

DIAGNOSTIC YIELD CHARACTERIZATION EXPERT (DYCE)

A Diagnostic Knowledge Based System Shell for Automated Data Analysis

**Donald D. Pierson
George J. Gallant**

IBM Corporation
1000 River Road
Essex Junction, VT 05452
dpierson@btvlabvm.vnet.ibm.com
gallant@btvlabvm.vnet.ibm.com

Abstract

Diagnostic Yield Characterization Expert (DYCE) is a knowledge based system shell used for automated data interpretation in semiconductor manufacturing. Using a combination of artificial intelligence techniques, DYCE assists manufacturing and development personnel to diagnose problems by automatically interpreting hundreds of pieces of data, just as a human expert would do. Diagnosis time has been reduced from several hours to minutes. DYCE's inference engine has generic meta-rules to represent the heuristic problem-solving knowledge of the experts; this permits the representation of hundreds of traditional expert system rules with a few dozen meta-rules. This generic shell architecture has important benefits in maintaining an expert system in a dynamic environment without the need for a knowledge engineer or AI training.

Task Description

Background

The introduction of a new generation of semiconductor technology can cost hundreds of millions of dollars and take three or more years to develop. The detection and prevention of defects in the semiconductor manufacturing and development processes are critical to providing high yield and product reliability within cost objectives. Information from many data sources must be interpreted by numerous manufacturing personnel in order to understand the causes of any process problems during production. As technology advances to yet higher and higher levels of integration, the amount of data required for interpretation increases dramatically.

It is typical to have manufacturing support groups whose missions are to design appropriate testing and

analyze the results to insure that product quality goals are met. In reacting to yield loss situations during the production process, manufacturing personnel study information from process tool monitors, visual microscope inspections, product flow information, and routine failure analysis evaluations. Proper diagnosis and resolution of the process defects require the cooperation of experts from numerous manufacturing departments, as well as from the data analysis (characterization) support groups.

The use of knowledge based systems for semiconductor process diagnosis is not new (Roundtree 1986) (Pan and Tenenbaum 1986). Methods ranging from manual database queries (Sher, et al. 1990) to automated discovery (Saxena 1993) have been used. DYCE is a belief-based system using compiled knowledge for automatically interpreting electrical test data taken during semiconductor processing. Compiled knowledge refers to knowledge that experts acquire through experience (Punch 1992).

Goals

The primary objective of DYCE was to automate the analysis of hundreds of electrical tests performed during the manufacturing process. The motivation was two-fold. (1) The support groups became overwhelmed with data to the point of not being able to react to all of it. Only critical data was being monitored in real time, with the rest of the data stored for later analysis. (2) Market pressures precluded the addition of more human resource. Automated data analysis was required to handle the increasing amount of data and insure product quality without adding people.

Another objective was the implementation of the knowledge based system as an inclined shell, enabling DYCE to be used in numerous manufacturing areas without the need for customization. DYCE also was required to function in a dynamic environment, where production rules, specifications, and tests are constantly changing.

Application Description

There are two optional modes of operation of DYCE, automatic and interactive. Users have the option of deploying their DYCE applications in one or both modes.

Automatic mode: The automatic mode can be triggered each time the results of an electrical test is loaded into the manufacturing database. In this case, the knowledge base system is automatically consulted and the conclusions are sent to a list of subscribers (engineers and technicians) via electronic mail (see Figure 1: Automatic DYCE Diagnosis). Users have been delighted to have the computer send them results through the mail.

```

SUBJECT: Lotname: K33002H0S10L (6301)

The following problems were found by DYCE:

WAFER ID      PROBLEM
ALL           HIGH ACLV
ALL           LOW POLY LWB
MCC54VUQ      HIGH ACLV
MEA54SXQ      HIGH ACLV
MEA54SXQ      LOW DEFECT PROBE YIELD
MIC54WEQ      LOW DEFECT PROBE YIELD
MIC54WEQ      LOW POLY LWB
  
```

Figure 1. Automatic DYCE Diagnosis via e-mail.

Interactive mode: The interactive mode is initiated by a user who is looking for diagnosis of electrical tests from a specific batch of product which has test results already in the manufacturing database. The KBS analysis is triggered by the user from menu-driven panels. Product name and test identification are input to start the process. The resulting diagnoses are displayed on the DYCE Conclusion Panel, as shown in the bottom half of Figure 2. In the interactive mode, the DYCE inferencing defaults can be changed to do customized reasoning. In addition, supporting information in the form of exception reports, graphics, and explanations of DYCE conclusions are accessible during this mode (see the top half of Figure 2: DYCE Conclusion Panel). The exception reports show only data that is not meeting specification as defined by the expert. Previously,

the user had to sift through all the data, including acceptable results, in order to find the unacceptable results. Examples of graphic output are 3D maps of test values across entire product batches, contour charts showing patterns on individual wafers, and histograms of data distributions. Wafer maps are plots of test readings that were measured across each wafer. The wafer maps are used by the experts to study regionality problems. This supporting information is defined by the expert to be pertinent to the problems concluded by DYCE. Typically, these charts are used for additional trouble-shooting or presentations. The user can choose from a list of charts and graphics to display the additional information. The advantage of having the knowledge based system generate the charts is that fewer unneeded reports are made.

There is an option to review which rules fired during the consultation. This helps the user understand how DYCE came to the conclusions presented on the DYCE Conclusions Panel or in the electronic mail.

```

----- DYCE CONCLUSIONS PANEL -----
+-- SELECTION MENU --- 01:14 PM --- 3/03/15 ---+
|
| DYCE Knowledge Based System conclusions.
|
| R - Displays the EXCEPTION REPORT
| S - Displays the SNAPSHOT REPORT
| A - Displays the ALL PARAMETERS REPORT
| M - Displays the ALL WAFER MAPS REPORT
| P - Display PICTURES of each choice
|
| PF1 - HELP          PF3 - Return
| PF7 - Scroll up     PF8 - Scroll down
|
+-- PRESS 'ENTER' AFTER COMPLETING SELECTIONS --+

-----
KBS CONCLUSIONS LOT: K33002H0 PNPID 6301
-----
Lot ID: K33002H0

WAFER ID      PROBLEM                                CONFIDENCE
                                           FACTOR
ALL           HIGH ACLV                                0.94
ALL           LOW POLY LWB                             0.90
MCC54VUQ      HIGH ACLV                                0.94
MEA54SXQ      HIGH ACLV                                0.94
MEA54SXQ      LOW DEFECT PROBE YIELD                   0.96
MIC54WEQ      LOW DEFECT PROBE YIELD                   0.96
MIC54WEQ      LOW POLY LWB                             0.90
  
```

Figure 2. DYCE Conclusions Panel (interactive mode)

Access another KBS

Another helpful feature of DYCE is the ability for the user to seamlessly interact with another knowledge based system called DEPICT (Digitized Expert PICTures)¹. The user does not need to exit the DYCE knowledge based system to initiate the DEPICT consultation. DEPICT contains digitized images of previously discovered semiconductor defects and textual information about how to eliminate them. This option allows the user direct access to the institutional knowledge about any product problems just diagnosed by DYCE.

DYCE Architecture

The architecture of DYCE is a unique combination of knowledge representation techniques which allows the domain expert to modify knowledge based system rules at run-time, without the need for a knowledge engineer to modify source code and recompile the knowledge base system.

The artificial intelligence techniques combined in DYCE are pattern matching, fuzzy logic, certainty, and normalization. These enable the use of generic meta-rules in the inference engine and are described in subsequent sections of this paper. The use of pattern matching techniques allows heuristic knowledge to be represented and processed by computers. Pattern matching is used to recognize important information in data in a relatively fast, efficient manner. Fuzzy logic techniques enable the inference engine to deal with missing or incomplete data. The knowledge base system can continue with its goals, rather than stop for lack of information. The final conclusions are based on the amount of evidence in the data. Confidence in the knowledge base system conclusions is measured with certainty factors. Certainty factors represent a weighting technique that is applied to the data. Normalization techniques enables simplification of the pattern matching scheme. This allows minimization of the total number of rules required by the inference engine to represent application-independent knowledge.

DYCE Software Shell

A system overview is presented in the schematic representation in Figure 3. DYCE was developed using

ISPF/PDF² utilities, PL/I, 'C', and an object-oriented AI shell (ART-IM³). DYCE is currently deployed on an MVS⁴ mainframe environment. Access to DYCE is provided as an option on a ISPF/PDF menu. This deployment strategy enables users to access the system easily as part of their routine data analysis.

Selecting the DYCE option invokes a TSO C-list. The C-list checks for database availability, allocates files, runs routines to test the terminal for graphics capability, allocates the external tables, and invokes the ART-IM AI shell. All routines use ISPF/PDF services to display menus and selection lists. The programs provide the graphics interface for the user with GDDM.⁵

The automatic and interactive software routines enable the two modes of operation of DYCE. The automatic software portion essentially gathers the appropriate data from the database(s) and formats it into a repository for use by the inference engine. The inference engine's meta-rules, which represent domain experience, match with specific instances of patterns in the repository to formulate diagnoses.

Knowledge Representation

The knowledge engineering bottleneck of getting application knowledge into expert systems is well known. The DYCE shell was designed to minimize this problem. Application knowledge is input into DYCE by entering information in simple table format. This feature offers a practical method for easily modifying the system to meet changing requirements in continually evolving applications.

Table Driven: There are three tables required for inputting knowledge into DYCE. One table describes the patterns to be recognized in the data. This table is called the *expected* data pattern table. An example of this table is shown in Table 1. The second table (example is shown in table 2) is used to input the specifications for each measurement or variable. A third table (not shown) represents the rules for automated generation of supporting information (graphics, trend charts, and wafer maps). It is similar in format to the *expected* data patterns table.

¹ DEPICT is an internal IBM knowledge based system shell.

² ISPF/PDF (Interactive System Productivity Facility / Program Development Facility) is a trademark of IBM Corp.

³ ART-IM (Automated Reasoning Tool for Information Management) is a trademark of Inference, Corp.)

⁴ MVS (Multiple Virtual Systems) is a trademark of IBM Corp.

⁵ Graphical Data Display Manager (GDDM) is a trademark of IBM Corp.

System Overview

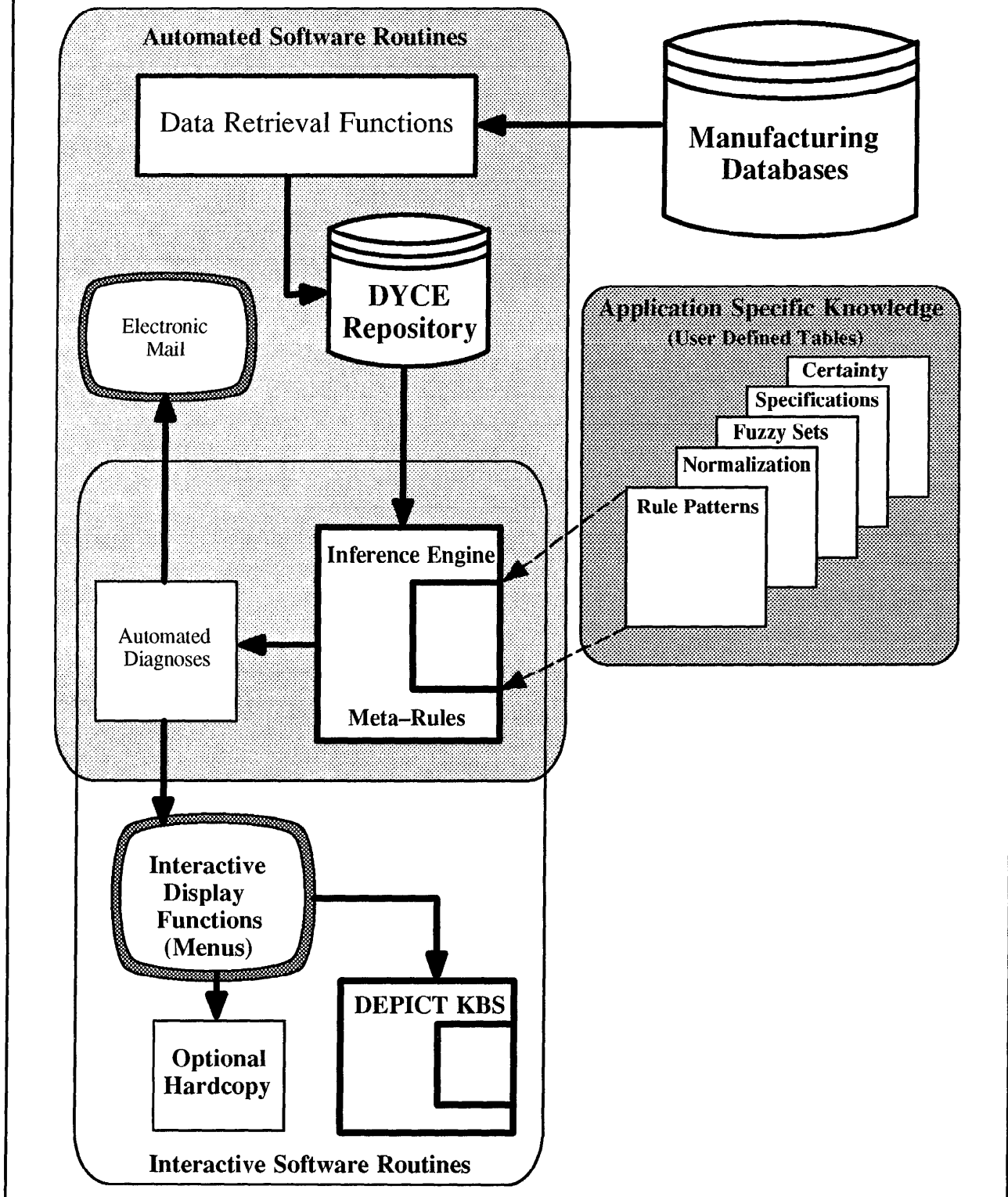


Figure 3. This figure is a schematic representation of the architecture of DYCE.

Table 1. Example of DYCE Rule Representation (Expected table).						
PROBLEM TO BE DIAGNOSED	VARIABLE ID.	VARIABLE NAME	LOW LIMIT	HIGH LIMIT	CERTAINTY FACTOR	RULE CODE
PROBE-FAIL	311	PROBE-TEST	-5	-3	0.8	0
PROBE-FAIL	311	PROBE-TEST	-3	-2	0.4	0
HIGH LWBIAS	311	PROBE-TEST	-5	-2	-1	0
HIGH LWBIAS	312	POLYSILICON RS	-2	2	0.9	A
HIGH LWBIAS	313	POLYSILICON BIAS	2	3	0.9	A
LOW LWBIAS	311	PROBE-TEST	-5	-2	-1	0
LOW LWBIAS	312	POLYSILICON RS	-2	2	0.9	B
LOW LWBIAS	313	POLYSILICON BIAS	-3	-2	0.9	B
EACH RULE PATTERN IS ADDED IN AN ADDITIONAL ROW						

Pattern matching: The *expected* data pattern table essentially describes the rules (patterns) that DYCE uses to determine conclusions about the data (see Table 1: DYCE Rule Representation). These are the application-specific patterns that are matched in the meta-rules of the inference engine. The table is a representation of how a human expert thinks when determining conclusions from the values and relationships of the data. The domain expert uses this table to input knowledge about each problem to be diagnosed, variables known to affect each problem, and how much weight (certainty) is to be applied to each piece of information that exists in the database.

Normalization: Within DYCE, each variable's value is normalized with respect to its target value. The normalization reduces the coding requirements of the inference engine, simplifies end-user maintainability, and allows DYCE to be implemented as an inclined KBS shell. Without normalization, each variable would need a rule associated with its range of real values. In the semiconductor applications, that means a 10X increase in the number of rules is required to represent the knowledge. The basis of the normalization method is similar to statistical process control limits, as shown in Table 2. The variable's target or nominal specification is considered to be *zero*. Each variable's values are normal-

ized over a range from '-5' to '+5' with respect to the target value. Values inclusive of '-2' to '+2' are "*within specifications*." There are low and high control limits at '-1' and '+1', respectively. Values greater than '+2' are categorized (somewhat arbitrarily) into three levels of *high* and values less than '-2' are classified into three levels of *low*. The heuristic values as shown in Table 2 are also arbitrary. Actually, the whole normalization method is arbitrary; any set of numbers and names would suffice. These values were chosen to keep the table representation scheme simple and related to the semiconductor applications.

Fuzzy logic: There are eleven membership sets, or categories for the semiconductor data analysis applications. The number of fuzzy sets is arbitrary; more categories allow finer granularity. However, this number adequately meets the requirements in these applications. The normalization function is a continuous, but not necessarily linear, function of the variable's possible values. Depending on the variable's real value, the normalized value will cause the variable to be a member of one of the eleven categories.⁶ In the example in Table 2, variable #312 is within specification for values between 20 ohms/sq. and 60 ohms/sq. ('-2' and '+2' for normalized values).

⁶ The membership function for each category is a step function in these applications.

Table 2. Example of DYCE Data Specifications Table.											
Variable	Lower Censor Limit	Lower Action2 Limit	Lower Action1 Limit	Lower Spec Limit	Lower Control Limit	Target	Upper Control Limit	Upper Spec Limit	Upper Action1 Limit	Upper Action2 Limit	Upper Censor Limit
Normalization	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5
Heuristic Values	very low	medium low	low	within spec	in control	on target	in control	within spec	high	medium high	very high
311 (%)	0	25	50	75	80	85	90	94	96	98	100
312 (ohm/sq)	0	.1	1	20	25	40	55	60	100	400	999
313 (micron)	0	.01	.02	.03	.04	.05	.06	.07	.08	.09	1
Specifications for each variable are defined in separate rows.											

Meta-rules: The meta-rules in the inference engine are made up of generic premises and associated actions. Think of this as:

WHENEVER <SOMETHING IS MATCHED> ,
THEN DO <WHATEVER THE ACTION IS.>

This is represented in ART-IM by asserting instances of schemas⁷ using rows from the *expected* data patterns and matching them against schemas populated by *actual* data patterns from the repository. When a variable's normalized value (from the repository) is within the range the domain expert listed in the expected data patterns table, the schemas are considered to be matched. The generic premise is satisfied for this instance. This causes the rule to fire and carry out the action, which is also part of the expected data pattern. Typically, the action is to apply additional certainty to a specific diagnosis.

Example:

Assume that the real value from the semiconductor electrical testing for variable #311 is found to be 30%. The software takes this value from the database, normalizes it using Table 2 (DYCE Data Specifications Table), and puts it in the repository. In this case, the normalized value is -3.6 and it becomes a member of the fuzzy set with the heuristic value of "medium low." The variable and its normalized value make up part of the information that gets loaded into an *actual* schema from information in the repository. The expert put a data pattern into the

expected data patterns table that deals with variable #311 (see the first row of Table 1). Therefore, there is an *expected* schema that will potentially match with the *actual* schema. Since the normalized value (-3.6) of variable #311 is within the specified range (between '-5' and '-3') in Table 1, the meta-rule's generic premise is matched and the rule fires the action.

The first line (pattern) in table 1 is:

PROBE-FAIL 311 PROBE-TEST -5 -3 0.8 O

The meta-rule in DYCE's inference engine interprets this first pattern in a way similar to the following syntax:

"Whenever the yield of variable #311 (PROBE-TEST) is between the values of 0% and 50%, then there is probably (0.8 evidence) of the PROBE-FAIL problem."

After the generic meta-rule fires, the problem "PROBE-FAIL" is given a certainty of 0.8. The "rule code" at the end of all the *expected* data patterns allows single or double premise patterns. A rule code 'O' tells the inference engine to treat that row as a single premise pattern. Rows with the same rule code other than 'O' are combined as double premise patterns and are matched only if both patterns are true. See Table 1 for more examples.

Certainty factors: For diagnoses involving several variables, the evidence needs to be tallied for all the matched patterns. The resulting certainty factor is

⁷ Schemas are hierarchical representations of information about objects used in ART-IM.

determined by the MYCIN⁸ algorithm for combining certainty factors. If the final certainty exceeds the certainty threshold specified in DYCE (either by default values or input by the user during an interactive session), then the problem is output as a DYCE diagnosis.

It is important to point out that DYCE matches problems only when the real data is in the appropriate range. This is similar to a human expert recognizing that data is "*within-spec*" or "*out-of-spec*."

Knowledge about new problems can simply be added as a new line in the DYCE expected patterns table. New variables, or changes to a variable's specification limits are updated in the DYCE data specifications table. In this manner, the knowledge based system is easily kept current without the need of an AI specialist.

Why multiple AI techniques?

Earlier experience with conventional data analysis software showed us that methods for processing the raw data into meaningful information were popular with our users. For example, statistical information in appropriate columnar formats helped the experts recognize problems more quickly. Many of the popular routines eventually became menu selections for ease of use.

As the amount of data increased, the amount of human resources usually remained constant (or sometimes decreased). Critical data was prioritized for real time analysis, with less important data reviewed as time permitted or if the key indicators could not explain processing problems. It was recognized that artificial intelligence techniques could help automate some of the interpretation effort and help decrease the risk of missing important diagnoses.

Our first attempt using a traditional rule-based only approach taught us that hundreds of rules were needed to make a useful system. Further, we noticed that constant rule modification was required to keep pace with the dynamic environment of manufacturing process improvements. We realized that expert systems that were brittle, hard to maintain, and required constant knowledge engineering resource were likely to become obsolete. After all, why not just apply the resource involved in developing the knowledge based system (expert, knowledge engineer, and programmer) directly to the manufacturing diagnosis task? The answer was to provide knowledge based system shells that the domain expert can easily modify without AI training or programming skills.

The combination of the techniques described in earlier sections allowed us to successfully develop DYCE as an application specific shell. The inference engine architecture is implemented using the procedural programming, pattern matching, and object oriented features in ART-IM. We found these features to be very useful attaining our architectural goals for DYCE. The rest of the input/output functions use procedural techniques.

Application Use and Benefits

DYCE is being used internally in IBM's development and manufacturing facilities in Essex Junction, Vermont. Primary users of the system are the characterization support groups whose jobs are to determine any manufacturing problems by reviewing the electrical test results. It has become a popular analysis tool used by the domain experts themselves to alleviate the data explosion brought on by advancing technology. This is in contrast to most traditional expert systems, which are built primarily for other (non-expert) users. The obvious advantage DYCE holds for the experts is several orders of magnitude increase in speed.

There are over four dozen DYCE applications in these areas; it has become standard business practice. For example, users have developed application-specific expected pattern tables dealing with three types of automated analyses using the DYCE shell. (1) product dispositioning, (2) quality screening, and (3) data interpretation. Product dispositioning is used to scrap defective product or ship acceptable product on to the next step. Quality screening is a method to categorize estimated product reliability by analyzing special electrical testing as the product is being processed. Data interpretation is key to detecting process defects in the manufacturing and development areas.

For many applications, each time a batch of product is tested, the automatic DYCE mode is invoked to determine one or more of these tasks. This frees the user from mundane data collection and interpretation activities and allows more time for important engineering efforts. From a business perspective, this means that more time is spent on critical tasks of determining the root cause of the problems. When the electronic mail indicates problems, users consult the knowledge based system in the interactive mode to efficiently review the supporting information.

DYCE is performing the equivalent work of dozens of engineers and technicians. The average time to determine results from the electrical tests have been reduced from hours to seconds (minutes if

⁸ "MYCIN" is a knowledge based system for medical diagnosis developed at Stanford University, CA.

there is heavy processor contention). It is estimated that hundreds of hours a month are being saved relative to manual diagnosis of production problems. Off-shift analysis is consistent and available at the expert level whenever needed. The ability to automatically review *all* the data has resulted in the early detection of manufacturing defects. Early detection, in turn, has saved additional product from less than optimum manufacturing processes. The cost of a *missed* problem approaches several million dollars in a large semiconductor fabrication facility.

Limitations: The shell architecture of DYCE, in conjunction with the use of multiple AI techniques, has important benefits to the successful implementation and continued use of these applications. However, awareness of DYCE's limitations is important. The architecture of DYCE, as presently deployed, has no learning ability. Experts must "teach" DYCE all application knowledge in the table-driven environment. New problems, not previously known to DYCE, will not be output in the conclusions. Care must be taken NOT to infer that product is completely free of problems even if DYCE concludes there are no problems diagnosed. This simply means that there is no evidence of "known" problems. Additionally, DYCE does not know if the expert has left out important rules needed to correctly determine diagnoses. There is no automated verification for application knowledge. Experts must also carefully review any table editing for inadvertent deletions or incorrect format during maintenance. Finally, for complex applications, the table-driven environment becomes difficult to manage without the use of a knowledge editor.

Application Development and Deployment

Approximately 16 person-months were needed to design and develop DYCE to the production version released in January 1992. The development team consisted of one part-time knowledge engineer and one part-time mainframe applications development programmer over a two year period. The responsibility of the programmer included database access, input and output functions, and integration of the inference engine. The knowledge engineer designed and developed the inference engine, developed the overall knowledge based system architecture, and performed the knowledge engineering responsibilities.

It is the implementation of generic meta-rules that allows the application-specific rules to be external to the inference engine. Therefore, validation is simplified into two parts. One part deals with the correctness of the meta-rules, which is independent of any application. The second part begins to look more like a database question, where validation deals with the completeness of the application knowledge. For example, a change to the application knowledge

(expected data patterns tables) does not affect the underlying inference engine. This is not the case for a traditional rule-based expert system.

Validation of the generic meta-rules was accomplished by running hundreds of production patterns and checking DYCE output against expected (known) conclusions. Further, by knowing which rules are expected to fire, an accounting of the resultant certainty factors can be done to cross-check the DYCE diagnosis for accuracy. Domain experts for each application were responsible for validation of application knowledge. This includes specifications, application rules for each diagnosis, and associated certainty factors for each rule.

The mainframe deployment of DYCE effectively allows the users access to automated data interpretation as part of their daily tasks. DYCE is an extension of the existing business practices and represents an advanced software diagnostic tool used in conjunction with existing analysis systems. Training activities are minimized because programming, AI, and knowledge engineering skills are not required for maintaining the application-specific knowledge representation tables. User interaction is relatively simple, since all of the analysis is performed by DYCE. Access to the interactive mode is by selection from familiar analysis menus. The interactive mode is menu-driven for ease of use. The automatic mode issues electronic mail; no daily user interaction is necessary.

Maintenance

Our observation is that a traditional expert system's inference engine of several hundred rules can be represented by tens of generic meta-rules in DYCE's architecture. This greatly simplifies the maintenance of the inference engine. The only need for changes to the inference engine deal with adding advanced reasoning ability and are separated from application-specific modifications, such as specification changes. The shell architecture and meta-rules segment the traditional knowledge engineering effort required to maintain the system. Application-specific knowledge is handled independently by the domain expert in a table-driven environment. The inference engine is modified by the developers without worrying about the specifics of each application. However, our experience with the table-driven environment has shown that for complex applications, the tables can become very large and difficult to manage. Users are required to verify the completeness and accuracy of application knowledge manually. Even though the tables are straight-forward, a software system for table management is seen to be extremely beneficial in helping maintain complex applications. We plan to add a knowledge editor function to assist users with application table development and maintenance.

Conclusions

In the 12 months that DYCE has been deployed, we have found that this architecture has some important benefits concerning the use and maintenance of knowledge based systems in dynamic environments.

The combination of a fast-paced industry and the explosion in available data may contribute directly to the reason that DYCE is used both by the domain experts and by non-experts as a tool for increased productivity. Virtually all of our users like to have the computer send them "diagnoses by wire" and have an integrated system for their data analysis and interpretation requirements.

Even for table-driven applications, a knowledge editor is desirable to manage the dynamic application-specific patterns. Manual verification techniques are cumbersome in continuously changing applications.

We have observed that a few, but powerful meta-rules can be used to represent the heuristics used by the experts for problem-solving in this class (data interpretation) of applications. This has encouraged us to plan to add more meta-rules to the inference engine to increase the reasoning ability of DYCE. The aim is to allow the computer to reason with other aspects of semiconductor process diagnosis. The goal is to eventually automate root-cause analysis.

ACKNOWLEDGEMENTS

The participation of the many experts and users who have encouraged the development of this application is greatly appreciated. Special thanks goes to Mike Marceau, Mary Brooks, and Bob Desmarais for helping to shape the system and to integrate it into the development and manufacturing areas. Thanks also goes to the IBM Corporate Manufacturing Expert System Productivity Center in San Jose, CA.

for assistance and training. A special note of acknowledgement goes to Michael Potter, whose management vision has empowered us to succeed.

REFERENCES

- Roundtree, R. 1986. Software Defect Analysis for Memory Products, *Texas Instruments Engineering Journal* 3(1):46-50.
- Pan, J., and Tenenbaum, J. 1986. PIES: An Engineer's Do-it-yourself Knowledge System for Interpretation of Parametric Test Data. *AI Magazine* 7:62-69.
- Sher, G.; Eaton, D.; Fernelius, B.; Sorenson, J.; and Akers, J. 1993. In-line Statistical Process Control and Feedback for VLSI Integrated Circuit Manufacturing. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology* 13(3):484-489.
- Saxena, S. 1993. Fault Isolation during Semiconductor Manufacturing using Automated Discovery from Wafer Tracking Databases. In Proceedings of the Ninth Conference on Artificial Intelligence for Applications, 313-320. Orlando, FL.: Institute of Electrical and Electronics Engineers, Inc.
- Punch, W. 1992. Large Interactions of Compiled and Causal Reasoning in Diagnosis. *IEEE Expert* 7(1):28-35.
- Gallant, G., and Thygesen, J. 1993. DEPICT, Digitized Expert PICTures, Forthcoming.
- Buchanan, G., and Shortliffe, E. eds. 1984. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA.: Addison-Wesley.