Abstract—There are many causes would lead to the software application crashes or unstable stage. In the event that this circumstance happens, the software applications will never able to log error event correctly into their error log files. Hence, if the root cause analysis is solely depending on software application error log files, it is insufficient to identify the actual error accurately. This is the good opportunity and potential to introduce Prescriptive Analytical Logic Model (PAL) under the proposed methodology. In summary, the contributions of PAL under the proposed methodology are: (1) Literature review indicates that there is “future work” of Murinova, 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. (2) Since there is no published article showing that AHP had been applied to software application error analysis, and therefore the proposed research is to fill up the “knowledge gap” in this area.

Index Terms—Analytic Hierarchy Process, Log File Analysis, Software Application Root Cause Analysis, Software Application Error Analysis, Top-Down Design Approach

I. INTRODUCTION

Software is being used by human widely in today’s modern world. Many companies utilize it heavily to process daily transactions in order to sustain their operations. Some companies even utilize it as an Information Technology (IT) enabler to create competitive advantage in the business industry. Hosting a software can be many ways. However, despite the piece of software whether it is hosted on premise or in the cloud, software has become a crucial service and this service cannot afford to be down. On the other hand, software in general can be categorized into software application and mobile application. This paper focuses on only software application.

It is common that a software application encounters error and stops functioning in the production environment. The cause can be either within the software application layer or it may be due to any other factor beyond the software application layer. This is because software application is required to host on a server box (regardless it is a physical hardware or a virtual machine), and the server box is required to have Operating System running in it. As for the software application that is hosted on a physical server box, it stacks on top of an Operating System with network communication ability to interact with other server boxes. With the network communication infrastructure ability, the data of the software application may be required to store inside a database. This database can be hosted on another server box. Due to such complexity of the infrastructure, and hence, to identify the root cause of the actual failure point for a software application will be insufficient by analyzing the software application log file alone.

II. MOTIVE OF THE STUDY

The main reasons to cause the motivation of this paper are shown as follows:

1. With the software application error log file alone, it may not be adequate for analysis to identify the root cause of software application error if the cause is outside the software application boundary.

2. Time consuming to read through the software application log file manually during root cause analysis activity.

3. 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.

4. When software application support team becomes costly, it leads to high expectation from company’s management to have the lowest software application down-time. This has caused the job of software application support team to become extremely stressful.

The consequences of the reasons:

(a) The business users face difficulty to continue their daily tasks as the software application is unavailable or malfunction.

(b) 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.

III. JUSTIFICATION THE SELECTED TOPIC

There are several important points to justify the studies of this paper. First of all, software applications cannot afford to have downtime as it can cause the business operation to cease. This has been stated clearly at Labels: Data Center, Downtime, www.evolven.com (2014). Secondly, REDDIT (2015) and StackOverflow (2015) supported that the time spent (duration) on analyzing or trouble-shooting the
software application error can be prolonged. This is due to identify the root cause can be difficult if the input information such as log files or past history of the software application error is huge and the analysis activity is conducted manually. Thirdly, the root cause analysis activity is conducted by human, and it is very subjective to the person would make a right or wrong judgment on a software application error. Lastly, when software application support team becomes costly, it leads to high expectation from company’s management to have the lowest software application downtime. This has caused the job of software application support team to become extremely stressful. Margulius (2006) mentioned that IT jobs become the most stressful job among other jobs such as engineering, sales, finance, HR, and others.

IV. SIGNIFICANT OF THE STUDY

In a multiple-tier environment, where the front-end is the Web tier providing the Web interface interacts with the users in the Internet. There are servers at the service provider premise hosting the business logic tier, application tier and the database tier. In addition, there is a middleware hosted in the application tier to provide a messaging queue to convey the data files to other application server. Then, on another application server (which it is still in the application tier) is hosting the backend agents. These backend agents extract the data from data files for processing and later uploading the processed data into the database under the database tier. Finally, the database at the database tier manages and handles the data management. There are mainly three different perspectives that should be looked into as for the significance of the study, which are:

- **IT innovation perspective** - From the IT innovation perspective, by looking at the proposed algorithm in the proposed logic model, named Prescriptive Analytical Logic Model (PAL). It improves the software analysis technique into a more human-like intelligent to identify and to decide which the valid software application error is.

- **Business operation perspective** - From the business operation perspective, with understanding the importance of software applications are being adopted in the today’s business operation. The business impact is high when these highly depended software applications become non-operative during business hours.

- **IT support perspective** - Software application support team has high challenge to resolve the software application errors in a very slim duration. At the same time the support team members have equally high tension to ensure the overall duration of software application availability during business hours in a year. Indeed, suggesting a new logic model to incorporate a set of proposed algorithm to react or even to pro-act to handle software application errors is a necessity to the software application support team. This creates a better position of responding to any software application error, to fulfill service level agreement (SLA) signed with the company management, to deliver quality resolution to software application error, and to eliminate software application error permanently (if possible).

V. LITERATURE REVIEW

In the past research, there are tools and techniques on software application error log analysis had been introduced. However, the scope, area of focus as well as the point of view varied from each other.

- i. David (2012) is focusing on debugging real-time software application error using logic analyzer debug macros,
- ii. Stephen, Michael and Jeffery (1994) are focusing on presenting the error logs in a readable manner.
- iii. Wendy (1993) is solely focusing on error detection in software application at the time of software development and maintenance, and
- iv. Felix and Steffen (2015) are focusing on analyzing error logs by applying the proposed algorithms in order to predict future failure, so called online failure prediction.
- v. Murinová (2015) has an overview of monitoring and log analysis. The paper had the details such as the specifics of application log analysis and log file formats definitions. Furthermore, the author mentioned various available systems for log analysis. Which are from both proprietary and open-source approaches. It is all compared and categorized with overview comparison tables of supported functionality. Although the paper mainly focuses on the Web log analysis, which is the analysis of logs generated in web communication and interaction. A good technique can be learned and applied on the proposed research which is the technique of having the multiple log analysis systems were compared and categorized. Especially, the categorization was based on the information available about their functionality in attempt to get an overview of possible solutions varying by requirements. There is an important item that Murinová, J (2015) stated a “future work” in the paper. This “future work” will be the key component to provide better comparison and troubleshooting capabilities in the algorithm. The first item is using nested queries to improve functionality. It means that multiple log messages are stored in a specific session and then uses nested queries to filter out unwanted log messages. The nested queries can be integrated into a custom software application and by executing the software application it will do the filtering works. Send item is to standardize the date and time format for better log integration and filtering during the session of executing the nested queries. However, both items are not implemented yet based on the published paper of Murinová (2015). In the proposed research, these two items can be incorporated into the proposed algorithm. Of course these two items are also subjected for enhancement under any possible opportunity in the proposed algorithm. In addition, the missing piece of Murinová (2015) is where the proposed algorithm does not apply any machine learning or AHP technique to perform the decision.
making of the valid error and the preferred resolution based on the valid error is identified. Therefore, the proposed algorithm of this proposed research will incorporate the AHP for automating the decision making of valid error, as well as or automating the decision making of preferred resolution.

vi. Valdman (2001) had described some features are presented in more detail while others are just intentions or suggestions, which is given a research gap which we can continue this area of research to incorporate the log data searching technique (introduced in the past research) into this research of prescriptive analytical logic. Apart of adopting techniques of previous reaches, this research introduces AHP to carry out decision making actions such as shortlisting the valid software application error among multiple software application errors are found, and shortlisting the final resolution to the software application error identified. This is because with AHP applied, it helps the proposed model to be more accurate to evaluate and identify the valid software application error, and of course using AHP again to be more accurate to evaluate and identify the preferred resolution to the software application error.

With the above secondary data to compare with the proposed logic model, it is understood that the boundary of the focus regarding to software application analysis are very much different. In addition, even the technique to identify the root cause of the real error is also different. They focus on software application boundary whereby the proposed 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 also different.

VI. PROBLEM STATEMENT

From the motive of study and the significant of study, it helps to derive the main problem. Which is, it is time consuming to identify the root cause during the root cause analysis activity is conducted whenever software application error has been occurred. These are the possible causes leading to the software application failure as well as making the root cause of the failure hard to be determined.

1. Software application defect escapes from software application testing and this defect has been released to production environment.
2. Software application support engineer overlook the important clues and lead to the entire analysis activity taking longer time as expected.
3. Software application support engineer is unable to identify the root cause in a speedy manner due to inexperience in analysis technique.
4. Prolonging the downtime on software application would increase the volume of pending business transactions for processing.
5. 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.
6. 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.
7. 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.

Hence, to generalize the problem statement, it is:
“The time duration on conducting the root cause analysis activity carries crucial impact to the service restoration to the business operation”.

VII. RESEARCH OBJECTIVES

There are several events that lead to the objectives on resolving the actual problems. Here are the events shown as follows.

1. Software application behaves in an unexpected manner or even it stops running due to error occurred.
2. Analysis activity conducted manually by human would overlook important clue based on the given information.
3. Lack of analysis experience would waste a lot of time when conducting analysis activity.
4. Business users face difficulty to continue their daily tasks as the software application is unavailable or malfunction.
5. The person who conducts the analysis activity will be under pressure on how fast to get the software application error rectified.
6. 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.

With all do respect to the mentioned events, the primary objectives can be targeted:
(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.

VIII. PROPOSED DESIGN

The proposed logic model is the outcome of the objective to achieve. This logic model can later be developed as a software application plug-in to reside at the logic tier (without disturbing all the existing software applications' functionality) to execute its functionality.
The proposed software application analysis algorithm analyses multiple software application log files.

Fig. 1. The proposed software application analysis algorithm analyses multiple software application log files

The logic model includes the following algorithm activities shown as follows:

<table>
<thead>
<tr>
<th>Step</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>With the granted software application access, use “time” field as the linkage to retrieve and to relate all the necessary log data across multiple databases.</td>
</tr>
<tr>
<td>2</td>
<td>Identify whether the newly reported software application error is first time occurrence or re-occurrence by cross-checking the database which is associated to the prescriptive analytical logic.</td>
</tr>
<tr>
<td>3</td>
<td>Identify possible log data and select the necessary log data for analysis under the defined software application error classification.</td>
</tr>
<tr>
<td>4</td>
<td>Allocate weight to each possible software application error based on Analytic Hierarchy Process (AHP).</td>
</tr>
<tr>
<td>5</td>
<td>Shortlist the software application error under the highest assigned weight.</td>
</tr>
<tr>
<td>6</td>
<td>Analyze the selected log data for shortlisted software application errors and define possible resolution option.</td>
</tr>
<tr>
<td>7</td>
<td>Allocate weight to each possible resolution option based on AHP.</td>
</tr>
<tr>
<td>8</td>
<td>Shortlist the preferred resolution option under the highest assigned weight.</td>
</tr>
<tr>
<td>9</td>
<td>Deploy the preferred resolution option to fix the software application error under the predefined condition if the resolution does not involve SDLC. Otherwise, produce the Analysis Report.</td>
</tr>
<tr>
<td>10</td>
<td>Store the analysis result and resolution action into a database which is associated to the prescriptive analytical logic for future reference and knowledge base activities.</td>
</tr>
</tbody>
</table>

IX. ALGORITHM DESIGN APPROACH

According to the information provided by Quora (2016) and Wikipedia (2018), there are basically two different approaches for algorithm design, which are top-down and bottom-up approaches. The discussion in this paper is only applying top-down design approach for the proposed logic model. The top-down design approach is splitting the entire design boundary into two or more for the smaller design areas in a hierarchy structure manner. Then it is followed by designing the function in each smaller design area. Based on the algorithm stated in Table 1, there are several modules that can be proposed. Which means, there are three major modules can be consisted inside the PAL for log file extraction.

A. Software Application Log Extraction Module

In this proposed module design, it consistently read the software application log and extract specific software application row of log which is suspected to be error log event. To identify the error log event is based on error keywords stored in predefined error list. The support reason of consistent scanning and extracting software application error log is to proactively discover any error event logged into software application log, and react with immediate action of analysis. Thus this action which is the scanning and extracting software application error log must be always the first action to be performed in order to identify any error has been occurred. Only then followed by the decision of entering either simple or complex analysis conduction section based on the predefined conditions.

With the extracted row of software application log, the module designer can utilize the time of error event as the dependency, to 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. Lastly, it is required to integrate various rows of log obtained from different software application databases to form a dataset for analysis.

Fig. 2. The Software Application Log Extraction module inside the top-down algorithm design
B. Simple Analysis Conduction Module

In this proposed module design, there are several important activities for identification of the new-found software application error before the action of applying predefined resolution. In another word, it is targeting to tackle the new-found software application which is matched one of the standard software application error list and apply the standard resolution action. Indeed, firstly it identifies whether the newly reported software application error can be found in the standard error list by cross-checking the knowledge base database of PAL and if the “Standard Error Verification” which cannot be found in the standard error list, proceed to “Complex Analysis Conduction”. Secondly, it identifies whether the newly reported software application error is first time occurrence or re-occurrence by cross-checking the knowledge base database of PAL. Thirdly, it identifies the error log data and categorizes the error log data for analysis under the defined software application error classification. Fourthly identifies the preferred resolution to the software application based the outcome of analysis. Fifthly, it applies resolution based on the predefined configuration of the PAL. Lastly it stores the analysis activity and resolution information in the knowledge base database of the PAL.

Fig. 3. The Simple Analysis Conduction module inside the top-down algorithm design

C. Complex Analysis Conduction Module

In this proposed module design, it is required to have the capability to identify whether:

(a) It is a first time (newly) reported software application error but falls into the error category that no standard resolution can be resolved, or
(b) It is re-occurrence software application error.

The identification is through the validation process by cross-checking the knowledge-based database of the PAL. Indeed, if this error is identified as a re-occurrence case, then the next activity is to retrieve the past analysis information to further understand what the past analysis result and the conducted activity were as the input information of the current found software application error. On the next proposed design activity under this proposed module design, it is important to identify the current error log data by performing the next level of error categorization under the defined software application error classification within this proposed design module. This error classification is different from “Simple Analysis Conduction” module as this error classification will involve the boundary beyond the software application layer, which means the error involves the boundary of Operating System, Networking, Database, middle-ware, etc. In this proposed design module, there is no activity to perform log integration due to this action is already conducted back in Software Application Log Extraction Module, which has extracted and integrated the various rows of log obtained from the different software application databases.

With the integrated log data, the next crucial action is to allocate weight to each possible software application error based on Analytic hierarchy process (AHP) and shortlist the software application error under the highest weight.

1) Module: AHP Finalization Valid Software Application Error

By referring to Table I, (a) The activity number 4 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) The activity number 5 will perform

DOI: http://dx.doi.org/10.24018/ejece.2020.4.1.184
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.

Once the software application error is identified to be the top priority to be resolved, the analysis activity is required to define possible resolution option against the shortlisted software application error based on the error falls into which predefined error category, and the followed by applying the predefined mechanism to identify the possible resolutions to the error. Finally allocate weight to each possible resolution option based on AHP. With that, it then proceeds to shortlist the preferred resolution option whichever resolution is with the highest weight. Once the resolution is finalized, the following action is to deploy the preferred resolution option to fix the software application error under the predefined condition.

2) Module: AHP Finalization Preferred Resolution

By referring to Table I, (a) The activity number 7 handles the process of assigning weight on each possible resolution after the analysis activities has been conducted at 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. (b) The activity number 8 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.

Finally, all the analysis and action of resolution will be stored into a database which is associated to the PAL for future reference and knowledge-based activities.

3) Module: Update Knowledge Base Database

By referring to Table I, the activity number 10 consists of (a) Store the past analysis activities and resolution actions into table forms under the predefined database schema. (b) Allow to use Structured Query Language (SQL) statement to retrieve the past analysis activities and resolution actions from the tables.

![Fig. 4. The Complex Analysis Conduction module inside the top-down algorithm design](image)

4) Top-Down Algorithm Design - The Hierarchy and All Modules

DOI: http://dx.doi.org/10.24018/ejece.2020.4.1.184
X. NETWORK TIME PROTOCOL (NTP) SERVER

As “time” is the key factor for log event retrieval from the involved software applications within the target environment, hence the prerequisites of the PAL model are shown as follows;


b. All involved servers must point to NTP Server to synchronize their server time.

Whenever a software application involves in a multi-tier environment, it is good to standardize the server time across all the involved servers. The supporting reason is that the proposed prescriptive logic model is required to collect log data from different software applications based on the specific given time as the primary key, and therefore the server time on all the involved servers are required to have the network time synchronization. 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.”.

XI. CONCLUSION

Despite the technology innovation in the 21st century, software application is still an important tool to sustain business operation. However, due to the complexity of a software application setup in a multi-tier environment, it is fairly hard to determine the root cause whenever the software application is facing malfunction or even failure. In such circumstance, a strategy on identifying the root cause is required. The strategy is to propose an algorithm that is sitting at the logic tier and conduct the root cause analysis without creating any interference to the existing software application execution. The proposed algorithm resides in a proposed logic model named PAL that will not only conduct the root cause analysis. It can in fact provide resolution to the root cause based on the result of the root cause analysis.

To design the PAL, it is adopted the top-down design approach, that is splitting the entire design boundary into two or more smaller design areas in a hierarchy structure manner. The final PAL under top-down design approach will serve as a guidance or even as a software development standard for the logic model. This PAL will help to mitigate prolonging time duration on conducting root cause analysis activity, as well as to improve accuracy on identifying the root cause whenever error is occurred.
REFERENCES


DOI: http://dx.doi.org/10.24018/ejeece.2020.4.1.184