# MONITORING, MAINTENANCE AND DIAGNOSIS IN A COMPUTER-INTEGRATED ENVIRONMENT FOR SEMICONDUCTOR MANUFACTURING

by

Norman H. Chang

Memorandum No. UCB/ERL M90/61

18 July 1990

# MONITORING, MAINTENANCE AND DIAGNOSIS IN A COMPUTER-INTEGRATED ENVIRONMENT FOR SEMICONDUCTOR MANUFACTURING

by

Norman H. Chang

Memorandum No. UCB/ERL M90/61

18 July 1990

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

# MONITORING, MAINTENANCE AND DIAGNOSIS IN A COMPUTER-INTEGRATED ENVIRONMENT FOR SEMICONDUCTOR MANUFACTURING

by

Norman H. Chang

Memorandum No. UCB/ERL M90/61

18 July 1990

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

# MONITORING, MAINTENANCE AND DIAGNOSIS IN A COMPUTER-INTEGRATED ENVIRONMENT FOR SEMICONDUCTOR MANUFACTURING

Norman H. Chang

Ph.D. Dissertation

Department of Electrical Engineering and Computer Sciences

Signature:____________________

Committee Chairman

## ABSTRACT

Thanks to recent advances in data acquisition, storage and processing, we are now in a position to dramatically improve the operation of semiconductor manufacturing equipment. This dissertation presents the development of the Berkeley computer-aided manufacturing (BCAM) framework and its application for monitoring, maintenance and diagnosis of semiconductor manufacturing equipment.

Monitoring improves process quality by providing real-time access to critical variables. Statistical abstractions of the real-time monitoring data are stored in the relational database for trend and correlation analysis.

An equipment maintenance record keeping system combines a form-based user interface with a relational database to record preventive maintenance (PM) and equipment repair events as they occur. Storing equipment PM and failure information in an

organized database has several benefits: Accumulated information is automatically indexed to aid diagnosis of failures as they occur by quickly producing a history of similar failures. Equipment failure information is available to other utility programs for display and statistical analysis. Charts to summarize equipment downtime and frequent failures are easily produced. This application has resulted in significant improvement in the way information is used for the management of preventive maintenance and equipment repairs.

The diagnostic system employs evidential reasoning to conduct malfunction diagnosis by combining different sources of evidence originating from maintenance information, online sensor data and statistically filtered inline measurement information. The system is capable of handling two types of evidence: qualitative ones, originating from special patterns of abnormal machine behavior. Also, quantitative ones, that originate from the violation of numerical constraints. These constraints are derived from physical, empirical or semi-empirical equipment models, specifically created and characterized through experimentation. The violation of each of these constraints gives rise to continuously varying belief functions that are used to infer the specific type of equipment failure.

The functions of generic equipment monitoring, maintenance and diagnosis have been built within BCAM's object-oriented programming environment. This environment enables applications to be integrated and shared by many pieces of equipment.

In this dissertation we have chosen the Low Pressure Chemical Vapor Deposition (LPCVD) operation and the Tylan diffusion furnaces as a test vehicle for the development of the above mentioned methodologies.

# ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my research advisors, Professors David A. Hodges and Costas J. Spanos for their inspiration, support and guidance throughout this study. I also thank Profs. N. Cheung and S. Adiga for serving in my dissertation committee. I am grateful to Dr. Chi-Yung Fu for his technical advice and help at the early stages of this research.

I would like to acknowledge A. Gaduh, H. Guo, D. Mudie and S. Miles for their contribution on the generic monitoring system. Special thanks to P. Byrne for offering us the original version of *LAMlink* and many of his constructive comments.

Special thanks to D. Mudie for his work on the co-development of the equipment maintenance record keeping system, and to G. May for the compilation of the plasma etcher knowledge. I would also like to acknowledge the staff of Berkeley Microelectronics Laboratory for their contribution of equipment maintenance knowledge, and especially, B. Hamilton and K. Voros for supporting this project.

Several equipment experts have offered their knowledge and help. Specifically, P. Rothe of Harris Semiconductor, T. Booth and J. Huang from the Berkeley Microlab. I am also grateful to K.K. Lin for supplying the quantitative equipment models of the Tylan furnace.

Thanks to Prof. L. Rowe, Mr. B. Sindahl and the Berkeley CIM students K.K. Lin, G. May, C. Hegarty, C. Williams, B. Smith, D. Mudie, H. Guo, S. Lee, E. Boskin, and also to Dr. Z-M. Ling for many informative and stimulating conversations. Special thanks to E. Boskin, G. May, C. Hegarty, S. Lee, and D. Mudie for proof-reading parts of this dissertation.

Finally, I am grateful to my wife, Lina and my parents for their encouragement and support over the past years.

This work was sponsored by NSF under grant MIP-8909095.

*To Lina*

# Table of Contents

# *Chapter 1*
# *Introduction*

## 1.1. Background and Motivation

The art of semiconductor fabrication has advanced rapidly in the recent years. Powerful CAD tools have greatly shorten the design cycle of integrated circuits compared to just a few years ago. Unfortunately, not the same level of attention has been given to semiconductor manufacturing itself, particularly in the quality and productivity aspects [1].

Quality and productivity are intimatedly linked in semiconductor manufacturing. The final yield of salable product depends strongly on tight process control through several hundred steps in a manufacturing process. This yield can be improved by capturing and using five important data sets. They are:

(1) production lot history (work in process, or *WIP*) data

(2) "inline" physical and electrical measurements collected between process steps

(3) final electrical test data

(4) equipment maintenance records

(5) real-time process monitoring data collected during process steps

The first three of these data sets have been routinely collected in IC manufacturing in order to detect process problems. However, because of very tight process specifications, a large (and often highly variable) fraction of finished chips fail the final functional test *even when all inline measurements are within specified control limits*. The fact is that today it is impractical, costly, or impossible to make enough inline measurements to detect all possible process failures. When process failures do not become evident until the final test, the specific cause of yield drop may be hard to identify before a large amount of product gets damaged. This problem must be addressed and overcome.

The key to early detection of process problems is to use the data sets (4) and (5) through the collection of maintenance information and the real-time monitoring of critical variables during the process. The proper management of maintenance data could help forecast equipment malfunctions. Similarly, well-organized real-time monitoring data will facilitate the timely detection of process and equipment problems.

Today, in most U.S. plants maintenance data is collected in an ad-hoc manner - usually in a lengthy text report, while monitoring data is not collected at all. Most of the equipment in use is equipped with a CRT that displays real-time sensor data in alphanumeric form. Unfortunately operators cannot effectively monitor more than a small fraction of the critical parameters, and sensor data is usually not saved for later analysis.

In this dissertation we will describe a framework to address equipment operation problems. Within the Berkeley computer-aided manufacturing (BCAM) framework, real-time monitoring data is combined with inline measurements and maintenance information in a logically integrated database. Our objective is to investigate new techniques that improve equipment operation by the timely diagnosis of process and equipment problems.

## 1.2. Thesis Organization

The subject of this dissertation is the construction of a computer-aided manufacturing framework and its application to equipment monitoring, maintenance and diagnosis for semiconductor manufacturing. The Berkeley CAM framework and its relationship to the computer-integrated manufacturing (CIM) architecture is described in Chapter 2. Chapter 3 examines the requirements for an equipment monitoring system and our experience with an early prototype of the system. A new scheme for equipment maintenance information management is presented in Chapter 4. In Chapter 5 we describe an off-line equipment maintenance system that aims to provide step-by-step guidance for lead operators and technicians to perform off-line maintenance. Chapter 6 discusses continuous equipment diagnosis based on evidential reasoning. Conclusions and future work are presented in Chapter 7.

## References for Chapter 1

[1] D.A. Hodges and L.A. Rowe, C.J. Spanos, *Computer Integrated Manufacturing,* Int. Electronic Manufacturing Technology Symp., September 1989.

# Chapter 2
# The Berkeley Computer Aided Manufacturing Framework

## 2.1. Introduction

The Berkeley computer-aided manufacturing (BCAM) System aims to support all aspects of equipment operation and process control in semiconductor manufacturing. BCAM is a part of a computer-integrated manufacturing (CIM) project, whose objective is to develop and implement a prototype of the semiconductor manufacturing plant of the future [1]. In this Chapter, Section 2.2 discusses the evolution of the BCAM system. The overview of the BCAM framework is illustrated in Section 2.3. Section 2.4 describes the Berkeley CIM system. The relationship between Berkeley CIM and BCAM is finally discussed in Section 2.5.

## 2.2. Evolution of the BCAM Architecture

A precursor of BCAM was the Berkeley Intelligent Processing System (BIPS) [2]. BIPS, completed in 1987, aimed to advance semiconductor manufacturing by applying a combination of expert system technology with traditional quantitative tools towards the design and control of integrated-circuit fabrication processes. BIPS was intended to simplify the process recipe writing task and to provide tools for tighter process monitoring. BIPS was built in ART [3], a commercial expert system shell, which provides various knowledge representation paradigms, multiple inference methods, and primitives to construct a friendly user-interface.

Several lessons were learned from our experience with BIPS. The major lesson is BIPS's difficulty in system integration. We found that ART is a very good tool for quick prototyping of an application but it is not suitable for practical deployment of a CAM system. The reason is that we would like to operate in a distributed, multi-tasking workstation environment. In this case, the preferred user-interface would be an industry standard such as the X window system.

Another lesson is that the recipe generation is a very complex operation. BIPS was limited in that respect by its simple heuristic algorithm. It is now recognized that a statistically-based equipment model must be employed in order to create a successful recipe [4]. A third lesson is that Boolean diagnosis, such as that used in BIPS, can be quite unstable in a noisy manufacturing environment, and it cannot provide a measure of the seriousness of a fault.

Because of these limitations in BIPS, BCAM, a ''second'' generation CAM system, was designed by the CIM group in Berkeley. In BCAM many generic tools have been developed and work together to streamline equipment operation. An overview of BCAM is given below.

## 2.3. The BCAM Framework

BCAM [5] uses inline, maintenance and real-time monitoring data that are being collected and stored in an integrated relational database. Six functions that contribute to the profitable operation of manufacturing equipment have been identified and implemented. These are real time monitoring, statistical process control (SPC), equipment maintenance record keeping, fault diagnosis, the efficient development of new recipes, and the development and maintenance of equipment models. It is evident that each of these capabilities must be specialized for each of a multitude of equipment in a cleanroom facility (Figure 2.1).

The six functions are also tightly coupled. For example, time plots of specific parameters can be displayed when the online diagnostic program suspects that the process is out of control. Similarly, equipment models and SPC procedures are used by the diagnóstic system. These capabilities also share many basic, reusable primitives such as numerical optimizers, reasoning agents and statistical routines. Obviously, the proper integration of these functions will benefit the efficient development of the system and its effectiveness within a greater CIM architecture.

### 2.3.1. Object-Oriented Integration

To optimize its basic primitives, BCAM is being built using the object-oriented features of the CLOS (Common Lisp Object System) [6] and C++ [7] programming languages. Under each BCAM function, specific equipment application inherits the knowledge and functionalities from a generic equipment object. For example, a furnace cluster consists of 16 tubes used for different purposes such as dry or wet oxidation, nitride deposition and low-pressure chemical vapor deposition (LPCVD). Although these reactors serve different purposes, they have many elements in common. For instance, they use the same proportional-integration-derivative (PID) control algorithm for temperature control as well as the same pump system. Therefore, knowledge that is common to all these reactors is made available to all the application programs that have been written for the generic reactor. Specific application programs use special knowledge for each reactor.

In the same manner, generic functionalities are shared by all the applications. Functionalities such as output evaluation, sensitivity analysis of generic equipment models, etc. are inherited by all the specific equipment models of different reactors [8].

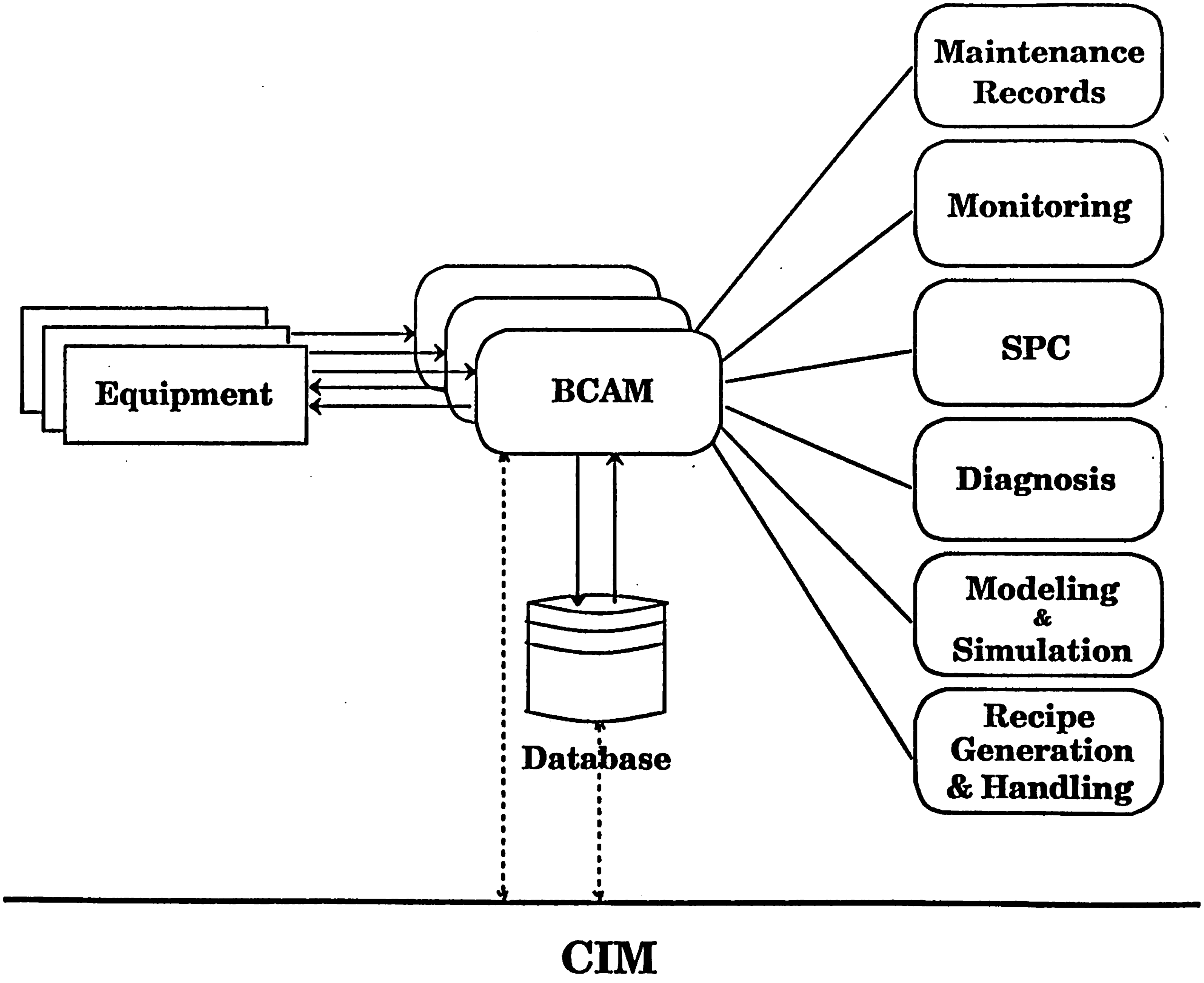


Fig. 2.1 The framework of the Berkeley CAM system.

The benefit of organizing the knowledge and functionalities in an object-oriented structure is three-fold. First, knowledge and functionalities are reusable and are not redundantly stored. Secondly, the object-oriented structure is modular and easily maintained. Because knowledge and functionalities are explicitly defined for each equipment cluster, it is easy to update them. Finally, clearly defined *external* and *internal* functions make the interface between two incompatible systems easier [9]. For example, diagnostic modules built in CLOS can call *external* functions provided by an equipment model library written in C++.

### 2.3.2. Implementation

BCAM is a workstation-based CAM system expressly built to allow us to experiment with the ideas mentioned above. Parts of BCAM have been written in CLOS, C, and C++. X window primitives are used by the monitoring, SPC, and diagnostic modules. The equipment maintenance module combines a form-based interface with a relational database to record preventive maintenance and field repair events. All modules use INGRES, a commercial relational database system, as a central data depository. Direct communication with semiconductor manufacturing equipment is accomplished through the Semiconductor Equipment Communication Standard (SECSII) protocol [10].

BCAM is also designed to support inter-equipment control in workcell configurations [11]. Other packages such as numerical optimization routines, process simulation tools (e.g. SUPREM, SAMPLE, SIMPL) and the statistical package RS/1, are also available as external resources for our environment.

In the next section we briefly describe the Berkeley CIM architecture and then discuss how BCAM fits in this bigger picture.

## 2.4. The Berkeley CIM Architecture

As pointed out by Hodges [1], current CIM architectures are costly and complicated due to the limitations of early-generation hardware, operating systems, local-area networks, programming languages, and database systems. To provide the needed data processing capability, these systems typically consists of 4 to 6 hierarchical levels, as illustrated in Figure 2.2. The problems of today's CIM architectures are summarized below.

(1) There is no easy way to achieve a logically integrated database for all types of manufacturing data.

(2) Application programs cannot communicate across different hardware, operating systems and database systems.

(3) The large number of different user interfaces limits productivity.

(4) Global process monitoring across a heterogeneous environment is too tedious.

A two-level CIM architecture is designed to address the problems above.

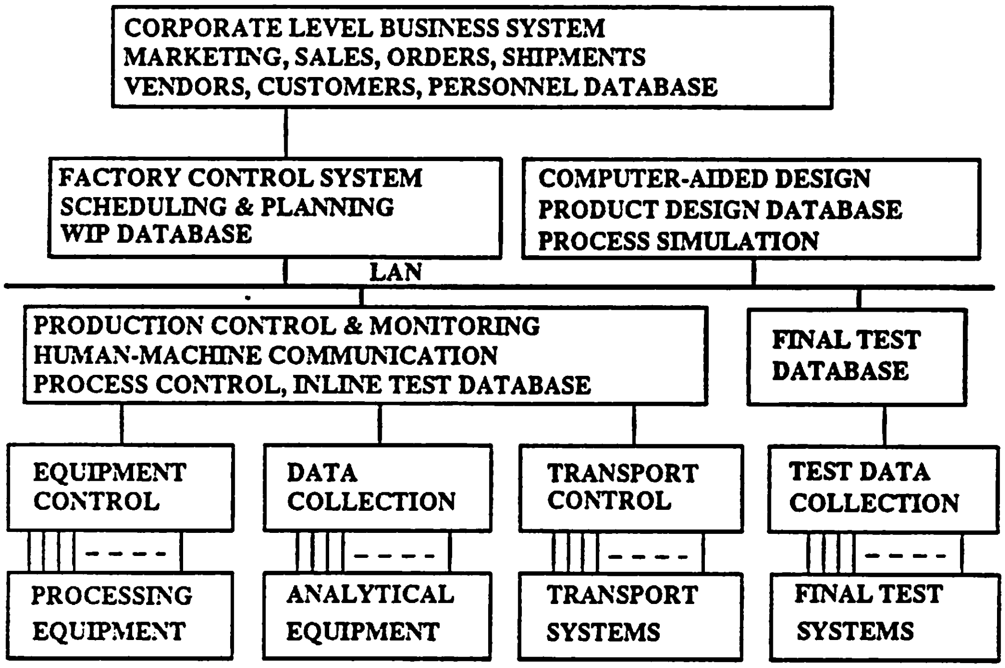


Fig. 2.2 First-generation CIM architecture

With current computer technologies, it is possible to construct a physically distributed but logically integrated database. This will greatly facilitate data manipulation across the manufacturing floor and will lead to high productivity. Four recent advances contribute to this realization. The most important advance was the development of relational database systems that reduce the effort required for both the initial development and the subsequent maintenance and modification of a system. This is because relational databases support easy-to-use interfaces that allow end-users (in this case process, maintenance and yield engineers) to manipulate the information stored in the database.

The second major development has been the industry-wide acceptance of high-bandwidth communication standards (local-area networks, or LANs) for linking systems from different vendors in a cost-effective way. LANs make it possible to connect process control applications directly to the fabrication equipment. Consequently, inline and in-process measurements can be monitored by computers in real time.

A third important development is the emergence of distributed database management systems. Thanks to distributed database systems, information is physically stored in many nodes, yet it appears to the user as a coherent entity. The distributed database system determines where the data is located, generates an efficient plan to retrieve or update it, and ensures the consistency and integrity of the data.

The fourth important element is the spreading use of AI technologies. Many knowledge intensive, error-prone activities in semiconductor manufacturing can be automated by the use of AI techniques. The need for automated decision making in planing, scheduling, diagnosis, maintenance, etc., becomes even more pressing in view of the complexities of the new sub-micron ULSI processes.

Thus, thanks to relational databases, efficient communication, distributed data management and AI technology, it is now possible to propose a CIM architecture that is dramatically simpler and more flexible than those used in earlier systems. The proposed architecture consists of two levels, as shown in Figure 2.3. The lower level in this architecture includes the embedded controllers that provide real-time control of metrology and processing equipment. Personal computers used as equipment controllers or for other related tasks belong to this level as well.

The second level comprises a distributed network of multi-tasking workstations, file and compute servers linked to a common distributed relational DBMS. A high-speed LAN (10 to 100 Mb/s) provides communications among processors. Some computers will be located in process areas and will function as "workcell" controllers. Workcell controllers are linked directly to embedded equipment controllers, using SECSII or another suitable protocol. Workcell controllers provide the feedback/feed-forward control, recipe and process handling among equipment, in addition to the basic single-equipment control functions. A multi-tasking workstation will be able to handle 5 to 10 major items of process equipment.

Other computers at this same logical level may be devoted to support archival file systems, production planning and scheduling aids. With a distributed relational database, users issue the same commands to enter or query data in the database, no matter where the users or the data are located. Menu or icon-based query systems permit engineers to create ad-hoc queries without the help of a database specialist [12].

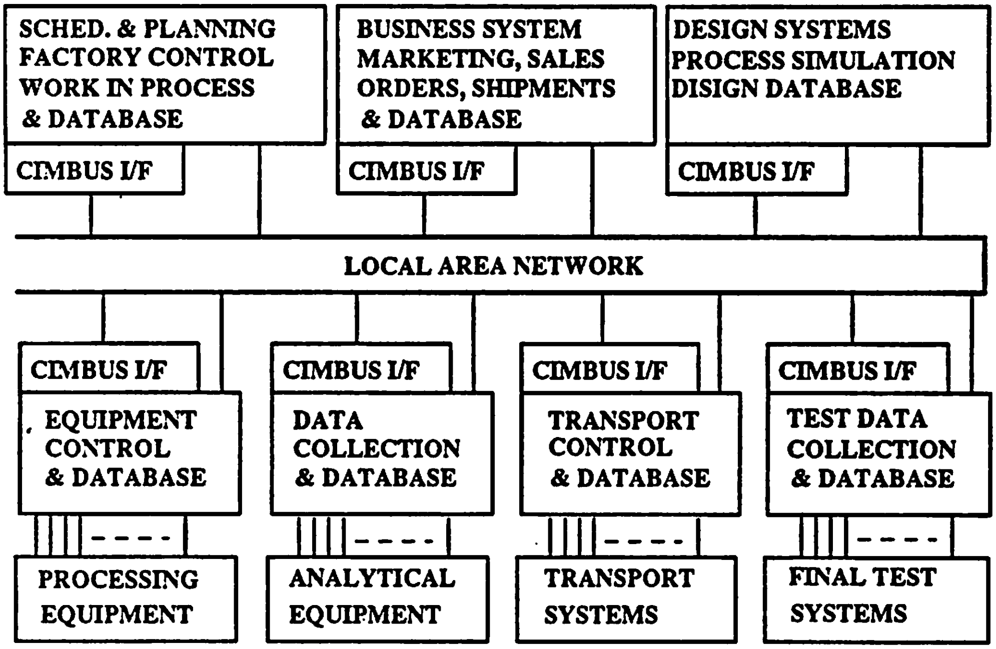


Fig. 2.3 Two-level architecture for CIM

The objective of the Berkeley CIM architecture is to develop software modules for controlling VLSI processing steps, and to demonstrate a flexible architecture for combining these modules into an integrated CIM system based on this model. A unified approach is taken to manage data involved in product and process design, control of equipment, processes and facilities, quality assurance and testing, and production planning. Some issues of special importance to the Berkeley CIM architecture are described below.

### 2.4.1. CIMBUS

The computer resources used in the IC manufacturing industry are *heterogeneous*. There are different programs working in different operating systems, different user-interface methods, different database systems. If we are going to interchange information among N applications on different systems, N x (N - 1) interfaces need to be developed. This task is overwhelming in a stable environment and impossible in a rapidly changing environment such as semiconductor manufacturing.

One solution to this problem is the creation of a computer integrated manufacturing bus (CIMBUS) that acts as a coordinator among different applications. A major function of CIMBUS is to change a data format from the source application to an intermediate format and then change it to another data format that is used by the target application. Each application requires one interface to this intermediate format and therefore we now only need to build N x 1 interfaces instead of N x (N - 1). CIMBUS is also a communication bus that provides utilities to broadcast, receive, and send messages among applications [13].

### 2.4.2. Planning and Scheduling

Another important application is company-wide production planning, along with factory floor scheduling for the fabrication, assembly, and test of semiconductor products. The goal is to develop a modular set of application programs utilizing linear programming techniques to make automated planning calculations which optimize corporate cash flow [14]. In the area of factory floor scheduling, two broad problems have been examined: (i) How should global information be summarized and used for real time resource allocation within the fab? and (ii) How much improvement in manufacturing system performance can be expected from the use of global up-to-date information instead of old or local data? Research on production planning and scheduling is complemented by an effort to improve the simulation models that describe these activities.

### 2.4.3. Berkeley Process-Flow Language

The Berkeley Process-Flow Language (BPFL) is used to describe microelectronics manufacturing processes [15]. According to BPFL, there are multiple application-specific *views* of one process. Each view is derived from a BPFL specification with an application-specific interpreter. The primary views are *simulation* and *fabrication*, describing how to simulate and implement a process-flow. One goal of BPFL is to derive the multiple views from the same basic information, in order to eliminate errors introduced by having redundant specifications of the same process in different notations.

BPFL will be the input language for the PROSE process simulation system [16] and a work-in-progress (WIP) system [17]. Information from manufacturing can be used to simulate actual steps using the Profile Interchange Format [18].

### 2.4.4. Work in Progress

The work-in-progress (WIP) interpreter executes process specifications, handles equipment allocation and control, and collects and stores data used to monitor the performance of a manufacturing facility and its products. The function of the WIP system is to automate the process of running the fab in order to improve throughput and product quality.

The WIP system uses a BPFL specification to direct and schedule processing steps in a laboratory. All data used by the WIP system, including recipe specifications, are stored in the INGRES database. Equipment scheduling is carried out with a separate scheduling system. In order to optimize equipment utilization, the WIP system groups compatible lots of wafers requiring identical treatment. A heuristic system that works in concert with a production-scheduling system is being developed to automate wafer grouping.

## 2.5. BCAM and the Berkeley CIM Architecture

BCAM operates at the lower level of the CIM architecture. In particular, BCAM receives commands from BPFL/WIP to run, monitor and report a processing step. The workcell controller works with BPFL/WIP to issue the possible recipe modification for adjusting subsequent processing steps (feed-forward) or for fine-tuning a step in question (feedback).

## References for Chapter 2

[1] D.A. Hodges, L.A. Rowe and C.J. Spanos, *Computer Integrated Manufacturing,* Int. Electronic Manufacturing Technology Symp., September 1989.

[2] C.Y. Fu, N.H. Chang and K.-K. Lin, *"Smart" Integrated-Circuit Processing,* IEEE Transactions on Semiconductor Manufacturing, Dec. 1989.

[3] B.D. Clayton, *ART Programming Tutorial,* Inference Corp., 1989, Los Angeles, CA.

[4] K.-K. Lin, J. Huang, and C.J. Spanos, *Statistical Equipment Modeling for VLSI Manufacturing,* Symposium on Automated Semiconductor Manufacturing, 176th Electrochemical Society Meeting, Oct. 1989.

[5] N.H. Chang, C.J. Spanos, *A CAM Framework and its Application for Monitoring and Diagnosis for VLSI Manufacturing,* SRC IC Manufacturing Workshop, Michigan, 1989.

[6] S. Keene, *Object-Oriented Programming in Common Lisp,* Addison-Wesley, 1989.

[7] S.B. Lippman, *C++ Primer,* Addison-Wesley, 1989.

[8] G.S. May, C.J. Spanos, *Private Communication.*

[9] Bertrand Meyer, *Object-Oriented Software Construction,* Prentice Hall Inc., 1988.

[10] *The SECS Message Service for Bidirectional Transfer of SECS Messages: Part 2: Definition of Protocol,* SEMI Communication Committee, MAP/SECS Subnet, Semiconductor Equipment and Materials International, Jan. 1988.

[11] Z.-M. Ling, C.J. Spanos, *Private Communication.*

[12] B. Smith, L.A. Rowe, *FMTool,* SRC IC Manufacturing Workshop, Michigan, 1989.

[13] D.A. Hodges, M. Baudin, *Private Communication.*

[14] R. Leachman, Glassey, S. Adiga, *Private Communication.*

[15] C. Williams, *Berkeley Process-Flow Language,* ERL Research Summary, EECS, University of California, Berkeley, 1990.

[16] A.S. Wang, *An Integrated Graphical Environment for Operating IC Process Simulators,* UC Berkeley ERL, Memo. No. UCB/ERL M89/67, May 1989.

[17] C. Hegarty, *Work-in-Progress Interpreter,* ERL Research Summary, EECS, University of California, Berkeley, 1990.

[18] S. Duvall, *An Interchange Format for Process and Device Simulation,* IEEE Trans. Computer-Aided Design, vol. CAD-7, no. 7, pp. 741-754, July 1988.

# *Chapter 3*
# *A Generic Equipment Monitoring System*

## 3.1. Introduction

An equipment monitoring system is needed in order to improve process control. The standard approach to process control employs physical measurements of critical parameters such as film thicknesses, line widths, and particle counts. These so-called *inline* measurements are made on a sampled basis at intermediate points in the process sequence. For example, after a critical step film thickness or line-width is measured at several locations on one or two wafers in each lot. These measurements are recorded as a function of time and plotted on SPC charts. Control limits are established and corrective action is taken when these limits are exceeded.

The complexity of the manufacturing technology however, has increased dramatically. Over the past decade, the number of distinct process steps in wafer fabrication has roughly tripled, to about 300. The total number of critical process variables has increased even faster, to almost 1000. Many undesirable process aberrations that reduce yield and reliability cannot be detected from the standard inline measurements alone.

Process quality, however, can be improved through real-time monitoring of critical variables during process execution. For example, the consistency of plasma etching of polysilicon films is critically dependent upon gas composition, pressure, and power supplied to the electrodes. Modern plasma-etching equipment has sensors for these and other variables. A built-in microprocessor (known as the *embedded controller*) monitors this information in real time. Most modern semiconductor equipment also displays all the real time information on a built-in CRT. Unfortunately, operators cannot effectively monitor more than a small fraction of the critical parameters, and the ones that are not monitored are usually not saved for later analysis. Direct communication with a host computer can easily solve this problem.

Further, although most equipment is sufficiently automated and capable of performing complex process sequences, the organization and management of control recipes is done in an ad-hoc fashion. Typically, recipe management consists of uploading/downloading using cassette or floppy disk, or direct recipe modification on the panel. Hence, operators have to keep track of the "right" version of recipe for each process run. This is complicated by the fact that recipe editing on the panel is usually awkward and error-prone. As a result many mistakes happen due to using an improper recipe. If a communication link existed between the equipment and host computer, recipe management could then be automated in the host computer, eliminating a serious cause of misprocessing and improving efficiency and productivity.

In this Chapter, we describe a generic equipment monitoring tool that has been designed to address these issues. Section 3.2 describes the client-server architecture of the generic equipment monitoring system. Management of real-time data, process information and statistical abstraction is discussed in Section 3.3. Section 3.4 describes the user interaction with the monitoring system. Requirements of recipe management is outlined in Section 3.5. The interaction between the equipment monitoring system and other systems under the BCAM framework is discussed in Section 3.6. Section 3.7 summarizes some comments from users and Section 3.8 describes the current status of the system. Conclusions are presented in Section 3.9.

### 3.2. Client-Server Architecture

Recently, there has been a tendency to use single-task personal computers to monitor equipment [1]. The drawback is that the real-time data is still not accessible by a host computer (workcell controller or factory controller). This complicates the correlation analysis across data sets from different stages of the process that is often necessary.

In order to facilitate access to a multi-tasking environment, the operation of the generic monitoring system is based upon the client-server model [2]. The SEMI Equipment Communication Standard (SECSII) [3] is a communication protocol for exchanging data and control commands between equipment controllers and host computer systems. A SECSII protocol handling (*SECStalk*) program, adaptable to any equipment that can handle SECSII communication protocol, acts as a *server*. An equipment monitoring (*monitor*) program acts as a *client* that collects relevant process information and sensor data, and produces descriptive real-time graphics of a process run. Both the SECSII handling (*SECStalk*) and monitoring (*monitor*) programs are generic and can easily be adapted to additional equipment. Figure 3.1 shows the client-server structure of the monitoring system.

Several equipment such as Tylan furnaces, Lam etchers, Nanospec ellipsometers (thin-film thickness measurement machine), etc., are connected to the *SECStalk* server in the Berkeley Microlab. The *monitor* program can be initiated on a workstation or on an ASCII terminal to collect, view, and store information into a database. Real-time data plotting is available on bitmapped displays supporting X windows [4].

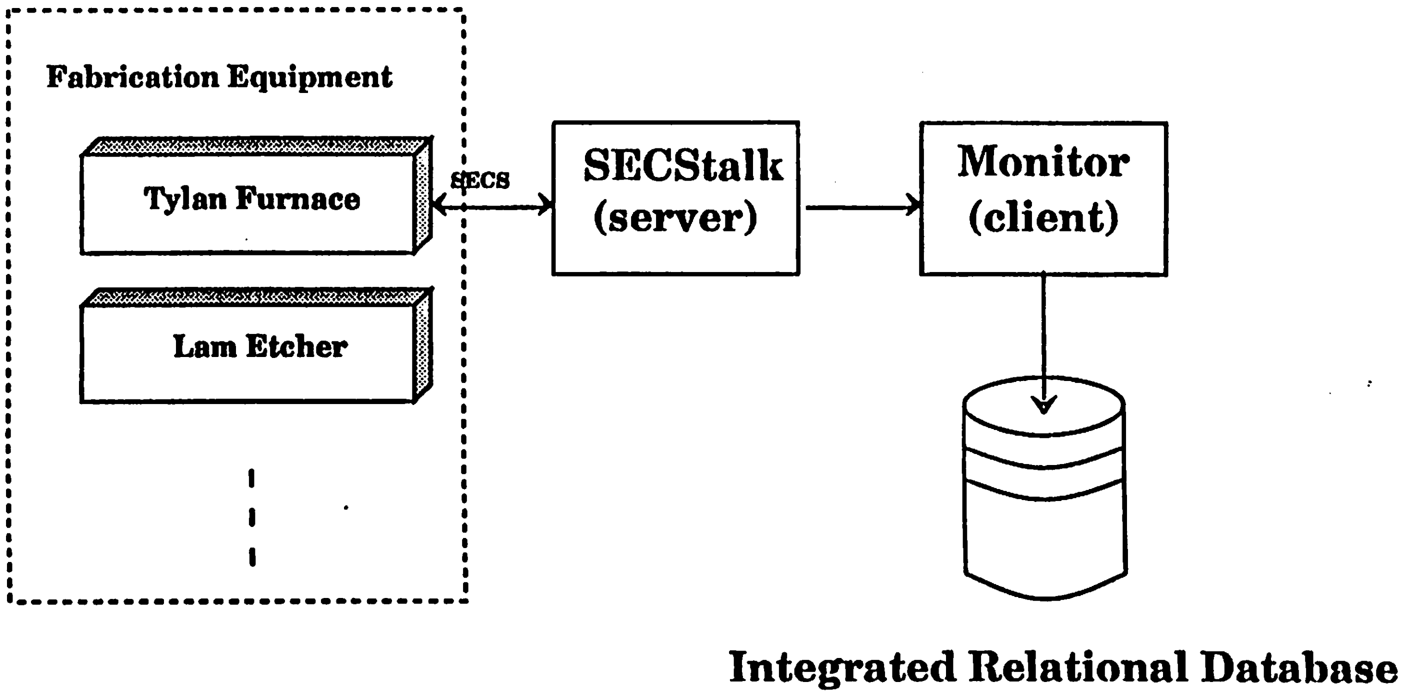


Fig. 3.1 The client-server structure of the generic monitoring system

### 3.3. Management of Real-Time Data, Process Information and Statistical Abstraction

With the collection of real-time monitoring data and process information (e.g. operator, recipe, lot ID) at each process step, data storage requirements will be at least two orders of magnitude greater than before. For example, a typical 3-hour LPCVD process step, monitored at a moderate sampling rate of 3 samples/min with 60 sensor values in each sample, needs approximately 1 Mbyte of storage (compared to inline measurements with 5 thickness measurements per wafer and 20-wafer per lot, that need about 2 Kbytes of storage.) For a typical 300 step process, real-time monitoring data can easily amount to 300 Mbytes per lot. Data management becomes an obvious concern if such a large amount of data is going to be collected. We propose to employ statistical abstractions in order to cope with this problem.

Statistical abstractions consisting of means, variances, and other important process information are stored in the database. With this information in the database, it is very easy to analyze trends and correlations.

For example, a process drift during an LPCVD run can be detected by analyzing the trend of the means of deposition temperature retrieved from the database. Or we can do correlation analysis by comparing the power output efficiency for different deposition temperatures in the same furnace.

The graphics interface for standard trend and correlation analysis will be supported by the monitor program. It is also possible to write standard query language (SQL) [5] "scripts" to support specific user-defined operations, although this is a task usually reserved for database specialists. This way one can pre-define frequently used sets of SQL queries. More queries can always be added to the set by the database specialist.

An innovative approach to data manipulation is to provide a graphics interface for users to "point" and "pick" related objects in order to form a query. This method has been demonstrated in the Berkeley Facility Management Tool (FMTool) [6]. Appendix 3.1 outlines some of the implementation details of the data management module of the monitoring system.

## 3.4. User Interaction with the Monitoring system

### 3.4.1. X-Y and Various Graphing Capabilities

The interactive functions of the monitoring tool are based on the Xgraph and Xmenu facilities [7]. In addition to the X-Y graphing capability, options such as zoom-in, zoom-out, splines, etc. are also available Figure 3.2 shows a display example of real-time monitoring of a Tylan furnace in the Berkeley Microlab. There is also a user specified file for customizing the monitoring functions. (See Appendix 3.3a and 3.3b for the UNIX manual pages of these programs.)

### 3.4.2. Historical Monitoring

A generic monitoring tool must be able to do real-time monitoring as well as "historical" monitoring. "Historical" monitoring is the ability to *playback* old process runs either explicitly or through their statistical summaries. For example, center-zone temperature averages collected for the same recipe on the same equipment can be retrieved from the database with an SQL query and displayed by the "online" program in order to perform a trend analysis.

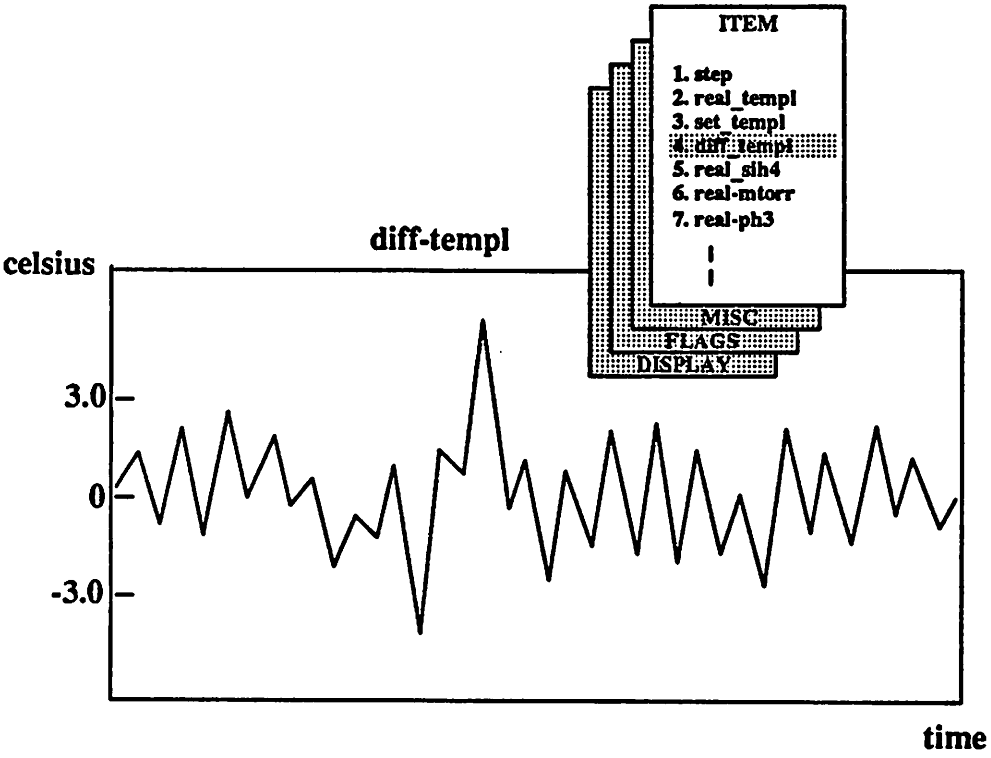


Fig. 3.2 An example of real-time monitoring of a Tylan furnace. Here it displays the difference between the real and set value of load zone temperature during the deposition step. Options can be chosen from the stack menus.

### 3.4.3. Statistical Process Control

Statistical process control (SPC) methods are intended to identify significant process deviation in the presence of the actual manufacturing noise. Such events are attributed to ''assignable causes'', i.e., special reasons that can be identified and corrected. Such events might include material changes, equipment failures, operator error, environmental or procedural changes, etc.

Two common SPC techniques involve the use of the Shewhart chart with supplementary runs rules and the Cumulative Sum (CUSUM) chart [8]. The Shewhart and CUSUM chart techniques for SPC are also used by the diagnostic system to generate a corresponding belief measure depending on the severity of violating the error detection limits.

The SPC module is currently under development. A more detailed discussion on this subject is presented in Chapter 6.

## 3.5. Recipe Management

With the proliferation of multiple recipes in an application-specific integrated circuit (ASIC) production line, recipe management becomes an important issue. Several functions can be supported on a host computer through a SECSII interface. First, before downloading a recipe for a process run, the recipe can be verified by comparing it to the specification in the process flow. Second, the creation of a new recipe or the modification of one can be done with a recipe editor [9]. Lastly, automatic recipe generation can be accomplished through interface to the recipe generator module [10] of BCAM, or as part of the automatic feedback and feed-forward control in workcell configurations [11]. The recipe management module is currently under development.

### 3.6. Interaction with Other BCAM Modules

The generic monitoring tool provides the data to all other BCAM modules. For example, the generic monitoring tool uses the SPC libraries for the combined real-time Shewhart - CUSUM chart as mentioned in Section 3.4.3. When a malfunction is identified by the diagnostic system, the monitoring tool can be used to automatically display the relevant sensor parameters.

### 3.7. User Feedback

The generic monitoring system has been installed in the Berkeley Microelectronics Fabrication Laboratory since March 1990. Students and staff have been using the monitoring system to monitor their processes. They have found the monitoring system very useful for process examination and equipment evaluation. Suggestions for improving the monitoring system have been collected in view of the next release.

### 3.8. Current Status

The current version of the monitoring tool has been based on X/10 window primitives. INGRES, a commercial relational database management system, is used to store the statistical information. The SECSII communication protocol is used to interface between host computers and equipment. Currently supported equipment include all the Tylan furnaces (16 tubes), Lam etchers (2 etchers), Nanospec ellipsometers (optical film thickness measurement), and the Alpha Step (profile film thickness measurement) in the Berkeley microfabrication laboratory. Our goal is to provide data collection capabilities for all the processes in the Berkeley Microlab.

### 3.9. Conclusions

Data collection at the equipment level is regarded as one of the most important factors for the success of sub-micron IC manufacturing [12]. In this Chapter, we have presented several important issues that arise when data collection is fully automated at the manufacturing floor. The issues considered here include the interactive display of real-time monitoring data, the management of the data and their statistical abstractions, as well as the implementation of recipe management along with the necessary statistical process control functions.

## References for Chapter 3

[1] P. Byrne, *Private Communication.*

[2] A.D. Birrell, R. Levin, R.M. Needham and M.D. Schroeder, *Grapevine: An Exercise in Distributed Computing*, Communications of the ACM 25, 4 (April 1982), 260-274.

[3] *The SECS Message Service for Bidirectional Transfer of SECS Messages: Part 2: Definition of Protocol*, SEMI Communication Committee, MAP/SECS Subnet, Semiconductor Equipment and Materials International, Jan. 1988.

[4] O. Jones, *Introduction to the X Window System*, Prentice Hall, 1989.

[5] *INGRES/SQL Reference Manual*, Relational Technology, 1986.

[6] B. Smith, L. Rowe, *Facility Management Tool*, SRC IC Manufacturing Workshop, Michigan, 1989.

[7] D. Harrison, A. Casseto, *Xgraph, Xmenu Interface Primitives*, U.C. Berkeley, 1989.

[8] J.M. Lucas, *Combined Shewhart-CUSUM Quality Control Schemes*, Journal of Quality Technology, Vol. 14, No. 2, April 1982.

[9] C.Y. Fu, N.H. Chang and K.K. Lin, *"Smart" Integrated-Circuit Processing*, IEEE Transactions on Semiconductor Manufacturing, Dec. 1989.

[10] K.-K. Lin, J. Huang, and C.J. Spanos, *Statistical Equipment Modeling for VLSI Manufacturing*, Symposium on Automated Semiconductor Manufacturing, 176th Electrochemical Society Meeting, Oct. 1989.

[11] Z.-M. Ling, *Private Communication.*

[12] D.A. Hodges, L.A. Rowe and C.J. Spanos, *Computer Integrated Manufacturing*, Int. Electronic Manufacturing Technology Symp., September 1989.

# Appendix 3.1

## Data Management Implementation Details

The real-time monitoring information is collected into two files. One file is an *info* file that contains information about the process (e.g. equipment, user, etc.) and a set of statistical abstractions (e.g. average, standard deviation, stabilization time for furnace operation, etc.). The other file is a *data* file that contains the digitized real-time values (e.g. temperature, pressure, etc.) for a process run. The suffix of the *info* and *data* files consists of a unique equipment name, the unique lot ID, and the process monitoring starting time. Each piece of equipment has its own set of analog sensor values that can be monitored. Appendix 3.2a shows the data format in the *data* file for a typical LPCVD process.

After the run is complete, statistical abstractions are derived from the *data* file and stored in the *info* file. For example, the arithmetic average and standard deviation of temperature, gas flow, and pressure during the deposition step are calculated from an LPCVD run. Additional statistical information might also be important for a particular process or a piece of equipment. For example, in a furnace run, the *temperature stabilization time*, i.e., time needed for a furnace reactor to reach its stabilized temperature, carries important information and can be acquired and stored in the *info* file. Appendix 3.2b shows the contents of an *info* file for a typical LPCVD process.

After the process run is completed, the information is moved from the *info* file to the INGRES database. In particular, the *info* file is moved to two tables. The *main* table is identified by the unique recipe name, the equipment type, and an index that points to the corresponding equipment monitoring table. The equipment monitoring table contains the process-related information and the relevant statistical abstractions.

# Appendix 3.2a

## A Typical *Data* file for LPCVD processes

The following parameters are characteristic to an LPCVD process. A typical *data* file contains several hundreds of lines of numerical values for these parameters.

| *PARAMETER* | *UNIT* | *DESCRIPTION* |
|---|---|---|
| step | step | recipe step # |
| set_templ | celsius | set value for load zone temp |
| real-templ | celsius | real value for load zone temp |
| set_tempc | celsius | set value for center zone temp |
| real_tempc | celsius | real value for center zone temp |
| set_temps | celsius | set value for source zone temp |
| real_temps | celsius | real value for source zone temp |
| set_$N_2$ | sccm | set value for $N_2$ gas flow |
| real_$N_2$ | sccm | real value for $N_2$ gas flow |
| set_$SIH_4$ | sccm | set value for $SIH_4$ gas flow |
| real_$SIH_4$ | sccm | real value for $SIH_4$ gas flow |
| set_$PH_3$ | sccm | set value for $PH_3$ gas flow |
| real_$PH_3$ | sccm | real value for $PH_3$ gas flow |
| set_mtorr | mtorr | set value for pressure |
| real_mtorr | mtorr | real value for pressure |

**data.tylan16.22810.Nov.13.1989 (a typical *data* file name)**

```
7010 0070 595.0 595.2 595.0 594.5 595.0 595.0 0.0 0.1 200.0 199.7 0.0 0.0 380.0 384.5
7113 0070 595.0 594.7 595.0 594.7 595.0 595.9 0.0 0.0 200.0 199.7 0.0 0.0 380.0 382.2
7159 0070 595.0 595.7 595.0 595.5 595.0 595.2 0.0 0.0 200.0 200.2 0.0 0.0 380.0 384.2
7211 0070 595.0 596.1 595.0 596.2 595.0 595.6 0.0 0.0 200.0 200.5 0.0 0.0 380.0 381.2
7252 0070 595.0 595.1 595.0 595.7 595.0 595.1 0.0 0.0 200.0 199.7 0.0 0.0 380.0 380.8
7288 0070 595.0 594.1 595.0 594.7 595.0 594.1 0.0 0.0 200.0 199.3 0.0 0.0 380.0 382.8
7329 0070 595.0 593.8 595.0 594.2 595.0 594.1 0.0 0.0 200.0 200.3 0.0 0.0 380.0 383.6
7367 0070 595.0 595.2 595.0 595.2 595.0 594.5 0.0 0.0 200.0 200.5 0.0 0.0 380.0 381.6
7407 0070 595.0 595.5 595.0 595.3 595.0 594.8 0.0 0.0 200.0 200.2 0.0 0.0 380.0 383.6
7455 0070 595.0 594.5 595.0 594.3 595.0 595.8 0.0 0.0 200.0 199.8 0.0 0.0 380.0 380.6
7496 0070 595.0 593.8 595.0 594.0 595.0 594.8 0.0 0.0 200.0 199.5 0.0 0.0 380.0 381.6
7543 0070 595.0 594.2 595.0 594.5 595.0 595.2 0.0 0.0 200.0 199.8 0.0 0.0 380.0 382.6
7596 0070 595.0 595.2 595.0 595.5 595.0 596.2 0.0 0.0 200.0 200.5 0.0 0.0 380.0 379.6
```

# Appendix 3.2b

## A Typical *Info* file for LPCVD processes

The following information of the file are also stored in the main table that identifies the equipment, the generic processing recipe and its associated user-specified parameter values.

| *PARAMETER* | *VALUE* | *UNIT* | *DESCRIPTION* |
|---|---|---|---|
| EQUIP | tylan16 | | equipment name |
| PROCID | FUPLY16C | | process ID |
| $SIH_4$ | 249.9 | sccm | set value for $SIH_4$ gas flow |
| $PH_3$ | .0 | sccm | set value for $PH_3$ gas flow |
| PRESSURE | 550.4 | mtorr | set value for pressure |
| TEMPL | 620.0 | celsius | set value for load zone temp |
| TEMPC | 620.0 | celsius | set value for center zone temp |
| TEMPS | 620.0 | celsius | set value for source zone temp |

The following information is also stored in the monitoring table that identifies the specific monitoring information for the equipment in question.

| *PARAMETER* | *VALUE* | *COMMENT* |
|---|---|---|
| DATE | 12/20/1989 | |
| START-MONITOR-TIME | 11:24:44 | |
| LOTID | 10311 | |
| USER | Norman | |
| STABILIZATION-TIME | 185 | time needed to reach temp stabilization (min) |
| DEPO-TIME | 70 | set value for deposition (min) |
| LDepMean | 620.23 | load zone deposition step temperature average |
| LDepSig | 1.33 | load zone deposition step temperature standard deviation |
| CDepMean | 620.51 | center zone deposition step temperature average |
| CDepSig | 1.30 | center zone deposition step temperature standard deviation |
| SDepMean | 620.17 | source zone deposition step temperature average |
| SDepSig | 0.94 | source zone deposition step temperature standard deviation |
| $SIH_4$DepMean | 249.79 | deposition step $SIH_4$ flow average |
| $SIH_4$DepSig | 0.08 | deposition step $SIH_4$ flow standard deviation |
| $PH_3$DepMean | 0.00 | deposition step $PH_3$ flow average |
| $PH_3$DepSig | 0.00 | deposition step $PH_3$ flow standard deviation |
| mtorrDepMean | 550.74 | deposition step pressure average |
| mtorrDepSig | 1.27 | deposition step pressure standard deviation |
| depogroup | 82 | the total collected samples during deposition step |

**info.tylan16.22810.Nov.13.1989 (a typical *info* file name)**

```
DATE 11/13/1989
START-MONITOR-TIME 15:50:12
EQUIP tylan016
PROCID FUPLY16C
RUNID 22810
LOTID bob10
USER bob
STAB-TIME 78.6
DEPO-TIME 240
SIH4 200.0
PH3 .0
PRESSURE 380.0
TEMPL 595.0
TEMPC 595.0
TEMPS 595.0
LDepMean  594.86
LDepSig  1.07
CDepMean  594.65
CDepSig  1.01
SDepMean  594.39
SDepSig  0.85
sih4DepMean  199.06
sih4DepSig  0.23
ph3DepMean  0.00
ph3DepSig  0.00
mtorrDepMean  380.43
mtorrDepSig  1.57
depogroup 145
```

# Appendix 3.3a

## Manual Page for the Tylan Furnace Monitoring Program
## (tytasks)

NAME

tytasks – a program that monitors Tylan diffusion furnaces.

SYNOPSIS

tytask -[cvl]

DESCRIPTION

*Tytask* is a generic monitoring program for all the Tylan furnaces. There are three options for tytask.

-c To collect data.
When monitoring with the -c option, the program will ask the user to provide the tube name (i.e. 1 - 16), non-critical step monitoring frequency (default is a sample every 2 minutes), critical step monitoring frequency (default is a sample every 20 seconds), and names of *data* and *info* files.

-v To collect and view data on X window display (workstation only).
The user will have to provide the same information as in option -c.

-l To *playback*, *copy*, and *pretty-print* the *data* and *info* files, and *quit* the program.

Once *tytasks* is running, two files will be generated and stored under /home/argon1/micro/lib/bcam/data/monitoring/<equipname>/data_dir/:
data.<equipname>.runID.date contains the monitoring data according to the predefined format below, while info.<equipname>.runID.date contains the date, start-monitor-time, procID, lotID, runID and other information unique to the process being monitored.
After process run, information in the *info* file is then stored in the *main* and *<equipname>mon* tables of the INGRES database.

## FILES

micro/lib/bcam/bin/ directory containing the source code

## SEE ALSO

online

## AUTHORS

Norman H. Chang, Adhi Gaduh, Dave C. Mudie, Scott Miles

# Appendix 3.3b

## Manual Page for General Interactive Use of the Monitoring Program (online)

### NAME

online - graphical user interface for the real-time monitoring program

### SYNOPSIS

online <equipname>

### DESCRIPTION

*Online* provides the graphical user-interface for the real-time monitoring program. It displays real-time monitoring data in an X-Y window with options specified in a stack menu.

The stack menu provides four frames. The *DISPLAY* menu has the options for showing maximum display in the window, zoom-out, display of grid lines and bounding box. The *FLAG* menu has the options for splines, display of big marks or small marks, change of time unit to hh:mm:ss, and display of bar lines. The *MISC* menu contains commands for clearing the window, saving the plot and quitting the program. The sensors to be monitored are selected from the *ITEM* menu.
The program groups data entries with similar units into one X-Y window and a new window is added automatically when entries with new units are selected from the *ITEM* menu.

Command lines are selected by clicking and holding the middle or right mouse button on the respective menu option. Zooming is accomplished by clicking and dragging the left mouse button to an arbitrary square area on the monitoring window.

The program will automatically search for position, initial active graph entries and sensor name substitution. If no such user-defined file exists, the program will use its default definition.

The default file specified by the user is saved as ".<equipment>MONDEF" (equipment monitoring defaults) in the user source directory for a piece of equipment. In the default file, the user can specify the following parameters:

XPOS <value> - the x-coordinate of the first monitoring window
YPOS <value> - the y-coordinate of the first monitoring window
XSPC <value> - the incremental x spacing for the next monitoring window
YSPC <value> - the incremental y spacing for the next monitoring window
WIDTH <value> - the width of a monitoring window
HEIGHT <value> - the height of a monitoring window
PLOT <value> - plot the sensor #<value> when *online* initiated
<sensor> <substitute name> - the <sensor> name is substituted by the <substitute name>

The following is an example default file.

```
XPOS    50
YPOS    200
XSPC    0
YSPC    100
WIDTH   800
HEIGHT  200
PLOT    1
PLOT    3
Gas_1   NITROGEN
Gas_2   ARGON
Gas_5   OXYGEN
```

FILES

/home/argon1/bcam/src/monitoring/tylans directory containing the source code

## SEE ALSO

tytasks

## AUTHORS

Norman H. Chang, Adhi Gaduh, Dave C. Mudie

# *Chapter 4*
# *Managing the Equipment Maintenance Records*

## 4.1. Introduction

The current practice of equipment maintenance in the semiconductor industry consists of three activities, namely *equipment qualification*, *preventive maintenance*, and *field repair*. Equipment qualification includes a number of tests that must be completed before the equipment is released to production. Preventive maintenance is conducted at regular intervals and consists of routine inspection and cleaning. .Unscheduled field repairs are often necessary during production to resolve equipment malfunctions.

Today, most of the information concerning *equipment qualification* and *preventive maintenance* is collected on paper records. Operators or technicians must refer to these records while conducting the respective activities. Information concerning *field repairs*, however, has been managed in an ad-hoc manner. Malfunctions have been reported orally or recorded in a free-style text report by the operators. Even when using a paperless, computerized system such as COMETS [1] (a popular work in progress (WIP) system), or the Berkeley lab information system (BLIS), the causes and actions to correct malfunctions are lost in long, unstructured reports. Since this information is not structured properly it is very difficult to perform even the simplest analysis in order to highlight the most frequent failures or to discover and correct common causes behind malfunctions.

In this chapter we present a new scheme for managing the information concerning equipment maintenance activities. The scheme allows the user to interact with a relational database through a form-based interface in order to record preventive maintenance and field repairs of manufacturing equipment. All the information necessary to identify the type and cause of equipment failure, time of occurrence, time of repair, etc. is stored in the relational database. Contrary to previous applications, this information is highly organized and stored in a symbolic form, so that application programs can be developed to perform process-wide analysis.

Next we outline the goals of the system. Section 4.3 describes the information structure, while Section 4.4 illustrates the system functionalities. An application example is given in Section 4.5. Finally, the conclusions are given in Section 4.6.

## 4.2. Goals of the Equipment Maintenance Record Keeping System

The system is used by equipment operators, maintenance technicians, and facility managers and it serves four major functions. The first function is to help the fab management keep track of equipment maintenance events. This is accomplished through easy access to maintenance records and equipment failure statistics. A second function is to help maintenance technicians conduct off-line equipment maintenance by reviewing previous maintenance cases. A third function is to enhance the operation of automated diagnostic systems; since failure history is highly structured, we can use it to automatically update the knowledge base of such a diagnostic system.† The last function is to facilitate off-line equipment maintenance by providing lead operators or maintenance technicians with multimedia (graphics, text, video, etc.) step-by-step guidance.

## 4.3. Information Structure

In order to support the above mentioned functions, the system must contain clear and unambiguous maintenance reports. To achieve this, the semantics of equipment maintenance events must be formalized. In this regard, we have treated an equipment maintenance event as a collection of many-to-one mappings between observable

† Chapter 6 contains a detailed account of this application.

*symptoms* and actual *faults*. Observable symptoms are what an operator sees when the equipment malfunctions. Such a symptom might be the fact that the motorized wafer boat is stuck during a furnace run. Faults are the causes behind the equipment malfunction. These are derived from the diagnosis conducted by a maintenance technician, and are verified by the person who actually repairs the equipment. For example, a malfunction in the boat controller could be the cause behind the symptom "boat stuck". Observable symptoms such as "boat-stuck" and "controller-screen-blank-out" are reported by the operator, while the cause behind the symptom, i.e., the fault ("boat-controller-down") is identified and reported by the maintenance personnel.

In order to avoid ambiguities, one standard description must be agreed upon for all the symptoms and faults by all operators and technicians. Enforcing this standard description is easy since it is coded using descriptive keywords, each paired with a long description for better understanding.

Faults and symptoms for a specific piece of equipment are logically grouped into hierarchical categories to simplify their organization. In order to avoid duplication, these hierarchical structures are extended upwards to include equipment groups, or clusters. Complete hierarchical categories are initially acquired for each cluster of similar equipment.

When there are different specialized equipment in the cluster, the corresponding specialized symptoms and faults are marked for the equipment while the others are generic throughout the cluster. For example, hierarchical categories are created for a furnace cluster that contains 16 individual reactors. Among them, there are about 10 different kinds of reactors running with different sets of gasses, in varying temperature and pressure ranges. Specialized symptoms and faults for these reactors are in use when a particular equipment is in question.

### 4.4. System Functionality

The system supports the functions of problem and resolution reporting, report browsing and downtime statistics generation. These functions are described below.

#### 4.4.1. Reporting a Problem

When an operator completes a processing step, the system asks if anything abnormal has occurred during operation. If so, the equipment control program automatically enters the equipment maintenance record keeping system. The operator is prompted to choose one or more *symptoms* from a comprehensive list of symptoms known to occur on that particular equipment. Alternatively, a problem report can also be automatically created by a diagnostic system.†

The symptoms are structured in a tree structure that allows the operator to be more and more specific about the nature of the observed malfunction. If the operator is not sure of the exact nature of the problem, or if none of the items on the list match the observed symptoms, the operator may choose the "can't-tell" or "none-of-above" options. Since ambiguity might arise lower in the decision tree, these options are available at all levels. For example, "can't-tell" is chosen when the operator identifies that the observable symptom is in the *pressure* area but cannot tell whether it can be best described as "fail-leak-test" or "pressure-out-range".

The operator is then prompted for a free-form text description of the problem. This free form report might contain details to aid the equipment maintenance technician in diagnosing the problem.

When a new report is generated, the symptom keyword and the accompanying comments are entered into the database. The reported event is then treated as a "pending"

† See Section 4.5 of this chapter.

problem and is electronically posted where it will be seen by other operators of the equipment. At the same time the system notifies the designated maintenance technician about the malfunction via electronic mail.

### 4.4.2. Diagnosing the Problem and Reporting the Repair

Repair reports are generated when a technician repairs the malfunctioning equipment and enters an assessment about the problem. At this time, the repairing technician is asked to choose a *fault* from the list of failures known to occur on that equipment. Like the list of symptoms, the faults are also organized in a tree structure.

Although there could be multiple symptoms associated with each report, only one fault may be reported. Therefore, when multiple faults are identified for the problem in question, the technician can *split* the report into several reports, each with a fault and corresponding symptoms. The technician has the same "can't-tell" and "none-of-above" options as the reporting operator. In addition, the technician can also modify existing entries or add new ones to the fault and symptom lists through a friendly user interface.

After diagnosing and repairing the problem, the technician has the option to provide a longer, free-form description of the event. When complete, the report is stored in the database and the warning to the operators is removed, thereby *clearing* the equipment problem.

On occasion a technician might identify a problem without being able to repair it, as some special-order parts may be needed or the problem may not be very important. In this situation, diagnosing a problem without clearing it serves an important administrative function; production managers and equipment users know that the problem has been looked at, and the technician is given the opportunity to enter further comments such as "parts on order, due on 3/3/90".

### 4.4.3. Report Browsing

Clearly, all the maintenance information that has been collected in the database should be easily accessible to operators and technicians as well as to the fab management. In our system a simple form-based query interface allows users to search current and archived reports by specifying equipment name, user name, or problem types or any combination such as "When was the *last time* we had a *pump failure* on tylan16?" Also, the set of pending reports is retained in the *equipment status board*; since most users are only interested in current problems, the system displays this status board by default.

While browsing through a particular report, privileged users (members of the maintenance staff) have the option to:

1. *Update* a report by adding new text comments
2. *Diagnose* a report, or change an earlier diagnosis
3. *Clear* the report from the list of pending problems
4. *Delete* an erroneous report from the database
5. *Split* a report (when multiple faults are present)
6. Create a *New* report

Most of these options are self-explanatory. The *split* option is used when two faults have been identified and the user has listed symptoms from both failures in the same report. If symptoms from more than two problems are reported together, the report may be split several times.

### 4.4.4. Failure Analysis and Downtime Statistics

The information accumulated in the equipment maintenance database is made available to independent analysis programs. One important function of these programs is to provide users with database queries to support basic downtime statistics. For example, lab managers frequently ask "What are the uptimes of all the equipment in the past three months?", "What is the equipment maintenance cost?", etc. Technicians, frequently ask questions such as "How often has been symptom A associated to fault B in the past?", or "What are the failure frequencies of a specific item of equipment?", etc. Simple plotting of various downtime statistics such as equipment uptimes and failure frequencies ("Pareto" chart) often gives powerful visual summaries of the cleanroom condition. Figure 4.1 shows an example of the annual fault frequency chart for Tylan furnace.

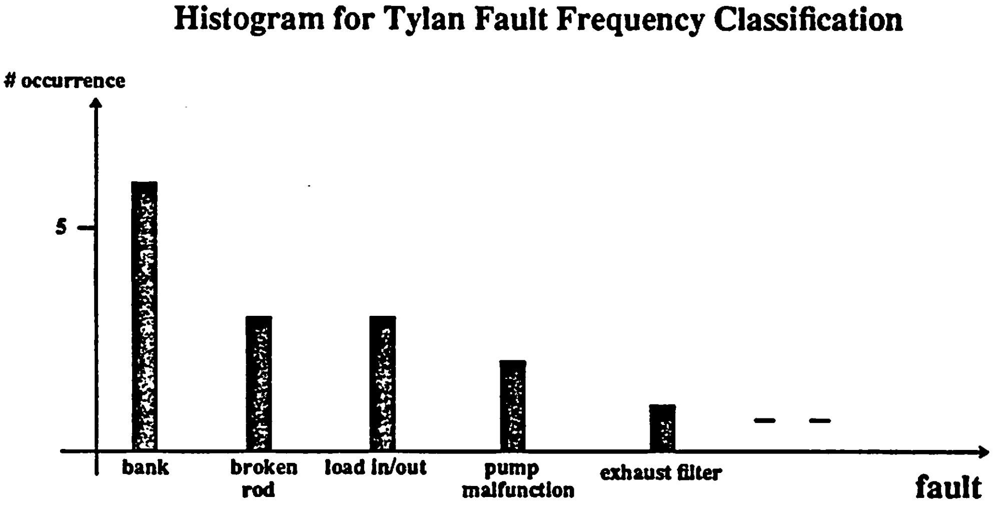


Fig. 4.1 A histogram showing the annual fault frequency for Tylan furnace.

## 4.5. Reports Generated by an Automated Diagnostic System

An automated diagnostic system interacts with the equipment maintenance record keeping system by initiating "pending" maintenance events. In this case the symptom categories, however might include symptoms that cannot be detected by the operators. This is because the automated diagnostic system has a more detailed "view" of the operation through real-time monitoring. Appendix 4.1a shows the symptom categories for Low-Pressure Chemical Vapor Deposition (LPCVD) reactors. Those symptoms marked with * can be seen by the automated diagnostic system but not by the operators.

Similarly, there is a difference in the level of detail in detecting a fault from a maintenance technician's point of view. This is because the diagnostic system cannot evaluate subtle pieces of evidence that are not represented in its knowledge base. So, it cannot always pin-point the actual faults. For example, the diagnostic system may conclude that there is something wrong in the *pump-system* without offering any more details. After going through several testing procedures and detailed examination, the technician might discover that the actual fault is ''filter-dirty'', a level of detail not accessible by the automated diagnostic system.

Note that the interaction between the diagnostic system and the maintenance database is bidirectional. Indeed, an important piece of information that can be derived from the equipment maintenance information management system is the *symptom-fault incidence*. These are the cumulative frequencies of symptom-fault observed by the operators and diagnosed by the technicians. The cumulative frequencies are used to update the symptom-fault belief list of the diagnostic system (see chapter 6).

### 4.6. Implementation Details

The equipment maintenance record keeping prototype is implemented with Relational Technology's INGRES database system [2] and the Application-By-Forms (ABF) development package [3]. The system is designed to run on ASCII terminals as well as on bitmapped displays using the X Window System.

### 4.6.1. Database Design

The equipment maintenance database is composed of three relations representing the main "objects" of the system, and several relations that describe how these objects are connected.

The first relation, *faults*, represents the hierarchy of fault and symptom entries. Each entry has a unique ID number, a short name, a long description, and the ID of its parent to identify a location in the hierarchy. Currently, faults and symptoms are stored as two disjoint hierarchies in the same data structure because there are no difference between them in nature.

A second relation, *report*, records the information unique to each reported problem. Each entry stores a unique ID, the equipment involved, and an associated comment ID. Additionally, each entry records the user who reported the problem, the time of reporting, the user who diagnosed the report, the time of diagnosis, and the time of final problem resolution.

Finally, the relation *comments* is used to store text blocks of arbitrary length. Each entry consists of a *comment ID*, a text sequence number, and 80-characters of text. Entries with the same ID are sorted in sequence to reproduce the original block of text.

#### 4.6.1.1. Database objects

The equipment maintenance record keeping system also makes use of the following relations from the existing Microlab database:

1. *labusers* - people who use the facility
2. *resource* - equipment in the facility
3. *prgroup* - identifies groups of equipment

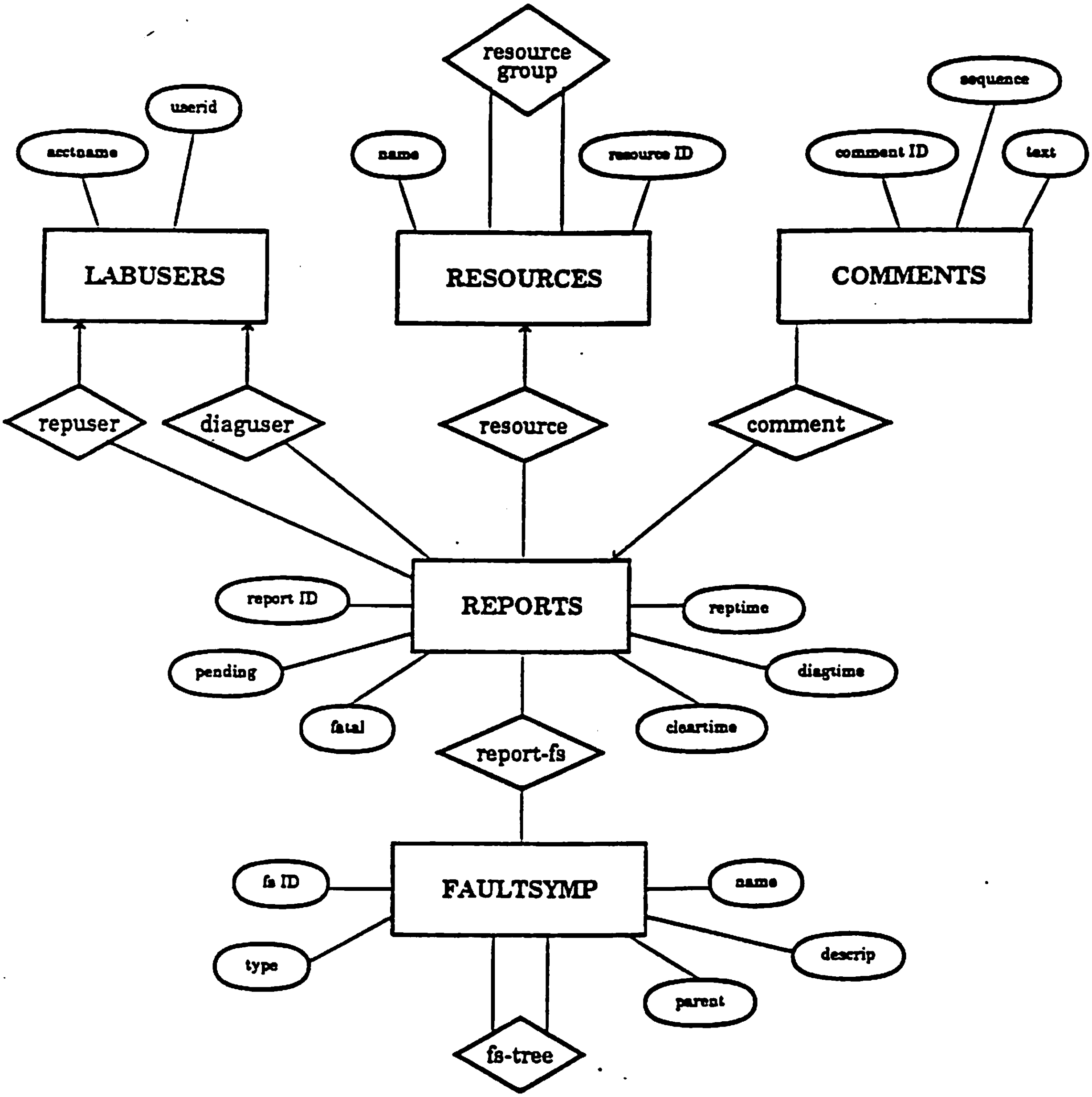
#### 4.6.1.2. Mapping relations

The following relations are used to specify relationships between the above objects:

1. *reportfault* - identifies which faults are associated with a report
2. *reportsymp* - identifies which symptoms are associated with a report
3. *faulttree* - cache of hierarchy in *faults*

### 4.6.2. Database Summary

Figure 4.2 summarizes the database relationships according to the Entity-Relationship diagram convention [4].



**Entity-Relationship diagram for FAULTS database**

Boxes indicate database objects

Ovals indicate attributes of an object

Diamonds indicate relationships between objects

An arrowhead indicates a "many-to-one" relationship

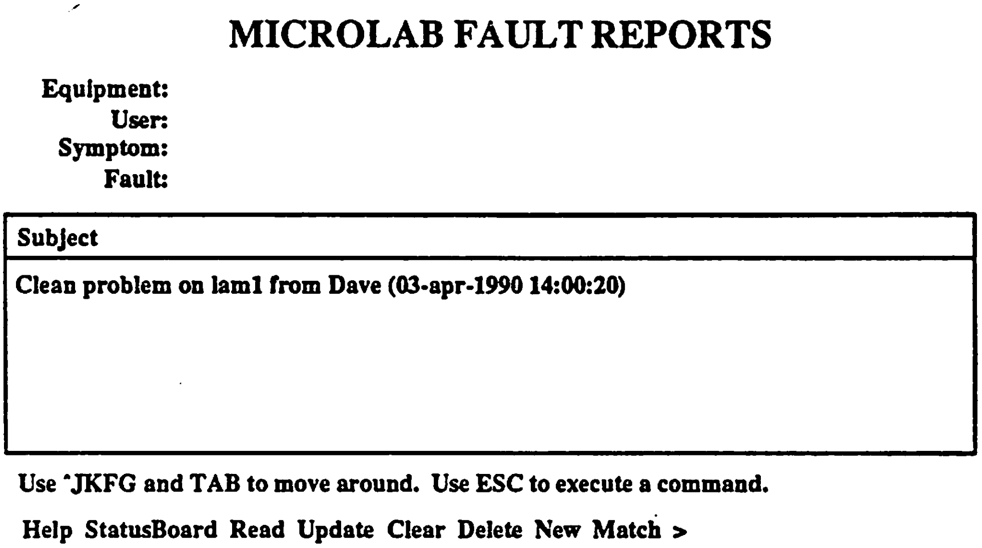
Figure 4.2 The database Entity-Relationship diagram.

## 4.7. An Application Example

Here we show an example of using the equipment maintenance records to report, diagnose, repair and clear a problem.

During an LPCVD processing step, John (the operator), observed that the controller screen (TYCOM) of the LPCVD reactor suddenly stopped responding. After trying anything he knew, he failed to make the TYCOM respond. He then actuated the equipment maintenance record keeping system to report the problem on an ASCII terminal next to the malfunctioning TYCOM. Figure 4.3 shows the first screen of the equipment maintenance records. This screen reports a pending problem on a different machine. Clearly, the TYCOM problem has not been reported yet, so John chose the command "new" to initiate a new problem report as shown in Figure 4.4. He checked the problem report before he released the report to the "pending" problem status board (Figure 4.5). Once the problem report is released as "pending", the system will notify the responsible technician and supervisor via electronic mail or via broadcasting message if the problem is serious. Later, Susan, the technician, diagnosed the problem and chose the actual fault (central control module -> defective electronic cage -> defective microcomputer board (MCB)) from the fault menu of the Tylan furnace as shown in Figure 4.6 and 4.7. Susan then cleared the report (Figure 4.8). This fault (MCB) is then paired with the symptom (TYCOM) and stored in the database.

A help menu is also available throughout the operation of the equipment maintenance record keeping system. An example of the help menu is shown in Figure 4.9. Figure 4.10 shows how the technician can edit the symptom and fault menus through a friendly user-interface.



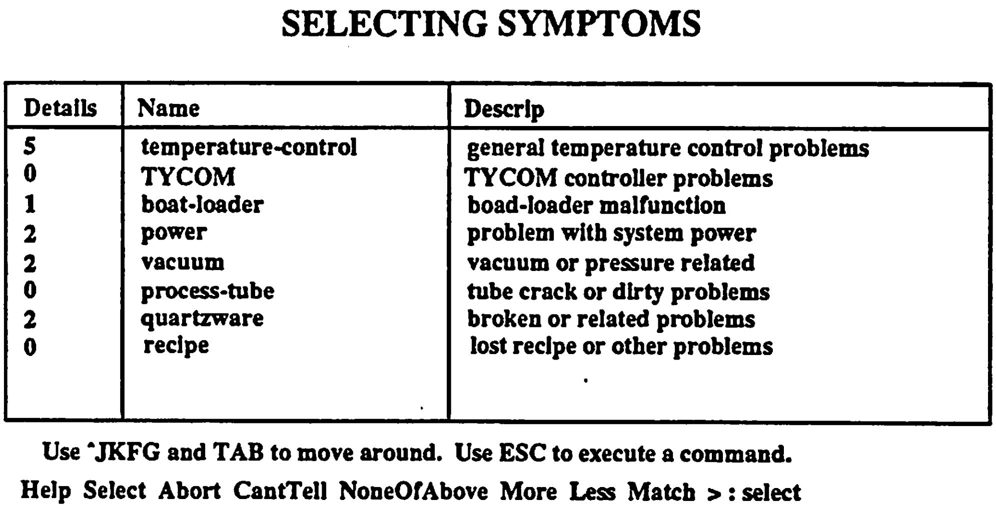
MICROLAB FAULT REPORTS

Equipment:
User:
Symptom:
Fault:

| Subject |
|---|
| Clean problem on lam1 from Dave (03-apr-1990 14:00:20) |

Use ^JKFG and TAB to move around. Use ESC to execute a command.

Fig. 4.3 The first screen of the equipment maintenance records. Users can look at the current "pending" problems in the status board (It shows a Lam etcher problem), initiate a new problem report, or look at old reports.



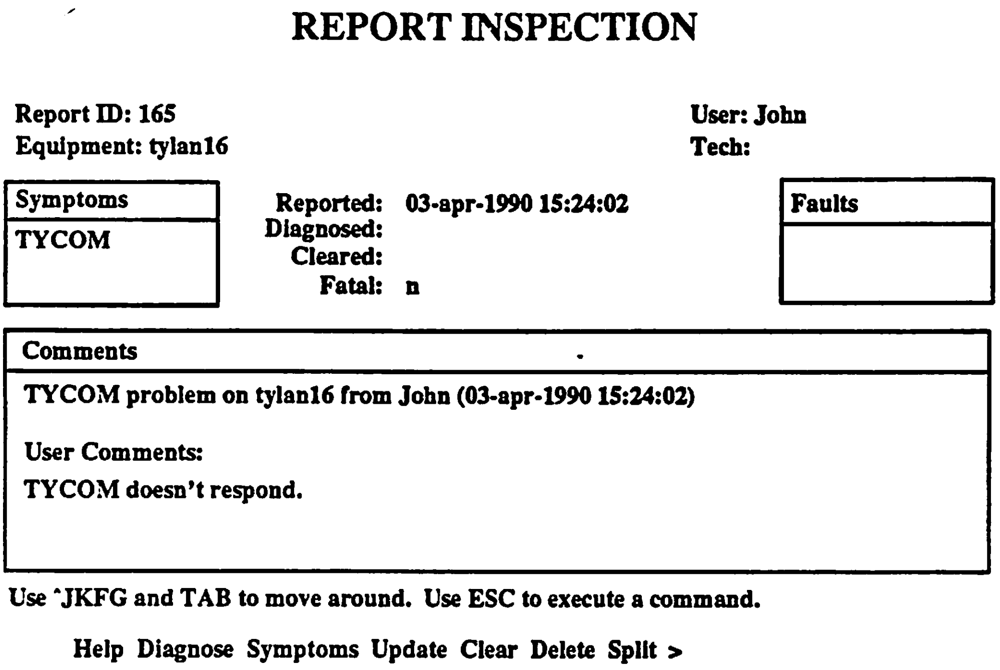
SELECTING SYMPTOMS

| Details | Name | Descrip |
|---|---|---|
| 5 | temperature-control | general temperature control problems |
| 0 | TYCOM | TYCOM controller problems |
| 1 | boat-loader | boad-loader malfunction |
| 2 | power | problem with system power |
| 2 | vacuum | vacuum or pressure related |
| 0 | process-tube | tube crack or dirty problems |
| 2 | quartzware | broken or related problems |
| 0 | recipe | lost recipe or other problems |

Use ^JKFG and TAB to move around. Use ESC to execute a command.

Help Select Abort CantTell NoneOfAbove More Less Match > : select

Fig. 4.4 An operator observed a TYCOM malfunction (the control screen has no response) and selected the TYCOM symptom from the symptom menu of the Tylan furnace. More than one symptom can be chosen.



REPORT INSPECTION

Report ID: 165
Equipment: tylan16

User: John
Tech:

Symptoms
TYCOM

Reported: 03-apr-1990 15:24:02
Diagnosed:
Cleared:
Fatal: n

Faults

Comments

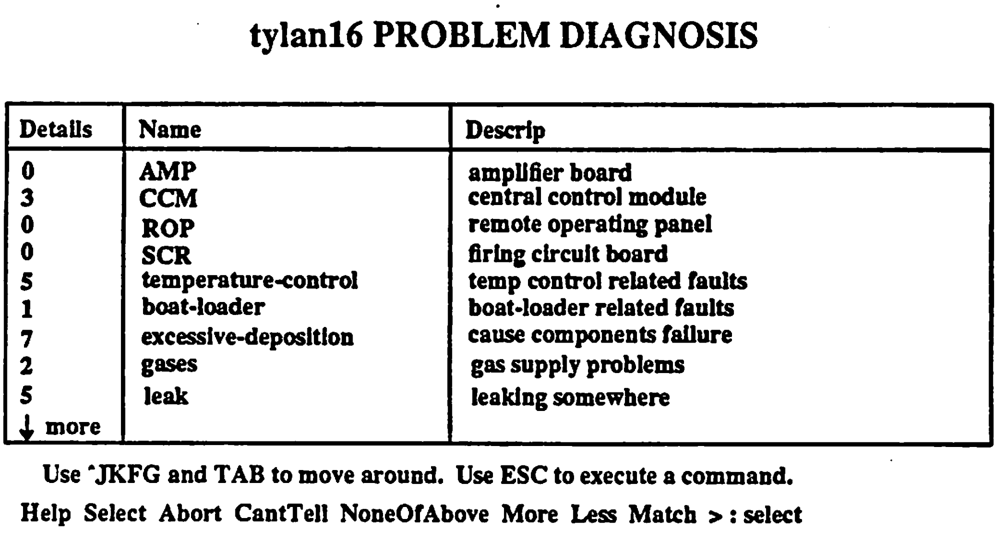
TYCOM problem on tylan16 from John (03-apr-1990 15:24:02)

User Comments:

TYCOM doesn't respond.

Use ^JKFG and TAB to move around. Use ESC to execute a command.

Fig. 4.5 The malfunction report can be checked and modified by the operator before releasing the problem report. Once the problem report is active (or "pending"), the system will notify the responsible technician and superviser via electronic mail or via broadcasting message if the problem is serious.



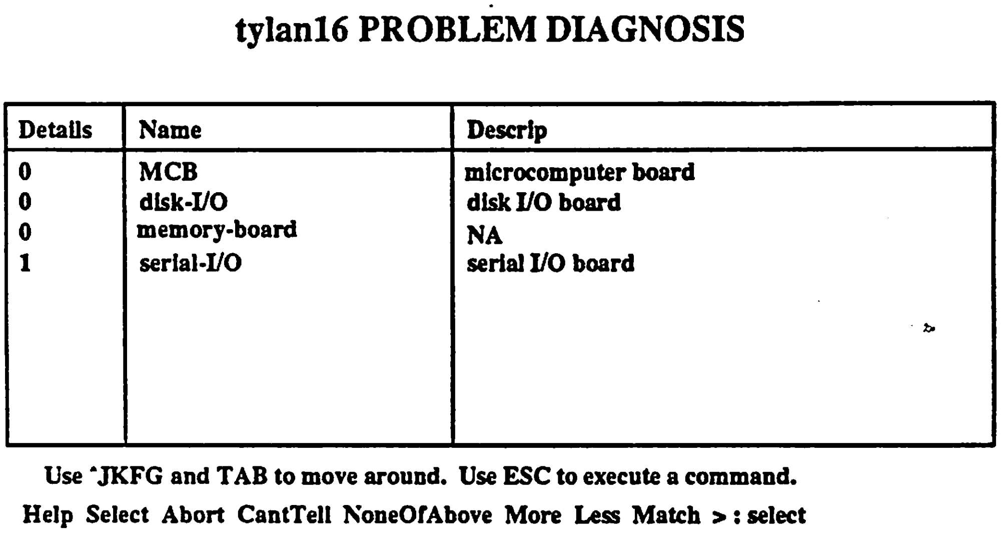
tylan16 PROBLEM DIAGNOSIS

| Details | Name | Descrip |
|---|---|---|
| 0 | AMP | amplifier board |
| 3 | CCM | central control module |
| 0 | ROP | remote operating panel |
| 0 | SCR | firing circuit board |
| 5 | temperature-control | temp control related faults |
| 1 | boat-loader | boat-loader related faults |
| 7 | excessive-deposition | cause components failure |
| 2 | gases | gas supply problems |
| 5 | leak | leaking somewhere |
| ↓ more | | |

Use ^JKFG and TAB to move around. Use ESC to execute a command.

Help Select Abort CantTell NoneOfAbove More Less Match > : select

Fig. 4.6 Later, the technician diagnosed the problem and chose the actual fault from the fault menu of the Tylan furnace. Here the CCM (defective central control module) fault was selected by the technician.



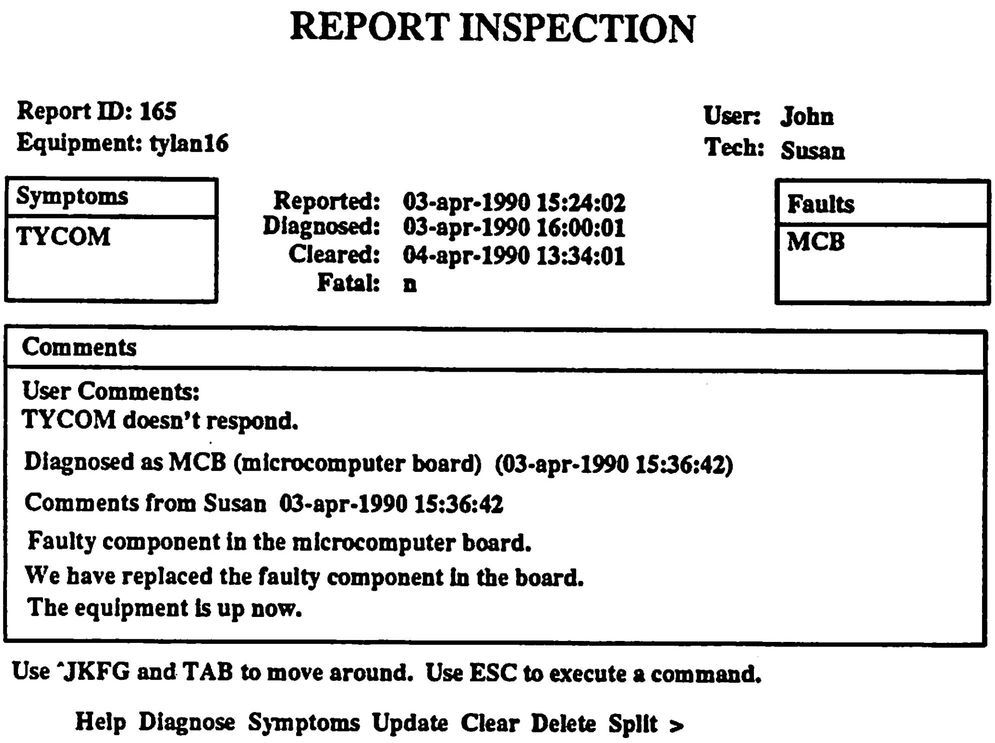
tylan16 PROBLEM DIAGNOSIS

| Details | Name | Descrip |
|---|---|---|
| 0 | MCB | microcomputer board |
| 0 | disk-I/O | disk I/O board |
| 0 | memory-board | NA |
| 1 | serial-I/O | serial I/O board |

Use ^JKFG and TAB to move around. Use ESC to execute a command.

Help Select Abort CantTell NoneOfAbove More Less Match > : select

Fig. 4.7 Electronic-board-cage (defective electronic board cage) was selected under the CCM fault. Finally, MCB (defective microcomputer board) was chosen from the four choices under electronic-board-cage by the technician. This fault (MCB) is paired with the symptom (TYCOM) in the database. If multiple faults are present, the report can be split into multiple reports each with the corresponding symptoms and one actual fault.



REPORT INSPECTION

Report ID: 165
Equipment: tylan16

User: John
Tech: Susan

| Symptoms |
|---|
| TYCOM |

Reported: 03-apr-1990 15:24:02
Diagnosed: 03-apr-1990 16:00:01
Cleared: 04-apr-1990 13:34:01
Fatal: n

| Faults |
|---|
| MCB |

Comments

User Comments:
TYCOM doesn't respond.

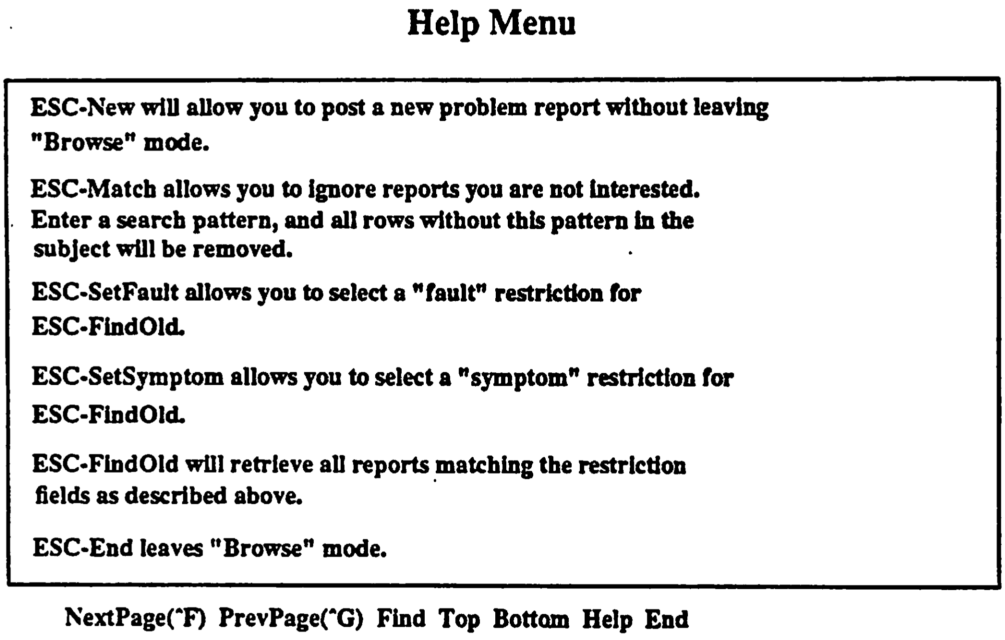
Diagnosed as MCB (microcomputer board) (03-apr-1990 15:36:42)

Comments from Susan 03-apr-1990 15:36:42

Faulty component in the microcomputer board.
We have replaced the faulty component in the board.
The equipment is up now.

Use ^JKFG and TAB to move around. Use ESC to execute a command.

Fig. 4.8 The final report shows all the information concerning the maintenance event. When a report is cleared by the technician, the problem is considered solved and the equipment is up again.



Help Menu

ESC-New will allow you to post a new problem report without leaving "Browse" mode.

ESC-Match allows you to ignore reports you are not interested. Enter a search pattern, and all rows without this pattern in the subject will be removed.

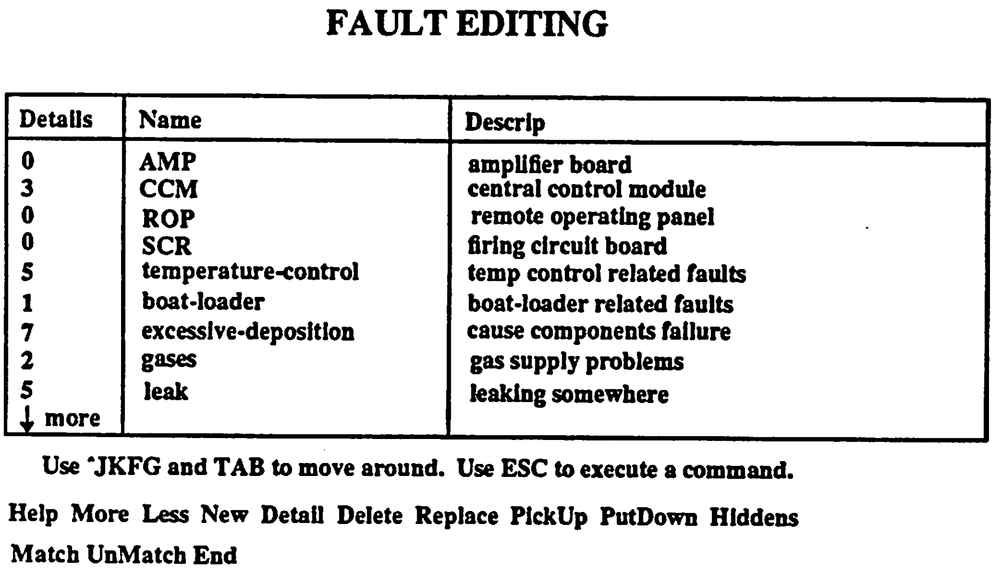
ESC-SetFault allows you to select a "fault" restriction for ESC-FindOld.

ESC-SetSymptom allows you to select a "symptom" restriction for ESC-FindOld.

ESC-FindOld will retrieve all reports matching the restriction fields as described above.

ESC-End leaves "Browse" mode.

Fig. 4.9 A Help menu of the current screen is available when needed.



FAULT EDITING

| Details | Name | Descrip |
|---|---|---|
| 0 | AMP | amplifier board |
| 3 | CCM | central control module |
| 0 | ROP | remote operating panel |
| 0 | SCR | firing circuit board |
| 5 | temperature-control | temp control related faults |
| 1 | boat-loader | boat-loader related faults |
| 7 | excessive-deposition | cause components failure |
| 2 | gases | gas supply problems |
| 5 | leak | leaking somewhere |
| ↓ more | | |

Use ^JKFG and TAB to move around. Use ESC to execute a command.

Help More Less New Detail Delete Replace PickUp PutDown Hiddens
Match UnMatch End

Fig. 4.10 The technician can edit the symptom and fault menus. Here a complete editing menu is shown.

## 4.8. Conclusions

The first release of the system was integrated into the current Berkeley Lab Information System (BLIS) in March, 1990. The new equipment maintenance information management system handles all the equipment maintenance events. The old text-based maintenance system is kept for read-only browsing and archival purposes. Appendix 4.1b shows the fault categories of LPCVD reactors.

The system has successfully provided technicians and management with a tool for monitoring the equipment maintenance events and resolving equipment malfunction. The possibility of updating the knowledge base of the online diagnostic system through automatic collection of downtime maintenance statistics is being tested. An alternative system to facilitate off-line equipment maintenance by providing lead operators or maintenance technicians with step-by-step guidance with schematics and graphics is described in Chapter 5.

## References for Chapter 4

[1] *COMETS Reference Manual*, Consilium, 1990.

[2] C.J. Date, *A Guide to INGRES : A User's Guide to the INGRES Product*, Addison-Wesley Pub. Co., 1987.

[3] *INGRES/ABF Reference Manual*, Relational Technology, 1986.

[4] H.E. Korth, A. Silberschatz, *Database System Concepts*, McGraw-Hill, Inc., 1986.

# Appendix 4.1a

## Symptom Categories for LPCVD Reactors

(Symptoms marked with * are observable by the automated diagnostic system but not by the operators.)

1. **temp-control**
   - 1.1. **abnormal-stabilization-time**
   - 1.2. **critical-temp-differs**
   - 1.3. **temp-differs-TYCOM-ROP**
   - 1.4. **T/C-open**
   - 1.5. **T/C-out-calib**
   - 1.6. templ–out–control*
   - 1.7. tempc–out–control*
   - 1.8. temps–out–control*
2. **flow-control**
   - 2.1. sih4–out–control*
   - 2.2. ph3–out–control*
3. **pressure**
   - 3.1. **fail-leak-test**
   - 3.2. **pres-out-range**

4. **boat-loader**

    4.1. **unload-load**

5. **process-tube**

    5.1. **broken**

    5.2. **general-leak**

    5.3. toxic–gas–leak*

6. **quartzware**

    6.1. **broken**

    6.2. **excess-deposition**

7. TYCOM

8. **power**

    8.1. **tube-power**

    8.2. **bank-power**

9. program–on–hold*

10. **recipe**

11. inline–thickness–drift*

12. **preventive-maintenance**

# Appendix 4.1b

## Fault Categories for LPCVD Reactors

1. terminal

    1.1. keyboard

    1.2. CRT

2. printer

3. central-control-module

    3.1. power-supply

    3.2. floppy-control

    3.3. electronic-board-cage

        3.3.1. MCB (*microcomputer board*)

        3.3.2. memory-board

        3.3.3. disk-IO

        3.3.4. serial-IO

            3.3.4.1. software

            3.3.4.2. hardware

4. slave-module

    4.1. CPU

    4.2. digital-temp-controller

    4.3. I/O

- - **4.4. power-supply**
- **5. AMP (*amplifier board*)**
- **6. SCR (*firing circuit board*)**
- **7. ROP (*remote operating panel*)**
- **8. power**
    - **8.1. boat-loader**
    - **8.2. pump-system**
    - **8.3. gas-panel**
    - **8.4. load-station**
- **9. gasses**
    - **9.1. MFC (*mass flow controller*)**
    - **9.2. no-gas-in-cylinder**
        - **9.2.1.** $SIH_4$
        - **9.2.2.** $PH_3$
- **10. boat-loader**
    - **10.1. stuck**
- **11. process-tube**
    - **11.1. boat**
        - **11.1.1. missing-dummy-wafers**
        - **11.1.2. broken-dummy-wafers**
        - **11.1.3. broken-boat**
    - **11.2. baffle**
    - **11.3. cantilever-sheath-broken**

**12. pump-system**

**12.1. mechanical-pump**

**12.1.1. oil-pres-gauge**

**12.1.2. motor**

**12.1.3. inlet-filter-screen**

**12.1.4. pump-worn-out**

**12.2. oil-filtration-unit**

**12.2.1. filter-can-corroded**

**12.2.2. oil-gauge-replace**

**12.2.3. oil-pump**

**12.2.4. rubber-hoses**

**12.2.5. oil-filter**

**12.3. manifold**

**12.3.1. transition**

**12.3.2. bellows**

**12.3.3. inline-filter**

**12.3.4. gate-valve-defect**

**13. pressure-control**

**13.1. butterfly-valve**

**13.2. servomotor**

**13.3. cap-manometer**

**14. temp-control**

**14.1. temp-controller**

14.2. heating-element-worn

14.3. thermal-insulation

14.4. T/C-out-calib (thermocouple out of calibration)

14.5. T/C-open

15. leak

15.1. door-Leak

15.2. cantilever-leak

15.3. rear-transition-leak

15.4. rear-manifold-leak

15.5. quartzware-leak

15.5.1. tube-crack

15.5.2. T/C-sheath-crack

15.5.3. injector-crack

15.5.4. cantilever-sheath-crack

16. excess-depo

16.1. process-tube

16.2. boats

16.3. transition

16.4. bellows

16.5. inline-filter

16.6. butterfly-valve

16.7. gate-valve

17. Recipes

**18. maintenance**

- **18.1. temp-calib**
- **18.2. cap-manometer**
- **18.3. pres-servo-calib**
- **18.4. clean-excess-depo**
- **18.5. MFC-calib**
- **18.6. inline-filter-change**

# *Chapter 5*
# *An Off-Line Equipment Maintenance Aid*

## 5.1. Introduction

Scheduled and unscheduled maintenance is needed in order to keep manufacturing equipment operating properly. This can be achieved more effectively if abnormal machine conditions are detected as early as possible. A Generic Equipment Maintenance System (GEMS) was developed to assist operators in conducting off-line equipment maintenance and diagnosis. This work was carried out at Harris Semiconductor in the summer of 1988.

The goals of GEMS are to capture and retain the knowledge of equipment maintenance (EM) engineers. This will help operators to diagnose equipment malfunctions and increase mean time between failures (MTBF). GEMS can also be used as a training tool for novice EM engineers and equipment operators.

A diffusion furnace is a commonly used, complicated piece of equipment, so it was chosen to demonstrate the capability of GEMS. Without GEMS, lead operators can only solve 10% of furnace malfunctions. GEMS's first objective is to provide the guidance and knowledge so that lead operators can resolve 50% of furnace malfunction problems.

The structure of GEMS is designed to be general, so that many pieces of equipment can be accommodated. This can be realized by constructing GEMS in an object-oriented programming style, so that while the user interface is standardized, the reasoning strategy is chosen to suit various equipment maintenance and diagnosis problems.

GEMS is built on KEE, a commercial expert system shell [1] that provides menu and graphics interface, frame-based knowledge representation, and an object-oriented, rule-based inferencing scheme. Next, Section 5.2 describes the general user interface style employed by GEMS. Section 5.3 explains the knowledge acquisition process, inferencing strategy and knowledge scope of GEMS. The software structure is described in Section 5.4. Conclusions will follow in Section 5.5.

## 5.2. The User-Interface of GEMS

The GEMS display is partitioned into *prompt*, *comment* and *graphics* areas, as shown in Figure 5.1. Messages (questions and suggestions) for the users appear in the prompt area. The comment area shows additional information such as the reason why a question was asked. In the graphics area, graphic illustrations are shown along with the diagnosis session. A dynamically changing menu guides the user through the diagnostic tasks.

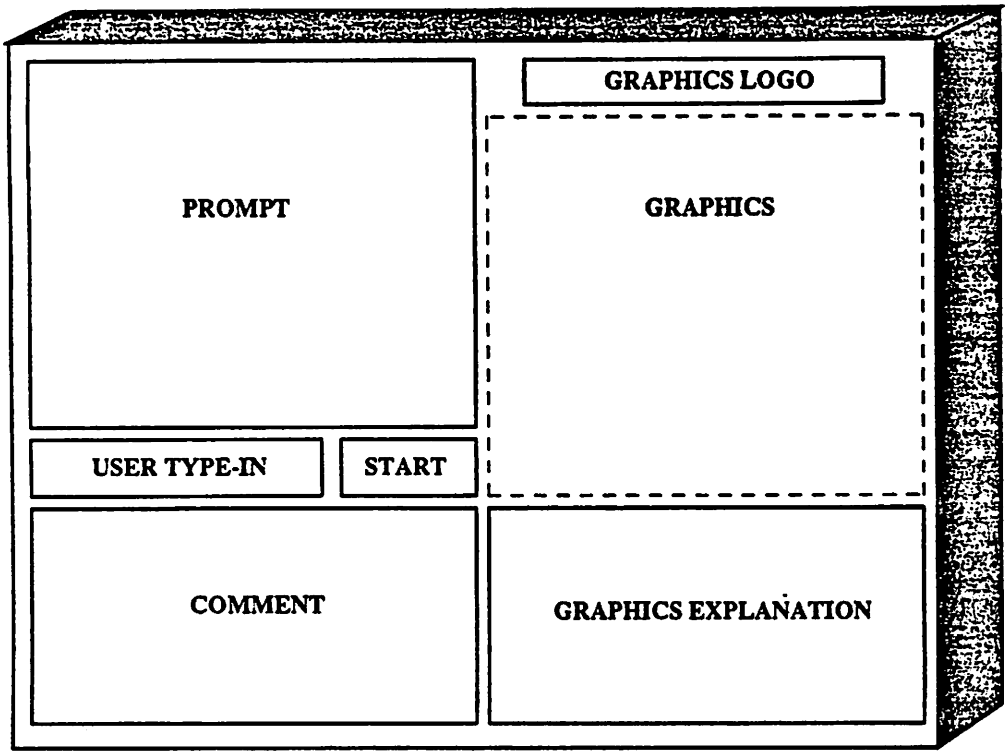
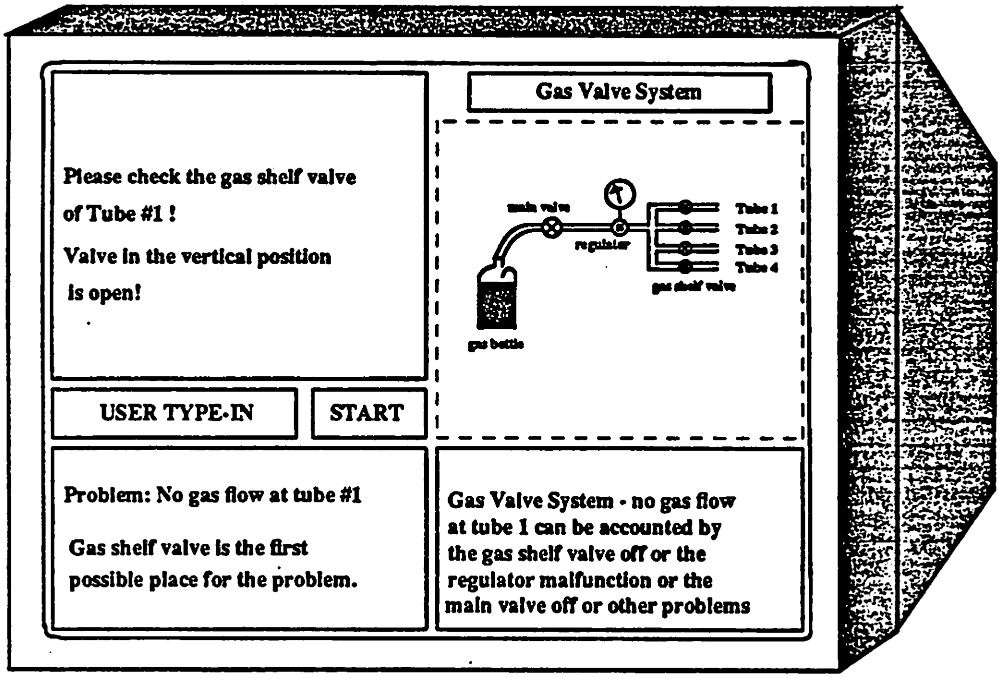


**Fig. 5.1** The front-end user interface design

Graphical representations are very important in equipment maintenance and diagnosis. An equipment problem which is difficult to describe in words very often can be effectively illustrated with simple graphics. Therefore, in GEMS, helpful schematics or graphics are provided wherever are needed. For example, a problem may be that no $H_2$ flow is detected. The reason may be because the main gas valve is not open, the gas shelf-valve is not in the right position, the regulator is not functioning properly, or a mass flow controller is malfunctioning. These problems can be communicated by a simple schematic illustrating the relationship between a gas flow system and gas valves. In Figure 5.2, we show an example of the off-line equipment maintenance aid, where the system prompts the user to check the gas shelf-valve of tube 1, while the corresponding schematic is displayed.



Please check the gas shelf valve of Tube #1 !

Valve in the vertical position is open!

Problem: No gas flow at tube #1

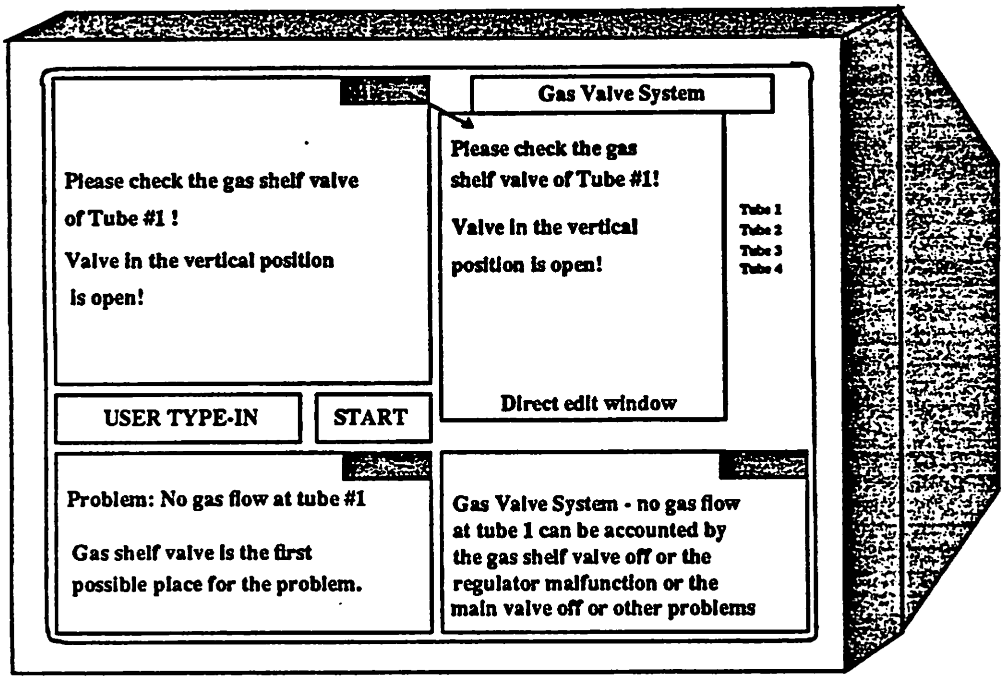
Gas shelf valve is the first possible place for the problem.

Gas Valve System - no gas flow at tube 1 can be accounted by the gas shelf valve off or the regulator malfunction or the main valve off or other problems

Fig. 5.2 The off-line equipment maintenance aid helps the user to diagnose the gas valve system problem.

The capability to generate reports is also an essential element of the off-line maintenance system. Reports are generated at the end of each session and include all the operations, their sequence, the specific tube, the time,· the name of the operator, etc. The usual maintenance event reporting is done at COMETS, a work-in-progress tracking system used by Harris Semiconductor. Since there is no direct link from GEMS to COMETS, GEMS guides the user to enter the information into COMETS at the end of each session.

Another important feature of GEMS is the fact that the expert maintenance engineer can edit the maintenance knowledge. This is done in the "modification mode" that allows editing the prompt, entering comments, modifying graphics explanations and reports for every diagnosis route, etc. This flexibility is needed because whenever the equipment-related conditions change, the knowledge base must reflect the change. Figure 5.3 shows the "modification mode."



Please check the gas shelf valve of Tube #1 !
Valve in the vertical position is open!

Please check the gas shelf valve of Tube #1!
Valve in the vertical position is open!

Problem: No gas flow at tube #1
Gas shelf valve is the first possible place for the problem.

Gas Valve System - no gas flow at tube 1 can be accounted by the gas shelf valve off or the regulator malfunction or the main valve off or other problems

Fig. 5.3 Editing the knowledge of GEMS. Here, the expert is editing the prompt question.

## 5.3. Knowledge Acquisition, Inferencing and Knowledge Scope of GEMS

There is no standard approach to the knowledge acquisition process, as knowledge can be attained from observations [2] or from first principles [3]. For GEMS, most of the knowledge comes directly from the expert engineers. After discussing the furnace problems with the domain experts, we organized the equipment symptoms, faults and solutions hierarchically. Figure 5.4 shows an example of this structure for the gas valve system.

According to how often problems have been happening in the past, step-by-step guidelines have been set to guide the user through an optimum sequence of diagnostic actions. The human expert usually checks the most likely problem first. Occasionally however, problems that are not as likely are checked first if they are easier to verify. For example, if the symptom is that there is an intermittent gas flow, the first thing to do is to check the main valve, the gas shelf-valve and the regulator before considering the more likely but harder mass-flow controller problem. Consequently, the basic knowledge structure is built on the heuristic hierarchical structure of symptom, fault, and solution.

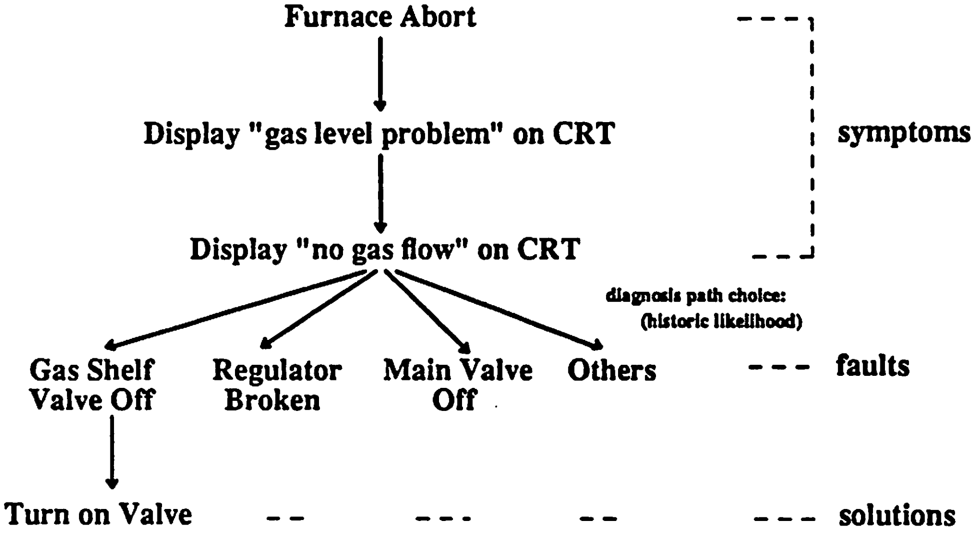


Fig. 5.4 Hierarchical decomposition of the gas valve system faults organized in symptoms, faults and solutions.

The knowledge scope of GEMS is partitioned into three levels: the lead operator level, the process/maintenance technician level, and the expert equipment maintenance engineer level. Figure 5.5 compares the knowledge levels for the diffusion furnace. Very large increases of knowledge are needed to move to the next more sophisticated level. The initial goal of GEMS is to acquire the complete scope of knowledge to help the lead operators solve 50% of the total furnace malfunction problems. (Typically, an unaided lead operator can solve 10% of the diffusion furnace problems). This will greatly increase machine up-time and hence efficiency of production.

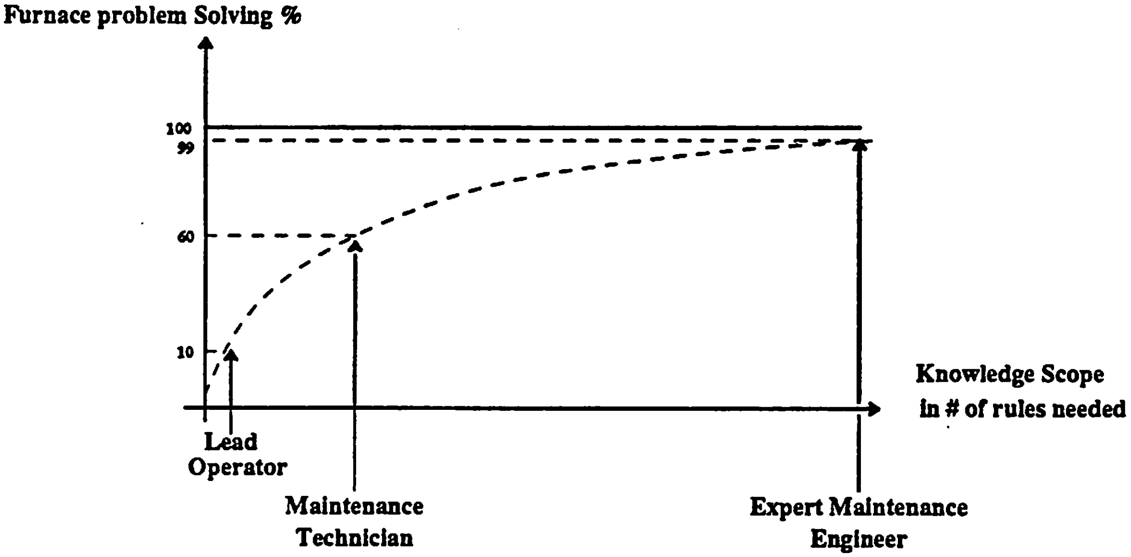


Fig. 5.5 The knowledge scope in terms of rules needed to match the different levels of experience.

## 5.4. The Software Structure of GEMS

GEMS is built on top of KEE, a commercial hybrid expert system shell, and runs on a SUN 3/60. GEMS makes extensive use of the object-oriented programming methodology in KEE for its knowledge structure and control strategy. GEMS performed sufficiently in an object-oriented programming environment.

GEMS consists of five objects.

(1) Basic GEMS Object : contains comments, graphics, prompt, and their associated functions.

(2) Question-Type-in Object : asks the user to type in some information in response to a question.

(3) Question-Choice Object : asks the user to pick an item from a number of available choices.

(4) Pure-Action Object : includes functions for guiding the user in completing an action.

(5) Yes/No-Action Object : asks the user a yes/no question and takes the appropriate actions.

Any of the subsequent objects (*units* in KEE's terminology) will either inherit the characteristics from the Basic GEMS object or from the combination of the Basic-GEMS object and one of the other four objects.

Symptoms are hierarchically decomposed into levels as shown before. There may be some actions or questions needed to lead to the next symptom level. When a symptom can be accounted for by single or multiple faults, heuristic diagnostic sequence is taken to investigate which fault can best explain the symptom. During the investigation of the possible fault, some actions and responses are needed for guiding the user through fault verification. A solution will be proposed once the fault is identified.

## 5.5. Conclusions

This project demonstrates how a Generic Equipment Maintenance System (GEMS) works for a diffusion furnace. GEMS is developed using KEE, fully utilizing its features such as its object-oriented programming, the menu and graphics capabilities.

The construction of the hierarchical knowledge base is object-oriented. Graphics and schematics are used to illustrate symptoms, questions or actions. After a fault has been identified, the system proposes a solution. The object-oriented style of the knowledge base makes the task of adding knowledge to the system easier, and also facilitates the construction of other equipment maintenance systems.

Since GEMS is a generic tool for equipment maintenance, it has the potential to become a large-scale equipment maintenance system for all the fabrication equipment in the production line. This way, GEMS could integrate and standardize the maintenance of all pieces of equipment in an IC fabrication environment providing the advantage of central control and quick adaptation of knowledge.

Currently, a specific GEMS has been built for the DDC furnace in use at the Harris semiconductor facility in Melbourne, Florida. The knowledge base contains about 2000 objects, and an equal number of rules. The DDC furnace GEMS has been in use on the manufacturing floor since January, 1989. We have conducted several operator training sessions for GEMS, and have found that the end-users (lead operators) are pleased with its performance. Some user comments are summarized below:

(1) The user cannot pay attention to several places on the screen at the same time. Therefore, the most important information should be presented in the prompt section, while the comment section should contain the additional information or the explanation of why the question in the prompt section is asked. The prompt and graphics sections are very important to the user.

(2) Using a mouse is direct and simple, and very convenient for clean room operators.

(3) The hierarchical decomposition into symptom, fault and solution is very comprehensive. The explanation of the diagnostic analysis greatly enhances a user's knowledge about the equipment.

Other applications of GEMS are currently planned for the Harris manufacturing facility.

GEMS's functions are complementary to the equipment maintenance record keeping system described in the last chapter. GEMS provides an explicit diagnostic route from observed symptoms to actual fault. We are investigating a case-based diagnostic methods to provide functions of both systems.

However, with the equipment maintenance and real-time monitoring data available (through SECSII link), a continuous diagnostic system can be built that aims to detect equipment problems automatically. The next chapter describes such an automated diagnostic system.

## References for Chapter 5

[1] Renate Kempf, Marilyn Stelzner, *Teaching Object-Oriented Programming with the KEE system*, OOPSLA, 1987.

[2] R. Michalski, et. al., *Machine Learning - An Artificial Intelligence Approach*, Chap. 11, 1986.

[3] R. Reiter, *A Theory of Diagnosis from First Principles*, Technical Report TR-187/86, Computer Science Department, University of Toronto.

# *Chapter 6*
# *Continuous Equipment Diagnosis Using Evidence Integration*

## 6.1. Introduction

Modem integrated circuit (IC) production requires the completion of hundreds of process steps. Most of these steps are conducted by sophisticated manufacturing equipment [1], whose reliable operation is vital to IC production [2,3]. Occasionally, equipment fails to operate properly. Serious malfunctions might result in lengthy downtime. Even minor malfunctions, if not identified at an early stage, can result in long runs of misprocessed product, and might eventually cause serious equipment damage.

Traditionally, an experienced equipment operator will attempt to identify and diagnose malfunctions by drawing on experience from the equipment operation history, real-time readings and observations of the finished product. Even assuming that an experienced operator is always present, this task is often time consuming and difficult. Hence, we are motivated to investigate an automated malfunction diagnosis system to aid equipment operators detect and analyze equipment problems [4].

Early diagnostic systems were based on the manipulation of experiential knowledge and pattem recognition techniques [5,6]. Unfortunately, early systems were restricted by not being able to accommodate quantitative models. They were further limited in their representation of ''uncertain'' knowledge. More recently, *Kramer* [7] proposed a diagnostic approach that uses quantitative models and probabilistic reasoning. Although this application shows significant promise for the continuous processing common in the chemical industry, its direct use in semiconductor manufacturing is questionable. First, semiconductor processing is mostly a "batch" process. Also, the development and use of first principle quantitative models of the process is not as advanced in the semiconductor industry as it is in the chemical industry. The reason for this is that most processing steps in semiconductor manufacturing are too complex to afford the development of practical, physically-based models. Recently a number of semi-empirical and empirical equipment-specific process models [8,9] have been developed. These models have been created and characterized through experimentation with the equipment in question. In this chapter we show how these models can be used within an automated diagnostic system.

The diagnostic system we are presenting here is based on processing information (or "evidence") regarding the behavior of the manufacturing equipment in question. Because of the batch character of the manufacturing process, "evidence" accumulates over time in an irregular fashion. Evidence regarding equipment condition can be collected before processing by checking the equipment maintenance records. It can also be collected during processing from online equipment sensors, and after processing from inline physical and electrical wafer measurements. Often, diagnosis cannot be conclusive until later evidence verifies an early conjecture about the problem.

An important feature of the system is based on the continuous accumulation of information coming in at different times during a manufacturing sequence. In order to accommodate the batch character of the semiconductor fabrication process, we have developed a "chronological" approach to equipment diagnosis. In this approach diagnosis is broken into three stages, where the system analyzes historical information *before* the run, real-time data *during* the run, and product inspection data *after* the run.

We use the Dempster-Shafer (D-S) evidential reasoning approach [10,11] to represent and combine evidence. This method provides for consistent and unambiguous evidence combination. A prototype of this diagnostic system was implemented in an object-oriented programming environment. This implementation enables knowledge and functionalities to be shared by different pieces of manufacturing equipment.

We have applied the D-S diagnostic method to a reactor used for Low Pressure Chemical Vapor Deposition (LPCVD) of undoped polysilicon films. Our results indicate that this diagnostic system is sensitive, stable, and accurate.

In the remainder of this chapter, a brief review of the Dempster-Shafer theory is given in Section 6.2. Section 6.3 describes the diagnostic methodology in detail. The adaptive adjustment of the belief functions is discussed in Section 6.4. The implementation of the diagnostic software is covered in Section 6.5. Section 6.6 discusses the application of the prototype system, and our conclusions are given in Section 6.7.

## 6.2. Dempster-Shafer Model for Fault Inference

Since its introduction in 1976, the D-S theory has demonstrated its usefulness in a number of practical applications [12,13]. This theory is briefly reviewed below.

### 6.2.1. Support, Plausibility and Uncertainty

According to the D-S theory, each proposition A is assigned a *probability range*, represented as [Bel(A), Pls(A)], a subset of the interval [0, 1]. The term Bel(A) is the *belief* measure of A (also known as "support" of A) and is directly proportional to the available evidence that supports the proposition A. The term Pls(A) refers to the *plausibility* of A. The plausibility is by definition related to the lack of evidence that contradicts A. Accordingly, if ¬A symbolizes the negation of the proposition A, the plausibility can be derived from:

$$\mathrm{Pls}(A) = 1 - \mathrm{Bel}(\neg A) \tag{6.1}$$

where Bel(¬A) indicates the measure of belief attributed to the proposition ¬A. Also,

$$\mathrm{Bel}(A)+\mathrm{Bel}(\neg A) = 1-(\mathrm{Pls}(A)-\mathrm{Bel}(A)) \leq 1 \qquad (6.2)$$

which means that the commitment of belief to a proposition does not force the remaining belief to be committed to its complement. This is a departure from classical probability and gives to the D-S method the power to express *uncertainty* u(A), defined as:

$$u(A) = \mathrm{Pls}(A)-\mathrm{Bel}(A) \qquad (6.3)$$

For example, A[0.4, 0.8] means that there is some belief for both A and $\neg A$. In particular, Bel(A) = 0.4, Pls(A) = 0.8 which means that Bel($\neg$A) = 0.2 and u(A) = 0.4. In terms of classical probability, we can say that the exact probability of A is known to be between 0.4 and 0.8.

### 6.2.2. Basic Probability Mass Distribution

The information that can be analyzed by our diagnostic system is categorized with the help of a set of descriptive "labels". For example, temperature readings will come under "temperature", film thickness readings under "film thickness" and information as to whether the equipment has been recently cleaned will come under the label "cleaning". Each label thus defines a category of information. We will call this set of categories the "evidence space" and we will symbolize it by E. Furthermore, the diagnostic system has a fixed number of failure hypotheses, one that relates to proper equipment operation and one for each of the recognizable failures such as "thermocouple out of calibration" or "pressure controller drift". We will call this set the "fault space", and we will symbolize it by $\theta$. In D-S theory terminology, $\theta$ is the *frame of discernment.*

Consider now a multivalued mapping $\Gamma$ from space E (evidence space) to space $\theta$ (fault space). Each element in E is mapped to a subset of elements in $\theta$, i.e., each piece of evidence can be mapped either to individual hypotheses or to any subset of $\theta$. The objective of this mapping is to distribute belief to each of the hypothetical faults that are related to the observed evidence.

To specify the way that belief from an evidence is distributed in θ, we define the concept of the *Basic Probability Mass Distribution* (BPMD). The BPMD is a set of numerical *beliefs* that result from the distribution of the unit belief of an evidence, to the set of hypotheses in the frame of discernment θ. In other words, the BPMD distributes belief over the set of propositions in the fault space. The BPMD is represented by a set of *Basic Probability Masses* (BPM). For instance, the BPM of A, symbolized as m<A>, represents the belief attributed to A which may be either a single fault or any subset of the fault space. Any unassigned belief is assigned to the entire set θ. Unassigned belief serves to introduce uncertainty into the diagnosis.

Evidential intervals for individual hypotheses [Bel(A),Pls(A)] can be derived from the BPMD. Using this paradigm, the belief and plausibility of a proposition A are given by:

$$\mathrm{Bel}(A) = \Sigma m_i\langle A_i\rangle \qquad (6.4)$$

$$\mathrm{Pls}(A) = 1 - \Sigma m_j\langle \neg A_j\rangle \qquad (6.5)$$

where $A_i \subseteq A$, $\theta = A \cup \neg A$ and $\neg A_j \subseteq \neg A$. Thus, the total belief in A is the sum of beliefs ascribed to A and all the subsets of A.

### 6.2.3. Rules of Combination

Let us now assume that the system has observed two independent sets of evidence, $E_1$ and $E_2$, both subsets of the evidence space E. If $m_1$ and $m_2$ are the two BPMDs induced on θ by the multivalued mappings $\Gamma_1$ ($E_1$ to θ) and $\Gamma_2$ ($E_2$ to θ), the combined BPMD can be computed using Dempster's rules of combination [10]:

$$m(C) = \frac{\sum m_1(A_i)\, m_2(B_j)}{(1-k)}, \quad \text{where } A_i \cap B_j = C, \qquad (6.6)$$

$$k = \sum m_1(A_i)\, m_2(B_j), \quad \text{if } A_i \cap B_j = \phi \qquad (6.7)$$

In other words, under the assumption that evidence comes from independent sources, the BPM of the intersection of $A_i$ and $B_j$ is the product of the BPMs of $A_i$ and $B_j$. The denominator (1−k) is used to normalize against the "lost" belief which otherwise would have been assigned to empty sets.

As an example, consider the frame of discernment $\theta = \{F_1, F_2, F_3, F_4, \zeta\}$ where $F_1$, $F_2$, $F_3$, and $F_4$ are faults and $\zeta$ is the hypothesis of fault-free operation. Assume there are two BPMDs induced in this frame of discernment for respective mappings of $\Gamma_1$ and $\Gamma_2$:

$$m_1(F_1 \cup F_2, F_3, F_4, \zeta, \theta) = (0.48, 0.12, 0, 0.2, 0.2)$$

$$m_2(F_2, F_1 \cup F_3, F_4 \cup \zeta, \theta) = (0.0, 0.7, 0.1, 0.2)$$

The combination of $m_1$ and $m_2$ is depicted in Table 6.1.

Each cell in this table contains the intersection of the corresponding propositions from $m_1$ and $m_2$ along with the product of their individual beliefs. Note that the intersection of any proposition and $\theta$ is the original proposition. Consequently, by applying Eq. (6.6) we have:

$$m(F_1, F_1 \cup F_2, F_3, \zeta, F_1 \cup F_3, F_4 \cup \zeta) = (0.42, 0.12, 0.135, 0.075, 0.175, 0.025)$$

and the corresponding probability intervals, derived from Eq. (6.5), are: $F_1[0.42, 0.765]$, $F_2[0, 0.17]$, $F_3[0.135, 0.36]$, $F_4[0, 0.075]$ and $\zeta$ $[0.075, 0.15]$.

| $m_1$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $F_1 \cup F_2$ | 0.48 | $F_2$ | 0 | $F_1$ | 0.34 | $\emptyset$ | 0.048 | $F_1 \cup F_2$ | 0.096 | |
| $F_3$ | 0.12 | $\emptyset$ | 0 | $F_3$ | 0.084 | $\emptyset$ | 0.012 | $F_3$ | 0.024 | |
| $F_4$ | 0 | $\emptyset$ | 0 | $\emptyset$ | 0 | $F_4$ | 0 | D | 0 | |
| $\zeta$ | 0.2 | $\emptyset$ | 0 | $\emptyset$ | 0.14 | $\zeta$ | 0.02 | $\zeta$ | 0.04 | |
| $\theta$ | 0.2 | $F_2$ | 0 | $F_1 \cup F_3$ | 0.14 | $D_4 \cup \zeta$ | 0.02 | $\theta$ | 0.04 | |
| | | $F_2$ | 0 | $F_1 \cup F_3$ | 0.7 | $F_4 \cup \zeta$ | 0.1 | $\theta$ | 0.2 | $m_2$ |

Table 6.1 Illustration of BPMD Combination

### 6.2.4. Efficient Implementation of Dempster's Rules of Combination

In semiconductor manufacturing, timely diagnosis is critical. It is therefore important to focus on the efficient implementation of Dempster's rules of combination. To achieve this goal, we assume that only single faults can be present at any given time. This reduces the number of mappings to the hypothesis space from $2^{\theta}$ to $\theta$. This assumption is not very restrictive since multiple faults rarely occur independently. It is however possible for secondary faults to be triggered by the occurrence of primary faults. Since these faults always happen together, they can be modeled as a single fault in the fault hypothesis space.

To further increase the speed of computation, Dempster's rules of combination have been implemented as an efficient bit-array operation in Lisp [13].

## 6.3. Overview of the Diagnostic System

Due to the batch character of semiconductor processing, malfunction diagnosis takes place in three consecutive steps. The first step is *maintenance diagnosis*, performed by analyzing the relevant utilities, inventories and maintenance history, such as the time elapsed since the last tube cleaning. The second step is *real-time diagnosis*. This is based on the analysis of real-time sensor data such as temperature, pressure, gas flow, etc. Finally, there is *inline diagnosis*, based on physical and electrical tests performed on processed wafers. An example of the three stages of diagnosis for LPCVD reactors is shown in Figure 6.1. The basic components of the diagnostic system are described below.

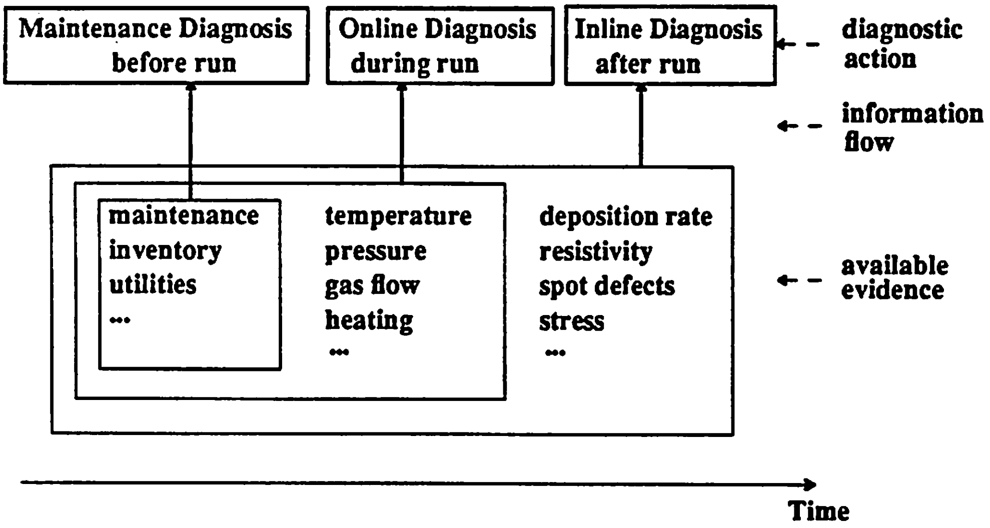


Fig. 6.1 The three stages of diagnosis for LPCVD reactors.

### 6.3.1. Knowledge Base and Inference Engine

The *knowledge base* is a formalized description of the structure and function of the processing equipment. It contains the relationships between the various types of evidence and fault hypotheses. Evidence can originate from maintenance records, from real-time sensor data during processing, and from wafer measurements after processing. Accordingly, the evidence is organized into three groups, namely *maintenance*, *real-time*, and *inline*.

The *inference engine* is a program that performs the mapping from the evidence space E to the fault hypothesis space $\theta$. In terms of the D-S theory, each fault hypothesis is assigned a number between 0 and 1 as a measure of the *belief* about the fault, given the evidence that the system has collected so far. For each new piece of evidence, the D-S theory provides a formal belief accumulation method to calculate the total belief contributed by the separate pieces of evidence.

### 6.3.2. Diagnostic Methodology

During each of the three diagnostic steps, malfunction diagnosis is performed via the resolution of qualitative and quantitative constraints. These two classes of constraints are described below.

#### 6.3.2.1. Qualitative Constraints

The qualitative constraints are rules that specify the normal operation of a piece of equipment. For example, one kind of qualitative constraint determines the texture of the deposited polysilicon under normal conditions. If the texture is different from what is normally expected, that qualitative constraint is violated. Other qualitative constraints are violated when the evidence shows other signs of abnormal behavior. Ideally, the qualitative constraints are compiled from the collective experience of a number of equipment experts.

Qualitative constraints are applied through all three stages of diagnosis. The status of equipment maintenance can be checked by retrieving the relevant information from the database. Furthermore, status patterns originating from binary sensor information are generated during the process. These violations are resolved by assigning belief to the respective fault hypotheses. Evidence derived from observation of the film texture for example, will map to the fault about contamination on the reactor wall. The actual "value" of the texture, i.e. whether the texture is smooth or coarse, determines the final belief assignment on the relevant fault. For another example on the resolution of a qualitative constraint, consider the on/off indicator of a heating element. This indicator should normally be "on" during temperature ramping steps. If it stays "off" for an unusually long time, this would violate an equipment constraint and would clearly indicate a serious problem with the temperature controller.

### 6.3.2.2. Quantitative Constraints

There are two types of quantitative constraints. One class of quantitative constraints is based on numerical equipment models. These models have been created and characterized through experimentation [8,9]. Such constraints will be violated when the actual observations differ substantially from the model predictions. If, for example, the system observes polysilicon layers that are significantly thicker than expected, and if there are reliable temperature and pressure readings, then the conclusion would be that the mass flow controller is out of calibration.

The other class of quantitative constraints applies to the real-time monitoring of sensor readings using statistical process control techniques [14,15]. This class of constraints is described in Section 6.3.3.2.

The quantitative constraints generate belief that is proportional to the seriousness of their violation. More specifically, let $\varepsilon$ be the difference between the measured and the predicted polysilicon thickness. Under the assumption that the equipment is operating properly, $\varepsilon$ is a normally distributed random variable of zero mean and of constant variance $\sigma^2$ (Figure 6.2), which can be estimated from measurements. Under the assumption that the process is operating normally, we can find a symmetrical (1 - $\alpha$) confidence interval for $\varepsilon$:

$$t_{\frac{\alpha}{2},n-1}\frac{s}{\sqrt{n}} \le \varepsilon \le t_{1-\frac{\alpha}{2},n-1}\frac{s}{\sqrt{n}} \qquad (6.8)$$

where $s^2$ is the estimated variance (using n samples), and $t_{1-\frac{\alpha}{2},n-1}s/\sqrt{n}$ is the allowed deviation of $\varepsilon$, calculated using the student-t distribution [14]. If the allowed deviation is set at the +/- 3sigma level, then $\alpha$ is 0.0027. The resulting $6\sigma$ process control limits ensure that, when the equipment is operating properly, we will only get one point beyond the control limits for each 200 observations.

Typically, an alarm is signaled when the residual, $\varepsilon$, exceeds a control limit (*tol*). This, however, is equivalent to the generation of a high belief that will be distributed to the respective faults. Making belief behave as a step function at the *tol* value makes the diagnostic system unstable in the presence of noise. A smoothed belief function that will yield incremental changes in belief associated with the incremental changes in operating conditions is preferred (also see Figure 6.2).

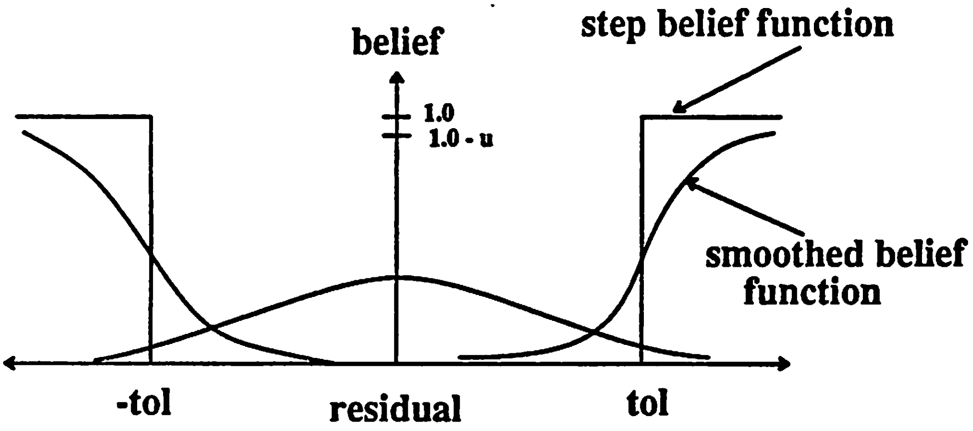


Fig. 6.2 Residual distribution under normal equipment operation. A step belief function is replaced by a smoothed belief function to generate gradual fault belief around the tolerance limit.

The following sigmoid belief function has been used in neural network applications [16].

$$b(\varepsilon) = \frac{(1-u)}{1+\exp^{[-G(\frac{\varepsilon}{tol}-1)]}} \quad , \qquad (6.9)$$

where b(ε) is the belief function related to a constraint. The function above is also called a "squashing" function because its response has maximum sensitivity around *tol*. This means that the belief function will be sensitive at the crossing point and insensitive when ε is either in the normal operating range or far into the abnormal range.

Another benefit from this function is its simplicity. Eq. (6.9) uses an adjustable parameter G to control the sharpness of the belief transition. When G approaches ∞, the belief function becomes a step function at *tol*. Figure 6.3 shows the belief versus G when ε changes from 0σ to 6σ with *tol* set at 3σ. The default value for G is set at 5 because belief is modeled to be smaller than 0.1 when ε is 2σ and larger than 0.2 when ε is 2.5σ (assuming that uncertainty u is zero). This can be seen in Figure 6.3. The factor (1−u), the maximum belief that can be supported by the constraint, is initially set from experience and is different for each source of evidence.

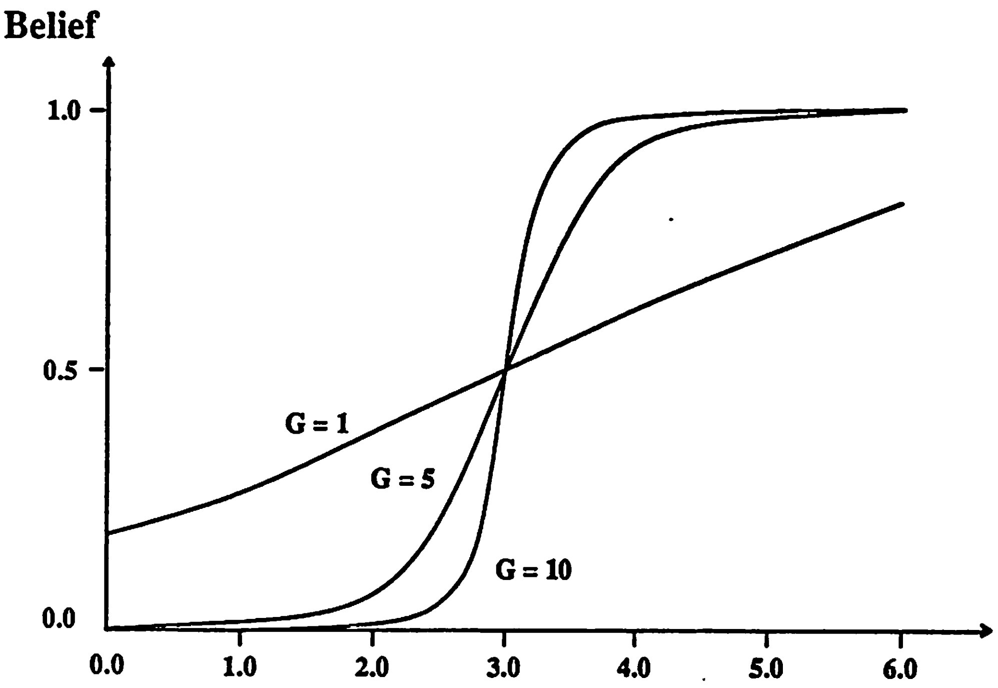


Fig. 6.3 Belief versus G when ε changes from 0σ to 5σ with *tol* set at 3σ

So, given ε and the distribution depicted in Figure 6.2, we can derive the belief that originates from the evidence. When ε > 0, belief b is generated as in Eq. (6.9), and then distributed to the various fault hypotheses.

### 6.3.2.3. Belief Distribution

Once the collective belief from one source of evidence has been generated, it has to be distributed to the various faults, i.e. a BPMD must be formed. The way belief is distributed among the various fault hypotheses merits some discussion.

First, we define the fault hypothesis sets of constraint violations. When $\varepsilon >$ (or $<$) 0, from Figure 6.2, the violated fault hypothesis set is designated as $H^+$ (or $H^-$). $H^+$ (or $H^-$) may consist of a subset of faults from fault hypothesis space. For example, the expression

$H^+ = (F_1 \cup F_2, F_3)$ means that the suspected fault set (induced from $\varepsilon > 0$) contains faults $F_1 \cup F_2$ or fault $F_3$. Now suppose that from equipment maintenance experience, we can associate $H^+$ with a frequency distribution $(F_1 \cup F_2, F_3) = (0.8, 0.2)$. This frequency distribution of $H^+$ indicates that when belief is generated (with the particular $\varepsilon > 0$), $F_1 \cup F_2$ takes the 80% of the belief while $F_3$ takes the remaining 20%.

BPM's are assigned to the fault hypothesis sets $H^+$, $H^-$, and $H^o$ (the set of the remaining faults and the "no fault") in the following manner:

$$m<H^+, H^-, H^o, \theta> = <b, 0, 1-b-u, u>, \quad \varepsilon > 0 \tag{6.10}$$

$$m<H^+, H^-, H^o, \theta> = <0, b, 1-b-u, u>, \quad \varepsilon < 0 \tag{6.11}$$

The evidential intervals implied by this belief assignment indicate that when the constraint residual is greater than zero, the probability that the constraint is actually high is between b and b + u, whereas the probability that it is low is between zero and u. When the residual is less than zero, these intervals are reversed. Note that the uncertainty involved in evaluating the constraint is assigned directly to $\theta$. This means that u supports all relevant hypotheses and cannot be attributed to any particular subset in the frame of discernment. Finally, the BPMD's for all evidence sources are subsequently combined according to Dempster's rules of combination (Section 6.2.3).

Consider the example of a frame of discernment $\theta = \{F_1, F_2, F_3, F_4, \zeta\}$, where $F_1$, $F_2$, $F_3$, and $F_4$ are fault hypotheses and $\zeta$ is the hypothesis of fault-free operation. Assume that there are two constraints with the fault distributions given in Table 6.2.

| | $\varepsilon > 0$ | $\varepsilon < 0$ |
|---|---|---|
| constraint 1 | $(F_1 \cup F_2, F_3) = (0.8, 0.2)$ | $(F_4) = (1)$ |
| constraint 2 | $(F_2) = (1)$ | $(F_1 \cup F_3) = (1)$ |

**Table 6.2** Example for defining belief distribution list for a set of two constraints

When $\varepsilon > 0$, $(F_1 \cup F_2, F_3) = (0.8, 0.2)$ means that the fault hypothesis $F_1 \cup F_2$ gets the generated *belief* multiplied with the factor 0.8 as specified in the belief distribution list. Similarly, $F_3$ gets 0.2**belief* from constraint 1. Further, assume that $b_1 = 0.6$, $u_1 = 0.1$ and $\varepsilon_1 > 0$ for the first constraint. Also assume that $b_2 = 0.7$, $u_2 = 0.2$ and $\varepsilon_2 < 0$ for the second constraint. Then,

$$m_1(F_1 \cup F_2, F_3, F_4, \zeta, \theta) = (0.48, 0.12, 0, 0.2, 0.2)$$

$$m_2(F_2, F_1 \cup F_3, F_4 \cup \zeta, \theta) = (0.0, 0.7, 0.1, 0.2)$$

where $\zeta$ signifies fault-free operation. The combination of $m_1$ and $m_2$ was depicted in Table 6.1. Consequently, by applying the rules of combination again, we have:

$$m(F_1, F_1 \cup F_2, F_3, \zeta, F_1 \cup F_3, F_4 \cup \zeta) = (0.42, 0.12, 0.135, 0.075, 0.175, 0.025)$$

and the corresponding probability intervals, derived from Eq. (6.5), are: $F_1[0.42, 0.765]$, $F_2[0, 0.17]$, $F_3[0.135, 0.36]$, $F_4[0, 0.075]$ and $\zeta$ $[0.075, 0.15]$. Hence, the hypothesis about the fault $F_1$ has the largest support in the group. (This can also be seen in Table 6.1.)

### 6.3.3. Belief Generation in Three Diagnostic Stages

Since semiconductor manufacturing consists of a sequence of batch processes, diagnostic belief is generated in three stages, namely before, during and after the completion of each process.

#### 6.3.3.1. Maintenance Diagnosis

As discussed in Chapter 4, the computerized equipment maintenance records not only keep the field repair reports but also have the preventive maintenance history of each piece of equipment. The maintenance diagnostic module predicts the possible faults by examing the preventive maintenance history of a piece of equipment.

For example, the *tube cleaning history* is treated as a source of evidence for an LPCVD reactor. Tube cleaning is a preventive maintenance activity usually conducted every 10 hours of deposition time (approximately every 4 typical LPCVD runs). Belief of the fault *excessive deposition* is derived from the evidence *tube cleaning history* by treating accumulated deposition time since the last tube cleaning as the *residual* and the 10 hours as the *tol* limit in Eq. (6.9).

#### 6.3.3.2. Real-Time Diagnosis

When performing diagnosis using real-time sensor readings, quantitative constraints are examined by the application of statistical process control (SPC) principles. Next, we focus on how belief is derived from quantitative constraints using SPC concepts.

Real-time monitoring data usually comes in the form of multivariate sampling at a relatively high sampling rates (3 samples/second). It is possible to expect significant auto- and cross-correlation among the samples. After some initial study of the autocorrelation structure of real-time monitoring data, we found that there is not a simple consistent time series model [24] that describes a given data set. So we decided not to process data through time series modeling for this analysis. Other studies [25] are pursuing this topic.

Traditional SPC is based on the application of either Shewhart or cumulative sum (CUSUM) charts. Shewhart charts can monitor large shifts efficiently while CUSUM charts are more applicable when small continuous drifts are present.

Since large shifts and small drifts can occur together, a combined Shewhart - CUSUM control chart technique [15] is used for the analysis of real-time monitoring data. To generate belief as a function of the severity of the tolerance limit violation, we use the statistic

$$z_i = \frac{y_i - m}{\sigma} \tag{6.12}$$

where m is the target mean value, $\sigma$ is the standard deviation of y (which is obtained from previous runs) and $y_i$ is the *ith* observation of a single real-time reading. The CUSUM scheme requires two cumulative sums, one for positive shifts and one for negative shifts. The formulas used are

$$S_{H_i} = \max[0,(z_i - k) + S_{H_{(i-1)}}] \tag{6.13}$$

$$S_{L_i} = \max[0,(-z_i - k) + S_{L_{(i-1)}}] \tag{6.14}$$

where k signifies the deviation (expressed in standard deviations) that we can tolerate. If either $S_H$ or $S_L$ becomes greater than a decision value h, this signals that the process is out of statistical control. The starting CUSUM value, $S_o$ is set equal to zero in a standard CUSUM scheme. (When $S_o$ is not set equal to zero we have a fast initial response (FIR) CUSUM scheme with a ''headstart'' of $S_o$.) The parameter k for the CUSUM scheme is obtained directly as $k = \frac{\Delta}{2}$ where $\Delta$ is a displacement that we want to detect. The chart can also be designed for a given in-control average run length (ARL) [14]. ARL is the average number of observations between alarms. Ideally, we would like ARL to be long when the process is running well and short when there is a deviation.

Another control parameter to be defined is h, the decision value (or the tolerance limit) for $S_{H_i}$ and $S_{L_i}$. Therefore, to generate belief from the CUSUM scheme, we treat h as the *tol* limit and treat $S_{H_i}$ and $S_{L_i}$ as residuals in the quantitative constraint during real-time diagnosis. In general, we have

$$b(S_{H_i}) = \frac{(1-u)}{1+\exp^{[-G(\frac{S_{H_i}}{h}-1)]}} \qquad (6.15)$$

where $b = b(S_{H_i})$ is the belief function related to the CUSUM quantitative constraint of the control parameter which could be temperature, gas flow or pressure.

Similarly, to generate belief from the Schwhart scheme, we treat $3\sigma$ as the *tol* limit and treat the difference between the real control parameter reading and the expected set value for the control parameter as residual.

### 6.3.3.3. Inline Diagnosis

Between the batch process steps the system examines the residuals of the inline measurements. These are compared to the corresponding values predicted by the equipment models. The comparison forms quantitative constraints that are derived from physical, empirical or semi-empirical equipment models. For example, we examine the *residual* of *deposition rate* from a low-pressure chemical vapor deposition (LPCVD) reactor. The *residual* is defined as the difference between the measured deposition rate and the one predicted by the model. In this work, we are using the deposition rate model developed by Lin [8].

$$R(T, P, Q, X) = A\, P^{\alpha} \exp^{\frac{-\Delta E}{RT}} \left[ \frac{1-\frac{k_3}{Q}}{1+\frac{k_3}{Q}} \right] \left[ \frac{1-\eta(P, Q, T, X)}{1+\eta(P, Q, T, X)} \right] \qquad (6.16)$$

where T, Q, P, X are the temperature, gas flow, pressure, and wafer position, respectively. The $3\sigma$ of deposition rate is obtained from the prediction error of the deposition rate model and is used as the *tol* limit for the generation of belief.

### 6.3.4. Information Flow within the Diagnostic System

Figure 6.4 shows the flow of information in the diagnostic system. The maintenance records, historical statistical information and inline measurements are stored in the INGRES relational database [17]. From there, the diagnostic system can automatically retrieve information using the Standard Query Language (SQL) [18]. During the online diagnostic stage, the system applies the SPC rules to evaluate the behavior of the equipment. Finally, in the inline diagnostic stage, the system examines the residuals of the inline measurements and corresponding values predicted by the equipment models. All these pieces of evidence are fed into the inference engine and the corresponding BPMDs are generated and combined.

When the belief associated with a particular fault hypothesis exceeds the defined threshold value of the fault, various actions are initiated. For example, the system might create a report, notify the technician about the problem, display the monitored parameter on the screen, and possibly terminate the process when the problem is considered serious.

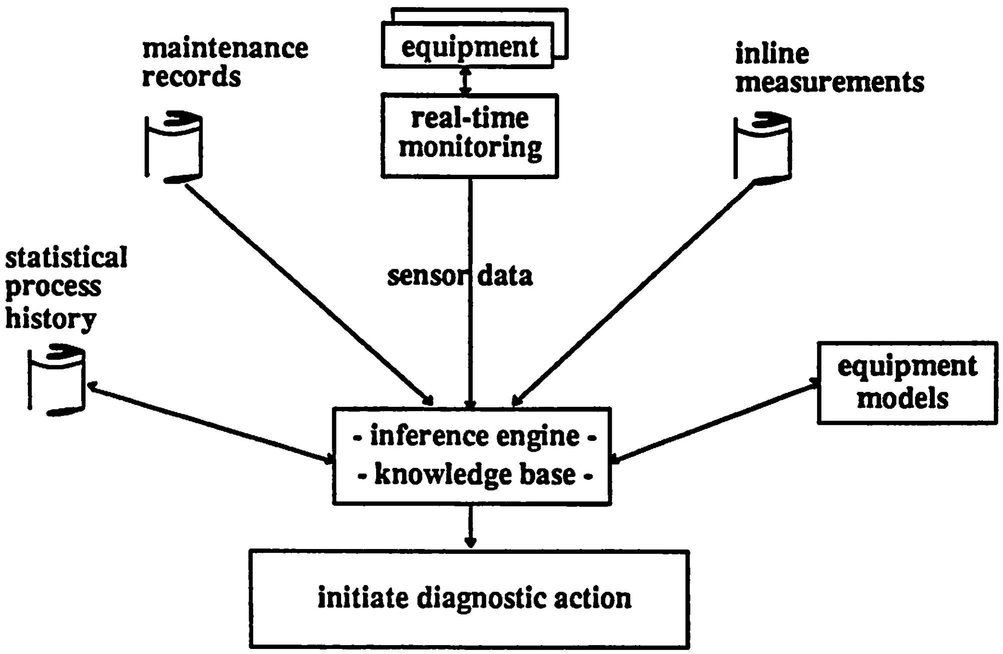


Fig. 6.4 The information flow within the diagnostic system

### 6.4. Adaptive Tuning of Belief Functions

Although the use of smoothed belief functions ensures the stability of the diagnostic system, the final probability intervals depend on the choice of the belief function parameters. If there is no feedback mechanism to refine the evidence-fault association, the diagnostic inferences might suffer from poor choices of *tol*, G, u, and the evidence-fault belief distribution list.

In order to solve this problem, a method has been developed to adjust the belief functions automatically. There are three flexible parameters in the belief function: one is the critical value (*tol*), for which the belief takes the value of (1−u)/2. Other function parameters that can be adjusted are the uncertainty u, and the parameter G which controls the sharpness of the transition.

Adaptive adjustment is also needed for the belief distribution mechanism, which governs the distribution of belief to the various fault hypotheses associated with a piece of evidence. Next, we explain how *tol*, u and the belief distribution list can be set and updated automatically. For this implementation, the parameter G was fixed (see Section 6.3).

### 6.4.1. Adaptive Adjustment of the Tolerance Limit

The limit *tol* has a statistical meaning: it is the +/- 3σ limit around zero for the residual value of each of the quantitative constraints. As such, *tol* can be updated using the observations from the n most recent samples. Obviously, during the collection of these observations, an ''out of control'' group should be excluded from the calculation. Determining the subgroup size n for the calculation of *tol* requires some judgement. In general, the subgroup size should be large, so that we can get a good estimate of the variation. Also, the choice of the subgroup should be such that the appropriate variation is represented, i.e., multiple readings from one wafer should be used to represent wafer-level variation, etc. This problem is also discussed in [19]. In this implementation, *tol* is set at the 3σ level, and it is estimated from all the real-time readings collected for the last three runs of the furnace.

### 6.4.2. Adaptive Adjustment of Evidence Uncertainty and Belief Distribution

To adjust the uncertainty u associated with each piece of evidence and to properly set the belief distribution list, we use historical information from previously observed problems. This historical information however, is stored in a symbolic form as it has been described in Chapter 4. Each maintenance event forms a maintenance report that contains a *symptom* (that the diagnostic system will use as evidence) selected from a prespecified symptom hierarchy. Upon identification and repair, the technician reports the *cause* (i.e. what the diagnostic system recognizes as the "fault"), again using a prespecified fault hierarchy. This record is then stored in a relational database [20]. Symptoms (and their explanation, if available) can be stored in the database by the operator or directly by the diagnostic system. The diagnostic system can later browse through this database and adjust its knowledge base according to the statistical historical information of various symptoms to confirmed faults.

The statistics extracted from the equipment maintenance records can be used to upgrade the uncertainty u and the belief distribution list. For example, when the thickness of the deposited polysilicon film deviates more than 3$\sigma$ from the estimated target, the initial fault-belief distribution list attributes a belief factor of 0.6 to the fault hypothesis of the thermocouple being out of calibration, and a belief of 0.2 to the union of the faults in the pressure controller and flow controller. This evidence carries an initial uncertainty of 0.2. After collecting the downtime statistics in the equipment maintenance records for a while, we found that a better distribution breakdown is 0.8 and 0.1 with an uncertainty of 0.1 for the same evidence-fault mapping.

Since the initial knowledge given by the maintenance engineer comes from experience, in the long run it will not be more correct than the downtime statistics collected in the equipment maintenance records. After enough reports have been collected, statistically significant patterns begin to emerge, and a new belief distribution should be created and installed in the knowledge base. Although we have not yet automated this operation, we expect to do so in the future.

### 6.5. System Implementation

The diagnostic system described in this chapter has been built as a module of the Berkeley Computer Aided Manufacturing (BCAM) system [21]. This module is implemented in CLOS (Common Lisp Object System), an object-oriented programming language built on top of Common Lisp [22]. The BCAM system is written in C, C++ and CLOS and is running under UNIX on SUN 3 and SUN 4 systems.

The structure of the diagnostic module is general and it can accommodate various semiconductor manufacturing processes. The knowledge base was also designed to facilitate easy implementation and modification for different pieces of equipment.

We have used an object-oriented architecture to closely mimic the conceptual hierarchy of the knowledge structure. The basic objects are the *evidence* and the *fault*. The multivalued mapping between evidence and faults is stored in each evidence ''instance''. When a fault belief exceeds its threshold value, general purpose subroutines (''methods'') are dispatched to perform different activities such as the creation of a ''pending'' problem entry in the maintenance database, the notification of the maintenance engineer, the display of relevant evidence (processing parameters) on the screen for real-time monitoring, the termination of the process, etc. The Dempster-Shafer reasoning and the statistical process control routines have also been implemented as independent ''methods.''

The knowledge base also has a hierarchical structure in order to facilitate applications on clusters of similar equipment. For example, the furnace diagnosis module contains knowledge about 16 individual reactors. Some of this knowledge is generic and applies to all reactors, some however is specialized and applies only to the reactors used for chemical vapor deposition. At a lower level we have knowledge that only applies to special types of CVD reactors. Whenever diagnosis is initiated on a specific furnace, the knowledge base uses ''inherited'' generic knowledge enhanced with the appropriate specialized elements.

As part of the greater BCAM framework, the diagnostic module works very closely with other modules. For example, foreign function procedures are provided to invoke a real-time monitoring tool. The diagnostic module also interacts with the INGRES relational database for storing and retrieving information pertinent to equipment operation.

### 6.6. Application Examples

A prototype of the diagnostic system has been applied to a furnace cluster that contains 16 reactors used for silicon oxidation, nitride deposition, LPCVD, etc. Here we present an example drawn from the operation of a low-pressure chemical vapor deposition (LPCVD) reactor used for the deposition of undoped polysilicon. The knowledge base for this particular example contains 22 faults and their mappings to 24 different types of evidence. Appendix 6.1 contains the definitions for the *evidence* and *fault* as classes. Appendix 6.2 then depicts the mapping relationships and their frequency distribution list between the *instances* of *evidence* and *fault*. About 80% of the knowledge base contents is common to all the reactors, with the rest specifically written for the reactor used in this example. In all, the diagnostic module consists of about 7000 lines in CLOS, out of which about 1500 lines are specific to the furnace cluster.

The output of the diagnostic system is presented to the operator in a graphical form as shown in Figure 6.5. On the left side of this graph we start with the beliefs associated with the various faults after examining the maintenance records of the reactor. During the deposition, sensor readings are interpreted and the belief of the various faults is plotted in real-time. Finally, after the inline wafer measurements, the final beliefs are displayed on the right side of the same diagram. For the example presented in Figure 6.5, the system first conducted maintenance diagnosis and found that there may be a slight chance for excessive deposition for the next run. The system reached this conclusion by analyzing the tube cleaning history. Since the belief given to this problem is small (0.13), no action is taken and the process continues.

At the start of deposition, the system examines the time needed to reach a stable deposition temperature. This was found to be longer than usual and it contributed to the belief associated with the faults *thermocouple-out-of-calibration* and *temperature-controller-problem*. During deposition however, the pressure readings were consistently higher than expected. So, the belief of the *pressure-controller-problem* quickly reached a high value (0.76), overshadowing all other faults. Finally, after the wafer measurements, some belief was assigned to *thermocouple-out-of-calibration*, while the *pressure-controller-problem* stayed at the top of the ranked fault list. These inferences were later verified by the maintenance technician.

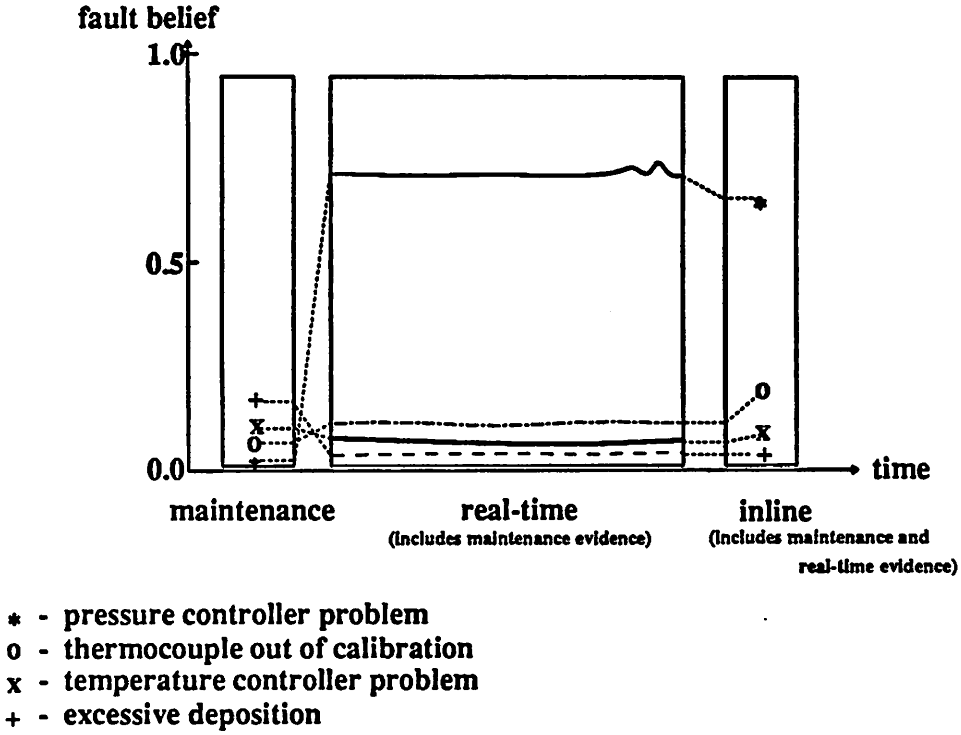


Fig. 6.5 Existing pressure controller problem. Belief of top faults is shown along maintenance, real-time and inline diagnosis stages from a process run on an LPCVD reactor.

A second (in this case simulated) example is presented in Figure 6.6. With the same maintenance and inline data as before, a pressure drift slowly develops during processing. Here, the gradually rising belief of the *pressure-controller-problem* can be directly monitored by the operator during deposition. An alarm can be initiated automatically when this belief crosses a given threshold.

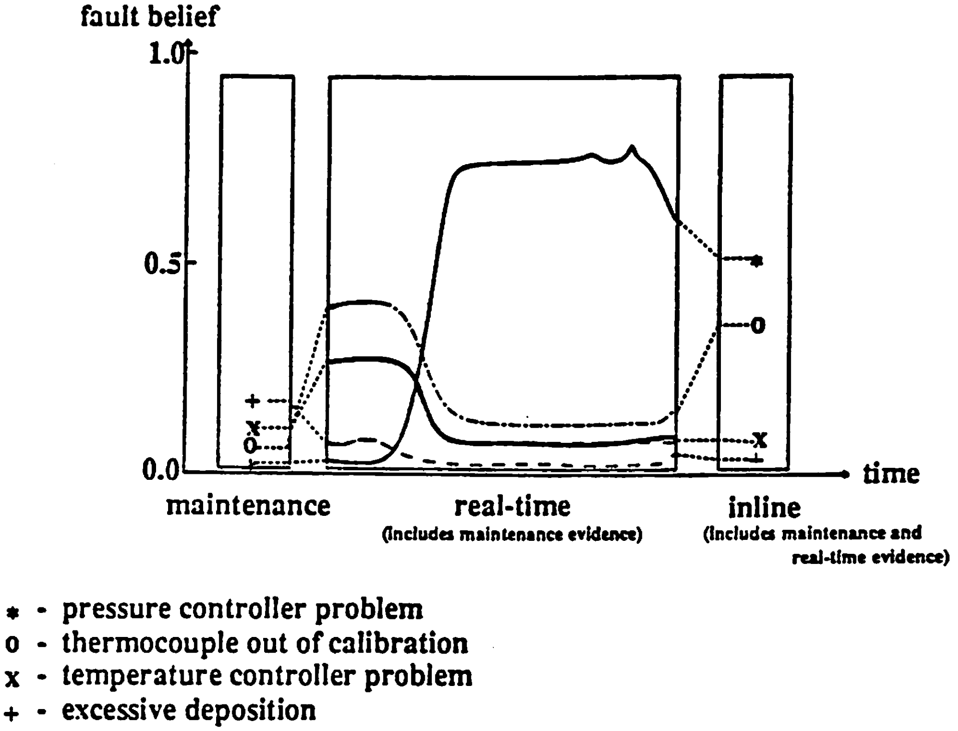


Fig. 6.6 A simulated example of detecting an emerging pressure controller problem during processing.

A third example is shown in Figure 6.7. Here we found no problem at the maintenance diagnostic stage. However, during online diagnosis, the temperature stabilization time was 180 minutes, about three times longer than expected. After the problem was reported to the operator, the long stabilization time was found to be the result of special processing procedures. So, the high fault belief attributed to the *thermocouple-out-of-calibration* and *temperature controller problem* hypotheses were deliberately eliminated. This intervention was later justified by the inline diagnosis; after several wafer readings, the system verified the hypothesis of proper equipment operation.

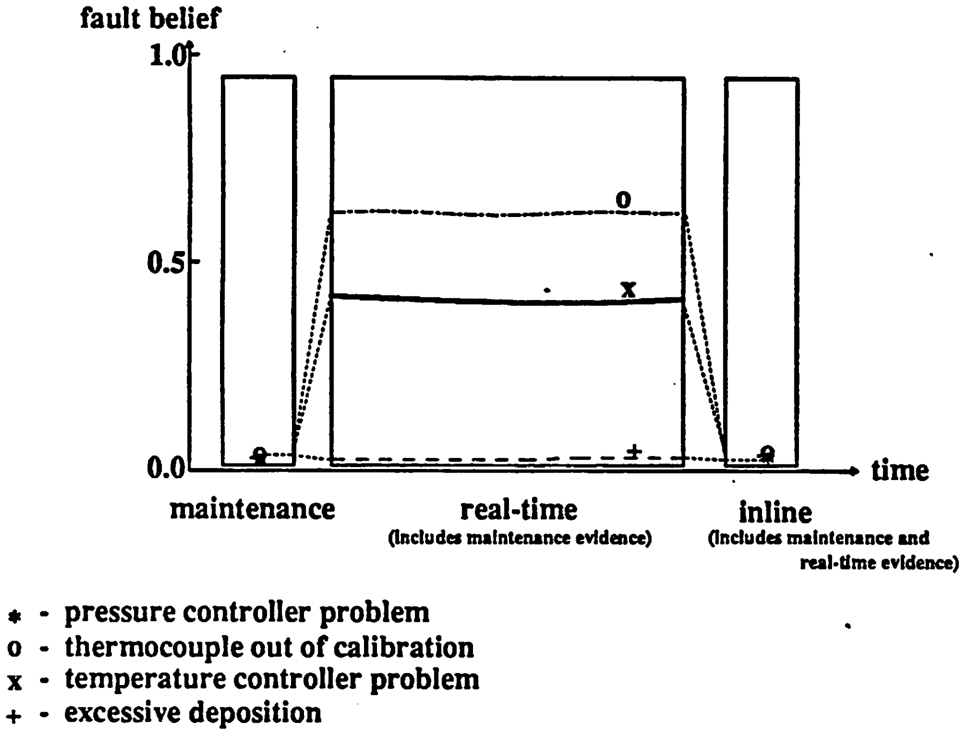


Fig. 6.7 An example of operator intervention during diagnosis. Belief on *thermocouple out of calibration* and *temperature controller problem* were eliminated after consultation with the operator.

The final example is shown in Figure 6.8. The CUSUM belief generation technique is used during real-time diagnosis in this example compared to the Shewhart belief generation technique used in the last three examples. CUSUM can detect small drifts of real-time parameters during a critical process step. Here, the system detects a *temperature controller problem* at the middle of the process run.

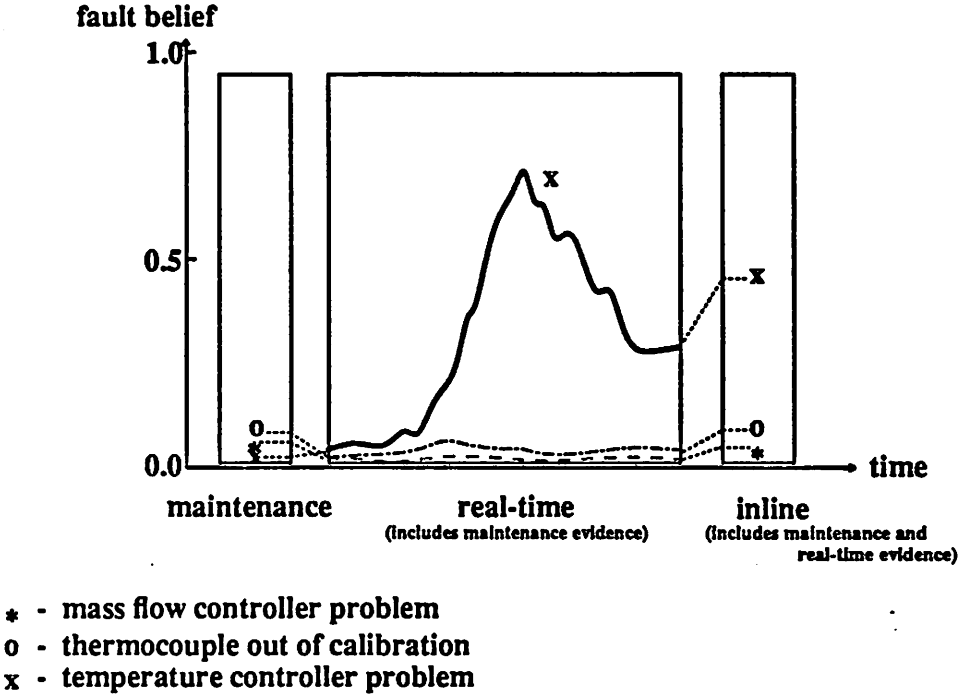


Fig. 6.8 An example of detecting temperature drift by using CUSUM belief generation during real-time diagnosis.

## 6.7. Conclusions

In this chapter we have presented a diagnostic system based on the Dempster-Shafer evidential reasoning method. This method has been shown to work well in batch production environments. This system has the capability to self-adjust by using a computerized equipment maintenance record facility. The algorithms and software used for the automatic adjustment, though still under development, show significant promise.

There are three distinct benefits gained by using the D-S evidential reasoning method. First, the D-S theory provides a way to integrate evidence from multiple, asynchronous sources. This is an important consideration for batch manufacturing processes. Second, it provides a ranked list of faults at any given moment of equipment operation.

The ranked list always incorporates the latest available evidence. Finally, thanks to the implementation of smoothed, continuous belief functions, the D-S method yields fewer false alarms. Because incremental changes in equipment operating conditions result in proportional rise of fault beliefs, this method is also sensitive to slowly emerging faults.

There are, however, four main limitations on the application of the D-S theory for diagnosis. First, significant effort is required for the fine-tuning of belief functions. This problem is partially solved by the self-adapting capability of the knowledge base. A computerized equipment maintenance record system can be used to add evidence-fault mappings that might have been overlooked. This is very important since we are working under the assumption of a "complete" hypothesis space, and omitting evidence-fault mappings will seriously limit the accuracy of the inferences.

Secondly, the approach has limited real-time performance, because of the rather complex operations needed in order to perform BPMD combination. The LPCVD application requires an average of 10 BPMD combinations for each new evidence reading, in addition to the related I/O operations to collect the data. This performance is acceptable here since the sampling rate is rather slow (3 samples per minute). However, for applications such as plasma etching or rapid thermal annealing, the sampling rates are typically set to several samples per second. Although we employed an efficient implementation using bit-array computation, it is still difficult to cope with sampling rates this fast. The temporary solution for this problem is to store the incoming information and let the diagnosis module catch up later. The long-term solution is to run the BPMD combinations on multiple CPUs in parallel, since the order of BPMD combination does not affect the outcome of the diagnosis. In another attempt to improve the real-time performance, we are porting the current version of the diagnostic system to C++ [23].

Thirdly, using the D-S theory under the single fault assumption can be restrictive, especially when multiple, induced faults are present. One way to remedy this situation is to treat a particular combination of faults as a single fault in the fault space.

Finally, a problem arises from the fact that the D-S theory assumes the independence of evidence in the evidence space. This assumption becomes very limiting when we deal with rapid sampling rates, where the data can have substantial auto- and cross-correlations. We are currently experimenting with statistical abstraction techniques for data processing during online and inline diagnosis [24,25]. These techniques will let us focus on processing trends rather than raw data.

## References for Chapter 6

[1] R.C. Dehmel and G.H. Parker, *Future VLSI Manufacturing Environment,* Solid State Technology, May 1987.

[2] D.A. Hodges and L.A. Rowe, C.J. Spanos, *Computer Integrated Manufacturing,* Int. Electronic Manufacturing Technology Symp., September 1989.

[3] D.A. Hodges, W.C. Holton, J. Plummer, B. Wu, *Computer Intergrated Manufacturing for Semiconductors,* JTECH Panel Report on Advanced Computing in Japan, Science Applications International Corporation Japanese Technology Evaluation Program, sponsored by the National Science Foundation, 1989.

[4] C.Y. Fu, N.H. Chang and K.K. Lin, *''Smart'' Integrated-Circuit Processing,* IEEE Transactions on Semiconductor Manufacturing, Dec. 1989.

[5] W. Fong and T.B. Cline, *Use of Expert Systems in Photolithography,* Proc. of SPIE's 1987 Santa Clara Symposium on Microlithography.

[6] S.B. Dolins, A. Srivastava, and B.E. Flinchbaugh, *Monitoring and Diagnosis of Plasma Etch Processes,* IEEE Transactions on Semiconductor Manufacturing, Feb. 1988.

[7] M.A. Kramer, *Malfunction Diagnosis Using Quantitative Models with Non-Boolean Reasoning in Expert Systems,* Vol. 33, No. 1, AIChE Journal, Jan. 1987.

[8] K-K Lin, J. Huang, and C.J. Spanos, *Statistical Equipment Modeling for VLSI Manufacturing,* Symposium on Automated Semiconductor Manufacturing, 176th Electrochemical Society Meeting, Oct. 1989.

[9] G.S. May, J. Huang, and C.J. Spanos, *Statistical Experimental Design in Plasma Etch Modeling,* Submitted to IEEE Transactions on Semiconductor Manufacturing, Jan. 1990.

[10] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

[11] D.S. Vaughan, B.M. Perrin, R.M. Yadrick, and P.D. Holden, *Comparing Expert Systems Built Using Different Uncertain Inference Systems*, Proc. of the Fifth Annual AAAI Workshop on Uncertainty and AI, August, 1989.

[12] G. M. Provan, *A Logical Interpretation of Dempster Shafer Theory, with Application to Visual Recognition*, Proceedings of the Fifth Annual AAAI Workshop on Uncertainty and AI, August, 1989.

[13] J. Yen, *GERTIS: A Dempster-Shafer Approach to Diagnosing Hierarchical Hypotheses*, Communications of the ACM, May 1989, Volume 32, Number 5.

[14] D.C. Montgomery, *Introduction to Statistical Quality Control*, John Wiley & Sons, 1985.

[15] J.M. Lucas, *Combined Shewhart-CUSUM Quality Control Schemes*, Journal of Quality Technology, Vol. 14, No. 2, April 1982.

[16] R.P. Lippmann, *An Introduction to Computing with Neural Nets*, IEEE ASSP Magazine, April, 1987.

[17] C.J. Date, *A Guide to INGRES : A User's Guide to the INGRES Product*, Addison-Wesley Pub. Co., 1987.

[18] *INGRES/SQL Reference Manual*, Relational Technology, 1986.

[19] R.L. Guldi, C.D. Jenkins, etc., *Process Optimization Tweaking Tool (POTT) and its Application in Controlling Oxidation Thickness*, IEEE Transactions on Semiconductor Manufacturing, May. 1989.

[20] N.H. Chang, D.C. Mudie, *An Equipment Maintenance and Repair System Using a Relational Database*, submitted to 1990 IEEE Electronics Manufacturing Technology Symposium.

[21] N.H. Chang, C.J. Spanos, *A CAM Framework and its Application for Monitoring and Diagnosis for VLSI Manufacturing*, SRC IC Manufacturing Workshop, Michigan, 1989.

[22] S. Keene, *Object-Oriented Programming in Common Lisp*, Addison-Wesley, 1989.

[23] G.S. May, C.J. Spanos, *Private Communication.*

[24] A. Pankratz, *Forecasting with Univariate Box-Jenkins Models*, John Wiley & Sons, 1983.

[25] H.-F. Guo, B. Sindahl, C.J. Spanos, *Private Communication.*

# Appendix 6.1

## Definitions for *Evidence* and *Fault* classes

*Evidence* class definition for LPCVD Reactors

```
(defclass evidence ()
 ((name
   :reader evidence-name)
  (participated-arg
;;; participating parameters in this evidence
   :type list
   :accessor participated-arg)
  (associated-faults
;;; the evidence - fault mappings for this evidence
   :initform nil
   :type list
   :accessor associated-faults)
  (uncertainty
;;; the uncertainty defined for this evidence
   :initform nil
   :type float
   :accessor evidence-uncertainty)
   (belief-function
;;; the belief generation function for this evidence
   :initform 0
   :accessor belief-function)
  ;;; G is the sharpness parameter for the belief function
   (G
   :initform 4
   :accessor G-val)
  (equipment
;;; the generic or specific equipment this evidence belongs to
   :initform nil
   :type list
   :accessor equipment)
 )
 (:documentation "The foundation of all the evidence"))
```

Subclasses of ***evidence*** class

```
(defclass maintenance-evidence (evidence)
 ((maintenance-evidence-list
   :initform nil
   :type list
   :allocation :class
   :accessor maintenance-evidence-list)
 )
 (:documentation "Evidence occurs during maintenance period"))

(defclass online-evidence (evidence)
 ((online-evidence-list
   :initform nil
   :type list
   :allocation :class
   :accessor online-evidence-list)
 )
 (:documentation "Evidence occurs during online period"))

(defclass inline-evidence (evidence)
 ((inline-evidence-list
   :initform nil
   :type list
   :allocation :class
   :accessor inline-evidence-list)
 )
 (:documentation "Evidence occurs during inline period"))
```

*Fault* class definition for LPCVD Reactors

```
(defclass fault ()
 ((name
   :initarg :name
   :reader fault-name)
  (equipment
   :initarg :equipment
   :accessor equipment-name)
  (effect
;;; There are two effects: one is a deadly fault which causes equipment
;;; breakdown. The other is a warning fault.
   :initarg :effect
   :accessor effect)
  (support
   :initarg :support
   :initform 0
   :reader support)
  (plausibility
   :initarg :plausibility
   :initform 1
   :reader plausibility)
   (explanation
;;; text documentation when this fault occurred
    :initform 0
    :accessor explanation)
   (supported-evidence
;;; the list of evidence that will lead to this fault
    :initform nil
    :accessor supported-evidence)
   (threshold
;;; fault threshold defined that can initiate diagnostic actions when
exceeded
    :initform nil
    :accessor threshold)
  (threshold-list
;;; the list of current faults that exceed their threshold values
for the fault class
   :initform nil
   :type list
   :allocation :class
   :accessor threshold-list)
  (fault-list
;;; the current fault list that has a above-noise level belief
for the fault class
   :initform nil
   :type list
   :allocation :class
   :accessor fault-list)
  (fault-BPMD-bit-value-list
;;; the current BPMD bit-value list for the fault class
   :initform nil
   :type list
   :allocation :class
   :accessor fault-BPMD-bit-value-list)
  (fault-BPMD-bel-pls-list
;;; the current fault belief-plausibility list for the fault class
   :initform nil
   :type list
   :allocation :class
   :accessor fault-BPMD-bel-pls-list)
 )
 (:documentation "The foundation of all faults"))
```

Subclasses of ***Fault*** class

```
(defclass no-fault-and-theta (fault)
 ((no-fault-and-theta-list
   :initform nil
   :allocation :class
   :accessor no-fault-and-theta-list)
 )
 (:documentation "no-fault and theta in the BPMD list")
)

(defclass online-fault (fault)
  ((online-fault-list
   :initform nil
   :allocation :class
   :accessor online-fault-list)
  )
 (:documentation "An online fault"))


(defclass maintenance-fault (fault)
 ((maintenance-fault-list
   :initform nil
   :allocation :class
   :accessor maintenance-fault-list)
 )
 (:documentation "A maintenance fault"))

(defclass inline-fault (fault)
 ((inline-fault-list
   :initform nil
   :allocation :class
   :accessor inline-fault-list)
 )
 (:documentation "An inline fault that happens after the process run"))
```

## Appendix 6.2

## The Mappings Between the Instances of *Evidence* and *Fault*

### Explanation of Terminology

Evidence : A

;;; A is an instance of the ***Evidence*** class

Class : LPCVD

;;; the class of equipment applied is the LPCVD reactors

Uncertainty : 0.1

;;; the uncertainty of this evidence A is 0.1

Mapping Faults:  
(positive (A, B) 0.6)  
(positive (C) 0.4)  
(negative (D) 1.0)

;;; The evidence - fault mappings  
;;; When $\varepsilon > 0$ for this evidence, the frequency distribution  
;;; is ($A \cup B$, C) = (0.6, 0.4), where the belief generated by  
;;; the evidence is distributed proportionally to the fault $A \cup B$  
;;; and fault C according to the frequency distribution.  
;;; When epsion < 0, the frequency distribution is (D) = (1).

### *The Maintenance Evidence - Fault Mappings*

Evidence : *tube-clean-history_evi*
Class : LPCVD
Uncertainty : 0.1
Mapping Faults:
(positive (*excess-depo-gen*) 1.0)
(negative (*excess-depo-gen*) 1.0)

Evidence : *T/C-calib-history_evi*
Class : LPCVD
Uncertainty : 0.1
Mapping Faults:
(positive (*out-of-calib*) 1.0)
(negative (*out-of-calib*) 1.0)

*The Online Evidence - Fault mappings*

Evidence : *kms-out-cntl_evi*
Class : generic
Uncertainty : 0.15
Mapping Faults:
(positive (*S-heating-element-worn* *S-thermal-insulation*) 1)
(negative (*S-heating-element-worn* *S-thermal-insulation*) 1)

Evidence : *kmc-out-cntl_evi*
Class : generic
Uncertainty : 0.15
Mapping Faults:
(positive (*C-heating-element-worn* *C-thermal-insulation*) 1)
(negative (*C-heating-element-worn* *C-thermal-insulation*) 1)

Evidence : *kml-out-cntl_evi*
Class : generic
Uncertainty : 0.15
Mapping Faults:
(positive (*L-heating-element-worn* *L-thermal-insulation*) 1)
(negative (*L-heating-element-worn* *L-thermal-insulation*) 1)

Evidence : *tempc-out-cntl_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults:
(positive (*temp-controller*) 1)
(negative (*temp-controller*) 1)

Evidence : *temps-out-cntl_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults:
(positive (*temp-controller*) 1)
(negative (*temp-controller*) 1)

Evidence : *templ-out-cntl_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults:
(positive (*temp-controller*) 1)
(negative (*temp-controller*) 1)

Evidence : *tempc-drift_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*temp-controller*) 1)
(negative (*temp-controller*) 1)

Evidence : *templ-drift_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*temp-controller*) 1)
(negative (*temp-controller*) 1)

Evidence : *temps-drift_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*temp-controller*) 1)
(negative (*temp-controller*) 1)

Evidence : *SIH4-drift_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*mass-flow-controller*) 1)
(negative (*mass-flow-controller*) 1)

Evidence : *PH3-drift_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*mass-flow-controller*) 1)
(negative (*mass-flow-controller*) 1)

```
Evidence :  *PH3-out-cntl_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*mass-flow-controller*) 1)
(negative (*mass-flow-controller*) 1)

Evidence :  *SIH4-out-cntl_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*mass-flow-controller*) 1)
(negative (*mass-flow-controller*) 1)

Evidence :  *pressure-out-cntl_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*pressure-controller*) 1)
(negative (*pressure-controller*) 1)

Evidence :  *pressure-drift_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*pressure-controller*) 1)
(negative (*pressure-controller*) 1)

Evidence :  *n2-drift_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults:
(positive (*mass-flow-controller*) 1)
(negative (*mass-flow-controller*) 1)

Evidence :  *n2-out-cntl_evi*
Class : generic
Uncertainty : 0.2
Mapping Faults :
(positive (*mass-flow-controller*) 1)
(negative (*mass-flow-controller*) 1)
```

Evidence : *pump-without-gas_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults :
(positive (*oil-backstream*) 1)

Evidence : *program-on-hold_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults :
(positive (*gas-panel-not-auto*) 1)

Evidence : *temp-cntl-not-on_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults :
(positive (*digital-temp-cntl-off*) 1)

Evidence : *templ-reach-max_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults :
(positive (*L-T/C-open*) 1)

Evidence : *tempc-reach-max_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults :
(positive (*C-T/C-open*) 1)

Evidence : *temps-reach-max_evi*
Class : generic
Uncertainty : 0.1
Mapping Faults :
(positive (*S-T/C-open*) 1)

Evidence : *sih4-valve-leak_evi*
Class : LPCVD
Uncertainty : 0.1
Mapping Faults :
(positive (*toxic-gas-leak*) 1)
(negative (*toxic-gas-leak*) 1)

Evidence : *ph3-valve-leak_evi*
Class : LPCVD
Uncertainty : 0.1
Mapping Faults :
(positive (*toxic-gas-leak*) 1)
(negative (*toxic-gas-leak*) 1)

Evidence : *n2-valve-leak_evi*
Class : LPCVD
Uncertainty : 0.1
Mapping Faults :
(positive (*leak-in-general*) 1)
(negative (*leak-in-general*) 1)

Evidence : *abnormal-stabilization-time_evi*
Class : LPCVD
Uncertainty : 0.1
Mapping Faults :
(positive (*T-C-out-calib*) 0.6)
(positive (*temp-controller*) 0.4)
(negative (*T-C-out-calib*) 0.6)
(negative (*temp-controller*) 0.4)

*The Inline Evidence - Fault Mappings*

Evidence : *inline-thick-drift_evi*
Class : LPCVD
Uncertainty : 0.05
Mapping Faults:
(positive (*T-C-out-calib*) 0.7)
(positive (*pres-controller* *MFC* *excess-depo-gen*) 0.3)
(negative (*T-C-out-calib*) 0.7)
(negative (*pres-controller* *MFC* *excess-depo-gen*) 0.3)

# *Chapter 7*
# *Conclusions and Future Work*

This work focuses on the development of a Computer Aided Manufacturing (CAM) framework and its application on equipment monitoring, maintenance and diagnosis for semiconductor manufacturing. The monitoring module is now in common use in Berkeley Microlab. The maintenance module is now under beta test in the Berkeley Microlab, while a prototype of the diagnostic system has been demonstrated for LPCVD of undoped polysilicon. A summary of the major results and future directions follows.

## 7.1. Major Results

Real-time monitoring of equipment operation is vital to the success of today's semiconductor manufacturing. It provides a tool for lab personnel to detect problems that cannot be detected when only *inline* measurement data are taken. We have built a generic monitoring tool that provides many engineering utilities for interactive real-time monitoring. A method for data abstraction is developed to deal with the growing amounts of data from equipment monitoring. This method extracts the essential statistical information from real-time monitoring data and saves it in the database for easy correlation and trend analysis.

A computerized equipment maintenance record keeping system has been developed that combines a form-based user interface with a relational database for recording preventive maintenance and field repair events. The semantics of preventive maintenance (PM) and repair events are formalized to create unambiguous and clear maintenance reports.

Organizing equipment PM and failure information symbolically in a database offers several advantages. Accumulated information is automatically indexed to aid diagnosis of failures as they occur by quickly producing a history of similar failures. Equipment failure information is available to other utility programs for display and statistical analysis, for summaries such as preventive maintenance intervals, mean time between failures, performance trends, etc. Charts to summarize equipment downtime and failure frequencies are easily produced.

The system is now in use at the Berkeley Microfabrication Facility. Lab technicians have entered knowledge of over a hundred different pieces of equipment into the database. This application has resulted in significant improvement in the way information is used for the management of preventive maintenance and equipment repairs.

We have also developed a diagnostic system that employs the Dempster-Shafer (D-S) evidential reasoning technique to conduct malfunction diagnosis on semiconductor manufacturing equipment. This is accomplished by combining evidence originating from equipment maintenance records, from real-time equipment data, and from measurements on the finished product. Using this information, the causes of equipment malfunctions are inferred through the resolution of qualitative and quantitative constraints. The qualitative constraints describe the ''normal'' operation of the equipment. The quantitative constraints are numerical models that apply to the manufacturing step in question. These models are specifically created and characterized through experimentation and statistical analysis. The violation of these constraints is linked to the evaluation of continuous ''belief functions'' for the calculation of the ''belief'' associated with the various types of failure. The belief functions encapsulate the experience of many equipment maintenance specialists. Once created, the belief functions can be fine-tuned automatically, drawing from historical maintenance records. These records are stored in symbolic form in order to facilitate this task.

A prototype of this diagnostic system was implemented in an object-oriented programming environment. This implementation enables knowledge and functionalities to be shared by different pieces of manufacturing equipment. The D-S diagnostic method was first applied to a reactor used for Low Pressure Chemical Vapor Deposition (LPCVD) of undoped polysilicon films. Experimental results indicate that this diagnostic system is sensitive, stable, and accurate.

## 7.2. Future Work

### 7.2.1. A Unified Equipment Maintenance and Repair System

The equipment maintenance record keeping system lacks the capability to guide the user through step-by-step diagnosis. While the off-line equipment maintenance aid guides the user through step-by-step diagnosis, it does not store the failure history in an organized database as the equipment maintenance record keeping system does. It would be interesting to pursue an approach that combines the strength of the equipment maintenance aid and the equipment maintenance record keeping system. Such a system would help the user perform step-by-step diagnosis as well as record the information in the database by using a case-based diagnostic reasoning method [1].

### 7.2.2. Improving the Diagnostic System

The *evidence* and *fault* instances are defined in the knowledge base of the diagnostic system. Although a bigger set of *fault* and *evidence* instances are defined in the database of the maintenance record keeping system, any changes of the instances in either system will cause knowledge inconsistency problems. A better solution is to unify and store these instance definitions in the maintenance database and provide the diagnostic system with a direct link to the database. This way, the knowledge base will contain a dynamic and accurate representation of the manufacturing process.

Of the three diagnostic stages, the *inline* diagnostic stage could produce more accurate fault belief by solving the empirical equipment models in reverse [2].

The interactive interface for the current diagnostic system can be further improved by linking the fault belief display to visual alarm functions.

Finally, while the diagnostic system works sufficiently for the LPCVD application, rapid process steps such as rapid thermal annealing or plasma etching might pose a problem. The speed of the diagnostic system is not sufficient for conducting diagnosis of rapid process steps. The efficiency of the diagnostic system can be improved by implementing it in C++ [3].

## References for Chapter 7

[1] K.J. Hammond and N. Hurwitz, *Extracting Diagnostic Features from Explanations,* Case-Based Reasoning Workshop, May, 1988.

[2] C.J. Spanos, *Using Regression Equations for Model-Based Diagnosis in the Berkeley Computer-Aided Manufacturing System,* Workshop on Intelligent Diagnostic and Control Systems for Manufacturing, Boston, MA, July 1990.

[3] G.S. May, C.J. Spanos, *Private Communication.*

# APPENDIX A

# PUBLICATIONS

[1] N.H. Chang, C.J. Spanos, *Continuous Equipment Diagnosis Using Evidence Integration*, submitted to *IEEE Transaction on Semiconductor Manufacturing*, November 1990.

[2] N.H. Chang, D.C. Mudie, *An Equipment Maintenance and Repair System Using a Relational Database System*, submitted to 1990 IEEE/CHMT International Electronics Manufacturing Technology Symposium.

[3] N.H. Chang, C.J. Spanos, *Continuous Diagnosis of Low-Pressure Chemical Vapor Deposition Reactors Using Evidence Integration*, 1990 Symposium on VLSI Technology.

[4] N.H. Chang, C.J. Spanos, *Chronological Equipment Diagnosis with Evidence Integration*, Invited Paper to the SPIE Conference on Applications of Artificial Intelligence VIII, 1990.

[5] N.H. Chang, C.J. Spanos, *A CAM Framework and its Application for Monitoring and Diagnosis in VLSI Manufacturing*, SRC Manufacturing Workshop at University of Michigan, August 1989.

[6] C.Y. Fu, N.H. Chang, K.-K. Lin, *"Smart" IC Processing*, *IEEE Transaction on Semiconductor Manufacturing*, November 1989.

[7] N.H. Chang, P. Rothe, *GEMS - Generic Equipment Maintenance System*, ISMSS in SEMICON/West, May 1989.

[8] N.H. Chang, K.-K. Lin, C.Y. Fu, D.A. Hodges, *BIPS Application for LPCVD Polysilicon*, *Texas Instruments Technical Journal*, Winter 1987, pp. 30-38.

Best Paper Award for the Texas Instruments (TI) Paper Competition in TI conference under AAAI 1987.

[9] C.Y. Fu, N.H. Chang, K.-K. Lin and D.A. Hodges, *Expert System for IC Computer-Aided Manufacturing*, SEMICON/EAST, Boston, September 1987, *Technical Proceedings*, pp. 21-30.

[10] N.H. Chang, C.Y. Fu, K.-K. Lin, D.A. Hodges, *Real-Time Processing with Knowledge-Based System in the IC Manufacturing Environment*, AAAI-87 workshop on *Real-Time Processing in Knowledge-Based Systems*.

[11] K.-K. Lin, C.Y. Fu, N.H. Chang and D.A. Hodges, *Recipe Generator for LPCVD Deposition*, Symposium on Automated Semiconductor Manufacturing, 172nd Electrochemical Society Meeting, Honolulu, Hawaii, October 1987.