A Prescriptive Logic Model for Software Application Root Cause Analysis

Hoo Meng Wong, Sagaya Sabestinal Amalathas, and Tatana Zitkova

Abstract—Software application stops functioning in the production environment can be due to the error either within the software application layer or any other factor outside the software application layer. With the complexity of an enterprise level of software application involving multiple tiers, it has log file at each tier. The root cause analysis activity will easily prolong as more than one log file required for the root cause analysis activity. This would increase the total time taken on restoring the software application service back to the users. To identify the root cause of software application error in a more accurate manner, and shorten the duration of root cause analysis activity conducting on software application error. A Prescriptive Analytical Logic Model (PAL) incorporates with Analytic Hierarchy Process (AHP) is proposed. The logic model contributes a new knowledge in the area of log file analysis to shorten the total time spent on root cause analysis activity. At the same time, it contributes the knowledge in AHP to close the knowledge gap.

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

I. INTRODUCTION

Business companies adopt Information Technology (IT) as a strategic enabler to sustain their business operations. When the 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.evolver.com (2014), and indeed software application downtime can cause the business operation ceased. Business company has option that it can run an IT production support team to provide support service to whichever business-as-usual (BAU) system running in the organization. Another option is that it can engage service provider to provide the same IT support service. Regardless whichever option is chosen, the time spent on conducting root cause analysis on software application error is crucial. Without accurately identifying the valid error, it creates impact to the service restoration to the software application.

As of the fact, identifying the root cause of the software application error is crucial before the resolution is decided and deployed into production environment. There are several required actions in the root cause analysis activity. These actions are:

- i. Collecting related information from different log files.
- ii. Selecting the related log events based on the time event when software application error occurred.

Most of the time, collecting input information 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.". Hence, it is crucial to look for an efficient method to reduce the prolonging time at the root cause analysis activity.

Generally, each software application has its built-in event logging ability. The purpose of this event logging records the information of what activity is carried out or even what incident is occurred at that exact time. This information includes appropriate debugging information, and later the same information can be analyzed for software application root cause analysis purpose. The concern raised to the required information logging is that how much logging information is accepted as sufficient for software application root cause analysis. In addition, what is the appropriate category for the logging event such as information, error, debug, and fatal 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.

- i. The first issue is that, the performance of software application is reduced by comparing with before and after extensive event logging option is enabled.
- ii. 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). As per the following figure 1.0, Operating System communicates between software application and assigned resources (such as CPU, Memory, Network, and Hard Disk) on the virtual machine. Software application has to interact with Operating System to obtain allocated server resources to handle software application processing. This is because software application has high dependency on server resources to carry out its execution. Without the server resources, software application cannot execute itself at the software application layer. Furthermore, software application requires to communicate to its database server

through the Local Area Network (LAN) for retrieving and updating software application data.

Fig. 1. Software Application is required sufficient server resources to execute all its functionality.

To identify the root cause of software application error, it can be prolonged under the following challenges.

- (a) Software application log is hard to be understood.
- (b) Error is occurred beyond software application layer, such as Operating System, Network, and Database layers. Software application log alone is insufficient to identify the actual error.
- (c) IT support personnel has insufficient knowledge and experience in performing analysis.
- (d) Root cause analysis is conducted manually, crucial information is overlooked.
- (e) Historical error events are taking even longer time to be located.

With all the mentioned concerns and scenarios, 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. Hence a proposed research is required for establishing a prescriptive analytical logic model. This proposed logic model incorporated the proposed algorithm to conduct the root cause analysis activity. 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.

II. SIGNIFICANT OF THE STUDY

There are two great points regarding to the significant of the study.

A. For the contribution to the business

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 business industry at the nationwide or the international business boundary. This is because with today's rapid business competitive world, time consuming on analysis and trouble-shooting activities is unacceptable, and it is continuous battle for the support team 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. On the other hand, business companies can still continue to utilize their existing software applications (without incur any additional operation budget) and at the same time to allow the companies to save the investment budget on spending the capital amount to replace all or partial of the software applications and re-training their users on using the new software applications. This

propose model not only can bring the above benefits to business industries but other industries which are using software application for their daily operation, they need to have the software application error fixed without further re-occurrence.

B. For the technical beneficial

Over the years there were various researches had been done at this area such as consolidate the logs or integrate the logs for analysis but there had been very attempts to propose a model to deliver a complete package for analyzing and fixing software application error which consists of the activities such as log integration, error analysis, decision making of preferred resolution, and automated on applying the error fix. There is a great potential in this research which brings contribution to business intelligent studies.

III. MOTIVATIONS

The main reasons to cause the motivation of the proposed research are shown as follows:

- i. 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.
- ii. Time consuming to read through the software application log file manually during root cause analysis activity.
- iii. 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.
- iv. 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:

- v. The business users face difficulty to continue their daily tasks as the software application is unavailable or malfunction.
- vi. 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.

IV. STATEMENT OF PROBLEM

The duration of root cause analysis on software application error carries crucial impact to the service restoration.

V. RESEARCH OBJECTIVE

To mitigate prolonging on conducting root cause analysis activity.

VI. LITERATURE REVIEW

In the past research, Stewart, D (2012) is focusing on debugging real-time software application error using logic analyzer debug macros, whereby Eick, S, Nelson, M, and Schmidt, J (1994) they are focusing on presenting the error logs in a readable manner. Moreover, Wendy, P and Dolores, R (1993) suggested to focus on error detection in software application at the time of software development and maintenance. However, Salfner, F and Tschirpke, S (2015) are focusing on analyzing error logs by applying the proposed algorithms in order to predict future failure. Their software application error analysis approaches focus on software application error log obtained from software application database. Some literature suggested that the software application error analysis would be better if it is built in during the software development process. This approach is still within the same application development boundary without factoring in any other area of concerns. It can cause the software application failure. In another way of explanation, whenever hardware CPU and memory utilization is running high, or even storage disk space is running low, software application logging may not be accurate anymore. Hence, the root cause analysis would not be accurate to identify the real issue to understand the main reason to cause the software application failure. Murínová, 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. 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. From a different point of view, this approach is good because it can be adopted into the proposed model to detect the software application error. Hence, this is a great potential to propose a logic model to research a new approach towards developing an algorithm for software application error analysis. At the same time to contribute new knowledge in the area of software application root cause analysis using AHP. By comparing the above secondary data with the proposed logic model, it can be noticed that the focus boundary on software application analysis. The technique to identify the root cause of software application is different. They focus on software application boundary whereby the proposed 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.

Although there are many market products available for review, as for the literature review, we choose the most popular three market products for features comparison. The product information in the following tables are referred by Solarwinds (2018), Logstash (2018) and AppDynamics (2018).

TABLE I: MARKET PRODUCT COMPARISON FOR MONITORING AND ANALYZING LOGS

Product	Solarwinds Log Analyzer
Features	Solarwinds (2018) claimed the features that: - "1. Allows collecting and centralizing event logs from multiple software applications in a single location. 2. Provides several pre-defined filters organized by categories. 3. Drills down to a specific event log to get a detailed listing of the source/destination machines, IP address, ports, and etc. 4. Correlates all event logs from network devices, applications, servers, storage, security appliances, and other systems in real-time with Log & Event Manager which includes hundreds of built-in event correlation rules."
Product	Logstash
Features	Logstash (2018) claimed the features that: - "1. Ingests event logs from a multitude of sources simultaneously, all in continuous, streaming fashion, and then sends it to a decided location. 2. Supports a variety of inputs that pull in event logs from a multitude of common sources. 3. Provides the feature to filter and to parse each event log and followed by to identify named fields to build structure, and to transform them to converge on a common format for easier and accelerated analysis. 4. Has a variety of outputs that allows user to route event logs where user wants. 5. Modules orchestrate a turnkey ingest-to-visualize experience with popular data sources."
Product	Cisco AppDynamics
Features	AppDynamics (2018) claimed the features that: - "1. Monitors and manages production infrastructure - servers, databases, and network by automatically discovers, maps, and visualizes each application service and infrastructure dependency. 2. Provides deep diagnostic capabilities to identify root-cause down to the individual line of code. 3. Tracks every Business Transaction across a user session by automatically capture errors, crashes, network requests, page load details, and other metrics for an entire user session. 4. Allows to stream contextual insights for every transaction, customer segment, code release, or complete business service. 5. Allows for business health baselines, anomaly detection, and alerts are automated and immediately actionable, so support teams can proactively manage risk and opportunity of the software application. 6. Provides monitoring to even get insights into machine and infrastructure-level metrics like disk I/O, throughput, CPU utilization, and memory usage with our Server Visibility Module. 7. Provides agentless database collector which no installation or configuration on the database server. 8. Provides auto-discover load balancers and visualize network performance metrics in the context of application performance."

On the other hand, Kanojia (2018) proposed a product which is named Micro Focus Operations Bridge, this product will sit on top of all the existing monitoring tools, and to consolidate events and correlate them in real time. This product introduces an Analytic Engine which can perform predictive analytic based on the data that has been collected over a period (historical data from log files and/or events) and claims that it can predict issues before the issue occurs.

From the above table, it is easily to understand from the product feature comparison in term of the common features. The common features are such as collecting all possible logs into a central location, and for each type of collected log, it gets massaged and presented into a graphical view under a widget. After all, by putting all the widgets together into a so-called dashboard, it creates a bird-view to oversee all software application status in a graphical visualization for better viewing. In addition, the market product allows users to configure predefined resolution action for the standard software application error. However, the decision making on identifying the valid software and predefined resolution steps are still on human.

VII. RESEARCH PROPOSED

With a server (regardless it is physical or virtual) box, you have multiple layers. Most common logs that are required is:

- System monitoring log for server resources such as CPU, Memory and Hard Disk usage.
- Network monitoring log for the server such as network communication within the Local Area Network (LAN).
- Operating System event log for server.

The illustration of multiple layers for a server can be referred to Figure 2.0. On the other hand, the software application especially for those that are rated at enterprise level category, it involves multiple tiers such as Web Client tier, Web Container tier, Application Container tier, and Database tier. Beside the Web Client tier, each tier has its own log file. Certain tiers are even having multiple log files based on the software application design. The proposed research is to develop an algorithm towards the PAL for analyzing software application error. The analysis is based on the logs retrieved from various software applications by referring to the following figure. With this proposed logic model, it is aiming to resolve the statement of problem, which is prolonging the time consumption on identifying the root cause of the software application error would increase the total time taken on restoring the software application service to the users.

Fig. 2. The proposed software application analysis algorithm will analyze across multiple databases.

A. Proposed Research Scope

The proposed scope of this research is to define the algorithm. The algorithm consists of simple and complex analysis inside the PAL for software application error analysis. Therefore, by having the proposed logic model in the production environment, the PAL is required to react to software application error when the error is detected in the software application log file. With this PAL, it is also required to retrieve other related log files through various software applications shown as Figure 2.0. The proposed algorithm mainly consists of two analysis areas, which are simple and complex analysis to form a prescriptive analytical logic.

B. Proposed Simple Analysis

For simple analysis area, it is required to build the predefined logic to handle the common software application error, whereby this model is to guide the system builder on answering a set of predefined questions on common

software application errors and carry out the predefined activities only react to these common software application error, for example, restarting the software application process if it is stopped.

C. Proposed Complex Analysis

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 data will serve 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 PAL is able to 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 PAL that handles the common software application incidents. We call this logic 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.

The software application errors which have no uniform pattern or feature, for this type of software application errors. The percentage of human involvement is high. This is because the person who handles the incident requires to obtain the software application log files and to search any similar error logged in the past. We call these files and records as 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. Then, 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. Of course, this logic has the ability to handle unpredictable software application errors by performing simulated analysis activities comparing with human, we call this logic as complex analysis.

Whenever the complex analysis is triggered. It will pull the related logs based on specific time frame (duration) before and during the software application failure from various application logs. These logs are:

- Software Application logs is the beginning to trigger the root cause analysis,
- Configuration management logs is for understanding any recent applied software application patches or Operating System patches,
- Performing and capacity monitoring logs is for identifying any hardware resources running insufficient, and
- Production support ticketing tool logs is for cross-checking any related issue recently occurred under the predefined database scheme.

These above logs as input information will be utilized crucially for root cause analysis to resolve the software application issue. Indeed, 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). Base 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.

D. Proposed Algorithm and Methodology

The proposed algorithm under the PAL includes the following crucial activities:-

TABLE II: PROPOSED PROCESS ACTIVITY UNDER PROPOSED ALGORITHM

No.	AHP	Proposed Algorithm
1		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.
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 PAL.
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.
10		Store the analysis result and resolution action into a database which is associated to the PAL for future reference and knowledge base activities.

The methodology of the PAL is derived from the proposed algorithm. The figure is shown as follows:

Fig. 3. The methodology of Prescriptive Analytical Logic Model.

The PAL algorithm design consists of simple and complex analysis design, proposed analytic hierarchy process design and knowledge based database design. From the simple and complex analysis design, it further derives to top-down and bottom up design. All of these designs will form PAL algorithm design as the complete model design. The PAL consists of two configuration files that are the brain of the PAL to identify the known errors and the action of the preferred resolution.

VIII. ANALYTIC HIERARCHY PROCESS (AHP)

AHP was developed by Thomas L. Saaty stated in Wikipedia (2015). The proposed algorithm of the prescriptive analytical logic can adopt the decision making process of AHP to decide the valid software application. Then, followed by using the same decision making process to shortlist the best resolution. This is because by referring to R. W. Saaty (1987), the three principles which are the decomposition principle, the comparative judgments, and the synthesizing priorities. In addition, Vaidya and Kumar (2004) had provided the discussion on how to apply AHP under the analytic hierarchy process. This helps to understand how the proposed algorithm applies in the scenario after the valid software application error, and preferred resolution are identified during the root cause analysis activity.

Based on those past researches or similar area of researches in the area of software application log file analysis. The AHP was not introduced or applied for handling the activities such as shortlisting the valid software and shortlisting the preferred resolution. Since AHP is a decision making process as a tool, it 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, which are at the row 4, 5, 7 and 8 in the following Table.

By referring to the proposed activities in table 2.0, that are activity number 4, 5, 7 and 8 (with the Note *) 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: -

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.

The activity number 5 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.

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.

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.

With the PAL, it carries high potential on knowledge contribution as the new logic model has the algorithm to be able to identify the root cause of the software application error more accurately under the AHP processing approach, and deliver the outcome to shorten the analysis duration of conducting root cause analysis activity during the software application downtime.

IX. PROPOSED PREREQUISITES OF PAL

A. Multiple Tiers Environment

In this proposed research, it is required to run on a multiple-tier environment which involves Client tier, Web Container tier, Application Container tier, and Database tier. The proposed algorithm will be implemented into a software plug-in component sitting at the logic tier to integrate all the required log event as input data from various software applications and store in a separate location for information retrieval later. A new standard database schema may be required on this research, and it will be applied across all the required databases of the involved software applications. This will help on retrieving log events as data from various databases when the root cause analysis activity is triggered at the logic tier whenever software application error is detected in the main software application log file. Otherwise, by granting necessary permission to retrieve log events as data from the involved software applications should be sufficient.

B. Network Time Protocol (NTP) Server

Whenever a software application involves in a multiple-tier environment, it is good to standardize the server time across all the involved servers. The supporting reason is that the proposed PAL 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."

C. Technologies

The proposed prototype of the software plug-in component will be coded using Java Programming as this programming language is platform independent. At the platform, it required to install Java Runtime Environment (JRE) and configure the Java home path and memory allocation for the JRE.

X. CONCLUSION

There are many software application error analysis tools available in the IT industry today. However, these tools mainly analyze based on the collected software application error logs which is reasonable if the hardware resources of the system are under a healthy stage. On the other hand, software application comes along with its event logging feature. This feature is mostly embedded into the software application and its design is based on the situation that software application has adequate server resources. In the scenario whenever server resources are running low, software application will face issue during its execution. Other issues can even come from network connectivity, no response from database server and etc that is beyond the software application layer. Hence, if the root cause analysis is solely depending on software application error log file alone, it is definitely insufficient and the result cannot be accurate. By improving the software root cause analysis technique into a more human-like intelligent, a proposed logic model is required. Therefore, the proposed logic model that incorporates with AHP is a great potential to fill up the knowledge gap in AHP.

ACKNOWLEDGMENT

I would like to thank to Tatana Zitkova, for providing encouragement on my proposed research and PhD studies.

REFERENCES

- Andrews, James H. (1998). *Theory and practice of log file analysis*. Dept. of Computer Science, University of Western Ontario. London, Ontario, Canada. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.292&rep=rep1&type=pdf> (Accessed on 28th November 2015)
- Andrews, James H. (2017). *A Framework for Log File Analysis*. Dept. of Computer Science, University of Western Ontario. London, Ontario, Canada. Retrieved from <https://pdfs.semanticscholar.org/9d1a/97988d8b41354cd4bf85ace96648d1684555.pdf> (Accessed on 28th November 2015)
- AppDynamics. 2018. *Transform the way modern business is built and run*. AppDynamics, Cisco. <https://www.appdynamics.com/> (Accessed on 2nd July 2018)
- Atwood, Jeff. *The Problem With Logging*. (2008). Retrieved from <https://blog.codinghorror.com/the-problem-with-logging/> (Accessed on 28th November 2015)
- Bloom, Alex. (2014). *The Cost of Application Downtime is Priceless*. StatusCast. Retrieved from <http://www.statuscast.com/cost-application-downtime-prices/> (Accessed on 28th November 2015)
- Da-Yong Chang. 1996. *Applications of the extent analysis method on fuzzy AHP*. ELSEVIER. Retrieved from http://ac.els-cdn.com/0377221795003002/1-s2.0-0377221795003002-main.pdf?_tid=e251d09a-7732-11e7-b173-00000aacb35d&acdnat=1501644789_e6ae23503d2e2f883d2413693f63275a (Accessed on 12th September 2018)

- Dr. Benito L. Teehankee and De La Salle University. 2009. *The Analytic Hierarchy Process: Capturing Quantitative and Qualitative Criteria for Balanced Decision-Making*. ResearchGate. Retrieved from https://www.researchgate.net/publication/256009323_The_Analytic_Hierarchy_Process_Capturing_Quantitative_and_Qualitative_Criteria_for_Balanced_Decision-Making (Accessed on 12th September 2018)
- Eick, Stephen G., Nelson, Michael C., and Schmidt, Jeffery D. (1994). *Graphical Analysis of Computer Log Files*. Communications of the ACM. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.4832&rep=rep1&type=pdf> (Accessed on 12th December 2015)
- Error Handling, Auditing and Logging. (2015). Retrieved from https://www.owasp.org/index.php/Error_Handling_Auditing_and_Logging (Accessed on 28th November 2015)
- Evangelos Triantaphyllou and Stuart H. Mann. 1995. *Using the Analytic Hierarchy Process for Decision Making in Engineering Applications: Some Challenges*. Retrieved from http://s3.amazonaws.com/academia.edu.documents/31754775/AHPap1s1.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1498819440&Signature=iIMAp3tvmRbBTX8k6rUvRIorb7o%3D&response-content-disposition=inline%3B%20filename%3DUSING_THE_ANALYTIC_HIERARCHY_PROCESS_FOR.pdf (Accessed on 12th September 2018)
- Hasan, Mohamad K. (2010). *A Framework for Intelligent Decision Support System for Traffic Congestion Management System*. Kuwait City, Kuwait. Retrieved from <http://www.umsl.edu/divisions/business/pdfs/Mohamad%20Hasan%20IDSS.pdf> (Accessed on 13th November 2015)
- 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. Advances in Intelligent Systems and Computing, vol 1046. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-30329-7_24. https://link.springer.com/chapter/10.1007/978-3-030-30329-7_24
- Hoo Meng Wong and Sagaya Sabestinal Amalathas. (2019). *An Approach towards developing an Algorithm for Software Application Error Analysis*. David Publishing Company. DOI:10.17265/2328-2185/2019.04.006. <http://www.davidpublisher.com/index.php/Home/Article/index?id=39805.html>
- Horvath, Kristof. (2015). *Using Root Cause Analysis to Drive Process Improvement*. Intland Software. Retrieved from <http://intland.com/blog/safety-engineering/using-root-cause-analysis-to-drive-process-improvement/> (Accessed on 28th November 2015)
- Jan Valdman. 2001. *Log File Analysis*. Retrieved from <https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2001/tr-2001-04.pdf> (Accessed on 18th January 2017)
- Klass, Lina. 2018. *Machine Learning - Definition and application examples*. MM International. https://www.spotlightmetal.com/machine-learning--definition-and-application-examples-a-746226/?cmp=go-aw-art-trf-SLM_DSA-20180820&gclid=Cj0KCQjwYLoBRCxARIsAEf16-vBbITDXGp3-EuK0AsxQu90ckOqGbTyXEvjQBWSA3D3MHB51TxDhQUaAIPDEALw_wcB
- Krejčí, Jana and Stoklasa, Jan. 2018. Aggregation in the analytic hierarchy process: Why weighted geometric mean should be used instead of weighted arithmetic mean. Expert Systems With Applications 114 (2018) 97–106 Contents. Elsevier. Retrieved from <https://www.sciencedirect.com.ezproxy.taylors.edu.my/science/article/pii/S0957417418303981> (Accessed on 12th March 2019)
- Loggly Inc. (2017). Retrieved from <https://www.loggly.com/blog/measuring-the-impact-of-logging-on-your-application/> (Accessed on 28th November 2015)
- Logstash. 2018. *Centralize, Transform & Stash Your Data*. Elasticsearch. <https://www.elastic.co/products/logstash> (Accessed on 2nd July 2018)
- Management Logic. (2012). Root-Cause Analysis. Retrieved from <http://www.management-logic.com/toolbox/sales/Root-Cause%20Analysis/Index.html> (Accessed on 28th November 2015)
- Margulius, David. (2006). *Tech jobs take stress to whole new levels*. InfoWorld, Inc. Retrieved from <http://www.infoworld.com/article/2655363/technology-business/tech-jobs-take-stress-to-whole-new-levels.html> (Accessed on 13th November 2015)
- Masterclock. (2016). *Network Time Synchronization - Why You Need An NTP Server*. Masterclock, Inc. Retrieved from [https://www.masterclock.com/company/masterclock-inc-](https://www.masterclock.com/company/masterclock-inc-blog/network-time-synchronization-why-you-need-an-ntp-server)
- blog/network-time-synchronization-why-you-need-an-ntp-server (Accessed on 4th September 2018)
- 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/pdfft?md5=e114106f24719a22492bdc5df17f2742&pid=1-s2.0-S0167404818306333-main.pdf). Retrieved from <https://www.sciencedirect.com/science/article/pii/S0167404818306333/pdfft?md5=e114106f24719a22492bdc5df17f2742&pid=1-s2.0-S0167404818306333-main.pdf> (Accessed on 2nd April 2019)
- Mercer, Edward. (2015). *How Technology Affects Business Operations*. OpposingViews.com. Retrieved from <http://science.opposingviews.com/technology-affects-business-operations-1659.html> (Accessed on 28th November 2015)
- Murínová, Júlia. (2015). *Application Log Analysis*. Retrieved from http://is.muni.cz/th/374567/fi_m/thesis_murinoval.pdf (Accessed on 28th November 2015)
- Nathasit Gerd Sri and Dundar F. Kocaoglu. 2007. *Applying the Analytic Hierarchy Process (AHP) to build a strategic framework for technology roadmapping*. ScienceDirect. ELSEVIER. Retrieved from http://ac.els-cdn.com/S0895717707001069/1-s2.0-S0895717707001069-main.pdf?_tid=e0165298-5d78-11e7-9000-00000aabb0f6b&acdnat=1498816119_34b7b92d988eca0c318b8d23810e293a (Accessed on 21st February 2019)
- Omkarprasad S. Vaidya and Sushil Kumar, 2004. *Analytic hierarchy process: An overview of applications*. Department of Mechanical Engineering, Army Institute of Technology, Pune 411 015, India. National Institute of Industrial Engineering (NITIE), Vihar Lake, Mumbai 400 087, India. Retrieved from http://ac.els-cdn.com/S0377221704003054/1-s2.0-S0377221704003054-main.pdf?_tid=39850adc-5d78-11e7-b5bf-00000aabb0f6b&acdnat=1498815840_e0c9a10c99c46ad30db8da4ef17e817b (Accessed on 12th September 2018)
- Ornelas, Linda V. (2003). *Important of Event Logging*. SANS Institute. Retrieved from <https://www.giac.org/paper/gsec/3297/importance-event-logging/105437> (Accessed on 28th November 2015)
- Panda, Archanaa. (2011). *High Performance and Smarter Logging*. DZone. Retrieved from <https://dzone.com/articles/high-performance-and-smarter> (Accessed on 12th December 2015)
- R. W. Saaty. (1987). *The Analytic Hierarchy Process – What it is and how it is used*. Pergamon Journals Ltd. Retrieved from <https://core.ac.uk/download/pdf/82000104.pdf> (Accessed on 9th April 2018)
- Razvan Serbu, Bogdan Marza and Sorin Borza. 2016. *A Spatial Analytic Hierarchy Process for Identification of Water Pollution with GIS Software in an Eco-Economy Environment*. MDPI. Retrieved from https://www.google.com/url?sa=t&rcit=j&q=&esrc=s&source=web&cd=8&ved=0ahUKewjTk9bz-cLbAhUFupQKHcXaCmAQFghkMAc&url=http%3A%2F%2Fwww.mdpi.com%2F2071-1050%2F8%2F11%2F1208%2Fpdf&usq=AOvVaw3k7wSGKCoC_Exqr0vJKHK7 (Accessed on 12th September 2018)
- REDDIT. (2015). Experienced Dev's: *How much of your time do you spend troubleshooting?* Retrieved from https://www.reddit.com/r/webdev/comments/3slccc/experienced_devs_how_much_of_your_time_do_you/ (Accessed on 14th November 2015)
- Robert Rinnan. (2005). Benefits of Centralized Log File Correlation. Gjøvik University College. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.8787&rep=rep1&type=pdf> (Accessed on 16th March 2017)
- Saaty, R.W. (1987). The Analytic Hierarchy Process—What It Is and How It Is Used. Mathematical Modelling, 9, 161-176. Retrieved from [http://dx.doi.org/10.1016/0270-0255\(87\)90473-8](http://dx.doi.org/10.1016/0270-0255(87)90473-8) (Accessed on 12th September 2018)
- Salfner, Felix and Tschirpke, Steffen. (2015). *Error Log Processing for Accurate Failure Prediction*. www.usenix.org. Retrieved from https://www.usenix.org/legacy/event/was108/tech/full_papers/salfner/salfner_html/ (Accessed on 12th December 2015)
- Solarwinds. 2018. *Automate log analysis, improve IT operations, mitigate threats, and help ensure compliance*. SolarWinds Worldwide, LLC. https://www.solarwinds.com/topics/event-log-analyzer-tool/step_benefit_1 (Accessed on 2nd July 2018)
- StackOverflow. (2015). *How much time do you spend in production troubleshooting?* Retrieved from <http://stackoverflow.com/questions/1425069/how-much-time-do-you-spend-in-production-troubleshooting> (Accessed on 14th November 2015)
- Stewart, David B. PhD. (2012). *Troubleshooting real-time software issues using a logic analyzer*. InHand Electronics, Inc. Retrieved from <http://www.embedded.com/design/debug-and->

- optimization/4236800/Troubleshooting-real-time-software-issues-using-a-logic-analyzer (Accessed on 12th December 2015)
- SuccessFactors, (2015). *Using Technology to Increase Your Business Productivity*. Retrieved from https://www.successfactors.com/en_us/lp/articles/using-technology-to-increase-your-business-productivity.html (Accessed on 28th November 2015)
- Thomas L. Saaty. 2008. *Decision making with the analytic hierarchy process*. Int. J. Services Sciences, Vol. 1, No. 1. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.409.3124&rep=rep1&type=pdf> (Accessed on 12th September 2018)
- W. Peng, Wendy and R. Dolores. (1993). Wallace. *Software Error Analysis*. NIST Special Publication. Retrieved from <http://www.geocities.ws/itopsmat/SoftwareErrorAnalysis.pdf> (Accessed on 12th December 2015)
- Wikimedia Foundation Inc. (2017b). *Logfile*. Retrieved from <https://en.wikipedia.org/wiki/Logfile> (Accessed on 28th May 2017)
- Wikimedia Foundation, Inc. (2017a). *Application Server*. Retrieved from https://en.wikipedia.org/wiki/Application_server (Accessed on 28th June 2017)
- Wikipedia, (2015). *Analytic hierarchy process*. Wikimedia Foundation, Inc. Retrieved from https://en.wikipedia.org/wiki/Analytic_hierarchy_process (Accessed on 13th November 2015)
- www.evolver.com, Labels: Data Center, Downtime, (2014). *Downtime, Outages and Failures - Understanding Their True Costs*. Evolver Software. Retrieved from <http://www.evolver.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html> (Accessed on 13th November 2015)

Hoo Meng Wong is a PhD student in Taylor's University, Malaysia. He holds two bachelor degrees and a master degree. The bachelor degrees are Bachelor of Science in Computer Science which is awarded by La Trobe University, Melbourne, Australia. The another bachelor degree is Bachelor of Business (Hons) in Banking and Finance which is awarded by Wawasan Open University, Malaysia. For the master degree, it is Master of Science in Technology Management which is awarded by Staffordshire University, United Kingdom. Hoo Meng is new to publishing journal paper although he had started the PhD studies back in 2016. At that time he was with UNITAR International University, Malaysia. The reason for him transferred to Taylor's University because his supervisor had decided to join Taylor's University as one of the outstanding universities in Malaysia. Hoo Meng is currently conducting a research and preparing a thesis with the title Prescriptive Analytical Logic Model on Software Application Error Analysis. This topic of research is focusing on decision making performed on two activities, that are first activity is to identify the valid software application error from the root cause analysis activity, and second activity is to identify the preferred resolution based on the involvement of software development life cycle (SDLC) on the fix. Indeed, the research is to proposed a new algorithm for the Prescriptive Analytical Logic Model. The decision making process is incorporated with the Analytic Hierarchy Process (AHP).