An Experiment in Encapsulation in System Diagnosis

Don Gartner
McDonnell Douglas Aerospace
PO Box 516
St. Louis, MO 63166
(314) 234-2997
gartner@iddsrv01.mdc.com

John W. Sheppard
ARINC
2551 Riva Road
Annapolis, MD 21401
(410) 266-2099
sheppard@arinc.com

Abstract–**The maintenance of fighter aircraft has become increasingly problematic as evidenced by the numerous studies and research and development projects (both contracted and independent) that have addressed the problem in recent years. As a result of these efforts, the "easy" problems have, at least to some extent, been attacked and remedied. The more intractable problems remain. Because of this, the sponsors of the Aircraft Maintenance Integrated Diagnostics Demonstration (AMIDD) project perceived that the application of traditional methods used to perform maintenance and diagnostics have begun to produce diminishing returns. Thus, non-traditional methods using advanced diagnostics appear to hold the most promise for alleviating the difficult diagnostic problems currently faced. In this paper, we describe the initial results of an AMIDD experiment with advanced diagnostics that demonstrate some potential improvements for aircraft maintenance associated with using these methods.**

I. BACKGROUND

AMIDD was conceived in the early 1990s as part of the A-12 program. The U.S. Navy initiated the original AMIDD project to explore ways to convert the Navy's existing maintenance structure from a paper- to a computer-based operation. After the demise of the A-12, the Navy contracted the next phase of AMIDD exploration to add advanced diagnostics and transition the demonstration platform to the F/A-18 C/D aircraft. The final phase involved a diagnostics demonstration with a Marine Corps F/A-18 squadron at El Toro, California that began in 1995. The major requirements of the last two phases included developing expert system-based diagnostics for the APG-65 radar using concepts from rule- and model-based reasoning to demonstrate the advantages of these advanced technologies in an operational environment.

Specific objectives of the diagnostics demonstration can be summarized as follows. First, the demonstration was expected to provide an effective diagnostic system with the intent of reducing the incidence of cannot duplicates and no fault founds (designated in the Navy as A-799 codes). Second, the demonstration would apply the principles emerging from the IEEE AI-ESTATE (Artificial Intelligence and Expert System Tie to Automatic Test Equipment) initiative defining "encapsulated" diagnostic systems with plug and play capabilities. Finally, the demonstration would provide a mechanism for supporting advanced maintenance feedback and analysis to facilitate process and design improvement.

II. INFRASTRUCTURE

Figure 1 shows the AMIDD infrastructure with its top-level diagnostic and maintenance system. Under this system, the fault isolation process is initiated at debrief, where mission and operational data from a flight are transferred to a ground-based collection point. The data are transferred via a data storage unit (DSU) which is brought from the aircraft to debrief by the pilot. At this stage, the AMIDD system uses all available data (built-in-test [BIT] data, environmental data, aircrew and maintenance technician observations, aircraft historical data, and design data in the form of dependency models) to isolate faults to the maximum extent possible. This initial fault isolation work saves a significant amount of maintenance time at the aircraft.

The diagnosis developed during debrief is sent electronically to Maintenance Control, which must manage all base maintenance to provide as many operational aircraft as possible. The corresponding maintenance action form (MAF) contains a fault code that represents the isolated fault. Maintenance Control uses the debrief diagnosis to assign the problem to a Work Center, which completes the diagnosis (when necessary) and performs the actual maintenance work.

When further diagnosis is required, the Work Center receives the fault code in a task statement which specifies one or more Interactive Electronic Technical Manuals (IETMs) and any associated data required to complete the fault isolation. Moreover, the specified data and IETM software is downloaded onto a portable computer, termed the Portable Electronic Display Device (PEDD) by AMIDD, which the maintenance technician uses at the aircraft in lieu of a paper technical manual. The PEDD also collects the maintenance data entered by the technician or taken from the aircraft and uploads it to the Work Center.

As shown in Figure 1, Debrief, Maintenance Control, and the Work Center all have access to the Diagnostic Database, both for storage and retrieval of data. As such, all data

0-7803-3379-9/96 $5.00 © 1996 IEEE

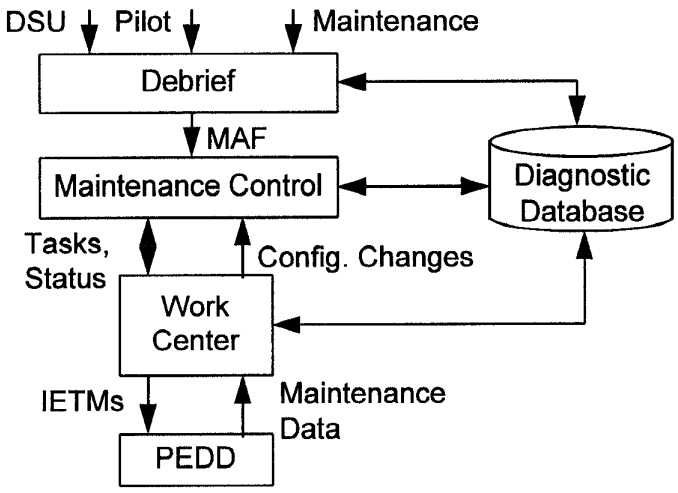

Figure 1. AMIDD Infrastructure

associated with every maintenance action can be easily and accurately collected and stored.

III. Diagnostics on AMIDD

The AMIDD team performed its experiment using a diagnostic inference model (DIM) which may be thought of as a cause and effect model [4], [7]. This means that for every test that can be applied to a system, one can associate a test failure with the fault or failure mode which "caused" that test to fail. If these causal relationships are defined and then linked together, the resulting construct is called a DIM. From this, test results (pass or fail) can be used to isolate failures to a small group of components called an ambiguity group. Several software tools are commercially available for constructing and evaluating DIMs. AMIDD uses ARINC's System Testability and Maintenance Program (STAMP®)[3] for this function. STAMP is not an embedded component of AMIDD but an off-line application.

A DIM can be used in diagnostics in several ways. It can be the basis for a fixed test tree (static diagnostics) or for test selection on an individual case basis (dynamic diagnostics). The AMIDD team initially designed a static implementation. However, later in the program, the team recognized an additional requirement for a dynamic diagnostic capability. As a result, the AMIDD diagnostic architecture now features both capabilities. Static diagnostics have been incorporated through the use of the C Language Integrated Production System, or CLIPS [5]. The dynamic implementation is provided through the Portable Interactive Troubleshooter, or POINTER™ [6].

CLIPS is a NASA-developed, rule-based expert system inference engine which operates in conjunction with a knowledge base (KB). The knowledge base is a collection of if/then rules that can be derived from either design data or human experts. The inference engine controls rule execution by firing rules that match the current state of diagnostics.

Unlike CLIPS, POINTER does not use a fixed test tree. POINTER can directly import the STAMP-generated DIM, which it uses in making diagnoses. The POINTER used in AMIDD is a customized version. Prior to AMIDD, POINTER existed as interactive software which had been designed to perform dynamic diagnostics. Dynamic diagnostics offer a library of tests from which POINTER may choose. POINTER analyzes prior test results and decides which test should be performed next. This is an iterative process that is repeated until POINTER believes it has enough data to state a high-confidence diagnosis. The AMIDD application did not require POINTER to be interactive, since essentially all of the tests available were BIT tests and all test results would be known following a flight. Therefore, the AMIDD application called for an embedded, background diagnostic reasoner which would process test results in a preset sequence chosen to maximize performance. This version, called Passive POINTER™, consists of an inference engine which performs batch processing of BIT data. It analyzes information from multiple tests using symbolic logic, statistical inference, and pattern recognition.

IV. CLIPS / POINTER Comparison

The AMIDD team constructed the CLIPS diagnostic system around static fault trees and thus it is sequential in nature. This point can be illustrated by considering any path in the fault tree. Each path consists of a pre-defined sequence of tests derived from some assumed context or set of requirements for testing the system (e.g., minimum time to fault isolate). In

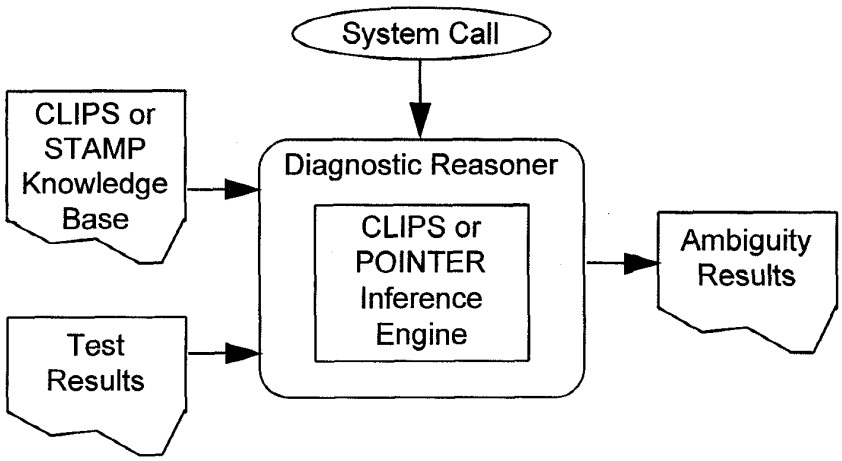

Figure 2. Reasoner Encapsulation

an operational environment, this context is not constant and test results may not be certain. Thus the most relevant tests to be considered may change. Under these changing conditions, diagnosis according to the fixed sequence can lead to erroneous results.

Passive POINTER, on the other hand, follows from the encapsulated diagnosis philosophy [8]. Should Passive POINTER determine a test to be relevant, it will process the test result whether or not the test appears in some pre-defined sequence. This feature illustrates the superior power and sophistication of POINTER diagnostics as compared to the simplicity of the CLIPS fault trees. While the fault trees can adequately diagnose routine failures, they are brittle and fail to diagnose faults outside the defined context.

Another aspect of POINTER's flexibility lies in its ability to process trinary test outcomes. Test trees are binary and will not tolerate any result except pass or fail. With POINTER, however, an unknown outcome may also be designated. When POINTER encounters this result, it will simply not use that test and continue diagnosis by requesting and evaluating another test.

Finally, in POINTER one may assign confidence values to tests which offers several advantages. First, POINTER's reasoning mechanisms will take these confidence values into account in making diagnoses. Further, because of its ability to process uncertain test data, POINTER has the capacity for handling conflicting information. This, in fact, has been a major concern in the operational environment in which errors can be made in the data read from the DSU and the data recorded by pilots and maintenance personnel.

V. ENCAPSULATION

Because the Navy wanted to explore the flexibility of future systems to accommodate different kinds of diagnostic tools and techniques, the AMIDD team had to encapsulate the diagnostic reasoner to allow the use of both CLIPS and POINTER for the diagnostic function. An encapsulated procedure exports an abstract interface that allows the procedure to be selected and applied without having to understand the details of its implementation. The effect is that with respect to the rest of the system, one procedure can be substituted for another with no other changes required. This is popularly known as "plug-and-play." For AMIDD, the goal was to be able to swap a CLIPS diagnostic reasoner for a POINTER diagnostic reasoner, or vice versa. This meant that whatever data was furnished to one reasoner would also have to be furnished to the other, and that the reasoners' outputs would have to be identical in format.

As part of its encapsulation efforts, AMIDD also has addressed a related and growing problem: the wide variety of software applications and databases from diverse suppliers that must interact. Currently, many large systems consist of several entities, each using a different software language and requiring different data formats and communications protocols.

As a first step toward remedying this problem, the Navy directed the AMIDD team to apply two emerging standards to the encapsulated diagnostic reasoner. These were the Artificial Intelligence Expert System Tie to Automatic Test Equipment (AI-ESTATE)[1] and the Test and Maintenance Information Management Standard (TMIMS)[2]. AI-ESTATE will be a specification for a standard representation of data elements required for system diagnosis and test. The intent is to ensure portability of test-related knowledge bases for intelligent system diagnosis and test. The purpose of TMIMS is to facilitate the sharing of historical test and maintenance information across the system life cycle. Since both are emerging standards which are still being defined, all parties agreed that the AMIDD implementation would comply with the *intent and spirit* of the documents.

As illustrated in Figure 2, the encapsulated portion of the diagnostic system has been termed (generally) the "diagnostic reasoner." Specifically, the reasoner can be either Passive POINTER or the CLIPS interface code, whichever is installed. For clarity, we have chosen to apply the name "diagnostic executive" to the body of diagnostic executable software external to the reasoner.

The interface between the diagnostic reasoner and the diagnostic executive is accomplished through a system call and a set of ASCII files. Aircraft BIT results are loaded into the test results file by the debrief function. The diagnostic reasoner decodes the system call so as to select the appropriate test tree (CLIPS) or model (POINTER) and acts upon the content of the test results file to provide a diagnosis via the ambiguity results file, which contains a list of diagnostic conclusions.

The test results file, generated from aircraft BIT results, is provided as an input to the diagnostic reasoner. Each line in the file consists of four fixed-length, blank-delimited fields: test labels, test results, test confidence factors and sequence numbers. The ambiguity results file is an ASCII text file. When Passive POINTER is the reasoner, all possible ambiguity groups are listed in order of decreasing probability. When CLIPS is the reasoner, only a single ambiguity group is presented–the one reached by traversing the test tree to an end (leaf) node.

VI. Diagnostic Architecture

For AMIDD, the diagnostic architecture had to accommodate the encapsulated diagnostic reasoner. However, the team found that many related ancillary diagnostic functions were best left outside of the encapsulated portion–those we call the diagnostic executive. A third component was also essential to the diagnostic capability: the database. In the database, the models, trees, and test universe do not vary unless the system is changed to improve or correct operation. The BIT Test Results table is repopulated after each flight. The diagnostic executive's role is to obtain the correct data from the database, transfer it to the diagnostic reasoner, interpret results passed back from the diagnostic reasoner, and place a fault code in the IETM database. As for the diagnostic reasoner itself, it must accept test results and the appropriate model or tree from the executive, generate a diagnosis, and provide the ambiguity group of fault suspects to the executive.

The CLIPS diagnostic reasoner, built by McDonnell Douglas Aerospace (MDA), consists of a set of rules that allow the reasoner to import the appropriate test tree and test results, traverse the tree to obtain a diagnosis, and store the diagnosis for evaluation by the diagnostic executive in the ambiguity results file. Furnished by ARINC, the Passive POINTER diagnostic reasoner consists of software that decodes the system call to access the correct model, performs reasoning to produce a diagnosis, and places the diagnosis in the ambiguity results file.

VII. Results

The AMIDD diagnostic system has been well received by Marine Corps users, who include both pilots and maintenance technicians. This finding stems from a number of meetings held during AMIDD development to familiarize users with the software, obtain their impressions, and incorporate changes based on their input.

From these meetings, we have learned that the system's data collection capabilities are proving extremely valuable. For the first time, we are able to obtain both accurate and complete data associated with every fault reported and every maintenance action undertaken. Recurring problems are easily recognized; in addition, we have found that some maintenance solutions can be obtained without major investigation. In some cases expert system rules to rectify problems can be speedily written and implemented. Both the CLIPS and POINTER diagnostic reasoners are performing well and are returning accurate diagnoses. The combination of model- and rule-based reasoning works seamlessly.

At the time of this writing, we had not completed the demonstration or the data collection. However, our initial results are both interesting and encouraging. So far, we have identified 112 cases in which the system invoked the diagnostic reasoner. We considered these 112 cases in four different modes: 1) CLIPS processing weapons replaceable assembly (WRA)-level rules, 2) CLIPS processing system-level rules, 3) POINTER processing WRA-level models, and 4) POINTER processing system-level models. Thus, we examined a total of 448 cases. The failure distribution by WRA is as follows:

Transmitter	34
Receiver/Exciter	33
Antenna	23
Radar Target Data Processor	7
Servo Electronics	6
Run "Initiated BIT"	5
Computer Power Supply	4

The average running times for the four scenarios using a 486/66 PC under UNIX amounted to less than 10 seconds for all but the POINTER processing system-level models, and slightly more than 20 seconds for the latter case. Of course, much of the run-time need not be apparent to the user if AMIDD performs other required functions in parallel. Thus, the run-times should be acceptable to the user.

When considering the diagnostic results, the reasoner is able to respond with either an identification of a failed WRA or with a non-WRA fault code. The non-WRA codes include *no*

fault found, run IBIT, conflict encountered, wiring fault, or *circuit breaker tripped.*

With the exception of two fault codes (*wiring fault* and *circuit breaker tripped*), non-WRA codes cause the system to run the initiated BIT (IBIT). Initiated BIT is BIT that must be manually commanded and runs only once. Of the 448 cases considered, 254 were WRA faults, and the remaining 194 cases were non-WRA codes. Of these, 67 instances were wiring faults and 2 were tripped circuit breakers. Of special interest regarding these diagnoses is the fact that the BIT was not designed to address wiring faults, but by modeling the BIT, we discovered that sufficient information was available to find these faults. This aspect demonstrates the significant advantage of using a model-based approach for fault diagnosis.

Of the remaining 125 non-WRA cases, 104 corresponded to *no fault found*, possibly indicating insufficient coverage in periodic BIT (PBIT), and 21 corresponded to *conflict*. Periodic BIT is the implementation of BIT that runs continuously as long as the system is powered. After examining the 104 cases of *no fault found*, it was decided that these cases correspond to 40 unique events with the following distribution:

Legitimate "Run IBIT"	2
IBIT manually terminated	1
No confirming BIT results	16
Model error suspected	18
BIT deficiency suspected	3

The conflict diagnosis arose only when POINTER was invoked since the CLIPS system had no ability to deal with conflicting data. These instances of conflict may indicate problems in data recording, mistakes in data entry, or errors in the model.

VIII. CONCLUSION

Although the results presented here are preliminary, some significant observations can nevertheless be made about the demonstration. First, we are encouraged to see that computational burdens for the reasoner have not been significant or surprising. Second, we have evidence to indicate that a diagnostic inference model provides more robust fault diagnosis than traditional methods by providing increased flexibility and better understanding of the information being furnished by the BIT results. In fact, the dynamic nature of the diagnostics provided by POINTER is leading to a more reliable diagnosis by providing confidence values for the possible faults (including the possibility of conflict). Finally, the discrepancies identified between CLIPS and POINTER results are identifying areas for process improvement by flagging BIT deficiencies, maintenance process glitches, and errors in diagnostics.

For the past 20 years, many claims have been made regarding the significant advantages of applying AI to advanced diagnostics for military weapon systems. Until now, the "results" of these claims have been anecdotal at best. With AMIDD, the Navy has made a concerted effort to quantify the maintenance advantages and disadvantages between existing procedures and the application of various AI techniques. The Navy has also begun to quantify the advantages of providing standardized approaches to intelligent diagnosis. While the current AMIDD results are preliminary, we anticipate that the end results will provide clear evidence as to the value of using advanced diagnostics for complex weapon system maintenance. The success shown in our initial results leads us to believe that the final outcome will be both positive and even more enlightening.

REFERENCES

[1] IEEE Std 1232-1995. *Trial Use Standard for Artificial Intelligence and Expert System Tie To Automatic Test Equipment (AI-ESTATE)*, Piscataway, New Jersey: IEEE Press

[2] IEEE P1389. *Standard for the Management of Test and Maintenance Information*, Draft. 1995

[3] Johnson, F. and Unkle, C.R. "The System Testability and Maintenance Program (STAMP): A Testability Assessment Tool for Aerospace Systems," *Proceedings of the AIAA/NASA Symposium on Maintainability of Aerospace Systems*, New York: AIAA. 1989

[4] Peng, Y. and Reggia, J. *Abductive Inference Models for Diagnostic Problem Solving*, New York: Springer-Verlag. 1990

[5] Riley, G. "CLIPS: A Tool for the Development and Delivery of Expert Systems," *Proceedings of the Technology 2000 Conference*, Washington, D.C. November, 1990

[6] Sheppard, J. W. and Simpson, W. R. "Incorporating Model-Based Reasoning in Interactive Maintenance Aids," *Proceedings of the National Aerospace Electronics Conference*, Piscataway, New Jersey: IEEE Press, pp. 1238-1243. 1990

[7] Sheppard, J. W. and Simpson, W. R. "A Mathematical Model for Integrated Diagnostics," *IEEE Design and Test of Computers*, Vol. 8, No. 4, pp. 25-38. 1991

[8] Simpson, W. R. and Sheppard, J. W. *System Test and Diagnosis*, Norwell, Massachusetts: Kluwer Academic Publishers. 1994