

The Use of Decision Trees to Empower Production Technicians to Troubleshoot Routine Process and Equipment Problems

by
Shawn P. Lambert

Bachelor of Science in Chemical Engineering
University of Maine at Orono, 1987

Submitted to
the Alfred P. Sloan School of Management and
the Department of Chemical Engineering
in partial fulfillment of the requirements for the Degrees of

Master of Science in Chemical Engineering
and
Master of Science in Management

at the
Massachusetts Institute of Technology
June 1996

© 1996, Massachusetts Institute of Technology
All Rights Reserved

Signature of Author _____
Department of Chemical Engineering
Sloan School of Management
May 10, 1996

Certified by _____
Rebecca M. Henderson
Professor of Management

Certified by _____
George Stephanopoulos
Professor of Chemical Engineering

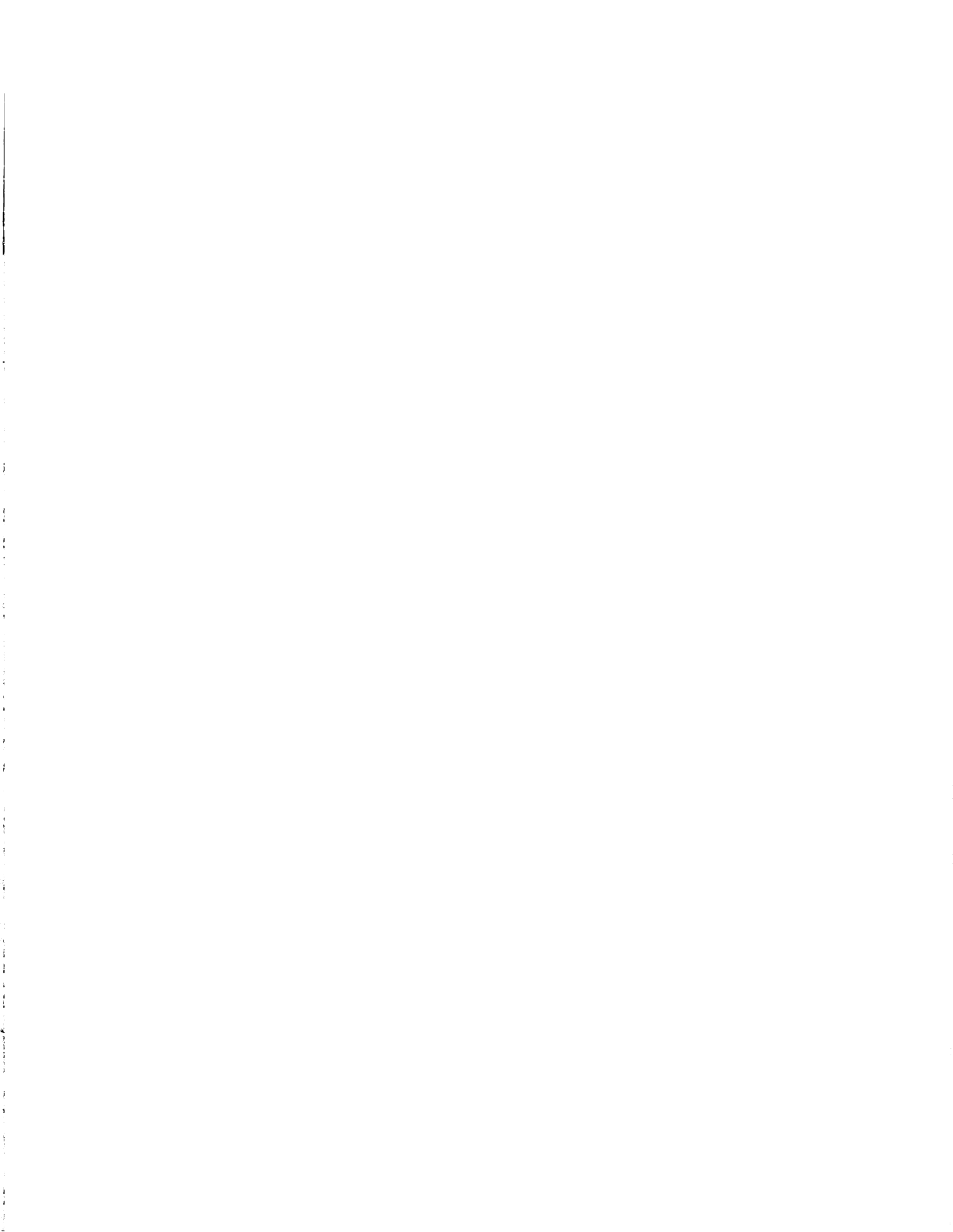
Accepted by _____
Jeffrey A. Barks
Associate Dean, Sloan Master's and Bachelor's Programs

Accepted by _____
Robert E. Cohen
Chairman, Committee on Graduate Students

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

JUN 14 1996 ARCHIVES

LIBRARIES



The Use of Decision Trees to Empower Production Technicians to Troubleshoot Routine Process and Equipment Problems

by

Shawn P. Lambert

Bachelor of Science in Chemical Engineering
University of Maine at Orono, 1987

Submitted to
the Alfred P. Sloan School of Management and
the Department of Chemical Engineering
in partial fulfillment of the requirements for the Degrees of

Master of Science in Chemical Engineering
and
Master of Science in Management

Abstract

Producing semiconductors in a wafer fabrication (fab) facility requires tight control of many process and equipment parameters. When a problem occurs with any one of these parameters, the production technicians traditionally call in representatives from the technical group to troubleshoot the issue. If the production technicians could troubleshoot these problems on their own, they would be resolved more quickly since there would be no response delay. Also, the technical personnel would have more time to perform other tasks such as increasing product yields, improving product performance and decreasing manufacturing cycle time.

This thesis looks at the problem of transferring troubleshooting expertise from the technical personnel to the manufacturing technicians. The goals are to understand the technical and organizational issues, and to develop a troubleshooting guide system that addresses these issues within the constraints of the manufacturing environment. The main component of the troubleshooting guide system is a decision tree based expert system used to capture and transfer the troubleshooting knowledge. However, equally important are the features of the troubleshooting guide system that address the organizational issues of knowledge transfer. Together, these represent a complete system that provide the psychological safety needed for organizational change to occur.

Two case studies using this troubleshooting guide system are presented. Results from these case studies indicate that a troubleshooting guide system based on an expert system can effectively transfer troubleshooting knowledge. However, the difficulties of knowledge acquisition and the time constraints of technical resources to manage such a system were not fully addressed by this troubleshooting guide system.

Thesis Advisors:

Rebecca M. Henderson, Professor of Management Science
George Stephanopoulos, Professor of Chemical Engineering

ACKNOWLEDGEMENTS

I wish to acknowledge the Leaders for Manufacturing Program (LFM) for its support of this work. The internships are one of the many ways that LFM's goal of bringing industry, academia, and student together is accomplished. Also, I thank Polaroid Corporation for sponsoring me in the LFM program.

The research for this thesis was performed at Digital Equipment Corporation's semiconductor division, Digital Semiconductor, in Hudson, Massachusetts. I wish to thank the entire Fab 4 organization for their support and hospitality during my six month internship. Without their candor and assistance, this project could not have been completed. In particular, I would like to thank my company supervisor Dan Welch and members of his group, Heather Benson, Antonio Berti, Ed Bonner, Orion Farina, Kevin Mill and Rodney Renaud for their contributions to TSGuide and the two case studies described in chapter 5. Also, I wish to thank Donna Pendleton, Roger Socha, Beth Comeau, Sandy Marchessault, Al Sandy, and Jose Santana for letting me into the fab to see what really happens. Finally, I wish to acknowledge Erik Bettez, my original company supervisor, Tracy Harrison (LFM '92) and Debbie Sniderman for their help.

I owe a debt of gratitude to my thesis advisors Rebecca Henderson and George Stephanopoulos. Their frequent visits and advice helped me to focus and added greatly to the learning experience.

Finally, I wish to acknowledge and thank my family for giving me the confidence and emotional support to pursue this program. Words can not describe my gratitude to my wife, Paula, for her patience and sacrifices over the last two years.

I dedicate this thesis to my daughter Annika. Her birth at the end of my internship made this experience even more memorable.

Table of Contents

CHAPTER 1 INTRODUCTION	8
BACKGROUND	9
RELEVANT LITERATURE.....	10
OVERVIEW AND SUMMARY OF THESIS	11
CHAPTER 2 ORGANIZATIONAL ISSUES IN TRANSFERRING KNOWLEDGE	14
ISSUES IN TRANSFERRING TROUBLESHOOTING RESPONSIBILITY	14
ORGANIZATIONAL ISSUES IN FAB 4	19
REQUIREMENTS FOR THE TROUBLESHOOTING GUIDE SYSTEM.....	25
CHAPTER 3 TRANSFER OF TROUBLESHOOTING KNOWLEDGE TO PRODUCTION TECHNICIANS: TECHNICAL ISSUES	28
TROUBLESHOOTING PROBLEM DOMAIN	28
METHODS USED FOR KNOWLEDGE TRANSFER	31
IMPLICATIONS OF USING DECTREE.....	37
CHAPTER 4 TROUBLESHOOTING GUIDE SYSTEM	42
DECTREE	42
TSGUIDE MENU	49
CHAPTER 5 TSGUIDE CASE STUDIES	55
CASE 1: MDT ALUMINUM DEPOSITION TIME ADJUSTMENT	55
CASE 2: OXIDE LOT MONITOR PARTICLE FAILURE	63
TRANSFERRING TSGUIDE THROUGHOUT FAB 4	69
CHAPTER 6 CONCLUSIONS AND FUTURE RECOMMENDATIONS	71
CONCLUSIONS	71
RECOMMENDATIONS	74
APPENDIX A: EXAMPLES OF METHODS USED TO TRANSFER KNOWLEDGE	77
NOTES FILE CHECKSHEET	77
OPERATING SPECIFICATION SECTION	78
DECISION TREE FLOWSHEET	79
APPENDIX B: TSGUIDE MENU PROGRAM CODE LISTING	80
TSGUIDE_MAIN_MENU.COM.....	80
TSGUIDE_MDT_MENU.COM	88
APPENDIX C: MDT ALUMINUM DEPOSITION TIME ADJUSTMENT DECTREE	94
MDT TIME ADJ SUBTREE	94
INITIALIZE VARIABLES SUBTREE: TOP	95
INITIALIZE VARIABLES SUBTREE: MIDDLE	96
INITIALIZE VARIABLES SUBTREE: BOTTOM	97
GET ENTITY DATA SUBTREE.....	98
APPENDIX D: OXIDE LOT MONITOR PARTICLE FAILURE TREE	99
LOT MONITOR FAILS SUBTREE	99
WALKS/LL TESTS SUBTREE.....	100
CHAMBER MQC TESTS SUBTREE.....	101
RETEST 2 AND GAS ONLY WALKS SUBTREES	102

RUN RETEST SUBTREE	103
LOT DISPOSITIONING SUBTREE	104
BRING SYSTEM UP SUBTREE	105
APPENDIX E: RESEARCH METHODS AND SOURCES OF DATA	106
BIBLIOGRAPHY	111

LIST OF FIGURES

FIGURE 2-1: PROCESS FOR TRANSFERRING TROUBLESHOOTING KNOWLEDGE	15
FIGURE 3-1: DECISION TREE	33
FIGURE 4-1: DECTREE DEVELOPMENT WINDOW.....	43
FIGURE 4-2: DECTREE DECISION TREE NODES	44
FIGURE 4-3: DECTREE EXAMINE INPUT NODE DIALOG BOX	45
FIGURE 4-4: DECTREE SUBROUTINE NTCDAT.....	47
FIGURE 4-5: DECTREE SUBROUTINE SPCGRAPH	48
FIGURE 4-6: TSGUIDE MAIN MENU SCREENS.....	50
FIGURE 4-7: TSGUIDE MDT TOOL GROUP MENU SCREENS.....	51
FIGURE 4-8: AREA2 SECTION OF CODE FROM TSGUIDE MAIN MENU PROGRAM	53
FIGURE 5-1: SCHEMATIC DIAGRAM OF SPUTTER PROCESS	56
FIGURE 5-2: DEPOSITION TIME ADJUSTMENT PROCEDURE.....	57
FIGURE 5-3: MDT DEPOSITION TIME ADJUSTMENT DECISION TREE	58
FIGURE 5-4: STRUCTURE OF THE MDT ALUMINUM DEPOSITION TIME ADJUSTMENT DECISION TREE	60
FIGURE 5-5: SCHEMATIC OF OXIDE CLUSTER TOOL.....	63
FIGURE 5-6: SIMPLIFIED DECISION TREE FOR OXIDE LOT MONITOR PARTICLE FAILURE	65
FIGURE 5-7: STRUCTURE OF OXIDE LOT MONITOR PARTICLE FAILURE DECTREE DECISION TREE	66

LIST OF TABLES

TABLE 3-1: COMPARISON OF KNOWLEDGE TRANSFER METHODS USED IN FAB 4.....	35
TABLE 3-2: EVALUATION OF REQUIREMENTS FOR AN EXPERT SYSTEM IN FAB 4	38

Chapter 1 Introduction

Producing semiconductors in a wafer fabrication (fab) facility requires tight control of many process and equipment parameters. When a problem occurs with any one of these parameters, the production technicians traditionally call in representatives from the technical group to troubleshoot the issue. If the production technicians could troubleshoot these problems on their own, they would be resolved more quickly since there would be no response delay. Also, the technical personnel would have more time to perform other tasks such as increasing product yields, improving product performance and decreasing manufacturing cycle time.

The knowledge that the technical personnel have acquired from performing troubleshooting activities needs to be transferred to the production technicians before they can troubleshoot on their own. However, transferring knowledge from the technical personnel to the production technicians is difficult. First, there is the issue of the actual knowledge. The technical representatives start with specialized training and then build their troubleshooting expertise over time. Extracting this knowledge and putting it into a format that the production technicians can use must be done. Second, having the production technicians responsible for troubleshooting involves organizational change, control systems, and incentive systems. Finally, this knowledge transfer must occur within the time and resource constraints of a manufacturing operation.

This thesis looks at the problem of transferring troubleshooting expertise from the technical personnel to the manufacturing technicians. The goals are to understand the technical and organizational issues, and to develop a troubleshooting guide system that addresses these issues within the constraints of the manufacturing environment. The main component of the troubleshooting guide system is a decision tree based expert system used to capture and transfer the troubleshooting knowledge. However, equally important are the features of the troubleshooting guide system that address the organizational issues

of knowledge transfer. Together, these represent a complete system that provide the psychological safety needed for organizational change to occur.

Two case studies using this troubleshooting guide system are presented. Results from these case studies indicate that a troubleshooting guide system based on an expert system can effectively transfer troubleshooting knowledge. However, the difficulties of knowledge acquisition and the time constraints of technical resources to manage such a system were not fully addressed by this troubleshooting guide system.

Background

This thesis is based on research conducted during an internship under the Leaders for Manufacturing Program at the Massachusetts Institute of Technology. The internship was performed at Digital Equipment Corporation's semiconductor manufacturing division, Digital Semiconductor, located in Hudson, Massachusetts. The Hudson site is Digital Semiconductor's (DS) headquarters. All phases of semiconductor manufacturing from design to final package test occur on this site. Two semiconductor wafer fabrication (fab) facilities are located at the Hudson plant. The project completed as part of this thesis was carried out in the Fab 4 wafer manufacturing organization.

Fab 4 Wafer Manufacturing Organization

Digital Semiconductor's Fab 4 operation produces the company's flagship Alpha¹ microprocessor and other peripheral chips. The operation uses a 0.5 μ m, 4-metal layer, complimentary metal oxide semiconductor process (CMOS 5).

The Fab 4 organization consists of two groups—the production group and the technical group. Both groups report to the Fab 4 manager. The production group is responsible for manufacturing products consistent with the operating specifications. The technical group is responsible for improving the manufacturing process to increase product yield, lower cycle time and decrease cost. The technical group also supports the production

¹ Alpha is a registered trademark of Digital Equipment Corporation

organization by troubleshooting process and equipment problems. Additional information about the Fab 4 organization is provided in Appendix E.

Thesis Problem

The project description provided by DS is as follows:

When we run into process problems or tool faults on the manufacturing line, frequently process engineering representatives are called to the line to determine problem sources and solutions. In many cases, the manufacturing technicians can be empowered to carry out this same analysis, particularly when technicians have significant experience on a given tool or process. Decision trees are used to walk techs through a process of problem identification, verification, and solution. They are already implemented in some areas of the fab, but inconsistently and not at all in some areas. How do we further and standardize our use of decision trees? Pilot a project in one technical area of the fab, understanding the process issues, risks, potential problems, etc... Apply learnings across the fab.¹

In short, the project description calls for transferring the knowledge to troubleshoot routine process and equipment problems from the technical group to the production group. This problem description served as the focus of this thesis. The emphasis is on developing a troubleshooting guide system that meets these requirements and implementing it in Fab 4. This troubleshooting guide system is referred to as TSGuide.

Relevant Literature

Much has been written on the subjects of knowledge transfer, expert systems, and organizational change. A brief review of some of that literature follows. Additional references are cited directly in this thesis.

The concept for developing a troubleshooting system that addresses both the technical and organizational issues came from an article by Edgar Schein, "How Can Organizations Learn Faster? The Challenge of Entering the Green Room", (Sloan Management Review, Winter 1993). TSGuide is an attempt to provide the Fab 4 organization with the necessary psychological safety for organizational change. Also, the concepts in Schein's

¹ Electronic mail message from Tracy Harrison of Digital Equipment Corporation

“Organizational Culture and Leadership” were very useful in the culture analysis of the Fab 4 organization presented in chapter two.

The issues of creating knowledge have been discussed in the literature in a number of texts. P. Senge’s “The Fifth Discipline”, D. Leonard-Barton’s “Wellsprings of Knowledge”, and R. Thomas’s “What Machine’s Can’t Do” are all excellent sources of the issues affecting knowledge creation and transfer within organizations. They approach the topic with the goal of creating or understanding learning organizations. Introductory papers providing the reader with a feel for these texts can be found for Senge and Leonard-Barton in “The Leader’s New Work: Building Learning Organizations” (Sloan Management Review, Fall 1990) and “The Factory as a Learning Laboratory” (Sloan Management Review, Fall 1992) respectively.

The late 1980’s saw a number of books published on expert systems. For basic concepts and examples of industrial applications the interested reader should look at “The Prentice Hall Guide to Expert Systems” by R. Edmunds, and “Putting Expert Systems into Practice” by R. Bowerman and D. Glover. More recent information on expert systems can be found in two conference proceedings: “Progress in Case-Based Reasoning” edited by I. Watson and “Moving Towards Expert Systems in the 21st Century” edited by J. Liebowitz. The latter is a large collection (over 1500 pages) covering a range of topics in the field of expert systems. The “IEEE Transactions on Semiconductor Manufacturing”, a technical journal, is a good source of semiconductor specific expert systems.

Overview and Summary of Thesis

This thesis describes TSGuide, a troubleshooting guide system, developed to address both the technical and organizational change issues of transferring troubleshooting responsibility to the production technicians.

Chapter two examines the non-technical factors of organizational culture, control mechanisms, incentives, and support systems. These factors are necessary to support empowerment of the manufacturing technicians to troubleshoot process and equipment problems. Five requirements for the troubleshooting guide system based on these non-

technical issues are identified. The troubleshooting guide system must (1) solve complex problems; (2) be easy to use by the PIMT's; (3) be created and maintained by the technical groups using minimal new skills; (4) document troubleshooting actions; and (5) be compatible with existing systems.

Chapter three examines the problem domain for troubleshooting the DS, Fab 4 process and equipment issues. Characteristic of these problems are that both heuristic and algorithmic approaches are used to solve them. The problem diagnosis often requires information from many different sources. Also, domain independent knowledge is required to solve these problems. Four methods currently used in Fab 4 to transfer troubleshooting knowledge are evaluated against the requirements identified in chapter two. None of the methods adequately addressed these requirements. A proposed method, DECTree, is shown to be superior to the current methods in meeting these requirements. The advantages of DECTree over other expert systems is discussed. These advantages include the decision tree method of knowledge representation, compatibility with Fab 4 computer systems, and availability.

The two components of TSGuide, DECTree to create troubleshooting guides and the TSGuide menus for distributing the troubleshooting guides, are described in chapter four. The goal of this chapter is to show how the components of TSGuide meet the requirements identified in chapters two and three.

In chapter five, two case studies from a technology area of Fab 4 are presented to illustrate the use of TSGuide. The two cases experienced different results regarding implementation and use. Also, the progress of transferring TSGuide to other technology areas of Fab 4 is described. Transferring TSGuide to other technology areas of Fab 4 has been linked to another Fab 4 initiative—process empowerment program (PEP).

Chapter six presents conclusions based on the results of chapter 5 and makes recommendations for proceeding with TSGuide. The main conclusion is that TSGuide does address both the technical and organizational issues of knowledge transfer. However, certain characteristics of troubleshooting problems have an effect on the

implementation and use. Also, the issues of knowledge acquisition and time constraints of technical personnel are not adequately handled by TSGuide. Recommendations include establishing a focus person to facilitate transferring TSGuide throughout the technology areas in Fab 4, using a continuous improvement approach to developing troubleshooting guides, and initiating TSGuide in Fab 6.

Chapter 2 Organizational Issues in Transferring Knowledge

The goal of this chapter is to identify requirements for the troubleshooting guide system, TSGuide, that will address the non-technical factors of knowledge transfer. First, the issues of organizational change, control and incentive systems, and support functions will be presented. Next, the existence of such factors in the Fab 4 organization will be discussed. Finally, the requirements for TSGuide to address these issues will be identified.

Issues in Transferring Troubleshooting Responsibility

Figure 2-1 depicts a process for transferring and upgrading troubleshooting knowledge. This process emphasizes that the lack of expertise by the production technicians to troubleshoot production problems is the primary reason that a troubleshooting system is necessary. The expertise that the technical personnel possess must first be transferred to the production technicians before they can troubleshoot their process and equipment issues. Expert systems are frequently used to accomplish this by distributing the knowledge of experts to non-expert users.

However, there are other factors in addition to the lack of technical expertise involved in transferring the troubleshooting responsibility from the technical personnel to the manufacturing technicians. The process in Figure 2-1 does not occur in a vacuum. How might the existing organizational culture resist such a process? Can control over the quality of the work still be maintained? What incentives and support groups are necessary

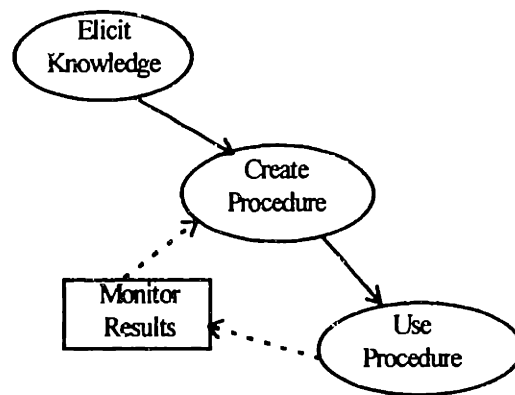


Figure 2-1: Process for Transferring Troubleshooting Knowledge

to make this process succeed? These are the non-technical issues involved in knowledge transfer. Ignoring these factors could hinder the success of a well devised technical solution, such as an expert system, that only considers the expertise and not the organization that must create and use the expertise.

This section discusses the issues of organizational change, control, incentives and support systems required to transfer the troubleshooting responsibilities to the production technicians.

Organizational Change

Organizational change requires new organizational learning. According to Schein, for organizational learning to occur the anxiety associated with change (Anxiety 1) needs to

be reduced to a level below that of the anxiety of not changing (Anxiety 2).¹ He proposes that providing psychological safety is the key to reducing and managing Anxiety 1.

Sources of Anxiety

The sources of both Anxiety 1 and Anxiety 2 come from the underlying assumptions of the current organizational culture. People resist change because it upsets the comfortable equilibrium that their current culture provides. The artifacts that one can see when observing an organization, such as dress code, hierarchical structure, information systems, and office layout, are all attributed to the shared underlying assumptions of that organization.

Edgar Schein formally defines culture as:

*A pattern of shared basic assumptions that the group learned as it solved its problems of external adaptation and internal integration, that has worked well enough to be considered valid and, therefore, to be taught to new members as the correct way to perceive, think, and feel in relation to those problems.*²

The sources of anxiety 1 and anxiety 2 for the change process under consideration can be determined after the underlying assumptions of an organizations culture have been identified. To the group members, these shared basic assumptions exist on an unconscious level which makes identifying them difficult. However, the key to a successful change process involves doing just that—identifying key assumptions and using them.

Psychological Safety

Providing psychological safety means lowering the risk, and therefore the anxiety, of behaving in a manner not consistent with the current organizational culture. For example, if people are fearful of taking on new tasks because failure results in punishment then, psychological safety comes from providing an opportunity for people to take on new tasks without the consequences of failure.

Control Systems

Maintaining control over the process in Figure 2-1 is a non-technical issue associated with transferring troubleshooting responsibilities to the production technicians.

Control systems act to insure that the correct procedures are followed while accomplishing desirable outcomes. Without adequate control systems, employees can pursue objectives in a manner that is not acceptable to the company. For example, a production worker given the goal of decreasing cycle time might experiment with reducing the frequency of measurements designed to test the outcome of the process. This production experiment provides valuable insight and if successful leads to a lower cost process. However, what if that same production worker decided that the quality testing procedures were unnecessary and stopped running them, but continued to enter results into the database that indicated that the process was meeting specification? Cycle time will still be reduced, but if the testing procedure turns out to be important a disaster could happen.

Control Levers

How can control be maintained when troubleshooting responsibility is transferred to the production technicians? Simons describes four different control levers that can be used to monitor organizations without hindering innovation.³ The four levers are belief systems, boundary systems, diagnostic control systems and interactive control systems.

By applying each of the four levers described above, the creative potential of employees to create value for the company can be unleashed. The four control levers help to monitor, constrain, and steer the work force to prevent disaster while insuring that the correct objectives are focused on and that the approaches are consistent with the companies expectations.

Belief Systems

The belief system of an organization are the core values that enable employees to seek out guidance in their decision making. Simons cites the Johnson and Johnson credo which during the Tylenol crisis limited the types of solutions that managers developed for dealing with the problem. Belief systems, like J&J's credo, inspire, motivate and give purpose to employees.

Boundary Systems

Boundary systems clearly spell out what behavior is not acceptable to accomplish goals. By drawing these boundaries around what not to do, employees are free to innovate within the established boundaries. Because the purpose of boundary systems is to prevent certain actions, but not explicitly state how to accomplish a goal, creativity and innovation are not stifled.

Diagnostic Systems

Diagnostic control systems are used to track and monitor performance of individuals and groups toward established goals. These systems provide feedback on results. However, there is no indication of how the results were accomplished. An SPC chart that monitors a critical quality parameter is an example of a diagnostic control system.

Interactive Systems

Interactive control systems allow managers to interact with their subordinates about the assumptions and action plans for various problems and opportunities. The focus of these systems are broader with a longer time horizon than diagnostic control systems. For example, new technologies, government regulation, customer's preferences; information that managers consider strategic. By regularly sharing and discussing this information with employees, actions are adjusted to reflect the changing environment.

In the context of empowering the production technicians to perform troubleshooting, interactive control systems provide an opportunity for the technical personnel and manufacturing technicians to share feedback and address ongoing concerns.

Incentive Systems

Ideally, an organization's incentive system rewards desirable behavior and deters undesirable behavior. However, these systems can drive behavior in an unintentional way or hinder success of new approaches.

Behnke, et al describes the evolution of employee empowerment at Harris Semiconductor.⁴ They identified that their current individual based system for career pathing, performance appraisal, and pay did not support their move to self directed work

teams (SDWT's). For example, the emphasis on individual contributions in performance reviews did not promote team work, and the pay system did not provide a mechanism to reward the results of SDWT's. A new review process that weighed individual, fab and team goal performance equally was instituted. A monetary incentive system called "gainsharing" was implemented that returned a percentage of the dollar savings to the SDWT's.

However, sometimes the most powerful incentives to motivate workers are self contained. Managers typically underestimate the desire of employees to take on more responsibility. The increase in job satisfaction from having greater control over one's own job can motivate these workers. One manager at DS described his experience at another semiconductor company he worked at that had implemented a troubleshooting guide system. To his surprise, the production workers readily accepted the additional responsibility because of this increase in job satisfaction.

Support Systems

Several support systems are necessary to aid the production technician in performing troubleshooting. Additional training on troubleshooting procedures and information systems to access the data needed to diagnosis production problems are examples of such support systems.

Organizational Issues in Fab 4

This section will examine the organizational issues discussed in the previous section for the specific case of Fab 4. The findings of this section will be used to develop the requirements for TSGuide needed to address these organizational issues and provide psychological safety to the Fab 4 organization for transferring the troubleshooting responsibilities to the production technicians.

Appendix E contains information on the organizational structure of Fab 4 that serves as background information for the discussions that follow.

Organizational Culture

This section will examine three basic assumptions of the Fab 4 culture at Digital Semiconductor. The implications of these assumptions for TSGuide to provide psychological safety is the goal of the analysis.

Shared Assumptions

The three assumptions are:

- *Product Driven vs. Manufacturing Driven:* Product performance and time to market are more important than manufacturing cost.
- *High Involvement of Technical Expertise:* Operating high performance processes requires constant monitoring by technical personnel who possess expertise.
- *Cost of Empowerment Outweigh Benefits:* The cost of mistakes made by production technicians outweigh the benefits of pushing the decision making down to a lower level of control.

These assumptions are not universally shared across Digital Semiconductor. In fact, an espoused value at Digital Semiconductor stresses the importance of lowering the costs of manufacturing products as important to the continued success of the Division. Many individuals in the Fab 4 organization have previous work experience at other companies whose cultures are different. Also, change is occurring to the basic assumptions as the groups succeed in addressing the divisional challenges of being a business unit versus a cost center. However, these assumptions are prevalent enough to affect the change process of empowering production employees to troubleshoot process and equipment problems.

Product Driven vs. Manufacturing Driven

The primary benefit that Digital Semiconductor provides to Digital Equipment Corporation is the delivery of state-of-the-art microprocessors and other semiconductor devices. DS accomplishes this by combining design and microelectronic fabrication expertise. The initial *new* design takes full advantage of the latest generation of CMOS process technology design rules. Additional learning, once in production, occurs to increase die yield and further improve performance on key parameters, i.e. speed. The

final phase in the product's life cycle is a migration, called a shrink, to the next generation of CMOS technology being introduced for the next *new* design. This life cycle is on the order of a few years.

Examples of where product performance is emphasized over manufacturing cost during this life cycle can be seen at all stages. In the initial design phase, the die size is specified. This decision will impact the amount of space the designers can allocate to transistors which correlates to product performance. The production cost is also impacted by this decision. A larger die increases the probability of a defect rendering it useless. Also, the larger the die the fewer chips per silicon wafer. During the design phase for Digital's flagship microprocessor, Alpha¹, a large die size was chosen to increase the performance at the expense of manufacturing cost through lower theoretical yields and fewer die per wafer.

During the process development phase, the process technology is chosen to deliver the specifications. The extremely fast transistors employed by the Alpha design requires a novel isolation technique. The implication of this technique is that many more steps are needed in the manufacturing process over conventional isolation technology. These additional steps increase manufacturing costs, but product performance is enhanced.

The choice of planarization technology in the development of CMOS 5 is an example of where time to market was more important than manufacturing cost. An existing local planarization technique was chosen over a global planarization technology which required further development and had risk associated with meeting the product timeline.

The three examples described above of decisions made in favor of product performance over lower manufacturing cost are artifacts of the assumption that product delivery and performance are more important than production cost. The success of computer systems in the market built with the Alpha microprocessor validate these decisions and reinforce this belief.

¹ Alpha is a registered trademarks of Digital Equipment Corporation

High Involvement of Technical Expertise

The Fab 4 organization was historically a pilot line for CMOS process development and a second source for semiconductor production. Three generations of CMOS process technology have been developed in Fab 4. To support the learning required for process development, extensive amounts of process variables are tracked, machine quality checks (MQC's) are performed frequently, and lot monitors are used to verify process conditions prior to running a product lot. The technical personnel with specialized backgrounds, formal training, or many years of experience are involved in analyzing this data to increase the level of knowledge about the process. The focus is on improving yields and increasing the clock speed.

Using Fab 4 as a pilot line served as a source of innovation. However, when Fab 4 evolved into the primary supplier for CMOS 5 devices, this source of innovation became a constraint on low cost, high volume production. For example, the procedure of verifying that equipment is functioning correctly prior to running product is a major constraint to material flow through the production processes. Also, the presence of technical personnel with process expertise caused the production operation to become dependent on those resources since no opportunity existed for them to develop their own expertise. This is particularly true for troubleshooting knowledge.

Cost of Empowerment Outweighs Benefit

In a risk/benefit analysis, the benefits of an action are compared to the cost associated with performing that action. Actions are implemented only if their benefit outweighs the cost. A formal risk benefit analysis requires accurate information about both the costs and the benefits of proposed actions. This information is often difficult to quantify, but for large investments management control systems require such an analysis.

For smaller projects, particularly those whose benefits are lower operating costs and the expenses are the risk of making off-spec product, risk/benefit analysis is less likely to be done. Examples of this type of project would be elimination of a measurement that rarely causes an SPC violation or training of production technicians to make changes to the

software recipes. In each case, the high cost of making bad product is not offset by the low probability that such an occurrence will actually happen.

The empowerment of production technicians to troubleshoot their own process and equipment problems falls into this perceived low benefit, high risk category.

Psychological Safety

A major source of anxiety 1 for the production technicians is the lack of knowledge on how to troubleshoot their processes. For the technical personnel, anxiety 1 comes from transferring knowledge about troubleshooting to non-technical production workers. The cultural assumptions are the source of anxiety 1.

The source of anxiety 2, the anxiety associated with not changing, also has its roots in the underlying assumptions of the group. In particular, the assumption regarding product performance versus manufacturing driven. Anxiety 2 has come from the need to transfer technical resources to the new fab to help deliver the next generation of CMOS products. The opportunity to lower production costs or increase capacity of Fab 4 have also been presented as a source of anxiety 2, but does not motivate the technical groups as much as helping with the start-up of Fab 6.

Control Systems

Additional uses of the four control levers described previously in this chapter will be required to transfer the troubleshooting responsibilities to the PIMT's. The current boundary control systems are the specifications for each tool group and operation. These specifications are based on the rules used by the designers when laying out the electrical circuits of the chip. The TSGuide system will act as a boundary control system by identifying which problems the PIMT's can work on and when to call in additional resources. By documenting the troubleshooting actions taken, TSGuide will serve as an interactive control system. Also, interactive control will be provided by the annotation software scheduled for installation in the near future. This software will be connected to the SPC limits of critical parameter charts. A violation of an SPC rule automatically puts the lot on hold and the tool down in the resource tracking software. Closure of the

annotation is required to continue processing. Outstanding annotations are tracked and will be discussed at weekly meetings.

Incentives

The technical groups are responsible for the yield, equipment availability, and cycle time metrics. Availability is defined as the percentage of time that the tool is available to production to produce product. The PIMT's are measured on personal performance, skill level, and attendance.

Transferring the troubleshooting responsibility to the PIMT's raises two concerns because of the current incentive system. First, "Why should I do someone else's job and not get paid for it?" is a concern from the PIMT's. Second, the technical groups ask "Why should I give up control of the metrics that I am measured on?".

Resolving these two questions is not as simple as using the metrics of the technical groups to measure the PIMT's and eliminating these metrics from the evaluation of the technical group. The two groups will be required to work together to resolve many equipment and process issues.

The incentive of increased job satisfaction might have an impact. Discussions with the PIMT's indicated an interest in being able to perform their own troubleshooting. Much of the work to diagnose routine problems is currently done by the PIMT's. The technical group representative is used as a consultant to guide the process based on the reported findings. Having control over the entire process has a certain appeal to the PIMT's.

Support Systems

The information systems needed to provide data used by the troubleshooting guides and the diagnostic and interactive control systems already exist in Fab 4. Formal and informal training programs are needed to teach the production technicians how to use the troubleshooting guide system and enhance their skills around repairing equipment and understanding the manufacturing process technology. The process technicians and engineers need training on how to create the troubleshooting guides. Computer system

support is needed to modify and maintain the computer code to integrate the troubleshooting system into the existing shop floor control system.

The infrastructure for all of these support services already exists at DS. There are databases where production, engineering and test data can be accessed. There is a training group which teaches a variety of courses included a program aimed at providing an overview of the CMOS process technology to new employees and PIMT's. A computer classroom is equipped with workstations and an overhead computer screen projection system. A VAX¹ cluster serves as the shop floor control network and is linked to the rest of the corporation.

Unfortunately, the groups that support all of this infrastructure are resource constrained, particularly the computer integrated manufacturing technology (CIMT) group. The support provided to Fab 4 is primarily to maintain production critical applications. The resources to modify and create new applications are committed to the start-up of the new manufacturing plant, Fab 6.

Requirements for the Troubleshooting Guide System

The troubleshooting guide system, TSGuide, will be used to capture, represent, and transfer the knowledge necessary to identify and resolve frequently encountered manufacturing problems. The technical requirements for such a system will be described in chapter 3. However, the non-technical factors discussed in this chapter are important to the implementation and ultimate success of TSGuide to transfer the responsibility for troubleshooting to the PIMT's.

The previous sections discussed these organizational issues and described how the current Fab 4 organization satisfies them. This section will identify requirements that TSGuide needs to meet to insure that these organizational issues are addressed. By incorporating these requirements into TSGuide, the system should provide psychological safety and facilitate a more successful implementation.

¹ VAX is a registered trademark of Digital Equipment Corporation

Handle complex problems

The troubleshooting guides must be powerful enough to resolve most problems. Instead of relying on monetary incentives or changes to other incentives, this system will motivate through increased job satisfaction. If the troubleshooting guide system does not provide a solution to the problem, then the PIMT's will not get increased job satisfaction.

Identify Problems

To act as a boundary control system, the situations that are appropriate for the PIMT's to troubleshoot need to be identified. Also, troubleshooting sessions that require special training to fix or do not follow the normal diagnostic pattern should be terminated by instructing the PIMT's to contact the appropriate technical group representative.

Terminating the troubleshooting session by having the PIMT's contact a technical group representative seems to contradict the previous requirement that the troubleshooting guides must be powerful enough to resolve complex problems. However, a non-routine problem for which expertise does not readily exist or a solution that the PIMT's are not trained to perform are valid exceptions to this requirement. The fewer exceptions that occur, the greater the incentive to use the troubleshooting guide system. Therefore, additional training and continuous improvement to minimize these exceptions is important.

Document Troubleshooting Activity

Documenting the actions taken during a troubleshooting session will function as an interactive control system since these log files can be monitored by personnel in the technical groups. The technical personnel can then have regularly scheduled meetings to discuss usage of the troubleshooting guides and improvement opportunities.

The documentation of actions taken will also facilitate continuation of the troubleshooting activity if it needs to be completed by another shift or someone in the technical group.

New Skills

The shortage of support group resources dictates that TSGuide is designed so that the technical group personnel can create their own troubleshooting guides without acquiring substantial new skills. Additionally, the entire system needs to be easily integrated into the existing shop floor system since resources for doing this integration will be limited. Future updates and improvements must also be easily handled by the technical group personnel.

Easy to Use

TSGuide must be easy to use by the PIMT's. The incentive of increased job satisfaction will not be realized if the PIMT's find using the system burdensome. Also, the available time that the PIMT's have to troubleshoot is limited. A user interface and system that maximizes that time is important. Integrating information from other sources is one way to make the system easy to use and save the end user time.

Chapter 3 Transfer of Troubleshooting Knowledge to Production Technicians: Technical Issues

Chapter 2 discussed the non-technical issues involved in empowering the production technicians (PIMT's) to troubleshoot their process and equipment issues. Several requirements for the troubleshooting guide system were proposed based on that analysis. This chapter will discuss the technical issues of transferring troubleshooting knowledge from the technical personnel to the PIMT's—the primary requirement of the troubleshooting system.

First, the problem domain of troubleshooting microelectronic fabrication processes and equipment will be analyzed. Next, current practices that were observed in Fab 4 to transfer troubleshooting knowledge will be reviewed against the requirements for the troubleshooting guide system developed in chapter 2. Finally, DECtree, an alternative method, will be considered as a general solution for transferring expertise for the entire troubleshooting problem domain.

Troubleshooting Problem Domain

Microelectronic wafer fabrication is an extremely demanding task. To produce the three-dimensional devices that form the electrical circuits of a microprocessor, a silicon wafer must undergo hundreds of individual processing steps. These steps must be executed to tolerances that allow critical dimensions of $0.5 \mu\text{m}$ to be formed in an essentially particle free environment.

The equipment used to manufacture semiconductor devices is also very complex. High energy plasma sources, ultra-high vacuum pumps, mass flow meters, robotics, sophisticated electronic controls and software can all be combined in one tool. More than 25 groups of these tools are used in Fab 4.

The hundreds of processing steps from 25 different tool groups result from the multiple processes run on each tool. These processes utilize chemistry, physics, and material science to add or remove layers of material that form the transistors, interconnects, dielectrics and metal lines of a semiconductor device. Each process is affected by the variation in dozens of variables from previous processing steps, raw materials, machine settings, and hardware aging. Troubleshooting complexity is increased because each tool runs multiple processes.

Narrowing the Problem Domain

The technical group shown in Figure E-2 in appendix E is necessary to support the fab. The typical EE has general training in areas of electronics, mechanics, and mathematics. Further specialized training is obtained from courses for particular tool groups. This training is generally done by the equipment manufacturer. Similarly, the typical PE has an engineering background and has attended specialized training on particular process technology. This baseline knowledge is the starting point for learning the technological knowledge needed to keep the fab operating and improving. Much of this knowledge is not specific to the problem domain, but is necessary to develop the domain specific expertise.

When a new fabrication line is started, the level of technological knowledge inherited from the previous generation of process and equipment technology does not transfer to the new line at the same level. Roger Bohn describes levels of technological knowledge as ranging from complete ignorance (level 1) to complete knowledge (level 8).⁵ When a new fab is brought on-line, processes that use to be understood at the level of process characterization (level 6) and know why (level 7) fall back to process capability (level 5) status because of the new technology that is introduced.

Bringing the level of knowledge back to the previous level requires learning from experiments, problem solving and continuous improvement. These activities are the source for acquiring the domain specific expertise necessary for troubleshooting the process and equipment problems.

Transferring the complete problem domain of troubleshooting semiconductor production problems to the PIMT's is unreasonable. First, not all of the processes and tool group interactions are understood at a high enough level. Also, certain tasks require specialized training in equipment maintenance procedures. Therefore, the appropriate subclass of troubleshooting problems are issues that are well understood and require skills that are already possessed by the PIMT's or can be learned with a reasonable amount of training.

Frequently Occurring Problems

In a mature semiconductor manufacturing plant there are many routine troubleshooting problems that fit the description of the limited problem domain for transferring troubleshooting to the PIMT's. For these problems, troubleshooting usually begins when a monitored parameter violates a rule on a control chart. The PIMT who observes the violation contacts a representative from the technical group. The technical group representative responds to the request and uses his or her expertise to diagnose and fix the problem.

Diagnosing the problem involves several different steps depending on the particular problem. Examples of various diagnostic procedures are examining the SPC chart which was violated for particular patterns, analyzing additional information provided from the measurement tool, reviewing results of other tools running the same process for commonality; and running additional tests to isolate which process step or equipment chamber is at fault. Fixing the problem might involve adjusting process parameters, inspecting and cleaning external parts, replacing monitor wafers, running cleaning recipes, or restricting which processes can be run on the tool until a more extensive repair can be made.

These diagnostic and repair procedures are all reasonable tasks for an experienced PIMT. The knowledge of which diagnosis steps to run for a particular problem and how to interpret the results is the expertise that the troubleshooting guide system needs to transfer.

Characteristics of Problem Domain

The troubleshooting problem domain described in the previous section has several characteristics:

- *The knowledge is highly procedural.* For a given failure the domain expert (EE's and PE's) follow a series of steps designed to identify the particular cause to a problem that potentially has several possible causes.
- *Heuristic judgments are used in diagnosis.* The procedure followed and conclusions made regarding the cause depend on specific facts that initially describe the failure or are uncovered during the diagnosis. The expert is making heuristic judgments—rules of thumb—that are based on experience rather than algorithmic computation.
- *Algorithmic computations are used to determine new recipe settings.* The use of empirical or first principal models to calculate process adjustments are necessary or beneficial for certain solutions.
- *Procedures require information from many sources.* The facts needed to diagnose a problem can be contained in several different databases.
- *Domain independent knowledge and skills are required.* To retrieve the needed information and make recipe adjustments, software and basic computer skills are needed. To make repairs requires mechanical skills.

Methods Used for Knowledge Transfer

Various methods have been tried in Fab 4 to transfer the troubleshooting knowledge to the PIMT's. Examining the four technology areas, four different methods were observed and two other methods were being considered. The four methods in use are troubleshooting section in the operating specifications, Notes file checksheets, decision trees, and a custom C+ program. Examples of the first three methods are included in Appendix A.

This section will describe these four methods and qualitatively rank them on how well they meet the requirements for TSGuide identified in chapter 2. The two methods under consideration, Workstream¹ and DECtree, will also be examined to see if they address the shortcomings of the four current methods.

¹ Workstream is a registered trademark of Consillium, Inc.

Specification Section

Each tool group in the Fab 4 has an official operating specification that describes the procedures for each process that can be run on the tool group. A troubleshooting section was included in the operating specification for most of the tool groups. These troubleshooting sections were typically written as a procedure to be followed when a machine quality check (MQC) failed. Procedures were generally limited to a re-test and contacting either EE or PE if the re-test failed.

Notes file Checksheet

The VAX computer system at Digital Equipment Corporation supports a VMS¹ software program called Notes. This program allows computer users to share information on different topics through Notes conferences. A Notes conference is a collection of topics and replies which pertain to the main conference topic. Users either reply to a topic or create a new topic. The resulting conference serves as a knowledge sharing tool.

Each of the technology areas have Notes conferences that are specific to their area. Within these conferences are topics for each tool group or individual tool (entity). These Notes conferences document troubleshooting activity, equipment preventative maintenance, and shift passdown information. This information can be searched and read by other shifts to help them do their job. The EE and PE groups are the primary users of the system in Fab 4.

The Notes file checksheet is a troubleshooting guide template stored as a topic in a Notes file conference. When a troubleshooting situation arises, the user electronically extracts the corresponding checksheet, follows the written procedure, fills in the requested information and stores the completed checksheet as a Notes file reply. The checksheet serves two purposes: to guide a troubleshooting session and to collect data about the session. The user (typically a PIMT) initiates the troubleshooting activity. If the checksheet procedure does not resolve the problem, then a technical representative is

¹ VMS is a registered trademark of Digital Equipment Corporation

contacted to complete the process. The data collected with the checksheet then serves as background information about the problem for the technical representative.

Decision Trees

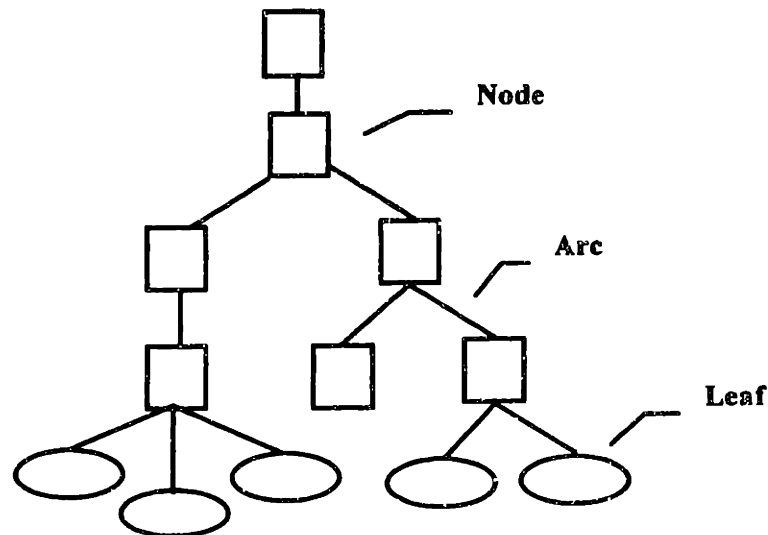


Figure 3-1: Decision Tree

A decision tree—a network of nodes connected by arcs—is a flowchart representing the problem solving procedure. A symbolic decision tree is shown in Figure 3-1. The path followed is determined by the answers to questions at each node. The different options cause the tree to *branch* out. The final node representing the last action to take is referred to as a *leaf*.

The decision tree approach to transferring troubleshooting knowledge was used in two ways. The first was as a graphic attachment to the troubleshooting section in the operating specification. The other usage was as a local *cheat sheet*. To act as a cheat sheet, the decision tree was printed out on cleanroom paper, laminated and posted near

the tool for quick reference. These cheat sheets were seen displayed in the fab for a number of tools.

Custom C+ Program

The most elaborate method of transferring troubleshooting knowledge in use in Fab 4 is C.A.T.S., a custom C+ program. C.A.T.S. stands for Computer Assisted Trouble Shooting. The user invokes the program from the VMS prompt and the program automatically extracts the relevant information about the problem from two databases. The program then uses this information to make a recommendation to the user about appropriate actions to resolve the problem.

A process engineer from the Etch PE group wrote the program to assist operators in troubleshooting particle failures during the metal etching process.

Evaluation of the Four Methods

Table 3-1 summarizes the rankings of the four methods for transferring troubleshooting knowledge against the five requirements developed in chapter two. The rankings are qualitative and based on the potential of the method rather than a particular application.

None of the four methods described satisfy all of the requirements with a score of three or more for each requirement. The C+ program scores the highest overall, but does not satisfy the *New Skills* requirement. The personnel in the technical group do not possess the software skills to write C+ code. The use of any software language to create troubleshooting guides, without additional resources to write the code, prevents the system from being developed and maintained by the technical groups. The high score for complexity of problems is based on potential since the actual application does not currently handle a complex problem.

The three other methods did not score as high overall as the C+ program because they only excelled on the *New Skills* requirement. The technical group personnel are familiar with creating Notes file checksheets, writing process specifications and generating decision tree flowcharts. However, all three of these methods lack the potential to *handle complex problems* because they are static files. The user must skip around the document depending on the facts of the problem. Long and complicated procedures become unwieldy to follow. The C+ program avoids this by incorporating expertise in the source code; transparently moving to the correct set of questions and instructions. The three methods also scored low on the compatibility requirement because they can not incorporate information from other databases or sources. They are however, all compatible with the computer systems used in Fab 4. The Notes file checksheet is the

Method	New Skills	Handle Complex Problems	Document Actions	Ease of Use	Compatible with Systems	Overall Score
Notes file Checksheet	●●●●●	●●	●●	●	●●	●●
Specification Section	●●●●●	●●	●	●●	●●	●●
Decision Trees	●●●●●	●●	●	●●●	●●	●●●
C+ Program	●	●●●●●	●●●●●	●●●●●	●●●●●	●●●●●

Poor ● Fair ●● Acceptable ●●● Good ●●●●

Table 3-1: Comparison of Knowledge Transfer Methods Used in Fab 4

only one of the three methods that meets the *Document Troubleshooting Activity* requirement. The decision tree method is the easiest for the PIMT's to use; especially when posted in the fab near the tool. The specification section is more complicated to use because the PIMT must search the appropriate specification with the on-line document control system. The Notes file checksheet is the least user friendly of the group since the PIMT must extract and edit a text file to complete the troubleshooting procedure.

Proposed Methods

Two other methods of transferring troubleshooting knowledge were under consideration as alternatives to the four previously described methods. The first alternative was to use Workstream, the shop floor control software, as the way to document troubleshooting activity. Process and product measurements are currently taken and stored by the PIMT's using Workstream. The second alternative was to use DECtree. DECtree is a proprietary software tool developed by Digital Equipment Corporation that allows users to graphically build an expert system using a decision tree format. The decision tree is then compiled into a C+ program. Virtually no computer programming skills are needed to build applications.

Using Workstream addresses the requirement to document the troubleshooting actions taken. Specific events and engineering data collection (EDC) parameters could be created in the Workstream database. The PIMT's would then log different events depending on the actions taken. However, the Workstream software does not allow a script (sequence of events) to be attached to an entity (tool). This limitation means that the Workstream method of transferring troubleshooting knowledge would require another method (e.g., a decision tree) to guide the PIMT through the troubleshooting procedure.

Using DECtree has all of the advantages of the C+ program method without the drawback of new skills being needed to create the C+ programs. The DECtree software will be described in more detail in chapter 4, but a brief overview is given here.

To create a troubleshooting guide with DECtree, the application creator selects from a palette of icons to graphically create the different decision tree elements. Icons exist for input and output nodes that at runtime are viewed by the user as questions and answers, respectively. Other nodes support mathematical calculations, queries to databases, and calls to other programs. The arcs connecting the nodes control the flow and cause branching to occur based on user inputs to the questions during run time. The final step in creating an application is to compile the tree into C+ code. This is a menu driven process that requires the user to indicate the starting point on the decision tree and provide a name for the executable file. The executable file can be started on any VMS system containing

the DECtree runtime libraries. During run time, the user interface prompts the user for answers to the questions needed to traverse the decision tree that embodies the expertise for a particular troubleshooting problem. Other runtime features are the ability enter user comments and to backup to previous questions. The questions, answers and user comments are all stored in a log file which enables actions taken during a troubleshooting session to be automatically documented.

Implications of Using DECtree

The features of DECtree described in the previous section support that it meets the requirements for transferring troubleshooting knowledge better than the other methods already used in Fab 4. However, DECtree is an expert system. Does the problem domain for troubleshooting routine production problems justify the use of an expert system? If so, is DECtree the best expert system to use?

This section will first show that the Fab 4 problem domain justifies the use of an expert system to transfer troubleshooting knowledge. Next, two types of expert systems—rule-based and case-based—will be described. The rule-based expert system will be shown to fit the requirements of TSGuide better than the case-based expert system. Finally, the advantages of using DECtree over other rule-based systems will be discussed.

Requirements for an Expert System Solution

Waterman bases the feasibility of an expert system solution on three criteria categories: possibility, justifiability, and appropriateness.⁶ The evaluation of Fab 4's problem domain using Waterman's categories is shown in Table 3-2. The Fab 4 troubleshooting problem domain meets these requirements.

Is the expert system possible?

All of the following must be true:

- 1) *Task does not require common sense:* Common sense is useful, but not the only source of knowledge necessary.
- 2) *Task requires only cognitive skills:* Troubleshooting is inherently a cognitive process.
- 3) *Experts can articulate their methods:* The current methods in use in Fab 4 are evidence that methods can be proceduralized.
- 4) *Genuine expertise exists:* The EE and PE groups have developed valuable expertise that is not easily replaced.
- 5) *Experts agree on solutions:* Not certain. Individual experts currently approach problems differently. Previous methods suggest that agreement can be reached on simple tasks.
- 6) *Task is not too difficult:* The problem domain has been limited to routine problems
- 7) *Task is not poorly understood:* Same as 6.

Is the expert system justified?

At least one must be true:

- 1) Task solution has high payoff: Difficult to quantify.
- 2) Human expertise is being lost: Turnover in the semiconductor industry is high. Experts being internally transferred to other tasks.
- 3) Human expertise is scarce: Being more so as resources are transferred. Certain shifts are not well staffed with experienced experts.
- 4) Expertise is needed in many locations: Yes. This is why resources are being transferred.
- 5) Expertise is needed in hostile environments: There is a time delay for an expert who is not already in the fab to change into the cleanroom clothes and enter the fab.

Is the expert system appropriate?

All of the following must be true:

- 1) *Task requires symbol manipulation:* Problem characteristics require a mixture of both symbols and algorithms.
- 2) *Task requires heuristic solutions:* Experts use heuristics to solve problems
- 3) *Task is not too easy:* This depends on the end users perspective. However, to increase job satisfaction will require that complex problems can be resolved.
- 4) *Task has practical value:* Absolutely. These problems must be resolved everyday.
- 5) *Task is of manageable size:* The problem domain has been limited to routine problems.

Table 3-2: Evaluation of Requirements for an Expert System in Fab 4

Comparison of Expert Systems Types

Expert systems are a form of artificial intelligence (AI) which allows human knowledge to be encapsulated and intelligently accessed.⁷ Two types of expert systems, rule-based and case-based, will be described and compared in this section.

The predominate expert system over the last decade has been rule-based systems.⁸ These systems are composed of three parts: a knowledge base, an inference engine, and a user interface. The knowledge base contains the rules elicited from the domain experts about the problem domain. These rules are in the form of If-Then statements. The inference engine searches through the knowledge base firing rules that match the users inputs and then displays the result to the user.

Case-based reasoning (CBR) is a relatively new development in expert systems.⁹ CBR expert systems are fundamentally different than rule based expert systems in the way that knowledge is stored and retrieved. The fundamental components of a CBR expert system are the case database, the index library, similarity or relevance metrics, and the explanation module.⁸ The case database is searched based on the users' inputs using the index library. Representative cases are then chosen based on how well the cases match the users' requests as measured by the relevance metric. The results of a search are cases that provide the best match to the current condition faced by the user. These cases are then used to solve the current problem by analogical reasoning. For example, the user applies the solution presented in the representative cases to the current problem after adjusting for differences.

Rule-based expert systems work well for problem domains where the knowledge is well understood and can be easily represented by If-Then rules. For example, procedural knowledge. The CBR expert systems handle problem domains that have nuances that are difficult to capture with If-Then rules. For example, experience-rich but knowledge-poor domains.⁸

There are two major issues associated with expert systems: knowledge elicitation and maintenance of that knowledge. Knowledge elicitation is the process of capturing knowledge from domain experts. Knowledge elicitation is the bottleneck in creation of most expert systems. For rule-based systems, the experts must translate what they do into If-Then type rules. This takes time and requires practice because experts do not think in terms of rules when solving problems. Acquiring knowledge for a CBR system only

requires that important features of already existing cases be identified. This is generally a much easier task for experts to accomplish.

Maintaining the knowledge in a rule-based system is more difficult than with a CBR system. Particularly, as the number of rules contained in the knowledge base become large. The difficulty is primarily associated with verification that the new rules do not conflict with existing rules. The CBR expert system can be maintained by simply adding new cases to the database as they become available. A task that many users can perform themselves.

Rule-based or CBR expert system?

The characteristics of the Fab 4 troubleshooting problem domain support the use of a rule-based system. First, these problems are well understood and highly procedural. Second, a history of cases does not readily exist. The Notes file conferences that the technical personnel use to share knowledge could be used as a starting point for developing cases, but much of the information needed to describe the problem in a case database is missing. Finally, the concerns with maintenance of rule-based systems are issues for large applications, not the small applications that will be needed to solve the knowledge transfer issues in the Fab 4 problem domain.

CBR expert systems work by presenting representative cases to the user. Adapting these cases to the current situation is a natural way to solve problems. However, if the user does not have adequate experience this method may not be very effective. Using a CBR expert system as a compliment to the Notes file conferences could be an effective method for knowledge sharing between the technical group members, but not transferring troubleshooting expertise to the PIMT's.

DECtree versus other rule-based expert systems

DECtree is a variant of the rule-based expert system. The If-Then rules are represented by decision trees. Each path from the top of the decision tree to a solution (leaf) can be decomposed into an If-Then rule.

Other commercially available rule-based expert systems were not evaluated against DECTree for this thesis for several reasons. First, finding such a product is not easy since the majority of commercially available products are not designed to run on the VMS operating system that is used in Fab 4. Second, the resources for installing another expert system were limited. DECTree is already installed on the DS computer network. Finally, the DECTree approach to representing knowledge as a decision tree appears to be novel. For the Fab 4 technical groups, this knowledge representation method is familiar to them from using the other method of knowledge transfer—decision trees. The advantage is in reducing the *new skills* requirement for creating troubleshooting guides using DECTree. Other advantages of using DECTree are presented in chapter 4 when the components of TSGuide are described in detail.

Chapter 4 Troubleshooting Guide System

A process to transfer troubleshooting responsibilities to the PIMT's was shown in Figure 2-1. The steps in that process are outlined below:

1. Acquire knowledge about how to resolve a troubleshooting problem.
2. Create a troubleshooting procedure from the acquired knowledge.
3. Use the troubleshooting procedure to solve that problem.
4. Monitor the results and improve the troubleshooting procedure.

This chapter will describe the components of TSGuide used to implement this procedure.

TSGuide consists of two software components: DECtree and a menu system. Steps one and two of the knowledge transferring process are accomplished by using DECtree. The output of that effort is an executable file that contains the troubleshooting expertise. The menu system is used to accomplish steps three and four by functioning as an interface between the user and the executable file.

DECtree

DECtree is used in TSGuide to create executable files which capture the troubleshooting procedures, referred to as troubleshooting guides, for a particular process or equipment problem. The troubleshooting procedure is first represented as a decision tree on paper by the technical personnel. The DECtree development environment is then used to graphically transform this decision tree into an executable C+ program. This is done using the development window shown in Figure 4-1. The icons on the left-hand side of the screen represent the seven different nodes used to create a DECtree decision tree. The arc icon, which looks like an arrow, is the connection tool that draws arcs between the nodes to determine the logic flow of the tree.

Node Functionality

Figure 4-2 shows the seven node types in decision tree format. This section gives a brief description of the functionality of each node type.¹⁰

- *Label Tree Node:* This node is used to identify a starting or continuation point of a decision tree. A *Call Tree Node* causes the program to branch to a *Label Tree Node* like a GOTO statement in the Basic programming language.
- *Input Node:* This node supplies the DECTree program with information during run time. The input source can be the end user, a database or a DECTree variable. Decisions are made based on the information supplied to the input nodes.

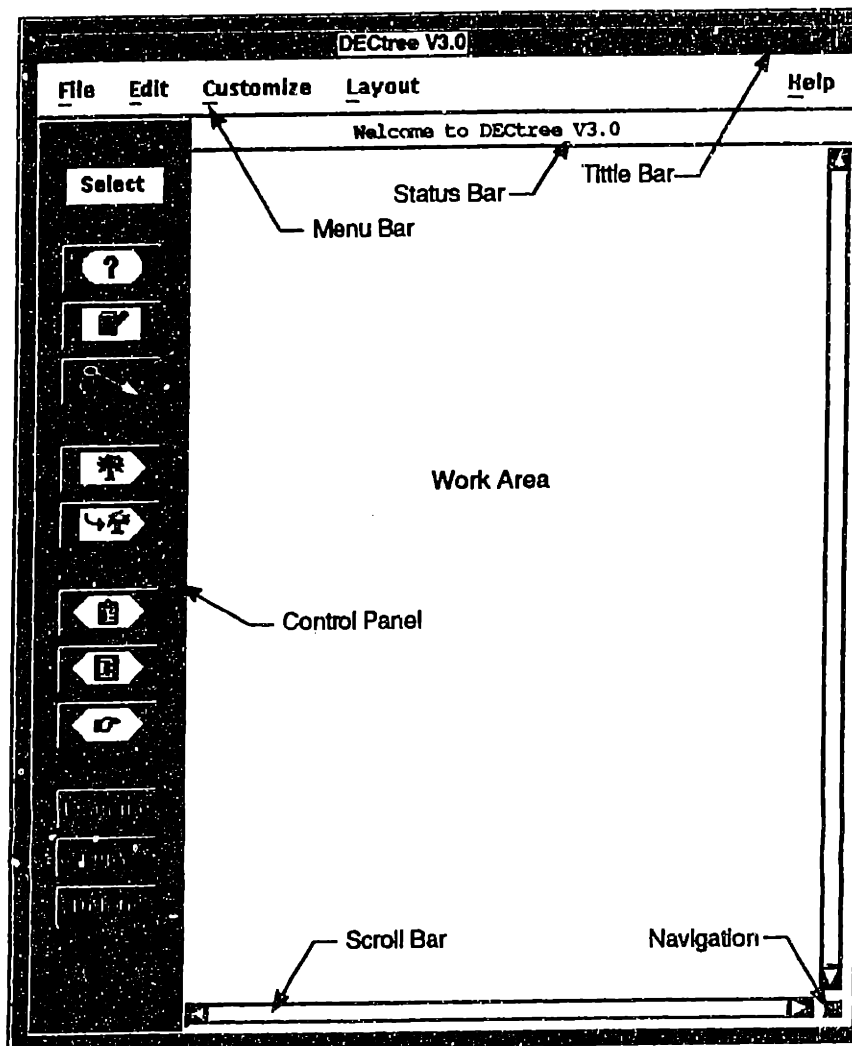


Figure 4-1: DECTree Development Window¹⁰

- *Output Node*: The output node displays information to the end user during run time. The output strings can contain DECtree variables allowing the developer to dynamically assign values to the output strings.
- *Call Tree Node*: This node causes the program to jump to a Label Tree Node of the same name. This label could be a subroutine tree or a different section of the same tree.
- *Assignment Node*: This node is used to perform calculations and define values of DECtree variables.

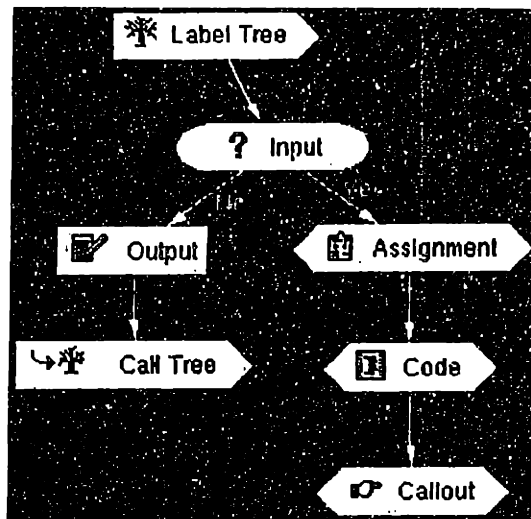


Figure 4-2: DECtree Decision Tree Nodes¹⁰

- *Code Node*: This node allows the developer to incorporate C+ code that will be compiled with the rest of the program. This allows additional functionality to be added to DECtree.
- *Callout Node*: This node allows an external program like a DCL command file to be accessed during run time.

The input, output and assignment nodes are used most frequently in creating a DECtree applications. The call tree node allows the developer to create subroutines trees for sections of the application which are used many times in one decision tree, other decision

trees or to break large trees into smaller modules. The callout node provides the developer with an easy way to interface the troubleshooting guides with other software programs. The code node allows advanced developers to create additional functionality if desired.

The attributes—prompt message, data type, variable name, etc.—of each node are

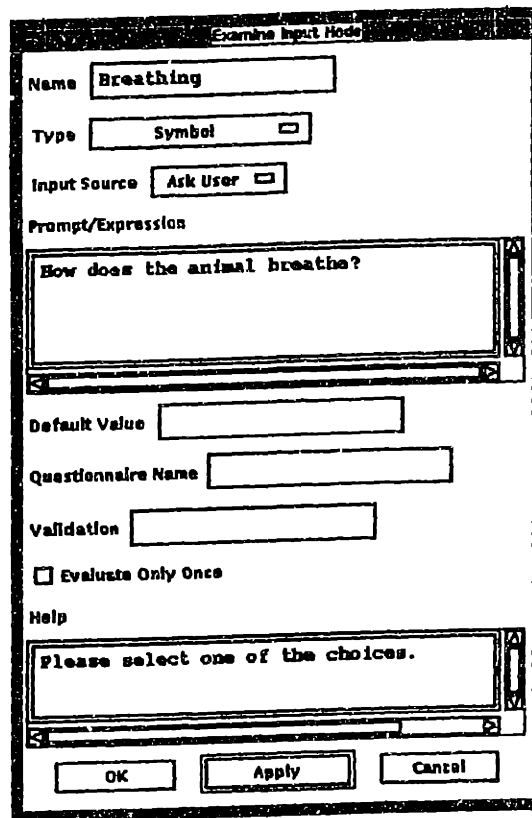


Figure 4-3: DECtree Examine Input Node Dialog Box¹⁰

configured using a dialog box. The dialog box is accessed by double-clicking the node with the right mouse button. An example of an input node dialog box is shown in Figure 4-3.

Generic Subroutines

Two generic subroutine trees were created to facilitate the DECtree software meeting the requirements discussed in chapter two. These subroutines reduce the new skills needed by the technical personnel to create troubleshooting guides that link to other information systems. These subroutines can be included in decision trees created by the technical group personnel and accessed by using the call tree node. The attributes of a call tree node include an argument list of parameters to be passed to the called subroutine. These parameters provide the subroutine with the values specific to the user's decision tree procedure.

The subroutine approach reduces the level of new skills needed by the technical group personnel because the values of the parameters passed to the subroutines are familiar to them. Making links to other information systems allows complex problems to be solved with troubleshooting guides without impacting the ease of use for the PIMT's.

NTCDat

The subroutine *NTCDat* is used to retrieve engineering data from the Real Time Relational (RTR) database table *NTCDat*. The DECtree representation of this decision tree is shown in Figure 4-4. *NTCDat* is the name of the RTR table that contains the Workstream entity engineering data. The RTR database contains all of the Workstream database records in RDB format. By retrieving this data directly from the RTR database, the troubleshooting guide can use data previously entered into Workstream to make decisions and calculations without requiring the PIMT's to reenter these values. This is an important capability since the PIMT's manually enter large amounts of data into Workstream; re-entering this data into a troubleshooting guide would be considered a disincentive to using TSGuide. Also, the PIMT needs to search the Workstream history files to obtain the requested data. The PIMT must use multiple VMS sessions to do this since Workstream and TSGuide are separate applications. This takes time and does not add value to the troubleshooting activity.

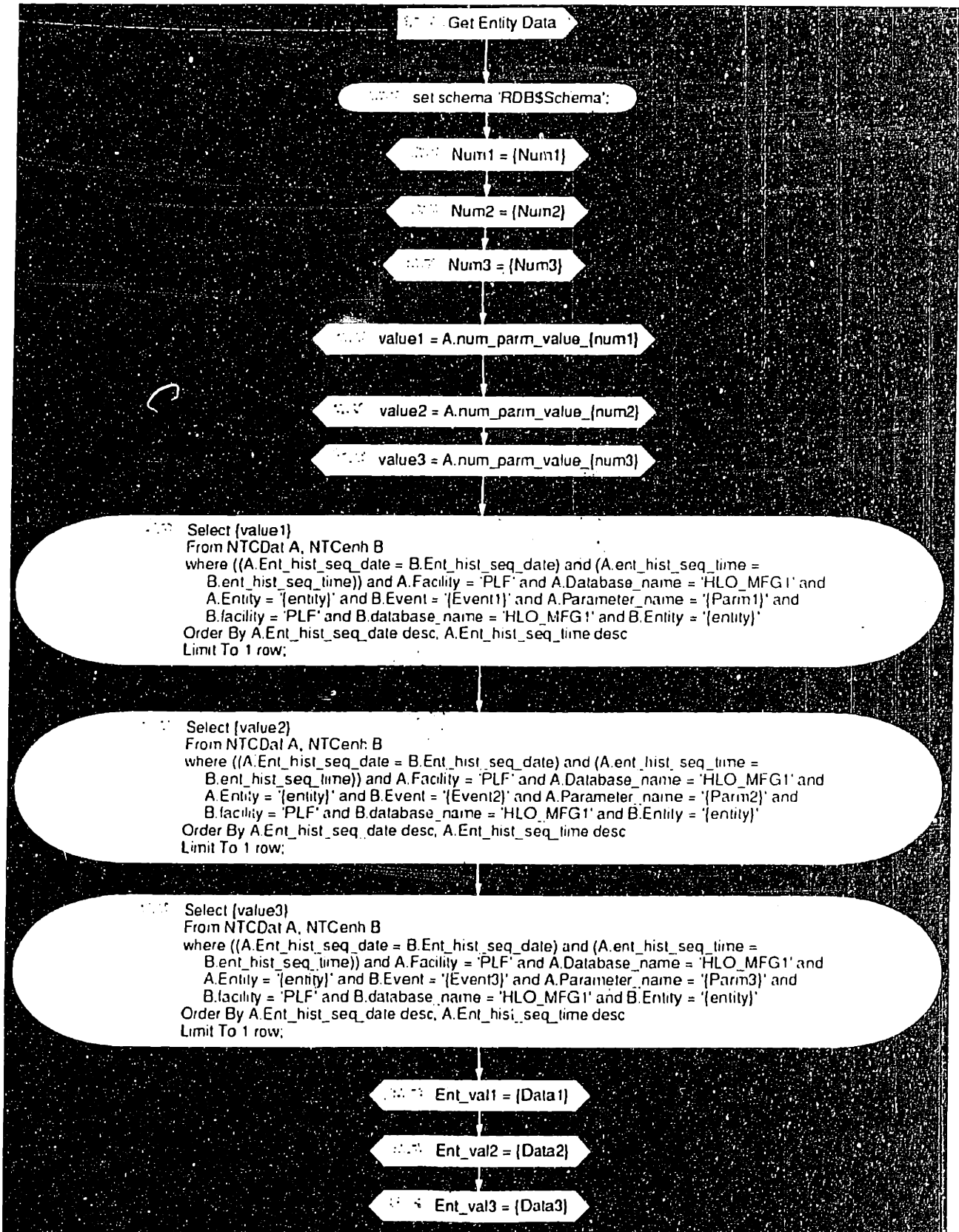


Figure 4-4: DECTree Subroutine NTCDat

The NTCDat subroutine uses the input node configured to get input from a database via structured query language (SQL) select commands. Constructing SQL select statements is not a skill generally possessed by the technical group personnel. However, to use the NTCDat subroutine in a DECtree troubleshooting guide, the developer only needs to know the names of the event, entity, and parameter value they wish to retrieve. These names are well known to technical group personnel since they originally defined them in the Workstream database.

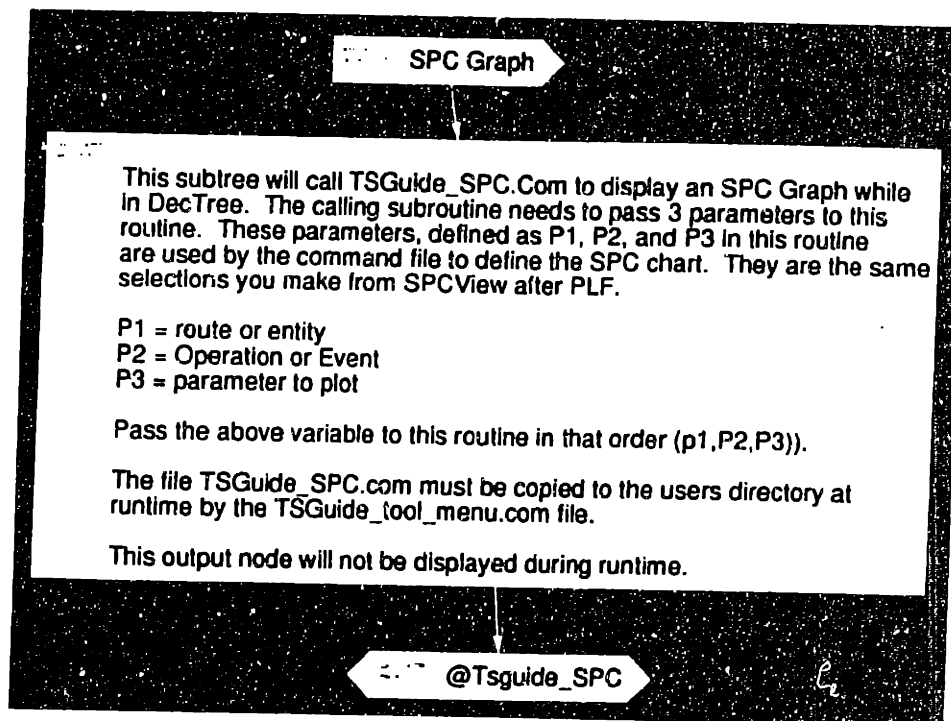


Figure 4-5: DECtree Subroutine SPCGraph

SPCGraph

The second subroutine is SPCGraph shown in Figure 4-5. This subroutine displays an SPC chart to the end user as part of the troubleshooting procedure. The SPC charts displayed by SPCGraph are managed by SPCView which is a separate software tool used by Fab 4 to create and display SPC charts. Displaying the SPC chart during the troubleshooting procedure allows the end user to answer qualitative questions regarding

trend patterns on that chart. Often, trends on charts from other entities in the tool group running the same process can provide useful diagnostic information.

The PIMT's are trained on how to use SPCView to display SPC charts. However, displaying a specific chart from the troubleshooting guide saves the operator from running SPCView and recalling a specific chart name.

The SPCGraph subroutine interfaces to the SPCView system through the callout node. The callout node executes a Digital Command Language (DCL) command file. DCL command files are analogous to batch files in the DOS operating system. The DCL command procedure issues the commands to display a graph, clear the screen, and return control to the troubleshooting guide. The developer of the troubleshooting guide needs to provide the SPCGraph subroutine with the disk location (path) and chart name of the SPC chart. Both the path and chart name are commonly known by technical group personnel since they must also supply this information to SPCView.

Run Time Features

The executable files created with the DECTree program use the DECTree run time library modules to provide additional capabilities during run time. These features include: backing up to previous troubleshooting guide steps, quitting the troubleshooting guide before completion, storing the troubleshooting actions in a log file, and allowing additional user comments to be entered into the log file.

TSGuide Menu

TSGuide is the interface between the end user and the troubleshooting guides created with DECTree. The TSGuide menu is divided into two levels of DCL command file programs—the TSGuide Main Menu and the TSGuide Tool Group Menu.

TSGuide Main Menu

The TSGuide main menu contains two levels of screens as shown in Figure 4-6. The first level lists each technology area in Fab 4. The second level contains a listing of the tool groups for each of the technology areas. The user's choice to the first level screen causes

the second level screen which corresponds to the choice to be displayed. The user's response to the second level screen causes the appropriate TSGuide Tool Group Menu to be started.

TSGuide Tool Group Menu

The TSGuide Tool Group Menu contains two levels of screens as shown in Figure 4-7. There is a different TSGuide Tool Group Menu for each tool group. The first level screen

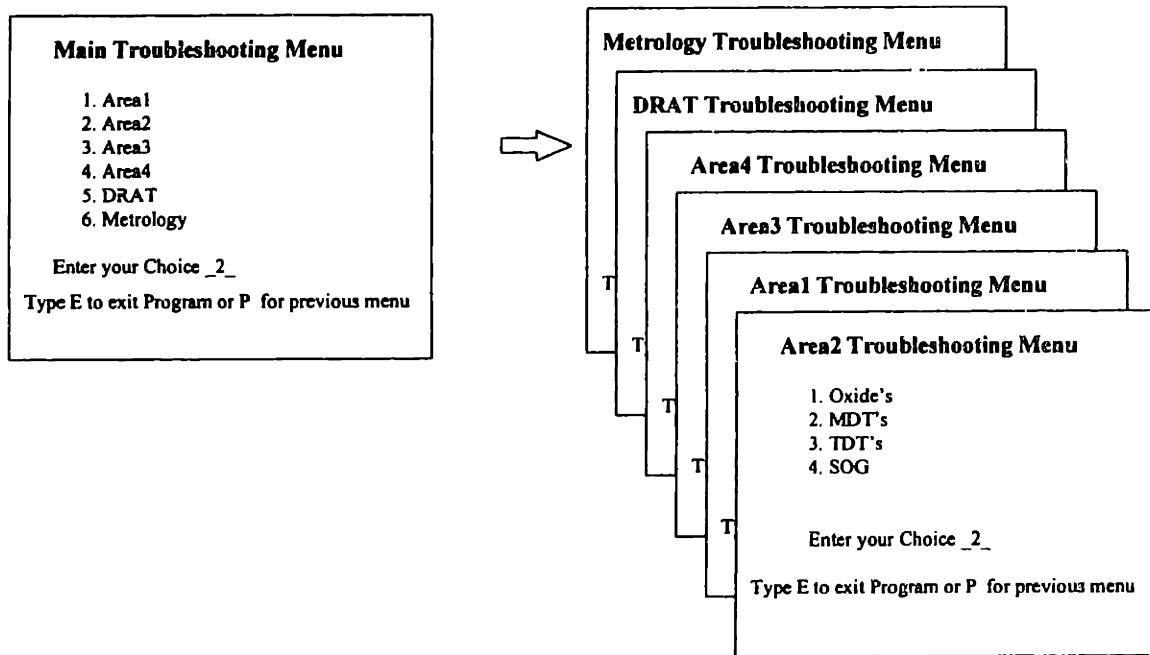


Figure 4-6: TSGuide Main Menu Screens

displays a list of entities for that tool group. The user's choice from this screen causes the corresponding second level menu to be displayed. The second level screen displays a list of troubleshooting guide descriptions for the entity chosen from the previous screen. The user's selection from the second level menu causes the executable file associated with that troubleshooting guide description to be run. The shaded screen in Figure 4-7 is the first screen of the DECtree troubleshooting guide associated with the second level menu choice number 2.

Management of Overhead

Prior to starting the troubleshooting guide executable file, the TSGuide Tool Group Menu manages the overhead associated with running DECtree executable files. First, the user's computer environment is modified to be compatible with the DECtree run time modules. Next, certain troubleshooting guides, like ones which use the SPCGraph subroutine, require additional files to be present in the users account. The TSGuide Tool Group Menu

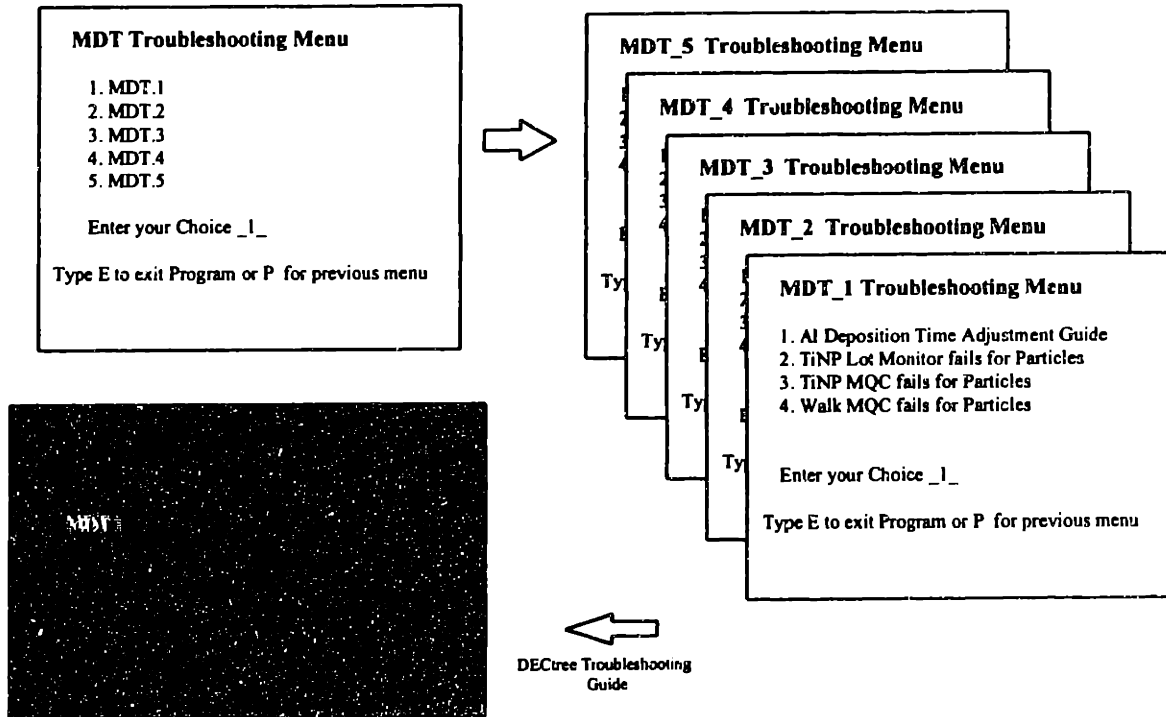


Figure 4-7: TSGuide MDT Tool Group Menu Screens

copies these files to the users' account. The troubleshooting guide executable file is then started.

After the troubleshooting guide is completed, the log file that automatically documents the actions taken during the troubleshooting session is posted to the correct Notesfile conference. This log file and any other files copied into the user's account are then deleted. The last TSGuide Tool Group Menu screen is then displayed to the user.

Meeting TSGuide Requirements

The TSGuide Menu increases the ease of use by the PIMT's, functions as a boundary control system, and minimizes the new skills needed by the technical group personnel.

Ease of Use by PIMT's

To use a troubleshooting guide the TSGuide menu is started by typing "TSGuide" from the VMS prompt. This starts the TSGuide Main Menu command file. The user selects options from the nested screens shown in Figure 4-6 and Figure 4-7 to get a list of troubleshooting guide descriptions that exist for the selections made on previous screens. The final screen selection starts the troubleshooting guide executable file associated with the description chosen.

The TSGuide menu makes using TSGuide for troubleshooting easier for the PIMT's. The TSGuide menu provides the user with a description of each troubleshooting guide and automatically starts the correct executable file. Without the TSGuide menu, the user would have to remember the file name, the location of that file, and the troubleshooting problem that file was used to resolve. Since the PIMT's are usually cross-trained to work on more than one tool group, the number of file names to remember could become large.

Boundary Control System

The TSGuide menu acts as a boundary control system for the PIMT's by displaying the problems they are authorized to perform troubleshooting on. The TSGuide menu could also be used to restrict access to certain troubleshooting guides based on a PIMT's certification level by checking the identity of the user and their skill level. This functionality does not currently exist.

The TSGuide Menu also functions as a boundary control system for the technical group personnel. To activate a troubleshooting guide, the technical group member must review the proposed procedure with their management and the production area manager where the troubleshooting guide will be used. Any issues around training, risks, and other concerns associated with the troubleshooting procedure must be resolved before activation. This approval process is currently done informally, but the TSGuide Menu

could formally link activation of a troubleshooting guide to a sign-off on an operating specification change.

New Skills Requirement

The TSGuide menus minimize the new skills required by the technical group personnel to distribute the DECTree troubleshooting guides. First, both the TSGuide Main Menu and Tool Group Menu programs are written in DCL command language. Few new skills are needed to modify the TSGuide Menu programs to add new troubleshooting guides since most of the technical group personnel are skilled in creating and modifying DCL command procedures.

The TSGuide Menu screens are divided between two levels of command procedures to facilitate modifications by the technical group members. The TSGuide Main Menu

```
$!
$! .....
$! * AREA2_MENU *
$! * *
$! * *
$! .....
$!
$ Area2_Menu:
$ Define Group$ "Node::Disk:[dectree.Area2]" |Area2 menu location
$ Menu := Area2_menu_choice
$ SCREEN ESC,[0H,ESC,[0]
$ SCREEN ESC,[2;6H Area2 Troubleshooting Menu"
$ SCREEN ESC,[4;16H 1. Oxide "
$ SCREEN ESC,[6;16H 2. MDT "
$ SCREEN ESC,[8;16H 3. TDT "
$ SCREEN ESC,[10;16H 4. SOG"
$!
$ Goto Selection
$!
$ Area2_menu_Choice:
$ IF (COM .eqs. "P" .or. COM .eqs. "p") Then Goto Main_menu | Menu to backup
$!
$ IF (COM .eqs. "1") Then @Group$:TSGuide_Oxide_menu
$!
$ IF (COM .eqs. "2") Then @Group$:TSGuide_MDT_menu
$!
$ IF (COM .eqs. "3") Then Goto Noexist | @Group$:Tsguide_TDT_menu.com
$!
$ IF (COM .eqs. "4") Then Goto Noexist | @Group$:Tsguide_SOG_menu.com
```

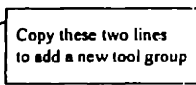


Figure 4-8: Area2 Section of code from TSGuide Main Menu program

program only needs to be modified when a new tool group is added to one of the technology areas. The TSGuide Tool Group Menu programs are created when a new tool

group is added and modified when new troubleshooting guides are added for that tool group.

Figure 4-8 shows the section of code in the TSGuide Main Menu program for the Area2 technology area. Each technology area has a section of code similar to this. The bolded parts are sections that are different from one technology area to the next. To enable the “SOG” tool group, the technical personnel need to modify the code in by removing the “*GOTO NoExist !*” command in the last line of Figure 4-8. The TSGuide Tool Group Menu program TSGuide_SOG_Menu.Com would be stored in location disk location Group\$ to complete the change. The TSGuide_SOG_Menu.Com program is created by modifying an existing tool group menu (e.g., TSGuide_MDT_Menu.com).

To add a tool group to the TSGuide main menu program that does not currently exist, the two lines indicated in Figure 4-8 need to be copied and edited to reflect the new name and menu number.

Changes to the TSGuide Tool Group Menu programs are done in a similar way. By splitting each tool group into a separate command file the changes can be done without disrupting the other tool groups.

A complete program listing for TSGuide_Main_Menu.Com and TSGuide_MDT_Menu.Com which shows the modular program structure can be found in Appendix B.

Chapter 5 TSGuide Case Studies

This chapter will present two case studies of troubleshooting guides created with TSGuide in the Area2 technology area. Also, the progress on transferring TSGuide to the other technology areas will be reported. These results will be analyzed in chapter 6 to answer the following two questions:

- Does the system meet the requirements identified in the earlier chapters
- What other issues occurred that were not anticipated

Case 1: MDT Aluminum Deposition Time Adjustment

The adjustment process used in this troubleshooting guide is based on the M.I.T. Leaders for Manufacturing thesis of Todd Barrett.¹¹ Barrett worked with the Area2, MDT group to develop the procedure. Please see his thesis for a detailed discussion of the troubleshooting procedure development.

Background

The metal deposition tool (MDT) is a semiconductor tool group used for metal film deposition in Fab 4. The MDT deposits metal atoms onto the surface of the silicon wafer by a sputtering process. Sputtering is a term used to describe the mechanism in which atoms are dislodged from the surface of a material by collision with high energy particles.¹² The high energy particles used are argon ions. Figure 5-1 shows a schematic view of a chamber used to sputter metal atoms from a target onto a silicon wafer. The argon ions are accelerated into the surface of the metal target and metal atoms are dislodged. Some of these metal atoms deposit onto the silicon wafer.

Over time, the metal target wears and the rate of deposition of the metal atoms onto the wafer decreases. The thickness of the metal film deposited is an important parameter to control for quality purposes. Therefore, the deposition time of the process is increased to maintain a constant film thickness while extending the life of the metal target.

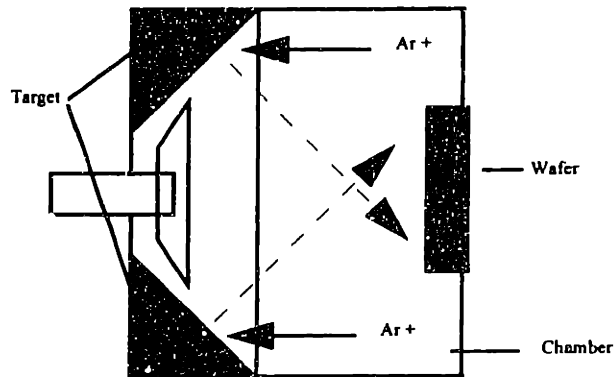


Figure 5-1: Schematic diagram of sputter process

Process Adjustment

The thickness of a metal film layer can be monitored indirectly through the relationship between film thickness and sheet electrical resistance (sheet rho). The sheet rho increases as the metal layer thickness decreases. When the sheet rho measurement exceeds a certain limit, the deposition time of that process recipe is increased. The PE representative makes the time adjustment based on experience and rules of thumb (i.e., add 1 second for every 2 unit increase in sheet rho). The heuristic is different for each metal layer, MDT machine and PE.

Troubleshooting Guide Procedure

The work described by Barrett formalizes the time adjustment process. For each MDT machine, a series of equations were derived empirically that relate the sheet rho of one metal layer to the sheet rho of the other metal layers. These derived sheet rho's are then used to calculate the new deposition time for each metal layer process recipe from equations relating the deposition time to metal layer film thickness. Figure 5-2 shows a flowchart for this adjustment procedure.

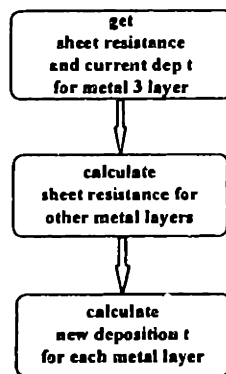


Figure 5-2: Deposition time adjustment procedure

Advantages of Proposed Deposition Time Adjustment Procedure

The proposed procedure has several advantages over the existing time adjustment procedure. First, since the adjustments for all of the metal layers are based on the machine quality check (MQC) of just one layer, the number of MQC's needed to adjust deposition times for all of the layers are reduced. Second, the consistency of the adjustments made across shifts would be greater. And third, the adjustment process could be done by individuals without expertise in making time adjustments.

Additional Expertise

The procedure in Figure 5-2 required three additional steps before it could be utilized as a troubleshooting guide. First, the MDT being adjusted must be known since the coefficients of the equations depend on which MDT is being adjusted. Second, the sheet rho measurement must be validated. Finally, the aggressiveness of the adjustment must be chosen. The resulting decision tree is shown in Figure 5-3.

Equation Coefficients

A different set of coefficients were regressed for each of the MDT's. By knowing which MDT is being adjusted, the values of these coefficients can be assigned as variables. Alternatively, a different troubleshooting guide could have been created for each MDT

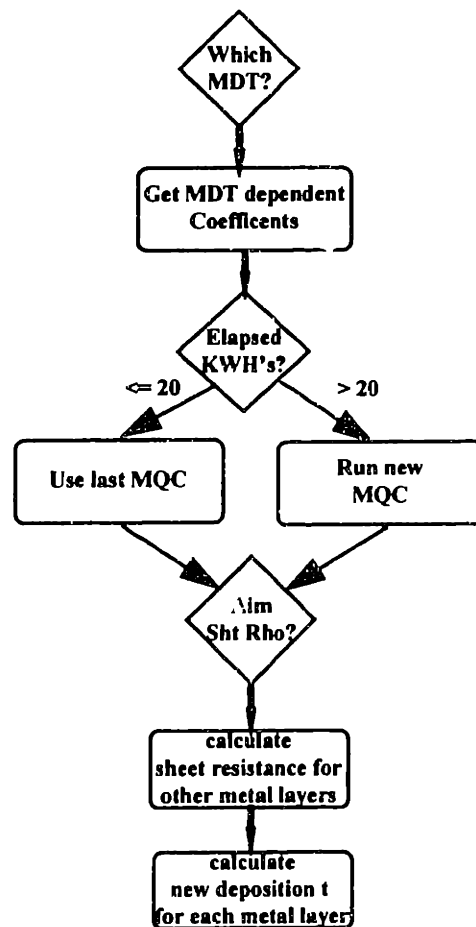


Figure 5-3: MDT Deposition Time Adjustment Decision Tree

and the coefficients entered as constants into the equations. However, using one troubleshooting guide with variable coefficients in the equations reduces the amount of work needed to create and maintain common elements between the decision trees. The end user is only asked one additional question, "Which MDT?", when running the troubleshooting guide to accomplish this combining of troubleshooting guides.

Validating MQC Measurements

The sheet rho measurement used to predict the other metal layer sheet rho's is obtained from an MQC measurement which is performed once every 24 hours. However, a process adjustment could be triggered by a sheet rho measurement on a lot monitor that is out of the specification range made between MQC measurements. A lot monitor is a wafer which receives the same processing as the lot wafers, but is used to make quality measurements. It is possible that the condition of the target when the last MQC was taken had changed significantly if many production lots had been run since the last MQC measurement was made. An adjustment based on these measurements would not be aggressive enough.

To insure that the last MQC measurement is still valid, the elapsed kilowatt hours on the target is checked. A new MQC measurement is requested if more than 20 kilowatt hours have elapsed since the last MQC was run. The threshold value of 20 kilowatt hours is arbitrary and was chosen based on the tool group owners' experience.

Aggressiveness of Time Adjustments

The target sheet rho corresponds to a target metal layer thickness. The specification for metal layer thickness has a mean and tolerance above and below this average. The process drift with time is always in one direction because of target wear. The frequency of time adjustments can be reduced by adjusting the deposition time so that the sheet rho starts out below the mean (the metal lines will be thicker than the specification). The PE has an option of how aggressive to be when re-centering the process. This aggressiveness can be controlled by changing the aim of the adjustment equations.

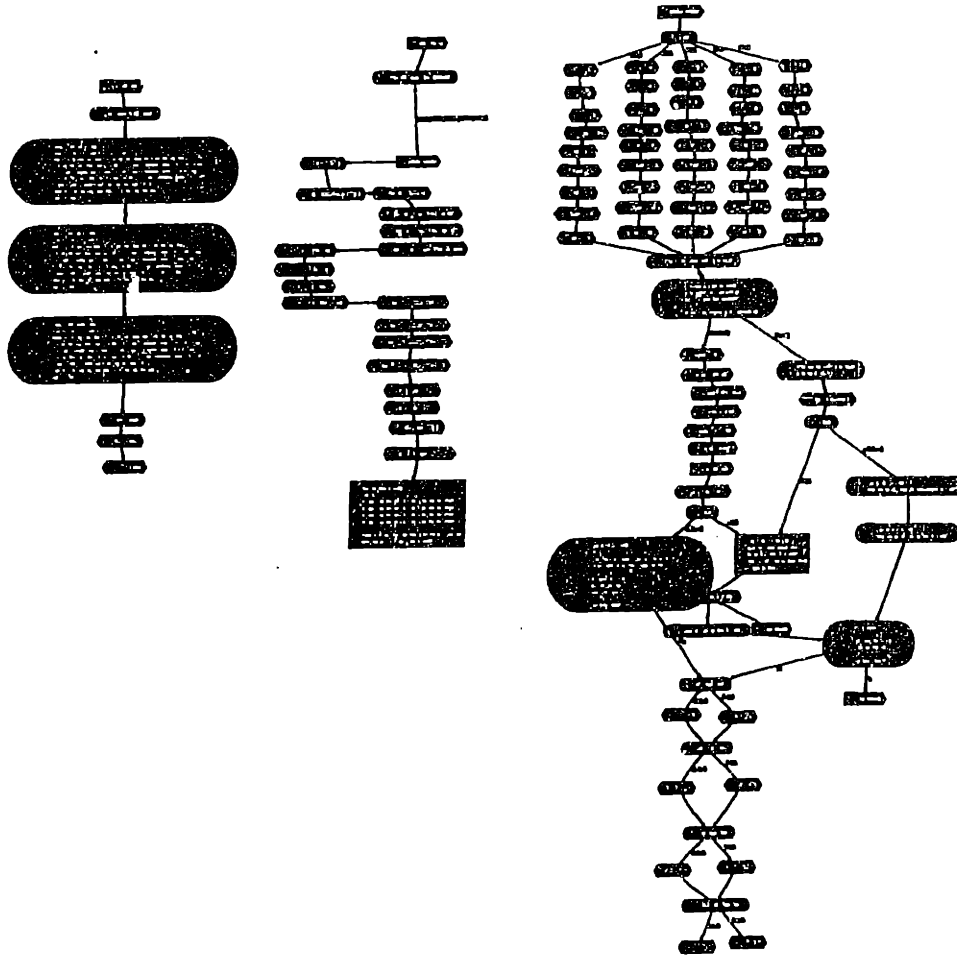


Figure 5-4: Structure of the MDT Aluminum Deposition Time Adjustment DECtree decision tree. (see Appendix C for a legible enlargement)

DECtree

Figure 5-4 shows the structure of the DECtree decision tree for the Aluminum Deposition Time Adjustment (Adj_T) troubleshooting guide. The tree is split into three separate trees to improve the maintainability of the troubleshooting guide. The three individual DECtree decision trees—MDT Time Adj, Initialize Variables, and Get Entity Data—are included in Appendix C in legible form. The values of certain parameters have been replaced by “C” for proprietary reasons.

MDT Time Adj Tree

The “MDT Time Adj” tree is the main tree in the troubleshooting guide. The user is asked which MDT needs adjustment and then the “Initialize Variables” tree is called to get the values of the equation coefficient variables, MQC sheet rho results and adjustment zone aim. Control is then returned to the main tree to perform the calculations of Figure 5-2. The last step is displaying these results to the end user.

Initialize Variables Tree

The “Initialize Variables” tree first defines the equation coefficients based on the MDT choice made in the main tree. Next the MQC sheet rho, current deposition time and kilowatt hours are retrieved from the RTR database using the “Get Entity Data” tree. The user is given the option of manually entering the data instead of automatically retrieving it so that the troubleshooting guide can be used if the RTR database is unavailable. The validity of the last sheet rho measurement is checked and the user is notified of the result. A new MQC is requested if the last MQC is no longer valid. The last section of the “Initialize Variables” tree is to determine the aggressiveness of the time change. The user is asked if a change from the default setting is desired. If not, the tree returns control to the main tree. Otherwise, the user is prompted for the new target zones.

Get Entity Data Tree

The “Get Entity Data” tree is the NTCDat subroutine tree described in chapter 4. The three large nodes each extract a value from the RTR database using interactive SQL select statements. DECTree limits each select statement to retrieving one column and one row (one value). The conditions of the select statement are formulated using variables passed to the subroutine from the calling tree.

Implementation

The aluminum deposition time adjustment troubleshooting guide was initially used by the Area2 PE group. A brief e-mail note was sent to the PE group members explaining how to use TSGuide and that the troubleshooting guide was active. Most of the PE's were familiar with the adjustment procedure from T. Barrett's work. After a month of confidence building, the troubleshooting guide was approved for the PIMT's to use. The

PIMT's who could use the tree were limited to those who were trained on making recipe changes with the MDT software. The PE's supporting the various production shifts trained the PIMT's on the use of TSGuide.

The implementation process was very successful. Minor modifications to the DECtree were made during this time based on user feedback and additional functionality becoming available. For example, the NTCDat subroutine was added to automatically extract the MQC data. The option of by passing the auto-extraction was added when the RTR database was found to be unavailable on a few occasions. The by-pass option was added to the SPC target zone changes to shorten the resulting log files.

Results

The Adj_T troubleshooting guide was been used over ninety times during a four month period. Toward the end of that period, essentially all of the adjustments were made using TSGuide. Informal feedback supported that the ease of use and increased satisfaction from greater control over their jobs was adequate to overcome the extra work of making the adjustments. For an adjustment not requiring a new MQC be run, the total troubleshooting time is less than 15 minutes. This increases to 30 minutes if the old MQC is invalid.

For this troubleshooting guide, the validation of the previous MQC serves as a boundary control system. This can easily be defeated since the elapsed kilowatt hours calculation depends on information provided by the user. Also, if the MQC is found to be invalid there is no interlock preventing the user from re-entering the last MQC results instead of running a new MQC. However, the log files posted in the Notes conference provide a record of the actions taken and serves as an interactive control system. Additional training about the importance of this validation test could result if this behavior is observed.

After several months of operation, the diagnostic control system, the SPC chart of sheet rho, indicated that the adjustments recommended by the troubleshooting guide were not aggressive enough. This caused more frequent time adjustments to be made than usual. The suspected cause is a shift in a parameter that the equations used to calculate the time adjustments do not capture. Use of this troubleshooting guide has been temporarily halted until the cause of the drift can be identified and corrected or included in the equations.

Case 2: OXIDE Lot Monitor Particle Failure

The OXIDE Lot Monitor Particle Failure (Ox_P) troubleshooting guide is based on the decision tree flowchart developed by an Area2 sustaining process engineer. She elicited the knowledge contained in this tree from EE and PE representatives of the Area2 group.

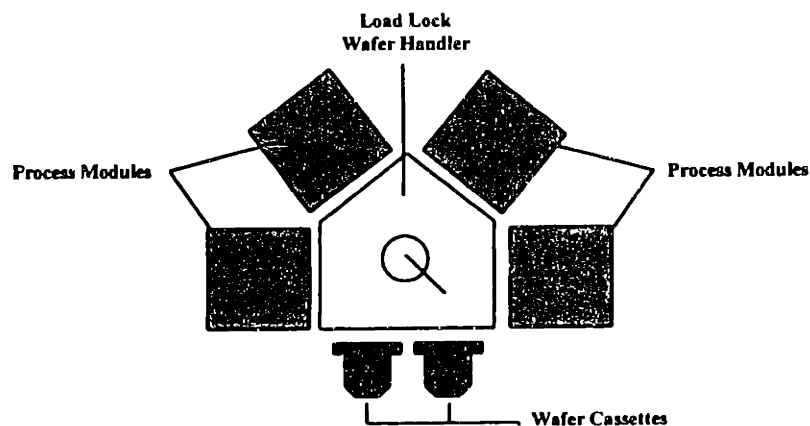


Figure 5-5: Schematic of OXIDE Cluster Tool

Background

The OXIDE is a cluster tool used in Fab 4 for depositing the dielectric insulating film layers between adjacent metal line layers. A cluster tool derives its name from “clustering” more than one processing module within a tool. These modules are typically connected to a central wafer handling robot as shown in Figure 5-5. For example, the four

chambers of the OXIDE can be configured to perform different process operations such as chemical vapor deposition (CVD) or plasma etching.

The measurement for particle contamination is made by scanning the surface of the wafer before and after processing. Surface defects are counted by size category (e.g., large, medium, and small). Limits are placed on the total number of particles added (the difference between before and after processing) for each particle size category. A reading in excess of these limits triggers troubleshooting to identify the cause.

The nature of cluster tools makes identifying and resolving the source of the particles complicated. A product wafer first enters the load lock and then might enter any or all four of the process chambers depending on the recipe sequence and equipment configuration. Particle contamination could occur from any of these sources including the wafer cassettes. Additionally, the particles could be caused by the process or from mechanical sources such as a wafer handler scratching the wafer.

Troubleshooting Procedure

The procedure to troubleshooting a lot monitor particle failure is shown in Figure 5-6. The first step is to re-test the system. Often a re-test produces a passing result. If the re-test fails, particle checks are run to isolate whether the source is mechanical or process related. These particle checks involve placing a different wafer into each chamber and removing it without starting the process recipe. This allows the source of the particles to be isolated from between a process or mechanical cause. If the cause appears to be mechanical, then the problematic chamber is cleaned with a vent and purge recipe and re-tested. A failure at this point indicates that the cause is more serious and will require EE involvement to open up the process chamber and perform a wet clean or make repairs. If the re-test passes than the process which initially failed is re-tested. If this re-test passes, the system is taken through a procedure to bring the system up. This involves checking to see if other lots have failed recently on this system. If they have, additional measurements are made on the next lot to be processed. If the re-test fails then the troubleshooting procedure continues as though the original failure was caused by the process instead of

the mechanical particles. This assumes that the mechanical particle problem has been resolved, but that a process issue remains.

To isolate a process problem, a series of MQC's are run which isolate the various sequences of the multi-step process recipes. A failure requires that the failed MQC be repeated with a gas only recipe. This isolates the cause of the particles from between a

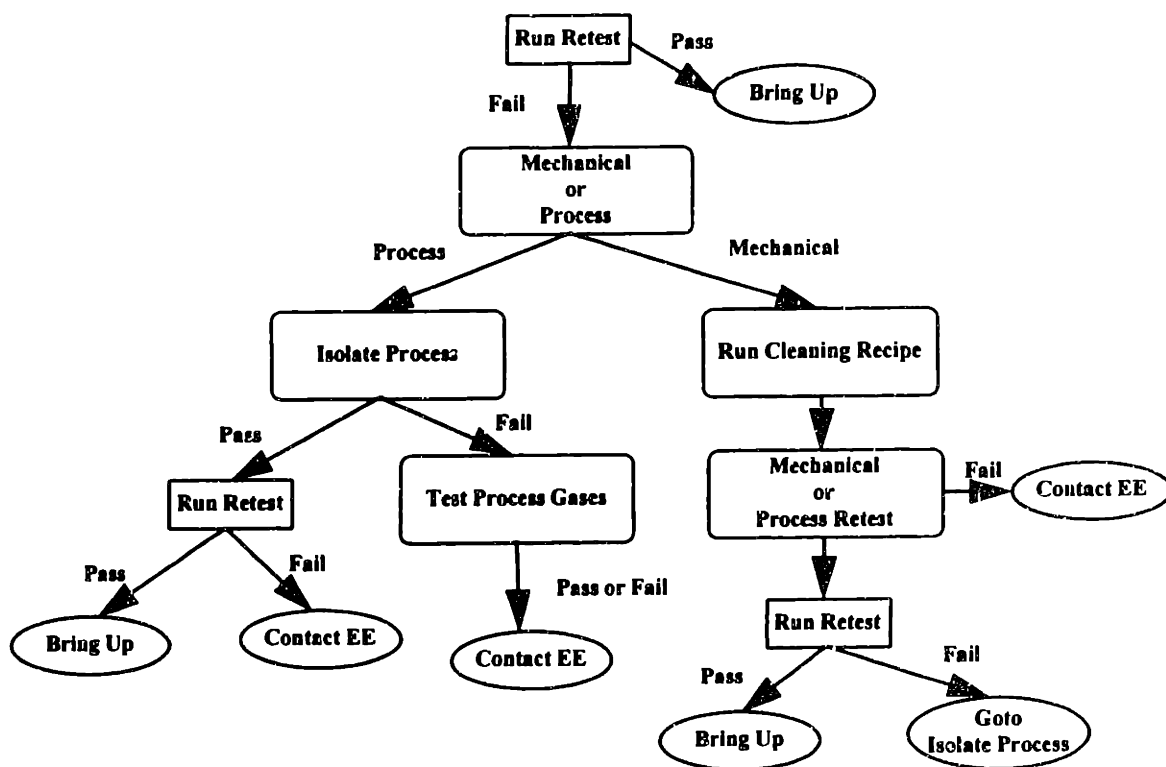


Figure 5-6: Simplified Decision Tree for OXIDE Lot Monitor Particle Failure

process interaction (plasma and reactants) and reactants only. The procedure is to contact EE regardless of whether the gas only test passes or fails since both conditions require EE intervention to resolve.

If the process MQC's run to isolate the process sequences passes, then a re-test of the originally failing process is run. If this re-test passes, the bring system up procedure is followed. If the re-test fails, the system is turned over to EE for additional diagnosis.

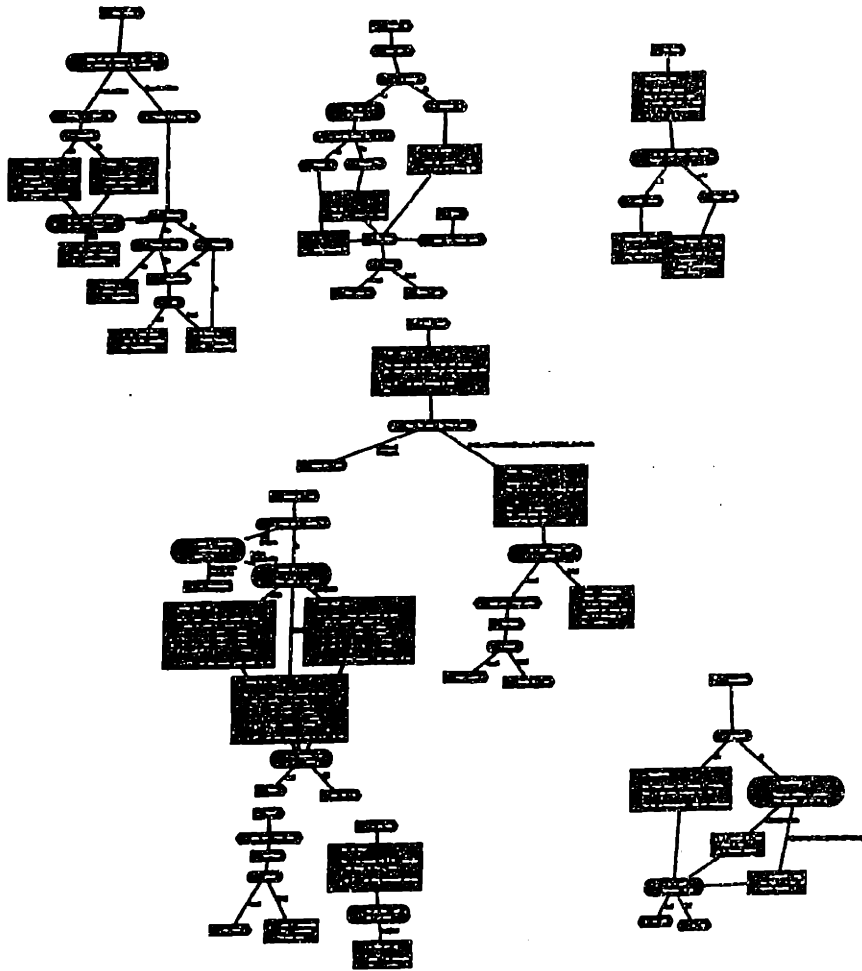


Figure 5-7: Structure of OXIDE Lot Monitor Particle Failure DECtree decision tree. (see Appendix D for a legible enlargement)

DECtree

The decision tree in Figure 5-6 was translated into the DECtree decision tree shown in Figure 5-7. Legible copies of each subtree are included in Appendix D. The procedure

was broken down into several subtrees to enable certain procedures to be called several times. For example, the subtree “Bring System Up” is called each time an action results in a re-test passing. Other subtrees were created to reduce the size of the main tree to facilitate printing and editing of the tree.

Like the Adj_T troubleshooting guide, a single Ox_P troubleshooting guide is used for all of the OXIDE tools. The particle troubleshooting procedure is similar for each OXIDE tool. However, the specific procedure varies depending on the particular tool and process step combination that fails. The troubleshooting guide determines the correct procedure based on the user inputs of which tool and recipe failed. This information is also contained in the operating specification, but is more readily available if accessed from the troubleshooting guide.

Implementation

The implementation of this troubleshooting guide met several delays. The first delay was a result of being part of the learning process for TSGuide. The procedure was considered a good candidate for testing out TSGuide because the original procedure already existed in a flowchart (decision tree) prior to the start of the internship. However, as the requirements for TSGuide evolved, changes were made to the procedure. For example, originally there were a number of actions that required the end user to log events (actions) to the Workstream database. This troubleshooting guide was originally intended to be implemented using the Workstream method discussed in chapter 3 which would require the events to be logged into Workstream. These events were eliminated to make the troubleshooting guide easier to use and because the DECTree log file made it unnecessary for documentation purposes.

The next delay in implementing the troubleshooting guide occurred during the approval process. Several concerns were raised. The first involved certain steps in the procedure. For example, there was some disagreement about whether the “Individual Sequence Test” subtree added any value to the diagnosis. As a result, it was eliminated from the DECTree shown in Figure 5-7.

Second, the procedure did not represent the actions that all of the technical personnel would follow if they were to perform the troubleshooting. For example, information regarding the pattern of the particles on the wafer might suggest a cause that eliminated the need to run certain diagnostic tests in the Ox_P troubleshooting guide. The OXIDE particle issue falls between the EE and PE boundaries as a process or equipment problem; representatives from both groups have expertise in resolving such issues. This creates multiple experts with different perspectives and adds to the complexity of knowledge elicitation. Some of these issues were included in a revision of Ox_P made after the internship was completed and are not shown in Figure 5-7. However, it was also agreed upon, that the technical groups did not have to use the troubleshooting guide when they were required to troubleshoot OXIDE lot monitor particle problems. This would allow them to follow a procedure they felt best fit the situation. This compromise allowed the procedure to be released to production while additional expertise was incorporated into future revisions.

The last concern was around the length of time it would take for the troubleshooting proposed in the DECTree to be completed. The time to complete the troubleshooting is estimated to be from 15 minutes to several hours depending on which path is followed through the tree. The total downtime of the OXIDE during the troubleshooting session could be even longer because the PIMT's would prioritize around the production schedule—starting and stopping to process product lots on OXIDE's that were operational. The issue was that the technical groups are responsible for the availability metric which this extended troubleshooting session negatively impacts. The availability metric measures the percentage of time that the tool is available to production to produce product. This concern was addressed by adding a new down code to the resource tracking system to indicate when the PIMT's were performing the troubleshooting. The technical group would be able to identify excessive downtime due to troubleshooting by the PIMT's.

Results

There has been one documented usage of the OXIDE troubleshooting guide over the two months since it was released to production. The reason for this is not clear. The Notes file indicates only six documented particle issues during this time. At one time, lot monitor particle failures were a major issue. However, a major cause of particulate contamination has since been identified and eliminated.

One possibility why the usage has been so low is that the first action in the troubleshooting guide, run a re-test, resolves the majority of the failures. Running a re-test is part of the operating specification so the PIMT's do this automatically without the need to run the troubleshooting guide. Therefore, these particle failures are not tracked by the Notes file conference where the log files are sent.

Transferring TSGuide Throughout Fab 4

The objective of transferring TSGuide throughout Fab 4 was included as part of a larger effort to increase the productivity of the fab.

PEP Team

The process empowerment program (PEP) team was formed to implement continuous process flow in Fab 4 to allow an increase in production capacity, lower production costs and increase the time PIMT's have to perform troubleshooting. The MQC and lot monitor quality measurements were examined to eliminate non-value added testing. The remaining tests were to be classified as critical or non-critical. Critical parameters were identified as being important to product quality. For example, the parameter was included in the technical file specification. The specification limits of critical parameter charts were to be separated from the SPC control limits. A system called Annotation was to be installed and integrated with Workstream to drive real time, closed loop SPC. This system would track all violations of limits on critical charts. Documentation of the cause of the violation would be required to close an annotation. TSGuide would be used to help PIMT's perform troubleshooting to close annotations. Creating and implementing

troubleshooting guides was planned as the last phase of the PEP process since removal of non-value added steps and identification of critical SPC parameters needed to occur first.

Training

Three training sessions were held to facilitate transferring TSGuide to the other technical areas of Fab 4. The objectives of the training sessions were to get an overview of TSGuide, to learn the basics of using DECTree, and to understand how to use the generic DECTree subroutines, NTCDat and SPCGraph. The first session was given to representatives from the PEP team. The last two sessions were open to all employees of Fab 4, but targeted the 5x8, and A and B shift technology areas. One PIMT attended the training. In total, 28 people received the four hour training.

Progress

Each major tool group in Fab 4 has the potential of 2 to 3 troubleshooting guides. There are over 25 major tool groups in Fab 4. Troubleshooting guides have been completed for three tool groups across two technology areas—Area1 and Area2. As the initial PEP team work is completed, additional troubleshooting guides are expected to be created.

Two people who attended the training session created these troubleshooting guides. One of whom had previously developed the OXIDE lot monitor particle failure troubleshooting guide. The other person is a PE in the Area1 technical group. She has identified a total of nine troubleshooting guides for her tool group. Three of these guides have been completed, but the approval process has not been initiated at this time. These trees use the generic subroutine SPCGraph to display SPC charts that the user is asked questions about. She also developed an additional generic subroutine to allow Notes file replies to be displayed to the user from within a DECTree troubleshooting procedure.

Chapter 6 Conclusions and Future Recommendations

Chapter five examined the use of TSGuide to transfer troubleshooting expertise from a technical support group to the production technicians they support. This chapter will discuss what can be learned from the results of chapter five. Also, recommendations for continuing the progress of TSGuide will be made.

Conclusions

It was hypothesized that the success of TSGuide involved addressing both the technical and the non-technical issues of knowledge transfer. The main conclusion is that TSGuide does address both the technical and organizational issues of knowledge transfer.

However, certain characteristics of troubleshooting problems have an effect on the implementation and use. Also, the issues of knowledge acquisition and time constraints of technical personnel are not adequately handled by TSGuide.

Meeting Requirements

There was evidence that TSGuide demonstrated the ability to handle complex problems and function as a boundary and interactive control system. However, with only two observations caution is needed in generalizing from these results.

The Fab 4 incentive system has not fundamentally changed to support TSGuide. Increase in job satisfaction was not measured, but user feedback and usage rates suggest that the T_Adj troubleshooting guide was easy to use. Short duration troubleshooting guides which resolve the problem are more likely to be used. DECTree decision trees that utilize algorithms or high levels of expertise meet this criteria.

Whether the system provided physiological safety and facilitated organizational change as a result of meeting these requirements is inconclusive. A rapid transformation of TSGuide applications that allowed the PIMT's to manage troubleshooting with fewer technical resources would be supporting evidence. The four months that TSGuide has

been in operation has not yielded that result. The use of TSGuide in Fab 4 is well below the potential number of applications. Perhaps this is because of the time line of the PEP team. The implementation of troubleshooting guides across the fab is scheduled to follow completion of reducing the non-value added work in the fab. This has not, yet, been completed.

Comparison of Case Study Results

The two cases described in chapter 5 experienced different levels of usage and difficulty in implementation. The MDT aluminum deposition time adjustment troubleshooting guide has been utilized more than the OXIDE lot monitor particle tree and was also easier to implement. The different frequencies of opportunity for using these two troubleshooting guides is more than adequate to explain the different usage rates. However, the two troubleshooting guides solve very different problems and these differences have an important impact on troubleshooting guide effectiveness and ease of implementation. From comparing these two cases, four problem characteristics were found to be important. These are: level of knowledge, probability of problem resolution, length of troubleshooting session, and multiple experts.

Level of Knowledge

The aluminum deposition time adjustment problem (T_Adj) is resolved using a series of equations while the OXIDE lot monitor particle failure (P_Ox) is solved by following a logical procedure. The level of knowledge about how to resolve T_Adj, the deposition problem, is greater than P_Ox, the OXIDE particle problem.

The more knowledge there is about a problem, the greater the expertise available to solve it. For example, in the future P_Ox might identify a solution from inputs about the failure scenario, like recipe and amount and distribution of particles, instead of following a logical procedure which grinds out a solution. This level of knowledge about diagnosing a particle problem does not currently exist. However, the most recent version of the P_Ox troubleshooting guide (not shown here) asks questions that allow certain diagnostic tests to be eliminated.

Probability of Resolution

For the two troubleshooting guides, the probability of the PIMT's resolving the problem is far greater for T_Adj than with P_Ox. The number of possible outcomes for T_Adj is one (Adjust Times) as opposed to two for P_Ox (Bring Up and Contact EE). The PIMT's only have a 64% chance of resolving the problem without contacting a technical group representative if all of the branches in the P_Ox tree procedure shown in Figure 5-6 have an equal probability of occurring.

The level of job satisfaction is greater when problems are resolved than when resolution must be passed on to another group. Troubleshooting guides can be improved if barriers to problem resolution are addressed by increasing training or developing better troubleshooting guide procedures.

Length of Troubleshooting Session

The time commitment for these two troubleshooting guides is very different. The longer it takes to perform a troubleshooting session might impact the level of job satisfaction the user derives from performing troubleshooting. For example, if a session continues over to the next shift, the user who initiated the troubleshooting procedure might not receive any satisfaction from their efforts.

The length of troubleshooting might be dictated by the diagnosis procedure. However, greater expertise can reduce the time needed to identify the problem. Also, structuring the tree to test conditions which are likely and easy to do will reduce the average troubleshooting time. Troubleshooting guides which require excessively long times might not be good candidates for TSGuide problems until better procedures can be developed.

Multiple Experts

The particle tree had a greater number of experts involved in developing it since the EE group is also involved in troubleshooting OXIDE particle problems. The time adjustment problem is primarily in the domain of the PE's. When multiple experts exist for solving problems, the knowledge acquisition task is more complex, but potentially more complete.¹³ The time to capture the expertise and get agreement among the multiple

experts takes longer. However, the benefit of getting the input from multiple experts is a decision tree which utilizes more expertise.

Issues not Addressed by TSGuide

TSGuide failed to address two issues that are important to the success of the troubleshooting system: difficulty of knowledge acquisition and time constraints of the technical personnel to create troubleshooting guides.

Troubleshooting guides can not be created without first obtaining the domain knowledge. It is well documented that knowledge acquisition is the bottleneck in the development of expert systems.^{14,15} The decision tree format of DECtree was considered an advantage for this reason since the technical personnel were already using decision trees for simple problems. However, while the decision trees were beneficial in verifying and coding the troubleshooting knowledge, using this format did not eliminate the task of eliciting true expertise. This is a skill that will improve with practice.

The constraint of minimal support resources dictated that TSGuide be managed by the technical personnel. However, the amount of time needed to create troubleshooting guides and the amount of time that the technical personnel have available to create troubleshooting guides were not addressed by TSGuide. The system was developed to minimize the new skills needed by the technical personnel to enable them to manage the system without support resources, but a finite amount of time is still needed to produce the troubleshooting guides. The advantages of having a self-contained system are lost if the group does not have the time to manage it.

Recommendations

The two cases studied show that TSGuide has the potential to transfer troubleshooting knowledge to the PIMT's. By continuing to implement troubleshooting guides in Fab 4, additional learning around using troubleshooting guides to transfer knowledge will be gained. This experience will be reusable after Fab 4 has been closed since the human resources will be transferred to the new fab. Based on conclusions from the previous

section, future troubleshooting guides should focus on the high frequency, well understood problems. A continuous improvement mentality should be used to facilitate rapid introduction of troubleshooting guides that will increase in expertise over time. Also, TSGuide should be initiated in Fab 6 so that the benefits knowledge transfer can be derived for a longer time. The emphasis should be on establishing troubleshooting procedures that increase in expertise over time.

To overcome the time constraints of the technical personnel and facilitate spreading TSGuide throughout the Fabs a divisional support person should be identified. This person should be a resource to help the technical group personnel develop troubleshooting guides and to provide training on using TSGuide. This is especially important during the infancy of the system. Once a critical mass of applications and experienced users were reached the need for this role would go away.

The sharing of knowledge about TSGuide experiences will help to reduce dependency on the resource person over the long run. Creating a user group to share knowledge about TSGuide experiences will facilitate this sharing of knowledge. For example, the resource person could establish a Notes file conference that allows users to share their experiences.

Appendix A: Examples of Methods Used to Transfer Knowledge

Notes File Checksheet

HIGH PARTICLE DECISION CHECKSHEET
 PARTICLE DECISION CHECKSHEET
 PARTICLE DECISION CHECKSHEET

DATE: 4/3/95 REACTOR#: #rf hours: TECH: B.C.

1) INITIAL PARTICLE CHECK:

CONTROL WAFER	YELLOW-0	ORANGE-0	BLUE-0	TOTAL-0
PEDESTAL 1A	YELLOW-84	ORANGE-66	BLUE-27	TOTAL-178
PEDESTAL 2B	YELLOW-34	ORANGE-46	BLUE-21	TOTAL-101

- If the control wafer is bad, then take note of any pattern, do a Production Weekly PM and retest.
- If retest proves Weekly PM is not sufficient, put system down provide EEG with your findings on the load chamber.
- If the control wafer is good and initial test was not viewed, rerun the wafers to see if a pattern does exist. Describe pattern if any and retest.
- If no pattern exists continue to #2.

FINDINGS:

2) RETEST USING THE LOADER SEQUENCE "RETEST"

CONTROL WAFER	YELLOW-	ORANGE-	BLUE-	TOTAL-
PEDESTAL 4A	YELLOW-	ORANGE-	BLUE-	TOTAL-
PEDESTAL 5B	YELLOW-	ORANGE-	BLUE-	TOTAL-

NOTES:

- If the retest is good contact Process Engineering for further instructions.
- If the retest is bad go to 3)

3) EEG WILL THEN ADDRESS THE PARTICLE ISSUE. PLEASE APPEND EEG CHECKSHEET HERE TO DOCUMENT ACTIONS TAKEN. CONTACT PRODUCTION FOR VERIFICATION TESTS.

4) VERIFICATION TEST (PRODUCTION):

PEDESTAL 1A	YELLOW-	ORANGE-	BLUE-	TOTAL-
PEDESTAL 2B	YELLOW-	ORANGE-	BLUE-	TOTAL-

- If the particles are now in spec and the initial particle check (step1) was only a subtle problem (<2xUCL), release system.
- If the particles are now in spec and the initial particle check was a gross problem (>2xUCL) continue to step #5.

5) RUN ONE PRODUCTION LOT AND DO ANOTHER PARTICLE CHECK:

PEDESTAL 1A	YELLOW-	ORANGE-	BLUE-	TOTAL-
PEDESTAL 2B	YELLOW-	ORANGE-	BLUE-	TOTAL-

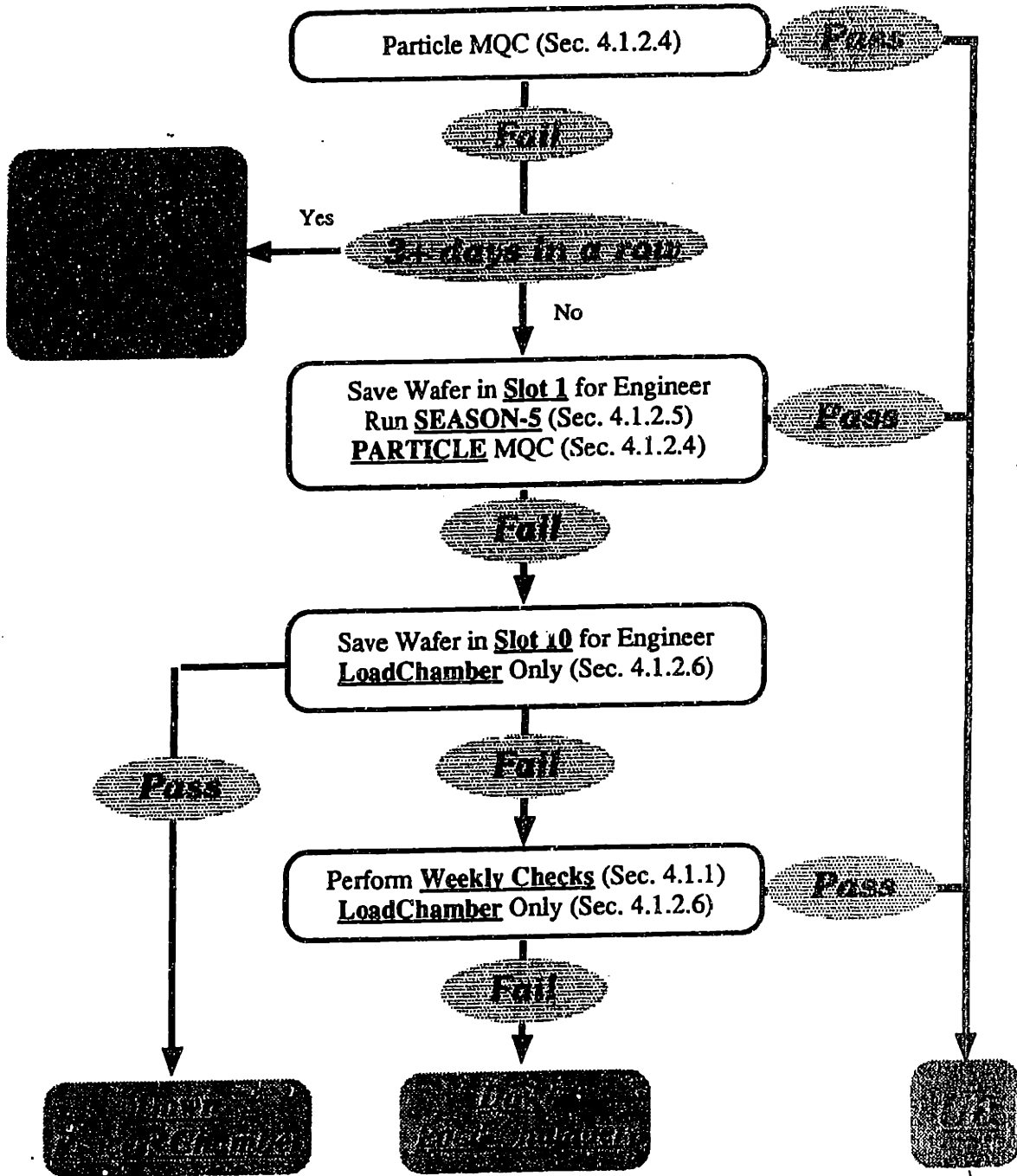
- If the particle check is still in spec, release the system
- If the particle check is out of spec place the system DOWN to EEG for further evaluations.

Operating Specification Section

- 7.3. Particle Trouble Shooting Procedure:
This procedure shall be used when Particle MQC is failed.
- 7.3.1 1st Failure of Daily Particle MQC:
You will need to examine the wafer map from surface scan. If particle distribution exhibits clustered pattern, Put the system down and state that the particle distribution is clustered at the corresponding location in the wafers. Also, if the particle MQC failed more than once in the past 7 days, put the system down immediately and state the reason of consecutive particle MQC failure. Otherwise, proceed to 7.3.2.
- 7.3.2 Save the (three) particle wafers for engineering evaluation. Run "SEAON-5" recipe, see section 4.1.2.5. Then re-do "Particle" MQC, see section 4.1.2.4.
- 7.3.3 If 2nd Particle MQC passes, system passes daily Particle MQC. When 2nd Particle MQC is failed for Process Chamber Test but not for Load Chamber Only Test, put system down for "Process Chamber or Transferring Mechanism". If Load Chamber test wafer failed, regardless Process Chamber wafer, save the (three) particle wafers for engineering evaluation. Perform Section 4.1.1 to ensure load chamber is clean. The re-do "Particle" MQC, see section 4.1.2.4.
- 7.3.4 If 3rd Particle MQC passes, system passes daily Particle MQC. When 3rd Particle MQC is failed, put the system down and state the failure categories of load chamber and/or process chamber.

Decision Tree Flowsheet

8330 Particle MQC Trouble Shooting Flow Chart



Appendix B: TSGuide Menu Program Code Listing

TSGuide_Main_Menu.Com

```
$!*****
$! *
$! * Program name: TSGuide_Main_Menu.Com
$! *
$! * Written by: Shawn Lambert Date: 10/27/95
$! *
$! *
$! *****
$!
$! This is the Main menu for the Troubleshooting Guide (TSGuide)
$!
$! The first menu (Main_Menu) points to the 6 other technical area menus (Area2_menu,
$! Area1_menu, Area4_menu, Area3_menu, DRAT_menu and metrology_menu).
$!
$! In turn, these Tech area menus point to com files representing specific toolgroup menus.
$!
$! When control is returned from the toolgroup com file the BKUP_Flag and Quit_flag are
$! evaluated to either return to the last menu or quit the program.
$!
$! Selections can be added to the menus by editing the menu choices and selection evaluation
$! sections of each menu.
$!
$! Each toolgroup menu choice must have a Com file located at the path defined by Group,$.
$!
$! This com file can be started with a parameter that indicates the Tech area that the user desires
$! to access. This the Main_menu and goes directly to that Tech area's menu.
$!
$! @TSGuide_Main_menu [tech option]
$!
$! ie: @TSGuide_Main_menu Area2 or @TSGuide_main_menu fi
$!
$! Would cause the program to branch to the Area2_menu section of this Com file. Only the
$! first two letters are needed and is case insensitive.
$!
$! ***** Initialization
$!
$ ON Error Then Goto Quit
$ ON CONTROL_Y THEN Goto Quit
$ ON CONTROL_C THEN Goto Quit
$! ESC[0,32]=%X1B
$ ESC[0,8]=%X1B
$ SCREEN := WRITE SYSS$OUTPUT
$ SET TERM/WIDTH=80
$!
$ Menu := Main_menu_choice
```

```

$ Quit_flag == 0
$ BKUP_Flag == 0
$!
$! *****
$! * MAIN_MENU                                     "
$! *                                             "
$! * This is the Main menu. It is different than the tech area menus      "
$! * because it looks for a passed parameter and does not allow backing    "
$! * up to the previous menu since this is the first menu.                 "
$! *                                                                            "
$! *****
$!
$ If (P1 .EQS. "") Then Goto Main_menu ! Checks for a parameter at P1
$ COM = F$Edit(F$Extract(0,2,P1),"UPCASE") ! Converts P1 to 2 char UpperCase
$ Goto Main_Menu_Choice ! and saves result as COM
$!
$ Main_Menu:
$ Menu := Main_menu_choice
$ SCREEN ESC,"[0H",ESC,"[0J"
$ SCREEN ESC,"[2;6H ain Troubleshooting Menu"
$ SCREEN ESC,"[4;16H 1. Area2"
$ SCREEN ESC,"[6;16H 2. Area1"
$ SCREEN ESC,"[8;16H 3. Area4"
$ SCREEN ESC,"[10;16H 4. Area3"
$ SCREEN ESC,"[12;16H 5. DRAT"
$ SCREEN ESC,"[14;16H 6. Metrology"
$!
$ Goto Selection
$!
$ Main_menu_Choice:
$ IF (COM .eqs. "P" .or. COM .eqs. "p")
$ Then
$ Screen ESC,"[20;16H This is the first menu. "
$ Screen ESC,"[22;16H Hit Return to Continue "
$ Inquire/nopunct INPUT ""
$ Goto Selection
$ EndIf
$ IF (COM .eqs. "F1" .or. COM .eqs. "1") THEN Goto Area2_menu
$ IF (COM .eqs. "ET" .or. COM .eqs. "2") THEN Goto Area1_menu
$ IF (COM .eqs. "PH" .or. COM .eqs. "3") THEN Goto Area4_menu
$ IF (COM .eqs. "IM" .or. COM .eqs. "DI" .or. COM .eqs. "4") THEN -
Goto Area3_menu
$ IF (COM .eqs. "DR" .or. COM .eqs. "5") THEN Goto DRAT_menu
$ IF (COM .eqs. "ME" .or. COM .eqs. "6") THEN Goto Metrology_menu
$!
$ Goto Wrong ! None of the answers are valid
$!
$! *****
$! * AREA2_MENU                                     *
$! *                                             *
$! *                                             *
$! *****
$!
$ Area2_Menu:

```

```

$ Define Group$ "Node::public$:[dectree.Area2]" !Area2 menu location
$ Menu := Area2_menu_choice
$ SCREEN ESC,"[0H",ESC,"[0J"
$ SCREEN ESC,"[2;6H rea2 Troubleshooting Menu"
$ SCREEN ESC,"[4;16H 1. OXIDE's"
$ SCREEN ESC,"[6;16H 2. MDT's"
$ SCREEN ESC,"[8;16H 3. TDT's"
$ SCREEN ESC,"[10;16H 4. SOG"
$!
$ Goto Selection
$!
$ Area2_menu_Choice:
$ IF (COM .eqs. "P" .or. COM .eqs. "p") Then Goto Main_meru ! Menu to backup
$!
$ IF (COM .eqs. "1") Then @Group$:TSGuide_OXIDE_menu
$!
$ IF (COM .eqs. "2") Then @Group$:TSGuide_MDT_menu
$!
$ IF (COM .eqs. "3") Then Goto Noexist ! @Group$:Tsguide_TDT_menu.com
$!
$ IF (COM .eqs. "4") Then Goto Noexist ! @Group$:Tsguide_SOG_menu.com
$!
$ If BkUP_Flag          ! If BKUP_Flag = 1 return to last menu
$ Then
$   BKUP_Flag == 0
$   Quit_flag == 0
$   Goto Area2_menu
$ EndIf
$!
$ IF Quit_Flag Then Goto Quit ! If Quit_flag has been set to 1 exit prog
$!
$ Goto Wrong          ! If incorrect response to menu
$!
$! *****
$! * AREA1_MENU *
$! * *
$! * *
$! *****
$!
$ Area1_Menu:
$ Define Group$ "Node::public$:[dectree.Area1]" !Area1 menu location
$ Menu := Area1_menu_choice
$ SCREEN ESC,"[0H",ESC,"[0J"
$ SCREEN ESC,"[2;6H real Troubleshooting Menu"
$ SCREEN ESC,"[4;16H 1. ETCHER1's"
$ SCREEN ESC,"[6;16H 2. ETCHER2's"
$ SCREEN ESC,"[8;16H 3. ETCHER3's"
$ SCREEN ESC,"[10;16H 4. ETCHER4's"
$!
$ Goto Selection
$!
$ Area1_menu_Choice:
$ IF (COM .eqs. "P" .or. COM .eqs. "p") Then Goto Main_menu ! Menu to backup
$!

```

```

$ IF (COM .eqs. "1") Then Goto Noexist ! @Group$:Tsguide_ETCHER1_menu.com
$!
$ IF (COM .eqs. "2") Then Goto Noexist ! @Group$:Tsguide_ETCHER2_menu.com
$!
$ IF (COM .eqs. "3") Then @Group$:Tsguide_lamArea1_menu.com
$!
$ IF (COM .eqs. "4") Then Goto Noexist ! @Group$:Tsguide_ETCHER4_menu.com
$!
$ If BkUP_Flag          ! If BKUP_Flag = 1 return to last menu
$ Then
$   BKUP_Flag == 0
$   Quit_flag == 0
$   Goto Area1_menu
$ EndIf
$!
$ IF Quit_Flag Then Goto Quit ! If Quit_flag has been set to 1 exit prog
$!
$ Goto Wrong          ! If incorrect response to menu
$!
$! *****
$! * AREA4_MENU *
$! * *
$! * *
$! *****
$!
$ Area4_Menu:
$ Define Group$ "Node::public$:[dectree.Area4]" !Area4 menu location
$ Menu := Area4_menu_choice
$ SCREEN ESC,"[0H",ESC,"[0J"
$ SCREEN ESC,"[2;6H rea4 Troubleshooting Menu" !Area4 main menu
$ SCREEN ESC,"[4;16H 1. STEPPER's"
$ SCREEN ESC,"[6;16H 2. Coat"
$ SCREEN ESC,"[8;16H 3. Dev"
$ SCREEN ESC,"[10;16H 4. ????"
$!
$ Goto Selection
$!
$ Area4_menu_Choice:
$ IF (COM .eqs. "P" .or. COM .eqs. "p") Then Goto Main_menu
$!
$ IF (COM .eqs. "1") Then Goto Noexist ! @Group$:Tsguide_STEPPER_menu.com
$!
$ IF (COM .eqs. "2") Then Goto Noexist ! @Group$:Tsguide_Coat_menu.com
$!
$ IF (COM .eqs. "3") Then Goto Noexist ! @Group$:Tsguide_Dev_menu.com
$!
$ IF (COM .eqs. "4") Then Goto Noexist ! @Group$:Tsguide_????_menu.com
$!
$ If BkUP_Flag          ! If BKUP_Flag has been set to 1
$ Then
$   BKUP_Flag == 0
$   Quit_flag == 0
$   Goto Area4_menu
$ EndIf

```

```

$!
$ IF Quit_Flag Then Goto Quit      ! If Quit_flag has been set to 1
$ Goto Wrong                       ! If incorrect response to menu
$!
$! *****
$! * AREA3_MENU                      *
$! *                                *
$! *                                *
$! *****
$!
$ AREA3_Menu:
$ Define Group$ "Node::public$:[dectree.AREA3]" !Area3 menu loc
$ Menu := Area3_menu_choice
$ SCREEN ESC,"[0H",FSC,"[0J"
$ SCREEN ESC,"[2;6H mp and Diff Troubleshooting Menu" !main menu
$ SCREEN ESC,"[4;16H 1. Implant"
$ SCREEN ESC,"[6;16H 2. Tube11/12"
$ SCREEN ESC,"[8;16H 3. Tube21/22"
$ SCREEN ESC,"[10;16H 4. Tube31/41"
$!
$ Goto Selection
$!
$ Area3_menu_Choice:
$ IF (COM .eqs. "P" .or. COM .eqs. "p") Then Goto Main_menu
$!
$ IF (COM .eqs. "1") Then Goto Noexist ! @Group$:Tsguide_Implant_menu.com
$!
$ IF (COM .eqs. "2") Then Goto Noexist ! @Group$:Tsguide_Tube11/12_menu.com
$!
$ IF (COM .eqs. "3") Then Goto Noexist ! @Group$:Tsguide_Tube21/22_menu.com
$!
$ IF (COM .eqs. "4") Then Goto Noexist ! @Group$:Tsguide_Tube31/41_menu.com
$!
$ If BkUP_Flag                       ! If BKUP_Flag has been set to 1
$ Then
$   BKUP_Flag == 0
$   Quit_flag == 0
$   Goto Area3_menu
$ EndIf
$!
$ IF Quit_Flag Then Goto Quit      ! If Quit_flag has been set to 1
$!
$ Goto Wrong                       ! If incorrect response to menu
$!
$! *****
$! * DRAT_MENU                      *
$! *                                *
$! *                                *
$! *****
$!
$ DRAT_Menu:
$ Define Group$ "Node::public$:[dectree.DRAT]" !Drat menu location
$ Menu := DRAT_menu_choice
$ SCREEN ESC,"[0H",ESC,"[0J"

```



```

$ SCREEN ESC,"[2;6H RAT Troubleshooting Menu"    !Drat main menu
$ SCREEN ESC,"[4;16H 1.  ??????"
$ SCREEN ESC,"[6;16H 2.  ??????"
$ SCREEN ESC,"[8;16H 3.  ??????"
$ SCREEN ESC,"[10;16H 4.  ??????"
$!
$ Goto Selection
$!
$ Drat_menu_Choice:
$ IF (COM .eqs. "P" .or. COM .eqs. "p") Then Goto Main_menu
$!
$ IF (COM .eqs. "1") Then Goto Noexist ! @Group$:Tsguide_????_menu.com
$!
$ IF (COM .eqs. "2") Then Goto Noexist ! @Group$:Tsguide_????_menu.com
$!
$ IF (COM .eqs. "3") Then Goto Noexist ! @Group$:Tsguide_????_menu.com
$!
$ IF (COM .eqs. "4") Then Goto Noexist ! @Group$:Tsguide_????_menu.com
$!
$ If BkUP_Flag
$ Then
$   BKUP_Flag == 0
$   Quit_flag == 0
$   Goto DRAT_menu
$ EndIf
$!
$ IF Quit_Flag Then Goto Quit    ! If Quit_flag has been set to 1
$!
$ Goto Wrong    ! If incorrect choice
$!
$! *****
$! * Metrology_MENU *
$! * *
$! * *
$! *****
$!
$ Metrology_Menu:
$ Define Group$ "Node::public$:[dectree.Metro]" !Metro menu location
$ Menu := Metrology_menu_choice
$ SCREEN ESC,"[0H",ESC,"[0J"
$ SCREEN ESC,"[2;6H etrology Troubleshooting Menu" !Metrology main menu
$ SCREEN ESC,"[4;16H 1. FT's"
$ SCREEN ESC,"[6;16H 2. Probe"
$ SCREEN ESC,"[8;16H 3. P1 & P2"
$ SCREEN ESC,"[10;16H 4. Coming soon"
$!
$ Goto Selection
$!
$ Metrology_menu_Choice:
$ IF (COM .eqs. "P" .or. COM .eqs. "p") Then Goto Main_menu
$!
$ IF (COM .eqs. "1") Then @Group$:TSguide_FT_menu
$!
$ IF (COM .eqs. "2") Then Goto Noexist ! @Group$:Tsguide_Probe_menu.com

```

```

$!
$ IF (COM .eqs. "3") Then Goto Noexist ! @Group$:Tsguide_P1&P2_menu.com
$!
$ IF (COM .eqs. "4") Then Goto Noexist ! @Group$:Tsguide_ComingSoon_menu.com
$!
$ If BkUP_Flag
$ Then
$ BKUP_Flag == 0
$ Quit_flag == 0
$ Goto Metrology_menu
$ EndIf
$!
$ IF Quit_Flag Then Goto Quit ! If Quit_flag has been set to 1
$!
$ Goto Wrong ! If incorrect choice
$!
$! *****
$! * SELECTION *
$! * *
$! * This section handles getting the prompt from the user *
$! * *
$! *****
$!
$ Selection:
$ SCREEN ESC,"[20;16H Enter Your Choice: 7m 0m "
$ SCREEN ESC,"[22;16H Type E To Exit Program or P for Previous Menu "
$ INQUIRE/NOPUNCT COM "20;36H7m 0m"
$ If (COM .eqs. "E" .or COM .eqs. "Q") Then goto quit
$ Goto 'Menu'
$!
$! *****
$! * NOEXIST *
$! * *
$! * This section writes out an error for a valid entry that doesn't *
$! * have any supporting menus, yet It redirects the pointer to *
$! * prompt for a different choice. *
$! * *
$! *****
$!
$ Noexist:
$ Quit_flag = 0
$ Screen ESC,"[20;16H No Troubleshooting Help Available, Yet "
$ SCREEN ESC,"[22;16H Hit Return to Continue "
$ Inquire/nopunct INPUT ""
$ Goto Selection

```

```

$!
$! *****
$! * WRONG *
$! * *
$! * This section writes out an error for an invalid entry. It *
$! * redirects the pointer to prompt for a different choice. *
$! * *
$! *****
$!
$! Wrong:
$! Screen ESC,"[20;16H Invalid Choice "
$! SCREEN ESC,"[22;16H Hit Return to Continue "
$! Inquire/nopunct INPUT ""
$! Goto Selection
$!
$! *****
$! * QUIT *
$! * *
$! * This section exits the com file in a controlled way by deassigning *
$! * Logicals and deleting files, etc. *
$! * *
$! *****
$!
$! Quit:
$! Deassign Group$
$! SCREEN ESC,"[0H",ESC,"[0J" ! Clears the screen
$! EXIT

```

TSGuide_MDT_Menu.Com

```
$! *****
$! *
$! * Program name: TSGuide_MDT_Menu.Com
$! *
$! * Written By: Shawn Lambert Date: 10/27/95
$! *
$! *****
$!
$! This is the menu for the MDT's Troubleshooting Guide. It is called by the Area2_menu in
$! TSGuide_Main_Menu.com. This Com file contains two levels of menus. The first lists the
$! specific entities to troubleshoot and the next level contains the various Decrees that exist
$! for that Entity. Generally, the second level menu is the same for all of the Entities.
$!
$! The menu choices will cause a Decree image file (*.exe) to be run via the spawn command.
$! When the image file is terminated this com file regains control. If the Notesfile_Flag is set
$! to 1 then the user is prompted to send the Decree log file to Notes. The last menu is prior
$! to executing the Decree is then displayed.
$!
$! This com file is terminated by either requesting to back-up to the previous menu or
$! requesting to quit the program.
$!
$! **** Key Variables ****
$!
$! Tree$ is assigned to the node, disk and directory where the DECtree's are stored. The
$! protection on the DECtree's must be set to at least read for S,O,G,W.
$!
$! The string variables notesfile, note, logfile, and subject are used to write the Decree log
$! file to the notesfile conference and note by creating a com file called [ ]temp.com. The notes
$! file conference must be set for Read and Write for Group and World Read. The notes_flag
$! variable needs to be set to 1 to cause a notesfile update request.
$!
$! The BKUP_Flag and QUIT_Flag are global variables used to direct the calling procedure upon
$! termination. If BKUP_Flag is set to 1 the main com file will display the last menu prior to
$! calling this routine. If BKUP_Flag is set to 0 and Quit_Flag is set to 1 then the main routine
$! will terminate.
$!
$! =====
$!
$! **** Initialization ****
$!
$ ON Error Then Goto Quit
$ ON CONTROL_Y THEN Goto Quit
$ ON CONTROL_C THEN Goto Quit
$ ESC[0,32]=%X1B
$ SCREEN := WRITE SYSS$OUTPUT
$ Define DECTREE$DISPLAY Terminal ! Sets interface to character terminal
$ SET TERM/WIDTH=80
$ BKUP_Flag == 0
$ Quit_Flag == 1
```

```

$ Define Tree$ Node::Public$:[dectree.Area2.MDT]
$ Define Mod$ Node::Public$:[dectree.Modules]
$ Notesfile := "Node::AppS1$:[Notes$library]Area2_pass.note"
$!
$! *****
$! * MDT_Menu
$! *****
$!
$ MDT_menu:
$ Menu := MDT_menu_choice
$ SCREEN ESC,"[0H",ESC,"[0J"
$ SCREEN ESC,"[2;6H DT Troubleshooting Menu"
$ SCREEN ESC,"[4;16H 1. MDT.1"
$ SCREEN ESC,"[5;16H 2. MDT.2"
$ SCREEN ESC,"[6;16H 3. MDT.3"
$ SCREEN ESC,"[7;16H 4. MDT.4"
$ SCREEN ESC,"[8;16H 5. MDT.5"
$!
$ Goto Selection
$!
$ MDT_Menu_CHOICE:
$ IF (COM .eqs. "P" .or. COM .eqs. "p")
$ Then
$ BKUP_Flag == 1
$ Quit_Flag == 0
$ Goto Quit
$ EndIf
$!
$ IF (COM .eqs. "1")
$ THEN
$ Entity := MDT_1 ! name of next menu
$ Note := 131.1 ! note is defined as the note for log file storage
$ Note_AI := 312.1
$ Goto 'Entity'
$ EndIf
$!
$ IF (COM .eqs. "2")
$ THEN
$ Entity := MDT_2
$ Note := 132.1 ! note is defined as the note for log file storage
$ Note_AI := 313.1
$ Goto 'Entity'
$ EndIf
$!
$ IF (COM .eqs. "3")
$ THEN
$ Entity := MDT_3
$ Note := 133.1 ! note is defined as the note for log file storage
$ Note_AI := 314.1
$ Goto 'Entity'
$ EndIf
$!
$ IF (COM .eqs. "4")
$ THEN

```

```

$ Entity := MDT_4
$ Note := 134.1 ! note is defined as the note for log file storage
$ Note_AI := 315.1
$ Goto 'Entity'
$ EndIf
$!
$ IF (COM .eqs. "5")
$ THEN
$ Entity := MDT_5
$ Note := 135.1 ! note is defined as the note for log file storage
$ Note_AI := 346.1
$ Goto 'Entity'
$ EndIf
$!
$ Goto Wrong
$!
$! *****
$! * ENTITY_Menu
$! *****
$!
$ MDT_1:
$ Menu := MDT_1_menu_choice
$ MDT_2:
$ Menu := MDT_2_menu_choice
$ MDT_3:
$ Menu := MDT_3_menu_choice
$ MDT_4:
$ Menu := MDT_4_menu_choice
$ MDT_5:
$ Menu := MDT_5_menu_choice
$!
$ SCREEN ESC,"[0H",ESC,"[0J"
$!
$ SCREEN ESC,"[2;6H nity," Troubleshooting Menu"
$ SCREEN ESC,"[4;14H 1. AI Deposition Time Adjustment Guide"
$ SCREEN ESC,"[6;14H 2. TiNP Lot Monitor fails for Particles"
$ SCREEN ESC,"[8;14H 3. TiNP MQC fails for Particles"
$ SCREEN ESC,"[10;14H 4. Walk MQC fails for particles"
$!
$ Goto Selection
$!
$ MDT_1_menu_Choice:
$ MDT_2_menu_Choice:
$ MDT_3_menu_Choice:
$ MDT_4_menu_Choice:
$ MDT_5_menu_Choice:
$ IF COM .EQS. "P" .or. COM .EQS. "p" THEN Goto MDT_Menu
$!
$ IF (COM .eqs. "1")
$ THEN
$ Define/user_mode Sys$input Sys$command ! redirects the program input
$ run Tree$:MDT_T_Adj_V1
$ Notesfile_flag = 1 ! Set to 1 to write logfile to notes
$ Subject := "AI deposition time adjustment" ! notesfile subject

```

```

$ Note := 'Note_AI'
$ logfile := MDT_T_Adj_V1.Log      ! file to be written to notes
$ Goto Return
$ EndIF
$!
$ IF (COM .eqs. "2")
$ THEN
$!   Define/user_mode Sys$input Sys$command ! redirects the program input
$ Goto NoExist      ! run Tree$:PSG_proc
$ Notesfile_flag = 1 ! Set to 1 to write logfile to notes
$ Subject := "Lot Monitor Fails for Particles" ! notesfile subject
$ logfile := PSG_Proc.Log ! file to be written to notes
$ Goto Return
$ EndIF
$!
$ IF (COM .eqs. "3")
$ THEN
$!   Define/user_mode Sys$input Sys$command ! redirects the program input
$ Goto NoExist      ! run Tree$:PSG_proc
$ Subject := "MQC Fails for Particles" ! notesfile subject
$ logfile := PSG_Proc.Log ! file to be written to notes
$ Notesfile_flag = 0 ! Set to 1 to write logfile to notes
$ Goto Return
$ EndIF
$!
$ IF (COM .eqs. "4")
$ THEN
$!   Define/user_mode Sys$input Sys$command ! redirects the program input
$ Goto NoExist      ! run Tree$:PSG_proc
$ Subject := "PSG Process Centering Adjustments" ! notesfile subject
$ logfile := PSG_Proc.Log ! file to be written to notes
$ Notesfile_flag = 0 ! Set to 1 to write logfile to notes
$ Goto Return
$ EndIF
$!
$ Goto Wrong
$!
$! *****
$! * SELECTION *
$! * *
$! * This procedure handles the users selection from the various menus *
$! * *
$! *****
$!
$ Selection:
$ SCREEN ESC,"[20;16H Enter Your Choice: 7m 0m "
$ SCREEN ESC,"[22;16H Type E To Exit Program or P for Previous Menu "
$ INQUIRE/NOPUNCT COM "20;36H7m 0m"
$ If (COM .eqs. "E" .or. COM .eqs. "Q") Then Goto Quit
$ Goto 'Menu'
$!

```

```

$! *****
$! * NOEXIST *
$! * *
$! * This procedure handles the selections without Options available *
$! * *
$! *****
$!
$ Noexist:
$ Screen ESC,"[20;16H No Troubleshooting Help Available, Yet "
$ Screen ESC,"[22;16H Hit Return to Continue "
$ Inquire/nopunct INPUT ""
$ Goto Selection
$!
$! *****
$! * WRONG *
$! * *
$! * This procedure handles invalid choices *
$! * *
$! *****
$!
$ Wrong:
$ Screen ESC,"[20;16H Invalid Choice "
$ Screen ESC,"[22;16H Hit Return to Continue "
$ Inquire/nopunct INPUT ""
$ Goto Selection
$!
$! *****
$! * RETURN *
$! * *
$! * This procedure is called when Dectree returns control. It tests *
$! * the value of notes_flag to create a com file which posts a note *
$! * to the defined Notesfile, note and subject. The com file is *
$! * deleted after execution *
$! * *
$! *****
$!
$ Return:
$ SCREEN ESC,"[0H",ESC,"[0J"
$ Screen ESC,"[22;16H Hit Return to Continue"
$ Inquire/nopunct Input ""
$ IF notesfile_flag
$ Then
$ SCREEN ESC,"[20;16H Send Log File to notes conference?(y): 7m 0m "
$ SCREEN ESC,"[22;16H "
$ INQUIRE/NOPUNCT LOG "20;56H7m 0m"
$ If Log .eqs. "" .or. LOG .eqs. "y" .or. LOG .eqs. "Y"
$ Then ! Create a temp file to update notes file
$ Set Protection=(G:R,W:R) 'logfile'
$ Open/write outfile [ ]Temp.com;100
$ Write Outfile "$ Notes/NoNotebook ",notesfile " ",note
$ Write Outfile "Reply ",logfile," /NoEdit/NoConfirm/NoExtract" -
, "/title=""subject""
$ Write Outfile "Exit"
$ Close Outfile

```



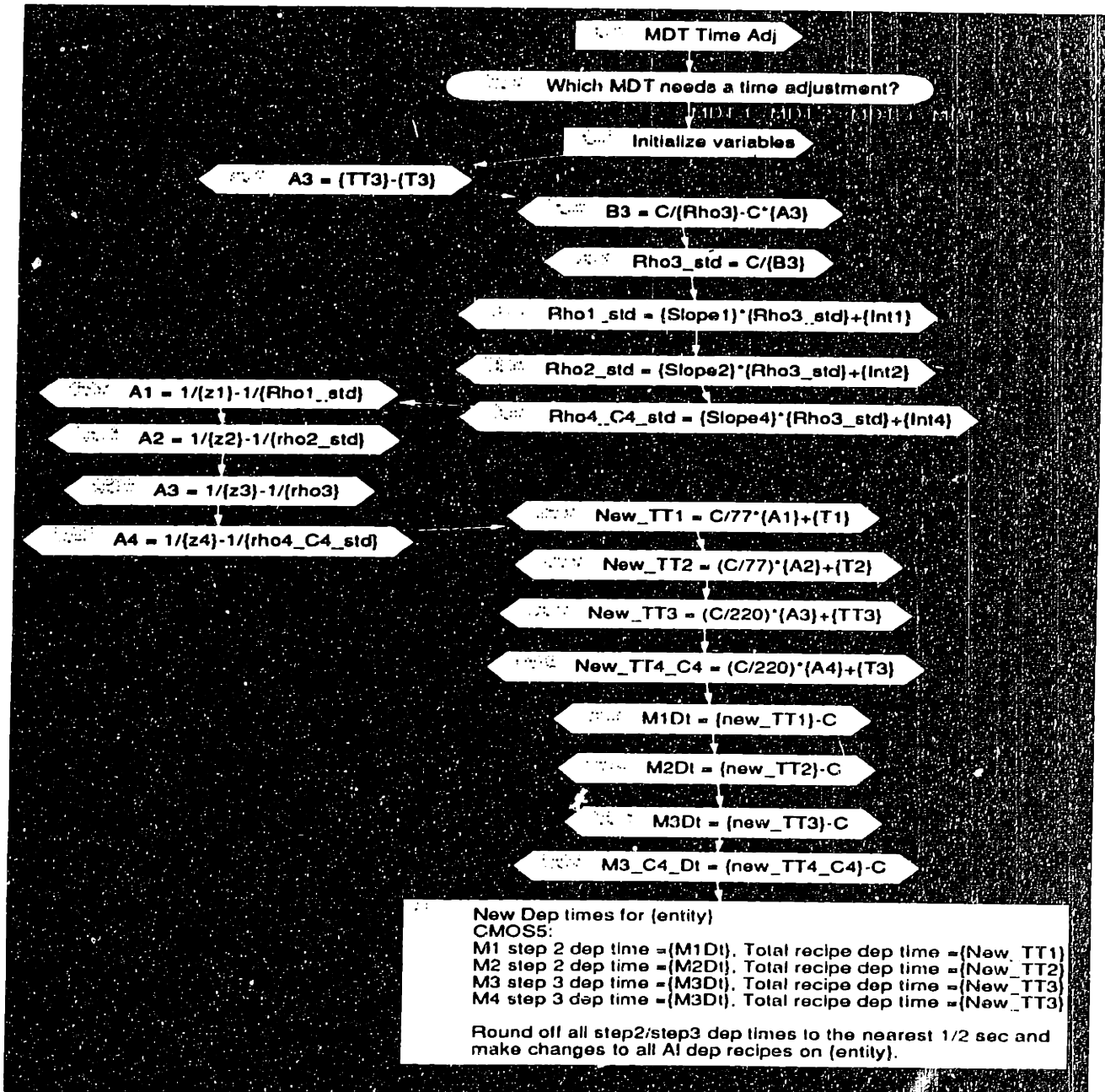
```

$ @[]temp
$ Delete/noconfirm Temp.com;100
$ EndIf
$ EndIf
$ Goto 'Entity'
$!
$! *****
$! * QUIT *
$! * *
$! * This procedure exits this com file and performs some clean-up *
$! * *
$! *****
$!
$ Quit:
$ Delete/noconfirm 'logfile';*
$ Deassign tree$
$ Deassign Mod$
$ DEASSIGN DECTREE$DISPLAY ! Returns to default display type
$ SCREEN ESC,"[0H",ESC,"[0J"
$ EXIT

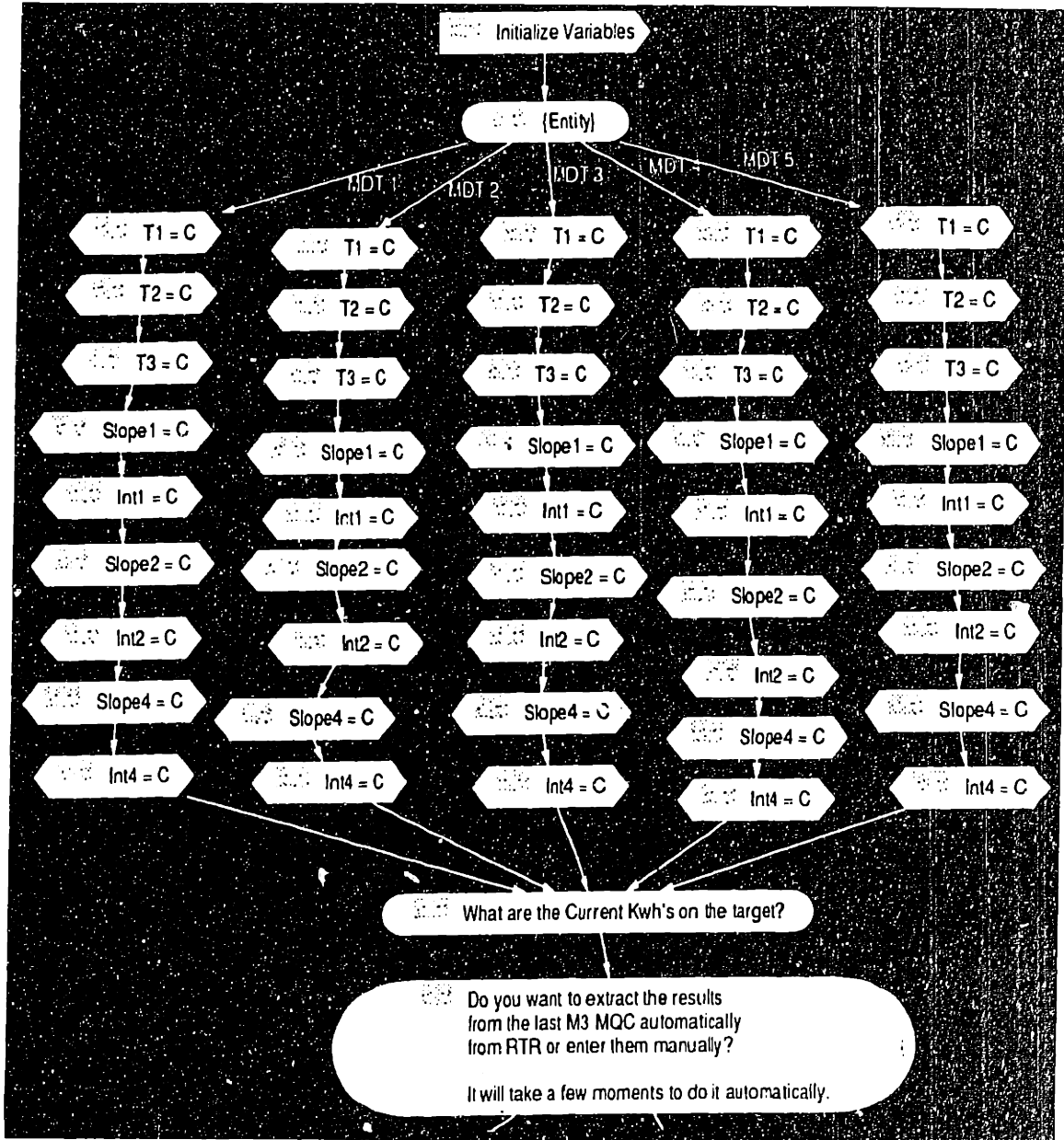
```

Appendix C: MDT Aluminum Deposition Time Adjustment DECtree

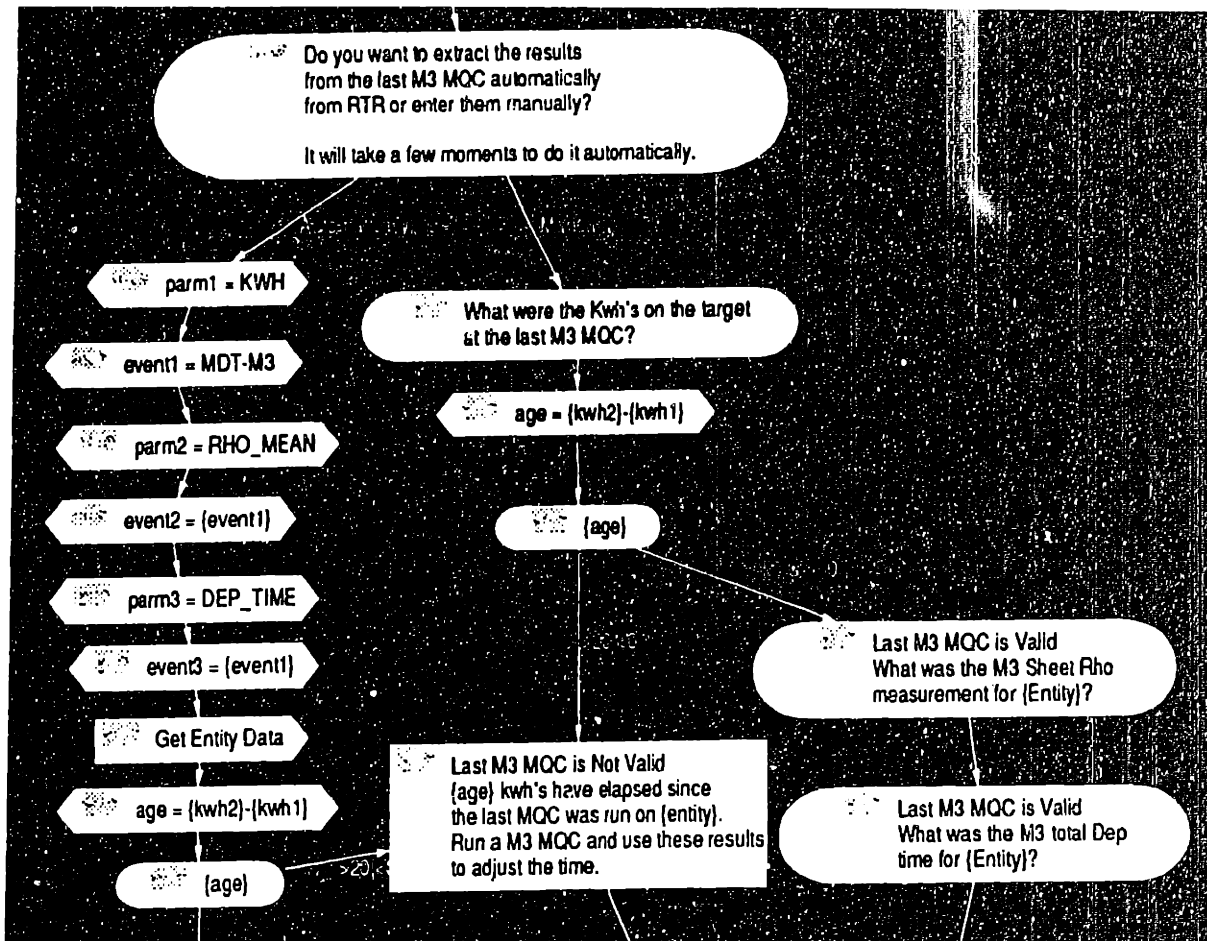
MDT Time Adj Subtree



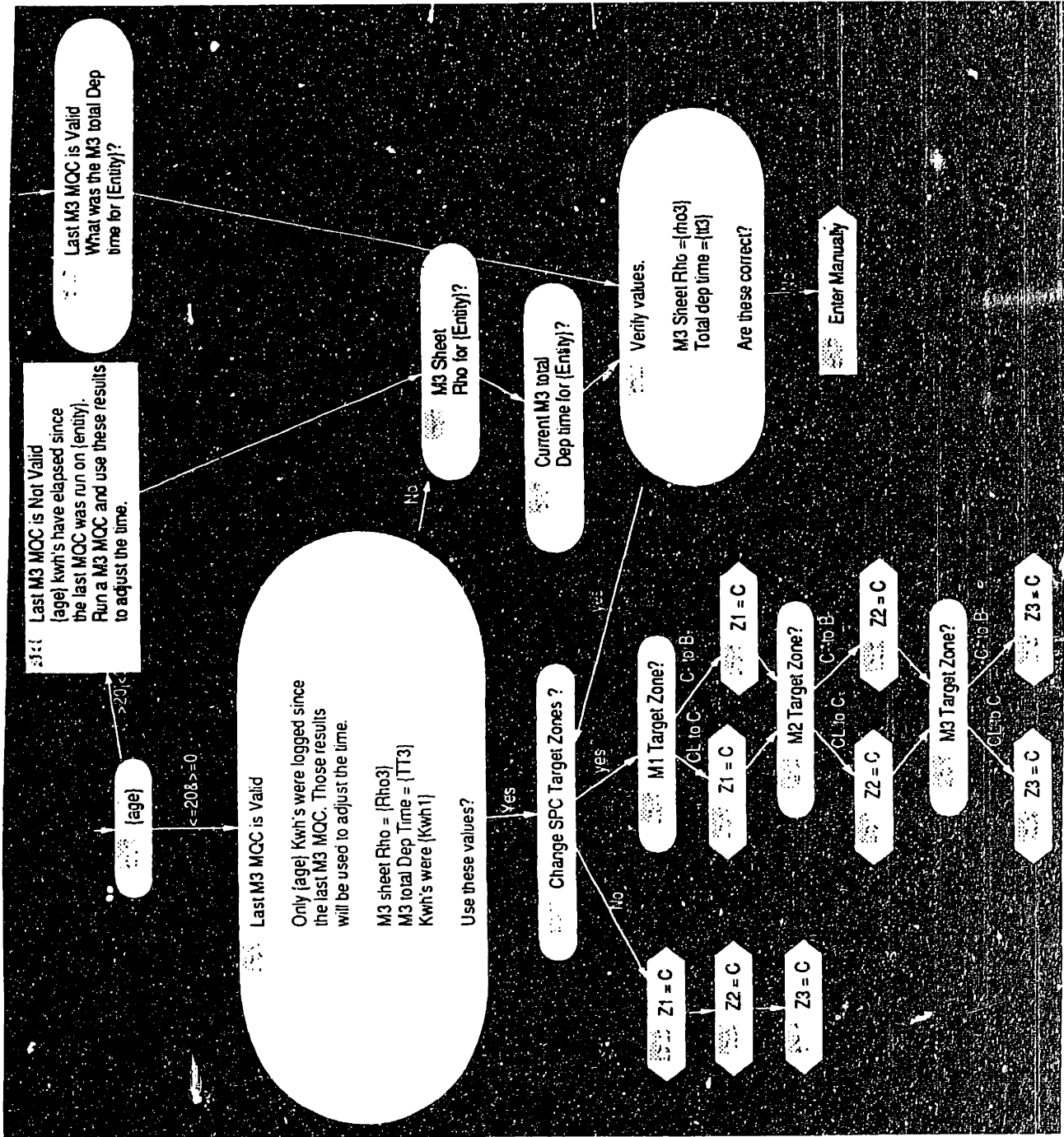
Initialize Variables Subtree: Top



Initialize Variables Subtree: Middle



Initialize Variables Subtree: Bottom



Get Entity Data Subtree

Get Entity Data

set schema RDB\$Schema;

```
Select A.num_parm value 1
From NTCDat A, NTCenh B
where ((A.Ent_hist_seq_date = B.Ent_hist_seq_date) and (A.ent_hist_seq_time =
B.ent_hist_seq_time)) and A.Facility = 'PLF' and A.Database_name = 'HLO_MFG1' and
A.Entity = '{entity}' and B.Event = '{Event1}' and A.Parameter_name = '{Parm1}' and
A.Ent_Hist_Seq_Date > 951001 and B.facility = 'PLF' and B.database_name = 'HLO_MFG1'
and B.Entity = '{entity}' and B.ent_hist_seq_date > 951001
Order By A.Ent_hist_seq_date desc, A.Ent_hist_seq_time desc
Limit To 1 row:
```

```
Select A.num_parm value 1
From NTCDat A, NTCenh B
where ((A.Ent_hist_seq_date = B.Ent_hist_seq_date) and (A.ent_hist_seq_time =
B.ent_hist_seq_time)) and A.Facility = 'PLF' and A.Database_name = 'HLO_MFG1' and
A.Entity = '{entity}' and B.Event = '{Event2}' and A.Parameter_name = '{Parm2}' and
A.Ent_Hist_Seq_Date > 951001 and B.facility = 'PLF' and B.database_name = 'HLO_MFG1'
and B.Entity = '{entity}' and B.ent_hist_seq_date > 951001
Order By A.Ent_hist_seq_date desc, A.Ent_hist_seq_time desc
Limit To 1 row:
```

```
Select A.num_parm value 1
From NTCDat A, NTCenh B
where ((A.Ent_hist_seq_date = B.Ent_hist_seq_date) and (A.ent_hist_seq_time =
B.ent_hist_seq_time)) and A.Facility = 'PLF' and A.Database_name = 'HLO_MFG1' and
A.Entity = '{entity}' and B.Event = '{Event3}' and A.Parameter_name = '{Parm3}' and
A.Ent_Hist_Seq_Date > 951001 and B.facility = 'PLF' and B.database_name = 'HLO_MFG1'
and B.Entity = '{entity}' and B.ent_hist_seq_date > 951001
Order By A.Ent_hist_seq_date desc, A.Ent_hist_seq_time desc
Limit To 1 row:
```

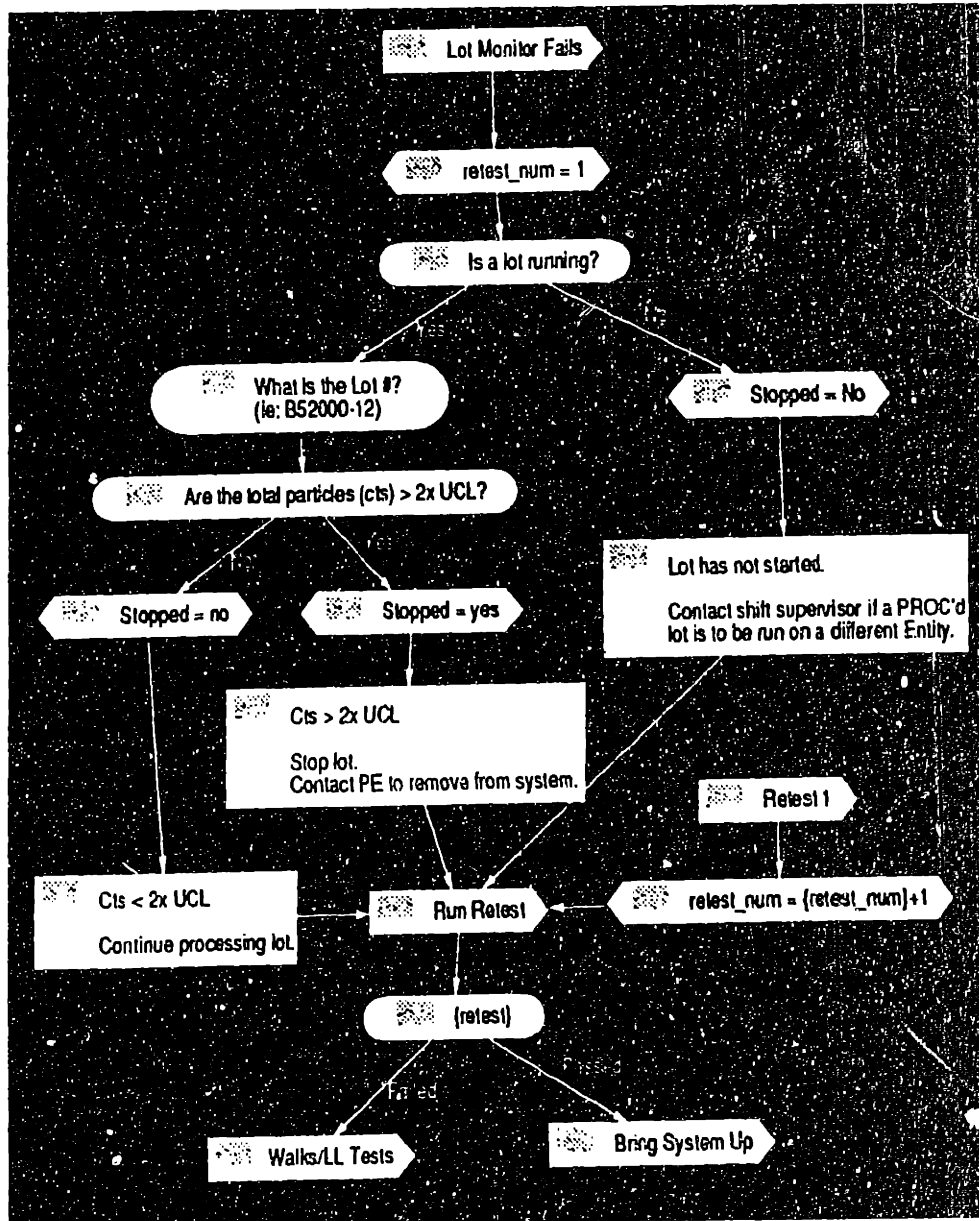
kwh1 = {data1}

Rho3 = {Data2}

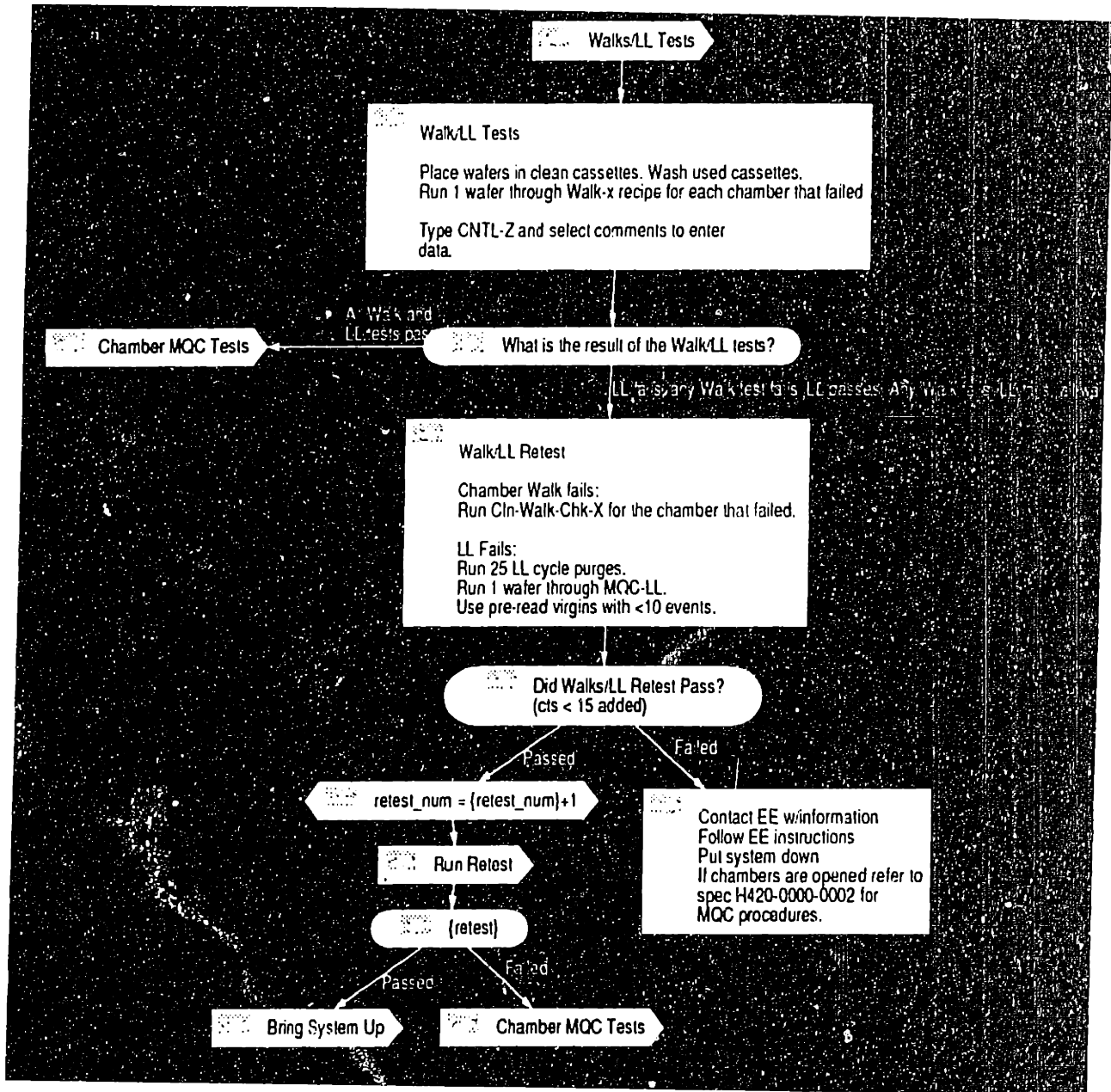
TT3 = {Data3}

Appendix D: OXIDE Lot Monitor Particle Failure Tree

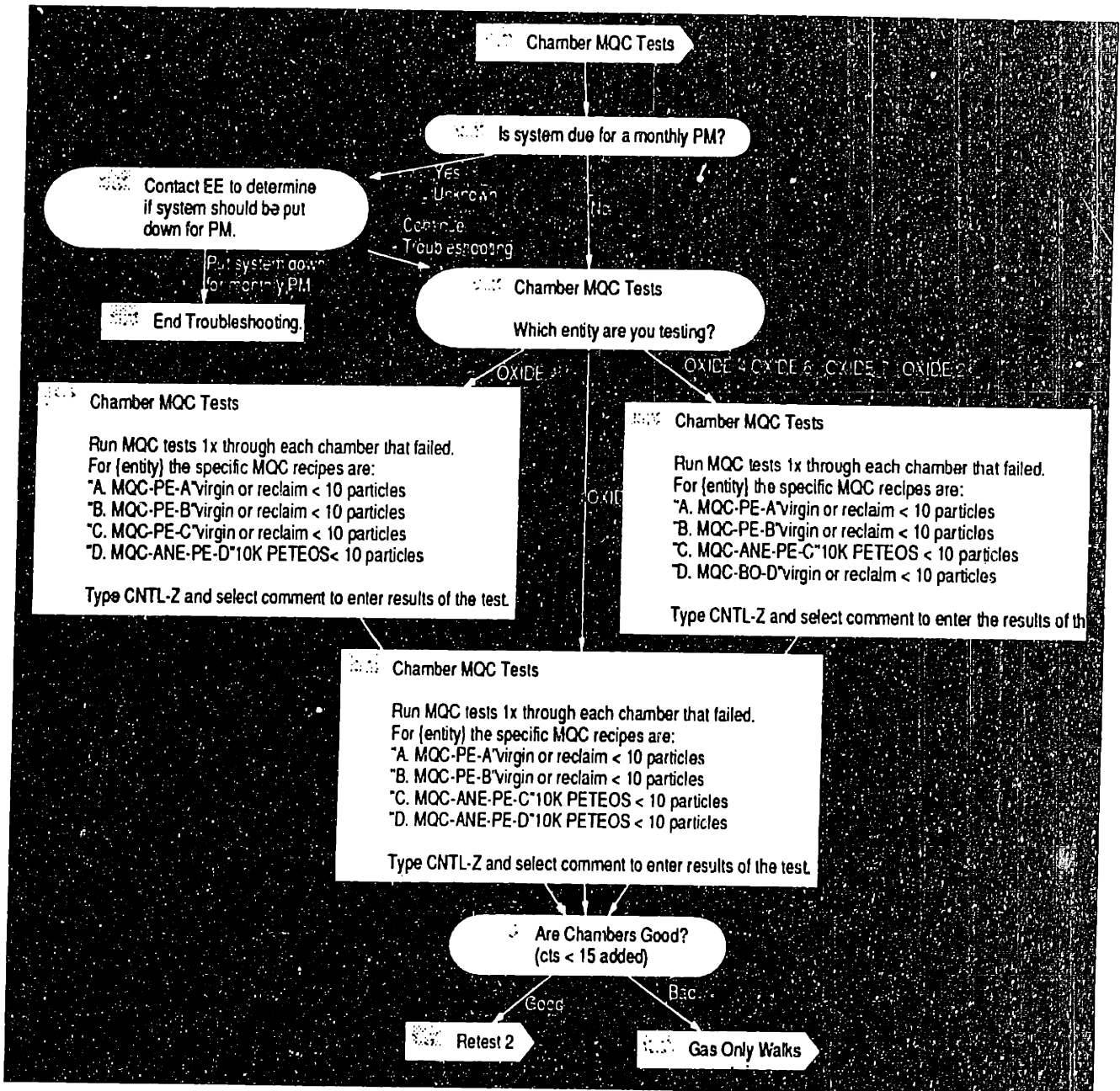
Lot Monitor Fails Subtree



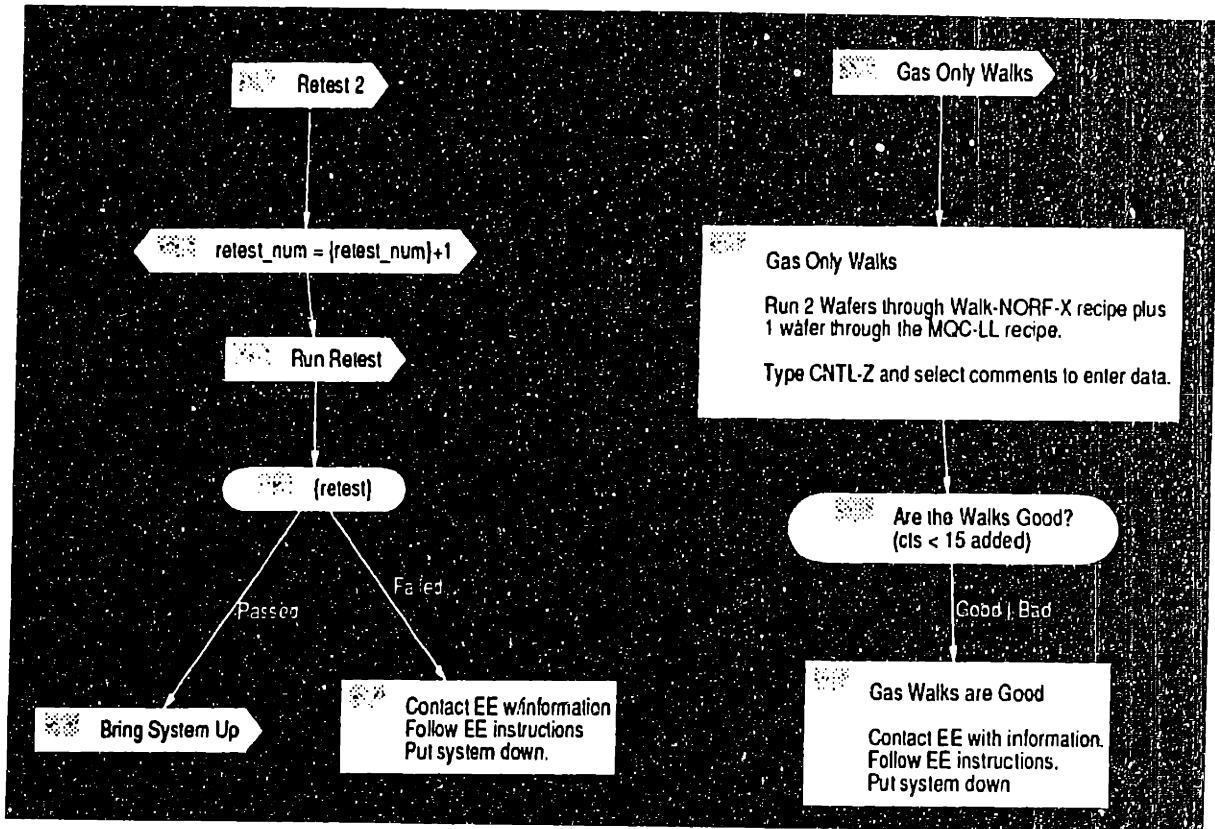
Walks/LL Tests Subtree



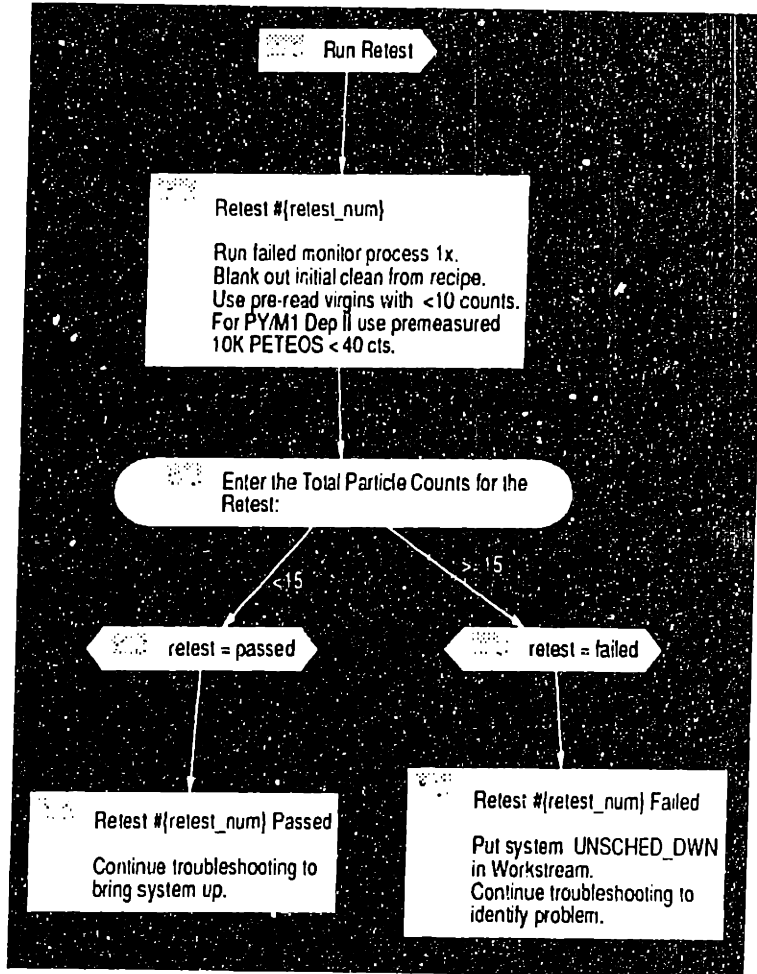
Chamber MQC Tests Subtree



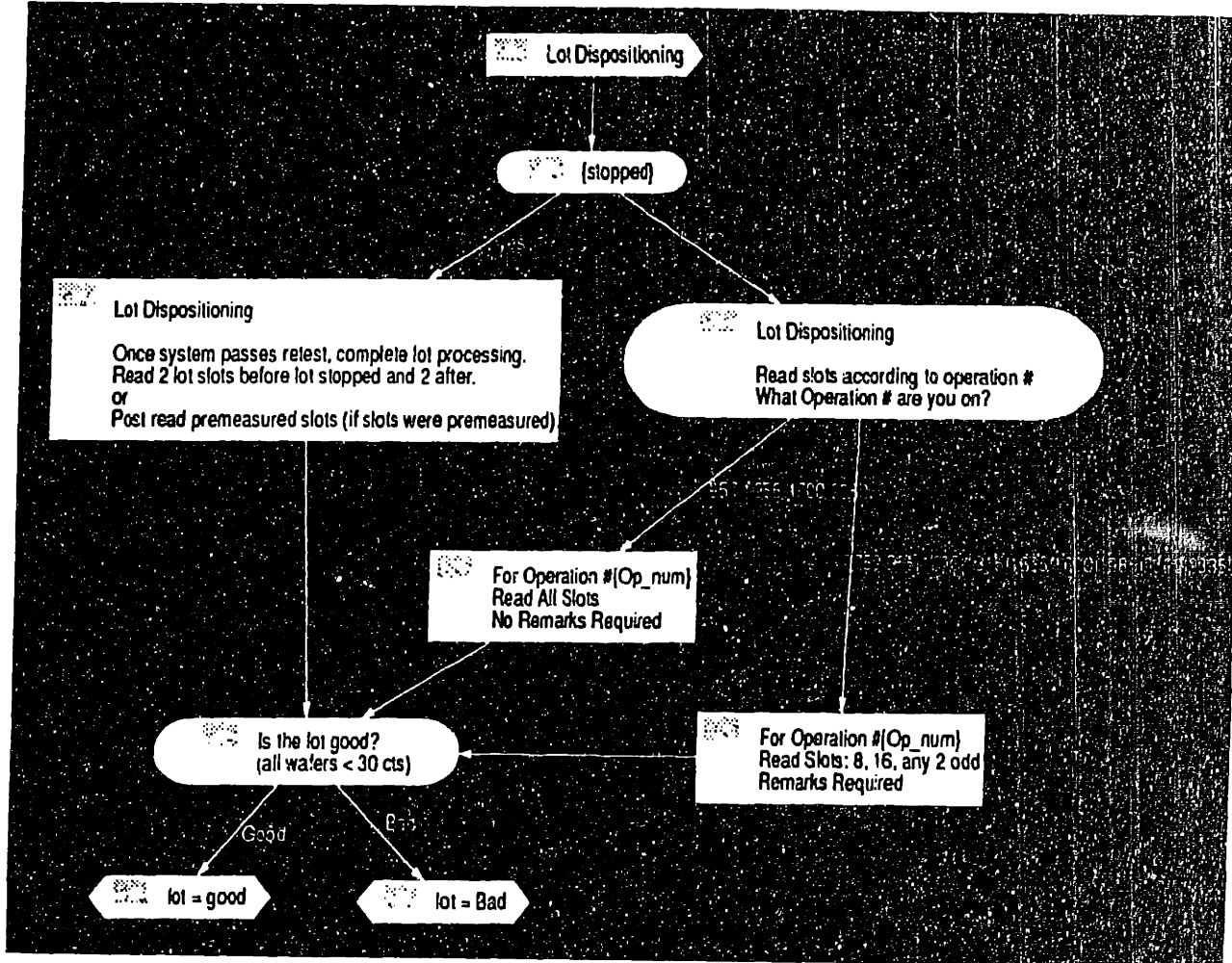
Retest 2 and Gas Only Walks Subtrees



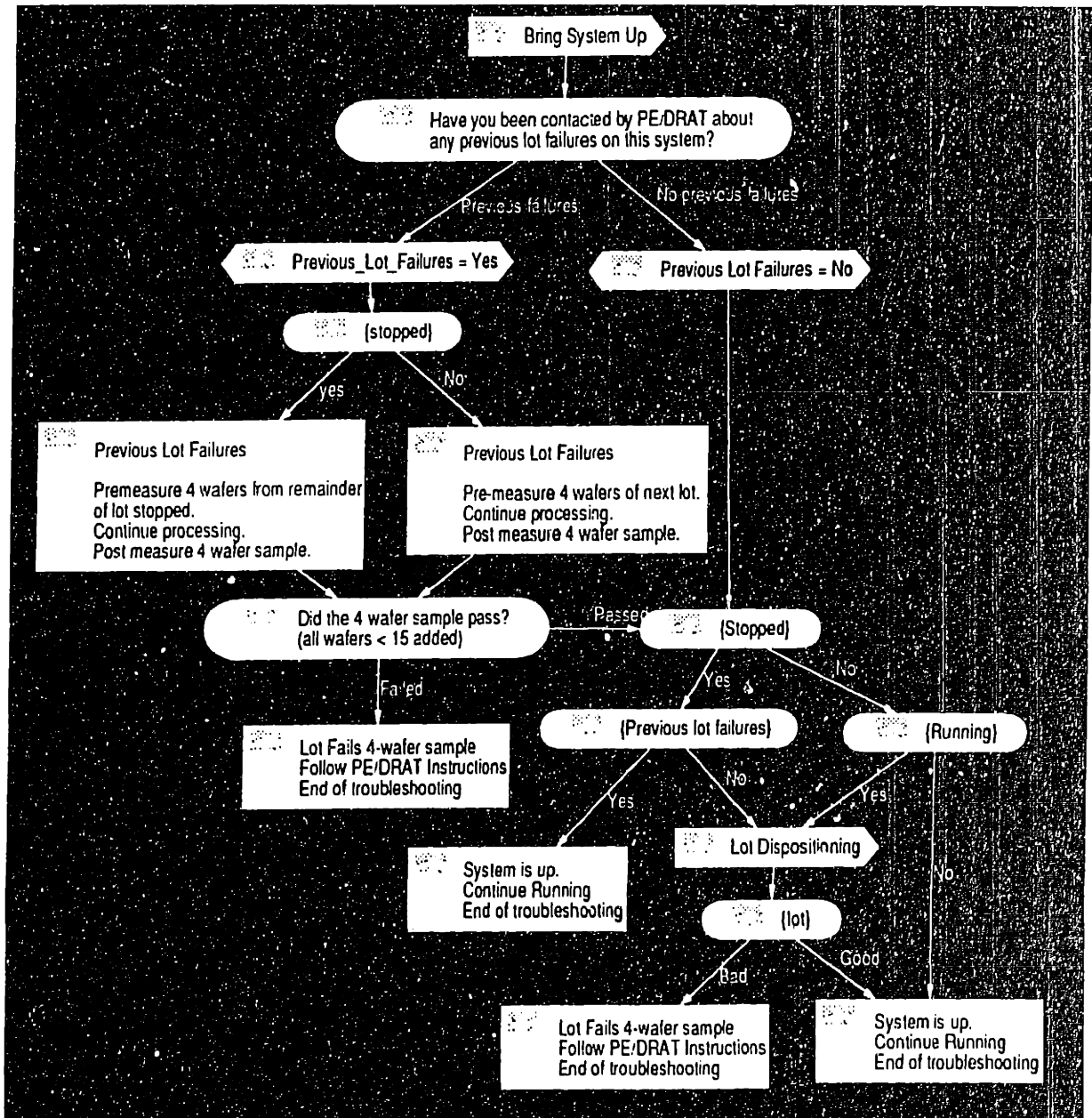
Run Retest Subtree



Lot Dispositioning Subtree



Bring System Up Subtree



Appendix E: Research Methods and Sources of Data

The primary source of data used in this thesis were interviews and direct observation of the Fab 4 members made during my six month LFM internship. My company supervisor was a process engineering supervisor of one of the technology areas. I was provided with office space located in the technical group area and given access, similar to that of a process engineer, to the Fab 4 computer information systems and to the fab manufacturing floor.

The following sections describe the Fab 4 organization and sources of data in more detail to provide the reader with a better perspective of my research methods and data sources.

Organizational Structure

The Fab 4 organization is divided into two primary parts: a production group and a technical group. Both of these groups report to the Fab 4 manager. Traditional support groups such as purchasing, human resources, etc. are administered locally to the Fab 4 organization, but report to other organizations and not the Fab 4 manager.

Production Group

The organizational chart for the production group is shown in Figure E-1. The fab is divided into two sections. Each section contains the operations of two technology areas. Fab 4 operates seven days a week, 24 hours a day. There are four (A/B/C/D) twelve hour shifts, two at the beginning of the week and two at the end of the week. This schedule is referred to as 7x12.

The production group is responsible for implementing the production schedule. The production technicians or PIMT's (production intensive manufacturing technicians) as they are referred to work in the fab processing wafers. The work at each successive level becomes increasingly administrative. The majority of management positions have been

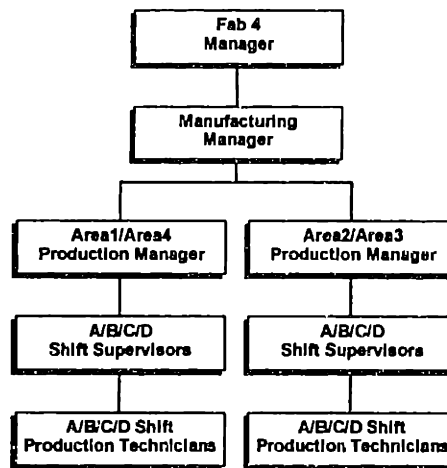


Figure E-1: Organizational Chart for Production Group of Fab 4

filled by individuals who originally started out on the production floor.

Technical Group

The technical group consists of engineers and technicians that support the Fab 4 operation. The technical group is organized into four technology areas (i.e., photolithography, etc.) and two support groups, DRAT (defect reduction) and metrology (measurement tools). The technical group organizational chart is shown in Figure E-2. Each technology area is further divided into equipment engineering (EE) and process engineering (PE). The EE groups are responsible for troubleshooting and making repairs to the fabrication tools in the fab. The PE groups troubleshoot and solve process related issues. Both groups are responsible for continuous improvement of yield, cycle time, and cost.

The EE and PE subgroups support the 7x12 production work week by having sustaining engineers and technicians who also work the 7x12 shifts. They are responsible for supporting all of the tool groups within their technology area. However, some specialization does occur within the EE group.

The sustaining engineers are supported by engineers and technicians who work a normal work week (5x8). The members of the 5x8 groups are assigned to particular tool groups

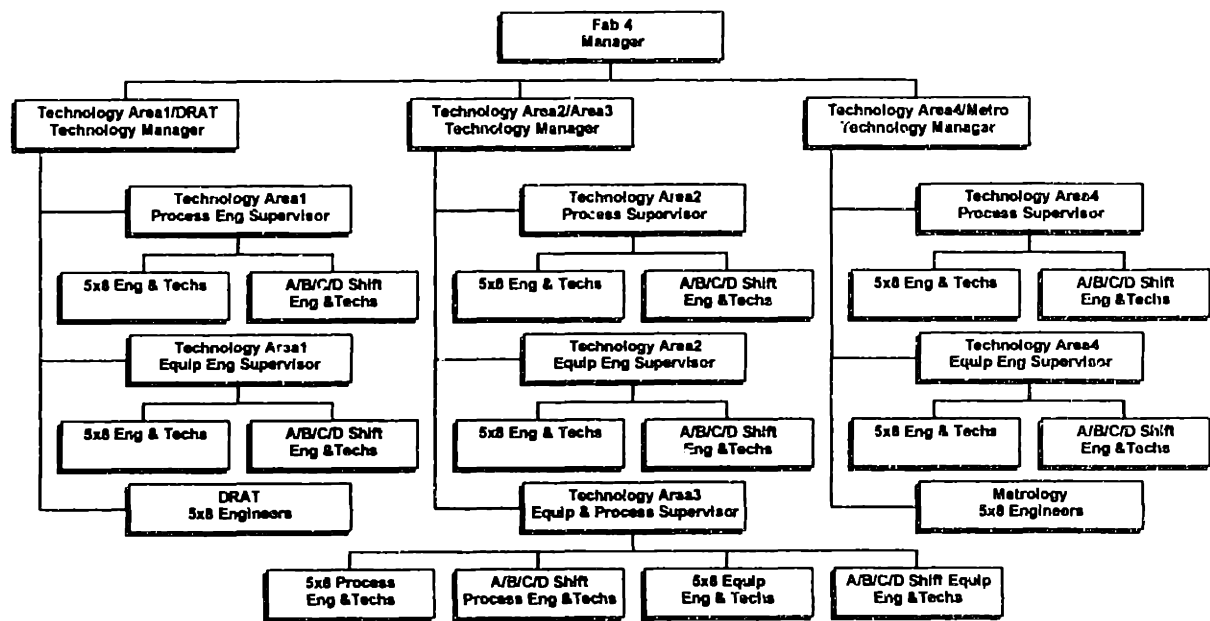


Figure E-2: Organizational Chart for Technical Group of Fab 4

within their technology area.

Training

During the first week of the internship, I went through the same new employee training program that newly hired PIMT's go through. This included fab protocol, safety, and use of the computer system.

Additionally, I attended a course called “Technology Overview”, which went over the basics of the Fab 4 CMOS process. The course was structured by technology area. A four hour presentation was given by a member of each technology area. The course is aimed at newly hired engineers, technicians, and senior PIMT’s.

Interviews

Shortly after arriving on site, I scheduled a series of thirty minute interviews with all of the PE technology area supervisors and the two production area managers. The line of questioning was intended to identify examples of troubleshooting knowledge transfer methods currently in use in Fab 4, their concerns about such a project, and opportunities within their areas for piloting such a project.

These initial interviews provided me with a several leads to follow-up on regarding current methods of knowledge transfer in use in Fab 4. A second round of 30 minute interviews were scheduled with the individuals involved in creating and implementing these methods. These people were primarily PE’s. Also, two of the production shift supervisors were interviewed to learn more about how these methods were working in the fab and to get a recommendation of PIMT’s in their areas I could spend time with.

Opportunities for Observation

I took advantage of many opportunities to observe the Fab 4 personnel at work. These opportunities included attending a series of 15 minute meetings held each morning. The first meeting was a shift pass-down meeting, this was followed by a production meeting where the production and technology managers discussed important issues from the morning pass-down meetings. The issues discussed were frequently related to the troubleshooting problems examined in this thesis.

I spent time in the fab shadowing PIMT’s in several different technology areas. I also spent time in the fab with a PE technician. This provided me with opportunity to observe the demands on their time as well as the work culture.

The regularly scheduled company and divisional communication meetings provided an opportunity to get a broader perspective of DS than just Fab 4.

Implementing

As part of this project, I worked with several individuals and groups to develop the TSGuide system. The case studies presented in chapter 5 are a direct result of this work. Other troubleshooting guides in the knowledge acquisition stage were also worked on providing additional information for this thesis.

Toward the end of the internship, the PEP team was formed and I served as a resource to the group. This allowed me to get a better feel for the entire technical group since members consisted of each of the technology areas.

Bibliography

- ¹ Schein, Edgar H., "How Can Organizations Learn Faster? The Challenge of Entering the Green Room," Sloan Management Review, Winter 1993
- ² Schein, Edgar H., "Organizational Culture and Leadership", 2nd Edition, Jossey-Bass Publishers, San Francisco
- ³ Simons, Robert, "Control in an Age of Empowerment", Harvard Business Review, March-April 1995
- ⁴ Behnke, Lawrence S., et al, "The Evolution of Employee Empowerment", IEEE Transactions on Semiconductor Manufacturing, Vol. 6, No. 2, May 1993
- ⁵ Bohn, Roger E., "Measuring and Managing Technological Knowledge", Sloan Management Review, Fall 1994
- ⁶ D. Waterman, "A Guide to Expert Systems", Addison-Wesley, 1986
- ⁷ Smith, P. *et al.*, "Expert Systems in Manufacturing in Europe," Moving Towards Expert Systems Globally in the 21st Century, Cognizant Communication Corp., 1994
- ⁸ Marir, Farhi, and Watson, Ian, "Can CBR imitate human intelligence and are such systems easy to design and maintain? A critique.," Progress in Case-Based Reasoning, First United Kingdom Workshop, Salford, UK, January 1995, Proceedings
- ⁹ Gupta, Uma G., and Brown, Carol E., "Case-based Reasoning and the Accounting Domain," Moving Towards Expert Systems Globally in the 21st Century, Cognizant Communication Corp., 1994
- ¹⁰ DECtree User's Guide, Version 1.2, June 1992, Digital Equipment Corporation
- ¹¹ Barrett, Todd S., "Leveraging Process Data to Reduce Manufacturing Costs", M.I.T. Leaders for Manufacturing Thesis, 1996
- ¹² Wolf, S. and Tauber, R.N., Silicon Processing for the VLSI Era, Volume 1 --- Process Technology," Lattice Press, 1986
- ¹³ Medsker, Larry, et al, "Knowledge Acquisition from Multiple Experts: Problems and Issues", Moving Towards Expert Systems Globally in the 21st Century, Cognizant Communication Corp., 1994
- ¹⁴ Greenwood, Susanne, and Nealon, John L., "A comparative Evaluation of Knowledge Elicitation Techniques for Expert Systems", Moving Towards Expert Systems Globally in the 21st Century, Cognizant Communication Corp., 1994
- ¹⁵ McAlister, M.J., et al, "What Dictates Knowledge Acquisition, and Can Changes be Made?", Moving Towards Expert Systems Globally in the 21st Century, Cognizant Communication Corp., 1994