A New Analysis Approach Incorporated with Analytic Hierarchy Process for The Software Application in A Multiple Tiers Environment

Hoo Meng Wong and Sagaya Sebastian Amalathas

Abstract—Software application becomes critical tool in a business operation today. Generally, there is no management staff in the business operation can tolerate the software application downtime beyond the Service Level Agreement (SLA). The time duration spent on root cause analysis activity of software application error carries crucial impact to the business transaction. This is because business operation cannot afford if the software application downtime gets prolonged. Especially an enterprise level of software application is involving multiple tiers. There will be at least one or even more log files at each tier. Of course the root cause analysis activity will easily get prolonged as more than one log file are required for analysis activity. The research is not only to mitigate the time duration prolonging on root cause analysis activity. It is also to increase the accuracy of identifying the actual root cause during analysis activity. This is a necessity to have a proposed logic model sitting at the logic tier. The logic model can analyze the required log files, and then identify the root cause without creating any interruption to the existing software application executing in the production environment. Therefore, this is the strong reason for a Prescriptive Analytical Logic Model (PAL) incorporated with Analytic Hierarchy Process (AHP) proposed. The proposed logic model will contribute a new knowledge in the area of log file analysis. It aligns with the two research objectives, (1) to shorten the total time spent on root cause analysis activity by adopting the decision making process and technique of AHP, (2) to reduce the frequency of attempt by increasing the accuracy of identifying the actual root cause. Furthermore, this proposed logic model contributes the new knowledge to close the existing knowledge gap of software application root cause analysis using AHP.

Index Terms—Analytic Hierarchy Process, Log File Analysis, Software Application Debugging.

I. INTRODUCTION

Software can be generally categorized into either (i) mobile application or (ii) software application. As for this paper, the focus is on software application, not mobile application. Software is adopted and used by today’s business operations. Especially for those companies which are using software as the Information Technology (IT) enabler strategically, the operation cost can be expensive for software licenses and maintenance. There are also companies which utilize software heavily just to sustain their business operations. Of course, there is no doubt on software that is impacting business operations or even human lives in many ways. It is the fact that wherever software application becomes crucial for processing the business transactions, it has lower toleration for downtime expected by the company management. This was clearly supported by Labels: Data Center, Downtime, www.evolven.com (2014), and indeed software application downtime can cause the business operation ceased.

In a multiple tiers environment, that involves client tier, business logic tier, application server tier, middle ware tier, and database tier. To select the related log events from the log files based on the time event when software application error occurred are required. However, at most of the time, collecting and filtering valid input information from log files for root cause analysis is time consuming. This statement is supported by Management Logic (2012) stated that "The most time consuming aspect of Root Cause Analysis (RCA). Practitioners must gather the all the evidence to fully understand the incident or failure.". On the other hand, Horvath, Kristof (2015) had also pointed out that "While the analysis itself can be time-consuming, the chance to mitigate or eliminate the root causes of several recurring problems / problem patterns is definitely worth the effort.". In addition, what is the appropriate category for logging the event information, and how to categorize the difference of error, debug, and fatal that should be fetched as the input information to the root cause analysis activity. In the situation that if the extensive event logging level is enabled, this can lead to excessive logging information generated. With that, there are two issues raised. The first issue is that, the performance of software application is reduced by comparing with before and after extensive event logging option is enabled. The second issue is that, the manual analysis activity is becoming much more difficult and even tedious to identify the root cause of the software application error. Therefore, in the software application development process, it is a great concern on how much detail event logging should be logged into the log file. At the same time, the event logging must mitigate the performance impact created to the software application. These mentioned concerns had also been highlighted by Loggly (2017) and Panda, A (2011). Hence, it is crucial to look for an efficient method to reduce the prolonging time at the root cause analysis activity.

II. SIGNIFICANT OF STUDY

The proposed research on the new logic model is focusing on reducing time duration on root cause analysis, and increasing accuracy of identifying the real error. With the new logic model, it helps to resume the software application service to the users and reduce impact to the business.
operation. This is because in today's rapid business competitive world, the longer time consuming on root cause analysis or so called the longer duration on trouble-shooting activities are totally unacceptable. Furthermore, it is a continuous battle for the IT support to face day-to-day software application error challenge in order to provide reliable up-time for the software application utilized in the business organization. Nevertheless, business companies must still continue to utilize their existing software applications (without incur any additional operation budget). At the same time, it must continue to allow the companies to save the investment budget on spending the capital amount to replace all or partial of the software applications. Without introduces any new software application, companies can avoid to re-train their users on using the new software applications. This new logic model is to get the valid error identified efficiently and to mitigate the time duration of root cause analysis prolonged. Over the years there were various researches had been done at this area such as consolidate the logs or integrate the logs for analysis. However, there had been very few attempts to propose an algorithm on root cause analysis beyond the software application layer. To support this, Valdman (2001) had showed more detailed log file analysis by comparing the past published techniques whereas other studies were only indicated intentions or suggestions. Moreover, there is no attempt to incorporate Analytic Hierarchy Process (AHP) for decision making on valid software application error. This can be supported by Vaidya and Kumar (2004) that they had reviewed Analytic Hierarchy Process (AHP) with 150 application papers in total. There is no knowledge in applying AHP on identifying valid error in software application boundary. Hence, this is a great potential in this research which brings contribution to business intelligent studies.

III. PROBLEM STATEMENT

A. The Problem
To conduct the root cause analysis activity on software application error, it is time consuming to identify the valid error.

B. The Events/Behavior
E1. Software application behaves in an unexpected manner or even it stops running due to error occurred.
E2. Analysis activity conducted manually by human would overlook important clue based on the given information.
E3. Lack of analysis experience would waste a lot of time when conducting analysis activity.
E4. Business users face difficulty to continue their daily tasks as the software application is unavailable or malfunction.
E5. The person who conducts the analysis activity will be under pressure on how fast to get the software application error rectified.
E6. The root cause analysis activity is conducted by human; it is very subjective to the person would make a right or wrong judgment on a software application error.

C. The Causes
C1. Software application defect escapes from software application testing and this defect has been released to production environment.
C2. Software application support engineer overlook the important clues and lead to the entire analysis activity taking loner time as expected.
C3. Software application support engineer is unable to identify the root cause in a speedy manner due to inexperience in analysis technique.
C4. Prolonging the downtime on software application would increase the volume of pending business transactions for processing.
C5. Prolonging software application downtime window would impact the percentage of software application availability over the entire calendar year. This would lead to penalty payout on breaching service level agreement signed for the software application support.
C6. The same software application error would be high chances to reoccur if the previous root cause analysis was based on human with the subjective mind set.
C7. High expectation on software application support team by the company management to resume software application service when software application service or certain functions are unavailable.

D. The Generalization to Become The Problem Statement
The time taken of root cause analysis on software application error carries crucial impact to the service restoration.

IV. RESEARCH OBJECTIVE
There are two crucial primary objectives to be achieved:
(1) To mitigate prolonging time duration on conducting root cause analysis activity.
(2) To improve accuracy on identifying the root cause whenever error is occurred.

With the primary objectives, it can derive the following sub objectives:

<table>
<thead>
<tr>
<th>No.</th>
<th>Sub Objectives</th>
<th>Expected Outputs</th>
<th>Methods To Accomplish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>To investigate the published log file analysis methods and techniques.</td>
<td>Research gap from the published log file analysis methods and techniques.</td>
<td>Literature Review</td>
</tr>
<tr>
<td>2</td>
<td>To identify any published or suggested steps or techniques on root cause analysis of software application can be adopted.</td>
<td>Standard practice on root cause analysis of software application.</td>
<td>Internet Search</td>
</tr>
<tr>
<td>3</td>
<td>To determine the common errors of the software application, Operating System, Network, database, and hardware.</td>
<td>Standard errors which will be incorporated into the proposed research.</td>
<td>Internet Search</td>
</tr>
<tr>
<td>4</td>
<td>To evaluate the Analytic Hierarchy Process (AHP) techniques from the published AHP documents.</td>
<td>Adoption of Analytic Hierarchy Process (AHP) on the proposed research.</td>
<td>Literature Review</td>
</tr>
<tr>
<td>5</td>
<td>To validate the overall time taken in the duration of root cause analysis activity.</td>
<td>Validation on the duration of before and after applying the proposed algorithm whether the duration of root cause analysis activity is shorter by how much in percentage.</td>
<td>Experiments</td>
</tr>
<tr>
<td>6</td>
<td>To validate the frequency of attempt for the accuracy on identifying the root cause whenever the actual error is occurred.</td>
<td>Validation the frequency of attempt for the accuracy on identifying the root cause whenever the actual error is occurred before and after the Prescriptive Analytical Logic Model (PAL) applied.</td>
<td>Experiments</td>
</tr>
</tbody>
</table>

V. LITERATURE REVIEW

Based on the past research, there are several strong points should be highlighted as follows:

(a) Stewart, D (2012) is focusing on debugging real-time software application error using logic analyzer debug macros.
(b) Eick, S, Nelson, M, and Schmidt, J (1994) they are focusing on presenting the error logs in a readable manner.
(c) Wendy, P and Dolores, R (1993) suggested to focus on error detection in software application at the time of software development and maintenance.
(d) Salfner, F and Tschirpke, S (2015) are focusing on analyzing error logs by applying the proposed algorithms in order to predict future failure.
(e) Murinová, J (2015) had attempted to integrate multiple log files from various software monitoring tool and network devices for better root cause analysis on Web application error. However, there is no proposed model stated in the research.
(f) Furthermore, there is “future work” of Murinová, J (2015) in the log file analysis to enable better comparison and troubleshooting capabilities. This is a potential area where the log file analysis technique can be incorporated with AHP approach.
(g) Even until Landauer et al (2018), they introduced an unsupervised cluster evolution approach that is a self-learning algorithm that detects anomalies in terms of conducting log file analysis. However, this approach is under machine learning rather than AHP.
(h) Vaidya and Kumar (2004) had reviewed Analytic Hierarchy Process (AHP) with 150 application papers in total, and there is no knowledge in applying AHP on identifying valid software application error. This means, the proposed logic model can contribute new knowledge in the area of software application root cause analysis using AHP.

By comparing the above secondary data, the technique to identify the root cause of software application in the past is different with the new logic model that is proposed in this paper. For the past research, they focus on software application boundary whereby the proposed new logic model focuses horizontally on all possible boundaries. In addition, due to the focus boundary is different, it leads to the technique to identify the root cause of software application will also be different. Therefore, after the literature review process. It is no doubt that the contributions of PAL under the proposed methodology are:

(i) Literature review indicates that there is “future work” of Murinová, J (2015) in the log file analysis to enable better comparison and troubleshooting capabilities. This is a potential area where the log file analysis technique can be incorporated with AHP approach. (ii) Since there is no published article showing that AHP had been applied to software application error analysis, the proposed research is to fill up the “knowledge gap”. Hence, this is a great potential to propose a new logic model towards developing an algorithm for software application error analysis.

VI. PROPOSED LOGIC MODEL DESIGN

In a server (regardless it is physical or virtual) box, it stacks with multiple layers. The Hardware (physical or virtual hardware) residents at the very bottom layer. Followed by Networking layer stacks on top of Hardware layer. The Operating System layer will be on top of Networking layer. After all, then the Software Application is

DOI: http://dx.doi.org/10.24018/ejece.2019.3.6.160
on top of Operating System layer. With such multiple layers in a server box, that most common logs can be found are such as:

a) System Monitoring log for logging the events such as Central Processing Unit (CPU), System Memory and Hard Disk (direct attached disk or storage disk).

b) Network Monitoring log for logging the events such as network communication of the Local Area Network (LAN).

c) Operating System log for logging the events of the Operating System behavior.

d) Database log for logging the events of the Database behavior.

e) Software Application log for logging the events of the Software Application behavior.

![Diagram](image)

Fig. 2. Multiple layers in a server box.

The proposed scope of this research is to define the algorithm in a proposed logic model. The algorithm would analyze different logs but these logs must be relevant. By having the proposed logic model in the production environment, the logic model is required to react to software application error when the error is detected in the software application log file. The algorithm is required to retrieve the related log files through System Monitoring log, Network Monitoring log, Operating System log and Database log for further analysis. The proposed algorithm mainly consists of two analysis areas, which are simple and complex analysis to form a Prescriptive Analytical Logic Model (PAL).

i. For simple analysis area, it is required to build the predefined logic to handle the common software application error. The PAL will prompt out a set of predefined unanswered questions one by one and allow the user to provide answers to it. These predefined questions are on common software application errors and user has to choose the preferred action from the suggested predefined activities. For example, in an event of the software application error. If both predefined unanswered questions one by one and allow the user to provide answers to it. These predefined questions are on common software application errors and user has to choose the preferred action from the suggested predefined activities. For example, in an event of the software application process is stopped. User can possible choose to restart the software application process.

ii. For complex analysis area, it is required to build a logic which collects necessary log events as data from the involved software applications. The collected log files will be served as input information at the initial stage. With the collected data, this model will base on the past incidents determined as the system behavior. By combined with the predefined templates, the automated analysis activities will be triggered and finally generate the analysis outcome along with the suggested resolution steps and action to the IT support team. This complex analysis would have three different modes which are “manual”, “semi-auto” and “fully-auto” offered to the IT support team. As for the complex analysis area, by predicting the software application behavior, it performs the suggested steps and carry out the action against the software application error based on the analysis. This will prevent future application failure based on the permission given to the offered mode by the IT support team. By focusing into the re-occur software application errors, these errors occur in a specific pattern or feature, and the solution is often straight forward (can be applied after validating the specific pattern or feature) to resolve the incidents. The human involvement on this type of incidents would require less analysis but more on validating activities. Hence if the validating activities can be predefined into a checklist, the logic model can pick up the ultimate predefined solution and react to the incident automatically. This can be achieved by the combination of the answers (yield from the validating activities in the checklist). This would be the preferred method in the logic model that handles the common software application incidents. This logic can be named as simple analysis. The same simple analysis logic can be applied to manage Server (a physical or virtual box running a vendor Operating System) or even Networking devices (such as switch or router) if they have incidents occur in the specific pattern or feature. Bare in mind that the software application errors which have no uniform pattern or feature. Therefore, for this type of software application errors, the percentage of human involvement is high. Reason is as simple as the person who handles the incident requires to obtain the software application log files and to search any similar error logged in the past manually. These files and records are treated as the input information. With the input information obtained, the person conducts the analysis activities before the person can identify the software application error root cause. Only the preferred resolution steps is agreed then it is applied to resolve the software application error. For the first time occurring software application error. If both yielding input information activities and analysis activities can be automated. Base on the outcome of the analysis activities, human expects to see a list down of each possible root cause along with the proposed resolution steps in a complete list. With that, the decision is on the person to choose which is the the preferred option. If the person chooses to proceed with the suggested resolution steps, then the person will receive the final question. The question is expecting the response from the person, whether agrees to let the automated activities execute the same suggested resolution steps automatically in the future if the same incident occurs again.

The simple analysis can be existed independently at the initial stage. However, when the specific number of reoccur incidents hits. The complex analysis will be activated to perform the required analysis activities automatically. It will
produce the complete analysis report and suggestion(s). Based on this suggested design, the complex analysis would have a loosely but it is fairly important relationship with the simple analysis. This is because the complex analysis needs to understand how many times the simple analysis has handled the same incidents in the past. This information is crucial to make a decision on suggesting the reasonable resolution steps to the human after the complex analysis produces the analysis report. In addition, since the PAL is required to collect logs from different monitoring systems and software applications based on the given time as the primary key. Therefore, it is crucial that the server time on all the involved monitoring systems and software applications must be synchronized by Network Time Protocol (NTP) server. As per Masterclock (2016), “Accurate time stamping is key to root-cause analysis, determining when problems occurred and finding correlations. If network devices are out of sync by a few milliseconds or, in extreme cases a few seconds, it can be very difficult for network administrators to determine the sequence of events.”.

The PAL is designed to have three major sections. Each section has multiple modules. Then each module consists of the proposed algorithm. The details are shown in the following table.

<table>
<thead>
<tr>
<th>No.</th>
<th>Section</th>
<th>Module</th>
<th>Proposed Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software Application Log Extraction</td>
<td>Error Search (in Software Application Log)</td>
<td>Extract software application log and identify error event based on error keywords stored in predefined error list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Extraction (From All Related Logs)</td>
<td>Based on the identified error event logged into software application log, using the time of error event as the dependency, locate additional log data from different software application databases, by either using the predefined database schema, or using the time information as the key reference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Integration</td>
<td>Integrate various row of log obtained from different software application databases.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard Error Verification</td>
<td>Identify whether the newly reported software application error can be found in the standard error list by cross-checking the knowledge base database of PAL. Note: If the “Standard Error Verification” which cannot be found in the standard error list, proceed to “Complex Analysis Conduction”.</td>
</tr>
<tr>
<td>2</td>
<td>Simple Analysis Conduction</td>
<td>Error Occurrence Verification</td>
<td>Identify whether the newly reported software application error is first time occurrence or re-occurrence by cross-checking the knowledge base database of PAL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error Classification &amp; Analysis</td>
<td>Identify error log data and categorize the error log data for analysis under the defined software application error classification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolution Identification</td>
<td>Identify the preferred resolution to the software application based the outcome of analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolution Deployment</td>
<td>Apply resolution based on the predefined configuration of the PAL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update Knowledge Base Database</td>
<td>Store the analysis activity and resolution information in the knowledge base database of the PAL.</td>
</tr>
<tr>
<td>3</td>
<td>Complex Analysis Conduction</td>
<td>Error Occurrence Verification (If the “Standard Error Verification” which cannot be found in the standard error list)</td>
<td>Identify whether the newly reported software application error is re-occurrence by cross-checking the knowledge base database of PAL. Note: If this error is re-occurrence, then retrieve the past analysis information for understanding the past analysis experience.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error Classification &amp; Analysis</td>
<td>Identify error log data and categorize the error log data for analysis under the defined software application error classification. The error classification under this proposed design module is the next level of error classification different from “Simple Analysis Conduction” module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHP to finalize the Valid Software Application Errors</td>
<td>Allocate weight to each possible software application error based on Analytic hierarchy process (AHP). Shortlist the software application error under the highest weight. Analyze the selected log data for shortlisted software application errors and define possible resolution option.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHP to finalize the Preferred Resolution</td>
<td>Allocate weight to each possible resolution option based on AHP. Shortlist the preferred resolution option under the predefined condition.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolution Deployment</td>
<td>Deploy the preferred resolution option to fix the software application error under the predefined condition.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update Knowledge Base Database</td>
<td>Store the analysis result and resolution action into a database which is associated to the PAL for future reference and knowledge base activities.</td>
</tr>
</tbody>
</table>
To identify the software application error along may not be straightforward if it is other factors forming a chain reaction to cause the software application fails. The root cause can be hard to determine if the log files from the involved software applications, Database, Operating System, and Networking are not factored into the root cause analysis process. The primary objective is to reduce the duration on conducting the root cause analysis for software application error.

As with the proposed algorithm inside the PAL model, this algorithm consists the Analytic Hierarchy Process (AHP) process for decision making on both identifying the root cause and applying the preferred resolution action. This AHP process plays an important role in the PAL, and this AHP process was introduced by Thomas L. Saaty. The source can be referenced in Wikipedia (2015). Therefore, the design of AHP hierarchy will depend on the software application errors and other involved errors found from different layers. These errors become the participants. The participants can be grouped further based on the error categories. Once the hierarchy is constructed, the participants analyze it through a series of pairwise comparisons that derive numerical scales of measurement for the nodes. Then, the priorities are associated with the nodes, and the nodes carry weights of the criteria or alternatives. Based on the priority, the valid error and the preferred resolution can be decided. In short, AHP is a decision making process as a tool, which can be utilized to weight the both the possible root causes and possible resolutions under a hierarchical structure. Then, narrow down to the final root cause, and decide the best resolution among the shortlisted resolutions. Ideally, the AHP will be utilized at the following two (highlighted) specific proposed activity under the PAL.

By referring to the proposed algorithm that are under “Complex Analysis Conduction” module in the above table. These proposed activities in the algorithm play an important role of AHP process. The activity details are explained respectively as follows:

a) It helps to identify the possible software application errors and to filter out all the false alarms. On those possible software application errors, each error will be further identified by the its error characteristic and categorized into specific error category under a predefined software application error category list.

b) It will perform assigning the weight to each possible software application error based on the error category the impact level of each error within the category. With the assigned weight on each error, it would easily shortlist the highest priority of software application error as the crucial error to be fixed.

c) It handles the process of assigning weight on each possible resolution after the analysis activities has been conducted as section 6. This is because possible circumstance can be happened when two similar resolutions are selected, but we need to identify which is the most suitable resolution can be applied to resolve the software application error.

d) It helps to identify the final preferred resolution to be applied for the crucial error after evaluating the weight, and this action will isolate multiple resolution actions to be applied to the crucial error to be fixed.

With AHP process, the proposed PAL under the complex analysis area will conduct the important actions on filtering errors and identify which is the crucial error to be fixed, and later it also helps to identify which is the preferred resolution action to be applied on the crucial error to improve effectiveness on resolving the software application error.

VII. CONCLUSION

Software applications today normally come along with the event logging feature. By depending software application log file alone is not sufficient to conduct software application root cause analysis whenever error is occurred beyond the software application layer. This is because a server box consists of Hardware, Networking, Operating System and Software Application layers. Whenever the hardware memory is running low, or the central processing unit (CPU) resource is saturated, or even hard disk space is running low, all these causes would lead to the software application crashes or unstable stage. Hence a proposed logic model named PAL is required. This proposed logic model incorporated with AHP to conduct the root cause analysis activity on software application. It must target to increase the accuracy for error identification, and to reduce the prolonging time spent on the duration of root cause analysis. Therefore, this is a good potential to contribute new knowledge to the software application analysis.

ACKNOWLEDGEMENTS

Special thanks to Tatana Zitkova for providing encouragement on my proposed research.

REFERENCES


DOI: http://dx.doi.org/10.24018/ejece.2019.3.6.160
spend-in-production-troubleshooting (Accessed on 14th November 2015)


