The Root Cause Analysis Algorithm Design Incorporated with Analytic Hierarchy Process for Software Application Error

Hoo Meng Wong and Sagaya Sabestinal Amalathas

Abstract—Software application normally comes along with its event logging feature today. However, with only the software application event log would be hard to determine the root cause if the error occurs beyond the software application boundary. In such circumstance, the time duration of conducting the root cause analysis activity will be easily prolonged. In order to identify the root cause more effectively, multiple event log files from different boundaries that are involved with software application such as Operating System, Networking, Middleware, Database Management System and etc are required. Therefore it will add additional complexity to the root cause analysis process. With such challenge, a new analysis approach is proposed. This new analysis approach is a logic model incorporated with Analytic Hierarchy Process (AHP). It will be sitting at the logic tier to conduct the analysis without interferes the existing software application. The objectives of the logic model 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. Furthermore, this proposed logic model contributes the new knowledge of analysis approach which helps to close the existing knowledge gap in AHP, that can be applied for software application root cause analysis.

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

I. INTRODUCTION

In today's business world, application is commonly adopted in any business operation such as accounting application, inventory application, payroll application, and etc. Application basically can be categorized into mobile application and software application. However, this paper is focusing on software application. According to the information contributed by both Mercer, Edward (2015) and SuccessFactors (2015), business companies are relying on software applications to sustain, to continue, or even to increase business productivity at their business operations as usual (BAU) in today's IT era. Indeed whenever any error arises during its execution, business users will face difficulty to continue their daily tasks to process business transactions. The software application support team will be under pressure on how fast to get the software application error rectified and resolved. In order to resume the availability of software application back to business users. Especially for those business operations which are heavily depending on software application to handle daily business transactions. The support team will be highly impacted under such situation that

software application is unavailable whenever the error is encountered.

On the other hand, a software application requires adequate server resources (such as Central Processing Unit (CPU), Memory, Disk Space, and Network Bandwidth) for execution. Only under adequate server resource such condition, accurate event information can be successfully logged into its log file. However, during software application execution, how much detail level of software application error should be captured and logged into the software application error log would be enough. The higher detail of the log information, it would increase the risk for performance impact to the software application during its execution. In addition, SANS (2001) and Grabner, Andreas (2012) mentioned that, detail event information is logged which may not be easily read and understand by human because by reading at a common software application error log file. It can consist of complex structure which can be hard to interpret without any written guidance. However, highly detail of a software application error log file would only lead to a situation to cause high consumption of computing resource such as CPU, memory, and disk space on a server (regardless it is a physical or virtual server) which is utilized to hosting this piece of software application.

Furthermore, 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. Incidents are such as any recent operating system patching activity had been conducted or hardware configuration change request had been conducted recently, it can cause the software application running unstable. In such scenario, the software application errors did not specifically indicate the root cause, and eventually the entire software application would require a full restoration from the backup tape along with database consistency check to validate the software application functions and data retrieval from its database. Time consuming will be longer to resume the software application for the business users, and the actual of fact (root cause) remains unknown.

II. SIGNIFICANT OF THE STUDY

This paper is to propose a new approach on root cause analysis for software application. The design of algorithm is

incorporated with Analytic Hierarchy Process (AHP), to perform the decision making on the root cause analysis based on multiple log files that are related to the software application in the enterprise environment. These involved log files are from the Operating System where the software application is running on, Networking, Middleware, Database System that software application is interacting with, and etc. The benefit of the proposed algorithm is to get the valid error identified efficiently and to mitigate the time duration of root cause analysis prolonged.

III. PROBLEM STATEMENT

The main concern arises at the companies which are running their business operations in a high competitive mode, especially for those companies that are heavily depending on Information Technology as the enabler. These companies are such as airline, credit card company, bank, insurance company and etc. They would not accept a longer downtime of their software application. This is because due to the high volume of transactions made within minute or even second via the said software application, they will face penalty or lost if the business transactions cannot be processed on time. Hence, whenever the software application is at downtime, there is no room to prolong the analysis activity of identifying the root cause. With the given justification, the problem statement is derived as “To conduct the root cause analysis activity on software application error, it is time consuming to identify the valid error.”

IV. RESEARCH OBJECTIVES

The business organizations utilize the software application as the IT enabler to strengthen its business operation. By lower down the risk of the software application error and improve the reliability of utilizing the software application would bring business advantages to compete in today's the business industry. Therefore, from the problem statement, the research objectives must be achieved by:-

- (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.

V. LITERATURE REVIEW

There are tools and techniques on software application error log analysis have been introduced, however the scope and focus vary from each other, such as:-

- i. David (2012) is focusing on debugging real-time software application error using logic analyzer debug macros,
- ii. Stephen, Michael and Jeffery (1994) they 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.

This is because as for ordinary software application error log analysis activity, it focus on software application error log

obtained from only software application file or database (depending the log is keeping in a file or into a database's table). As per Valdman (2001) mentioned that this kind of software application log is useful for debugging and profiling purposes if the software application error is within the software application boundary. This is because, Valdman (2001) introduced a formal notation for finding various data relations in heterogeneous tables. On the other hand, Murínová (2015) has an overview of monitoring and log analysis, specifics of application log analysis and log file formats definitions. He mentioned various available systems for log analysis both proprietary and open-source are compared and categorized with overview comparison tables of supported functionality. In his thesis, although it 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. 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. Indeed, 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 paper 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 algorithm 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.

TABLE I: THE ARRANGEMENT OF CHANNELS

Channels	Group 1	Group 2	...	Group c
Main channel	Channel 1	Channel 2	...	Channel c
Assistant channel	Channel 2	Channel 3	...	Channel 1

VI. PROPOSED ALGORITHM DESIGN

Regardless any given strong reason, there are three major concerns here. The primary concern is that the algorithm design should avoid any footprint left on the involved software applications. As for the secondary concern, it requires a field to link up all the related log events from the related log files. Since in every log event, the date and time are required. The best linkage to relate all the necessary log events across multiple software applications, which is the “time” field. Hence, all the involved software applications or

even the Operating System that these involved software applications are running on, all should point to the Network Time Protocol (NTP) Server, in order to synchronize the time across. Masterclock (2016) had also explained the importance of an NTP Server.

The initial proposed algorithm is shown as the following Table.

Activity	Algorithm
1	Collect or integrate all the related log files obtained from different software applications based on the given time.
2	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.
3	Identify possible log data and select the necessary log data for analysis under the defined software application error classification.
4	Allocate weight to each possible software application error based on <i>Analytic Hierarchy Process (AHP)</i> .
5	Shortlist the software application error under the highest weight.
6	Analyze the selected log data for shortlisted software application errors and define possible resolution option.
7	Allocate weight to each possible resolution option based on AHP.
8	Shortlist the preferred resolution option under the highest weight.
9	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.
10	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.

Table 1- The main activities in the proposed algorithm.

When a major software application error occurs, it would trigger a list of related and even non-related errors or warning messages in the software application log. With such the analysis process will need to retrieve software application log information under a specific time duration from the software application log file as for the input information of the analysis process. The AHP process is applied to allocate weight to the software application error as well as errors found from different layers under the same “timing”. This AHP process is playing an important role to allocate weight to each possible error based on the error aspect. Whenever the weight is allocated on each error, the AHP process will shortlist the error under the highest weight. Once the shortlisted error is determined, the AHP process will proceed to evaluate all the possible resolutions based on the weight allocation to each possible resolution. Finally, the AHP will shortlist the resolution based on the highest weight. If the possible resolution is more than one, then the proposed algorithm will generate a report by providing the possible resolution option(s).

By looking at the proposed algorithm design, it should be strictly based on the 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. From the understanding of the references, basically top-down design approach is splitting the entire design boundary into two or more smaller design areas in a hierarchy structure manner from top as a single node and branch out to the bottom like the root of a tree. Oppositely bottom-up design approach is listing down every specific design function involved in the design boundary and each specific design function will cover a small design area, then later all these smaller design areas within the design boundary will be grouped and merged interactively until the entire design is accomplished.

Top-down Design Approach	Bottom-up Design Approach
Splitting the entire design boundary into two or more smaller design areas in a hierarchy structure manner.	Identify all possible functions involved in the design boundary.
Design the function in each smaller design area.	Design each function, and then group these functions in area by area manner until they are merged into a prescriptive design.

Table 2- Top-Down versus Bottom-Up design approach.

The proposed algorithm is more suitable under the bottom-up design approach. This is because each algorithm activity can be designed and later developed to be an individual module. The support reason is that individual module is easier for module maintenance and enhancement in the long run. Each module at least must have one functionality to perform the required activity. Therefore, based on the proposed algorithm, the following table consists of the major modules.

Module Name	Description of Functionality
Error Detection	Extract software application log and identify error event based on error keywords stored in predefined error list.
Log Consolidation	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.
Log Integration	Integrate various log files obtained from different software application databases.
Analysis Report	1. Produce a report with complete information of current analysis activities and resolution action taken. 2. Provide an option to report past information of the analysis activities and resolution action taken.

Table 3- Each module and its associated function under bottom-up design approach.

As for the proposed functions in each of the proposed design module, the following table defines the proposed

functions in detail.

Module Name	Description of Functionality
Standard Analysis	<ol style="list-style-type: none"> 1. 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”. 2. Identify whether the newly reported software application error is first time occurrence or re-occurrence by cross-checking the knowledge base database of PAL. 3. Identify error log data and categorize the error log data for analysis under the defined software application error classification. 4. Identify the preferred resolution to the software application based on the outcome of analysis. 5. Apply resolution based on the predefined configuration of the PAL. 6. Store the analysis activity and resolution information in the knowledge base database of the PAL.
Complex Analysis	<ol style="list-style-type: none"> 1. 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. 2. Identify error log data and categorize the error log data for analysis under the defined software application error classification. 3. Analyze the selected log data against the software application error. 4. Should the analysis activities detect that more than one possible error is involved, then trigger Analytic hierarchy process (AHP) Error Evaluation Module. 5. Should the analysis activities detect that more than one possible resolution is involved, then trigger Analytic hierarchy process (AHP) Resolution Evaluation Module. 6. Deploy the preferred resolution option to fix the software application error under the predefined condition. 7. Store the analysis result and resolution action into a database which is associated to the PAL for future reference and knowledge base activities.

Table 4- Proposed function for the simple and complex analysis modules.

In addition, it should not be forgotten to design a module along with the proposed functions to categorize the category and type of all possible errors.

Module Name	Description of Functionality
Error Categorization	<ol style="list-style-type: none"> 1. Define error categories based on the common errors stated in the past research. 2. Using Regular Expression to identify the error from the log file and map the identified error with the most suitable error category.

Table 5 - Proposed function for error categorization module.

Finally, it is a need to have a module along with the proposed functions to store and to retrieve the past analysis activities and resolution actions.

Module Name	Description of Functionality
-------------	------------------------------

Knowledge Base	<ol style="list-style-type: none"> 1. Store the past analysis activities and resolution actions into table forms under the predefined database schema. 2. Allow to use Structured Query Language (SQL) statement to retrieve the past analysis activities and resolution actions from the tables.
----------------	---

Table 6- Proposed function for knowledge base module.

The crucial action to make decision which are required to handle the evaluation on shortlisting the software application error as well as the preferred resolution option. Both AHP actions should not be combined into a single module, and the support reason is to divide one major action in one module for better support and maintenance of the module.

Figure 1.0 - AHP module design.

Hence, the function design will be shown as the following Table.

Module Name	Description of Functionality
AHP Error Evaluation	<ol style="list-style-type: none"> 1. Allocate weight to each possible software application error based on Analytic hierarchy process (AHP). 2. Shortlist the software application error under the highest weight.
AHP Resolution Evaluation	<ol style="list-style-type: none"> 1. Allocate weight to each possible resolution option based on Analytic hierarchy process (AHP). 2. Shortlist the preferred resolution option under the highest weight.

Table 4 - Proposed functionality for each AHP module.

Note:

- For the module to handle “AHP to identify software application error to analyze”, in the proposed design we will rename it to “AHP Error Evaluation” module.
- For the module to handle “AHP to identify preferred resolution”, in the proposed design we rename it to “AHP Resolution Evaluation” module.

The entire proposed algorithm under the bottom-up design approach consists of error handler, log handler, analysis handler, and report handler, along with the knowledge-based interface section. In total there are six sections. Each section of handler will have its modules to carry out the respective major action, and each major action is one module. As for the knowledge-based section, this is separate module which is focusing on the interface programming with the knowledge-based database via the Application Programming Interface (API) provided by the database. Hence, the complete proposed design under the bottom-up design approach is shown as follows: -

Figure 2.0 - The complete bottom-up design approach for the proposed algorithm.

VII. CONCLUSION

Despite the speed of Information Technology (IT) evolution, software application is still a heavy dependence for most of the business operation today. The more software application becomes crucial to the business operation, the lesser toleration to the software application downtime. Whenever software application encounters error that causes downtime in the business operation, the root cause can be complex if the error is falling outside the software application boundary. To accurately identify the root cause is time consuming whenever more than one log file is required for the root cause analysis activity. Due to such complexity, it leads to the entire time duration on root cause analysis prolonged. This will increase the total time taken on restoring the software application service to the business operation. Therefore, to identify the root cause of software application error in a more accurate manner, and to shorten the duration of root cause analysis activity conducting on software application error, a proposed algorithm that is incorporated with Analytic Hierarchy Process (AHP) is proposed. The design of the proposed algorithm will be served as the new guideline to provide contribution to the log file analysis area.

ACKNOWLEDGMENT

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

REFERENCES

- [1] Hoo Meng W., Amalathas S.S. (2019). A New Approach Towards Developing a Prescriptive Analytical Logic Model for Software Application Error Analysis. In: Silhavy R., Silhavy P., Prokopova Z. (eds) Intelligent Systems Applications in Software Engineering. CoMeSySo 2019 2019. Advances in Intelligent Systems and Computing, vol 1046. Springer, Cham.

- [2] HM Wong and SS Amalathas. (2019). An Approach towards developing an Algorithm for Software Application Error Analysis. David Publishing Company. DOI:10.17265/2328-2185/2019.04.006.
- [3] Wong, H.M., Amalathas, S. and Zitkova, T. (2019). A Prescriptive Logic Model for Software Application Root Cause Analysis. European Journal of Electrical Engineering and Computer Science. 3, 5 (Oct. 2019). DOI:.
- [4] HM Wong, SS Amalathas and T Zitkova. (2020). A Business Operation Stability by Improving the Speed of Restoration on Software Application Service. David Publishing Company. DOI:10.17265/2328-2185/2020.01.008.
- [5] HM Wong, SS Amalathas and T Zitkova. (2019). An Approach Towards Designing A Prescriptive Analytical Logic Model for Software Application Root Cause Analysis. International Journal of Advanced Research and Publications (IJARP). <http://www.ijarp.org/published-research-papers/nov2019/An-Approach-Towards-Designing-A-Prescriptive-Analytical-Logic-Model-For-Software-Application-Root-Cause-Analysis.pdf>
- [6] Wong, H.M. and Amalathas, S. 2019. A New Analysis Approach Incorporated with Analytic Hierarchy Process for The Software Application in A Multiple Tiers Environment. European Journal of Electrical Engineering and Computer Science. 3, 6 (Dec. 2019). DOI:<https://doi.org/10.24018/ejece.2019.3.6.160>.
- [7] Edward Mercer, 2015. How Technology Affects Business Operations. OpposingViews.com. <http://science.opposingviews.com/technology-affects-business-operations-1659.html> .(Accessed on 28th November 2015)
- [8] [SuccessFactors, 2015. Using Technology to Increase Your Business Productivity. https://www.successfactors.com/en_us/lp/articles/using-technology-to-increase-your-business-productivity.html.(Accessed on 28th November 2015)
- [9] SANS Institute. 2001. Importance of Understanding Logs from an Information Security Standpoint. <https://www.sans.org/reading-room/whitepapers/logging/importance-understanding-logs-information-security-standpoint-200> (Accessed on 6th March 2017)
- [10] Andreas Grabner. 2012. Dynatrace LLC. Top Performance Mistakes when moving from Test to Production: Excessive Logging. <https://www.dynatrace.com/blog/top-performance-mistakes-when-moving-from-test-to-production-excessive-logging/> (Accessed on 6th March 2017)
- [11] David B. Stewart, PhD. 2012. Troubleshooting real-time software issues using a logic analyzer. InHand Electronics, Inc. <http://www.embedded.com/design/debug-and-optimization/4236800/Troubleshooting-real-time-software-issues-using-a-logic-analyzer>(Accessed on 12th December 2015)
- [12] Stephen G. Eick, Michael C. Nelson, and Jeffery D. Schmidt. 1994. Graphical Analysis of Computer Log Files. Communications of the ACM.<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.4832&rep=rep1&type=pdf>.(Accessed on 12th December 2015)
- [13] [Wendy W. Peng and Dolores R. 1993 Wallace. Software Error Analysis. NIST Special Publication. <http://www.geocities.ws/itopsmat/SoftwareErrorAnalysis.pdf>.(Accessed on 12th December 2015)
- [14] Felix Salfner and Steffen Tschirpke. 2015. Error Log Processing for Accurate Failure Prediction. https://www.usenix.org/legacy/event/wasl08/tech/full_papers/salfner/salfner_html/.(Accessed on 12th December 2015)
- [15] Jan Valdman. 2001. Log File Analysis. <https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2001/tr-2001-04.pdf>.(Accessed on 18th January 2017)
- [16] Júlia Murínová. 2015. Application Log Analysis. http://is.muni.cz/th/374567/fi_m/thesis_murinova.pdf.(Accessed on 18th January 2017)
- [17] Max Landauer, Markus Wurzenberger, Florian Skopik, Giuseppe Settanni and Peter Filzmoser. 2018. Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. ScienceDirect. [www.sciencedirect.com.https://www.sciencedirect.com/science/article/pii/S0167404818306333/pdf?md5=e114106f24719a22492bdc5df17f2742&pid=1-s2.0-S0167404818306333-main.pdf](https://www.sciencedirect.com/science/article/pii/S0167404818306333/pdf?md5=e114106f24719a22492bdc5df17f2742&pid=1-s2.0-S0167404818306333-main.pdf) (Accessed on 2nd April 2019)
- [18] Quora. 2016. What is the bottom up and top down approach? <https://www.quora.com/What-is-the-bottom-up-and-top-down-approach> (Accessed on 11th April 2018)
- [19] Wikipedia. 2018. Algorithm design. Wikimedia Foundation, Inc. https://en.wikipedia.org/wiki/Algorithm_design (Accessed on 11th April 2018)