Robust Strategies for Diagnosing Manufacturing Defects

Nancy E. Reed

Computer Science Department
University of Minnesota
reed@cs.umn.edu

Abstract

We describe a manufacturing screening task, the diagnosis of defects on a computer board. Then we describe several strategies used by expert troubleshooters performing the task. These strategies use "inexact models" of the components and connections on the board. A prototype expert system has been implemented that uses the strategies and models. The strategies and models are robust because they are applicable to a wide range of problems, including problems not previously encountered.

Brittleness is a common problem with diagnostic systems. We propose the use of these types of strategies to increase the robustness of diagnostic systems.

Introduction

In the domain of computer hardware diagnosis, we investigated a *board-level diagnosis* task in a manufacturing test environment [Reed and Johnson, 1993]. The goal of this task is to find and replace the smallest unit on a computer board using a minimum of resources and time. An additional constraint is that the time available to diagnose each faulty board has an upper limit based on the board's replacement cost. If the faults are not found within that amount of time, it can be more cost-effective to scrap the board. The boards are tested for faults after manufacture in a production environment. The board examined is an interface board for an IBM personal computer. The board is of moderate complexity, with over 30 components and approximately 200 connections. The level of integration ranges from MSI (medium-scale integration) to VLSI (very-large-scale integration). Some components on the board have internal states. The board uses both digital and analog signals.

The experts we worked with each had over 10 years of experience diagnosing various pieces of computer hardware. We used observation and techniques of verbal protocol collection and analysis [Ericsson and Simon, 1985] in the investigation.

The parts on computer boards can be classified into four types: components, solder, copper traces (on boards), and non-electrical components. The physical failures can also be classified into four types: shorts, opens, incorrect parts, and incorrectly positioned parts. Intermittent errors are excluded.

The boards are initially screened with a software test that returns one of approximately 100 numbers or error messages if an error was detected. This number or string can be considered the initial symptoms of the fault. Boards that do not produce an error message are functioning correctly and need not be examined further.

The number of possible faults is much greater than the number of initial symptoms (100), therefore the initial symptoms cannot discriminate individual faults. When the initial information is insufficient to determine a diagnosis, more data must be acquired. Tasks with this characteristic are called *sequential diagnosis tasks*, as described by Gorry and Barnett [1968]. Additional data are obtained by performing *operations*. The important decisions to make are what operations to perform, and in what order.

Operations are the means available to acquire data to solve problems, and are described in the next section. The following sections describe the strategies and models of the hardware and faults used by the experts. Then a prototype expert system, Blizzard, is described that implements the strategies and models. Blizzard suggests operations to perform to solve each case, but must rely on user input for the results of each operation. Adding sensor input to a diagnostic system in this type of manufacturing-test environment is then discussed. In the last section, we discuss the generality and usefulness of these kinds of strategies for diagnosis.

Operations

The task is described by the *operations* that can be performed. For each case (board), a sequence of operations is performed by a diagnostician until the case is solved, or a time limit is exceeded. We divide the operations into three types: data, information, and repair.

Data operations are used to provide information about the actual behavior of the board being tested.

There are two sub-types of data operations: context and result. *Context* data operations provide a testing condition where data may be obtained. *Result* data operations produce an observable result in the current context.

A number of different data operations are available, including visual inspection of the board, continuity testing between any two points, and probing points with an oscilloscope to produce a waveform, among others. The experts make routine use of many different types of data operations.

Information operations are used to provide information about the correct function of the components and connections on the board. Examples of information operations are using a schematic to find where a connection goes, or probing a functioning board to find a correct data value.

Repair operations are the means of repairing components or connections. Examples of repair operations are replacing a component, adding a wire, or cutting a copper trace.

There are a small number of each of the three types of operations (7-9), but many locations and/or contexts that most may be used. A complete list of the operations observed in the environment investigated is given in Appendix A of [Reed and Johnson, 1993].

In this type of diagnostic task, faults need only be identified to the *level* that will affect the repair necessary. For example, if the fault is within a component, the exact location is irrelevant since it will not change the repair necessary. There can be multiple faults within a component without changing the repair necessary (replacing the component). For other types of faults, including solder shorts, the exact location is critical to making the correct repair.

Specialized Strategies

The term "specialized strategies" is used to refer to some of the expert strategies described that are specific to this task, and possibly other similar tasks. These strategies may be at a local or global level. The ones observed in this environment are described below. In addition to the specialized strategies, the experts used general problem-solving strategies such as a "process of elimination" and "repair and test".

We found that a small number of operations, local strategies, and control strategies can be used to solve and describe a large number of problems in this computer hardware diagnosis task. The *strategies* are general because they are applicable to a large number of problems. They are powerful because they are fast and effective in solving many types of problems. The strategies use *inexact models* of the system and its components. These models contain less detail than causal models, but contain additional diagnostically relevant information about faults and system structure. The inexact models are described in more detail in the following section.

We describe solution strategies at two levels: local and global (control). *Local strategies* describe behavior over a small number of operations. They are used repeatedly in problem solving. *Global strategies* describe the behavior during an entire problem or a large portion of a problem, and can be described as being composed of a sequence of local strategies.

In a study of 24 boards [Reed and Johnson, 1990], four local-level specialized strategies were observed: compare and conquer, heuristic path following, stateless analysis, and endpoint analysis. In addition, one global-level specialized strategy was observed: difference pursuit. The strategies are reflected in the sequences of operations used in solving faults. Data and/or information operations are used to accomplish the strategies.

Compare and conquer is used to compare a data value with the correct value. This strategy is used to determine which data values acquired are correct and which are incorrect. The comparison value may be obtained from a specification or it may be obtained from a correctly functioning piece of hardware. Using a correctly functioning board directly eliminates the need for causal reasoning. This strategy is useful because it is fast and accurate. It may take minutes or longer to trace through a circuit to calculate the correct signal at a specific point in a specific context and this calculation is prone to errors. Using a correctly functioning piece of hardware as an "oracle" eliminates errors of this type. This strategy can be used with any type of data and at any level of abstraction. The evidence for this strategy is a pair of operations. One operation acquires a data value (data operation) from the board in question, and the other operation obtains the corresponding reference value (information operation). The experts verified some values from memory or mental models, without looking in documentation.

The following three specialized strategies focus on *where* to look for relevant data in contrast with compare and conquer which determines the *correctness* of data values.

Heuristic path following is used to focus on relevant portions of the board. The components on the board are connected in many ways and it is important to examine only the most relevant points. This strategy uses information about components and connections in a localized area to determine where to look next for relevant data. This is a simplified form of causal reasoning that reduces to a binary decision: *relevant* or *not relevant*. Relevant data points may be ordered or all of equal importance. This strategy produces two or more operations that examine particular components or connections. Knowledge about the type of component and the particular context is used to collect only the most potentially useful data values out of all possible data values.

Stateless analysis is used to examine components with state (components with some form of internal memory). The outputs of this type of component are

determined not only by their current inputs, but also by the sequence of previous inputs applied to them. This strategy reduces the complexity of examining this type of component by selecting specific context(s) and then obtaining data. This strategy ignores much of the global state of the system, reducing complexity by examining the components in their most informative state – the state where an error occurs.

Endpoint analysis is used to find a starting place when more detailed information is unavailable. Attention is directed toward the interface between the module examined and the rest of the complex system. In board-level diagnosis, this means examining the tabs (input/output) of the board if focus information does not suggest other specific areas or components. This strategy is often successful because it might find an observable symptom of the fault(s) that may be tracked by other strategies.

We observed one specialized control strategy that is general and utilizes the above local-level specialized strategies. *Difference pursuit* contains four steps. First, a context is established that produces an error. Secondly, a symptom or observable difference of the fault is obtained. This difference may be detected through any data operation. The third step locates the point on the board where the difference first appears. Finally, local testing is performed, if necessary, to distinguish between possible fault candidates.

Difference pursuit is general because it can be used on any problem. Many different types of knowledge can be used to increase its efficiency. For example, knowing *when* (the context) and *where* to look for symptoms of a particular type of error allows the acquisition of the first symptom (step 2) with little search. This strategy is also powerful because it ignores irrelevant information and focuses attention on the most relevant data. Compare and conquer is used to determine the correctness of data values obtained. Heuristic path following tracks symptoms through components and connections. Stateless analysis can be used on components with state. Endpoint analysis can be used to find a symptom when other focus information is unavailable.

Inexact Models

The strategies described above succeed using inexact models of the system and components. These models contain selected diagnostic knowledge for this task. We suppose that this knowledge is similar to that in the mental models of the experts. See Gentner and Stevens [1983] for a description of mental models.

We call these models *inexact* because they contain both more and less than causal models. They contain more than causal information because information about structure and possible faults is also present. They contain less information because only the diagnostically useful knowledge is present. The hardware is, in many cases, too complex to have a complete model of its function. The inexact models are based on and include much information traditional to model-based methods [Hamscher *et al.*, 1992].

The inexact models contain structural, functional and fault knowledge. *Structural knowledge* describes how the components are connected together. This is non-causal knowledge, often readily available from documentation.

Functional knowledge describes the constraints between a component's inputs and outputs when the component is functioning correctly. For complex components, or components with state, a full description of the constraints may be prohibitively large. For these components, only the basic constraints are initially described in the models. Basic constraints include the type of signals the component carries (TTL) and which pins carry power and ground. Other data values may be obtained as needed by examining a correctly functioning component in the same context.

Fault knowledge describes the faults that may occur and how to repair each type. For example, an "open" is indicated in a connection if the signals on the ends are different, or if a continuity test gives a "not continuous" result. The repair for an open connection is to solder in a wire to replace the connection. The repair for a fault is non-causal associational information.

Implementation

Blizzard, a prototype expert system, implements a computational model based on the strategies and inexact models described above. It performs diagnosis in a manner similar to the experts by using the same types of knowledge. Blizzard diagnoses faults on the IBM interface board and a demo board [Reed and Johnson, 1993]. Blizzard is written in Cleric, an experimental rule-based language developed at the University of Minnesota [Moen, 1988]. Cleric uses an assumption-based truth maintenance system for bookkeeping functions and is embedded in Common Lisp.

As shown in Figure 1, Blizzard requests data (the result of an operation) from the user, the user performs operations on the hardware, then gives the results of the operations to Blizzard. The user may enter the specific data that Blizzard requests, or any other data that have been obtained. The interaction continues until the problem is solved.

Blizzard uses the strategies to suggest operations in the following manner. Each case starts with an initial symptom. Visual inspection is suggested first to screen for visible faults. If a fault is detected, then the fault is repaired and the board is re-tested. If no faults are detected visually, then the case proceeds.

The initial symptoms may indicate one or more components relevant to the problem (based on previous causes of the initial symptoms). The first of these components is examined in more detail by requesting operations. If the tests indicate that the component is functioning correctly, and more components are on the "relevant" list, they are examined in order. If, during

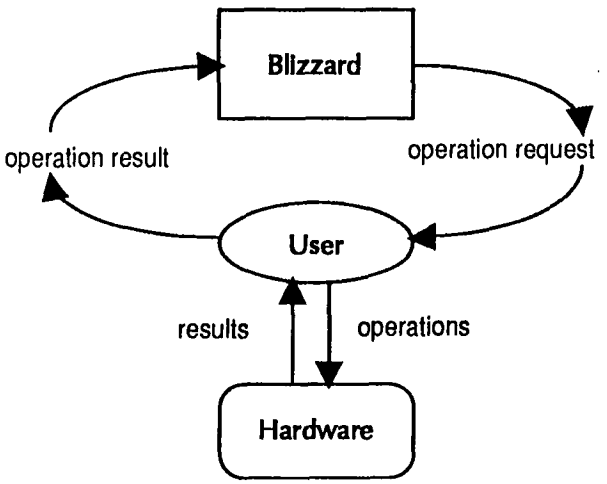

Figure 1: User interaction with Blizzard.

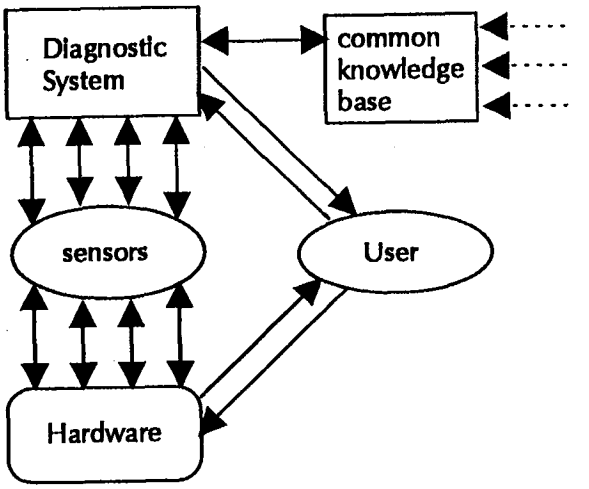

Figure 2: A diagnostic system using sensors.

the examination of a component an incorrect signal or data value is detected, the point where the symptom is obtained becomes a focal point for further investigation. This point indicates the success of the second step of difference pursuit. If an incorrect signal is detected on an output pin, then tests are performed on the input pins. An incorrect signal on an input pin suggests testing along the connection. The most relevant place to examine during diagnosis is the place where observable data changes from correct to incorrect (the third step in difference pursuit). Once located, local testing is performed (fourth step) to determine the fault. Local testing is often necessary since more than one fault can cause the same symptoms.

Blizzard's knowledge base contains data including the results of most types of information operations. The following information is obtained from documentation: power lines, ground lines, all connections, and component locations. Component definitions are in the form of the inexact models previously described. The user may be asked to provide the results of other information operations, primarily examining a good board in a specific context if the correct value in that context is not easily calculated.

As Blizzard is implemented, a human user provides all data from the board being tested. Figure 2 shows how a similar diagnostic system with sensors could interface with test equipment and the board (or other hardware). The most useful types of sensing in this environment are vision, probing, and continuity testing. Interfaces for each type of operation can be developed independently, relying on user input for operations without a sensor interface. The system would be increasingly autonomous as sensors and interfaces are added. It could solve routine problems, relying on user assistance only for more difficult problems.

If more than one diagnostic system is in use in the same environment, a common knowledge base could be used (as shown in the figure) to accumulate data from solved problems and new information operations. This common knowledge base would allow all systems to improve based on the problems seen by each diagnostic system.

Discussion

One reason for observing experts performing diagnosis is to try and understand how they handle novel defects and new situations. The experts are extremely robust. The experts we observed worked on a variety of different pieces of hardware during the course of a day, week, or year, making general methods important. The board described in [Reed and Johnson, 1990] was a new design. We were thus able to observe their general strategies and models as they gained experience on the new board.

One source of robustness in the experts is their use of many different types of operations such as visual inspection, probing, continuity testing, etc. Many faults can be detected by more than one type of operation. Using the fastest operation saves time. Restricting a diagnostic system to only one type of operation (for example probing a point), makes some faults much more difficult to detect.

As an example, the experts used visual inspection to detect many errors. In a manufacturing screening environment, the number of errors that can be detected visually is greater than in a field return environment. In particular, mis-oriented or mis-positioned parts, incorrect parts, and some solder problems occur much more frequently in this environment and can be detected vi-

sually. The difficulty in finding faults is related to the operations used to detect them. For example, a solder short that is visible can be detected by inspection in a few seconds. If testing and probing of signals is used to find the same fault, it may take minutes or hours. Visual inspection detected 8 of 24 faults on the new board design. A vision system that could detect these types of errors would save the time it takes to detect these errors with more time-consuming (and possibly destructive) tests.

Another source of the robustness observed in the experts is because they have general models of errors and of the parts they work with. For example, a short is a short, no matter where is occurs. The symptoms of a short may be different if the short is in one part of the board or another, but visually, they look about the same. Symptom-based categories for defects (for example "stuck at") are less general in this way than physical defect categories.

Their models of components contain general information. For example, TTL components must have valid voltage levels to work properly, so levels can be determined to be valid or invalid, before determining their correctness. If a component doesn't have power, it cannot be expected to produce the correct outputs from its inputs. Verifying good power and ground connections is important in addition to examining the components input and output signals. The same parts appear on many different boards. In particular, components with simple gates (AND, OR, NOT, etc.) are common. Using one general description of each type of component reduces memory requirements and makes this knowledge readily transferable to new board designs. This is one of the important benefits of model-based reasoning.

The use of a functioning board for comparison is another source of robustness for the experts. It avoids the need to have detailed models of complex components. Values that are not known or cannot easily be calculated, can be obtained from a functioning board, and saved for use in future problems. In this manner, the knowledge-base can grow as problems are solved, increasing the efficiency of solving future problems. It should be noted that the strategies described do not *depend* on the availability of a functioning board, but if one is available, it is often the fastest and most reliable way to determine what a correct value should be. Using a correctly functioning board for comparison can also compensate for some other types of problems, including equipment calibration errors and modified equipment settings.

The solution to a faulty board may be almost trivial once an important piece of data is obtained. The experts never appear stuck. They have a variety of useful strategies to apply. They use whatever information is available and perform only the most informative operations. If they have no useful clues to a problem, they may resort to almost random testing, trying to find a symptom of the problem. Getting some data values is more useful than wasting time thinking.

The experts continually improve their performance on the boards they work on. They transfer information about interesting errors to each other with written bug lists. They also keep track of the solution to faults with each of the original symptoms (by faulty part, short location, etc.). Blizzard has the capability to save this information to assist in future problems. Once a faulty part has caused a particular initial symptom, that part is a good place to examine when that same symptom appears again.

Acknowledgements

This work was supported by IBM and the University of Minnesota's Microelectronic and Information Sciences Center (MEIS). Thanks to the expert troubleshooters at IBM who allowed us to observe them.

References

[Ericsson and Simon, 1985] K. Anders Ericsson and Herbert A. Simon. *Protocol Analysis, Verbal Reports As Data.* The MIT Press, Cambridge,k Massachusetts, 1985.

[Gentner and Stevens, 1983] Dedre Gentner and Albert L. Stevens, editors. *Mental Models.* Lawrence Erlbaum, Hillsdale, N.J., 1983.

[Gorry and Barnett, 1968] G. Anthony Gorry and G. Octo Barnett. Experience with a model of sequential diagnosis. *Computers and Biomedical Research,* 1:490–507, 1968.

[Hamscher *et al.*, 1992] Walter Hamscher, Luca Console, and Johan de Kleer, editors. *Readings in Model-based Diagnosis.* Morgan Kaufmann, San Mateo, CA, 1992.

[Moen, 1988] James B Moen. Dynamically creating production rules using lambda closures. Technical report, University of Minnesota, Informatics Research Group, March 1988.

[Reed and Johnson, 1990] Nancy E. Reed and Paul E. Johnson. Generative knowledge for computer troubleshooting. In *Proceedings of the Ninth European Conference on Artificial Intelligence,* pages 535–540, Stockholm, Sweden, August 6-10, 1990. Pitman Publishing.

[Reed and Johnson, 1993] Nancy E. Reed and Paul E. Johnson. Analysis of expert reasoning in hardware diagnosis. *International Journal of Man-Machine Studies,* 38(2):251–280, 1993.

[Reed *et al.*, 1988] Nancy E. Reed, Elizabeth R. Stuck, and James B. Moen. Specialized strategies: An alternative to first principles in diagnostic problem solving. In *Proceedings of the Seventh National Conference on Artificial Intelligence,* pages 364–368, St. Paul, Minnesota, August 22-26, 1988.