A Comparative Analysis of Event Tupling Schemes

Michael F. Buckley[1] and Daniel P. Siewiorek†

IBM Thomas J. Watson Research Center Yorktown Heights, NY 10598
†Carnegie-Mellon University, Pittsburgh, PA 15213

Abstract

Event logs provide an effective means of improving system availability. However, the majority of faults produce many errors because faults propagate in the time and error detection domains. Thus, the ability to coalesce related events is critical.

The tupling heuristics developed at Carnegie-Mellon University provide one such methodology. These heuristics were applied to a new and larger set of data in order to evaluate the generality of the scheme and to extend the previous work. The extensions included deriving a semantic understanding of why the rules work, expanded statistical analysis, and a comprehensive sensitivity study to determine the effects of changes in the rules.

The results prove that tupling is a useful and general methodology. The sensitivity study enabled the identification of refinements to the rules, while the high degree of skew in the tuple variables enables us to propose that the extreme percentiles be used as an alarm threshold for proactive fault management.

1. Introduction

The availability and reliability of computer systems can be improved by the use of event logs and System Directed Diagnosis (SDD) techniques [1]. However, the "system usually detects the effects of the faults as many isolated errors" [2]. This is because faults propagate in both the time and hardware domains. Thus, a key step in event log analysis is the coalescing, or grouping, of related events.

There are two basic approaches to this classification problem. One uses 'bottom up' methods, such as those developed by Tsao [3], and Iyer, Young and Sridhar [2] to try and combine the individual events together into clusters. This is analogous to region growing in machine vision. The alternative is 'top down' decomposition of the event log into problems, which is equivalent to region splitting.

The objectives of this research were two fold. The first was to determine if the tupling heuristics would generalize to another data set. If they were deemed useful further work would be done to extend the previous research, so that a more comprehensive body of knowledge could be built up on tupling. The authors would note that this notion of building upon previous research is not pursued frequently enough in our field.

The generality was investigated by applying the tupling algorithms to a new, more diverse and significantly larger set of data. The event logs were collected from 193 VAX/VMS machines over a four year period. There were 335 machine years of data in total. The data came from three different types of VAX processor and spanned a range of sites, from manufacturing, through banks and universities, to internal Digital sites. A number of the analyses that were conducted previously by Tsao [3] and Hansen [4] were repeated for comparison purposes. The similarity of the results across all three studies proves that tupling is a general and useful scheme.

The extensions were in two directions, namely statistical and semantic. The statistical analyses included additional univariate statistics and plots and analyses of the extreme values of each variable. The semantic research included an assessment of the plausibility of the tuples and a comprehensive sensitivity study to determine the effect of each tupling rule and clause. The effect of filtering events from the logs was also considered.

These additional analyses enable us to recommend rule enhancements for the VAX/VMS logs, and to propose the idea that the extreme values of the tuple variables be used as an alarm threshold. The skewed nature of the tuple entry count and tuple span distributions, and the extremes analysis, suggest that the extreme percentiles can be used as an effective and robust proactive fault management alarm threshold. For example, the 95th or larger percentiles could be used and they would generate approximately the same number of alarms.

The previous work on event clustering is summarized in Section 2 and the methodology used in this research is described in Section 3. The results are presented in section 4 and the key contributions are summarized in Section 5.

2. Related Research

There is a significant body of literature on computer system availability and reliability, including [1-18]. The studies show that the majority of failures are due to temporary faults [11, 14, 19]. This has led to the

[1] Formerly with Digital Equipment Corporation. This work was supported by Digital Equipment Corporation and by the Office of Naval Research under contract number N00014-91-J-1289.

use of error handling and recovery methods [9, 10] and Symptom Directed Diagnosis (SDD) techniques, as mechanisms to improve reliability and availability. The key to these techniques is the creation and interpretation of event logs. The event monitoring process is discussed in [6] and the foundations for Symptom Directed Diagnosis are presented in [1].

The time-space relationship between events has been investigated by Tsao [3], Iyer, Young and Sridhar [2], and Hansen [4]. The *Tupling* concept developed by Tsao [3] is a method of organizing the information in the log into a hierarchical structure. A *Tuple* or *cluster* is a group of closely related events. Two rules were developed to coalesce the events into tuples. The grouping of events into tuples reduced the number of logical entities by a factor of twenty.

The work on tuples was extended by Hansen [4]. He compared the differences between and among uniprocessor and multi-processor systems. He used data from thirteen VAX 11/780s and five Tandem TNS IIs. He found that individual processors of the same type behaved the same, whereas multiple processors in fault tolerant systems generally exhibited different behavior. He found that the tupling algorithms were not sensitive to moderate variations in the clustering time. This supported Tsao's sensitivity experiment. Hansen also found that at least 15% of the tuples contained events from more than one problem. Sanders [20] reports a similar percentage for groups contained a collection of unrelated error records. The overlapping in Hansen [4] may be due to a deficiency in the tupling algorithm, or to the fact that multiple problems may be present in a system simultaneously.

Iyer, Young and Sridhar [2] developed a methodology for automatically detecting symptoms of frequently occurring errors. Events are combined into clusters, which are combined to form error groups, and eventually super events which correspond to individual problems. The methodology is based on probabilistic techniques and used data from two CYBER systems. The analyses were later extended to an IBM system 3081 [21]. The strength of the relationship between events is given by the ratio of the joint probability to the independent probability.

3. Methodology

The results and conclusions presented here are based on the examination of a large set of VAX event logs that were collected over a four year period. This is the same set of data that is discussed in [6]. The objectives of that research were to develop a thorough understanding of the data set and the event monitoring process, and to identify and correct any deficiencies that were discovered, before proceeding with other analyses. That exercise proved to be invaluable and provided key insights into the data set and the monitoring process. This research builds upon that foundation by using the knowledge that was gained to avoid incorrect assumptions and wrong conclusions. The authors would encourage all analysts to be as rigorous and cautious in their analysis of event logs.

The VAX event log is described next and this is followed by a description of the data set, the analysis software, and the tupling rules used in this research. The event logs, data sets and analysis procedures used by Tsao and Hansen are described in [3] and [4].

3.1 Event Logs

The event log contains records of events as they occur on the system. The events are collected concurrently with normal system operation, and as such reflect actual workload and usage. Some of the events are informational, such as disk mounts and timestamps, while others are related to errors or to system shutdown and start-up.

Each event record has a header section which includes the event type, the event date and time, the system identifier (SID), and the error sequence number (ESN). The information recorded in the rest of the event record varies by event type. The usefulness of specific fields within an event type depends upon the objective of the enquiry.

The steps in the VAX/VMS event log generation and collection process are described in [22, 23]. The main point to remember is that the majority of faults produce many errors and often more than one system failure, because the effects may be detected by multiple error checkers and they may repeat over time.

3.2 Data sets

The main set of data that was used consisted of event logs collected as part of the Mean Time Between Interruption project within Digital, and will be referred to as the MTBI data set. The event logs were collected by logging into each machine once a month and extracting the events for the previous month. The logs were usually filtered, i.e. specific event types were removed from the logs, before they were retrieved, to reduce their size.

The data came from 193 machines, that spanned three VAX processor types, and included customer sites as well as sites internal to Digital. There were 335 machine years of data, which is an order of magnitude more data than previous investigations [2-4], with the exception of [15] who used a similar amount of data. There were 2.35 million events in total spread over 46 different event types. The machines in the sample were running VMS versions from V4.7 to V5.2.

Tsao [3] employed ten machine years of DEC TOPS 20 system data collected from eight machines, while Hansen [4] used five machine years of data collected from 13 VAX 11/780 systems and five Tandem TNS II systems.

The MTBI event logs were supplemented by three other sets of data. The supplementary data included two sets of event logs, and a set of field service data, called LARS (Labor Activity Reporting System).

3.3 Analysis Software

The Error Log Analysis Software (EoLAS) system was developed to analyze the event logs. EoLAS is a collection of software tools that includes commercially available packages such as SAS, routines written by the authors, and code developed by other groups within Digital. The two main functions of the EoLAS software were extraction and manipulation of event log entries, and statistical analysis. The code developed by Hansen [4] was used to form the tuples. The pr template file and the tf clustering rules description file that were used in this research differ slightly from those in Hansen [4], and are described in [23]. SAS was used for the bulk of the statistical analyses. The elements of the EoLAS system are described in [23] and the details of the tupling code are presented in [4].

3.4 Tupling Procedure

The intent of the tupling rules is to coalesce related events into the same group or tuple. Tsao [3] used two rules, which were:

```
Rule 1:  IF next event within ε₁ minutes (2.8)
             include the event in the tuple

Rule 2:  IF the event does not satisfy rule 1
         AND it does not contain physical location information
         AND it is within ε₂ minutes (22.5) of the tuple
         AND its event type is already in the tuple
         THEN include the event in the tuple
```

Hansen [4] extended this methodology and added the following rules, which were applied before rule 1 and rule 2. Hansen also sorted the event logs chronologically before applying these rules.

```
ABORT X0:       IF Error Sequence Number not sequential
                THEN ABORT the current tuple and ignore the
                     subsequent events within ε₁ of the gap

IGNORE I0:      Ignore time stamp entries.

IGNORE I1:      Ignore events with bogus dates
```

There is an example of three tuples in Figure 1. The tuple header starts with the string 'TUPLE', which is followed by the tuple number (391 for the first tuple in this example), the start date of the tuple in seconds since 1-Jan-1960 (826534717), the number of entries in the tuple (2) and the time span of the tuple in seconds (105). The tuple header is followed by one line of text for each event in the tuple. The line contains the error sequence number (864 for the first event in the first tuple), the event date (11-mar-1986), the event time (08:58:37.46), the number of seconds since the start of the tuple (0), the device location information (not shown in Figure 1 since there was none for this event type), the event type (Machine Check), and the event subtype (not shown in Figure 1 since there was none in this case).

```
TUPLE  391  826534717  2  105
864    11-mar-1986    08:58:37.46       0      MACHINE CHECK
865    11-mar-1986    09:00:22.76     105      MACHINE CHECK

TUPLE  392  826538803  44 9529
872    11-mar-1986    10:06:43.57       0      MACHINE CHECK
873    11-mar-1986    10:06:54.53      11      MACHINE CHECK
 .         ...            ...           ..          ...
 .         ...            ...           ..          ...
930    11-mar-1986    12:42:47.11    9364      MACHINE CHECK
931    11-mar-1986    12:45:32.77    9529      MACHINE CHECK

TUPLE  393  826549795  3  558
935    11-mar-1986    13:09:55.28       0      MACHINE CHECK
936    11-mar-1986    13:12:51.83     176      MACHINE CHECK
938    11-mar-1986    13:19:13.01     558      MACHINE CHECK
```

Figure 1: Example of problem being spread over multiple tuples

Ideally each tuple should include all of the events related to one instance of a particular problem. For example, if there is a scratch on a disk surface and a write attempt produces four disk errors, two disk controller errors, and a bus error, within 2 minutes, all of these events should be grouped together into the same tuple. However, a single problem may have its signature spread over many tuples, because the same problem may be encountered many times, and each is likely to generate a new tuple.

The three different implementations of the tupling concept, namely Tsao [3], Hansen [4] and the present research are similar but not identical. The differences occur because the data set imposes some restrictions and hence the methodology must be changed slightly, or, more often, because of incomplete knowledge of the previous researcher's methodology. The main characteristics of the three implementations are summarized in Figure 2.

The use of Hansen's [4] code helped ensure consistency between his analysis procedures and this research. The authors did not have access to data from Hansen [4] or Tsao [3] and thus their data could not be compared to the MTBI data. The tuple matching (tuple type formation) work that was conducted by Tsao and by Hansen was not done in this study.

The main analyses that were conducted are as follows:

- Tsao's [3] and Hansen's [4] implementations were compared (see Figure 2).
- The tuples that were formed were examined to determine if they were plausible, that is, were the majority of the events in the tuples related or were the groupings random. This was done for all logs for all three processor types.
- An extensive sensitivity study was conducted to determine the effect and contribution of the various rules and clauses to the tupling process. This would show if modifications were needed for the MTBI logs. The effects of filtering were also considered because the majority of the MTBI logs were filtered.

  The effect of changes in ε_1 was investigated by Tsao [3] and by Hansen [4]. The methodology used in this research for the ε_1 sensitivity study is very similar to the one used by Tsao. Each of the other input variations was evaluated by doing a pair wise comparison, one with the rule or clause in use and one without the rule or clause. The sensitivity study is based on data from one particular VAX 86xx machine (named Vf), with the exception of the ε_1 analysis where additional data was used from two other machines. The procedure is described in detail in Buckley [23].
- Univariate statistics, histograms and crossplots were generated for the per file variables and the tuple variables. These were generated from all of the logs and the results were compared across the three processor types.

Characteristic	Tsao (1983)	Hansen 1988	Buckley 1992
Work Done			
Tuple formation	Yes	Yes	Yes
Tuple type formation	Yes	Yes	No
Univariate statistics	Yes	Yes	Yes
Distribution type	No	No	Yes
Correlation analysis	No	No	Yes
$\mathcal{E}_1$ sensitivity study	Yes, on KL	Yes, on Tandem	Yes, on VAX
Effect of rule changes	No	No	Yes
Code Used	Tsao 1983	Hansen 1988	Hansen 1988
Machine Years of data	10	5	336
Processor types	KL	VAX 11/780 Tandem TNS II	VAX 86xx VAX 6xxx VAX 8xxx
Rule 1	$E_1 < 2.8$ mins	$E_1 < 2.8$ mins	$E_1 < 2.8$ mins
Rule 2	if no location info and $\mathcal{E}_2 < 22.5$ min and intuple(KL)	if $\mathcal{E}_2 < 22.5$ mins and no location info and intuple(subfield)	if $\mathcal{E}_2 < 22.5$ mins and intuple(VAX)
Other Rules			
Abort Rule	No	Yes	Yes
Ignore old entries	No	Yes, before 1960	Yes, before 1980
Ignore Timestamps	Not applicable	Yes	Yes
Logs sorted	Unknown	Yes	Yes
Extremes eliminated	entries >3000 drop tuple	entries >500 drop tuple	entries >50000 drop tuple TTT, TBT >99999 dropped

Figure 2: Comparison of tupling implementations

4. Results and Discussion

The work that was done can be split into two major sections. The objective of the first portion of the tupling research was to develop a semantic understanding of the tupling rules. This included a review of the previous implementations, an evaluation of the plausibility of the tuples that were formed and an investigation of the effects of various changes in the rules. These items are discussed next and the second sub-section provides statistical information about the tuples. This is followed by general comments and concluding remarks.

4.1 Semantics

The bulk of the semantic work focused upon the usefulness and appropriateness of the different rules, and a number of the key findings from Buckley [23] are presented here. These are preceded by an explanation of how the tuples were evaluated for correctness.

Plausibility: It is not possible to objectively measure the effectiveness of the tupling algorithms because there are few absolutely right or wrong groupings. The events in the log can be looked at and grouped in various ways depending upon the objective of the analysis. Human diagnosticians try different associations based upon their current FRU (Field Replaceable Unit) theory [24].

Nevertheless, it was possible to examine the tuples which were formed and deem them reasonable or not. Hundreds of tuples were examined for the various analyses that were conducted over the course of the research and the tuples which were formed were reasonable in all cases. That is the tuples contained groups of related events. It was also clear that the tupling algorithms were not able to associate all of the events that belong to a given problem into one tuple in all instances. Streams of associated events were sometimes spread over many tuples, as shown in Figure 1 where events from the same problem are spread over three tuples. In fact, the tuples in Figure 1 only cover a portion of the problem which actually spanned a three month period. Thus, the problem was spread over more tuples than shown in Figure 1.

The example in Figure 1 suggests that rules are required at a number of levels. The lowest level would take events with exactly the same syndrome or which are very close in time and group them together. The next level would look for associations between these groups and pull all the events which belong to the same instance of a problem into the same group. This would continue through higher levels until all the events which were due to the same problem were grouped together. The probabilistic scheme developed by Iyer, Young and Sridhar [2] is a good example of such a rule hierarchy. The assumption that is made here when evaluating the correctness of the tupling is that the rules are intended to be lower-level ones. It is assumed that rule 1 is at the lowest level and that rule 2 is at the next level. Hence, the concept of truncations, as defined by Hansen [4] would not be useful in this case because we expect that a single problem may be spread over many tuples.

There are some examples of typical tuples in Figure 3. The last tuple in the figure, tuple number 34, is a good example of correct association of events. The tuple starts with a reboot message, which is followed by 13 disk mounts and the tuple ends with a SNDERR message. This tuple could be referred to as the "power up" tuple type because this is the standard sequence of events after a reboot on this machine. This can be seen from the cross plot in Figure 4. The majority of the tuple entry counts are 15 and these are mainly due to the "power up" tuple.

```
TUPLE  28   893145424  1   0
206 20-apr-1988  07:57:04.43    0  HSC01$DUA1:   ERL$LOGMESSAGE
TUPLE  29   893203552  1   0
304 21-apr-1988  00:05:52.91    0  HSC01$DUA13:  ERL$LOGMESSAGE
TUPLE  30   893285337  1   0
441 21-apr-1988  22:48:57.05    0  HSC01$DUA7:   ERL$LOGMESSAGE
TUPLE  31   893346436  1   0
544 22-apr-1988  15:47:16.59    0  HSC01$DUA7:   ERL$LOGMESSAGE
TUPLE  32   893363526  1   0
573 22-apr-1988  20:32:06.91    0  HSC01$DUA7:   ERL$LOGMESSAGE
TUPLE  33   893369322  10  51
584 22-apr-1988  22:08:42.86    0  HSC01$DUA1:   DISMOUNT VOLUM
585 22-apr-1988  22:08:46.97    4  HSC01$DUA2:   DISMOUNT VOLUM
586 22-apr-1988  22:08:48.84    6  HSC01$DUA3:   DISMOUNT VOLUM
587 22-apr-1988  22:08:50.61    8  HSC01$DUA6:   DISMOUNT VOLUM
588 22-apr-1988  22:08:55.06   13  HSC01$DUA8:   DISMOUNT VOLUM
589 22-apr-1988  22:08:56.31   14  HSC01$DUA10:  DISMOUNT VOLUM
590 22-apr-1988  22:08:58.06   16  HSC01$DUA12:  DISMOUNT VOLUM
591 22-apr-1988  22:08:59.50   17  HSC01$DUA13:  DISMOUNT VOLUM
592 22-apr-1988  22:09:00.17   18  HSC01$DUA14:  DISMOUNT VOLUM
593 22-apr-1988  22:09:33.01   51                FATAL BUGCHECK
TUPLE  34   893397754  15 182
594 23-apr-1988  06:02:34.39    0                SYSTEM STARTUP
595 23-apr-1988  06:02:41.11    7  DUA0:         MOUNT VOLUME
596 23-apr-1988  06:03:20.09   46  DUA1:         MOUNT VOLUME
597 23-apr-1988  06:03:27.59   53  DUA2:         MOUNT VOLUME
598 23-apr-1988  06:03:40.56   66  DUA3:         MOUNT VOLUME
599 23-apr-1988  06:03:48.00   74  DUA5:         MOUNT VOLUME
600 23-apr-1988  06:03:56.38   82  DUA6:         MOUNT VOLUME
601 23-apr-1988  06:04:04.99   90  DUA7:         MOUNT VOLUME
602 23-apr-1988  06:04:14.11  100  DUA8:         MOUNT VOLUME
603 23-apr-1988  06:04:22.29  108  DUA10:        MOUNT VOLUME
604 23-apr-1988  06:04:37.74  123  DUA12:        MOUNT VOLUME
605 23-apr-1988  06:04:45.75  131  DUA13:        MOUNT VOLUME
606 23-apr-1988  06:05:00.31  146  DUA14:        MOUNT VOLUME
607 23-apr-1988  06:05:13.20  159  DUA15:        MOUNT VOLUME
608 23-apr-1988  06:05:36.54  182                SNDERR MESSAGE
```

Figure 3: Example of typical tuples.

There is a corresponding "power down" tuple type that is less well defined. The "power down" tuple con-

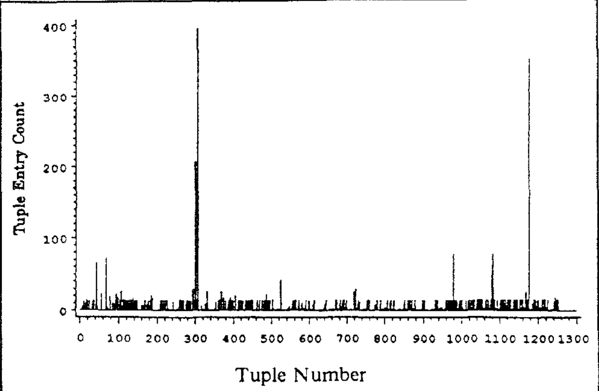

Figure 4: Cross plot of tuple entry count for file Vf

sists of a variable number of disk dismounts and is often terminated by a crash message. The second to last tuple (number 33) in Figure 3 is an example of the "power down" tuple. The number of disk dismounts and the presence of the crash message are variable because the shutdown sequence may be terminated by the operator, or the problem may cause an immediate crash. There were many tuples for file Vf with between eight and twelve entries and the majority of these were due to the "power down" tuple. The "power up" and "power down" tuples were discovered because the apparent "horizontal line" at the 15 entry count mark in Figure 4 aroused our interest. The patterns exhibited in this plot are typical of those obtained for other variables and other machines.

Although, no formal attempt was made to quantify the number of collisions they were rare and our estimate would be that they occurred in less than 1% of cases. This estimate is based upon the hundreds of tuples that were reviewed over the course of the research. The term collisions is being used here in the sense defined by Hansen which is "a collision occurs when the reports generated by two different faults overlap and are identified as a single tuple" [4]. However, one would occasionally encounter a tuple which included one or a few housekeeping events along with the errors from a single fault.

Effect of changes in Rule 1 clustering time: The rule 1 clustering time, ε_1, was varied to determine if it had a major impact on the results. This is similar to the ε_1 sensitivity studies that were done by Tsao and Hansen. Tsao evaluated the changes in the number of tuples, the tuple entry count and the tuple span for values of ε_1 between 11 and 5400 seconds. This research used a broader range of ε_1 values and the changes in the number of tuples, tuple entry count, tuple span, Time Between Tuples (TBT), Time To Tuple (TTT), and number of rule 1 firings, were observed.

The procedure was applied to three different event logs. The logs were selected so that the total number of events in each was similar to the total number of events in the two files that were used by Tsao [3]. Rule 2 was not used in the sensitivity study because it was not used in the sensitivity study in Tsao. Thus, the procedure used here is very similar to Tsao's.

The effects of the changes in ε_1 were evaluated using univariate statistics, box plots and log-log plots for each variable. The results show that the tuple parameters, with the exception of the tuple span, are insensitive to changes in ε_1 over a wide range of values. The tuple span changes linearly with ε_1. The other variables change an order of magnitude slower than ε_1. These results agree with the conclusions in Tsao [3] and Hansen [4].

There is an example of one of the log-log plots in Figure 5. This plot shows how the the tuple entry count varies with ε_1, and the plots for the other variables and files are very similar.

There is a pronounced 'knee' in the tuple count, TTT and entry count log-log plots at the 180 minute mark. A less severe change can be seen in the TBT and tuple span plots at the same point. The change at the 180 minute mark is due to different groups of events being pulled into the one tuple. Thus, it can be conjectured that a value of ε_1 less than 180 minutes coalesces events from the same instance of a fault into one tuple, while leaving the different instances in separate tuples. However, the different groups of events, or instances of a fault, are pulled into the same tuple once ε_1 exceeds 180 minutes.

There is a less obvious change in the slope of all of the variables at the .35 minute point. These two features suggest that the performance of rule 1 is not affected as long as ε_1 is kept within the .35 to 180 minute range, and thus a value in this range should be used for rule 1. The results in Hansen for the tuple count also suggest that there is a major change in the sensitivity at approximately the 1 minute mark, which is similar to the .35 minutes found here.

The reason for the lack of sensitivity to changes in ε_1 is that the tuple span is small relative to TTT and TBT. That is the case because events tend to occur close together in bursts or not at all. Thus, a relatively short tupling time will be sufficient to coalesce the events within a burst. A much longer tupling time is required before the bursts are pulled together.

Effectiveness of Rule 2: The intent of rule 2, as stated in Tsao [3], is to pull non hardware events that do not satisfy rule 1 into the current tuple, if they are within 22.5 minutes (ε_2) of the current tuple and are of the same type as an event that is already in the tuple. This rule is conceptually appealing and proved to be effective for the MTBI logs because there were related events in the log that were often missed by rule 1. The Single Bit Memory Errors (SBEs) are a good example of such events.

The SBE entries in the VAX logs are usually more than 2.8 minutes (ε_1) apart. In fact it was found that 75% of SBEs occurred within three to twenty three minutes of each other. These would be coalesced by rule 2 but not by rule 1. There is an example of rule 2 pulling SBEs into a tuple, which are missed by rule 1, in Figure 6. The top portion of the figure shows part of a tuple that is formed when rule 2 is used. The lower part of the figure shows that many tuples are formed when rule 2 is not used.

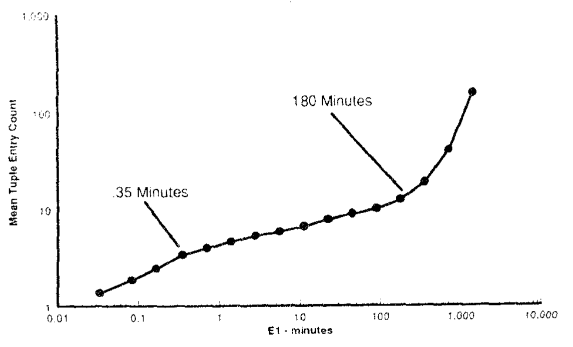

Figure 5: Log-Log plot of mean tuple entry count vs ε_1

```
RULE 2 USED: SBEs coalesced into one tuple.

TUPLE  1   894067276  51   20296
15523  1-may-1988 00:01:16.47    0  CORRECTABLE MEMORY ERROR
15524  1-may-1988 00:06:38.85  322  CORRECTABLE MEMORY ERROR
15526  1-may-1988 00:11:41.36  625  CORRECTABLE MEMORY ERROR
15527  1-may-1988 00:16:41.41  925  CORRECTABLE MEMORY ERROR
15529  1-may-1988 00:21:41.28 1225  CORRECTABLE MEMORY ERROR
  ..     ...        ...        ..              ...

RULE 2 NOT USED: SBEs are spread over many tuples.

TUPLE  1   894067276  1   0
15523  1-may-1988 00:01:16.47    0  CORRECTABLE MEMORY ERROR
TUPLE  2   894067598  1   0
15524  1-may-1988 00:06:38.85    0  CORRECTABLE MEMORY ERROR
TUPLE  3   894067901  1   0
15526  1-may-1988 00:11:41.36    0  CORRECTABLE MEMORY ERROR
TUPLE  4   894068201  1   0
15527  1-may-1988 00:16:41.41    0  CORRECTABLE MEMORY ERROR
TUPLE  5   894068501  1   0
15529  1-may-1988 00:21:41.28    0  CORRECTABLE MEMORY ERROR
TUPLE  6   894068801  1   0
  .    .     ...      .   .      .             ...
```

Figure 6: Example of Rule 2 coalescing SBEs into one tuple.

The summary statistics show that rule 2 was used 84,731 times over all the logs. This is 18% of the rule 1 firing rate and significantly higher than the rate reported by Tsao [3] who found that 95% of the events were classified using rule 1. Thus, it can be concluded that rule 2 is useful and should be retained.

Effect of filtered event logs: The majority of the MTBI logs had the device and volume entries filtered out. Therefore, the Vf log was filtered and the results compared to the unfiltered log to determine if there was a significant difference between the two, and to see if the tuples that were formed were plausible.

Although the results will depend on the event types that are filtered out, one would expect that filtering will produce fewer and smaller (shorter span and lower tuple entry count) tuples, because there are less events to start with. The tuples would be further apart (longer TTT and TBT) because the elapsed time is a constant. The expectations were borne out by the univariate statistics. There are major differences in the statistics for filtered and unfiltered logs.

The tuples in the filtered logs were examined and found to be correct. There were a few instances where a pair of crash/reboot messages that had been in the same tuple in the unfiltered log were in different tuples in the filtered logs. This occurred because intervening entries that were filtered out had acted as 'binding' events in the unfiltered case. That is, the elapsed time between the crash and the reboot was greater than ε_1, but there were intervening entries in the unfiltered logs such that no gap was greater than ε_1.

Thus, the tupling methodology can be applied directly to filtered logs. The flexibility of the Hansen [4] implementation allows one to implement new filtering rules easily and quickly.

Tupling Algorithm Modifications: The results thus far have shown that the tupling rules proposed by Tsao [3] are effective on another set of data and that the tuples that are formed are reasonable. This is not to say that the rules cannot be improved or tailored to a particular data set. This sub-section provides one example of how the rules can be improved, via the addition of the IGNORE rule proposed by Hansen, and one example where modification of the rules improves effectiveness. There are additional recommendations for improving the rules for VAX/VMS logs in [23].

The tupling implementation in Hansen [4] ignored events that occurred before 1-Jan-1960. The idea is to screen out what are probably bogus dates. The MTBI logs have bogus dates that must be dealt with and this is a reasonable approach. The 1-Jan-1980 was used as the cutoff because the bogus date entries in the MTBI logs have 1978 dates. Although, using 1-Jan-1980 instead of 1-Jan-1960 is a minor change it had a major impact on some of the per tuple univariate statistics. The statistics showed that the TTT and TBT mean and range changed significantly by using a cutoff of 1980 instead of 1960. For example, the mean TTT changed by one order of magnitude and the TTT range changed by more than two orders of magnitude.

Thus, the ability to ignore specific events is a useful addition. However, this example also shows that the rules may need to be modified for the data set at hand. The use of location information to keep events in separate tuples provides another example of where the rules should be modified to improve their effectiveness.

There is a clause in rule 2 in Tsao [3] and Hansen [4] which forces the formation of a new tuple if the event under consideration contains any location information. The original motivation for the clause stemmed from a belief that most logged errors would be detected by hardware (hence location information would be present) and be propagated to be detected by software (no location information). Thus, the occurrence of a problem would tend to produce a hardware error with location information which would be closely followed by a series of software errors that would not have any location information. The intent of tupling was that it was to be a level 1 (lowest level) grouping technique. Thus, different problems or instances of a problem should be kept separate and this could be done by forcing a new tuple if there was location information present.

Although, this may have been a valid hypothesis for the KL processor logs, it is not suitable for the VAX architecture, because events that are related will be forced into separate tuples just because they contain location information. Thus for example, SBEs would be forced into separate tuples, even if they had the same syndrome. This suggests that the clause should not be used, or at least that it should be modified such that a new tuple is formed only if the location information is different.

The effects of this clause were examined by conducting a tupling analysis with and without the clause. The expectation is that forcing a new tuple because of the presence of location information will cause the TTT, TBT, tuple span and entry count to decrease, since more tuples should be formed from the same number of events and elapsed time.

The univariate statistics show that the results are as expected. The mean and median TTT, TBT and tuple span decrease and the mean entry count decreases. The median entry count is unchanged. There are 33% more tuples formed when the location clause is used and rule 2 is satisfied and order of magnitude less often.

The tuples for both cases were examined to determine which were the most reasonable. There were many instances where related events were forced into separate tuples because they contained location information. There is an example in Figure 7. This shows a sequence of related ERL$LOGMESSAGE entries, some of which are more than ε_1 minutes apart, being split into two tuples because they contain location information. The events in tuple number 43 in the lower part of Figure 7 (no location clause) are split into two tuples, namely tuple numbers 70 and 71 in the upper half of the figure when the location clause is used.

LOCATION CLAUSE USED: Four tuples are formed

```
TUPLE 69   893734791  2   83
1274  27-apr-1988  03:39:51.66   0 MUA0:       DISMOUNT VOLUM
1275  27-apr-1988  03:41:14.68  83 MUA0:       MOUNT VOLUME
TUPLE 70   893735726  2   0
1278  27-apr-1988  03:55:26.33   0 HSC0$MUA0:  ERL$LOGMESSAGE
1279  27-apr-1988  03:55:26.35   0 HSC0$MUA0:  ERL$LOGMESSAGE
TUPLE 71   893735993  8   21
1280  27-apr-1988  03:59:53.19   0 HSC0$MUA0:  ERL$LOGMESSAGE
1281  27-apr-1988  03:59:53.20   0 HSC0$MUA0:  ERL$LOGMESSAGE
1282  27-apr-1988  03:59:53.48   0 HSC0$MUA0:  ERL$LOGMESSAGE
1283  27-apr-1988  03:59:53.50   0 HSC0$MUA0:  ERL$LOGMESSAGE
1284  27-apr-1988  04:00:14.57  21 HSC0$MUA0:  ERL$LOGMESSAGE
1285  27-apr-1988  04:00:14.58  21 HSC0$MUA0:  ERL$LOGMESSAGE
1286  27-apr-1988  04:00:14.64  21 HSC0$MUA0:  ERL$LOGMESSAGE
1287  27-apr-1988  04:00:14.65  21 HSC0$MUA0:  ERL$LOGMESSAGE
TUPLE 72   893736484  12  151
1289  27-apr-1988  04:08:04.41   0 MUA0:       DISMOUNT VOLUM
 ..     ...          ...       .    ...            ...
```

LOCATION CLAUSE NOT USED: Three tuples formed

```
TUPLE 42   893734791  2   83
1274  27-apr-1988  03:39:51.66   0 MUA0:       DISMOUNT VOLUM
1275  27-apr-1988  03:41:14.68  83 MUA0:       MOUNT VOLUME
TUPLE 43   893735726  10  288
1278  27-apr-1988  03:55:26.33   0 HSC0$MUA0:  ERL$LOGMESSAGE
1279  27-apr-1988  03:55:26.35   0 HSC0$MUA0:  ERL$LOGMESSAGE
1280  27-apr-1988  03:59:53.19 267 HSC0$MUA0:  ERL$LOGMESSAGE
1281  27-apr-1988  03:59:53.20 267 HSC0$MUA0:  ERL$LOGMESSAGE
1282  27-apr-1988  03:59:53.48 267 HSC0$MUA0:  ERL$LOGMESSAGE
1283  27-apr-1988  03:59:53.50 267 HSC0$MUA0:  ERL$LOGMESSAGE
1284  27-apr-1988  04:00:14.57 288 HSC0$MUA0:  ERL$LOGMESSAGE
1285  27-apr-1988  04:00:14.58 288 HSC0$MUA0:  ERL$LOGMESSAGE
1286  27-apr-1988  04:00:14.64 288 HSC0$MUA0:  ERL$LOGMESSAGE
1287  27-apr-1988  04:00:14.65 288 HSC0$MUA0:  ERL$LOGMESSAGE
TUPLE 44   893736484  24  1852
1289  27-apr-1988  04:08:04.41   0 MUA0:       DISMOUNT VOLUM
 ..     ...          ...       .    ...            ...
```

Figure 7: Example of the location clause in rule 2 forcing related events into separate tuples

The conclusion is that forcing events into separate tuples because of the presence of location information is inappropriate for the VAX logs. The clause should be modified such that new tuples are only formed if the location information is different, or preferably the clause should be dropped.

4.2 Statistics

The research included the computation and evaluation of a large variety of univariate statistics and plots. These statistics allowed us to compare our results to previous research and they provided insight into the usefulness and behavior of the tupling rules. They also increase our understanding of the physical behavior of the system which may enable us to discover suitable alarm mechanism for proactive fault management.

The statistical analysis that we conducted is more extensive than than in either Tsao [3] or Hansen [4] and a number of the key findings will be presented here. There is a more thorough exposition in [23]. The results will be presented in the following order, first summary statistics for the tupling process, followed by per file variable and per tuple variable univariate statistics and plots. The univariate statistics and plots were generated from all of the logs for each processor type unless otherwise noted.

Summary statistics on the tupling process: The data reduction ratio measures the degree of information compression that is produced by looking at tuples instead of at raw events. It was found to be .20 over all three processor types. This indicates that tupling can be used as a means of reducing the volume of data that one has to deal with, as pointed out by Tsao [3]. The data reduction ratio of .20 for the MTBI logs is very similar to the ratio of .24 given by Hansen [4] for the Tandem logs. The MTBI data reduction ratio is less than that obtained by Tsao [3] who typically obtained an order of magnitude reduction in the data for the KL processor logs.

The rule 2 to rule 1 firing ratio was .18 across all three processor types and it indicates that rule 1 is used about five times more frequently than rule 2.

Per File Univariate Statistics and Plots: The tupling summary that is output for each log processed includes a count of the number of events, tuples, rule 1, rule 2, rule I0 (timestamps), and rule I1 (times $<$ 1980), firings per event log. The univariate statistics, crossplots and horizontal bar charts for each of the per file variables were highly positively skewed. The crossplots are particularly effective at communicating this and clearly highlight the outliers in the data. There was an example of a crossplot in Figure 4.

It is always prudent to examine the outliers in the data and hence the extreme values of each variable were examined to determine if they were valid or if they were due to errors in the data or methodology. The extremes were valid in each case. There were a variety of reasons for the extremes. For example:

- The VAX 86xx file with the highest number of errors also had the highest rule 1 firing count. These were due to two large bursts of errors in the file. One burst of approximately 3,500 errors was produced by an EMM (Environmental Monitoring Module) which had detected that the temperature for sensor T1 was entering the yellow zone. The other burst consisted of approximately 4,000 logged MSCP messages which were recorded in a two hour period. The two items were not related.

- The two most extreme VAX 6xxx error counts and rule 1 firing counts per file, were tracked to two consecutive event logs. There were approximately 3,650 volume mount, 3,650 volume dismount, and 11,500 logged message entries in each file. These were caused by a tape problem.

The above also demonstrates that examining the distribution of the data and in particular the extremes provides a useful insight into the physical behavior of the system.

Per Tuple Univariate Statistics and Plots: The information that is available for each tuple includes the tuple span, the number of events in the tuple, the TTT and TBT. The histograms, crossplots and univariate statistics show that the distribution of each of the variables is positively skewed. This is particularly true of the tuple span and tuple entry count distributions which have the bulk of their values on the extreme left and a long sparse tail on the right. The patterns are similar for all three types of processor.

These results are in agreement with Tsao [3] and Hansen [4], where it was found that most tuples had a few entries and a short span. The tuple entry count histogram for the VAX 11/780 data in Hansen is nearly an exact match to the same histogram for the MTBI VAX 86xx data. The mean tuple entry count for the Tandem data in Hansen was 4.1 which falls within the range of values for the MTBI data, which was 3.82 to 5.82 entries per tuple. Tsao found that the majority of tuples had a single entry. The figure of 59% is given for one particular file. This is close to the values for the MTBI data, where it was found that 55%, 54% and 71% of tuples for the VAX 86xx, 6xxx, and 8xxx, respectively, had only one entry. This contributes to the skew for the tuple entry count and tuple span variables.

Once again the extreme values of each of the four variables were examined. The outliers for the tuple span and tuple entry count were legitimate, but the TTT and TBT values were wrong, as can be seen from the following examples:

- The VAX 8xxx has the largest tuple entry count value. This was due to a sequence of 8130 device errors that occurred over a three hour period. These events were pulled together by rule 1 and resulted in a high tuple span, tuple entry count, and rule 1 firing rate.
- The four longest tuple spans for the VAX 8xxx were generated by two adjacent files, which included a large number of correctable memory errors. The machine obviously had a Single Bit Memory Error (SBE) in a location that was being accessed on a regular basis. This produced a constant stream of SBEs over two months. The errors were occurring at a rate of three to four per hour, virtually every hour. The errors for long periods were pulled into one tuple by rule 2. These two logs produced a number of other long tuple spans in addition to producing the four longest spans for the VAX 8xxx. This problem was a good example of the usefulness of rule 2.
- The correct TBT and TTT range values are of the order of 31 days. However, the initial extreme values were calculated to be about one year. These were caused by a date from the previous year being in the current year's log. For example, a January 1987 date in a January 1988 log. These dates were obviously incorrect and were probably due to an operator entering the wrong year at the console when requested during system reboot. Thus, these extreme values were excluded from subsequent analyses. This is a good example of where an extremes analysis helped identify incorrect data.

The extremes analysis and the high degree of variability displayed in the graphical plots leads one to believe that the alarm threshold for proactive fault management algorithms could be set statistically. For example, the 99th percentile for the tuple entry count for the VAX 6xxx is 64, and thus an alarm would be raised once there were more entries in the tuple. This would have raised 290 entry count alarms. It should be noted that such a threshold rule need not be the only one that is used. It could be one of many rules, some more sophisticated and some less so. For example, there could be another alarm rule for the tuple span. The long tail means that the value of the threshold does not have to be set precisely because a wide variety of values should result in a similar number of alarms.

4.3 Discussion

The tupling concept is correct and surprisingly effective for its simplicity. The tuples that were formed are plausible groupings because they generally contained related events.

The concept of grouping events on a time basis, as is done in rule 1, is a good idea. For the VAX logs it is sensible to group events that occur within a relatively short period, such as a few minutes, because one frequently see bursts of events close in time that are due to the same instance of a problem. The period of 2.8 minutes is reasonable and is similar to that used in other studies, such as Iyer, Rossetti and Hsueh [25], where events in a five minute window are coalesced. The fact that event interarrival time distributions are often skewed to the left, as was found in previous studies also supports a rule with a short time period.

Rule 2 which incorporates both time and entry type information is also a good idea. For example, rule 1 would not coalesce related SBEs that occurred at intervals of five minutes. However, the SBEs are included into the tuple by rule 2 because they are within 22.5 minutes of each other, they have the same entry type, and they do not have any physical location information in the form that was defined by Tsao [3] or Hansen [4].

The two rules were intended to do lower level associations and they are effective in that regard. However, they are not sufficient to pull all of the events related to the same problem into a single tuple. Therefore more rules are required to group the tuples together. These rules could be probabilistic in nature, such as those developed by Iyer, Young and Sridhar [2], or they could be based upon knowledge of the processor architecture.

The sensitivity study showed that the rules are relatively robust and it also enabled us to recommend enhancements for the VAX/VMS logs. The main conclusions from the sensitivity study are that:

- The ability to ignore events which was added by Hansen [4] is useful because it allows the removal of events which are being generated by parallel or background processes. For example timestamps and events before 1980 should be ignored for the MTBI data.
- The results are not sensitive to changes in ε_1, as long as a value between 1 and 180 minutes is used.
- Rule 2 is useful and should be retained, as noted above. For example, it coalesces the 75 percent of SBEs that occur within three to twenty three minutes of each other.
- Filtering of the event logs does not invalidate the tupling algorithms or the results.
- Events should not be put into separate tuples just because they contain location information, because that would force many related events into separate tuples.

The comparison of the results across the three types of processor demonstrated that the rules were equally effective for all three processors. The comparison employed an evaluation of the plausibility of the tuples, univariate statistics, and graphical plots. The statistics and plots were similar overall for all three processor types, but there were variations. For example, the Rule 2 to Rule 1 firing ratio was .14, .34, and .20 for the VAX 86xx, 8xxx, and 6xxx processor types, respectively. These statistics demonstrate that Rule 2 is useful for all three processor types, but that it is most effective for the VAX 8xxx processor.

Although no formal attempt was made to determine if other factors influenced the effectiveness of the rules we can ´speculate´ about a number of parameters. These include software version, problem type, event type, and to some extent workload. The tupling rules would appear to be equally effective across all of these because the MTBI data included multiple versions of the software, 46 event types, a variety of problem types, and one would assume a variety of workloads since the data came from 193 machines from a variety of sites. Although, we did not explicitly test for variations over these variables we were not conscious of any ´groupings´ in the results, and thus it is reasonable to assume that the effectiveness was not influenced by these factors. By effectiveness we mean that related events were grouped together, collisions were rare or non existent, and there was a substantial reduction in the volume of information.

The factor that is likely to introduce the largest variation in the results is the variations in the error detection routines and event logging algorithms that are used by the processor. This will change the univariate statistics but it will not affect the effectiveness of the rules, or the overall patterns, such as fact that the tuple variable distributions are highly positively skewed.

5. Conclusions

The ability to coalesce related events in an event log is critical for successful fault diagnosis and recovery. The objective of this research was to take an existing grouping scheme, evaluate it effectiveness, and provide extensions to it, if it proved to be effective.

This was done using the tupling scheme developed at Carnegie-Mellon University by Tsao [3] and extended by Hansen [4]. This research used one of the largest and most diverse sets of actual event log data studied to date. The 335 machine years of data was collected from 193 VAX/VMS machines over a period of four years.

The research included the repetition of a number of analyses that were done by the previous researchers, to verify the generality of the scheme, and new analyses to extend the results. The additional analyses included a comprehensive study of the effectiveness of the various elements of the rules; a more extensive statistical analyses on the tupling variables; a comparison of the results across three different processor types; and an effort to obtain a semantic understanding of the tupling rules.

The major contributions of this research are:

- The results prove that tupling is a useful and general methodology for performing lower level associations between events. The usefulness is demonstrated by the substantial reduction in the volume of data that had to be analyzed, and by the fact that the rules coalesced related events. The similarity of the results to those of Tsao [3] and Hansen [4] shows the generality of the concept. The findings are especially convincing because the MTBI data set was substantially larger and more diverse than that used previously.
- It provides a semantic understanding of why the tupling rules work. For example, the fact that 25 percent of events occur within one minute of each other [23], explains why rule 1 is effective at forming tuples. The fact that SBEs are often five minutes apart is the reason for the effectiveness of rule 2.
- The different elements of the rules were evaluated via a comprehensive sensitivity study and enhancements for the VAX/VMS logs were identified. For example, the location clause in Rule 2 should be dropped, and the Ignore rule should be used to eliminate events before 1980.
- The proposal that the extreme percentiles of the tuple variable distributions be used as an alarm threshold for proactive fault management routines, is a new and valuable idea for tupling. The high degree of skew in the distributions of the variables and the extremes analyses indicate that statistical thresholds based on the extreme percentiles, such as the 95th or larger, would be effective and robust.

Acknowledgments

We are grateful to Digital Equipment Corporation for funding this research and are indebted to a host of people at Digital who provided invaluable insight and guidance, along with access to data and facilities. These include: Rick Howe, Tom Weyant, Mike Robey, Ted Gent, Anne Wein, Pat Moran, Peter Carr, Des Farren and John Arroyo. We are also grateful to Prof. Roy Maxion and Prof. John Shen for contributing many valuable ideas to the research, and to Jeff Hansen for providing his tupling code. We would also like to thank the referees for their comments which helped clarify and improve this paper.

References

[1] J.P. Shebell, "Symptom Directed Diagnosis Foundations and Practices," *Proc. IEEE Int. Conf. on Computer Design*, pp. 290-293, 1985.

[2] R.K. Iyer, L.T. Young and V. Sridhar, "Recognition of Error Symptoms in Large Systems," *Proc. of the Fall Joint Computer Conf., Dallas, TX*, November 1986.

[3] M.M. Tsao, "Trend Analysis and Fault Prediction," *Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA.*, 1983.

[4] J. Hansen, "Trend Analysis and Modeling of Uni/Multi-Processor Event Logs," *Masters Thesis, Carnegie-Mellon University, Pittsburgh, PA*, 1988.

[5] D.M. Andrews and E.J. McCluskey, "Final Report, The Measurement and Modeling of Computer Reliability as Affected by System Activity," *CRC Technical Report, No. 85-18, Stanford University, Stanford, CA*, 1985.

[6] M.F. Buckley and D.P. Siewiorek, "VAX/VMS Event Monitoring and Analysis," *Proc. 25th Int. Symp. on Fault-Tolerant Computing (FTCS 25)*, 1995.

[7] M. Sullivan and R. Chillarege, "Software Defects and their Impact on System Availability - a study of Field Failures in Operating Systems," *Proc. 21st Int. Symp. on Fault-Tolerant Computing (FTCS 21)*, pp. 2-9, 1991.

[8] R. Chillarege and D. P. Siewiorek, "Special Issue on Experimental Evaluation of Computer System Reliability," *IEEE Trans. Reliability*, vol. 39, no. 4, 1990.

[9] N.N. Tendolkar and R.L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex," *IBM Journal of Research and Development*, no. 26, pp. 78-88, January 1982.

[10] D.C. Bossen and M.Y. Hsiao, "Model for Transient and Permanent Error-Detection and Fault-Isolation Coverage," *IBM Journal Research and Development*, no. 26, pp. 67-77, January 1982.

[11] R.K. Iyer and D.J. Rossetti, "A Statistical Load Dependency Model for CPU Errors at SLAC," *Proc. 12th Int. Symp. on Fault-Tolerant Computing (FTCS 12)*, 1982.

[12] T.T. Lin, "Design and Evaluation of an On-Line Predictive Diagnostic System," *Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA.*, 1988.

[13] R.A. Maxion and F.E. Feather, "A Case Study of Ethernet Anomalies in a Distributed Computing Environment," *IEEE Trans. Reliability*, vol. 39, no. 4, pp. 433-443, 1990.

[14] S.R. McConnel, D.P. Siewiorek and M.M. Tsao, "Transient Error Data Analysis," *Technical Report, CS Dept.,Carnegie-Mellon University, Pittsburgh, PA*, 1979.

[15] P. Moran, P. Gaffney, J. Melody, M. Condon and M. Hayden, "System Availability Monitoring," *IEEE Trans. Reliability*, vol. 39, no. 4, pp. 480-485, 1990.

[16] D. Tang and R.K. Iyer, "Dependability Measurement and Modeling of a Multicomputer System," *IEEE Trans. on Computers*, vol. 42, no. 1, pp. 62-75, January 1993.

[17] M.M. Tsao and D.P. Siewiorek, "Trend Analysis on System Error Files," *Proc. 13th Int. Symp. on Fault-Tolerant Computing (FTCS 13)*, 1983.

[18] A.S. Wein and A. Sathaye, "Validating Complex Computer System Availability Models," *IEEE Trans. Reliability*, vol. 39, no. 4, pp. 468-479, 1990.

[19] Y.K. Malaiya and S.Y.H. Su, "A Survey of Methods for Intermittent Fault Analysis," *Proc. of the 1979 National Computer Conference*, 1979.

[20] D. Sanders, "Automatic detection of error patterns in computer systems," *Masters Thesis, University of Illinois at Urbana Champaign*, 1986.

[21] R.K. Iyer, L.T. Young and P.V.K. Iyer, "Automatic Recognition of Intermittent Failures: An Experimental Study of Field Data," *IEEE Trans. on Computers*, vol. 39, no. 4, pp. 525-537, April 1990.

[22] L. J. Kenah, R. E. Goldenberg and S. F. Bate, "VAX/VMS Internals and Data Structures," *Digital Press, Bedford, MA.*, 1992.

[23] M.F. Buckley, "Computer Event Monitoring and Analysis," *Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA*, 1992.

[24] R.A. Maxion, "Human and Machine Diagnosis of Computer Hardware Faults," *IEEE Computer Society Workshop on Reliability of Local Area Networks, South Padre Island, TX*, February 1992.

[25] R.K. Iyer, D.J. Rossetti and M.C. Hsueh, "Measurement and Modeling of Computer Reliability as affected by System Activity," *CRC Technical Report, No. 85-21, Stanford University, Stanford, CA*, 1985.