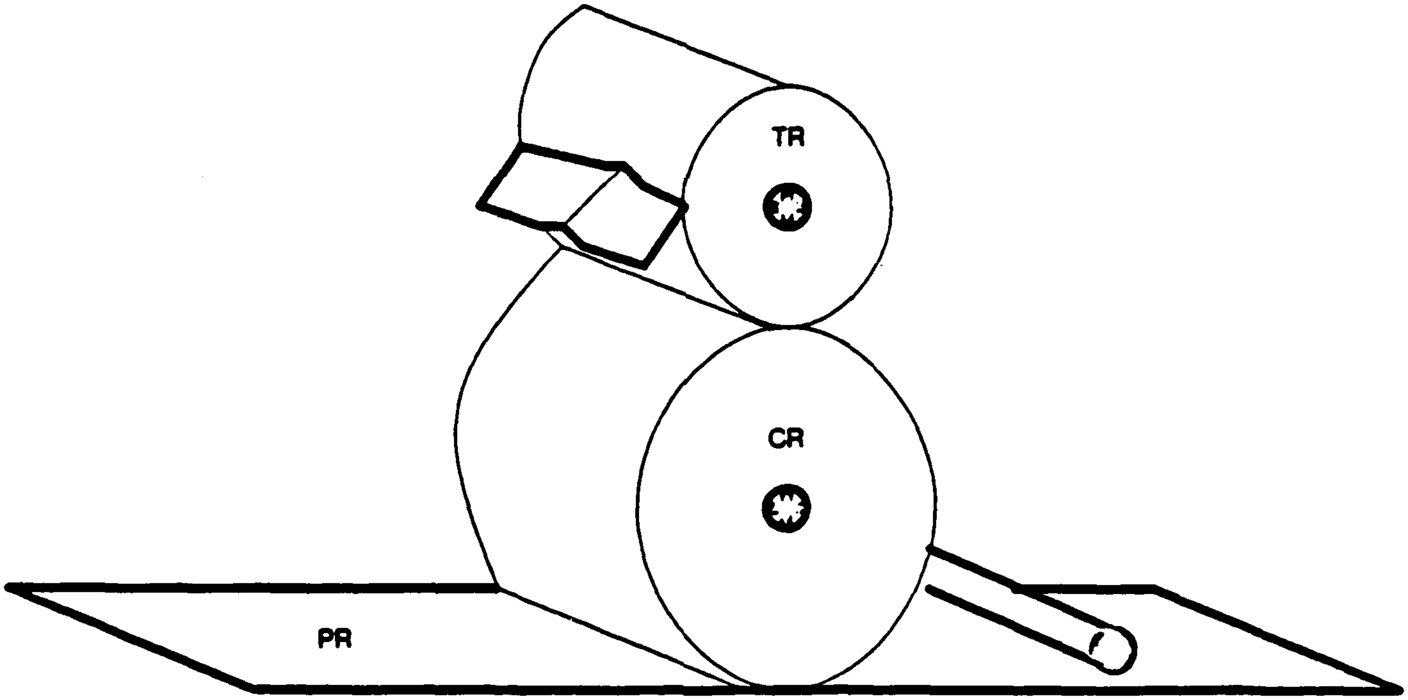


Figure 6-2: Xerographic Cleaner with Bent Blade

This fault can be described by replacing, from the example above, the line

```
(contact tight-sliding (cb small-rad ?1) (tr 270degrees ?1))
```

by the following line:

```
(and (or (< ?1 left-bend-edge)
         (> ?1 right-bend-edge))
     (contact tight-sliding (cb small-rad ?1) (tr 270degrees ?1)))
```

The reasoning proceeds exactly as above until the front line of toner coated tr surface reaches the line of tight sliding with the cb. At this point there are two rubbing processes active, on the outer sides of the two bend edges. This divides the line of contact into three segments of different activity.

Limit analysis stretches these segments into surfaces as it concludes that the process structure will not change until these new surfaces contact the cr. At this time, the previously active attraction process will become inactive because the capacity of the tr is saturated in the middle segment of contact. Although both rubbing and attracting will still occur in the outer segments, this middle segment will have just a rubbing with no transfer of toner.

Limit analysis will conclude that eventually a band of toned cr surface will make contact with the pr, and a similar capacity saturation argument will allow the system to conclude that the area on the pr to the left of the cr contact will get subdivided into three bands with the middle band toned. This is the final equilibrium state.

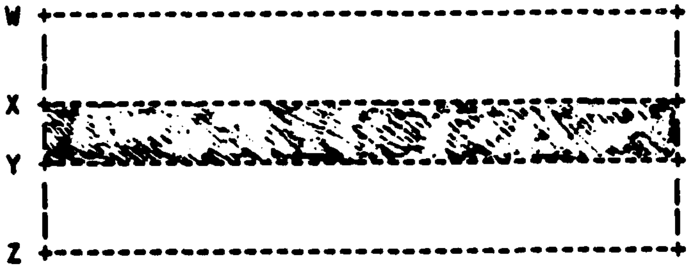


Figure 6-3: Streaked Photoreceptor Document Area

It is interesting to note that by assuming the spatial decomposition of the pr-doc-area as shown in Figure 6-3, one can summarize this defective behavior at the process level of Section 4.2.1 as shown below.

```
(def-dp cleaning    ;; bent cleaner blade
  (when (and (= (position pr-doc-area clean))
             (< (amt waste-toner) full))
        (:= (toner-density
               (sub-rect pr-doc-area ?var1 (W X))
               zero))
        (:= (toner-density
               (sub-rect pr-doc-area ?var2 (X Y))
               (sat-fn capacity
                       (toner-density
                           (sub-rect pr-doc-area ?var2 (X Y))))))
        (:= (toner-density
               (sub-rect pr-doc-area ?var3 (Y Z))
               zero))
        (increment (amount waste-toner)
                   (toner-density (sub-rect pr-doc-area
                                            ?var1 (W X))))
        (increment capacity
                   (toner-density (sub-rect pr-doc-area
                                            ?var2 (X Y))))
        (increment (amount waste-toner)
                   (toner-density (sub-rect pr-doc-area
                                            ?var3 (Y Z))))
```

6.3 Other Defects

There are numerous other interesting cleaner defects that can be modeled at this level. They include:

- Incorrect voltages on the cr and tr rollers.
- Faulty dicorotron, d. It could work only sporadically over time, distribute an uneven charge, discharge no charge, or discharge the wrong voltage.
- One of the rollers could be fixed rather than turn.

6.4 Issues for More Detailed Modeling

Although this abstraction level is appropriate for modeling many types of behavior, there are still some xerographic phenomena which require a more detailed model.

- The actual mechanism of the magnetic brush needs to be explained in terms of a particle model of toner. This model would also show why the magnetic brush is more like a sponge than a brush.
- A particle model is also necessary to show how scraping really works and to differentiate scraping from rubbing.
- The "piling up" behavior of toner given a scraper blade that contacts a moving surface at an acute angle. How this behavior relates to proper scraping.

- How capacity and saturation really work in terms of attraction and the particle model.

Modeling these phenomena will require considerable future work. I suspect that aggregation [30] will prove useful when dealing with the particle model.

7. Conclusions

It may soon be possible to build programs that dynamically generate mechanistic explanations of the behavior of complex mechanical devices. Before this can be achieved, however, the work on qualitative reasoning must be extended to deal with more expressive physics, spatial reasoning, and complex systems.

This paper has explored the issues which need to be addressed to accomplish these aims, and has tried to enumerate the possible approaches to solving the issues. I feel that all reasonable solutions must share the following properties:

- The use of multiple, interacting, and inconsistent models as a method for reasoning about different aspects of complex systems from different viewpoints. These models must be extended in some manner to delineate their own limitations.
- Some qualitative kinematics and dynamics based on geometrical primitives and reasoning. If a language violates the no-function-in-geometry principle, then it will no longer allow easy composition of physical descriptions.
- If both qualitative and quantitative information are available, then the reasoning engine should use both. A representation like that of Simmons' quantity lattice [23] is recommended.

Clearly, there is a great need for more work on these problems. The following list contains a few suggestions.

- A reasoner which uses the physics presented in Section 5 should be implemented.
- The physics should be extended to handle full dynamics and dynamically changing joints.
- A detailed model of the magnetic brush should be developed and the different models tied together.

Acknowledgments

Many thanks for comments, criticism, help and interesting discussions with Steve Bagley, Danny Bobrow, John Seely Brown, Dave Chapman, Johan de Kleer, Ken Forbus, Jim Greeno, Pat Hayes, Gregor Kiczales, Tom Moran, Dan Russell, and Brian Williams. Dave Chapman's fast-turnaround-draft-commentary-service was especially useful.

References

[1]
Allen, J.
Maintaining Knowledge About Temporal Intervals.
CACM 26(11):832-843. 1983.

[2]
Brady, M., Hollerbach, J., Johnson, T., Lozano-Perez, T., Mason, M.
Robot Motion: Planning and Control.
MIT Press, 1982.

[3]
Brooks, R., Lozano-Perez, T.
A Subdivision Algorithm in Configuration Space for Findpath with Rotation.
IEEE Transactions on Systems, Man, and Cybernetics SMC-15(2), March/April, 1985.

[4]
Canny, J.
Collision-Detection for Moving Polyhedra.
AIM 806, MIT AI Lab, October, 1984.

[5]
Chapman, D.
Cognitive Cliches.
In Publication, MIT AI Lab, 1985.

[6]
Collins, A., Gentner, D.
Multiple Models of Evaporation Processes.
Technical Report, BBN Labs, July, 1983.

[7]
de Kleer, J.
Qualitative and Quantitative Knowledge in Classical Mechanics.
AI-TR-352, MIT AI Lab, December, 1975.

[8]
de Kleer, J., Brown, J.
Assumptions and Ambiguities in Mechanistic Mental Models.
Cognitive and Instructional Sciences Series CIS-9, Xerox PARC, March, 1982.

[9]
de Kleer, J., Brown, J.
A Qualitative Physics Based on Confluences.
Artificial Intelligence , December, 1984.

[10]
de Kleer, J., Bobrow, D.
Qualitative Reasoning with Higher-Order Derivatives.
In *Proceedings of the National Conference on Artificial Intelligence*. 1984.

[11]

Dean, T.
Temporal Reasoning Involving Counterfactuals and Disjunctions.
In *Proceedings of the Ninth IJCAI*. 1985.

[12]

Forbus, K.
Qualitative Process Theory.
Artificial Intelligence , December, 1984.

[13]

Forbus, K.
Qualitative Process Theory.
AI-TR-789, MIT AI Lab. October, 1984.

[14]

Hayes, P.
The Second Naive Physics Manefesto.
In Hobbs, J., Moore, R., editors, *Formal Theories of the Commonsense World*. Ablex, 1985.

[15]

Hayes, P.
Naive Physics I: Ontology for Liquids.
In Hobbs, J., Moore, R., editors, *Formal Theories of the Commonsense World*. Ablex, 1985.

[16]

Hobbs, J.
Granularity.
In *Proceedings of the Ninth IJCAI*. 1985.

[17]

Kramer, G.
Representing and Reasoning About Designs.
IFIP Working Paper, Schlumberger Palo Alto Research, July, 1985.

[18]

Kuipers, B.
Getting the Envisionment Right.
In *Proceedings of the National Conference on Artificial Intelligence*. August, 1982.

[19]

Kuipers, B.
Commonsense Reasoning about Causality: Deriving Behavior from Structure.
Artificial Intelligence , December, 1984.

[20]

McDermott, D.
A Temporal Logic for Reasoning About Processes and Plans.
Cognitive Science (6):101-155, April, 1982.

[21]

Patil, R.
Causal Representation of Patient Illness for Electrolyte and Acid-Base Diagnosis.
TR-267, MIT Laboratory for Computer Science. October, 1981.

[22]
Simmons, R.
Representing and Reasoning About Change in Geologic Interpretation.
AI-TR-749, MIT AI Lab, December, 1983.

[23]
Simmons, R., Davis, R.
Representing and Reasoning about Change.
AI Memo 702, MIT AI Lab, 1983.

[24]
Singh, N.
Exploiting Design Morphology to Manage Complexity.
PhD Thesis, Stanford University, September, 1985.

[25]
Stanfill, C.
Form and Function: The Representation of Machines.
TR-1347, Computer Science Dept., University of Maryland, November, 1983.

[26]
Stanfill, C.
MACK, a Program which Deduces the Behavior of Machines from their Forms.
ACM SIGArt (93), July, 1985.

[27]
Stevens, A., et. al.
STEAMER: Advanced Computer Aided Instruction in Propulsion Engineering.
Technical Report 4702, Bolt Beranek and Newman Inc, July, 1981.

[28]
Weld, D.
Explaining Complex Engineered Devices.
Technical Report 5489, Bolt Beranek and Newman Inc, November, 1983.

[29]
Weld, D.
Switching Between Discrete and Continuous Process Models to Predict Genetic Activity.
AI-TR-793, MIT AI Lab, October, 1984.

[30]
Weld, D.
Combining Discrete and Continuous Process Models.
In *Proceedings of the Ninth IJCAI.* 1985.

[31]
Williams, B.
Qualitative Analysis of MOS Circuits.
Artificial Intelligence , December, 1984.

[32]
Williams, B.
Qualitative Analysis of MOS Circuits.
AI-TR-767, MIT AI Lab, July, 1984.

[33]
Williams, B.
The Role of Continuity in a Qualitative Physics.
In *Proceedings of the National Conference on Artificial Intelligence*. August, 1984.

Diagnostic Mechanism Modeling

Arthur M. Farley*
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403

Abstract

In this report, we propose a scheme for the modeling of complex mechanical and electronic systems. The models are designed specifically to support diagnostic reasoning. Diagnostic reasoning is a basic element of the troubleshooting process whereby an incorrectly performing system is restored to normal operation. Diagnostic reasoning has the goal of determining a minimal set of component faults that can account for observed symptoms. Our modeling scheme reflects two general principles of diagnostic reasoning: model variables assume values relative to normal levels (the "normality principle") and these values are propogated to account for the production of a single output (the "single product principle"). These two principles yield significant simplifications in the value spaces of model variables and in model structure. We illustrate our approach to diagnostic modeling and reasoning with examples from xerography.

Subject Categories: knowledge representation, reasoning, qualitative modeling, diagnosis.

* Research performed while author on sabbatical leave at Xerox PARC, Palo Alto, CA; research supported by ARI Contract #MDA 903-83-C-0189.

Introduction

We have come to depend increasingly upon complex mechanical and electronic systems in the performance of our daily tasks. Both the home and workplace have become filled with these tools, ranging from automatic appliances in the kitchen and entertainment systems in the livingroom to xerographic copiers and computer networks in business environments. With this expansion of use, problems associated with the maintenance of such systems have become critical. Due to the rapid turnover in products and personnel, there is a perennial shortage of people who are knowledgable about the troubleshooting of these systems. This situation can lead to long repair delays, as well as expensive maintenance contracts or service calls.

In our research, we have been investigating possible roles for computer-based modeling and simulation in the maintenance of complex systems. How can the general approaches and tools of modeling and simulation be specialized for the delivery of efficient, effective troubleshooting? What are important properties that models of complex systems must have if they are to be relevant to the task of fault diagnosis? We want models that can assist us either directly, by solving particular maintenance problems when they arise, or indirectly, as preliminary training tools for repair personnel or system users.

Troubleshooting refers to the process whereby a complex electronic or mechanical system that is exhibiting incorrect behavior (i.e., behavior not in accordance with its functional specifications) is restored to correct behavior. We can see that troubleshooting is an instance of *problem solving*, which we define to be the satisfaction of a goal state through elimination of differences between a desired goal state and a current, undesirable state. Troubleshooting is an instance of indirect problem solving, as the difference to be eliminated, that of incorrect system behavior, can not be affected directly. Rather, we must locate a source of error within the system and replace or adjust the believed culprit.

We find it advantageous to discuss problem solving within the framework of a problem space. A *problem space* consists of a set of states, representing situations in the task environment, and operators, representing actions we could take within that environment, each mapping one state to another (Newell and Simon, 1973). We have previously characterized troubleshooting (Farley, 1985) as taking place across three, interrelated problem spaces, as depicted in Figure 1. *Observation Space* is concerned with acquiring relevant observations of system behavior and interpreting these as symptoms. *Diagnosis Space* is concerned with mapping symptoms into beliefs regarding possible faults responsible for the observed symptoms. In Diagnosis Space we also determine what further symptoms may be of interest by noting those that are as yet unobserved and are also associated with possible faults. Evaluations as to the existence of these symptoms become problems to be solved in Observation Space, through subsequent observation acquisition and interpretation. Finally, *Repair Space* is where adjustment and replacement plans are determined that can eliminate possible faults.

Each problem space has its own set of operators, ranging from those elements of test and repair procedures from Observation and Repair Space that actually measure or manipulate physical components of the system to the more cognitive operators from Diagnosis Space that update degrees of belief in possible faults and generate new, relevant symptoms for evaluation. The different problem spaces do share certain aspects of their state representations. These shared elements form the bridges between the problem spaces, allowing problem solving to make transitions between the different realms when appropriate. Effective problem solving in each of the problem spaces requires its own specialized knowledge. This segmentation of applicable knowledge is one motivation for our characterization of troubleshooting in terms of multiple problem spaces.

The representation of knowledge for Diagnosis Space seems particularly well-suited to our modeling and simulation perspective. Within Diagnosis Space we are concerned with transforming symptoms into beliefs regarding the operational state of system components and from those beliefs to expected symptoms. It is in this reasoning that we come closest to simulating system activity, imagining the propagation of local effects of component faults to observable features in system behavior. Let us consider an example to illustrate our point, by attempting to account for light copies produced by a defective xerographic copier.

In our consideration of copy production, we focus on the xerographic subsystem, as summarized in Figure 2. The discussion below will remain at a qualitative level, ignoring certain details of the physical processes involved in xerography. There are seven stages that occur in the xerographic process, each realized in a copier by one or more components; representative components for the various stages are indicated below the schematic. A central element of the process, acted upon by components from six of the stages, is the photoreceptor. A photoreceptor has the unique property that it will hold a local, electrical charge until it is exposed to light; importantly, only those areas exposed to light are then grounded and lose their charge.

During the xerographic process, a photoreceptor is first given a uniform charge by being passed through a negative field emitted from a dicorotron. It is then exposed to a light image, patterned according to the copy to be made. The charged image on the photoreptor is then brought into contact with small, charged particles of plastic, which adhere to it during the development stage. The pattern of charged plastic on the photoreceptor is then brought near a paper sheet in the presence of another, stronger field, at which time most of the 'particles transfer to the sheet. After the transfer stage, the sheet is passed through hot rollers, fusing the plastic to the paper and producing the final copy. The remaining charge and plastic are removed from the photoreceptor at the cleaning stage; the photoreceptor is then ready to begin the process anew with charging.

Suppose we observe copies being produced which are too light. How can we remedy this situation? As noted, there is no direct way to eliminate the observed difference -- no action in the world which is "make darker copies". During troubleshooting, we must alter or replace a component to correct the performance of the system as a whole. Determining which component to affect is a task carried out primarily in Diagnosis Space, by accounting for observed symptoms in terms of faulty components. There are several possible accounts for light copies; we discuss three.

One account would have the charge field emitted by the dicorotron at the transfer stage being too low. This would result in too many plastic particles remaining on the photoreceptor. Thus, fewer particles would be transfered and then fused; the final copy would be light. Another account would have the charge field emitted at charging being low. This would result in an insufficient number of plastic particles adhering to the photoreceptor during development, which would propagate as too few particles being transfered and eventually fused. Finally, suppose the photoreceptor has become fatigued, no longer holding a charge as it should. An account similar to the charge dicorotron story would result. As all photoreceptors fatigue with use, more advanced copiers attempt to compensate by automatically increasing the field emitted at charging as fatigue is detected, thereby prolonging adequate copy contrast.

The three accounts above illustrate our notion of diagnostic reasoning based upon a model of a complex system. A fault in a component is seen to give rise to a local effect, which is then propagated by a form of simulation through other components to produce predictions as to observable symptoms. Such a model should likewise support a form of backward reasoning or reverse simulation, allowing us to determine a set of faults whose local effects could propagate to produce observed symptoms. Our goal here is to present a scheme for the definition and use of system models sufficient to support these simulation-based forms of diagnostic reasoning.

In the next section, we discuss general properties of diagnostic models, both requirements and possible simplifications. We follow this by a definition of our modeling scheme, based upon the properties discussed. We then define a process of diagnostic reasoning that makes use of our diagnostic models. We illustrate our notions with a diagnostic model of aspects of the xerographic subsystem of a copier as discussed above. We conclude with a discussion of the possible application of our modeling and reasoning schemes to actual troubleshooting situations and field personnel education.

Diagnostic Modeling

By *diagnostic model* we will mean a model that can serve as basis for diagnostic reasoning. A diagnostic model must achieve significantly high degrees of architectural fidelity and functional adequacy. *Architectural fidelity* is a measure of the extent to which a model incorporates structural and process elements from the design and operation of a complex system. A diagnostic model must directly reflect a system's structure by representing relevant components and their interconnections. This is required as diagnostic reasoning must isolate faults to elements within this architecture. As for process, we wish to generate causal accounts of how symptoms could arise from possible faults and propagation of their effects through neighboring system components. In other words, we want to represent the performance of a complex system in terms of the composition of locally-determined, component behaviors. One aspect of this goal is the *locality principle* (deKleer and Brown, 1984), which requires that a component's behavior be modeled solely in terms of values associated with its own state and connection interfaces. The composition of component behaviors will reflect not only the structure of component interconnections but also a specification of system process that captures the temporal relationships among component behaviors (e.g., sequential, concurrent, simultaneous).

Recent work on qualitative modeling of physical systems (Bobrow, 1985) has made an important distinction between system function and system behavior. *System function* is represented in terms of intended system purpose, while *system behavior* is neutral with respect to user goals, being just what occurs. Diagnostic reasoning, as part of a troubleshooting process that attempts to restore a system's behavior to its specifications, is heavily function-oriented. A system's behavior can not be judged incorrect unless it is evaluated relative to the system's intended purpose. For example, a copier always generates behavior consistent with the state of its components; it is judged to be performing improperly when the copies it produces (i.e., its intended purpose) are not adequate. We use the term *system performance* to refer to a system's behavior as evaluated relative its function.

Through the composition of component behaviors, a diagnostic model must transform a representation of component behaviors into a representation of system function. This transformation is an explicit aspect of our diagnostic modeling scheme and is a prerequisite for providing the second necessary property of diagnostic models, functional adequacy. *Functional adequacy* is a measure of the extent to which a diagnostic model represents the interactions whereby a system's function is realized through the composition of its component behaviors.

As part of our group's effort toward understanding the nature of diagnostic knowledge and reasoning, several protocols were collected of expert field personnel as they troubleshot faulted copiers. Two general features of these protocols have served as bases for principles that underlie our *modeling approach. One is the normality principle - that values attributed* to component behaviors and propagated though a diagnostic model are represented relative to normal levels. In our protocols, precise values tended to be discussed only when what would be normal must be evaluated in terms of observable system values. Such discussions reflect knowledge and problem solving in Observation Space. The normality principle significantly influences our design of diagnostic models, as symptoms propagated within such models will be represented by descriptive variables that may assume values at, above, or below normal levels.

The second principle underlying our modeling approach is the *single product principle.* In our protocols, we find that most discussions revolve around how one incorrect output, such as a too light copy, could have been produced. The single product principle allows us to adopt a production line perspective toward system process. A *production line* is a system that processes a set of inputs, possibly concurrently, through a succession of stages to produce an eventual output. Not only does this perspective eliminate feedback interactions that can be very difficult to model and understand, it also eliminates continuous time as a direct concern of the model. A trace of the behavior of a system can be represented as an alternating sequence of distinct product states and system operations. This trace is considerably easier to represent and interpret than would be a time-based account. It also closely resembles the traditional problem space representation of plans used in artificial intelligence models of problem solving (Newell and Simon, 1973).

A Diagnostic Modeling Scheme

Now that we have discussed several general principles of diagnostic modeling, it is time to present our diagnostic modeling scheme. The five basic types of elements in our scheme are the PART, COMPONENT, STAGE, LINE, and PRODUCT. The PART and COMPONENT types are used to represent basic architectural elements of a system, including how their behaviors are affected by possible fault states. The STAGE, LINE, and PRODUCT types are used to interconnect these elements into a production line, thereby capturing basic functional aspects of a system. The PRODUCT type is used to represent the entities operated on by the system; the STAGE and LINE represent how PARTs and COMPONENTs are brought into contact with PRODUCTs to realize a system's function. It is important to note that the interconnections between COMPONENTs of a diagnostic model are distinctly functional in nature, while still reflecting aspects of physical structure.

A PART is used to represent a primitive element of a system's architecture, one not broken down into further sub-components. A PART's behavior is represented by a set of Function Variables and associated Fault States and Causes. The Function Variables represent a PART's contribution to the behavior of its environment, being values that are exported to other elements of a model. The Fault States and Causes represent diagnostic knowledge about a PART. The Fault States indicate the known ways that a PART may fail. The Causes represent the effects such failures have upon values of a PART's Function Variables, causing one or more Function Variables to assume values that differ from normal.

The normality principle discussed above allows us to simplify the value space associated with a Function Variable. The value space consists of only three values -- UP, NORMAL, and DOWN. The value space of the Fault States associated with a PART or COMPONENT consists of a set of symbolic names representing possible, incorrect operational states. Causes consists of a set of associations between elements of Faults States and abnormal values of Function Variables. As such, our models of PARTs and COMPONENTs satisfy the locality principle discussed previously. The effects of any Fault State are represented solely in terms of deviations from normal in the values of local Function Variables. Any effects upon eventual system performance must be realized by propagation (simulation) through other aspects of a diagnostic model.

As an example, two PART type elements - a coronode and a bias shield - model a negative dicorotron component of a xerographic system. Figure 4 presents our diagnostic model of the coronode PART of a negative dicorotron. A negative dicorotron consists of a coronode, being a wire in a glass tube that creates a surrounding field of ionized air, and a bias shield. In a negative dicorotron the bias shield is negatively charged, so that the dicorotron emits a negative charge field. The coronode can fail to be correctly operating in several ways, as indicated by the elements of Fault States. Causes indicates that if a coronode assumes the Fault State of DIRTY, then the level of the charge emitted will be DOWN.

A COMPONENT corresponds to a physical element of a system that we choose to represent as a composite of subcomponents; a subcomponent could be either a PART or COMPONENT. Figure 5 presents our model of a negative dicorotron, represented as a COMPONENT having a coronode and a negative bias shield as its subcomponents. A COMPONENT's behavior is represented in terms of a set of Function Variables, whose values are seen to be related to those associated with Function Variables of its subcomponents. These relations are of two types: P (propagate) and I (invert). These relations correspond directly to the M+ and M- functions discussed by Kuipers (Kuipers, 1985). An M+ relation between two function variables indicates that their values are directly related mathematically. For diagnostic purposes, P is the M+ relation, which represents the direct propagation of an abnormal or normal value. As an example, our model indicates that if *Ch* of a coronode is DOWN, then *Nf* of the dicorotron is also DOWN. The I relation corresponds to the M- indirect functional relationship; I inverts UP to DOWN and vice versa, assuming other relevant variables are NORMAL, thus propagating the inverse abnormality. Faults at the COMPONENT level represent possible incorrect interactions among subcomponents. For example, if the bias shield and coronode are misaligned in a dicorotron, the evenness of the charge field emitted by the dicorotron will be DOWN.

Our model of the behavior of a negative dicorotron is already represented in functionally relevant terms according to our choice of Function Variables. However, it is independent of the dicorotron's actual use within the copying process. The Function Variables anticipate uses of a PART, but do not directly indicate its function in a given system. Nothing in our dicorotron model mentions a copy or other elements of the copying process. Due to our adherence to the locality principle, the dicorotron model still could be inserted anywhere in our model of the xerographic system. The model of a PART or COMPONENT is incorporated into a system model by becoming a member of the Components aspect of a STAGE. A STAGE represents the bringing of a set of input PRODUCTs into contact with one or more PARTs or COMPONENTs to produce a set of output PRODUCTs. The STAGE and PRODUCT are the basic elements in our representation of system function in a diagnostic model.

PRODUCTs represent the things being acted upon by system components. These correspond to inputs, sub-assemblies, and eventual outputs of a system. A PRODUCT is represented in a manner analogous to a PART, having Function Variables, Fault States, and Cause aspects. This reflects the notion that a PRODUCT acts like a COMPONENT at a STAGE to the extent that a Fault State associated with an input PRODUCT may adversely affect the output PRODUCT of the STAGE. In addition, a PRODUCT has an associated set of Effect Variables, representing those aspects of the PRODUCT that are altered by operations at one or more STAGEs. Figures 6 and 7 present our representation of the photoreceptor PRODUCT (PR) and the Charging STAGE of the xerographic system.

A STAGE is represented in terms of Inputs and Outputs, both being sets of PRODUCTs, and a set of PARTs and COMPONENTs that generate the STAGE's function. Using the P and I relations, a STAGE represents values of the Effect Variables of its Outputs as functions of values of the Effect and Function Variables of its Inputs and the Function Variables associated with its PARTs and COMPONENTs. For example, according to our model of the Charging STAGE, the level of charge held by a photoreceptor (PR) after charging is a direct function of the capacitance of the input PR and the level of the charge field emitted by the dicorotron. A fatigued photoreceptor results in a capacitance that is DOWN illustrating our notion that a PRODUCT can act like a COMPONENT during the interaction occurring at a STAGE; a faulty input can adversely affect outputs even when system components are functioning correctly.

To complete a diagnostic model, we must compose the individual functions represented by the various STAGEs into the overall function of the system. This is accomplished through definition of a LINE, capturing the process of a complex system from the production line perspective. A production line can be modeled by a directed acyclic graph, leading from an INPUT, an external source of PRODUCTs, through various STAGEs, to an OUTPUT of final PRODUCTs. A LINE description associates the Inputs of one STAGE with the Outputs of other, prior STAGEs, while introducing the implicit, external INPUT and OUTPUT stages to complete the model. Figure 8 presents our LINE representation of the xerographic system of a copier. Each element of a LINE indicates the type and source of a STAGE's Inputs.

Diagnostic Reasoning

Given a diagnostic model, how can we use it to realize the goals of diagnostic problem solving? We remember these goals are to determine a set of possible faults consistent with observed abnormalities in system outputs, to predict further symptoms implied by one or more of the possible faults, and to generate causal accounts which indicate how the local effects of possible faults could have propagated through the stages of a system's process to yield observed and predicted symptoms of the system's performance. To account for an observed abnormality in system output (i.e., an UP or DOWN value of an Effect Variable associated with an OUTPUT PRODUCT), we propagate values "backward" through STAGEs of the LINE according to the diagnostic reasoning rules of Figure 9. The propagation process terminates at a possible fault or when there is no further relation available. As the LINE is acyclic and each component is represented as a tree of subcomponents, the process will halt with a finite set of causal stories. Figure 10 presents a subset of the accounts for the observation of light copy (i.e., the amount of toner on the image at output is DOWN), being ways a charge dicorotron could be involved in producing this symptom according to our previously presented model.

Several comments about this reasoning process and its results are in order. First, much information is lost due to the simplified value spaces and relations. This can lead to conflicting predictions during propagation. If the value of function variable *v* is equal to the sum of values of *x* and *y* when represented quantitatively, we have P(*v*, *x*) and P(*v*, *y*) when this relation is represented diagnostically. If propagation has produced the possibilities that *x* is UP and *y* is DOWN, then we could predict the possibilities that *v* is UP and *v* is DOWN. These predictions would be found in independent causal stories. We argue that this is not completely inappropriate, as it represents the one-track focusing of diagnostic reasoning often observed in troubleshooting protocols. Another consequence of information loss in a diagnostic model is the potentially large number of faults generated to account for a given symptom. From a tutorial perspective this may not be a disadvantage, as a wide range of possible faults are described; but from a field application perspective, something must be done to reduce or order the possible faults.

One asssumption that is often made during troubleshooting is that at most one fault is present in the system at any time. This *single fault assumption* is useful for reasonably reliable systems and allows us to better prune the set of possible faults when several symptoms have been observed. We merge causal stories accounting for observed symptoms that begin with a common fault and do not contain direct conflicts regarding Function Variable values. The resultant causal story is a tree starting at the fault and ending in leaves that are observed abnormalities. This contrasts with the set of stories for an individual symptom, which is a tree having leaves that are possible faults and the symptom as root.

Discussion

Diagnostic reasoning has long been a topic of research in artificial intelligence. Many expert systems have had as their goal the effective diagnosis of either diseases in biological systems or faults in electronic or mechanical systems. Until recently, most of these expert systems adopted an *experience-based approach* to the representation of diagnostic knowledge. Diagnostic associations between symptoms and possible faults are represented directly, reflecting "compilation" of an expert's experience as to the co-occurrence of a fault or disease and the observation of particular symptoms. Research based upon this approach has resulted in several effective strategies for diagnosis (Clancey,1984; Reggia, et.al., 1978; Gomez and Chandrasekaran 1981).

Our scheme is representative of a *model-based approach* to diagnostic reasoning, whereby the associations between symptoms and faults are represented only indirectly and must be derived by propagation of the local effects of faults through behaviors of system components to observable symptoms. Kuipers and Kassirer (1984) characterize the differences between the experience- and model-based approaches in medicine. Genesereth (1985) reports on a model-based approach to the diagnosis of faults in complex digital computer circuits. His models are referred to as design descriptions, indicating their high degree of architectural fidelity. Genesereth presents techniques for manipulating the clauses of a design description to determine possible faults or generate symptom-related tests of interest. Genesereth's models propagate actual values rather than values relative to normal; however, the domain of digital circuits already limits the value space of descriptive variable to be {0, 1}.

Another feature discuss by Genesereth, also emphasized by Davis (1985), is the hierarchic nature of complex systems. Their hierarchical models differ from the hierarchy included in our COMPONENT models; theirs is a hierarchy over STAGEs that are represented as process lines. This suggests how our modeling scheme could be extended to realize a more general form of hierarchical mechanism model. We introduce the notion of a STAGE-LINE, which has its functional aspect represented as a LINE rather than instantaneously acting COMPONENTs. The Inputs and Outputs of a STAGE-LINE correspond to those of the LINE implementing its function.

An interesting point arises in Davis' discussion of the determination of bridge faults in computer circuits, where a "pctential connection" exists between two circuits due to their spatial proximity. A bridge fault can then create an actual connection between the two, effectively altering the architecture of the device (i.e., a different LINE is formed). As currently discussed, the modeling scheme we propose could not determine such errors, since our model of a system's architecture is fixed. The diagnosis of non-functional faults is a difficult issue to address. We would argue that not being able to easily diagnose faults outside the functional architecture of a mechanism is not a serious shortcoming of our modeling scheme. Rather, our scheme is defined so as to take advantage of the simplifications observed in normal diagnostic reasoning. These simplifications are heuristic in nature, but allow most problems to be solved with reasonable, appropriately focused effort. Only when a functionally-oriented diagnostic model fails, would one want to resort to more naive, physical or behavioral models. Typically, non-functional faults go undiagnosed or become part of an experience-based addendum to a model-based, diagnostic model. This is the case for bridge faults in circuits and most commonly occurring sets of multiple faults.

Conclusion

In this paper we have presented a general scheme for the diagnostic modeling of complex mechanisms. Basic elements of the associated reasoning methods for single and multiple symptoms have been implemented as part of a prototype system. The causal narratives presented in Figure 10 represent an example of this system's output. Features of the modeling approach and reasoning algorithm are being incorporated into a system that will assist with the training of troubleshooting personnel. A diagnostic model of the xerographic system will be used to explain why certain tests are being performed, showing how expectations of symptom observations follow from propagation of local effects of likely faults through system components to observable values. Similarly, the model can indicate why a particular fault is suspected on the basis of previously observed symptoms. The instructional system is enhanced by graphical displays of stages of the xerographic process. Educational applications of qualitative mechanism models hold much promise for improving instruction about operational and diagnostic procedures associated with complex systems.

In closing, we reemphasize the maxim that models are created for a purpose and are limited in their capability according to that purpose.

Diagnostic models reflect their purpose in the simplification of value spaces associated with model variables and in the representation of model architectures as acyclic process lines.

References

Bobrow, D.G. ed. (1985), *Qualitative Reasoning about Physical Systems*, MIT Press: Cambridge, MA.

Clancy, W.J. (1984), "Classification problem solving", in *Proceedings of the National Conference of the American Association for Artificial Intelligence*; Austin, Texas; August, 1984; p49-55.

Davis, R. (1985), "Diagnostic reasoning based on structure and behavior", in Bobrow (1985), p347-410.

DeKleer, J. and Brown, J.S., "A qualitative physics based on confluences", in Bobrow (1985), p7-84.

Farley, A.M. (1985), "A general model of troubleshooting and its application to computer support", in *Proceedings of the Fifth International Workshop on Expert Systems and their Application*; Avignon, Fr., April, 1985, p489-503.

Genesereth, M. R. (1985), "The use of design descriptions in automated diagnosis", in Bobrow (1985), p411-436.

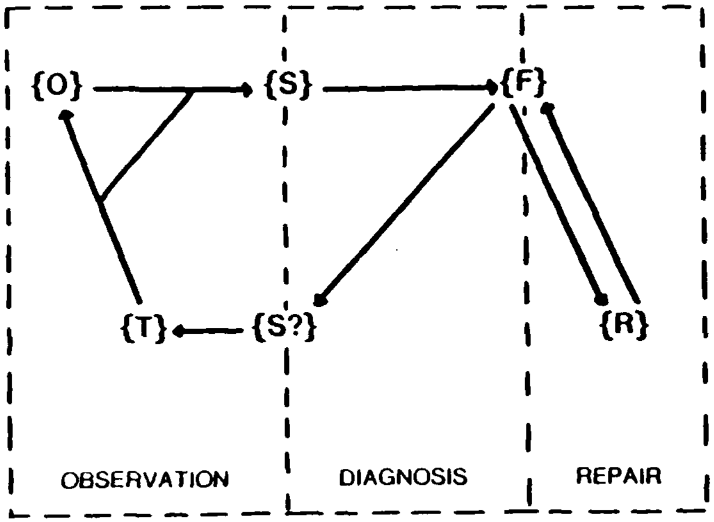
Gomez, F. and Chandrasekeran, B. (1981), "Knowledge organization and distribution for medical diagnosis", *I.E.E.E. Transactions on Systems, Man, and Cybernetics*, SMC-11(1), p34-42.

Kuipers, B. (1985), "The limits of qualitative simulation", *Proceedings of IJCAI-85*, Los Angeles, CA, August, 1985.

Kuipers, B. and Kassirer, J.P., (1984), "Causal reasoning in medicine: analysis of a protocol", *Cognitive Science*, 8, p363-385.

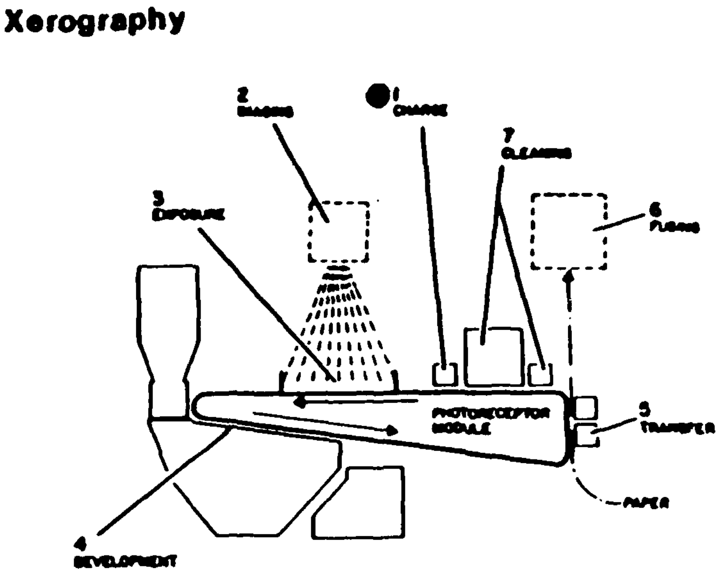
Newell, A. and Simon, H.A. (1973), *Human Problem Solving*, Prentice-Hall: Englewood Cliffs, NJ.

Reggia, J.A., Nau, D.S., and Wang, P.Y. (1978), "Diagnostic expert systems based on a set covering model", *International Journal of Man-Machine Studies*, 19, p437-460.



{O}	observations	{S}	symptoms
{F}	faults	{T}	test procedures
{S?}	expected symptoms	{R}	repair procedures

Figure 1 A General Model of Troubleshooting



1. Charging: Charge Dicorotron

2. Imaging: Image Light, Lens, Shutter

3. Exposing: Edge Erasers, Interdocument Lamps

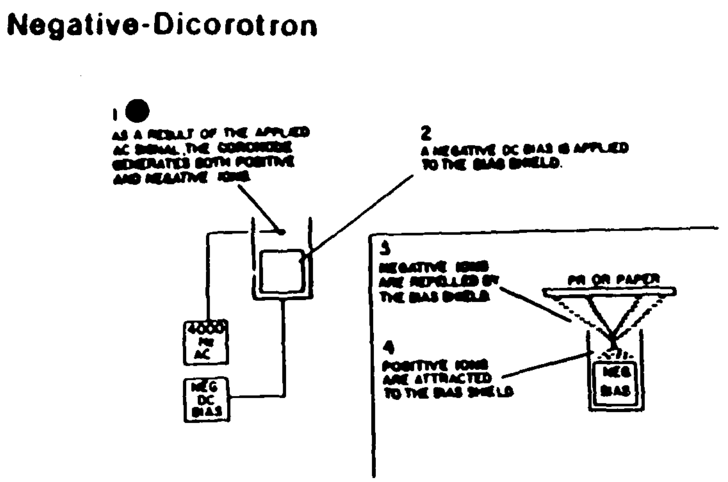
4. Developing: Toner Dispenser, Developer Housing

5. Transfer: Pre-transfer Light, Transfer Dicorotron

6. Fusing: Fusing Roller

7. Cleaning: Pre-clean Dicorotron, Cleaner Housing

Figure 2 The Xerographic Process


Negative-Dicorotron
1
AS A RESULT OF THE APPLIED AC SIGNAL, THE CORONODE GENERATES BOTH POSITIVE AND NEGATIVE IONS
2
A NEGATIVE DC BIAS IS APPLIED TO THE BIAS SHIELD.
3
NEGATIVE IONS ARE REPELLED BY THE BIAS SHIELD
4
POSITIVE IONS ARE ATTRACTED TO THE BIAS SHIELD
4000 Hz AC
NEG DC BIAS
PR OR PAPER
NEG BIAS

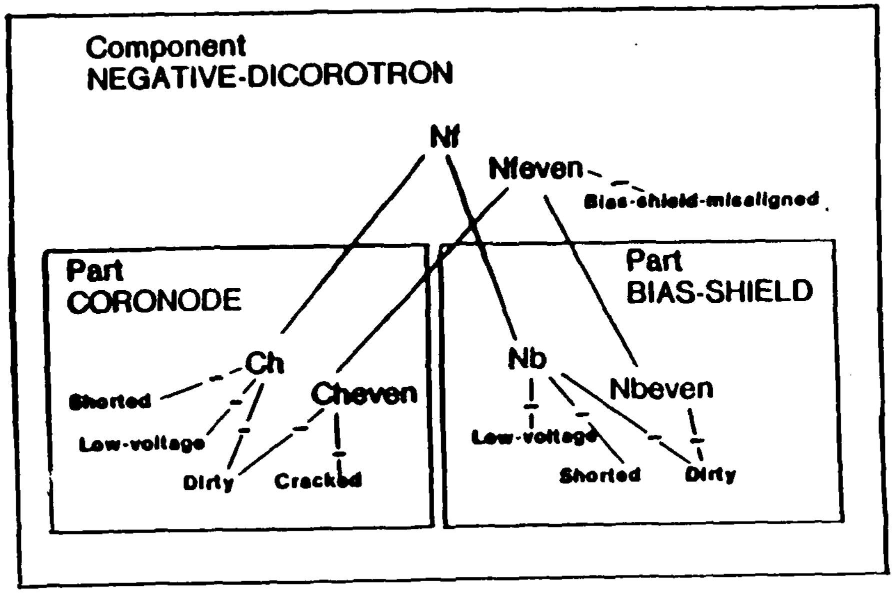
PART CORONODE

Function Variables
- Ch -- Level of charge from coronode
- Cheven -- Evenness of charge from coronode

Fault States
- Shorted -- Coronode wire sho.t.d
- Cracked -- Coronode glass cracked
- Dirty -- Coronode glass dirty
- Low-Voltage -- Insufficient voltage at coronode

Causes
(Ch DOWN) <== (Fault Low-Voltage) (Fault Shorted) (Fault Dirty)
(Cheven DOWN) <== (Fault Cracked) (Fault Dirty)

Figure 4 The Negative Dicorotron and a Diagnostic Model of a Coronode



COMPONENT: NEGATIVE DICOROTRON

Subcomponents
(CORONODE NEGATIVE-BIAS-SHIELD)

Function Variables
- Nf -- Level of negative field from dicorotron
- Nfeven -- Evenness of field from dicorotron

Function
- Nf <== (P Ch) (P Nb)
- Nfeven <== (P Cheven) (P Nbeven)

Fault States
- Misaligned -- Bias Shield not aligned correctly

Causes
(Nfeven DOWN) <== (Fault Misaligned)

Figure 5 A Diagnostic Model of the Negative Dicorotron

PRODUCT PR (Photoreceptor)

Effect Variables
- V -- Level of negative voltage held by photoreceptor
- Veven -- Evenness of voltage held by photoreceptor
- To -- Amount of toner on photoreceptor

Function Variables
- Vc -- Voltage capacitance of photoreceptor
- Vceven -- Evenness of capacitance of photoreceptor

Fault States
- Fatigued -- Photoreceptor fatigued
- Belt-Grounded -- Photoreceptor belt grounded

Causes
- (Vc DOWN) <== (Fault Fatigued)
- (Vceven DOWN) <== (Fault Belt-Grounded)

STAGE: CHARGING

Inputs
- (PR)

Outputs
- (PR)

Components
- (Negative-Dicorotron)

Function
- V <== (P Nf) (P Vc) (P V)
- Veven <== (P Nfeven) (P Vceven) (P Veven)
- To <== (P To)

Figure 6 Diagnostic Models of the PR PRODUCT
and the Charging STAGE

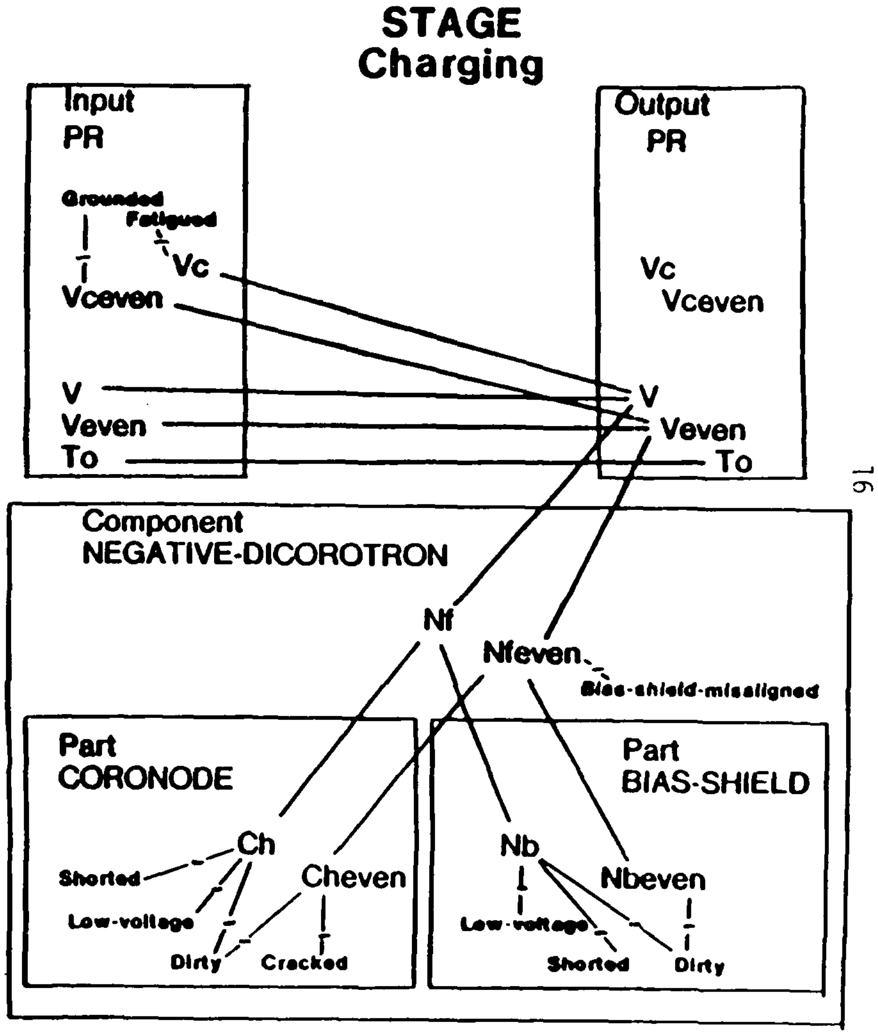


Figure 7 A Graphical Characterization of Charging

Xerography

Connections:

```
((((INPUT PR)) Cleaning)
  (((Cleaning PR)) Charging)
  (((Charging PR)) Exposing)
  (((Exposing Exposed-PR)) Developing)
  (((Developing Developed-PR) (INPUT PAPER)) Transfer)
  (((Transfer Developed-PAPER)) Fusing)
  (((Fusing Fused-PAPER)) OUTPUT))
```

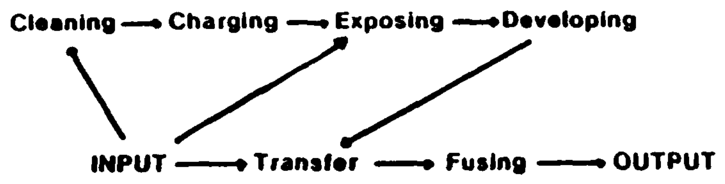


Figure 8. A Diagnostic Model of the Xerographic Process Viewed as a Process Line

```
R1: If have observed symptom or possible effect (x DOWN)
        and the model states that x <== (P y)
            then have possible effect (y DOWN).


R2  If have observed symptom or possible effect (x UP)
        and the model states that x <== (P y)
            then have possible effect (y UP).


R3: If have observed symptom or possible effect (x DOWN)
        and the model states that x <== (I y)
            then have possible effect (y UP).


R4. If have observed symptom or possible effect (x UP)
        and the model states that x <== (I y)
            then have possible effect (y DOWN).


R5. If have observed symptom or possible effect (x DOWN)
      and the models states that (x DOWN)  <==  (Fault y)
            then have possible fault y.


R6. If have observed symptom or possible effect (x UP)
      and the models states that (x UP)  <==  (Fault y)
            then have possible fault y.
```

Figure 9 Basic Diagnostic Reasoning Rules

```
The following represent causal stories for the effects:
((Toner-for-image-on-paper-at-Output DOWN))

(Negative-Voltage-on-PR-after-Charging DOWN)
   = => (Negative-Voltage-for-image-on-PR-after-Exposure DOWN)
    = => (Negative-Voltage-for-image-up-at-Development DOWN)
     = => (Toner-for-image-on-PR-after-Development DOWN)
      = => (Toner-for-image-on-paper-after-Transfer DOWN)
       = => (Toner-for-image-on-paper-after-Fusing DOWN)
        = => (Toner-for-image-on-paper-at-Output DOWN)


The following represent causal stories for the effects:
((Negative-Voltage-on-PR-after-Charging DOWN))

(Negative-Voltage-on-PR-before-Charging DOWN)
   = => (Negative-Voltage-on-PR-after-Charging DOWN)

(FAULT Photo-receptor-fatigued)
   = => (Negative-Voltage-capacitance-of-PR DOWN)
     = => (Negative-Voltage-on-PR-after-Charging DOWN)

(FAULT Voltage-low-at-negative-bias-shield-at-Charging)
   = => (Negative-field-at-shield-at-Charging DOWN)
     = => (Negative-field-from-dicorotron-at-Charging DOWN)
       = => (Negative-Voltage-on-PR-after-Charging DOWN)

(FAULT Negative-bias-shield-shorted-at-Charging)
   = => (Negative-field-at-shield-at-Charging DOWN)
     = => (Negative-field-from-dicorotron-at-Charging DOWN)
       = => (Negative-Voltage-on-PR-after-Charging DOWN)

(FAULT Coronode-wire-shorted-at-Charging)
   = => (Charge-from-coronode-at-Charging DOWN)
     = => (Negative-field-from-dicorotron-at-Charging DOWN)
       = => (Negative-Voltage-on-PR-after-Charging DOWN)

(FAULT Coronode-glass-dirty-at-Charging)
   = => (Charge-from-coronode-at-Charging DOWN)
     = => (Negative-field-from-dicorotron-at-Charging DOWN)
       = => (Negative-Voltage-on-PR-after-Charging DOWN)

(FAULT Voltage-insufficient-to-coronode-at-Charging)
   = => (Charge-from-coronode-at-Charging DOWN)
     = => (Negative-field-from-dicorotron-at-Charging DOWN)
       = => (Negative-Voltage-on-PR-after-Charging DOWN)
```

Figure 10. Causal stories for light copy that involve the negative dicorotron at charging

Talking About Machines

Social Aspects of Expertise

Julian E. Orr
Intelligent Systems Laboratory
Xerox Palo Alto Research Center

January 1987

Abstract: *The diagnostic process for copiers is intimately interwoven with the narration of the technician's understanding of the state of the machine. This follows from the fact that copiers are elaborate assemblages of relatively simple mechanisms, and the problem in diagnosis is not so much the testing of components as keeping track of the tests and making sense of their results. The anecdotal re-telling of this narrative to one's associates constitutes the mechanism for incorporating the diagnostic experience into the community expertise. These anecdotes are remembered and used or referred to during the diagnosis of other difficult problems or when seeking help. Individual expertise is in part the ability to interpret the anecdotes, to abstract the information about the machine from the context of the story.*

This report describes research done at the Intelligent Systems Laboratory of the Xerox Palo Alto Research Center. Support for this research was provided in part by the Army Research Institute under contract number MDA903-83-C-0189.

This paper presents some interim results from an ethnographic investigation of Xerox's service force. The research is part of a contract with the Army Research Institute to develop methods of training technicians. Since part of the original rationale for the project was to find ways to increase the number of technicians who became expert during their career, this work is an inquiry into the nature of expertise and the process of diagnosis. The original technological focus of the project was the Xerox 1075 copier, so the fieldwork was primarily among the Xerox Customer Service Representatives [CSRs] maintaining the 1075. Representation of this material necessarily involves a certain amount of technical information. I have tried to explain the technology to the level necessary to understand the cultural meaning of an utterance while the technical details may remain opaque.

The Machine

The 1075 was chosen because it was a new machine with some innovations that were said to be causing problems among the service force. Designed for a monthly volume of less than 40,000 copies, it was placed in situations demanding 6 and 7 times that, which quickly produced unanticipated problems. The 1075 relies more heavily on electronics than earlier machines, employs an internal network to connect relatively autonomous electronic subsystems, uses electronics for electrostatic adjustment and control, and has a simple feedback system in the electrostatics to maintain copy quality. It has extensive onboard diagnostics and maintains error and use logs, all of which are accessible through the control panel. One quirk of the system is that the diagnostic information is given only as values in the range of numbers which can be represented in the cells in the machine memory, not in common terms such as volts. Similarly, the procedures for adjusting the electrostatics are identified by procedure number, not by identifying the function being adjusted. This use of numbers instead of units or functions serves to distance the technician from the information in the machine. The information for conversion to more understandable terms is available but awkward to use. The interesting point with reference to the community of technicians is that they have assimilated the new representation of data and deal comfortably and apparently knowingly with this abstruse representation of the truth of the machine. They do not consciously translate, and it is not clear whether they know the equivalence between a numbered operation and the task of adjusting developer bias or the equivalence between a value and a voltage. It is only clear that dealing with the machine at this remove is not now an issue for them.

The Documentation

The National Service Organization of the Business Systems Group, the organization responsible for sales and service of Xerox equipment, intended that the Customer Service Representatives should be directed in their diagnoses of the 1075. It was supposed that careful following of the directive documentation would lead to the solution of all problems more quickly than could be accomplished by the reasoning of the technicians. The obvious corollary to this supposition is the belief that all problems can be anticipated and their solutions included in the documentation. The documentation includes set up and repair procedures, simplified schematic diagrams, and Fault Isolation Procedures, the diagnostic element. I have included samples of the Fault Isolation Procedures, or FIPs, and the Block Schematic Diagrams in Appendix A. The fault isolation procedures consist of a series of procedures, with each action defined with considerable detail, and each branching condition presented as a simple Yes/No choice. Solutions are branches to similarly directive repair procedures. There is no rationale presented; the explicit purpose of the tests and the interpretation of the results are both known only to the designers of the documentation. Later discussion will cover the failure of this intent to dictate all action and the ways in which the documentation is actually used.

Fieldwork: Training

The first portion of the field work was to attend the 1075 repair school at the Xerox International Center for Training and Management Development at Leesburg, Virginia. This is a three week course to teach experienced technicians how to repair the 1075. In keeping with the aim of accomplishing service by rote procedure following, there is considerable emphasis on learning to use the documentation. The organization and systematic use of the documentation are the subject of an early module of the course: as an exercise during the fault diagnosis problems, one keeps a score sheet of progress through the different levels of procedures to the repair procedure. However, students are also issued a book of the principles of operation for the 1075, and the course book includes relevant segments from the principles. The result is that one's learning about the 1075 is informed with as much machine knowledge as was present in earlier courses. The fact that the knowledge about the machine is available permits the students to extrapolate from their experiences with other machines to an understanding of the 1075 and an interpretation of the documentation which permits them to reason about their diagnoses. Another feature of the training method which unintentionally encourages understanding is that students work in pairs, doing exercises and debugging together. This requires frequent comparison of understandings which in turn aids the development of understanding.

Fieldwork: The Service Organization

Subsequent field work consisted of twelve days travelling with six technicians of a local branch office of the National Service Organization of Xerox's Business Systems Group. The branch service function is divided among various teams according to the product being serviced, nominally on the basis of copy volume capacity. Each team has a manager, a technical specialist who oversees technical skills and consults on difficult problems, and various categories of technicians. During the field work, two days were spent with each technician; this allowed both for variations in the daily work load and sufficient time to accommodate the different styles of each technician. Most of the technicians were from the mid-volume team of the branch, one was the technical specialist of that team, and one was from the low-volume team. The last was recommended by the technicians from the mid-volume team for his skill in managing his territory. At that time, the mid-volume team consisted of one Field Service Manager [FSM], one Customer Service Engineering Representative [CSER], the technical specialist of the team, and nineteen Customer Service Representatives [CSRs] and senior CSRs, the technicians, who are colloquially known as tech reps for Technical Representative, their former title. At the time of the study, four of the nineteen were women. Although the study was focussed on technicians servicing 1075s, the mid-volume team also services a smaller new machine, the 1055, and a dwindling population of older machines, the 4000 family and the duplicators. A significant number of the technicians have experience with the 4000 family. The 4000 series of machines were difficult to service and remained in use long beyond their expected life span; consequently, successful 4000 technicians developed considerable resourcefulness and a propensity for pooling their information with their fellows. Since the time of the study, the FSM has changed, some aspects of branch and region organization have changed, the CSRs' title has changed, and a new machine, the 1090, has been added to the machine population; the technicians, however, remain.

One of the most obvious aspects of tech rep culture is the eagerness with which they talk about machines and their fondness for war stories. Wherever technicians gather, much of the conversation is anecdotes of their experiences with machines and customers. During the working day, tech reps will meet at particular restaurants and coffee shops for breakfast, lunch, or just for coffee when things are slow; the conversation always includes the latest stories of unusual machine behavior. This is also true of the Parts Drop, where one can usually find several technicians. The team studied had meetings every week; the beginning of these meetings was usually deliberately delayed to give the team members time to swap stories. At the training center at Leesburg, conversation over meals or in the evenings covered strange and wonderful variations of how machines fail. The participation of technicians from all over the country with experience servicing a wide assortment of machines gave the stories here added variety and interest. Indeed, Xerox folklore includes a possibly apocryphal tale in which C. Peter McCullough, former CEO, President, and Chairman of Xerox, responded to criticism of the expense and inconvenience of sending technicians to Leesburg for training by asserting that the fact that they spent their evenings sitting around talking Xerox was of great benefit to the company as a sharing of experience. I contend, in fact, that war stories are crucial to the effective functioning of the technician community, and the rest of this paper is intended to illustrate and elaborate this point.

A Trouble-Shooting Session

One prolonged trouble-shooting session observed during the field work offers multiple examples of some of the functions of war stories. The machine had recently been installed at a new building of a major account and had never worked reliably; it suffered repeated, intermittent crashes of the controlling logic system but would run again after being off for some short amount of time. The control system is one of the weak points of the 1075; it is also one of the areas in which the technicians' experience is weakest and in which they receive the least training. Although the error message this machine produced in crashing, "Failed Self-Test", always indicates a specific subsystem, the nature of the interconnection of the subsystems makes this indication unreliable, and, in fact, the messages had indicated several different parts of the machine. Many subsystems had been replaced; the problem persisted. A cable damaged by mechanical components had been found and replaced: this seemed promising, but the problem persisted. The final trouble-shooting session was a five-hour effort involving a senior technician and the team's technical specialist, with some minor participation by the ethnographer. Examination of the transcript of this session yielded a dozen anecdotes told during the trouble-shooting, taking a variety of forms and serving a variety of purposes. *[Figure 3 in Appendix A shows the sequence of events during this session and how they led toward the goal of fixing the machine.]*

"It'll sit there and go . . . click-click"

The first anecdote [B1], [Story 1 in Appendix B], was told by the specialist during a casual consultation with the technician as the latter worked on a different machine at the Xerox branch office. The story is about yet another machine on which the specialist had been working, and the question was whether it applied to the problem machine under discussion here, given that the symptom is typical of all 1075s under certain conditions. The gist of the consultation was to look for the occurrence of this clicking symptom, but it was emphasized by a story telling how the meaning of the symptom had escaped the specialist and in consequence he had recently spent hours pursuing a simple mistake made during the installation of the machine. This is relevant to the fact that the problem machine was also a new installation.

On arrival at the customer site, the problem machine in question was down with the usual error message; the repair indicated by the documentation for these symptoms had already been done, and the pattern of recurrence combined with experience led to the premise that the replacement part was probably not at fault. The apparently random distribution of self-test failures suggested a problem with the power supply; and, since the low voltage power supply had already been replaced more than once, attention shifted to the AC supply. Preliminary examination revealed nothing wrong other than a possibly marginal feed: the voltage dropped perceptibly every time the heat rod in the fuser came on. Further thought suggested that irregularities in the AC supply would be unlikely to produce such consistent failures of self-test; however, the possibility of a problem with the AC supply remained with the technicians. The tests the documentation demanded before asking for the replacement of the part indicated by the error message were to check the 5 volt supply lines and the Reset line; examination of the schematics revealed that these are common to all of the subsystems of which the "Failed Self-Test" message had complained [and are, indeed, common to all electronic subsystems]. This commonality suggested an inspection of the wire harness for pinches, scrapes, or sharp bends; none were found. At this point, the decision was made to change one of the as-yet-unchanged subsystems, although there was no particular reason to suspect the chosen board. Throughout this part of the diagnosis, the dialogue is a constant flow of queries and tentative offerings, bits that might fill in blanks in a puzzle or in a narrative, or attempts to flesh out the explanation that would account for the machine behavior if the offerings proved to fit.[B2] We will return to this point later.

The decision to swap boards satisfied nobody; it was done merely to eliminate a remote but cheap possibility, and thoughts turned to further diagnostic strategies. Perturbed by the random distribution of faults and the fact that the machine often crashed while sitting idle, both technicians began thinking about problems in the power systems or the AC supply as they brought the machine back up. They made attempts to gain perspective, asserting that the problem could not be that hard, that they were just looking in the wrong place. They also began thinking about some expensive subsystem exchanges, just flirting with the idea for the time being. As they thought about these things, they told stories, and the following stories were interspersed among the reflections listed above. After one made the point that failing self-test opens the 24 Volt Interlock Relay, they told about a problem which can produce that symptom without being a 24 Volt problem. This tale, interestingly enough, was told by both in antiphonal form, with different meanings.[B3] The senior technician told his version, which was of a pattern-matching diagnosis associating a second error code with the first and a fix which related to the second error code, while the technical specialist pointed out that the second error code doesn't always occur, that this misleading error code was a new problem produced by a fix to an electronic subsystem [previously the fault in question destroyed the electronic subsystem], that one should look for apparently random and insignificant errors in the logs attributed to that particular electronic subsystem, and that failure to make the association would mean, as it had twice for him, long hours tracing the 24 Volt Interlock distribution until luck caused the significant error code to occur or memory brought the previous incident to mind. Furthermore, testing for the true fault requires a longer than normal period of stressing the system if the fault is infrequent. The different versions of the same story, and the different models informing those versions, make this an exceedingly interesting war story. However, given that the machine crashed while idle, it wasn't really relevant, as they realized.

Subsequent stories reverted to the idea of problems in the AC supply, which produced a string of anecdotes ranging from detailed war stories to rather elliptical references. The point is that war stories such as these and the preceding one are told as part of the process of considering possibilities, to refresh one's memory of the contextual details of earlier encounters with the machines and to aid in examining the applicability of that experience to the current problem context. There were several reasons to be considering the AC supply at this point: First, there is a history of having AC problems in new buildings, which fit the situation; second, there is a history of 1075s having AC problems even in older buildings due to internal differences in the way they switch the AC; third, they were running out of known possible causes in the 1075 and seeking to expand their problem space; and fourth, if the AC were the problem, it would be someone else's responsibility. With reference to the last, several of the tales also carried the theme of the recalcitrance of electricians and their reluctance to accept that there might be problems with the AC or its distribution or that these problems might cause difficulties for the copiers in the building. So, the senior technician told a somewhat wishful tale of irregularities in the Northern Indiana Public Service Power causing Whirlpool in Indiana to install battery banks with concomitant rectifiers, inverters, and backup generators to ensure smooth power delivery to the computer which handled their spare parts orders.[B4] Next, three stories between the two men dealt with some electricians demanding chart recordings to prove irregularities in the power while all the computers in the building are having line protection added and other electricians failing to grasp that voltage readings on an open line are not the same as current to a machine.[B5-B7] The emphasis then switched to problems with the neutral line, significant to 1075s and not usually critical in AC installations, followed by connector failures, which had little to do with the problem at hand but reflected poorly on electricians.[B8.9.10.11] The survey of the AC repertoire produced no inspirations, the machine was running, and the technical specialist was faced with the grim possibility of having to trade machines, an expensive process in terms of status and obligations. The group adjourned to lunch at Burger Thing to allow the machine to crash in private.

The machine duly crashed. When the group reassembled after lunch, the technical specialist was ready to commit to one of the expensive subsystem exchanges even before we had done any testing other than to verify that all voltages were at their expected levels. A signal from the board in question was observed to be in the wrong state; this support for the consideration that this board may be at fault apparently triggered the technical specialist's memory. He commented that he had seen it happen that a shorted ribbon cable could destroy this board. [The ribbon cable in question was earlier found to be mangled and was then replaced.] The cost of replacing this board is not simply the parts cost; this board holds the memory containing all the electrostatic settings, the use and error logs, and the billing meters. In the face of this, the technical specialist again commented that he'd seen the cable destroy this board or the seven-segment LED display on the front panel, or both.[B12.13] At that he went to the branch for a new board and the damaged cable. As additional evidence, this time the error log included a failure for this board. The senior technician said that the pain of changing this board prevented one from doing so in the absence of definite failure indications; this fact would also make them reluctant to suspect it. In addition to recording and reporting all the service, use, and billing meter data for the old board and the new, getting the change registered with the work support data base seems to be problematic and requires continued reiteration that the board and billing meters have, in fact, been changed.

One of the classic sources of information in diagnosis is the users in the customer site: a would-be user who appeared during this process commented that the machine had crashed even more frequently when selecting reduction. The significance of this became clear when the technical specialist reappeared with the board and the damaged cable; inspection of the cable indicated that the wires which had been mangled and shorted together were drivers for the seven-segment LEDs showing reduction. At this point, there was a chorus of those present listing the symptoms and how they tied together to convict that board.[B14] The two technicians proceeded to record the data, change the board and program the necessary data back into the machine. Several times various participants re-told how the cable was mangled, damaged the board, and caused all these machine crashes; the technical specialist said that he should have made the connection from the cable to the board sooner. It should be pointed out that despite all this confidence no one then actually knew that the machine had been fixed; after all, the machine would always run after it had been shut off. However, the problem really had been found and never recurred.

Part of the significance of this diagnosis is that the technicians were fixing a problem they couldn't understand and were not equipped to diagnose. The diagnostics had failed to isolate the problem, the schematics show only a few measurable signals and some extremely vague block diagrams of the logical functions of the machine, and the technicians are not trained to do diagnosis of electronic systems. Their only resource was experience, either their own or what they knew of others' experience. With this they could trade on known associations between symptom and cause, if any, or they could search the signal lines and power distribution lines for anything appearing out of the ordinary which might point to a specific component or subsystem. Accordingly, re-telling the tenuous connections between damaged wires, error log entries in a log characterized by randomness, and information from passing users is a way of acquiring sufficient confidence in the diagnosis to undertake an expensive repair; anecdotal references to comparable combinations of symptoms weigh heavily in this process. Repeating the litany of supporting evidence prepares one to tell the whole story to others, or to use as a new definition of the problem in the event the fix fails.

To sum up, then, this episode began with a suggestive anecdote of a symptom to look for and its possible ramifications. The dialogue during testing consisted of questions, answers, and tentative bits of explanation, a narration of the process of achieving understanding. A reflective pause in the diagnostic process yields the most anecdotes, as the technicians sort through their experience for points relevant to the present context; one story turns out to have two versions, one much more problematic than the other. The story subjects shift as different candidates for the root of the problem are considered, but, since the stories produce no inspiration, the story-telling ceases. Later, new symptoms suggest a different culprit; this jogs the memory sufficiently to produce two anecdotal references to experience as confirmation of the new hypothesis. These are added to the evidence in a recitation of conviction before the offending part is changed.

Before proceeding to a discussion of the fit between diagnosis and narrative which makes the war story such an apt tool for the community memory, it should be pointed out that large parts of a technician's life and work are omitted from war stories. Routine maintenance, although a major part of the work, is boring, and so no one tells stories about it. Most machine failures are boring; parts that break often, that everyone knows how to fix, are not interesting to talk about. Failures that were hard to diagnose, and therefore interesting, the first time get to be so well known that seeing the symptom is equivalent to knowing the solution; these don't make very interesting stories and so get the most elliptical of references. However, technicians who remember when it was a new problem can tell the whole story if asked. New problems are interesting, new manifestations of the extremes of machine behavior or of human behavior with machines. Problems that require consideration of the chains of cause and effect in the machines and that shed new light on those chains are interesting to talk about. From the perspective of the community memory, it is necessary to talk about these problems to preserve and circulate the knowledge of how to solve them, but it is interest and involvement that cause the technicians to tell these stories. The stranger the sequence of events and interactions producing the machine behavior, the better the story is, and the more fun it is to tell. A recent favorite is about the customer who refilled the 1075 toner bottle with 1048 toner, which is of the wrong polarity and which ruins the copy quality; this was a nearly impossible problem to diagnose because it could only occur with human intervention. It makes a wonderful story.

The Diagnostic Problem

To understand the utility of war stories in the technician community, one needs to consider the nature of the diagnostic problem with reference to copiers. From the technician's perspective, a copier is an elaborate assemblage of simple mechanisms. While the control systems and power supplies are not simple, they are types of devices that generally fall outside the experience of most technicians and traditionally have been treated as black boxes. Xerography is treated as a set of simple mechanisms by ignoring the physics of the process and simply describing the movement of components and the possible observations and measurements, tying these together with a functional description of the process. The rest of the machine is composed of solenoids, gears, clutches, cams, belts, chains, levers, and of other electro-mechanical components. One of the significant facts about these items is that it is possible to verify their operation or non-operation by direct observation, and it seems clear that the technicians understand the functioning of these electro-mechanical items. Understanding the functioning of the components is not the difficulty. The difficult part of diagnosing these machines is keeping track of the chains of events and ordering the results of observations and tests into an understanding of the overall functioning of the machine under consideration.

The production of this understanding appears to be a verbal process; troubleshooters talk to themselves, reminding themselves what they have done and what they think it means. This monologue also includes specific questions, asking what is the meaning of certain test results, what corollary symptoms should appear, and what testing should be done to confirm or deny the indication. Of course, the process of producing an understanding of the machine is more overtly verbal in joint trouble-shooting sessions, where the recitation also serves the obvious function of co-ordinating the participants. Even so, the speaker is not just telling but also listening, appraising and assessing his or her own narration of the trouble-shooting for its contribution to the understanding of the problem. In the trouble-shooting episode reviewed above, most of the dialogue during testing consisted either of questions about meaning or of pieces of narrative offered tentatively to fill a perceived gap in the understanding. When the last bits were added, the testimony of the user and the evidence of the damaged cable, there was virtually a group recitation of the salient points of the narrative, as all those present checked to see whether there was a satisfactory understanding of the state of the machine, including and accounting for all of the observed behavior.

Diagnosis, the process of reaching an understanding of the state of the machine, thus consists of the interaction between performing tests, which yield facts about the machine, and integration of those facts into an understanding of the flawed state of the machine, which is done through a narration of the testing and a verbal assessment of its results, producing a verbal description of this flawed state. The distinctive character of the diagnostic narrative comes from the fact that it combines a description of the state of the machine and an explanation of that state with a history of how the machine reached that state and a history of how the technician achieved an understanding of the situation. The fact that reaching understanding seems to have this verbal form has obvious benefits for the circulation of information through the community. *[Figure 4 in Appendix A is a schematic illustration of this process.]*

The Issue of Control

Before considering the social implications of these diagnostic narratives, there is a somewhat parenthetical parallel issue to mention. If the principal difficulty in diagnosis is creating and maintaining a coherent representation of the information about the machine gathered from relatively simple tests and observations, this imposition of order on information is matched by a drive to impose order on the world, to preserve order in one's work and to appear always to be in control of the situation. For example, a technician who walks up to the wrong machine cannot walk away, but must *do* something, perform a minor service and checkup, so as not to appear to have walked up to the wrong machine. Similarly, it seems important to technicians to keep tools and parts in order, to keep track of them and not have them scattered all around the room. It is not clear whether they wish to convince customers, colleagues, or themselves with this appearance of being in control. There appears to be a felt need to keep the situation as orderly as possible in the face of imminent chaos, and technicians have an almost Sisyphean view of the probability of chaos, as this quote from one shows: "One week everything's OK; the next you have calls up on all machines, all of them call-backs . . ." It is interesting, then, that technicians having difficulty may often be found working in a chaos of tools and parts, unable to give a coherent account of what they know or how they've proceeded. In fact, some experienced technical specialists report incoherence as the principal characteristic of a technician in trouble. As a counterpoint, one of the best technicians is famous for his control of his territory and of each task, carrying only those tools needed for a specific job, keeping all put away except those actually in use, and performing even the messiest operations with minimum fuss and in the cleanest possible fashion. His stories are clear and coherent but not very interesting because his machines are unproblematic. Coherence, or order, then, appears as a major theme of technician culture, generally in the desire to maintain control, specifically in the need in diagnosis to order the information into a narrative reflecting a coherent qualitative understanding of the machine.

The Use of Narratives

Diagnostic narratives, created from the need to arrange many disparate facts about the machine into a coherent understanding of its condition, are the raw material for the war stories of the technicians. Anecdotal retelling of the salient points of a narrative includes the significant symptoms and their import for the machine with as much contextual information as necessary to achieve the desired understanding of the war story. War stories, then, are the medium for preserving the narrative of diagnosis by conveying it to the rest of the community for incorporation into the community expertise. They will be retold as needed until the information is assimilated and the whole problem becomes routine. In the trouble-shooting episode discussed above, there were a variety of war stories and elliptical references to war stories. The more elaborate anecdotes were told in situations where it seemed necessary to spell out context more fully, to examine parallels or to set the stage for the technical points of the story. Stories became shorter in the series on AC once the ground had been established and as it also began to seem less likely to prove relevant to the problem. The final two references confirming the emerging diagnosis are not even full stories; still, enough is told to establish the relevance to the situation and to validate the technical points. Some war stories seem to be told primarily for amusement value. Another observed use of an anecdote was as a claim to status as a member of the community. A technician asserted that no one was a member of the team until they had broken something in a flagrant and glorious fashion: this particular technician had burned up a power supply in the presence of a customer by a particularly egregious oversight and so gained membership in the group.[815]

Sometimes, too, the technical details of a war story are not remembered. At worst, the technician will only remember that the problem has occurred before: a somewhat better memory will also recall whose problem it was. The first will lead to a call to the technical specialist; the second, to the technician who told the tale.

A special narrative case occurs when asking for help. This is perhaps the purest manifestation of the diagnostic narrative; the point is to present the evidence gathered and the tentative conclusions drawn, as well as the remaining questions and incongruities in order to get the person from whom help is desired to the same point in the problem space as the person seeking help. Another reason for maintaining close fidelity to the narration of the diagnostic process is that the process is still unfinished and it is not yet known which details are crucial and which are not. The social dynamics of asking for help are interesting: The person calling for help must demonstrate that they have done the groundwork in an orderly fashion: they must have performed the proper initial tests and observations, and should know what the groundwork indicates about the machine. It is socially unacceptable to ask for help without doing any testing of the machine at all. It is acceptable to have done extensive testing, to be drawing seemingly correct conclusions from the results, and still to have unexplained malfunctioning. The area between varies, depending on the problem and the reputation and experience of the technician seeking aid. A failure or refusal to draw obvious conclusions is not well received, but improper understanding of the evidence at a deeper level will get an answer, sometimes through questions leading the confused technician to understanding the connection between the evidence and the proper conclusion. The interesting point is that there seems to be a community expectation of proper presentation of the problem by those seeking help in their diagnosis.

Anecdotes and Models

It is the situated quality of anecdotes which makes them so peculiarly apt as units of community memory; war stories combine information about the machines with the context of a specific situation. The combination permits one to generalize from the information to an abstract mental model of the machine, while the contextual information provides a guide for the application of that model to real situations. Thus a war story told during a troubleshooting session is doubly situated, first in the situation of its origin and second in that of its application. There was a question at the beginning of this research as to whether technicians in the field reasoned from mental models of the machine or simply drew on extensive collections of case histories. From observations during the field work, it seems clear that the technicians in the field do reason from mental models, created from the information provided, extrapolations from other machines, and the accumulated experience of the community. Evidence from the field indicates some variation in completeness and accuracy of the model from one technician to another and from one area of the machine to another. These differences do interfere with communication and require additional explanation, but this in turn causes the models to converge over time. Given the weaknesses of most technicians in electronic theory or the physics of xerography, many of the extrapolations from field experience to the model are technically wrong at some level; they are, however, still functional in the service context.

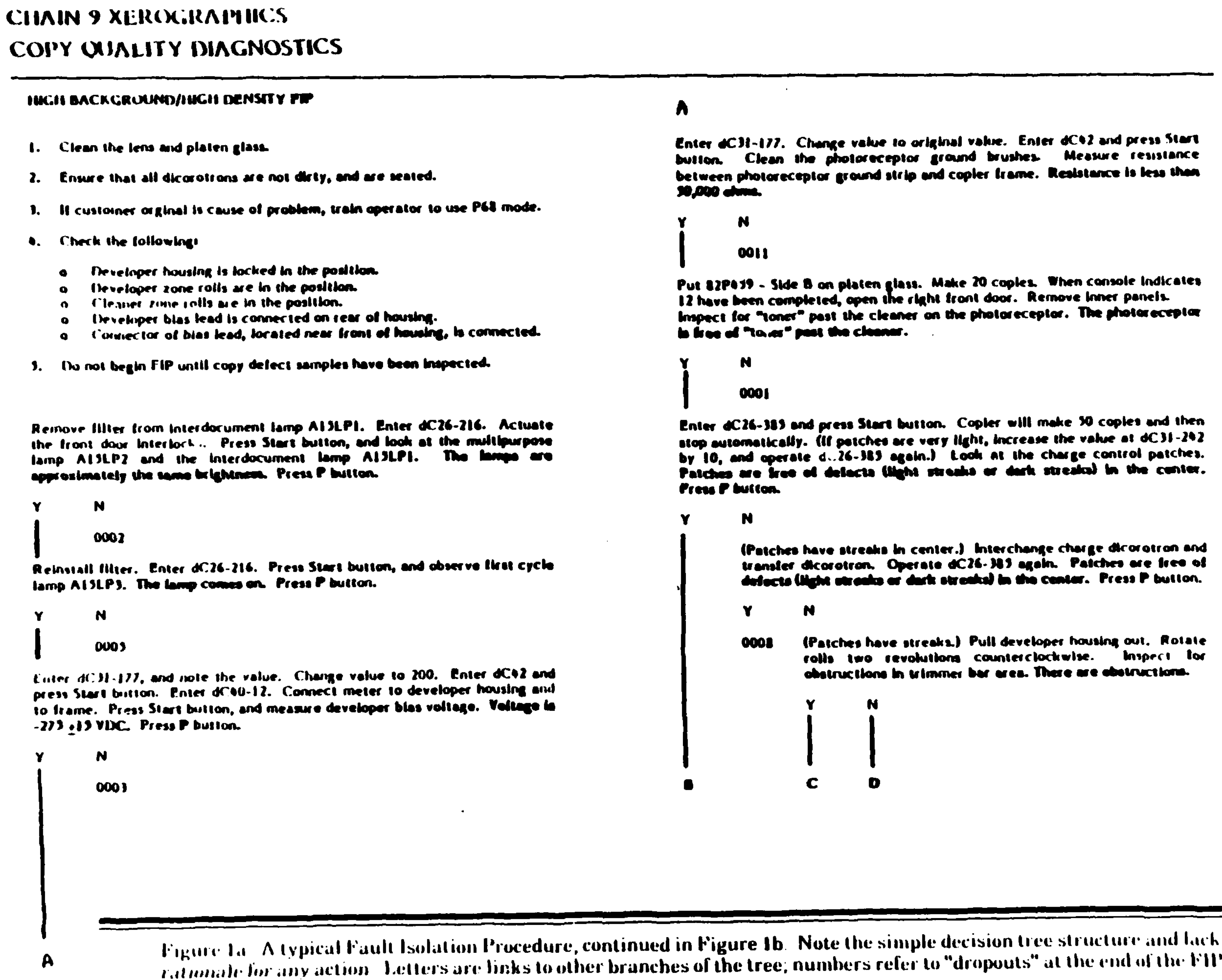
From the perspective of the task of servicing the machines, reasoning from a model clearly offers many advantages. First, a mental model provides flexibility for improvisation in the face of unanticipated problems. There are mnemonic advantages, in that a model can aid the organization of new knowledge and serve as a framework for the retrieval of information. In the absence of a model, use of experience requires remembering all of the details of the previous context and matching them rotely to the present. A model shared by the community is an artifact permitting sharing of experience, and, as a common basis for reasoning, enables consultation on recalcitrant problems. Anecdotes serve as a link between a model and the field, retrieving information from the model with the contextual information which constrains its application, or bearing information from the field to amplify the model, while preserving the context to guide generalization from the specific incident. A virtue of anecdotes is their utility by persons of differing levels of expertise, with different understandings of the machine: one would expect novices to use the details of context more, having a less general model, while an expert would be more interested in the improvement of the model by the anecdote.

Technicians' use of the documentation in diagnosis is clearly informed by their mental model. They do not use the documentation in the intended fashion: experience with unanticipated situations or incorrect diagnoses has taught them the brittleness of this approach. Instead, they analyze the Fault Isolation Procedure, reading through all the steps and examining the proposed solutions to determine what goal the procedure is pursuing and whether it is doing so in a reasonable way. Then they use those portions which seem consonant with their hypothesis of the problem and their current understanding of the state of the machine. Reluctance to rely on the documentation has several causes: Most of the real anticipated problems have become completely routine for any experienced technician; some of the anticipated problems do not occur; and many problems have appeared which were not anticipated and hence not covered in the documentation. This last case covers most of the serious problems presently occurring in the field; accordingly, any technician needs to understand the information yielded by the documentation's procedures in order to be prepared to proceed independently when the prescribed path ends short of solution. There is also some resentment of the documentation as discounting their skills, and they see the corporate trend to directive procedures as an effort to degrade their professional status.

The weakness of the documentation and its lack of congruity with the field situation leads to a discounting of the training program which leans so heavily on the documentation. This is not new; the service world has long believed that formal training did not cover real field problems and was little more than an introduction to the machine. It could be said that this introduction provides the skeleton of a mental model to use to assimilate the empirical lessons from the anecdotal community memory. The technicians' version is that the trip to Leesburg is a vacation and that they actually learn the machine by listening to their colleagues. During the 1075 course at Leesburg, it was observed that all of the stories were of other machines, even at the end. Inquiry of one technician produced the response that there was nothing to talk about because they didn't know anything about the 1075 yet. They wouldn't really know anything until they saw how it behaved in the field.

This need for empirical reference, perhaps, is the key to characterizing their understanding. Neither just a collection of case studies nor really a "deep understanding", it is a situated understanding, linking a mental model to the details of context. Assembling those details and interpreting them with reference to the model seems to be a narrative process; telling those narratives as war stories elaborates the shared community model, and provides an interesting, memorable, context-laden medium in which the information can circulate and be used.

Appendix A -- Illustrations



CHAIN 9 XEROGRAPHICS

COPY QUALITY DIAGNOSTICS

HIGH BACKGROUND/HIGH DENSITY FIP

1. Clean the lens and platen glass.

2. Ensure that all dicorotrons are not dirty, and are seated.

3. If customer orginal is cause of problem, train operator to use P68 mode.

4. Check the following:

   o Developer housing is locked in the position.
   o Developer zone rolls are in the position.
   o Cleaner zone rolls are in the position.
   o Developer bias lead is connected on rear of housing.
   o Connector of bias lead, located near front of housing, is connected.

5. Do not begin FIP until copy defect samples have been inspected.

Remove filter from interdocument lamp A15LP1. Enter dC26-216. Actuate the front door interlock.. Press Start button, and look at the multipurpose lamp A15LP2 and the interdocument lamp A15LP1. The lamps are approximately the same brightness. Press P button.

Y N

0002

Reinstall filter. Enter dC26-216. Press Start button, and observe first cycle lamp A15LP3. The lamp comes on. Press P button.

Y N

0005

Enter dC31-177, and note the value. Change value to 200. Enter dC42 and press Start button. Enter dC40-12. Connect meter to developer housing and to frame. Press Start button, and measure developer bias voltage. Voltage is -275 ±15 VDC. Press P button.

Y N

0003

A

A

Enter dC31-177. Change value to original value. Enter dC42 and press Start button. Clean the photoreceptor ground brushes. Measure resistance between photoreceptor ground strip and copier frame. Resistance is less than 30,000 ohms.

Y N

0011

Put 82P639 - Side B on platen glass. Make 20 copies. When console indicates 12 have been completed, open the right front door. Remove inner panels. Inspect for "toner" past the cleaner on the photoreceptor. The photoreceptor is free of "toner" past the cleaner.

Y N

0001

Enter dC26-385 and press Start button. Copier will make 50 copies and then stop automatically. (If patches are very light, increase the value at dC31-242 by 10, and operate dC26-385 again.) Look at the charge control patches. Patches are free of defects (light streaks or dark streaks) in the center. Press P button.

Y N

(Patches have streaks in center.) Interchange charge dicorotron and transfer dicorotron. Operate dC26-385 again. Patches are free of defects (light streaks or dark streaks) in the center. Press P button.

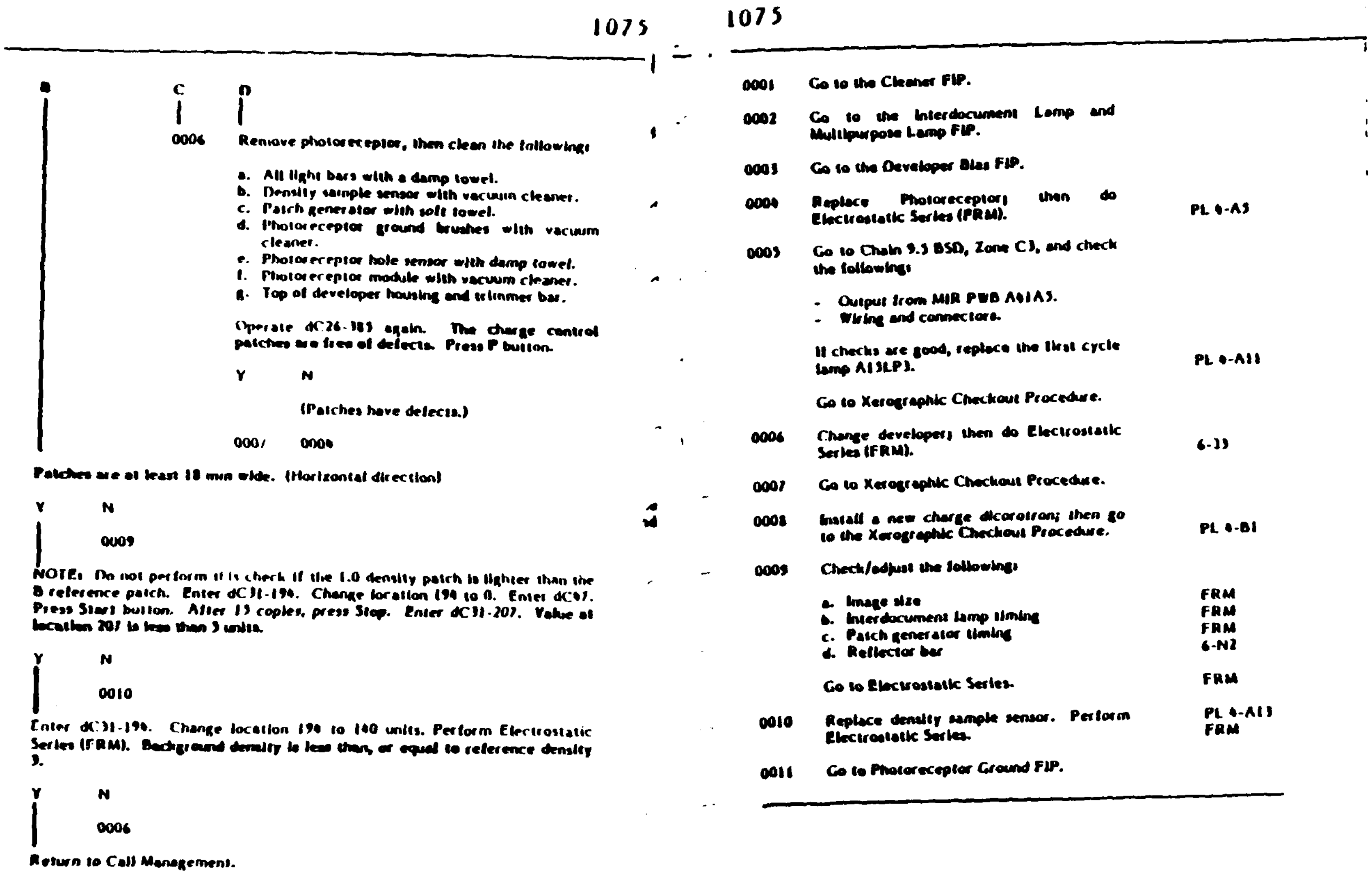
Y N

0008 (Patches have streaks.) Pull developer housing out. Rotate rolls two revolutions counterclockwise. Inspect for obstructions in trimmer bar area. There are obstructions.

Y N

B C D

Figure 1a A typical Fault Isolation Procedure, continued in Figure 1b. Note the simple decision tree structure and lack of *rationale* for any action. Letters are links to other branches of the tree; numbers refer to "dropouts" at the end of the FIP. Dropouts point to other FIPs or to repair actions.



1075

B

C D

0006 Remove photoreceptor, then clean the following:

a. All light bars with a damp towel.
b. Density sample sensor with vacuum cleaner.
c. Patch generator with soft towel.
d. Photoreceptor ground brushes with vacuum cleaner.
e. Photoreceptor hole sensor with damp towel.
f. Photoreceptor module with vacuum cleaner.
g. Top of developer housing and trimmer bar.

Operate dC26-385 again. The charge control patches are free of defects. Press P button.

Y N

(Patches have defects.)

0007 0008

Patches are at least 18 mm wide. (Horizontal direction)

Y N

0009

NOTE: Do not perform tl is check if the 1.0 density patch is lighter than the B reference patch. Enter dC31-194. Change location 194 to 0. Enter dC97. Press Start button. After 15 copies, press Stop. Enter dC31-207. Value at location 207 is less than 5 units.

Y N

0010

Enter dC31-194. Change location 194 to 140 units. Perform Electrostatic Series (FRM). Background density is less than, or equal to reference density 9.

Y N

0006

Return to Call Management.

1075

0001 Go to the Cleaner FIP.

0002 Go to the Interdocument Lamp and Multipurpose Lamp FIP.

0003 Go to the Developer Bias FIP.

0004 Replace Photoreceptor; then do Electrostatic Series (FRM). PL 9-A5

0005 Go to Chain 9.5 BSD, Zone C3, and check the following:

- Output from MIR PWB A61A5.
- Wiring and connectors.

If checks are good, replace the first cycle lamp A13LP3. PL 9-A11

Go to Xerographic Checkout Procedure.

0006 Change developer; then do Electrostatic Series (FRM). 6-33

0007 Go to Xerographic Checkout Procedure.

0008 Install a new charge dicorotron; then go to the Xerographic Checkout Procedure. PL 9-B1

0009 Check/adjust the following:

a. Image size FRM
b. Interdocument lamp timing FRM
c. Patch generator timing FRM
d. Reflector bar 6-N2

Go to Electrostatic Series. FRM

0010 Replace density sample sensor. Perform Electrostatic Series. PL 9-A13 FRM

0011 Go to Photoreceptor Ground FIP.

Figure 1b Continuation of FIP from Figure 1a The bottom of the left column, Return to Call Management, represents the failure of this procedure The numbered items on the right are the dropouts which either indicate repair procedures or point to other FIPs

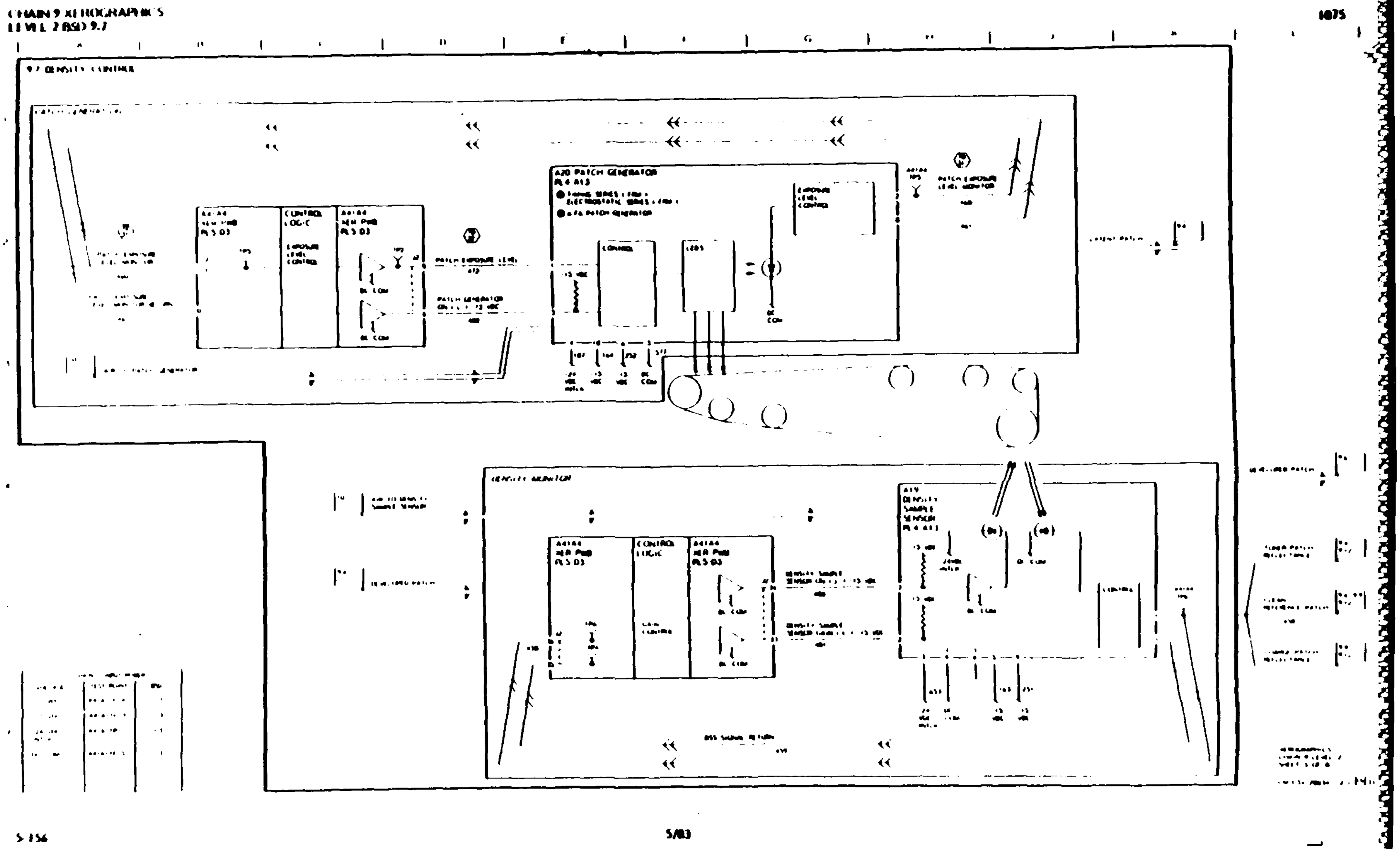


Figure 2 The Block Schematic Diagram [BSD] for the Density Control Circuit. Note that the information is principally interconnections, signal labels, and sometimes signal levels. Mechanical movement of photoreceptor and compressed air is also shown, but virtually no information about the electronic circuitry is presented.

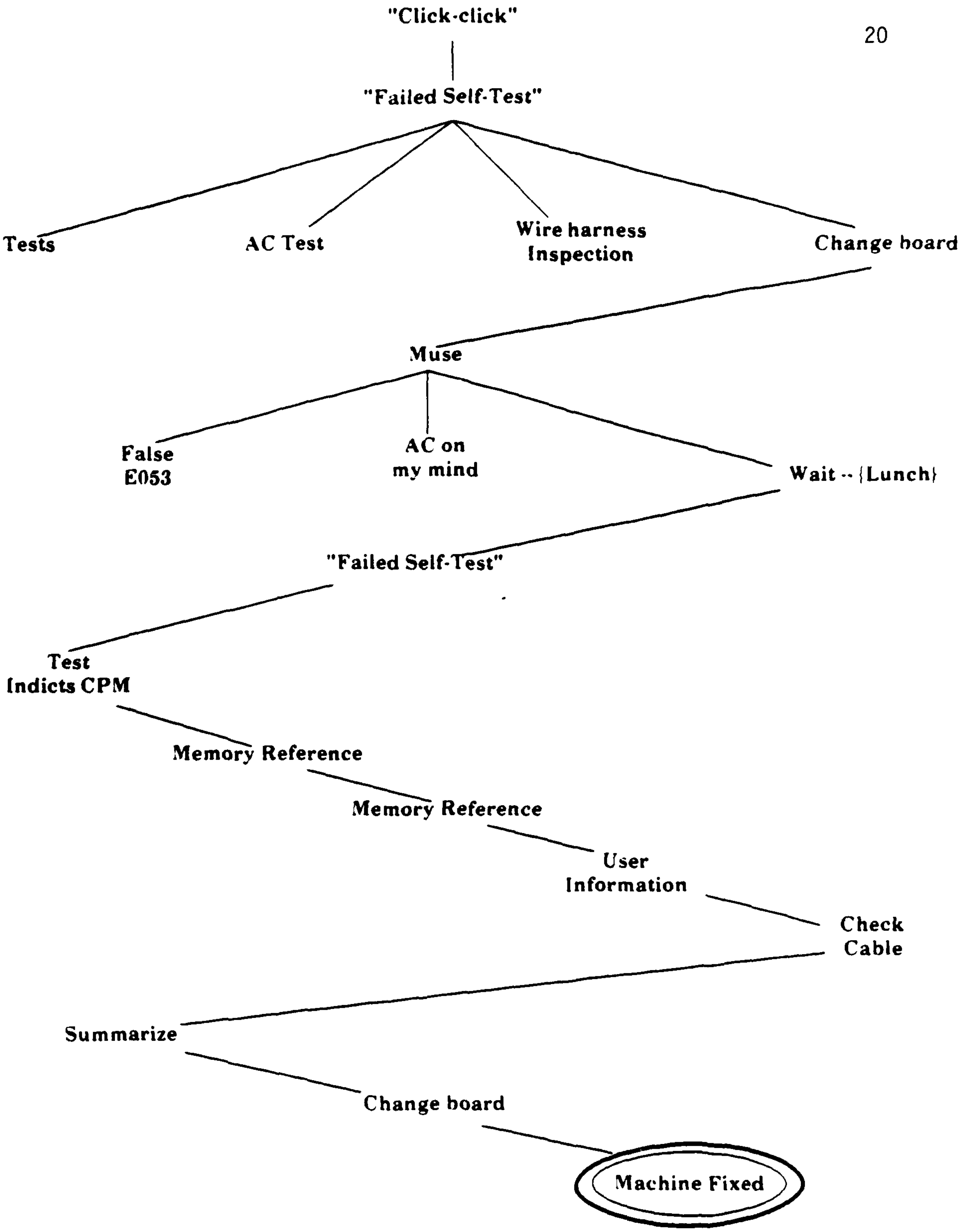


Figure 3. This shows the sequence of events and anecdotes during the trouble-shooting session described in the text. Events begin at the top, and progress down the page is progress toward the goal of fixing the machine. Events on a level happened in sequence from left to right, and represent no progress unless connected to a point further down the page. The "Click-Click" anecdote was told at the branch; all other events except lunch occurred at the site. The largest number of anecdotes was told during the Muse period. The False E053 story existed in contrasting versions, while AC On My Mind includes the series on AC and electricians.

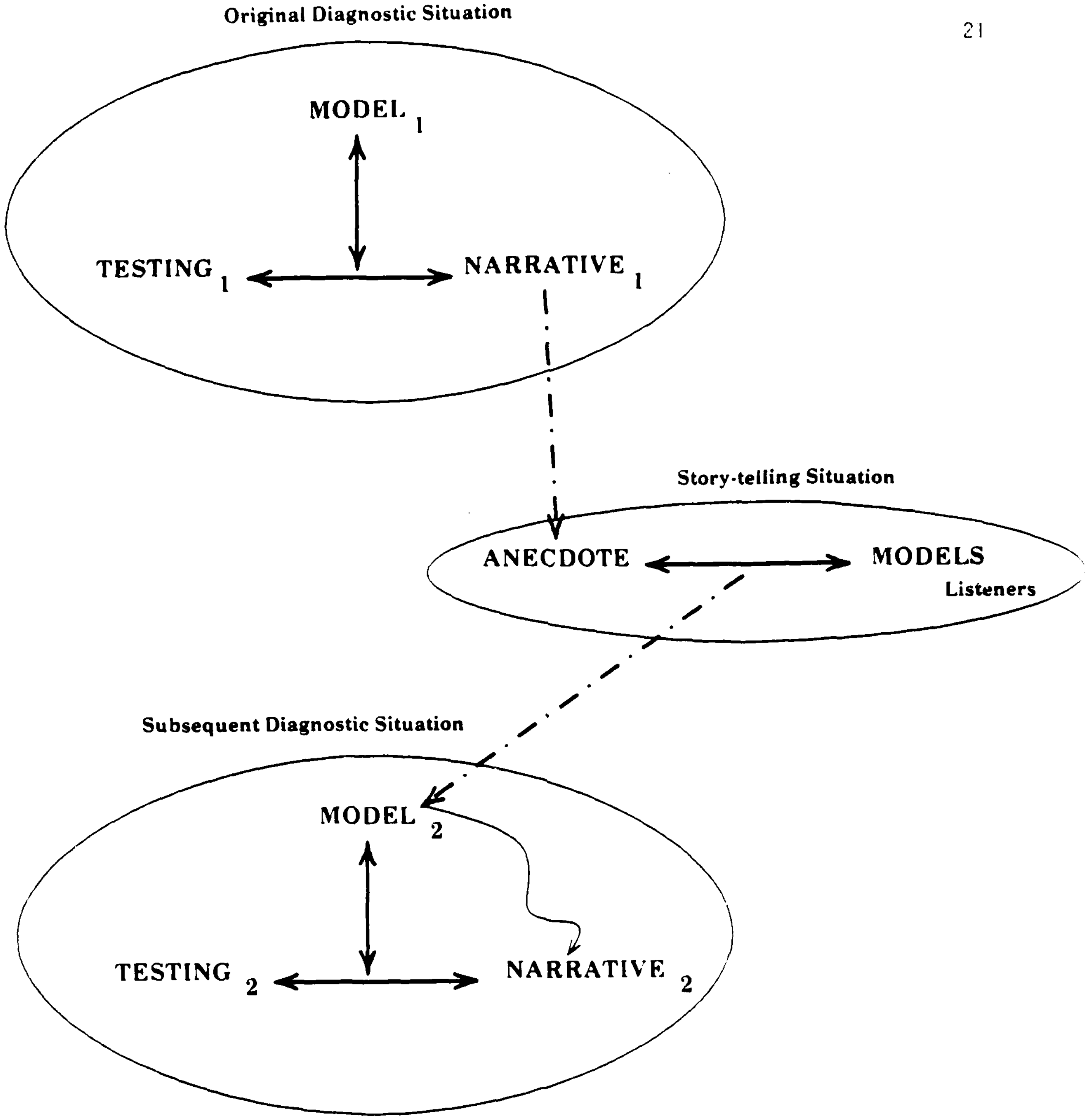


Figure 4 Shown here is the process of diagnosis and the mechanism by which the community acquires and uses its result. Diagnosis is defined as the process of reaching an understanding of the flawed state of the machine and consists of the interaction between performing tests and integration of the results of those tests through a narration of the testing and a verbal assessment of its results, informed by the technician's mental model of the machine. This produces a verbal description of the flawed state which may become the core of a war story or anecdote, told to one's colleagues and adding to their understanding of the machine. This both informs their model during subsequent diagnoses and may actually be told during the diagnostic process

Appendix B

B1 "It'll sit there and go . . . click-click"

Technical Specialist: Well, see, another thing that kind of blows me away, too, see, is if you've got a consistent problem such as that, a short, or something that's loadin', something's loading down, *the machine doesn't like to cycle up, initially.* It will sit there . it'll sit there and go click, click ,and then it will go through it again before . . at least once before it will come up from a power off. If you turn it on, it will sit there and go click, click, and instead of coming up to power it goes through it again--systems check again before it comes up. That tells me I've got a problem. Right there that tells me I've got a problem. I had that one in that damn dual sorter I put on there I was here until 6.30 last night trying to get that, and that was my fault because I didn't put a little jumper wire--I knew my symptoms, but I didn't know what the problem was. It would shut down intermittently too. It was an internal board problem though because I didn't put a little jumper wire on one of the plugs. To route some power going in the right direction. But that one there, you could drop it out, shut the machine off and turn it on and it would cycle up--it would have to go through the system check twice before it would come up. That tells me right there that I've got a problem.

B2 Dialogue during the checking of the AC

Senior Technician: Has input power anything to do with this? Do we have a power problem? Let's just hook this onto the power plug just to rule that out.

Technical Specialist: What the heck was that? See that little glitch?

Senior Technician: Were you on there--did you move it or anything?

Technical Specialist: It might have been me.

Senior Technician: Want me to turn it on to see if it draws anything across those two?

Technical Specialist. Give me some alligator clips.

Senior Technician. Pull the plug out before you stick them on. Is that the one that glitched on you?

Technical Specialist: Yes, that's the one that was giving me the trouble.

Senior Technician. Let's turn her on.

Technical Specialist: Go ahead. *(Power On. Warm-up noises)* If it just blipped it might--it would just shut it down like that and you have a power problem, yes. If it stayed off

Senior Technician: You wouldn't have an HO17 fault like that. A momentary glitch would give you a weird logic problem but a complete shut down loss along the lines would just shut the machine down.

Technical Specialist: Or a bad neutral. Neutral would give you more problems. I've had too many bad neutrals.

Senior Technician: That's a possibility with a brand new building. Woop, you just dropped it--or you took it off.

JO: He took it off.

Technical Specialist: The neutral isn't too swift.

Senior Technician: You went from neutral to . . .

JO: You're on ground now or neutral?

Technical Specialist: No, I'm on neutral.

Senior Technician: That's probably the heat rod pulses. Just regular clockwork, see that?

Technical Specialist: That's what that is. Course, the stupid thing won't act up while we're fiddling with it--you know that.

B3 The False E053 Error

Technical Specialist: See, this runs along with the problems we've run into when you have a dead shorted dicorotron. It blows the circuit breaker and you get a 24-volt interlock problem. And you can chase that thing forever, and you will NEVER, NEVER find out what that is.

Senior Technician: Yes, I know, E053, try four new dicors.

Technical Specialist: But, if you went in . . . Ok, you won't . . . You lose your 24, that's what it is. You're losing your 24-volt out of the power supply, but that's not what it's caused by. Now the key there, though, is when you pull up your DC20 log, you get hits in the XER board.

Senior Technician: Yeah. The other thing is as you're going on and on and getting E053s, you get, yeah, . . . F066 . . . in the sequence . . .

Technical Specialist: If you're lucky enough for it to run long enough, you'll get an FO66 problem which leads you back into the dicorotrons--you check them--yeah, I've got one that's a dead short. You change it and everything's fine, but if you don't . . . if you're not lucky enough to get that FO66 or don't look at the DC20 log; it's really a gray area . . .

Senior Technician: Well, DC20 logs . . . when I ran into that I had hits in the XER a few times previously, so I was tending to ignore it until I was cascading through after an EO53 which is primary, I'm cascading to see what else I've got--FO66--what the hell's this? Noise?

Technical Specialist: EO53, which one's that?

Senior Technician: Well, that's a . . . that's a 24 . . . lock.

Technical Specialist: 24 Interlock failure? Yeah. We did . . . I did that not knowing when they changed the circuitry in the XER board, normally if you had a shorted dicorotron, it'd fry the XER board--just cook it. Now they've changed the circuitry to prevent frying of that, but now it creates a different problem.

JO: This is with your dicor shorted to ground or . . . ?

Technical Specialist: Probably shorted to DC shield.

JO: Ah hah, yeah . . . yuck.

Senior Technician: Mm. I see.

Technical Specialist: OK, and then that goes (SNAP (fingers)), you know. That's where it's popping the breaker, and when it does that, that's when you end up with . . . through the boards, it pops it, before it pops the breaker, because you don't have any DC boards . . . you'll get a DC Interlock, 24-volt DC interlock failure. Now that came about after these boards came out and I've gotten burned twice on that same problem. I guess I . . . that four hours of sticking my head in the machine and tracing 24-volt interlock problem the first time didn't do it. The second time, it took me a long time, and it finally dawned on me -- what the hell am I doing -- get in there and that's what it was. Now I've had very intermittent dicorotron problem -- same thing -- guy sits there, and he says "I've got ", he says it's intermittent, once, maybe twice a day; you shut it off, turn it on, and it will run just a 24-volt interlock failure problem. So I asked him about the DC20 log and he said "Yeah, I had XER failure" -- I says OK -- he says, I checked all the dicorotrons-- I said, you're going to have to stress test 'em for a long period -- 4 to 5 minutes in the high you know, in your dicorotron checkout where you really, you've got the currents boosted up on 'em . and yeah, after about 4 minutes one of them (SNAP) glitched.

B4 Whirlpool

Senior Technician: Dropping out of voltage specs. Not a hell of a lot you can do. I know a computer problem, years and years ago, when I worked at Whirlpool, before I came to Xerox, they had a big hellacious computer that was tied in to all the districts and, uh, all the service repair parts came out of LaPorte, Indiana, for Sears and Whirlpool to the district warehouses and then, whatever there's a dozen of each, on each side of the house, and uh, all night long, the computer's accepting orders, from all these districts, and then . . there are some divisions and some distributors had daily shipments and some had every other day shipment, and when the damn computer went down in the middle of the night, and nobody was there to catch it, or they . . only two or three people to try and re-program it, get it back on line, they lost a lot of information. What they ended up doing, was running Northern Indiana Public Service power into the roof of the main building, into transformers, rectified it to DC, charged batteries, OK? . . Kept the batteries charged up, took the output off the batteries, and turned it back into 120 VAC to run the computers. Filtered it through batteries. And when they had power failures, they had gasoline generators up there that kicked in. And that's the only way they could get a steady 120 volts to run their computer banks. And talk about a hellacious expense, they put . . they added another story on to the top of the damn building.

B5,6,7 AC on my mind

B5 Technical Specialist: Yeah, so Wang is coming in and putting in filters or I don't know what all on the line to keep it steady. But they're . they say they won't do anything until we give them a hard copy of something, a monitor of . . a voltage recorder saying hey, this thing is fluctuating and you're going to have to fix it, 'cause nothing we can do about it.

- - - - - - - - - - - - - - -

B6 Technical Specialist: Well, I've fought with power problems too many times, I've even had to fight with an electrician, telling me he's got too small, you have to have too small a wire between here and the other side of the building, from where the breaker is, I mean, you're not getting enough juice to me, and I have to sit there and argue with an electrician! but that's what it was.

- - - - - - - - - - - - - - -

B7 Senior Technician: Electricians, habitually, will put a meter on an open circuit, nothing plugged into it, "There! 110! That's all you need!", and hook my supersucker [vacuum-cleaner] onto it, which draws 8 amps, er no, 5 amps (Technical Specialist: Five), 5 amps, and have that draw down to 90, and then hook 20 amps of warmup circuit off of a base 3100, little bitty copier, and have the breaker trip.

B8,9,10,11 **More AC: Neutrals and connectors.**

Technical Specialist: Oh, you end up it can be after the machine, you know, you've had the machine a while, and all of a sudden, you know, two weeks it's down there, after you've installed it, the thing's running, and it was an existing Xerox we've just replaced a, like, a duplicator or something, you plugged it in, it worked fine, checked out fine, two weeks down the road, the thing craps out, you go in and you've got a bad neutral, it's either in the plug or by the breaker, and it's crazy! I've run into more of it on this machine than I have on any of them. In fact, you know, I've run into a dozen of them since I've been working on this, where I bet I haven't had a dozen in ten years.

- - - - - - - - - - - - - - -

B9 Technical Specialist: Well, drive me crazy. I had it smoke a I had a loose neutral at the breaker box smoke about six inches of the neutral line . .

- - - - - - - - - - - - - - -

B10 Senior Technician: I once had a building electrician at uh, one of the bank buildings up San Mateo area, had a building electrician did all of the light bulbs and everything in the building, right? Supposedly an electrician, tell me that hey, if that machine was running on that, it's been installed there, that same plug for years, he said it can't be the plug. Plugs don't have no moving parts; it can't wear out . . . Right, but I got across that 115 to, uh, neutral, well, AC Neutral to ground, the safety wire, and you'd put your alligators on it and plug it into the wall, so that you can read zero, right, and you sit there and wiggle it, and you go 60, 20, hundred {Laughs} . . . .

- - - - - - - - - - - - - - -

B11 Technical Specialist: *No, it was . . when I got hold of the electrician up there and he pulled . .* pulled the receptacle out and the whole side of it was all smoked, just looked like it melted, it got hot and melted. I don't know, internally, I don't know why I didn't even screw with it, he just threw another one on it and took care of it.

B12 **"I have seen that . . . "**

Technical Specialist: You know what can happen and I've seen it -- that ribbon cable shorts and you blow the CPM Board.

JO: Is that the ribbon cable you replaced?

Senior Technician: Mm-hmm.

Technical Specialist: I have seen that you can trash that cable out and it trashes the CPM Board. So that we know the cable's good but we may still get the same indications cause the damn board's trashed. Strictly guesswork but I have seen that.

B13 **"Board's gone."**

Technical Specialist: So I've seen where the cable shorts, say, blows the light, here, everything works except the light don't work, put a cable in it, still don't work. Board's gone. Got to change the CPM board for a light. You know, they changed the cable and it didn't do it, they change the board here [front panel] and it don't take care of it, change the CPM board and that takes care of the light. Have you still got that old cable? I'd be real curious to see what lines those were on

B14 The Summary

Senior Technician: She said that when you reduce

Technical Specialist: . . . It ties it, original problem was wire tape shorting,

JO: . . . which stressed the drivers on the board . .

B15 Becoming a Member of the Team

"Until you really break something good, you're not a member of the team. I became a member of the team the day I smoked a transformer because they wired the, uh, cord at Lockheed . they put different caps on the cords. They wired it neutral on a hot line, and I put in the machine with the neutral on a hot line, and a hot on the neutral. They didn't like blue for the color code of neutral, and they wrapped white tape around a brown (or maybe ground) lead. And I installed it, dutifully, and didn't know there was a, uh, an actual program for installing the cord. And I didn't trust them, and checked the wall, but it did not occur to me to check the terminals, right, and that of course is something that I'll never do again. You should have seen it: I had my back to the machine and smoke is rolling out of the card cage. And the lady says 'OHHH, I think we have a fire' And I say 'Oh, no problem' and went over and unplug it and said 'OK. Now I am a member of the team.' All it took out was, probably all of the things in the Low Voltage Power Supply that had to be replaced, probably only that one transformer was gone. But, uh, I was impressed that the machine was able to, well, I ran, I ran, my vacuum cleaner that's designed for 1 . . . , my Super Sucker, it's designed for 115, I ran it on 220 at the utility plug."

IV Instructional Design and Delivery

Thomas P. Moran

Daniel M. Russell

Xerox Palo Alto Research Center
Intelligent Systems Laboratory

July 1987

This work was partially sponsored by a research contract
MDA903-83-C-0189 from the Army Research Institute to Xerox

DRAFT: 15 January 1987

The Problem Space of Instructional Design

Peter L. Pirolli and James G. Greeno

University of California, Berkeley

Acknowledgements. We would like to thank Andy diSessa, M. David Merrill and John Mitterer for comments on earlier drafts of this paper.

Abstract

Instructional design can be viewed as a problem solving task in which instructional design processes must choose among various design alternatives to produce instruction. We propose an *instructional design problem space*, which characterizes the alternative decisions and methods in instructional design, as a foundation for a framework for the analysis and study of instruction. We propose that the instructional design problem space involves nine subspaces which are the cross-product of three levels with three sets of issues at each level. The levels involve, (a) global issues that correspond roughly to issues involved in designing whole courses, (b) intermediate issues that correspond roughy to issues in designing lessons, and (c) local issues that roughly correspond to issues about specific presentations. The issues at each level involve (a) goals and constraints, (b) technological resources, and (c) theoretical resources. Each of these nine subspaces is discussed along with examples. We also discuss theory and methodology in our framework. In particular, we focus on how a problem solving approach to instructional design may refocus our ideas about prescriptive approaches to instructional design, on the relationships between theories of instruction and theories of learning, and the role of intelligent tutoring system technology in research on theories of learning, instruction, and instructional design.

The Problem Space of Instructional Design

I. Introduction

We propose a framework for analysis and study of the design of instruction. One objective is to contribute to the formulation of general principles in the field of intelligent tutoring systems that will facilitate systematic and cumulative progress in research. Another related objective is to understand relations between work on instructional design conducted by different groups of researchers, including intelligent tutoring systems, instructional-design scientists (e.g., Gagne & Briggs, 1979; Reigeluth, 1983a), and computer-assisted instruction (e.g., Bork, 1981; Heines, 1984; Suppes, 1981).

This framework considers instructional design as a problem-solving task, centered on a characterization of decisions and methods in what we call the *instructional design problem space.* Instructional designs are artifacts that are the products of design processes. General features of designed systems were discussed by Simon (1969), using classical problem solving terms (e.g., Newell and Simon, 1972). An instructional design, like other designs, involves selecting among alternative *means* to achieve certain *ends*. These ends are instructional goals that we wish to achieve under specified *constraints* of the instructional situation. These general terms have also been used in discussions of instructional design (e.g., Landa, 1983; Reigeluth, 1983b). If we consider the task of instructional design as a problem-solving process, we are led to formulate the task environment of instructional design using concepts that have been useful in analyses of other problem-solving tasks. In particular, it is important to characterize the *problem space* of instructional design.

In general, the problem space of a task characterizes the alternatives that a problem solver has available and the various states that can be produced during problem solving by the decisions that the problem solver makes in choosing among alternatives. As instructional scientists have recognized, the design of instruction involves alternatives and decisions at several levels or subspaces.

We have found it useful to distinguish nine subspaces, involving different levels and aspects of instruction, shown in Figure 1. The levels involve (A) global issues, (B) issues of intermediate generality, and (C) local details. These levels correspond approximately to issues in designing courses, designing lessons, and designing specific presentations and activities.

Insert Figure 1 about here

The issues at each level involve (1) goals and constraints, (2) technological resources, and (3) theoretical resources. These three classes of issues are interrelated, with technological resources providing the means for achieving goals and satisfying constraints, and theoretical resources providing reasons and explanations for decisions that are made in selecting from alternative goals and methods.

We consider the contribution of theories to design as a set of resources, not a set of prescriptions as they often are considered. To be sure, theoretical principles can have prescriptive force, as they do when they specify conditions that are required to achieve certain results. Another function of theory, however, is to provide analyses that identify and clarify alternative forms in which learning and instruction can be designed. Then the theoretical resource expands the space of possibilities in which the designer works, and principles link these possibilities to the decisions that the designer is able to make.

The reasons and explanations for instructional decisions, drawn from theoretical resources, can be included in a document along with instructional materials, and the Instructional Design Environment, described by Russell and Burton (this volume), is intended to facilitate construction of instructional designs that enable other designers or instructors to understand the design principles that provide reasons and explanations for the instructional materials.

A problem-solving framework for instructional design allows consideration both of the products of design, instructional materials, and the processes of designing instruction, in an integrated way. This paper is concerned mainly with characteristics of instructional materials and activities, but we also consider the framework potentially useful as a basis for empirical research about problem-solving processes and knowledge of involved in the task of designing instruction.

II. The Problem Space

In this section we sketch the main components of our characterization of the problem space for instructional design, organized according to levels of generality of the issues.

II.A. Global Issues. A general set of decisions is made, deliberately or by default, to determine the main content of instruction and the general goals that instruction will attempt to achieve.

II.A.1. Global goals and constraints. In most discussions of instructonal design, the goals of instruction are to transmit knowledge of concepts, principles, and specific skills to students. There are important alternatives to these essentially didactic goals of instruction and learning that can be considered. These include abilities for formulating questions and arguments in a domain, attitudes and beliefs about the domain, general reasoning skills, and metacognitive capabilities.

Goals of subject-matter content are often considered as "given" for the process of instructional design, based on judgments of experts in the subject-matter domain. Choices must be made, however, and these reflect judgments about the value of various aspects of knowledge in the domain -- for example, is it more important for students to know a broad set of facts and principles of a domain or to acquire a deeper understanding of a few topics? Recent research on misconceptions has emphasized that students may complete instruction that presents a set of concepts, but still understand the domain in terms of different concepts that they had before the instruction. Alternative ways to organize subject-matter concepts should be considered in planning instruction in a domain to make its structure more likely to result in acquisition of integrated understanding by students, including a correct set of intuitions and connections between their informal and formal knowledge.

Choices of subject-matter content also are influenced by constraints, regarding the capabilities of students to understand and learn, as well as relations of the material to other instruction that has preceded or will follow the material that is being designed.

II.A.2. Global technological resources. Until recently, theories of instructional design assumed that instruction would be presented to groups of students, largely in the form of lectures, demonstrations, and texts, or in a limited form of tutorial instruction in which the tutor guides a student through an organized body of concepts and information. Early forms of computer-assisted instruction presented information and exercises in a domain in the form of simple branching programs. These technologies of presenting instruction are appropriate for the goal of students' acquiring the content of a subject-matter domain, but are probably less effective for broader cognitive goals of instruction, such as forming questions and acquiring abilities to form judgments in a domain, acquiring general reasoning skills, and integrating new concepts with students' existing informal knowledge and understanding.

Alternative environments for instruction have included laboratories in which students perform experiments or otherwise gain practical experience, group discussions where students pose questions as well as give answers, and peer teaching situations where older or more knowledgeable students share the responsibility for instruction. More recently, computational systems have been developed in which students can engage in dialogues about physical systems (e.g., Bork, 1981).

Recent research has provided three futher kinds of instructional environments. One class of environment involves an exploratory microworld, where students can manipulate objects in a computational system that is designed to embody a set of theoretical principles. Examples of exploratory instruction include Dugdal's (1982) game "Green Globs," for algebra, in which students write expressions for functions so their graphs pass through sets of points that are displayed, diSessa's (1982) system Dynaturtle, in which objects move according to Newtonian principles, and Uarushalmy and Houde's (1986) Geometric Supposer in which students explore conjectures about objects and properties in Euclidean geometry. Another kind of environment involves a kind of apprenticeship, in which a teacher first models behaviors that he or she wants students to emulate and then coaches students as they work to acquire the skill. Examples include Palincsar and Brown's (1984) method of reciprocal teaching, in which students observe and then carry out activities of asking questions, summarizing, and other metacognitive activities involved in successful comprehension, and Schoenfeld's (1985) method of teaching problem solving in mathematics in which the teacher demonstrates and encourages active monitoring of the progress that students are making on problems, rather than focusing entirely on the final solutions that are achieved. A third kind of learning environment emphasizes collaboration, either among students or between the teacher and students on intellectual goals that they share. Examples include Fawcett's (1938) course in geometry in which the students and teacher worked together on developing the definitions and axioms of formal reasoning both in geometry and in other reasoning domains, and Lampert's (in press) method of conversational teaching in which she and her students work together on the task of making sense of elementary mathematical notation and procedures.

II.A.3. Global theoretical resources. Theoretical resources provide reasons and explanations for choices that are made in the design of instruction. The theories that apply to global features of instruction are in the domain of epistemology, involving general analyses of knowledge, including forms of knowledge and values of knowledge in different settings.

Theoretical resources for global features of instruction include empirically based theories that identify alternative forms of knowledge and learning and their general characteristics. Important examples include recent analyses of qualitative reasoning in formal domains such as physics (e.g., deKleer & Brown, 1984; Forbus, 1984) that provide rigorous hypotheses about characteristics of knowledge that enables reasoning about causal relations that is different from use of formulas. Studies of conceptual growth such as that of Carey (1985) provide understanding of ways in which knowledge in a domain can involve changes in theoretical principles that are used in interpreting a broad range of phenomena. Studies of reasoning in practical settings such as those of Lave, Murtaugh, and de la Rosa (1984) show characteristics of knowledge used in situations where problems are structured by social environments rather than by the organization of academic disciplines.

Epistemological principles also are involved in important dimensions of individual difference. Metacognitive beliefs about the nature of knowing and learning differ significantly among individuals, varying from belief that one's learning involves receiving truth from authorities to belief that knowledge is constructed through an active personal and interpersonal process (Belenky, Clinchy, Goldberger & Tarule, 1986). Individuals differ in what it means to know a subject, varying from a criterion of being able to solve text problems correctly to grasping the relations among concepts (diSessa, 1985; Schoenfeld, 1985).

Philosophical analyses of knowledge also provide important resources for choosing and justifying general goals of instruction. An example is Kitcher's (1984) study of mathematical knowledge, focused on a concept of knowing a mathematical practice, including mathematical language and established results, along with accepted questions, accepted reasonings, and metamathematical views that include an understanding of the goals of mathematical study. Values of alternative kinds of knowledge are also important, as Posner and Rudnitsky (1986) noted.

II.B. Intermediate Goals and Constraints. An instructional designer constructs a set of lessons and activities arranged in a sequence, possibly contingent on student progress. Choice of these units and their sequential arrangement is a major subject of many discussions of instructional design (e.g., Gagne & Briggs, 1979; Gropper, 1983; Landa, 1983; Mager, 1962; Posner & Rudnitsky, 1986; Reigeluth & Stein, 1983; Scandura, 1983).

II.B.1. Intermediate goals and constraints. Choosing the subgoals of a lesson is analogous to choosing the components of a device that one wants to build, or deciding about the parts of an essay that one wants to write. The parts of a lesson involve knowledge of components of subject matter and components of a cognitive skill. A coherent and progressive sequence of presentations and activities is constructed that will result the knowledge and cognitive capability that the instruction is intended to provide. If the instruction is interactive, as in tutorial instruction, the designer constructs a set of instructional activities and a set of rules that will determine the sequence that occurs in the instructional situation. In instruction that is provided by human teachers, the instructional plan usually does not specify rules for proceeding through the lesson, but the teacher's knowledge includes a rich set of routines and situated skill that determines the sequence of activities in the instructional context (cf. Leinhardt & Greeno, 1986).

Constraints on the subgoals and sequence of instruction are provided by relations among the units of knowledge, including necessary or facilitating prerequisites for learning new material or acquiring new components of skill.

II.B.2. Intermediate technological resources. The main resources for constructing lessons are known components of subject-matter knowledge and classes of problems and questions that students are asked to solve and answer. In many cases of instructional design, these resources are obtained from subject-matter experts. Subject-matter experts also can provide information about relations between tasks, guiding arrangements of instructional sequences and tests for progress that are used in branching CAI programs.

Instruction that uses games, exploratory environments, and collaborative work requires resources in the form of activities that are given or suggested to students. The activities provide goals for the student's exploration of the environment or the collaborative work that the students engage in together or with the teacher.

A broader range of resources is also becoming available through developments in cognitive research and artificial intelligence. These resources include cognitive models of successful performance of instructional tasks. These models can suggest tasks that focus on aspects of performance that are usually not instructed explicitly, such as the information required to understand word problems or the valid transformations of arithmetic expressions that preserve value (Greeno et al., 1986). Such models can also be used as expert problem solvers in tutorial systems to provide correct solutions that are compared with solutions provided by students. Examples include the Geometry Tutor and the LISP tutor, developed by Anderson and his associates (Anderson, Boyle, & Yost, 1985; Pirolli, in press; Reiser, Anderson, & Farrell, 1985), which provide detailed feedback using a scheme called model tracing in which each step of a student's performance is compared with the performance of the model. Models of successful performance also are used in tutorial systems for teaching proof skills in logic (Suppes, 1981) and for teaching programming in PASCAL (Bonar, this volume). These tutorial systems use the models to provide guidance and feedback to the students at the level of their goals and plans, as well as at the level of specific steps in solving problems.

More sophisticated resources have been developed in research on intelligent tutoring systems, including diagnostic systems such as BUGGY (Brown & Burton, 1978) that are based on analyses of a space of possible procedures for performing the task, including correct performance as well as variations that produce systematic incorrect performance, and coaching systems such as WEST (Burton & Brown, 1982) that diagnose students' strategies and provide hints and suggestions at the level of strategic performance.

Resources for designing lessons also include systems of representation that are used to display information to students. Traditionally, systems of representation have included demonstrations of laboratory phenomena, pieces of equipment or models of systems that the teacher is discussing, and concrete materials such as place-value blocks for displaying abstract relations in concrete form, in addition to the standard presentation media of texts and diagrams.

Recent additions to the armamentorium of representation systems include computer displays that simulate systems that students can interact with, and that provide graphical displays of data taken from real or simulated experiments. Additional systems of representation have been developed to display information explicitly to students that is usually implicit, embedded in the process of understanding or reasoning about problems. Examples of these include graphical displays of the steps taken in solving problems in algebra (Brown, 1983) and in geometry (Anderson et al., 1985). Additional examples include graphical displays of semantic relations in texts of word problems (Greeno et al., 1986).

II.B.3. Intermediate theoretical resources. The theoretical resources relevant to designing lessons include principles for analyzing the structure of subject-matter domains and requirements of instructional tasks for the purposes of learning and instruction. The methods of behavioral task analysis provide the theoretical basis of much of the designed instruction that has been developed in the past 20 years.

An alternative set of principles involves the structure of concepts and principles in the subject-matter domain, rather than in behaviors that students have to perform. Efforts to redesign the school mathematics and science curricula according to principles of subject-matter structure were pursued energetically in the 1950s and 1960s, with important innovative materials produced by groups such as the School Mathematics Study Group and the Project Physics Group. Recent discussions of course organization such as Reigeluth and Stein's (1983) elaboration theory have developed ideas about relations between topics according to a scheme involving degrees of generality of material, defined differently for different kinds of content.

Recent research has provided enhanced resources for analysis of cognitive requirements of instructional tasks. Cognitive models of knowledge structures and cognitive processes used in solving problems and answering questions have provided explicit hypotheses about requirements of successful performance that are usually tacit, and these hypotheses have been used as the basis for instruction directed at specific sources of difficulty in performance. Advances have also been made in developing models of the process of learning specific skills, and these provide guidance in the design of rules for feedback to students. In addition, important advances have been made recently in understanding misconceptions about theoretical concepts and principles, especially in physics. Efforts can be made now to develop new principles of organization for the content of subject-matter domains in ways that are more coherent with students' intuitive understanding and with natural progression of conceptual growth.

II.C. Specific Presentations and Activities. When instruction occurs, information is presented in specific texts and vocal presentations, conversations, diagrams, and specific tasks and questions.

II.C.1. Detailed goals and constraints. The instructional designer or developer constructs these specific presentations and activities, including such details as the size of print and arrangement of material in text or screen displays, and the amount of time that will be spent on specific problems or demonstrations. Relevant considerations include clarity of presentation, ease of reading type, consistency of layout, and other concerns that influence the ease of understanding and interaction with information systems.

II.C.2. Technological resources for presentations. Resources for designing specific presentations and activities are provided in the fields of graphics design and human-computer interaction (e.g., Norman & Draper, 1986), including discussions that are specifically oriented toward instructional materials (e.g., Heines, 1984). Recent discussions by Newell and Card (1985) and Pirolli (in press) have pointed out how intelligent tutoring systems offer significant opportunities for research in human-computer interaction by providing tools for gathering and modelling data on the communication of knowledge in the computational medium.

II.C.3. Theoretical resources for presentations. Principles of design for specific instructional transactions have been discussed in instructional science, notably by Merrill (1983), in his component display theory. Merrill's scheme distinguishes types of content and types of performance. Types of content are facts, concepts, procedures, and principles. Types of performance include remembering an instance or a generality, using a generality, and finding a generality. The theory was developed largely with a style of instruction involving expositions and inquisitions in mind, but Merrill (in press) has recently generalized the theory to deal with more experience-based learning of the kind found in intelligent tutoring systems that use expert and student models, interactive simulations, expert demonstrations, and coaching.

An issue concerning the human-computer interface that we consider important but relatively unexplored comes from recent developments in the theory of conversation. The communicative structure of human-computer interaction is generally considered in terms of either a student model in an instructional system or a user's model of a piece of technology, complete with "gulfs" between the user and the system (e.g., Hutchins, Hollan, & Norman, 1985). Analyses of communication in conversations by social scientists (e.g., Schegloff, 1972) and psycholinguists (e.g., Clark & Wilkes-Gibbs, 1986) present a more subtle and complex picture. Functions even as simple as reference to places, persons, and objects are achieved cooperatively, as part of a process of creating a shared pool of information, understanding, opinion, and belief. In the process, both partners know what is being said, and each one also knows that the other knows what is being said. An implication is that we should begin paying close attention to the student's understanding of the instructional system, in addition to the instructional system;s understanding of the student's state of knowledge. A tutor should provide indications that what the student has said or done has been understood, and the student should have a model of the tutoring system that makes those indications comprehensible. Communication between human teachers and students depends on shared knowledge about each others' cognitive states and teachers spend considerable effort teaching their students routines for classroom activities (Leinhardt, Weidman, & Hammond, 1984). Similar efforts to instruct students in how to use particular features of a computational instructional system seem warranted.

III. Theory and Methodology in the Instructional Design Problem Space Framework

We propose that instructional design be considered as a problem-solving task, and we have outlined a characterization of the problem space for instructional design that seems to us to make such a consideration feasible. We turn now to a discussion of some issues in the theory of instruction that we believe are clarified by taking the point of view of instructional design as problem solving.

III.A. <u>Prescriptive and Descriptive Principles.</u> One familiar distinction is that between *descriptive* and *prescriptive* theories (e.g., Landa, 1983). Descriptive, or natural sciences are concerned with descriptions of how things are; with accumulating declarative statements that characterize our world. A descriptive science of instructional design is concerned with statements of the form "if constraints C hold and actions A are taken then outcome G will occur." In contrast, prescriptive sciences are concerned with how artifacts ought to be in order to achieve certain goals. A prescriptive science of instructional design is concerned with imperative statements of the form "in order to achieve goal G under constraints C then perform actions A" (Landa, 1983). One of the important outcomes of Simon's (1969) analysis of design science is to point out how the classical problem solving framework reduces the imperative logic of design to the declarative, model-theoretic logic of natural science.

Considering instructional design as a form of problem solving enables the use of some general prescriptive principles of efficient problem solving as recommendations for instructional designers. These include heuristic principles that contribute to efficient search, such as identifying relatively decomposable components of the task and dealing with them as subproblems, considering more constrained aspects of the task before considering less constrained aspects, and delaying commitments to detailed implementations when higher-level goals can interact. These considerations discussed in the theory of planning, a well developed field of research in artificial intelligence (e.g., Cohen & Feigenbam, 1982).

The prescriptive view of instructional design suggested by the problem-solving perspective differs significantly from the sequential process that is often presented (e.g., Mager, 1962). Viewed as problem solving, the appropriate prescriptions are heuristics that can make the process more efficient and make a better solution more likely, rather than a sequence of steps that should be followed.

We expect that the problem-solving perspective also will favor productive descriptive studies of instructional design as a process. One promising approach would be to analyze the performance of designers to infer characteristics of their problem spaces, using and extending the methods of Newell and Simon (1972) and Kuipers and Kassirer (1984) in the use of protocol data for identifying the concepts, operators and constraints that problem solvers use.

III.B. Principles of Instruction and Learning. It has been pointed out often (e.g., Landa, 1983; Reigeluth, 1983b) that theories of instruction are not theories of learning. The critical feature in this distinction lies in the description of means to achieve goals. Landa's (1983) version of the distinction is typical. Instructional theories are concerned with how actions taken by an instructor achieve instructional goals. In contrast, theories of learning are concerned with how actions taken by a learner achieve learning outcomes. While we agree that this distinction is correct and useful, the distinction could be taken as a warrant for developing instructional theories that are independent or only loosely related to learning theory (e.g. Landa, 1983; Merrill, 1983; Reigeluth, 1983b).

We prefer a view in which instructional and learning theories are linked closely, partly for the development of more effective instruction and partly because a close relation to instructional practice is beneficial for the progress of learning theory. One way to consider this relationship is by using learning theory as a source of constraints in the problem solving task of instruction. In general, one can consider a subset of the constraints in the problem space of instruction to be natural laws that are relevant to the task. For example, there is an upper bound on the amount of information that can be perceived visually by a human looking at a display screen (Card, Moran & Newell, 1983). Learning theory can be used as a specification of a subset of the natural laws that are relevant to the instructional task and consequently constrain the sorts of instructional methods that are used.

Another way to consider the relation of learning theory and instructional theory is to use principles of learning as theoretical resources that provide guidance and explanations for the selections of methods to achieve certain instructional goals. Cognitive science offers tools for modelling the states of student knowledge throughout the instructional process and the processes of transition between states. The problem for the instructional process is to move the student from some current state of knowledge to a goal state of knowledge. The transitions between these states, in principle, can be predicted by learning theory given the student's experience and current knowledge state. In turn, we could, in principle, lay out the relations between instructional methods and student experiences. For example, we might lay out a space of alternative methods with probability distributions that tell us (a) how likely the methods are to produce the target student experience and (b) how likely the experience is to produce the target state of student knowledge.

It is apparent that different learning theories are explicitly or implicitly at the root of some differences among intelligent tutoring systems. For instance, PROUST (Johnson & Soloway, 1983), the BRIDGE Tutor (Bonar, this volume), and the LISP Tutor (Pirolli, in press; Reiser et al., 1985) are all targeted for instruction in introductory programming and use essentially the same representation of student knowledge.[1] PROUST and the BRIDGE Tutor use a goals and plans analysis of programming skill and misconceptions whereas the LISP Tutor uses a production system scheme to represent such knowledge. Both of these representational schemes can be considered variants of the GOMS model of human information processing in computer related tasks (Card et al., 1983).

Despite these similarities, the instructional methods used in these systems are quite different. The LISP Tutor makes explicit use of a set of instructional principles (Anderson, Boyle, Farrell, & Reiser, 1984) grounded in the ACT* learning theory (Anderson, 1983). Some of the differences among these systems can be examined in light of these principles. For example, one principle followed by the LISP Tutor is to provide instruction and feedback in

[1] PROUST and the BRIDGE Tutor are used to teach introductory Pascal whereas the LISP Tutor is used for introductory LISP.

the context of problem solving. In contrast, PROUST takes syntactically correct programs that have already been written by a student and searches for semantic errors in the code and relates these errors to its catalogue of misconceptions. The BRIDGE Tutor is like the LISP Tutor in this respect. It provides feedback in the context of problem solving, but in contrast to the the LISP Tutor it does not enforce a top-down refinement of programming goals. Rather, the BRIDGE Tutor proceeds by successively modifying a naive plan for an algorithm into an explicit program. The LISP Tutor is based on a principle of presenting the goal structure of the task whereas the BRIDGE Tutor appears to be based on the principle of illustrating how naive goals and plans can be transformed into programming goals and plans.

Another contrast can be made between Algebraland (Brown, 1983) and the LISP Tutor. Algebraland enables a student to reflect on the sequence of decisions that he or she made during an attempt to solve an algebra problem by providing a visual representation of the search space and searching erroneous solution paths is allowed. Similar search information is displayed in the LISP Tutor, but students are kept on a correct solution path. The LISP Tutor uses an instructional principle of immediate feedback for errors (Anderson et al. 1984) that is grounded in skill acquisition research (Lewis & Anderson, 1985) whereas Algebraland seems to employ a principle stating that reflection on erroneous solution paths leads to learning about the search space.

III.C. Design Processes and Products. Our discussion, so far, has largely focused on characteristics of instructional *products*. That is, we have discussed various desirable goals and the instructional methods that can be used to achieve them. We can also view instructional design as a *process*. Viewing design as a process shifts concern from the study of effective and efficient pieces of instruction to the study of effective and efficient means for producing optimal instruction. In the world of instructional design science, a variety of prescriptive process models (e.g., Mager, 1962) have been proposed to organize and optimize the design process. These models are basically flow charts of steps to follow--performing task analyses, constructing tests, etc. In our informal discussions with instructional designers in industry who explicitly attempt to follow such models, we have found that the flowcharts act as a general and rough guideline, however, the actual process of search for a design solution is much richer and complex than suggested by such design models.

Although we can distinguish between the processes and products of design, it is crucial to note that they are interrelated. Essentially, the relation is an extension of the notion that the representations used in problem solving have a substantial impact on style or form of problem solutions. For example, concepts and principles in instructional design science have led to Mager's (1962) prescriptive model of instructional design. On the other hand, concepts and principles from cognitive skill acquisition theory have led to a class of instruction called model tracing used in several intelligent tutoring systems (Anderson et al., 1985; Pirolli, in press; Reiser et al., 1985).[2] As in other engineering disciplines, it is important to have design concepts that allow the designer to manage the complexity of her design problem through useful abstractions while leading ultimately to an efficient and effective design solution.

Integration of the processes and products of instructional design is a major motivating idea for the Instructional Design Environment, described by Russell and Burton (this volume). IDE provides a notation and an electronic environment for construction of documents that include instructional materials along with reasons and explanations for the decisions that a designer makes in the process of designing the instruction. Including the principles drawn from theoretical resources in a document along with the instructional materials can provide a basis for reflective evaluation, criticism, revision, and comparative analysis of instructional materials.

Given the set of distinctions we have discussed, it is interesting to note a major gap in research on instructional design. There are descriptive and prescriptive theories of instructional design products and there are prescriptive theories of the instructional design process (e.g., Mager, 1962), but few, if any,

[2] No prescriptive process model is associated with the model tracing approach.

descriptive theories of what takes place in the instructional design process. We suggest that our understanding of prescriptive approaches to the process of instructional design can benefit from detailed studies of how instructional designers do in fact design instruction.

III.D. Relation to Other Frameworks. Our characterization of a problem space for instructional design makes many of the same distinctions that have been made in previous discussions of instructional design science. Comparison of our scheme with a recent discussion by Reigeluth (1983b) may be helpful. Figure 2 shows a diagram used by Reigeluth (1983b) to present a framework of variables that are important in instructional design.

Insert Figure 2 about here

Our distinction between levels of decisions about sequences of lessons and activities and specific presentations and tasks is much the same as Reigeluth's distinction between macro and micro strategies. Micro-strategies are methods for organizing instruction about a single idea (e.g., Merrill, 1983). Macro-strategies are methods for organizing instruction about several ideas (e.g., Reigeluth, & Stein, 1983).

On the other hand, there are several ways in which our problem-space view refocuses aspects of the design problem. First, we consider decisions about goals of instruction to be an integral part of the design process. Decisions about which topics to include and the general objectives of instruction seem to us to be issues that should be addressed in the context of instructional design, rather than considered as "givens" in the design environment.

We also understand management strategies to be integrally connected with the organization of subject-matter. Decisions about management (e.g., how and when to individualize instruction) are included in the process of deciding on instructional environments, sequences of topics and activities, and detailed presentations of materials. Delivery strategies are similarly integrated into the process of deciding about goals and sequences of topics and transactions.

III.E. Methodological Role of Intelligent Tutoring Systems. Instruction, learning, and instructional design are domains of extremely complex phenomena, and productive research in these domains requires matching complexity between theory and gathered data. Intelligent tutoring systems offer a technological resource that can be exploited to increase the complexity of theory and methodology.

Intelligent tutoring systems can drive theoretical development in at least two significant ways. First, by analogy to the role of computer simulations in developing models of performance, intelligent tutoring systems reify theories of instruction and learning, and facilitate the management of evermore detailed and complex theoretical formulations. Second, intelligent tutoring systems act as sophisticated data gathering tools that test the theories that they instantiate.

In our framework, a design process searches through the instructional design space selecting instructional means that achieve desired outcomes under certain constraints. In effect, the design process is making a prediction that a particular instructional product will achieve a certain possible world specified by the constraints and desired goals (cf. Simon, 1969). The standard validational logic of science thus applies: if the actual world differs from the predicted world then something is wrong with the theory underlying the design.

To use a concrete example, the LISP Tutor was designed to optimize the efficiency and efficacy of instruction in introductory LISP programming. Descriptive models of instruction and learning based on the ACT* theory plus a set of instructional prescriptions were used to generate and select means for optimizing this instruction (e.g., Pirolli, in press). The prediction is therefore that the LISP Tutor stands as the king of the mountain with respect to instruction in LISP. Observations gathered in studies of the LISP Tutor can be used to revise the underlying models of cognition, learning, and instruction used to produce the LISP Tutor. Similarly, experimental contrasts of the LISP Tutor with other systems such as PROUST or the BRIDGE Tutor might invalidate the king of the mountain claim.

It is important to note, however, that we are likely to need a more sophisticated methodology than simple experimental contrasts of systems or instructional principles, given the richness of competing theories and data. For instance, the hardcore empricist approach to determining the validity of the eight instructional principles outlined in Anderson et al. (1984) would involve performing a factorial designed experiment which contrasts all possible combinations of instantiating or not instantiating a principle in a piece of instruction. This would involve contrasts among 2^8 different pieces of instruction and would still ignore many other factors that are involved in the design of a system as complex as the Geometry Tutor or the LISP Tutor. It seems that more detailed and sophisticated methods of comparisons along the lines of competive argumentation (Van Lehn, Brown, & Greeno, 1984) will be needed.

Intelligent tutoring systems can thus be viewed as instructional products that embed models of cognition, learning, and instruction, and gather data concerning those models.[3] Furthermore, developing such systems drives our theories about instructional design products to much greater sophistication and detail because of the rigor required in producing a system that runs and attains desired outcomes. What about theories of the instructional design process? We suggest that intelligent tutoring systems research can drive theory in this area in much the same way. The example we have in mind is the IDE system discussed elsewhere in this book (Russell & Burton, this volume). IDE is a computer-based environment for the design, development, and delivery of computer-based instruction. As a design tool, the system allows an instructional designer to make decisions at a number of levels that can be mapped onto the levels we have proposed in our instructional design problem space. IDE also provides designers with tools that allow them to rationalize the particular decisions they have made. These rationalizations are arguments for why a decision has to be "just so" based on other decisions and instructional principles that may be grounded in the research literature. In essence, these rationalizations fit with the competitive argumentation scheme discussed by Van Lehn et al. (1984).

The IDE system is thus an attempt to reify the levels in the instructional design problem space and, with further research, lay out the alternatives within that space. IDE, or analogous tools, can

[3] This may be more of a prescription rather than an accurate description of the current state-of-the-art.

be used in similar ways to intelligent tutoring systems to collect data about the instructional design process--an area that we pointed out earlier has received no substantive attention. Further, systems such as IDE that encourage designers to provide theoretical rationale for their design decisions could prove to be important in a methodology of competitive argumentation among competing theories of learning, instruction, and instructional design.

References

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

Anderson, J.R., Boyle, C.F., Farrell, R. and Reiser, B.J. (1984). Cognitive principles in the design of computer tutors. In the *Proceedings of the Sixth Annual Conference of the cognitive Science Society* (pp. 2-9). Boulder, CO: University of Colorado, Institute of Cognitive Science.

Anderson, J. R., Boyle, C.F., and Yost, G. (1985). The geometry tutor. In A. Joshi (Ed.) *Proceedings of theNinth International Joint Conference on Artificial Intelligence* (pp. 1-7). Los Altos, CA: Morgan Kaufmann.

Belenky, M. F., Clinchy, B. M., Goldberger, N. R., & Tarule, J. M. (1986). *Women's ways of knowing: The development of self, voice, and mind.* New York: Basic Books.

Bork, A. (1981). *Learning with computers..* Bedford MA: Digital Press.

Brown, J. S. (1983). Process versus product: A perspective on tools for communal and informal electronic learning. In *Report from the learning lab: Education in the electronic age.* Educational Broadcasting Corporation.

Brown, J. S, & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science, 2,* 155-192.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 79-98). New York: Academic Press.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale NJ: Lawrence Erlbaum Associates.

Carey, S. (1985) *Conceptual change in childhood.* Cambridge MA: Bradford/MIT Press.

Clark, H. H., & Wilkes-Gibbs, D. (1986). Referring as a collaborative process. *Cognition, 22,* 1-40

Cohen, P. R., & Feigenbaum, E. A. (1982). *The handbook of artificial intelligence* (vol. 3). Los Altos CA: William Kaufmann.

deKleer, J., & Brown, J. S. (1984) A qualitative physics based on confluences. *Artificial Intelligence, 24,* 7-84.

diSessa, A. A. (1982). Unlearning Aristoteian physics: A study of knowledge-based learning. *Cognitive Science, 6,* 37-75.

diSessa, A. A. (1985, June). Knowledge in pieces. Address to the Fifteenth Annual Symposium of th Jean Piaget Society, Philadelphia, PA.

Dugdal, S. (1982). Green globs: A microcomputer application for graphing of equations. *Mathematics Teacher, 75,* 208-214.

Fawcett, H.P. (1938). *The nature of proof.* (The thirteenth yearbook of the National Council of Teachers of Mathematics). New york: Teachers College, Columbia University.

Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence, 24,* 85-168.

Gagne, R. M., & Briggs, L. J. (1979). *Principles of instructional design* (2nd ed.). New York: Holt, Rinehart & Winston.

Greeno, J. G., Brown, J.S., Foss, C., Shalin, V., Bee, N.V., Lewis, M.W., and Vitolo, T.M. (1986). *Principles of problem solving and computer-assisted instruction.* Berkeley CA: University of California, School of Education.

Gropper, G. L. (1983). A behavioral approach to instructional prescription. In C. M. Reigeluth (Ed.), *Instructionaldesign theories and models: An overview of their current status* (pp. 101-162). Hillsdale NJ: 1983.

Heines, J. M. (1984). *Screen design strategies for computer-assisted instruction.* Bedford MA: Digital Press.

Hutchins, E. L., Hollan, J. D., and Norman, D.A. (1986). Direct manipulation interfaces. *Human-computer Interaction, 1,* 311-338.

Johnson, W.L. and Soloway, E. (1983). *PROUST: Knowledge-based program understanding* (Tech. Rep. No. 285). New Haven, CT: Yale University Computer Science Department.

Kitcher, P. (1984). *The nature of mathematical knowledge* New York: Oxford University Press

Kuipers, B., & Kassirer, J. P. (1984). Causal reasoning in medicine: Analysis of a protocol. *Cognitive Science, 8,* 305-336.

Lampert, M. (in press). On knowing, doing, and learning multiplication. *Cognition and Instruction.*

Landa, L. N. (1983). The algo-heuristic theory of instruction. In C. M. Reigeluth (Ed.), *Instructionaldesign theories and models: An overview of their current status* (pp. 163-212). Hillsdale NJ: 1983.

Lave, J., Murtaugh, M., & de la Rosa, O. (1984). The dialectic of arithmetic in grocery shopping. In B. Rogoff & J. Lave (Eds.), *Everyday cognition: Its development in social context.* Cambridge MA: Harvard University Press.

Leinhardt, G., Weidman, C. and Hammond, K.M. (1984). *Introduction and integration of classroom routines by expert teachers.* Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Leinhardt, G., & Greeno, J. G. (1986). The cognitive skill of teaching. *Journal of Educational Psychology, 78,* 75-95.

Lewis, M.W. and Anderson, J. R. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology, 17,* 26-65.

Mager, R. (1962). *Preparing instructional objectives.* Palo Alto CA: Fearon.

Merrill, M. D. (1983). Component display theory. In C. M. Reigeluth (Ed.), *Instructionaldesign theories and models: An overview of their current status* (pp. 279-334). Hillsdale NJ: 1983.

Merrill, M. D. (in press). Component display theory: Instructional design for courseware authoring. *Insructional Science.*

Newell, A., & Card (1985). The prospects for psychological science in human-computer interaction. *Human-computer Interaction, 1*, 209-242.

Newell, A. & Simon, H. A. (1972). *Human problem solving.* Englewood Cliffs NJ: Prentice-Hall.

Norman, D. A and Draper, S.W. (1986). *User centered system design.* Hillsdale, NJ: Lawrence Erlbaum.

Palincsar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction, 1,* 117-176.

Pirolli, P. L. (in press). A cognitive model and computer tutor for programming recursion. *Human-computer Interaction.*

Posner, G. J., & Rudnitsky, A. N. (1986). *Course design: A guide to curriculum development for teachers.* New York: Longman.

Reigeluth, C. M. (Ed.) (1983a). *Instructional-design theories and models: An olverview of their current status.* Hillsdale NJ: Lawrence Erlbaum Associates.

Reigeluth, C. M. (1983b). Instructional design: What is it and why is it? In C. M. Reigeluth (Ed.), *Instructionaldesign theories and models: An overview of their current status* (pp. 3-36). Hillsdale NJ: 1983.

Reigeluth, C. M., & Stein, F. S. (1983). The elaboration theory of instruction. In C. M. Reigeluth (Ed.), *Instructionaldesign theories and models: An overview of their current status* (pp. 335-382). Hillsdale NJ: 1983.

Reiser, B.J., Anderson, J.R., and Farrell, R.G. (1985). Dynamic student modelling in an intelligent tutor for LISP programming. In A. Joshi (Ed.) *Proceedings of theNinth International Joint Conference on Artificial Intelligence* (pp. 8-14). Los Altos, CA: Morgan Kaufmann.

Scandura, J. M. (1983). Instructional strategies based on structural learning theory. In C. M. Reigeluth (Ed.), *Instructionaldesign theories and models: An overview of their current status* (pp. 213-246). Hillsdale NJ: 1983.

Schegloff, E. A. (1972). Notes on a conversational practice: foirmulating Place. In D. Sudnow (Ed.), *Studies in social interaction* (pp. 75-119). New York: Free Press.

Schoenfeld, A. H. (1985). *Mathematical problem solving.* New York: Academic Press.

Simon, H. A. (1969). *The sciences of the artificial.* Cambridge MA: MIT Press.

Suppes, P. (1981). *University-level computer-assisted instruction at Stanford: 1968-1980.* Stanford CA: Stanford University, Institute for Mathematical Studies in the Social Sciences.

VanLehn, K., Brown, J. S., & Greeno, J. G. (1984). Competitive argumentation in computational theories of cognition. In W. Kintsch, J. R. Miller, & P. G. Polson (Eds.), *Method and tactics in cognitive science* (pp. 235-262). Hillsdale NJ: Lawrence Erlbaum Associates.

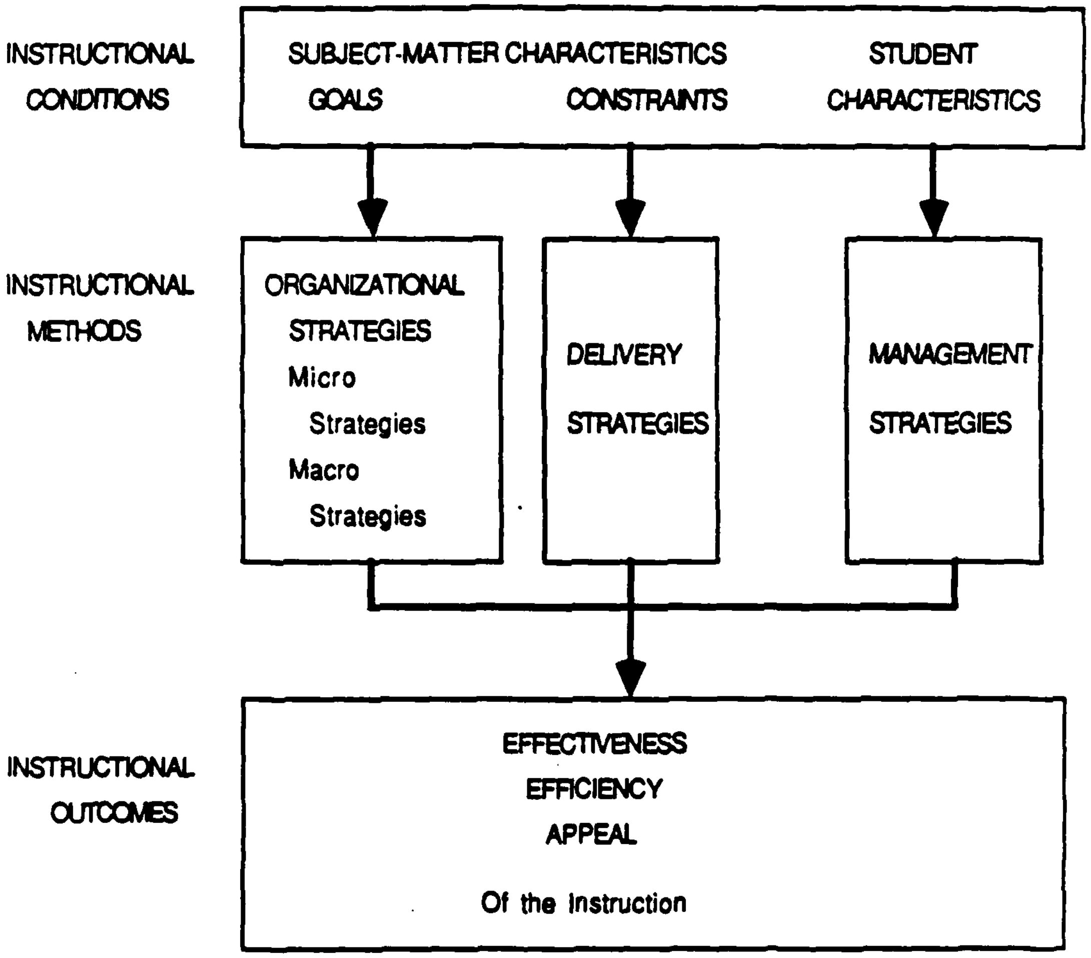
Uarushalmy, M., and Houde, R.A. (1986). The geometric supposer: Promoting thinking and learning. *Mathematics Teacher, 79*, 418-422.

Figure Captions

Figure 1. Components of the instructional design problem space.

Figure 2. Reigeluth's instructional design framework. From "Instructional Design: What is it and Why is it?" by C.M. Reigeluth, 1983, p.19. In C.M. Reigeluth (Ed.) *Instructional-design Theories and Models: An Overview of Their Current Status*, Hillsdale, NJ: Lawrence Erlbaum. Copyright 1983, by Lawrence Erlbaum. Adapted by permission.

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


INSTRUCTIONAL CONDITIONS
SUBJECT-MATTER CHARACTERISTICS
GOALS
CONSTRAINTS
STUDENT CHARACTERISTICS
INSTRUCTIONAL METHODS
ORGANIZATIONAL STRATEGIES
Micro Strategies
Macro Strategies
DELIVERY STRATEGIES
MANAGEMENT STRATEGIES
INSTRUCTIONAL OUTCOMES
EFFECTIVENESS
EFFICIENCY
APPEAL
Of the Instruction

The Instructional Design Environment

Daniel M. Russell
Thomas P. Moran
Daniel S. Jordan

Intelligent Systems Laboratory
Xerox Palo Alto Research Center

June, 1987

Abstract:

The Instructional Design Environment (IDE) is a prototype interactive design and development system which assists instructional designers with the process of creating complex instruction. IDE provides a representation for the substance of a course, and a language that expresses the rationale for the course design. These goals are achieved by representing the course design as sets of organization and implementation decisions that satisfy course objectives. IDE provides an infrastructure for instructional design and tools to support work in that structure. IDE implements a way of articulating the design and development process by allowing the creation of a *rationale* structure that explains why decisions were made, grounding the decisions in instructional principles. By providing a framework for design, yet allowing free representation within that framework, designers can work with the structure of a course in instructional terms.

In this paper, we describe the problems surrounding instruction design and development, describe IDE, and demonstrate how IDE helps to solve those problems.

This work was partially supported by a grant from the Army Research Institute MDA 903-83-C-0189

Acknowledgements: Jim Greeno and John Seely Brown developed the original statement of what IDE could be. Peter Pirolli helped in developing significant portions of both the code and the ideas. Anne-Marie Jensen, Julian Orr and Richard Burton participated in many of the discussions that helped us develop these ideas. Joe Psotka made many helpful comments on paper and the ideas. To each, we give our thanks.

1. The Problems of Creating Instruction

In a typical instruction creation setting, there are many interacting and competing goals to satisfy. Creating instruction is difficult: there is often a great deal of material to be covered, developers are under time and cost pressures, and often lack expertise in the domain to be taught. In a typical technical training creation task, a small team of instructional designers (2 -- 7 members) works together to create a workbook-based course to train technicians in the diagnosis and repair of a new machine. Often, only one of the team members will be expert in the field, and, almost certainly, none will have any specialized expertise in the course subject matter. Since creating training material is usually the final step of production, there are many pressures to produce the material rapidly. [Note 1]

IDE is an interactive design and development system to help instructional designers deal with the complexity of creating instruction materials, and to provide a system for the design and development of complex instruction. IDE can aid in creating the course design, structuring the course contents, and creating instructional sequences for standard or adaptive delivery.

Creating instruction that is firmly and coherently grounded in instructional principles is an important goal which is rarely achieved. Too often, a gap exists between understanding a set of instructional principles, and consistently employing those principles. This problem is magnified when creating instruction for complex or large domains, or designing instruction for complex delivery systems (e.g., VideoDisk or ICAI).

We see an opportunity to improve the quality of instruction. Because of the sheer volume of material, courses are often overburdened with redundant information, poor organization, or ineffective presentation of topics and concepts. These problems persist due to the lack of adequate tools for the designer and developers to manage and manipulate a large body of knowledge. Further, the time and resources required to develop instruction are costly. Computer-assisted development tools can decrease these requirements and simultaneously improve the quality of the instruction.

2. IDE

In light of the problems faced by instructional designers, there is enormous potential to improve instruction by providing a computational resource to help (a) manage the process of instruction creation, (b) assist in articulating and structuring the domain knowledge, and (c) assist in constructing the course.

IDE is a software environment which makes explicit the process of instruction design and development. It requires articulation of both the content of the instruction and the ***rationale*** of an instructional design. An IDE instructional design is a set of implementation and refinement decisions describing how a course is organized and based on an initial statement of instructional goals. A rationale is an explanation structure describing why a decision was made. IDE guides the design of instruction by providing the means to record and structure a set of decisions from course goals and specifications. Thus, the rationale is a data structure that records the reasons underlying a particular design. An IDE rationale records decisions about course structure, course content, instructional approach, and anticipated student behaviors.

Creating the rationale for a course design is important for several reasons:

* Changes in either instructional approach or course content can be made easily and accurately, since the rationale represents both the theory of instruction underlying the course design, and the course content.
* It is a useful tool for communication between colleagues or collaborators.
* It communicates to the instructor how and why a course is organized.
* Once created, it simplifies changes to the course material or course structure.

Building a complete rationale for a course is a formidable task. However, we believe the process of building the rationale with a knowledge management tool (IDE) can improve course design by reminding designers of options and opportunities available, as well as providing helpful analyses of partially completed designs. Requiring a rationale forces the designer to articulate the principles underlying the course construction. IDE requires the designer to think more carefully about why and how the course design accomplishes its original goals. Further, once created, a rationale structure makes explicit the design of a course, allowing collaboration between analysts, and makes feasible repair and reuse of the course.

3. IDE Description

IDE is built on Notecards [Halasz, et al., 1986], an idea-structuring hypertext system written in Xerox Lisp. IDE runs on any of the Xerox Lisp Machines (1108, 1109, 1132, 1185, 1186).

We view IDE as an entire course design / development / alteration system. During course development, IDE provides the designer with management support for building the detail structure of the course. At the same time, IDE helps keep the materials consistent with instructional principles. After the course has been created, a rationale simplifies rework or alteration of the course (due to content or structure changes, student population change, or shift in instructional principles).

IDE is especially advantageous when dealing with large or highly complex courses. In particular, it is useful when constructing material for adaptive course delivery, where the behavior of the student influences presentation of the material. (For example, [Russell, 1987])

3.1 Using IDE to Structure Knowledge

IDE is a knowledge structuring system, where the knowledge is course content, structure, and instructional method. At an abstract level, IDE accepts knowledge describing course goals as input, and assists the designer in creating a course as output. Figure 3.1 illustrates the flow of knowledge through IDE and represents a schematic map of the IDE user's display. In this figure, work proceeds roughly from left to right, moving from course objectives toward course decisions. Information about the course design is entered in the upper half of the column labelled "Requirements & Principles." This region contains course objectives, externally-imposed constraints (class size, delivery technology, student capabilities, etc.). In using IDE, a designer creates a course design, guided by these input requirements, working within the knowledge areas IDE defines. After refining the design, the designer makes decisions about the substance of the course in the areas of the rightmost column. Included in this output column are the "Knowledge Structure" (KS -- a representation of domain knowledge), the "Model of Student" (MOS -- a representation of the student), course control (strategy, pedagogy and tactics), and "Instructional Units" (IUs -- which comprise the course material). The Instruction column represents everything needed to teach the course specified by the Course Description. Note that both domain knowledge *and* instructional knowledge are included.

The atomic representation element in IDE is a Notecard with text describing what is represented by that card. There are a variety of card types (Principle, Decision, KnowledgeElement, etc.). Usually, a card represents a decision about course content or structure. To design a course, a user creates a linked network of Decision cards in the Decisions column. A Decision card is "rationalized" by creating linkages between the Decision card and other cards that act as warrants or reasons supporting that decision. A ***rationale link*** records the reasoning underlying the decision. The collection of Decisions forms a basis for the creation of Course Specification knowledge, which is represented in the right hand column. (Currently, IDE provides the superstructure for the designer to "fill in" the columns with decisions and structure of his devising, and little in the way of automatic structure checking.)

As Figure 3.1 shows, creating a course consists of evolving the representation of the course objectives (left column) into decisions about the course design (middle column) and generating a detailed representation of instruction (right column). This partitioning of areas separates differing sets of concerns within a course design.

Between each of the columns in Figure 3.1 are shaded zones indicating places where a rationalization is constructed to explain the relationships between elements in each of the areas. Figure 3.2 illustrates a rationale (R) link -- the basic element of a rationale structure -- between a principle (in the Principles area) and a decision (in the Instructional Decisions area). For example, cards in the "Principles" area represent instructional principles used by the designer. A given principle is"rationalized" by a rationale structure showing how entries in the "Literature" argue for its validity.

3.2 IDE Areas on the Screen

Figure 3.1.1 shows IDE in use by an instructional analyst. IDE partitions screen space into a set of ***columns*** divided into ***areas*** as in the schematic diagram of Figure 3.1. Each area contains a decision structure defined by the card types it contains, and how the cards link to cards in other areas. The areas are:

Literature -- contains references to primary sources for instructional principles. (Figure 3.3) Instructional and cognitive principles are rationalized in terms of the backing literature.

External Constraints / Course Objectives -- represent course-specific information such as course objectives, student abilities, available delivery media, amount of time available, etc. This "input" area changes with each new course developed. (Figure 3.4)

Principles -- are references to theories of learning and instruction. They encode what an instructional designer must remember while creating a course design. Principles (usually) do not change from course to course, although different sets become relevant according to the style of course being created. They are rationalized by elements in the Literature area. They, in turn, are used to rationalize decisions in the Decisions column. (Figure 3.5)

Epistemology -- contains decisions about the domain knowledge to be taught. Epistemological decisions are usually created by the Subject Matter Expert, and are rationalized in terms of epistemological principles. (Figure 3.6)

Cognitive -- decisions which record how the student will learn the skill or knowledge to be presented. These decisions represent how to accomplish the course objectives within the posted constraints, guided by the epistemology decisions. Because they are so closely linked to the domain knowledge, cognitive decisions may be rationalized in terms of cognitive principles (from the principles area) or by decisions in the epistemology. (Figure 3.7)

Presentation -- decisions which determine how the course is to be taught. They implement cognitive level decisions and are rationalized in terms of instructional knowledge. Cards in this region form a course specification in terms of presentations and tests. (Figure 3.8)

KS (Knowledge Structure) -- a concept structure that represents the course content (i.e., the domain knowledge). It is a representation of the elements of domain knowledge to be taught (concepts and skilss) and the relationships that exist between them. The final course design will teach some subset of this KS. (Figure 3.9)

MOS -- is the "Model of Student" (or student model). The MOS records information about the student: concept understanding, typical bugs, common learning styles, instructional conditions, etc. The MOS is a representation of the set of concepts and skills the student must acquire. At the same time, the MOS records information about the style in which the student learns. (Figure 3.10)

Course Control -- records the knowledge needed to guide the instruction of the course content. This knowledge is represented as rules and constraints describing how instruction proceeds. "Pedagogy" cards encode topic sequencing and topic inclusion information; "Strategy" cards encode the general instructional strategy; "Tactics" cards specify low-level information about how to implement instruction style and form. (Figure 3.11)

IUs (Instructional Units) -- are the units with which the student will interact. If the course delivery system is a VideoDisk system, then the IU will be a single VideoDisk segment (e.g., a frame in a frame-based interactive videodisk course). IUs may also be quite complex, for example, an IU may be a simulation system, or an entire learning environment. From the designer's viewpoint, IUs communicate concepts (individually or in groups) from the KS. (Figure 3.12)

4. Using IDE

4.1 IDE for Making and Recording Decisions

With this framework, IDE provides a mechanism for the designer to record and rationalize course content decisions, as well as structure and rationalize instructional knowledge. The IDE view of course creation consists of deciding:

* course content (what gets included, what gets left out),

* content structure (how the content is organized),

* how the student will learn,

- how the content is presented (instruction strategy),
- what the student must know (before, during and after instruction).

These decisions form the substance of the IDE course design. The IDE user has the task of making decisions (i.e., creating the decision / rationale structure) to implement the course objectives, creating an instructional strategy, and creating the Instructional Units (IUs) with which the student interacts during instruction.

Once developed, the decision structure can be used by the designer to understand how and why a choice was made in a course design. It is possible to work from entries in the KS or IUs back to the decisions and principles that give backing to those entries. Conversely, one can trace forward from a decision or principle to discover what the entailment of that choice is in the design.

4.2 IDE Analysis Tools

IDE is equipped with a pair of analysis tools: ***tracers*** for explaining why a decision was made and ***checkers*** for matching and mapping from one area into another.

The ***rationale tracer*** tool displays the rationale associated with a particular decision. IDE accomplishes this by following links back from a single decision to each of its parent. *In Figure 4.1, the graph* displayed is a rationale trace of a decision made in the Epistemology area. Epistemology decisions are rationalized in terms of elements from the Requirement and Principles column -- i.e., principles and course specifications. Here, the decision to organize the domain knowledge along a "Functional Decomposition," is based on the three principles "Advanced Organizer," "Hierarchy is Reconstructable," and "Functionality is Useful." When the rationale is displayed, each of the parents for the decisions (i.e., the three nodes on the left side of the rationale trace) are displayed immediately below the rationale window (see the title of the window just below the rational browser) and the rationale argument (the contents of the "R" node) is shown.

A ***checker*** maps elements from one area onto elements in another area following specific relationships between the areas. Thus, creating a checker that maps from Course Defs area into the KS will show how many of the Course Defs are implemented in the KS. (This gives the designer a simple way to determine how many Course Definitions have been left unsatisfied.) Figure 4.2 shows a checker that maps from the Instructional area into the IUs area. This display shows which of the Instructional objectives have been implemented by IUs. This checker yields information on topics that have been overlooked or omitted, and detects redundancies.

4.3 Using IDE: An Example

We have designed a course on Basic Xerography using IDE. Xerography is a representative domain because it combines many sub-domains (electrical, mechanical, chemical) into a single course. The domain is complex enough to be challenging for course design, yet still tractable as a test case. We built several different versions of the course (a lecture & workbook version, a frame-oriented computer delivered version, and an Interactive LaserDisk version) on this single design. In each version, the Epistemology and Cognitive area contents remained constant, while the linkages to the implementation varied from course to course.

Figure 4.3 shows a screen image of IDE in use during the design of the Basic Xerography course. Here, the Goals and Constraints are on the top of the screen (above the bar), the Principles are on the left side, and Decisions are grouped in the far right hand column. (The Instruction column contents do not appear in this image.) Each card in this image (plus many not in this display) was created by the user and linked into the growing course design. The design process is primarily top-down, guided by the designer's experience and knowledge of the domain. Generally, the process is to create the domain Epistemology, then perform the Cognitive analysis, and, finally, create the Instructional design. As in any design process, constraints and structure discovered during development can significantly impact the higher design levels. Thus, a designer may work in any column or area at any time, moving between design levels and areas as needed. IDE's ability to display rationale structure (the tree structure shown in Figure 4.3) makes modification of the complex design structure straightforward.

At this point in the design (Figure 4.3), the user is attempting to satisfy the objective shown in the upper right hand corner -- "Prepare for Troubleshooting." This objective was set by the designer as part of the course specification. To satisfy this goal, the designer must make and record in IDE the decisions about how to achieve it. As stated, the goal is a declaration about a cognitive objective. The designer chooses to implement this objective by creating a decision stating that the student must be able to generate Fault Predictions. (The decision and the <Implements> link to the course objective are in the "Fault Predictions" card located mid-screen, on the right side.)

Normally, after the user creates a decision, the rationale in support of that decision is built. Figure 4.4 shows the rationale the designer constructed to support the "Fault Predictions" decision. The rationale card shown in the figure explains why the designer made this decision. In this instance, the explanation is straightforward; the cognitive decision to teach "Fault Predictions" is rationalized by arguing that in order to do the predictions, the student must be able to perform forward reasoning, based on his understanding of causality and diagnostic symptoms. In more obscure cases, the rationale card may contain more elaborate arguments discussing trade-offs and counter-proposals. Note that the rationale states its argument in terms of Epistemology, Learning and Cognitive Task Analysis principles. [Note 3]

When building the rationale, the designer realizes that this decision impacts the way the KS is built -- that is, the Epistemology of the KS must be changed to support the Cognitive Objective decision. The designer then creates a new decision, "Cause-effect relations" (see Figure 4.5) recording the idea that the relationships between causes and effects are essential to doing "Fault Predictions."

This new decision must be rationalized as well. (Figure 4.6) This time, the rationale supports the original course objective, the new decision to teach Fault Predictions and principle of supporting inference by teaching casual understanding.

With these two new decisions, the designer shifts attention to the Instruction column. (See Figure 4.7.) The designer has decided to make the instruction of cause-effect relations a distinct portion of the course. This Cognitive area decision has its impact in the Course Control area. The decision to teach cause-effect requires altering the rules describing how the course will be taught. In particular, the (Teach BX) rule in the "Teach BX" window of Figure 4.7 is updated to include this clarified view of the course material.

4.4 IDE as a Cognitive Aid to Course Design

IDE functions as a cognitive aid to the designer by providing a representation in which to work with course designs, and a set of reminders about potential problem solutions. The Principles area operates primarily as an on-line library of instructional principles and techniques. Typically, elements in the Principles area are simply given to the designer, as part of IDE. However, this area may be tailored to the working requirements of a designer (as may all other areas). The user can add, delete or rearrange elements as necessary.

In addition to these resources, pieces of previously successful course designs can be stored in the Principles section as an addition to the data base. These elements can be abstracted from designs created within IDE by the user. Hence, IDE can be used to create new course material or it can be used to analyze existing courses. The process of using IDE as a "reverse engineering" tool is similar, but instead of making new decisions based on Subject Matter Expert information and course objectives, decisions are inferred from the structure and content of course materials. The analysis results are then used to update the IDE Principles data base.

The implementation of IDE described here may be tailored to a variety of design / development methodologies. We envision the Principles and Literature areas as libraries in which the designer can explore, choosing and incorporating elements from these sources into the design under construction as desired. There are no restrictions on how the knowledge structures or course material are created and developed. However, IDE could be operated in more tightly, reducing the variety of options available to the designer. This enforces consistency of course design and delivery over a wide range of developers and courses. Tailoring a set of principles and their pattern of use to capture a style of instruction would be an effective means of standardizing course design across a set of IDE users.

4.5 Changing A Course Design

Course design modification is one of the most important uses of a design tool. IDE supports modifications during the initial creation of the design and material, or after the course has been developed and is in use. Computing the modifications necessary to a design is simplified in both cases by having the rationale. Tracers and checkers can compute the consequences of a design change by following rationale links from objectives or design decisions to their entailments.

Maintaining a consistent design -- especially over a large course, or during course redesign or repair -- is usually a difficult task. Maintaining course consistency with IDE is simpler because the course design is explicitly represented (and inspectable), and because course presentation design is distinguished from course content.

4.6 Building a Conceptual Knowledge Structure (KS) of Domain Knowledge

IDE requires construction of a Knowledge Structure (KS) representing the conceptual structure of the domain knowledge. We argue that requiring the instruction designers to articulate the knowledge to be taught (regardless of the final delivery structure or style) focuses attention on the problem of determining instructional content. But, once the designer captures the salient knowledge of a domain, that knowledge can be structured within the KS to reflect accurately the best way of teaching that knowledge. [Greeno & Pirolli, 1987] IDE also provides a space for articulating this instructional strategy (the Course Control area) without carrying along the burden of all the course content. Of course, often, the two cannot be cleanly divided, but will spill into both areas. The separation enforces an organization of material without confusion.

The Knowledge Structure is necessary for several reasons: (1) the Knowledge Structure (in conjunction with the Instructional Units) determines the course content exactly; (2) it reduces the number of tasks the designer must manage at any one time; "getting the knowledge right" is separated from the task of creating instruction delivery; (3) distinguishing instruction knowledge from domain knowledge allows the course to be described intensionally, and allows the possibility of automatic generation of course outlines; (4) a Knowledge Structure opens the possibility of creating an adaptive course delivery system driven by the invariant knowledge in the Knowledge Structure. (See [Russell, et al., 1987].)

The Knowledge Structure is constructed by the Subject Matter Expert with the course designer. The Knowledge Structure form is rationalized by decisions made within the Epistemology area.

Figure 3.9 shows a window onto the entire Knowledge Structure of the Basic Xerography course, while Figure 4.8 shows a portion of the Knowledge Structure in more detail. Each rectangle represents a concept or skill, and the links signify relationships between the concepts. In Figure 4.8, concepts are related (linkec') by a variety of relations. The most common relationships in this diagram are *subconcept, subcomponent,* and *next-step*. Concepts and concept relationships represent the content the student should learn from the course. To the Subject Matter Expert and the course designer, the Knowledge Structure represents domain knowledge at a tractable level of instruction. Concepts in the Knowledge Structure map onto IUs; while concept-concept relationships in the Knowledge Structure guides the instructor in sequencing the delivery of IUs.

The Knowledge Structure browser is the designer's primary tool for creating and manipulating the Knowledge Structure. The browser is used to create the nodes, as well as the differing links (and link types). The global view browser is too complex, so the designer works with sub-browsers such as those seen in Figure 4.8. A sub-browser is created by the user by extracting a subgraph from the KS following only a subset of all the possible links types. Thus, the content of the Knowledge Structure maps directly onto the grain size of instructional content. Individual concepts are placed into the Knowledge Structure if they are dealt with individually in the course. This effectively defines the grain size the Knowledge Structure must represent.

Note that the structure of the Knowledge Structure as seen in these figures is simply a representation of the relational structure of the concepts of the domain, rather than a representation of the concepts themselves. The Knowledge Structure, as such, is a relatively large grain representation of knowledge. Each of the nodes in the Knowledge Structure represents a separable concept that must be taught. For the purpose of course design, this granularity is the desired resolution.

4.7 Representing Instruction and Presentation Methods

IDE supports articulation of the instructional strategy used to teach the course, in addition to representing course content . The skills required to teach a course are complex and occur at many levels of abstraction. IDE represents this knowledge as ***rules*** that create instructional goals, and as ***constraints*** on instructional performance. (Rules create instructional goals, while constraints restrict the way those goals are implemented.) This knowledge is broken into three categories:

Pedagogy rules and constraints are course-specific, representing the sequencing of topics in the course.

Strategy rules represent general approaches to the presentation of information to the student. They encode instruction methods such as "drill and practice," "present three concepts, then test for understanding," or "monitor a student's solution of a problem, and intervene as soon as the student varies from the solution path." Strategy rules do more than provide recipes of instruction approaches; they are used to generate the major goals of the instructional plan.

Tactical rules and constraints govern how instructional goals are implemented, taking into account low-level concerns such as display format, potential IU characteristics, etc. Tactical rules spell out how to implement an instructional goal in a particular instructional environment for a particular type of student.

The rules for the Basic Xerography course are shown in Figure 4.9.

Articulating the Course Control rules forces the designer to consider the subtleties of how course material will be taught. By distinguishing the way in which knowledge is taught from the representation of that knowledge, attention must be focused on the relationship of the teaching style to the mechanisms of learning. (This contrasts with standard instructional design practice, which confounds the issues. Instructional material is written with teaching style issues in mind, but teaching style and instructional practice are not separated.) Course Control rules capture the *designer's understanding of the most effective instructional approach* to the material, making that understanding explicitly available.

4.8 Representing Instructional Units (IUs)

IUs are created by the designer in response to decisions made in the Instruction area, and are the final output of the IDE design process. An IU is concrete piece of instruction that teaches a particular concept. It represents what the student will see and use. An IU may be a presentation of text on a concept, a segment of video, an exercise, a lecture segment, or a series of questions. In IDE, an IU is represented by a card that records (a) the presentation content (e.g., the text of a textual presentation), (b) what concepts the IU addresses, (c) an attribute / value list describing properties of the IU (e.g., presentation mode, level of detail, level of difficulty), (d) the mapping of questions and answers onto concepts and misconcepts. (Figure 3.12 illustrates an IU as specified in IDE.)

5. Putting It All Together

Creating a course design within IDE is essentially a process of creating design decisions and a rationale structure to completely specify each of the knowledge areas. Our design philosophy is very unrestrictive: the designer can work within any area at any time. (Other instruction design systems are very prescriptive [Branson, et al.,1975]. [Note 2]) This characteristic reflects our experience with IDE. Designing and developing a course from the initial course objectives down to a specific sequence of Instructional Units is largely a top-down refinement process. In practice, however, discoveries made as work progresses may have severe consequences on the overall design, often requiring many design iterations in a more structured design system. In IDE, the accessibility of each knowledge area permits changes to be made readily throughout the design.

After all IDE knowledge areas have been fully specified, the course must be transcribed into a delivery form. Each delivery mode (e.g., workbook & lecture, Interactive Videodisk, Computer-Based Training, etc.) has its own special requirements; but the process can be generally described as transferring the Course Specifications column into the chosen medium. For instance, creating a workbook-based training course from an IDE design consists of transferring (and editing) the contents of the Instructional Units into a textual representation following the sequence given by the Pedagogical rules (that is, converting the IU contents into a document). Creating an Interactive Videodisk course requires that some of the Strategy and Tactical rules be converted into the program that will drive the videodisk (in addition to transcribing the IU contents into video segments). In each case, the final transformation is fairly straightforward, and is driven by the set of IUs and the Course Control rules.

6. Summary

IDE is an online environment for dealing with the many decisions that must be made, recorded, structured, and accessed during the design and development of instruction. It is also a way to view instruction: content is distinguished from delivery, both content and delivery are explicitly represented, and everything in the final instructional product exists for a specific, rationalized reason. With this representation for instruction, we can begin to explore what kinds of intelligent assistance can be supplied to designers and developers.

Notes

[Note 1]

This is certainly the case within Xerox, as we have discovered after visiting most of the corporate training production centers. In discussion with colleagues from other corporations, this experience seems representative of other large producers of in-house training materials, as well.

[Note 2]

There are many instructional design models. They focus on managing designer resources, and giving designers a task profile to follow. Most of these models are very prescriptive, and none allows the generality of approach seen in IDE. See [Andrews, 1980] for a summary of 40 models of instructional design.

[Note3]

Notice that there are two parts to the rationale: the text which IDE stores in a card, and the links that organize a set of cards into an argument structure. IDE cannot process the contents of the cards, but is limited to working with the relationships that exist in the argument structure.

References

Andrews, D. H., Ludwika, G.

"A Competitive Analysis of Models of Instructional Design"

Journal of Instructional Development, 3(4), (1980)

Branson, R. K.

"Interservice Procedures for Instructional Systems Development. Phase I: Analyze"

(4 volumes plus executive summary)

Florida State University, Center for Educational Technology (1975)

(Also available from:

National Technical Information Service, Springfield, VA 22161)

Halasz, F. G., Moran, T. P., Trigg, R. H.

"Notecards in a Nutshell"

Intelligent Systems Lab Publication
Xerox PARC, Palo Alto, CA. (1986)

Greeno, J., Pirolli, P.
"Intelligent Tutoring Systems and Instructional-Design Science: Toward a Rapprochement," in
Intelligent Tutoring Systems: Lessons Learned
J. Psotka, L. D. Massey, and S. A. Mutter (Eds.).
Lawrence Erlbaum Associates, Inc., Hillsdale, N.J. (1987)

Russell, D. M.
"The Instructional Design Environment: The Interpreter"
Intelligent Tutoring Systems: Lessons Learned
J. Psotka, L. D. Massey, and S. A. Mutter (Eds.)
Lawrence Erlbaum Associates, Inc., Hillsdale, N.J. (1987)

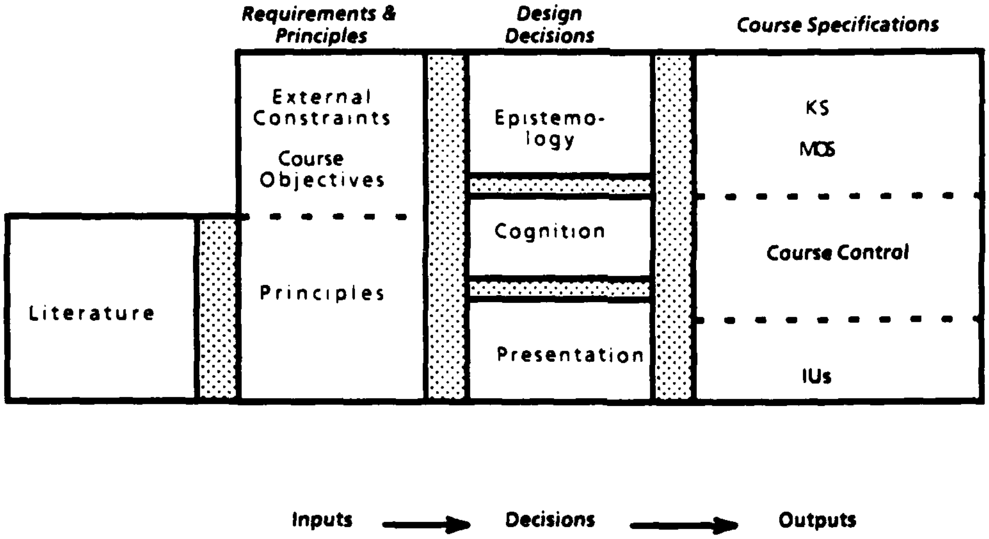


Figure 3.1 -- Knowledge areas of IDE

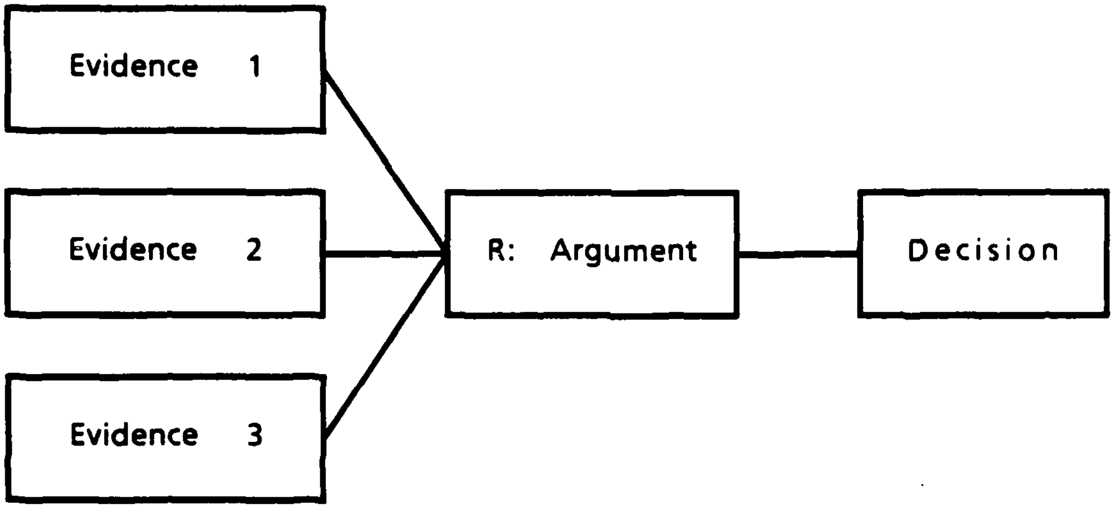
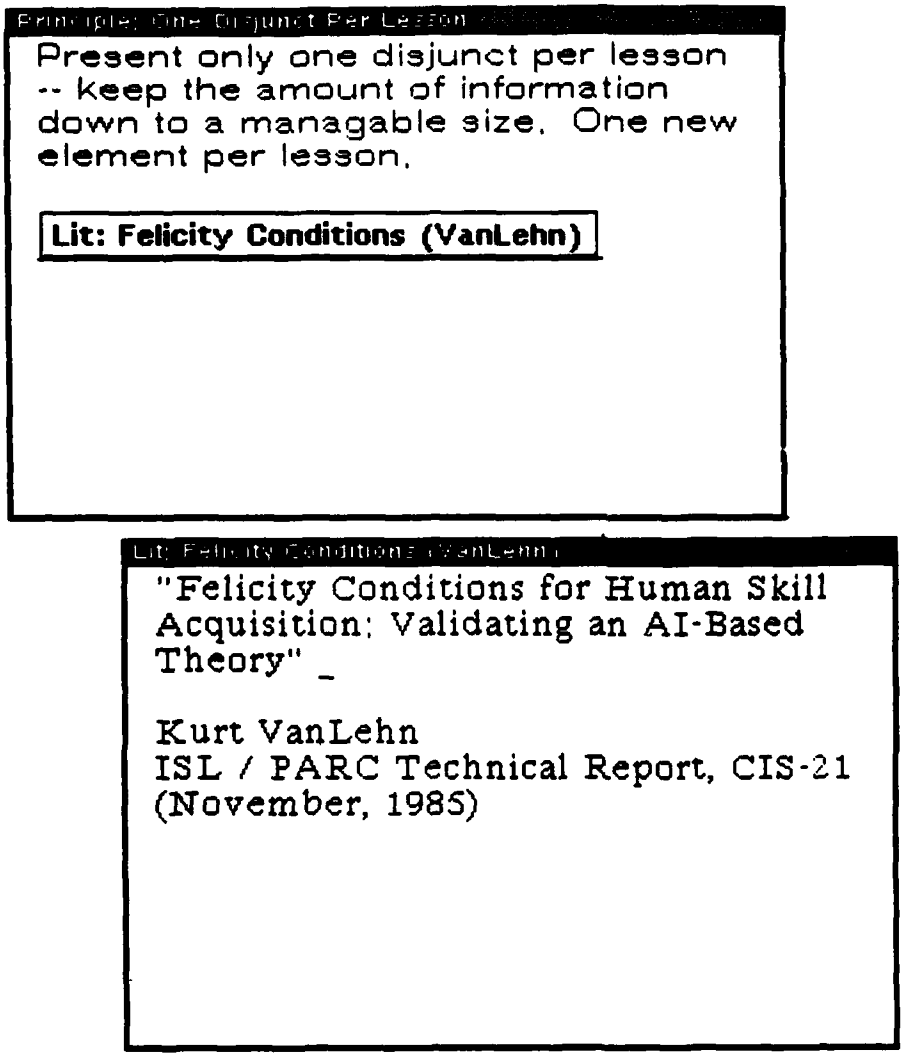


Figure 3.2 -- A rationale link "rationalizes" a decision in IDE by linking evidence that was the basis for making the decision to the decision record. The content of the rationale link is an argument specifying why the decision (on the right) follows from the evidence (on the left).

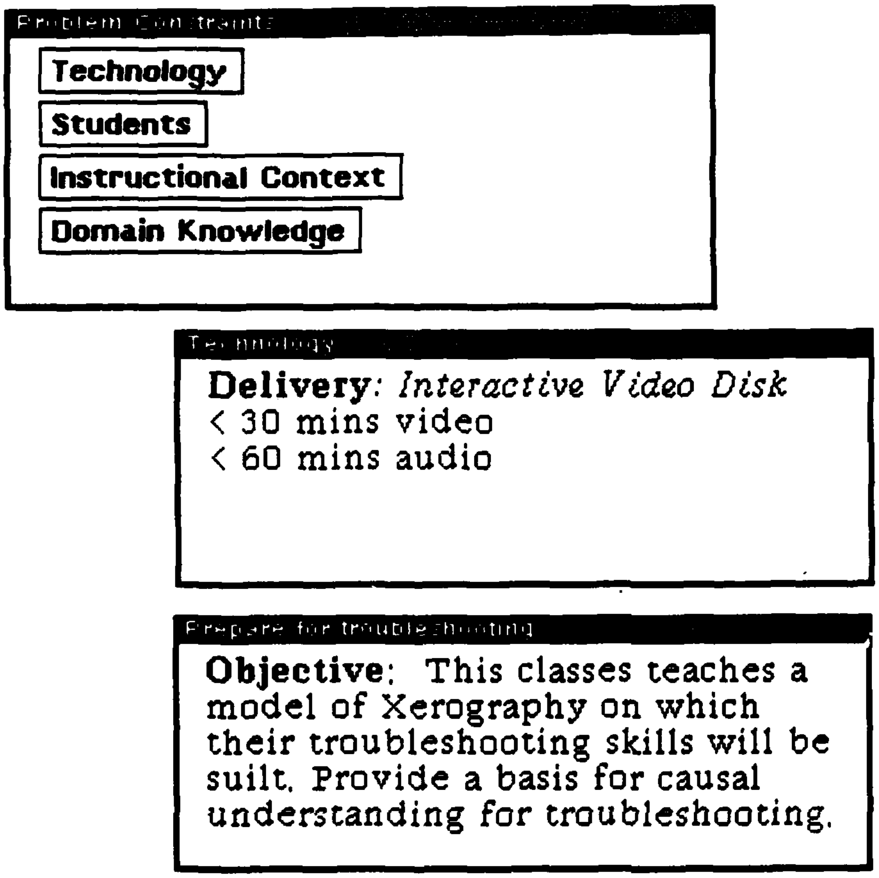

Principle: One Disjunct Per Lesson
Lit: Felicity Conditions (VanLehn)
Lit: Felicity Conditions (VanLehn)

Present only one disjunct per lesson -- keep the amount of information down to a managable size. One new element per lesson.

"Felicity Conditions for Human Skill Acquisition: Validating an AI-Based Theory" _

Kurt VanLehn
ISL / PARC Technical Report, CIS-21
(November, 1985)

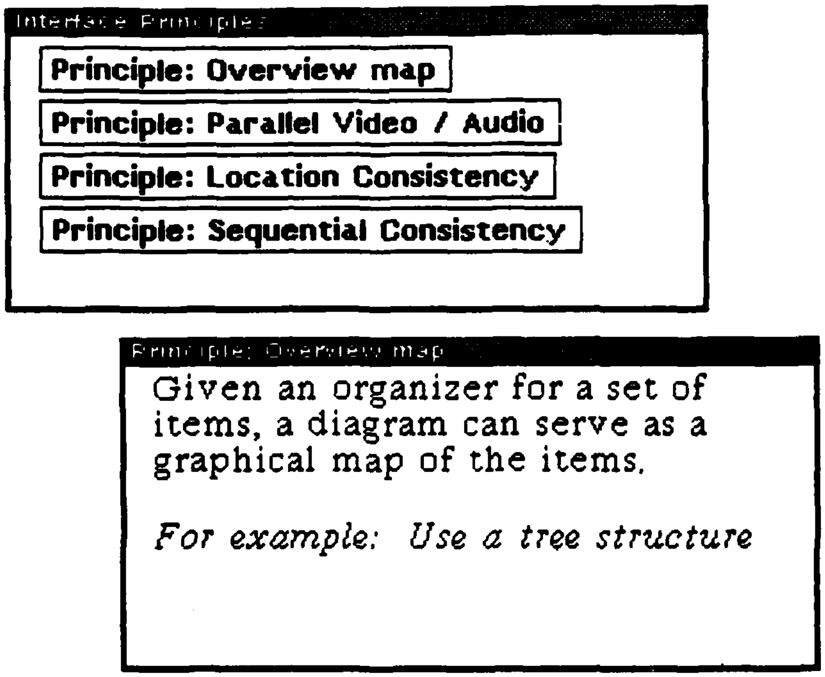
Figure 3.3 -- A Principle and a Literature Reference card. The rectangular icon in the Principle card is a typed pointer to the Literature card.



Delivery: *Interactive Video Disk*
< 30 mins video
< 60 mins audio

Objective: This classes teaches a model of Xerography on which their troubleshooting skills will be suilt. Provide a basis for causal understanding for troubleshooting.

Figure 3.4 -- An External Constraint / Course Defs card represents goals and constraints that imposed on the course design and development process.



Principle: Overview map

Principle: Parallel Video / Audio

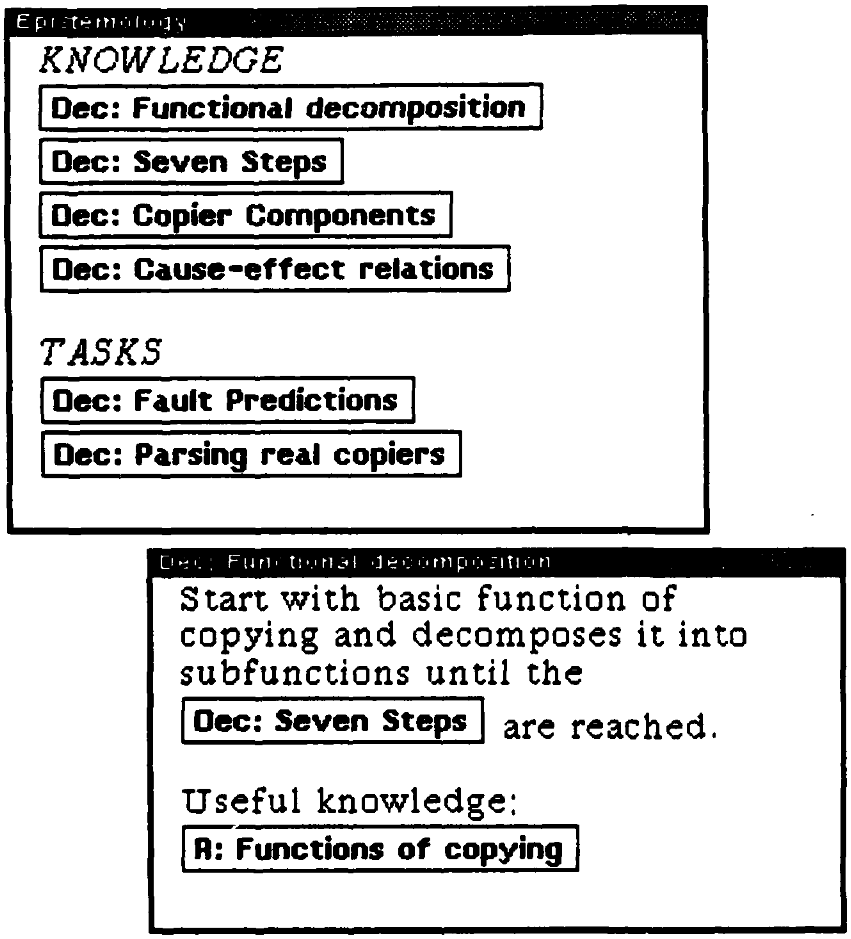
Principle: Location Consistency

Principle: Sequential Consistency

Given an organizer for a set of items, a diagram can serve as a graphical map of the items.

For example: Use a tree structure

Figure 3.5 -- A Principle card represents an instructional or learning principles that will be used to rationalize a decision in the Decisions column.



Epistemology

KNOWLEDGE

Dec: Functional decomposition

Dec: Seven Steps

Dec: Copier Components

Dec: Cause-effect relations

TASKS

Dec: Fault Predictions

Dec: Parsing real copiers

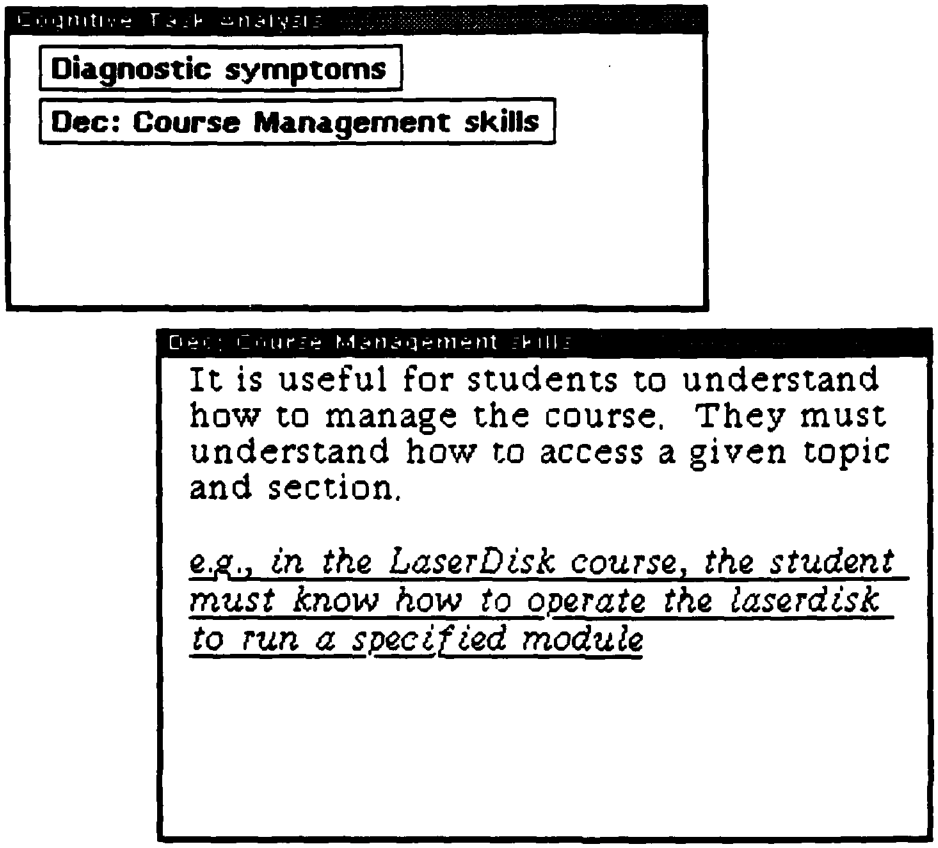
Dec: Functional decomposition

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

Useful knowledge:

R: Functions of copying

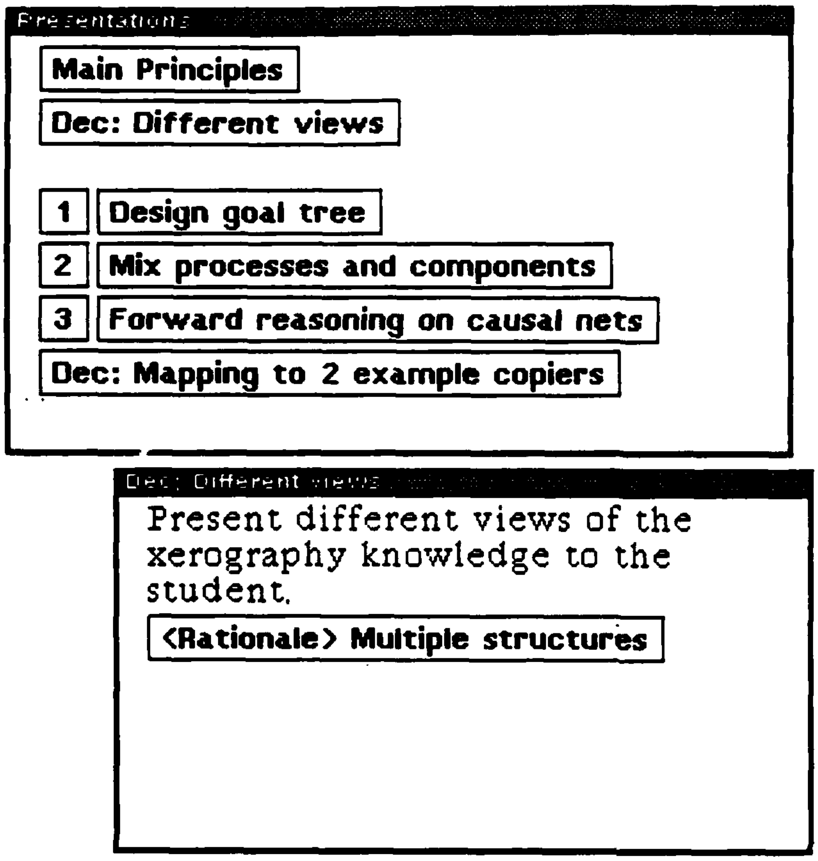
Figure 3.6 -- An Epistemological card contains a description of a chunk of domain knowledge. The "Epistemology" card records decisions about what domain knowledge must be taught, and how that knowledge is structured. The "Dec: Functional decomposition" card records a decision, illustrating how sub-decisions are used, and pointing to the rationale for this decision (the "R: Functions of copying" pointer).



It is useful for students to understand how to manage the course. They must understand how to access a given topic and section.

e.g., in the LaserDisk course, the student must know how to operate the laserdisk to run a specified module

Figure 3.7 -- Cards in the Cognitive area record decisions about the task analysis of the course. Such cards record decisions about what cognitive tasks must be accomplished by the course. They are rationalized by principles and epistemology.


Presentations
Main Principles
Dec: Different views
1
Design goal tree
2
Mix processes and components
3
Forward reasoning on causal nets
Dec: Mapping to 2 example copiers
Dec: Different views
<Rationale> Multiple structures

Present different views of the xerography knowledge to the student.

Figure 3.8 -- A decision in the Instructional area is specifies how the course will appear, how to present a concept, or gives details on how a presentation should appear. These cards are created by the designer, forming the substance of the design. They are rationalized by cognitive decisions, instructional principles and course constraints. This figure shows a set of Cognitive level decisions, and the expansion of the first decision made, "Different views."

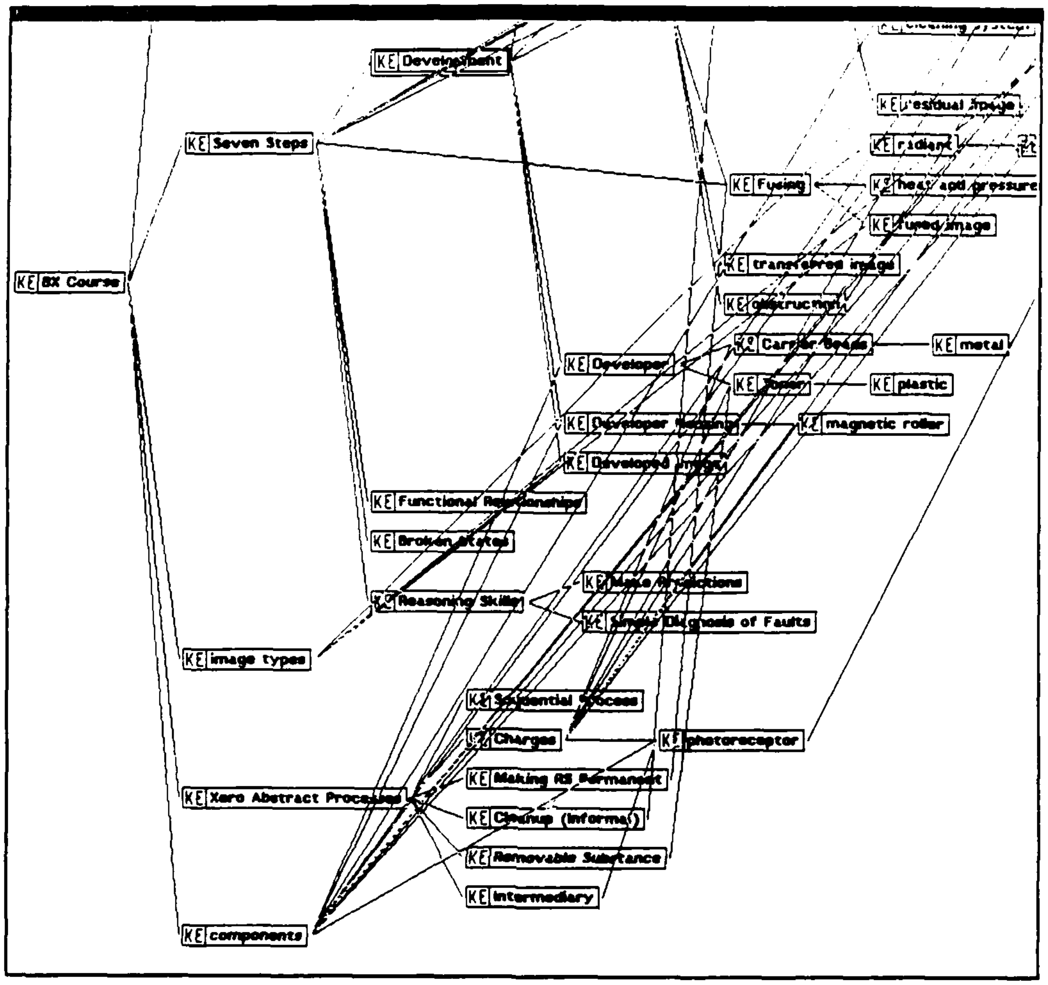
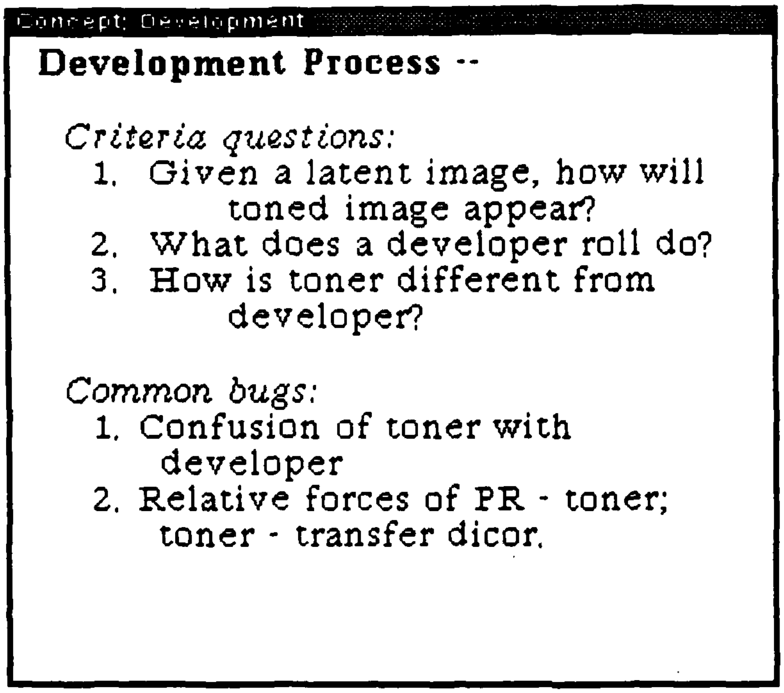


Figure 3.9 -- The Knowledge Structure (KS) is the conceptual structure of the course content. Nodes represent concepts or skills while links represent the relationships between them.



Concept: Development

Development Process --

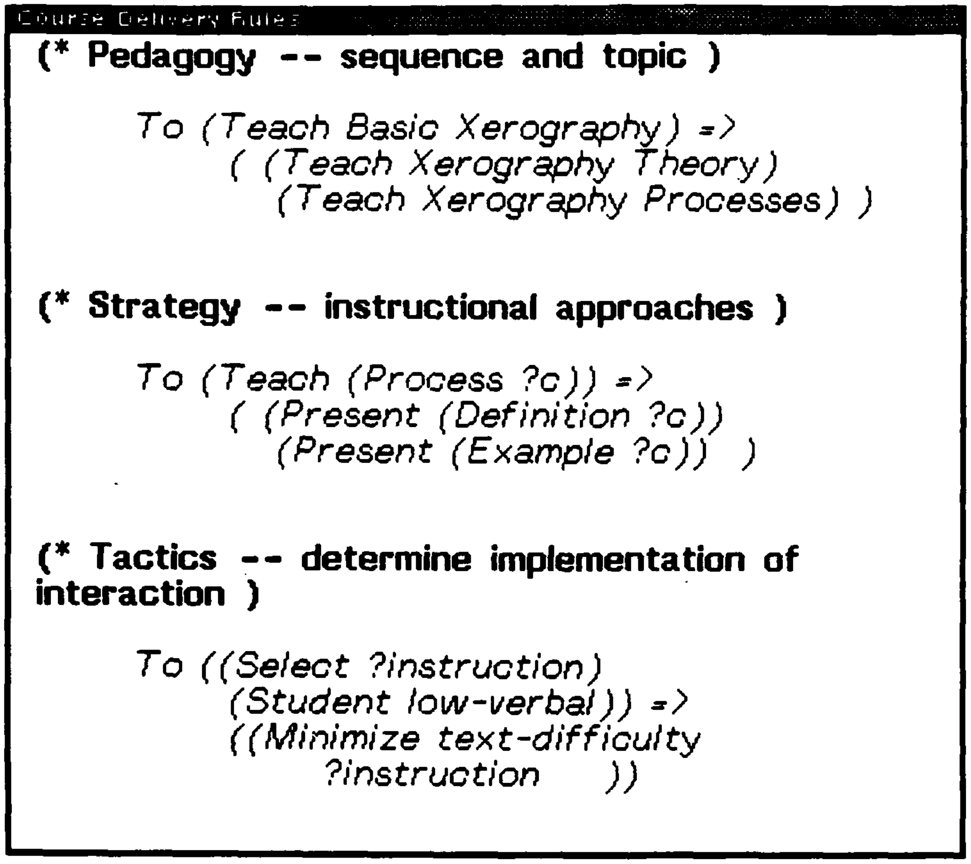
Criteria questions:

1. Given a latent image, how will toned image appear?
2. What does a developer roll do?
3. How is toner different from developer?

Common bugs:

1. Confusion of toner with developer
2. Relative forces of PR - toner; toner - transfer dicor.

Figure 3.10 -- An entry in the MOS is a card that represents either a student concept / skill, or a card representing a student's learning behavior.



(* Pedagogy -- sequence and topic)

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

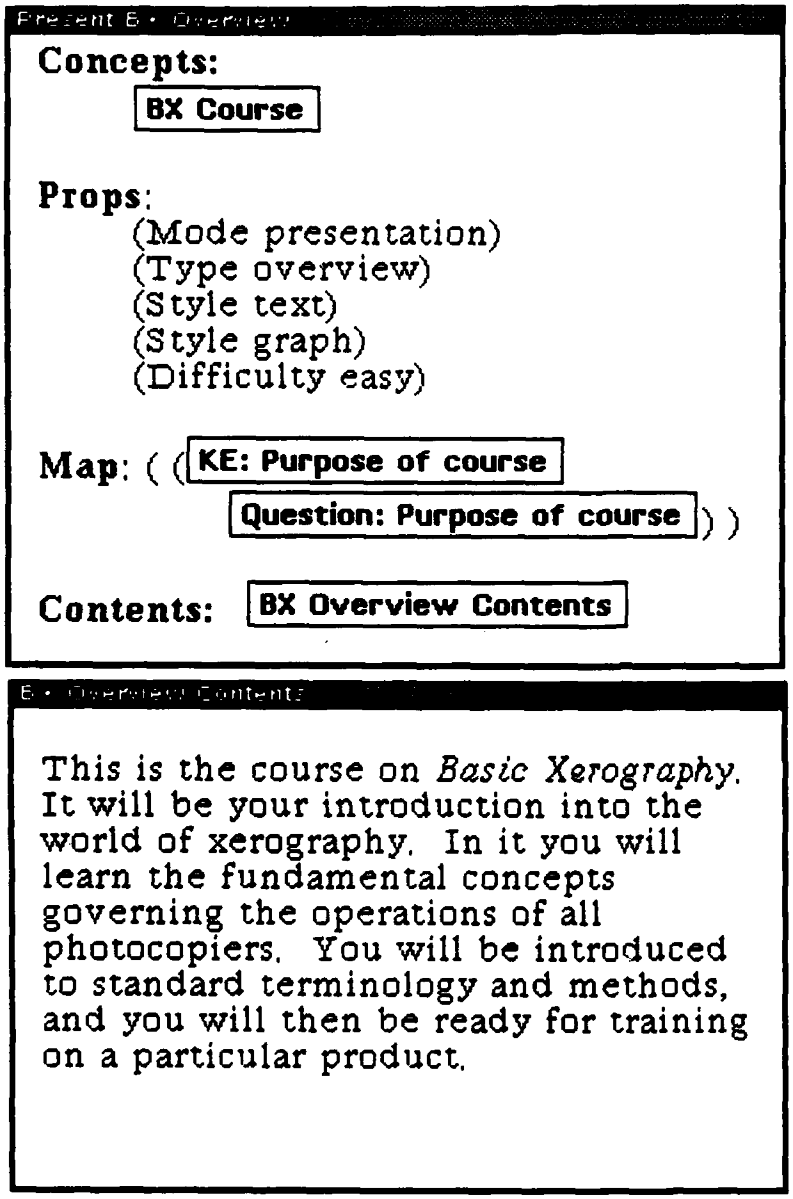
(* Strategy -- instructional approaches)

```
    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 3.11 -- Instructional strategy and tactics are represented in pedagogy, strategy or tactics rules. These cards specify how the course content -- whose conceptual structure is represented in the KS, and whose presentation form is represented by the IUs -- is to be delivered to the students.



Concepts:
BX Course

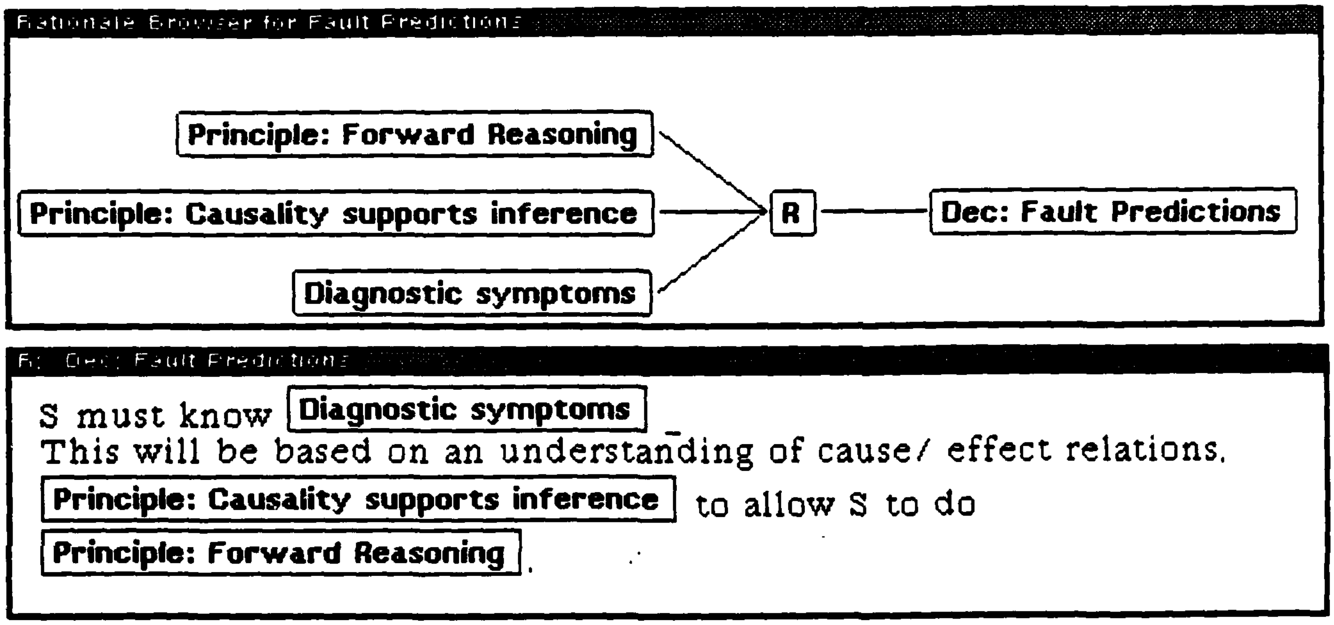
Props:
(Mode presentation)
(Type overview)
(Style text)
(Style graph)
(Difficulty easy)

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

Contents: BX Overview Contents

This is the course on *Basic Xerography*. It will be your introduction into the world of xerography. In it you will learn the fundamental concepts governing the operations of all photocopiers. You will be introduced to standard terminology and methods, and you will then be ready for training on a particular product.

Figure 3.12 -- An Instructional Unit contains the presentation of course material to the student. The IU represents an instruction segment. The Concepts slot records what concepts in the KS this IU teaches; the Props slot is an attribute / value list representing properties of the instructional display; the Map slot points to questions that test understanding of some segment of this knowledge; and the Contents slot points to the actual substance of the display. In this figure, the "BX Overview Contents" card is what the student would actually see.


Rationale Browser for Fault Predictions
Principle: Forward Reasoning
Principle: Causality supports inference
Diagnostic symptoms
R
Dec: Fault Predictions
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 4.1 --Rationalizing the Fault Prediction decision.

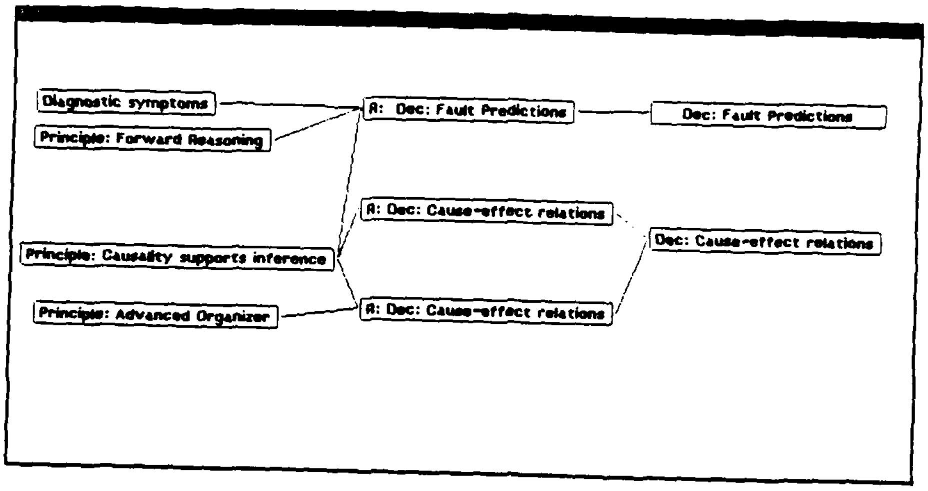
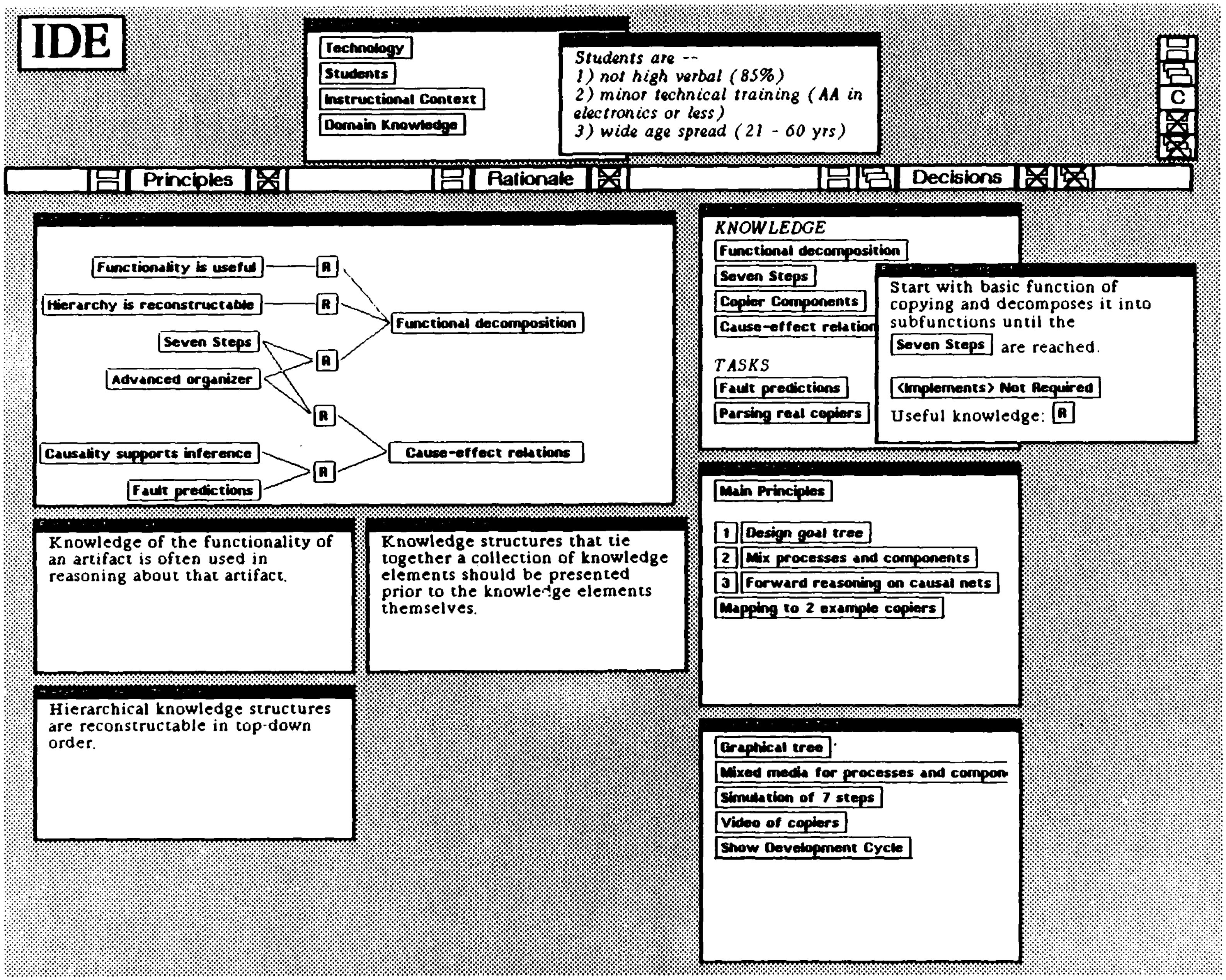

Diagnostic symptoms
Principle: Forward Reasoning
Principle: Causality supports inference
Principle: Advanced Organizer
R: Dec: Fault Predictions
R: Dec: Cause-effect relations
R: Dec: Cause-effect relations
Dec: Fault Predictions
Dec: Cause-effect relations

Figure 4.2 -- This checker shows the rationale links between Principles and Instruction. A checker browser shows how cards in one area map onto cards into another area via some specified path.



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

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 knowle·ge elements themselves.

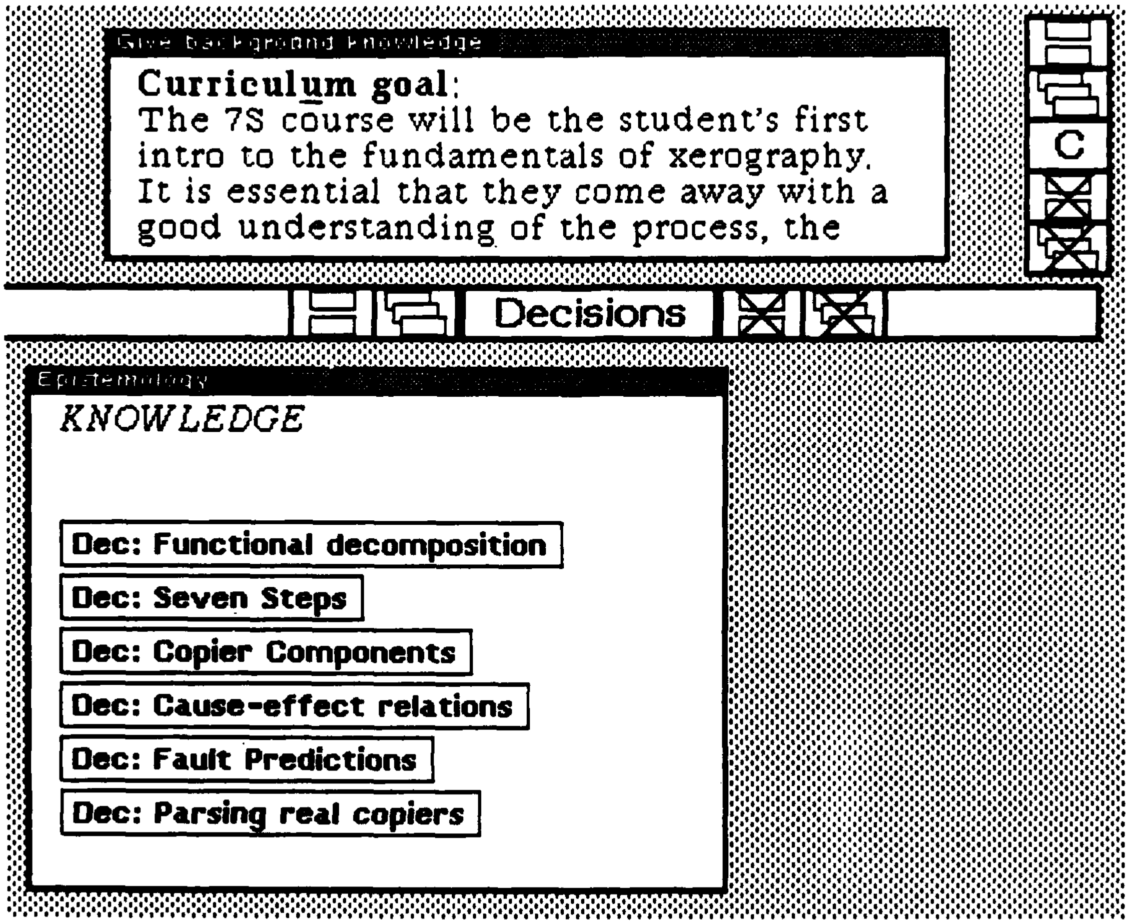
Hierarchical knowledge structures are reconstructable in top-down order.

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

(Implements) Not Required

Useful knowledge: R

Figure 4.3 -- An IDE screen showing the development of the Basic Xerography Course.



Curriculum goal:
The 7S course will be the student's first intro to the fundamentals of xerography. It is essential that they come away with a good understanding of the process, the

KNOWLEDGE

Dec: Functional decomposition

Dec: Seven Steps

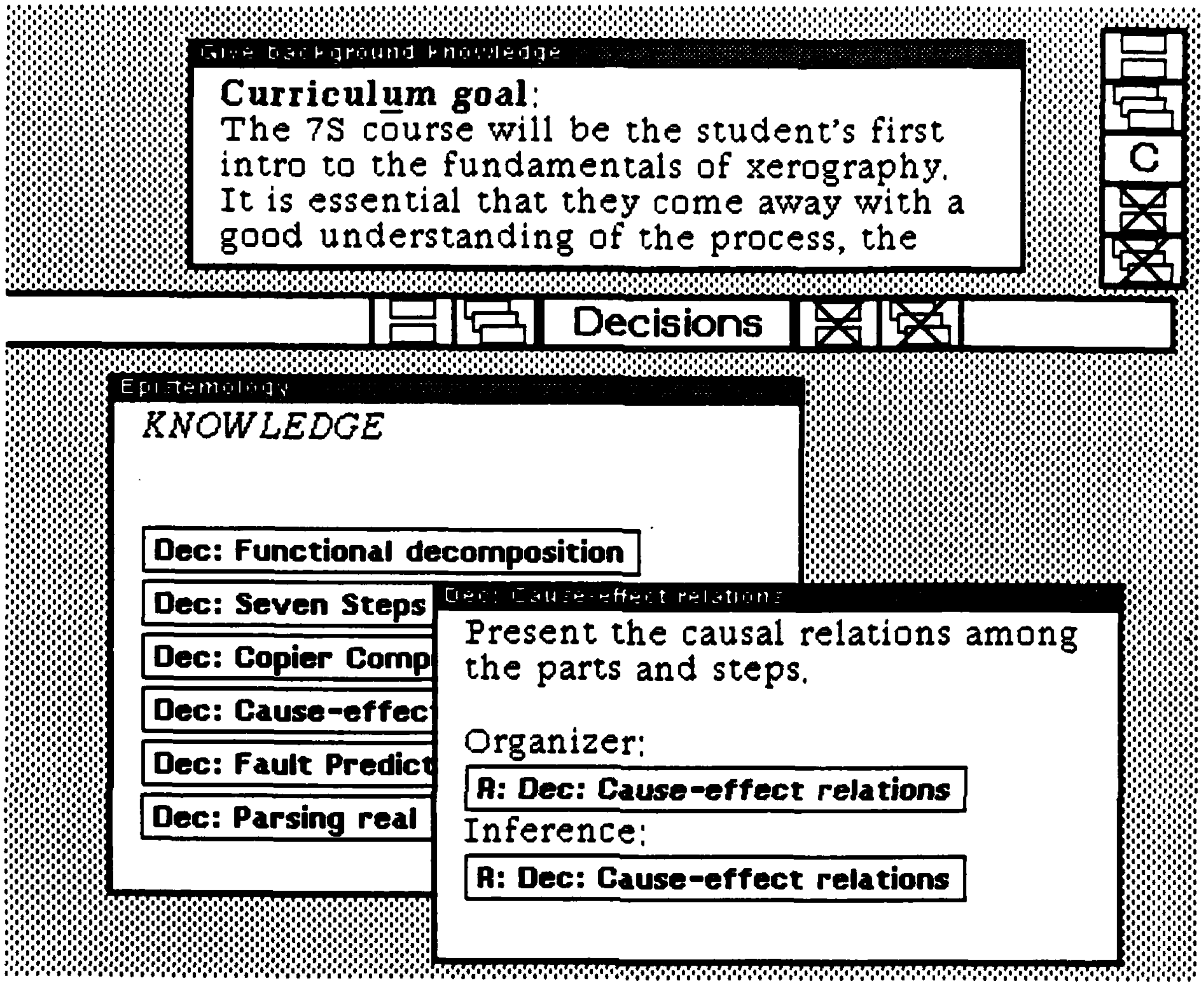
Dec: Copier Components

Dec: Cause-effect relations

Dec: Fault Predictions

Dec: Parsing real copiers

Figure 4.4 -- As a consequence of making Fault Prediction a cognitive task, the epistemology must be modified. The designer has introduced a decision in the Epistemology area to organize content along cause-effect relations.



Curriculum goal:
The 7S course will be the student's first intro to the fundamentals of xerography. It is essential that they come away with a good understanding of the process, the

KNOWLEDGE

Dec: Functional decomposition

Dec: Seven Steps

Dec: Copier Comp

Dec: Cause-effec

Dec: Fault Predict

Dec: Parsing real

Present the causal relations among the parts and steps.

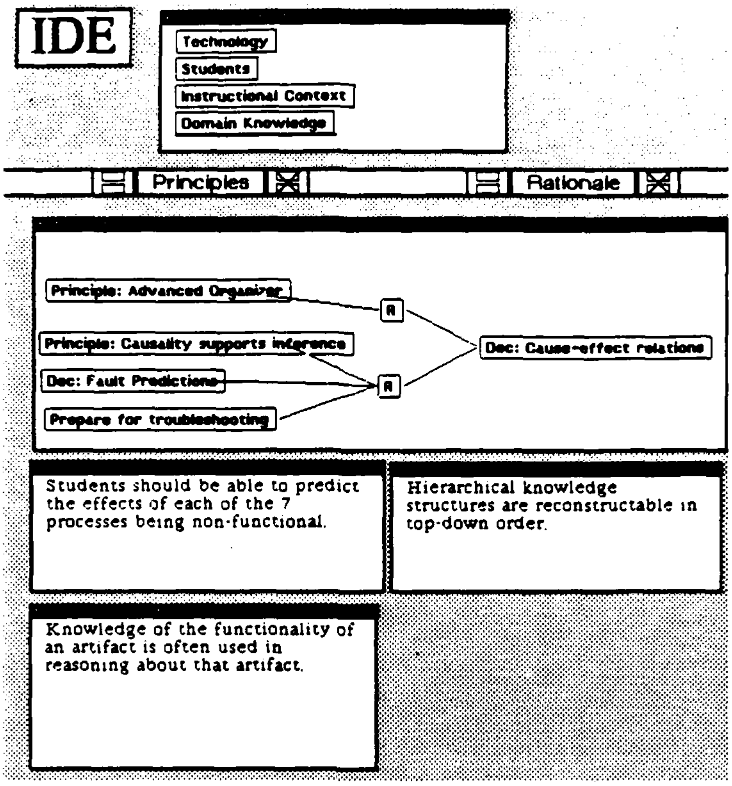
Organizer:

R: Dec: Cause-effect relations

Inference:

R: Dec: Cause-effect relations

Figure 4.5 -- Adding a new decision to teach "Cause-effect" relations.

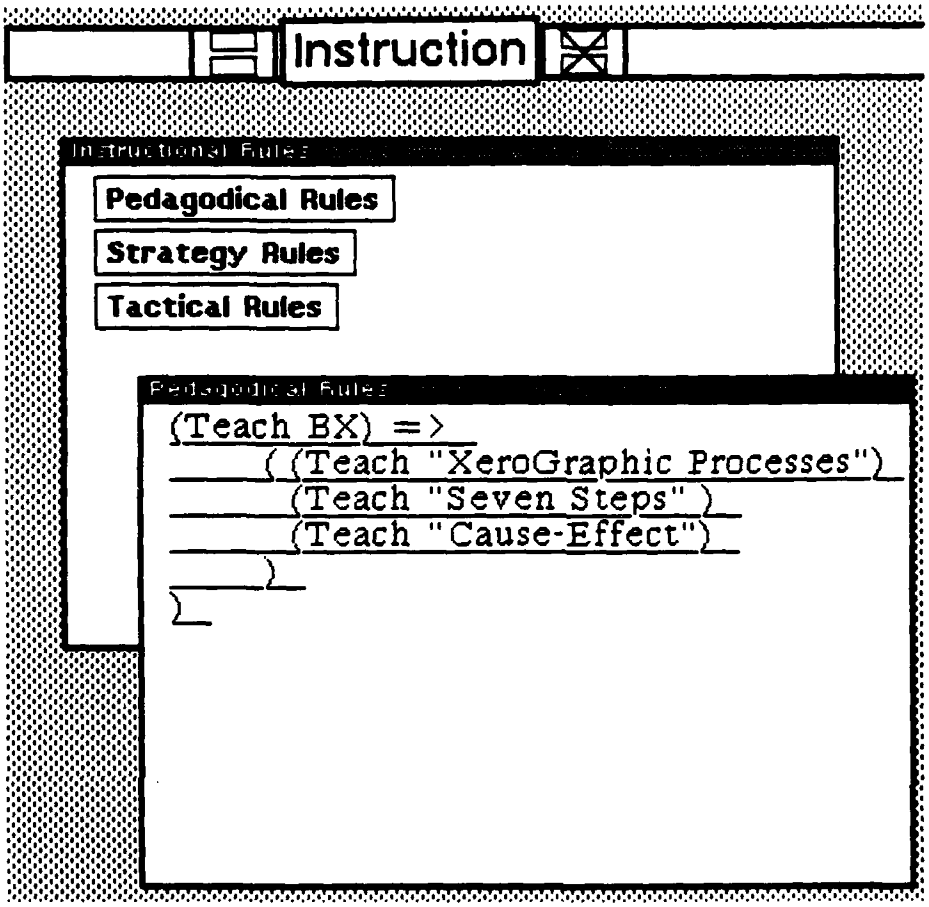

IDE
Technology
Students
Instructional Context
Domain Knowledge
Principles
Rationale
Principle: Advanced Organizer
Principle: Causality supports inference
Dec: Fault Predictions
Prepare for troubleshooting
R
R
Dec: Cause-effect relations

Students should be able to predict the effects of each of the 7 processes being non-functional.

Hierarchical knowledge structures are reconstructable in top-down order.

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

Figure 4.6 -- Rationalizing the Cause-effect Epistemology decision



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

Figure 4.7 -- The decision now affects the way the course will be taught. This is recorded in the Instruction column in the Pedagogy rules.

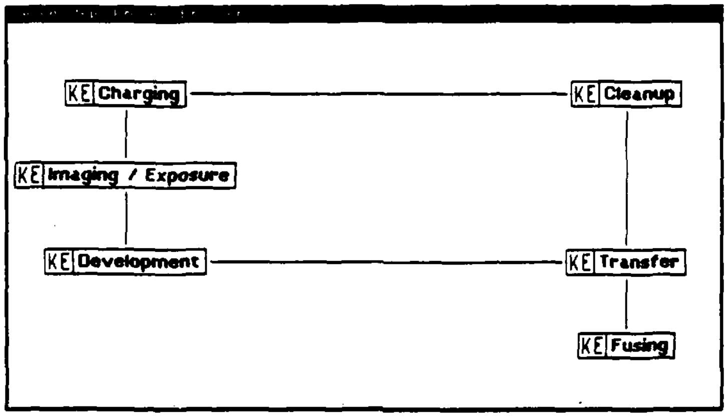


Figure 4.8 -- A sub-browser of the KS

Pedagogy rules:

```
    To (Teach BX) = >
        Segment 1: Teach functions
        Segment 2: Teach steps sequenced by next-step
        Segment 3: Teach normal operations
        Segment 4: Teach broken-state
```

Strategy Rules:

```
    To (Teach functions) = >
        1: Present Function
        2: Teach linked processes
        3: Teach sub-functions
        4: Present summary

    To (Teach broken-state) = >
        1: Present broken-state
        2: Test broken-state

    When (Present ?X, Present ?Y, Present ?Z) = >
        Test ?X or Test ?Y or Test ?Z
        (* after presenting 3 concepts in a row, test for understanding of
        one of the concepts )

    When (Segment beginning) = >
        Test segment understanding

    When (Segment end) = >
        Test segment criterion

    When (Misunderstand ?X) = >
        Remediate ?X
        (* Remidiate immediately on detection of misunderstanding)
```

Tactic Rules:

```
    To (Select IU) = >
        Keep each display as visual as possible
    To (Select IU) = >
        Minimize amount of text to read
    To (Select IU) = >
        Don't use same IU twice to present same concept
```

Figure 4.9 - The Course Content area represents the way the course should be taught. Rules and constraints represent the way an instructional plan should be formed.