Pimtool, an Expert System to Troubleshoot Computer Hardware Failures

Narendra Dev and Bart Anderson

Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, California 94043
ndev@mayfield.hp.com and bea@cup.hp.com

Abstract

This paper describes a tool to diagnose the cause of failure of a HP computer server. We do this by analyzing the dump of Processor Internal Memory (PIM) and maximizing the leverage of expert learning from one hardware failure situation to another. The tool is a rule-based expert system, with some nested rules which translate to decision trees. The rules were implemented using a metalanguage which was customized for the hardware failure analysis problem domain. Pimtool has been deployed to 25 users as of December 1996. We plan to expand usage to over 400 users by end of 1997. Using Pimtool, we expect to save over 15 minutes in Mean-Time-to-Repair (MTTR) per call. We have recognized that knowledge management will be a key issue in the future and are developing tools and strategies to address it.[1]

Introduction

Problem Description

Many of Hewlett-Packard's (HP's) high-end servers are used in Mission Critical environments. If a server fails due to a hardware problem, the cost of downtime to our customers is very significant. We need to be able to diagnose the cause of the failure rapidly, and restore the customer's computing environment so that the problem does not recur.

The objective of this project is to reduce the time to diagnose a hardware failure and improve the quality of the diagnosis. We do this by analyzing the PIM dump and maximizing the leverage of expert learning from one hardware failure situation to another.

Pimtool provides recommendations to the Field Engineers (FEs) as to which Field Replaceable Unit (FRU) to replace. This is for situations where there is a deterministic cause for failure. In other situations which are less clear, it guides the FE to get help from more technical experts.

Pimtool has been developed for use by trained HP engineers. Currently, it is not licensed for use outside HP.

Why Use AI?

One of the constraints we placed on the solution was to avoid hardcoding the troubleshooting logic; it should be easy to add new logic. We made this a constraint because we wanted to develop a troubleshooting environment usable for analyzing crashes on all HP's computer systems. This implied that the tool and logic would be separate. The tool would provide reasoning capability specific to the hardware failure analysis environment.

We were aware that a rule-based paradigm has been used successfully to develop a computer crash analyzer (Register and Rewari 1991). This was developed mainly for software and operating system crashes. We felt confident that a similar logic could be used to troubleshoot hardware failures.

The primary reasoning operator in the metalanguage is the IF verb. Simple cause-effect relationships are coded as IF statements. In these situations, the reasoning is like a rule-based expert system. There are 209 IF statements in the ruleset developed for troubleshooting the T500 server. The nested IFs in the rulesets translate to decision tree logic.

We evaluated whether the case-based reasoning paradigm was applicable. What we really wanted was a prescriptive solution since many of these hardware failure situations are deterministic in nature. The rule-based or decision tree paradigm was a better fit for these situations.

A model-based reasoning paradigm was not considered since our primary mode of analysis is expertise based on historical failure information.

Previous Approaches

In the past, we tried two different solutions to troubleshooting hardware failures.

Traditionally, troubleshooting was documented in a reference manual of several hundred pages and quick reference extracts of this information (Anderson 1993). Level-300 experts would decode registers, look up values in tables and come to conclusions about the cause of the problem. Unfortunately, the troubleshooting process on high-end servers is complex and requires a trained expert who regularly deals with hardware problems. Even so, the troubleshooting process can be lengthy and error-prone (for example, in decoding registers). Different engineers often reach different conclusions. Yet another problem with a paper manual is the difficulty in updating it for refinements in the troubleshooting algorithm.

To solve some of these problems, a web-based troubleshooting tool was introduced for mid-range HP servers. The tool guides level-300 experts (this skill classification system is discussed further below) through a troubleshooting process similar to that described in the hardcopy reference manual. Many hardware failures can be analyzed by keying in data. Other reference material is available by clicking on hyperlinks. Although this web-based tool has proved popular with level-300 experts, it does have several shortcomings. It is not designed for FEs and is not available to them. Also, the troubleshooting process on mid-range servers is considerably less complex than for high-end servers.

Application Requirements

We agreed upon the following minimum functionality for desired August 1996 completion:

1. Limit the scope of this application to the T500 high-end server platform. This is one of the most complex computer systems HP has manufactured. The tool should be leveragable to other products. Also, the troubleshooting logic should be extensible to future products in the same hardware technology family.
2. Usable by a level-200 T500-trained Field Engineer (FE). This means that it is targeted to an FE who has received basic training across HP's computer product families, specialized training on a particular product family and a few years experience. This individual can handle most normal customer on-site repairs independently but usually does more complicated repairs in conjunction with an expert in HP's central call center. About 300 FEs and call center engineers worldwide have been trained on the T500 product.
3. Want a "verbose" option for level-300 experts. This should provide detailed and technical information in addition to the simple recommendations provided in the default "non-verbose" mode. For example, the verbose option decodes important registers and shows the reasoning by which a conclusion is reached.
4. Run on the host environment. This is Unix(*) on T500. It should also run on the PC portable used by the FEs. The environment here is Windows 3.1.
5. Not be dependent on a LAN/Netscape connection for the rulesets. There should be minimal dependencies i.e. the tool should be largely self-contained.
6. Focus first on the functionality and user interface messages, not on the user interface medium. Hence an ASCII-in, ASCII-out user interface was deemed acceptable. A windows interface was not required.
7. Separate the troubleshooting logic from the rest of the program so that the troubleshooting expert can modify the logic without having to recompile the program.
8. Identify processor, memory and cache problems. If time permits, troubleshoot the more difficult problems relating to the I/O system.
9. Provide an interactive option to analyze the first PIM or all PIMs. When a system crashes, there is usually a cascade of failures. Selecting the right PIM to analyze is important to arrive at the correct recommendation.
10. Make the analysis tool independent from the capture of the PIM data. This would enable us adapt the capture technology to different HP computer products.
11. Tell the FE in English what to do i.e. replace memory board in slot xx

Application Description

System Architecture

The lab team recommended developing a rule-based metalanguage which would meet the above requirements within the required timeframe. This would be programmed in C. The metalanguage was developed on schedule by July 1996. A minimal set of rules was developed during August.

As a technology, the metalanguage syntax met the requirements of the authors and the syntax soon stabilized. Minor enhancements to the syntax were requested and were quickly implemented. For example, the ability to compute and display addresses optionally in decimal and hex was added.

Pimtool was developed to run on the host Unix(*) environment. It can run on any HP9000 computer with HP-UX 9.x or 10.x operating system. However, the Pdcinfo data acquisition software needs to run on the system which crashed.

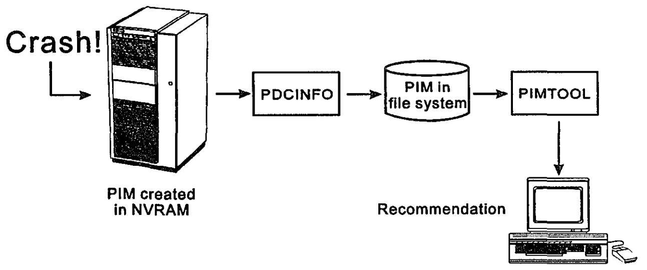

Figure 1. Pimtool Architecture

Pimtool is designed to run as a two-step process (see Figure 1). When a T500 system crashes, it saves troubleshooting information in a section of memory which is preserved across crashes and cycling power (non-volatile memory or NVRAM). This troubleshooting information is called Processor Internal Memory (PIM). When we bring up the operating system and file system after the hardware failure, we run a program called Pdcinfo. This program acquires the PIM from NVRAM and stores it in the Unix(*) file system. We can manipulate the PIM in the file system since it is an ASCII file. When we run Pimtool, the template file first maps the register values in PIM to variables. Some of the registers that are of interest are the Machine Check Type register, Processor Memory Bus Interface (PMBI) status register, the Memory Access Controller status register and the PMBI Slave Address log. This data acquisition template file is programmed in the same metalanguage as the data analysis rulesets.

In the next stage, the rulesets operate on the variables. Based on logic coded by the expert, the data is analyzed and an action recommendation for the FE is displayed on the terminal.

Pimtool today is a read-only system. The only interactive capability is in selecting a particular PIM to analyze. As an enhancement, we are investigating allowing the user to enter troubleshooting information in addition to the PIM data. This has been successfully implemented in the Canasta project (Register and Rewari 1991).

Having the programmer create a data capture template file for the T500 PIM dump was very helpful to the ruleset author since the author was new to ruleset programming. As we move Pimtool to other products with different PIM layouts, we will have to redo the template file.

In the future, our goal is to have programmers work with content-matter experts, setting up a structure and template for the rule files which can be completed by the content-matter experts. Pimtool encourages such collaboration permitting engineers to concentrate on what they do best.

Knowledge Representation

Knowledge is represented in the metalanguage of the rulesets (Curry and Gordon 1996). As mentioned earlier, this equates conceptually to rule-based expert systems or decision trees. Here is a sample of the metalanguage commands:

The DECLARE command is used to create an array. For example:

```
DECLARE error_type[1]
```

The SET command assigns a value to a variable. It is used to set flag values. It is assumed to be a hex value or a string. For example:

```
SET index[0] = 0
SET slot[0] = (pmbi_saddlog[0] BITNUM 14-17)
SET map[0] = (bc_slots[(index[1]+ 1)] CHARNUM 5-8)
```

The two search operators are FIND and GET. FIND searches for the next occurrence of the string. For example:

```
FIND "Unexpected"
```

FINDNC works the same as FIND but is not case-sensitive.

GET reads the fields specified by number from the current line in the PIM. For example:

```
GET 3,1 = error_type
GET ALL UNTIL ENDTOMB = bc_slots
```

Rules are implemented using the IF-THEN structure. There are 209 IF statements in the T500 ruleset. Here is an example using the IF statement to check the last three digits in the Upper Bus Converter port. Based on the value, we make a diagnosis whether power on the bus converter IO bus had been lost.

```
IF ((bc_slots[index[1] + 2)] ENDS d242
THEN
  PRINTX ("PIM Tool: UBC port, module 0, in slot", bc_slots[index[1]]," previously lost power.\n")
ENDIF
```

A number of syntax constructs within the test condition of an IF statement assist PIM analysis. For example:

```
IF ((pmbi_status[0] ENDS 0704)
  OR ((index[0] MOD 4) = 0)
  OR (iva[0] STARTS 0)
  .
  .
```

The WHILE structure is the looping structure and is used to evaluate statements several times. For example:

```
WHILE (index[0] < 9)
  DO
  .
 ENDWHILE
```

The PRINT statement is the primary way to communicate with the user. All error messages and user communication need to be embedded in print statements.

```
PRINT ("This is not an HPMC!\n")
```

PRINTX works just like PRINT but these messages print out only in verbose mode. NEXTLINE tells the program to skip the rest of the line and any blank lines and start the next FIND on the first line that contains values. ENDPROG tells the main program to end.

Here is a section of a Pimtool ruleset, to give the reader a flavor of the syntax.

```
#rule to identify when all MACs detect 6540
IF (estat_6540[0] = total_mem_card[0])
THEN
  PRINT ("\n")
  PRINT ("MEM Flg: All of the MACs logged estat = 6540! Run LOGTool using the
';detail' option with MEMRPT for analysis
by CE Assist personnel.\n")
ENDIF

#------ Double-Bit ECC Rules ---------
IF   ((pmbi_status[0]   ENDS   0a04)   OR
(pmbi_status[0] ENDS 0a06))
THEN
  PRINTX ("\n")
# This rule works for Double ECC faults
when the BLKID & slot number are decoded by
PDC
  IF (blkid[0] != 0)
  THEN
    PRINTX ("PIM Tool: PDC decoded the BLKID
as ", blkid[0], "; the bank as ",bank[0],
"; the PMB slot as ", memslot[0], ".   ***
PDC Versions 2.64/2.73/2.8x/3.82 decode the
BLKID incorrectly when main memory consists
of different bank sizes and when MPD is NOT
enabled. ***\n")
  ENDIF
```

Examples of Hardware Failure Analysis

Below is actual output text generated by Pimtool based on analysis of three separate PIMs. The first example is of a hardware failure probably caused by software (verbose mode).

```
**** Pimtool: decoding tombstone #1
****   Pimtool:   tombstone   occurred   on
01/14/96 at 00:47:36
**** Pimtool: PDC Version 2.80
**** Pimtool: using rule file tdefault

Tombstone for Processor [14/1].

Chassis Code: 16f0.
All codes starting with 1 apply to early
selftest for CPU and FPU.

H P M C   A N A L Y S I S   S U M M A R Y
Version - 14OCT96

CAUSE: Processor [14/1] forced this HPMC at
the start of a PMB transaction to a PMB
memory module when no Responder asserted
the 'SLAVE_ACK' signal!

STRATEGY: The no SLAVE_ACK signal is NOT
asserted for the following:
 - when the address in PMBI SADDlog is not
recognized by any PMB module;
  - when there is a parity error (in the
address or the BLK_ID & Trans_Type);
  - when the PMB module is NOT (ready or
installed or configured)!

Mem Scan: Begin scanning all of the memory
controllers (MACs) for errors.

Mem Err: Memory controller, MAC, in slot 4
previously detected a DRAM error!

Mem Err: Memory controller, MAC, in slot 11
previously detected a DRAM error!

FE  Action:  The  address  value  in  PMBI
SADDlog may exceed Main Memory! This may be
caused by software. No hardware replacement
is suggested.

**** Pimtool RULEs: Analysis is complete.

***************************************
```

The second example is of an ambiguous situation with a bus check error. Here the FE is directed to call the HP call center for help. (Header text is omitted.)

```
H P M C   A N A L Y S I S   S U M M A R Y
Version - 14OCT96
CAUSE:  Bus  Check  from  a  fatal  error
detected on motherboard for [12/0]!

FE Action: Processor [12/0] detecting a Bus
Check with 'no estat code' is improbable!
Call FE Assist for help!

Mem Scan: Begin scanning all of the memory
controllers (MACs) for errors.

Mem OK: Memory card in slot 8 did NOT log
any errors!

**** Pimtool RULEs: Analysis is complete.

*******************************************
```

The third situation is of a double-bit memory error. The FE is directed to replace the memory board in certain situations. (Header text is omitted.)

```
 H P M C   A N A L Y S I S   S U M M A R Y
Version - 14OCT96
CAUSE: Processor [13/1] found a Double-bit
ECC error reading main memory!

FE  Notice:  Do  nothing  if  Memory  Page
Deallocation (MPD) is enabled, however:

FE Action: Replace the memory card in slot
8 ONLY if MPD cannot be enabled!

Mem Scan: Begin scanning all of the memory
controllers (MACs) for errors.
```

```
Mem Err: MAC controller in slot 8
previously detected a hard or fatal error!

Mem Err: Suggest replacing the motherboard
for Processor in slot (hex d)!


Bus Converters:
  0  fff8fff0  50b00040  10000004  00dd4043
(upper)
 0001f048 fff7d008 0ffb0f83 (lower)
  0  fd79fd7d  c0f048c3  fed115c6  1ad53de3
(upper)
 00000000 00000000 00000000 (lower)

I/O Map: UBC port, mod 0, in PMB slot 0,
includes all I/O addresses from (hex
fff00000 to fff80000); its PATH is 0/.

I/O Map: UBC port, mod 2, in PMB slot 0,
includes all I/O addresses from (hex
fd7d0000 to fd790000); its PATH is 2/.

**** Pimtool RULEs: Analysis is complete.
```

Uses of AI Technology

Selection of Alternatives

We initiated a dual effort to meet the requirements. A small lab team was assembled to design the technology which would meet these requirements. The lab team recommended developing the rule-based metalanguage programmed in C which would meet the above requirements within the required time frame.

We also talked to an outside vendor to investigate an off-the-shelf solution to meet the requirements. The vendor proposed a rule-based development environment which would meet the requirements, with a better windows-based user interface. The data model in the vendor's solution was very sophisticated and seemed to be an overkill for our need. Also, we had decided that an ASCII-in, ASCII-out user interface was acceptable so the Windows user interface was not a major advantage. Since we were looking for a tool which long-term can be deployed across our HP computer product line, the cost of deployment to our 5,000 HP FEs becomes a significant cost. Since the off-the-shelf solution was not significantly better than the internal proposal, a decision was made to develop the application using the HP lab team.

Why Was AI Successful?

We built the tool specific to the hardware failure analysis domain and customized to the needs of the authors. We were successful in developing a knowledge representation language which meets the needs of the hardware failure analysis domain. The authors were comfortable using this language to capture their expertise. The syntax provided a number of commands and options which made analyzing crashes easier.

Insights Gained

The AI technology we used seems fairly straightforward. Most of the insights relate to the deployment of the technology and the management of the knowledge.

Security of the rulesets is a concern since the rulesets are ASCII files and can be easily modified or corrupted. We are monitoring this security concern and will add encryption of rulesets or some other security implementation if these concerns become high priority.

We have started recognizing the need to implement style guidelines, so that the rulesets are more easily maintainable. We are evolving the rulesets to a more object oriented style, using reusable modules.

One constraint we faced is that only one person at a time can work on the rulesets. We may want to consider separating the rulesets from the error messages. That way one author can be working on improving the usability of the error messages while the other is writing more rules.

Application Use and Payoff

Application Deployment

Pimtool has been deployed since August 1996 to 10 users. We added 15 users in December 1996, for a total of 25 users. Pimtool today has been deployed to a subset of level-300 experts and the FE-assist engineers. They number about 75 worldwide. Our target is to deploy to this audience by May 1997. They are the opinion leaders in the field. The rulesets have been enhanced significantly during this time. Deployment to the level-200 FEs trained on T500 is planned for the next phase in 3Q97. They are about 400 worldwide.

Payoff and Benefits

Our goal for this application is to cut the average T500 troubleshooting time for hardware failures by 15 minutes. The cost of downtime to the customer's business, for mission critical systems, is of the order of $1000/minute. So a 15 minute reduction of troubleshooting time can result in a 15K$ per event savings to the customer. Early feedback is that a 15-minute reduction is an achievable goal.

Today, Pimtool can successfully trap over 50% of the hardware failures encountered and make a recommendation with a degree of confidence comparable to recommendations from experts. It is also successful in directing the FE to get expert help in the remaining cases.

Pimtool will make the troubleshooting process consistent and repeatable. Once we have a consistent and repeatable process, we can use standard TQC principles to improve the process.

By building a closed-loop process for incorporating field feedback, we will be able to get objective design feedback back to the lab much more quickly. During the development of new computer systems, Pimtool will make it easier for the lab to identify hardware problems.

Impact on Proactive Support Process. We are considering using Pimtool in conjunction with a proactive monitoring tool. When a customer's hardware fails, monitoring software dials our central call center. In addition to the failure notification, we would receive the Pimtool analysis and Pimtool recommendation at the same time.

Impact on FRU Repair Process. We are investigating the possibility of making it a standard field practice to include the Pimtool recommendation with the defective FRU when the FE returns it for repair. This should help our FRU repair organization minimize the number of No Trouble Found (NTF) repair situations.

Software Distribution Process

We are distributing Pimtool through three channels:

1. FTP pull distribution of the software and rulesets. This exists now. Engineers can update their rulesets by using anonymous FTP or a Intranet web page.
2. CD-ROM push distribution to T500 trained FEs. We plan to cut this CD-ROM after there is a high confidence in the rulesets from our level-300 experts and the FE-Assist engineers. We will do this until the software is widely available using the push distribution strategy described next.
3. Push distribution by integrating the software with the diagnostics software, distributed through the HP-UX UNIX(*) operating system release process. It takes six months to get software through this process and realistically, it takes an additional year before customers migrate to the new HP-UX release in volume.

Application Development and Deployment

Development and Deployment Effort

It took three months to develop the metalanguage and template file. It took five months to evolve the T500 ruleset to the point today where the field feels comfortable with the ruleset recommendations. Below are the resource estimates to develop and deploy Pimtool.

Functional area	Engineering Months
Lab	6
Author	4
Documentation/usability	3
Program managment	2
TOTAL	15

15 engineering months at HP billing rate of 11.5K$ per month costs 172.5K$.

Application Development

The lab started with the requirements (described earlier) and created an External Reference specification documenting the functionality of the metalanguage. The lab also created an Internal Maintenance specification which described the programming logic in pseudocode. The lab did walk-throughs on this programming logic. The C programming from this pseudocode was fairly straightforward.

We have not yet ported Pimtool to the PC Windows or DOS platform. We want to maximize our learning on the UNIX(*) platform before we invest in this port. We recognize that once we are implemented on two platforms, the logistics of keeping the versions in sync will put additional burden on the development team. This porting effort will be completed in the time frame coincident with the rollout to level-200 FEs in 3Q97.

Formal Development Methods Used

The following formal methods were used with Pimtool:

Functional specification: We defined the required functionality of Pimtool in a functional specification. This is a useful means of communication between the field support engineers, subject matter experts and lab engineers.

External Reference Spec: This was the lab response to the functional specification. Since it met the requirements, there was very little discussion around this document

Beta testing: We invested in an extended beta test of Pimtool to ensure that the tool makes high-confidence recommendations. We asked experts in the field to use the tool and compare the results with what they would have recommended by experience.

Application Deployment

Deploying Pimtool has been a significant learning exercise. Deploying the tool directly to the level-200 T500-trained Field Engineer (FE) ignores the reality that this individual is backed up by a support chain which extends through the remote call center, the worldwide expert center and the labs (see Figure 2).

A key factor in the acceptance of Pimtool is whether the field has confidence in its recommendations. As we started deploying the tool in September 1996, a number of level-300 experts in the field recommended a change in Pimtool's approach. They suggested that troubleshooting recommendations be written in simple and unequivocal language for the Field Engineers. If a hardware problem lacked a clear solution, the experts felt that Field Engineers should call an expert rather than struggle with the problem themselves. Since AI is a new technology, the consequences of overly ambitious recommendations from Pimtool could be fatal to the success of the project.

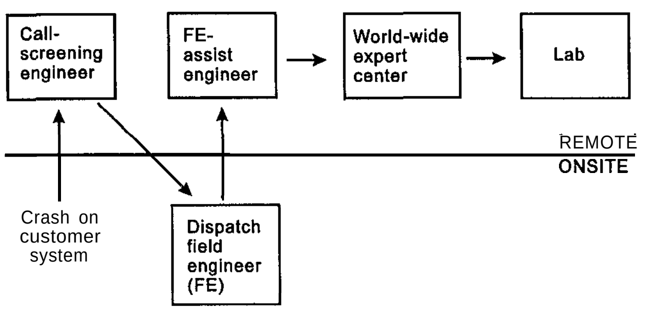

Figure 2. HP Support Chain for Pimtool

Since buy-in from level-300 experts was crucial to the success of the project, we decided to segment the deployment of Pimtool into two phases. In phase 1, we would concentrate our deployment efforts on the level-300 experts and the FE-assist engineers in the remote call center and use a hands-on troubleshooting class to provide them with theory and practice in troubleshooting. This was a manageable set of engineers numbering no more that 75 worldwide. We focused our efforts in designing a training curriculum for troubleshooting HP servers, for which Pimtool is one of the key tools. This class was deployed in December 1996, with 10 engineers trained. It is now being held once monthly with 10-15 engineers trained in each session. Our goal is to train the 75 engineers worldwide by May 1997. They in turn will train other remote engineers when they return to their home offices.

Simultaneously, we have been reworking the rules to make them more conservative, based on the feedback from the field experts. The feedback from the experts towards these changes is positive. We expect to re-release Pimtool to the level-300 experts and the FE-Assist engineers in January 1997 and re-release the tool to level-200 FEs by 3Q97.

Diffkulties

Getting sponsorship for the project has been a challenge. The project was initiated as a proof-of-concept prototype on a shoestring budget. This was done with the assistance of a college student who did the programming. When a new project manager joined the division field support team, he visualized the long-range potential of Pimtool, embraced it as a strategic tool and included it in the product division's diagnostic architecture. HP's field management will sponsor Pimtool as some of the benefits described earlier are realized.

Data acquisition from the target machine to the expert system machine is a significant short-term problem until customers upgrade to the next UNIX(*) release which integrates Pdcinfo and Pimtool. The Pdcinfo PIM data acquisition software has not yet been integrated with the UNIX(*) operating system. It is unrealistic to expect customers to have external Internet FTP access to download the Pdcinfo program on their computers. So the FE is expected to have a tape or CD-ROM with this software at the customer site, to download the PIM from NVRAM memory to the UNIX(*) file system. Once the PIM is in the customer's UNIX(*) file system, they can usually e-mail the PIM to HP's central call center where the FE-Assist engineer can run Pimtool. This process is cumbersome and has been a major disincentive for widespread Pimtool usage. As mentioned earlier, we are planning in the next few months to cut a CD-ROM for the FEs so that they can have a copy of the Pdcinfo program at the customer site.

Lessons Learned

The following are lessons we learned during the development process.

1. The rule file has become much larger and more complex than we had anticipated. In part, this was because the authors felt comfortable in the Pimtool environment and were able to write rules to cover many more situations than originally planned. Because the rule file now contains more that 1400 lines of code, we feel the need to pay closer attention to programming style (meaningful variable names, in-line comments, modularization, etc.).
2. Style issues with rulesets start becoming significant as the number of rules increase and we start bringing more authors on board. We have developed a draft style guide and will work on improving the style guide and conforming to it.
3. Failure information comes in bursts; so selecting the right piece of information to analyze is critical. We configured a burst to consist of a five-minute interval. As a default, Pimtool analyzes the first PIM in a five-minute burst interval. This design seems to be working well.

Maintenance of Tool and Knowledge

Maintenance of Tool

Pimtool will be maintained by the Diagnostics and Support tools lab of HP. This includes the metalanguage, the template files and the shell scripts. The External Specification and Internal Maintenance Specification are being kept up-to-date to reflect enhancements.

Maintenance of Knowledge

We have developed an authoring responsibility model for rulesets (see Figure 3). Responsibility for authoring rulesets for new products has been assigned to the product divisions. Responsibility for authoring changes to rulesets to incorporate feedback from the field resides jointly with the division and the field. Execution of this model is still in its infancy. We have just started understanding the issues with Pimtool knowledge management.

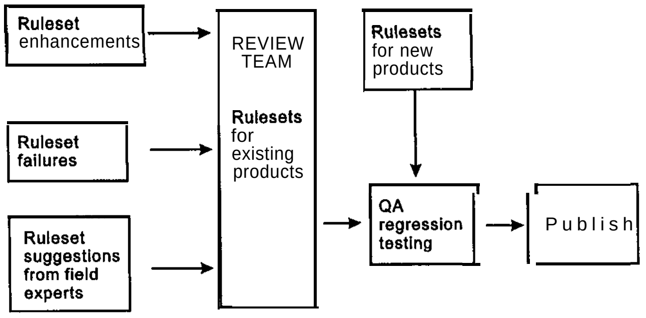
Ruleset enhancements
Ruleset failures
Ruleset suggestions from field experts
REVIEW TEAM
Rulesets for existing products
Rulesets for new products
QA regression testing
Publish

Figure 3. Pimtool Knowledge Management

We made the decision to extend Pimtool to the next HP server platform (called HP9000/K series). An author has been assigned and specification of the template file for K-series has started.

Knowledge Updates

Currently the rulesets are updated about every three or four weeks or when new versions of the firmware are released. They can also be updated on an interrupt basis if necessary. For example, if we discover a class problem or a major new failure signature, we would update the ruleset promptly.

Conclusion and Future Plans

We have been successful in developing an expert system to analyze hardware failures and make recommendations comparable to recommendations from experts today. Pimtool has been deployed successfully to 25 users as of December 1996. We plan to expand usage to over 400 users by end of 1997.

Our future plans are described below:

1. Expand our troubleshooting ruleset authoring to additional HP server products.
2. Standardize our style guide for authoring.
3. Build more validity checking into the authoring environment so that authors can write rulesets with minimal errors.
4. Create an authoring environment in which multiple authors can work on one ruleset.
5. Integrate hardware and software failure analysis. This become more critical as HP's servers become more complex with multiple processors and cluster management software.

Acknowledgments

We would like to recognize the contributions of the Pimtool team members. John Curry was the designer of Pimtool and still does enhancements to the tool. Elizabeth Gordon (a college student who worked at HP last summer) was the programmer and also wrote the T500 template file. David Wold is the ruleset author for T500. Lee Dobyns is the field support project manager. Vince Povio is the training engineer. Jim Thomas is a level-300 expert in the field who guided us through our deployment strategy. Phillip Klahr of Inference Corporation and Anil Rewari of Verity, Inc. read the paper and provided many helpful suggestions.

Authors

Narendra Dev is the program manager for Pimtool. Bart Anderson is the documentation, usability and testing engineer.

References

1. Register, Michael S. and Rewari, Ariil. Canasta: The Crash Analysis Troubleshooting Assistant. In Proceedings of the IAAI-91 Conference, 195-212. Menlo Park, Calif. AAAI Press.

2. Anderson, Bart ed. 1993. HP9000/T500 Corporate Business Servers: CE Handbook, Field Engineering Support (FES), Hewlett Packard, Cupertino, Calif.

3. Curry, John and Gordon, Elizabeth. 1996. PIM Analyzer External Reference Specifications, Rev 4.6, . Diagnostics and Support Tools Lab (DSTL), Hewlett Packard, Cupertino, Calif.

(*) UNIX is a trademark of AT&T