Management Studies, Jan.-Feb. 2020, Vol. 8, No. 1, 60-68
doi: 10.17265/2328-2185/2020.01.008

A Business Operation Stability by Improving the Speed of Restoration on Software Application Service

Hoo Meng Wong, Sagaya Sabestinal Amalathas
Taylor's University, Subang Jaya, Selangor, Malaysia
Tatana Zitkova

Software application is still a heavy dependence for most of the business operation today. Whenever software application encounters error that causes downtime in the production environment, the root cause of the error can be either within the software application layer or any other factor outside the software application layer. To accurately identify the root cause is difficult whenever more than one log file is required for the root cause analysis activity. Due to such complexity, it leads to the entire duration on the root cause analysis activity became prolong. This will increase the total time taken on restoring the software application service back to the users. In order 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 incorporates with Analytic Hierarchy Process (AHP) is proposed. The proposed Logic Model along with the algorithm will contribute a new knowledge in the area of log file analysis to shorten the total time spent on root cause analysis activity.

Keywords: analytic hierarchy process, business continuity, business operation stability, error log analysis

Introduction

The computing technology electronic data-processing era started in 1950s and used widely during the 1960s and early 1970s as mentioned in Wikimedia Foundation Inc. (2016). Even there were several eras had been passed through, the evolution is continuing today. Furthermore, despite how fast the speed of progress in the information technology (IT) evolution and innovation, software application is still playing an important role to support today's company business operation in many ways. However, there is possibility that the software application may encounter error during its execution in production environment. This is due to software application defect having possibility that it can still be escaped from the testing stage during software development cycle. In order to understand the actual software application error or any other related error events further, these error events require to be logged into a log file for reference or conducting analysis activities later. According to Wikimedia Foundation Inc. (2017) which it had clearly mentioned that software application is still requiring log file mainly for several important activities, such as recording event information for software application installation, monitoring activities during software application execution, and capturing alerts, error,

Hoo Meng Wong, B.Sc., BBuss (Hons), M.Sc., Ph.D. Student, Taylor's University, Subang Jaya, Selangor, Malaysia.
Sagaya Sabestinal Amalathas, Dr., Taylor's University, Subang Jaya, Selangor, Malaysia.
Tatana Zitkova.
Correspondence concerning this article should be addressed to Hoo Meng Wong, 34 Jalan Setia Indah U13/11U, Setia Indah 11, Setia Alam, 40170 Shah Alam, Selangor, Malaysia.

and warning. Of course, later this event information found in the log file is useful for analysis or debugging. 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.

(1) The first issue is that the performance of software application is reduced by comparing with before and after extensive event logging option is enabled.

(2) 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 Gifford (2015) and Panda (2011).

Significant of the Study

Business organizations utilize the software application as the IT enabler to strengthen its business operation. Therefore 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. With the proposed research, there are two great points regarding to the significant of the study. Firstly, it is mainly for the contribution to the business. This is because by lowering down the risk of the software application error and improving 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. Secondly, it is 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.

Motivations

For an enterprise level of software application, the setup is complex as it is across multiple tiers as well as middleware is adopted to facilitate certain business functions. Therefore, with the software application error log file alone, it may not be adequate for analysis to identify the root cause of software application error. Most importantly, if the cause is outside the software application boundary, such as: (i) Any recent operating system patching activity had been conducted or (ii) the hardware configuration change request had been conducted recently. Both can possibly cause the software application running unstable. In such scenario, the software application errors did not specifically indicate the root cause. 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. Therefore, the possibility of the same issue will be re-occurred is high. Based on the information contributed by Rinnan (2005), in this circumstance, software application support team requires to cross reference on other log files, such as system monitoring log to identify the server resource limitation, and even further cross reference configuration management details to identify any recent change activity conducted to cause the software application running into error occurred.

Statement of Problem

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

Research Objective

The objective of the research is to restore the software application service back to the users on the business operation.

The sub-objectives are:

1. To investigate the published log file analysis methods and techniques.
2. To identify any published or suggested steps or techniques on root cause analysis of software application can be adopted.
3. To determine the common errors of the software application, Operating System, Network, database, and hardware.
4. To evaluate the Analytic Hierarchy Process (AHP) techniques from the published AHP documents.
5. To validate the overall time taken in the duration of root cause analysis activity.
6. To validate the accuracy on identifying the root cause whenever error is occurred.

Literature Review

Whenever software application error arises during its execution, business users face difficulty to continue their daily tasks to process business transactions. The software application support team is under pressure on how fast to get the software application error rectified and resolved. More importantly is to resume the software application service back to business users soonest as they can. Especially for those business operations is heavily relying on software application to handle daily business transactions, the software application support team is being pressured by the management without any mercy if the error persists. According to the

information contributed by both Mercer (2015) and Success Factors (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. The fact is obvious that when a software application is adopted in the business operation, it has the possibility on up-time and down-time during execution. This reality is no doubt on day-to-day business operation. It implies that the challenges and tension are always on the software application support team's shoulder to resolve software application issues as soon as possible during it is at down-time. During software application downtime, it would lead to lose of money in business operation as per the article explained by Bloom (2014). With such fact, software application error is required to identify the root cause first before a resolution can be applied. Therefore, several required actions must be conducted, such as collecting related information and performing root cause analysis, which are crucial actions. However, it is time-consuming to read through the software application log file manually during root cause analysis activity. This is supported by Management Logic (2012) as it 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". Furthermore, Horvath (2015) had 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". Indeed, the software application error is taking longer time to read, to understand, and to analyze it before the root cause can be identified. There are comments obtained from the Internet regarding to time consuming on software application trouble-shooting, which are REDDIT (2015) and StackOverflow (2015).

On the other hand, running a software application support team would always be uneasy as the Operating Expense (OPEX) in a financial year is not cheap. This is because OPEX includes many cost items spending within a financial year, and what the most concerning cost item here is the cost of software application support team members who are paid to support the application(s) crucial to the business operation. This leads to high expectation to the software application support team to resolve the software application issues during the application downtime. Time to time the head of software application support team is always being questioned by the company management whether there is any method to reduce the OPEX and at the same time company management expects the software application support team to provide either the same level or even a better level of IT service to ensure the software application up-time at an agreeable percentage. This high expectation from the company's management would create stressful and tension to the software application support team. With this point, it is supported by Margulius (2006) as IT jobs become the most stressful job among other jobs, such as engineering, sales, finance, HR, and others.

Therefore, as for the contribution, there are two key points:

- Literature review indicates that there is gap in the log file analysis field using AHP approach applied on software application error.
- Since there is no published article showing that AHP had been applied to software application error analysis, the proposed research is to fill up the knowledge gap.

Research Proposed

The objective of this proposed research is to restore the software application service back to the users on the business operation. Therefore, by using a simple mathematics approach, it helps to evaluate the statement of problem.

Table 1

The Variables Along With the Respective Description Given

Variable	Description
SLA	SLA = Assume that Service Level Agreement (SLA) only allows eight hours in total for critical issue.
W	Let W = Total time taken to resume the software application service.
X	Let X = Root cause analysis activity duration, and assign it to have an initial value, two hours.
Y	Let Y = Development of fix, Testing the fix, and Deployment and regression test. Y has to be a constant value, six hours. This is because the proposed resolution is not fixing here.
p	Let p = Prescriptive Analytical Logic Model

Now, if W = X + Y, then W = eight hours.

X may not be always two hours as it is depending on how fast the root cause is identified manually, and how accurate to identify the actual error manually. In the worse case, X can be more than two hours due to the knowledge and experience of the engineer to conduct the analysis activity manually. That will cause the SLA breached.

X > 2 ← SLA Breached.

Hence, if X can be shorten or kept below two hours, this will impact the hours in total for W, which means p as the Prescriptive Analytical Logic Model to reduce the time spent on root cause analysis activity and keep X to be two or lesser.

X <= 2 if X – p, then

W = (X – p) + Y, W < SLA ← with that, W can be kept below eight hours.

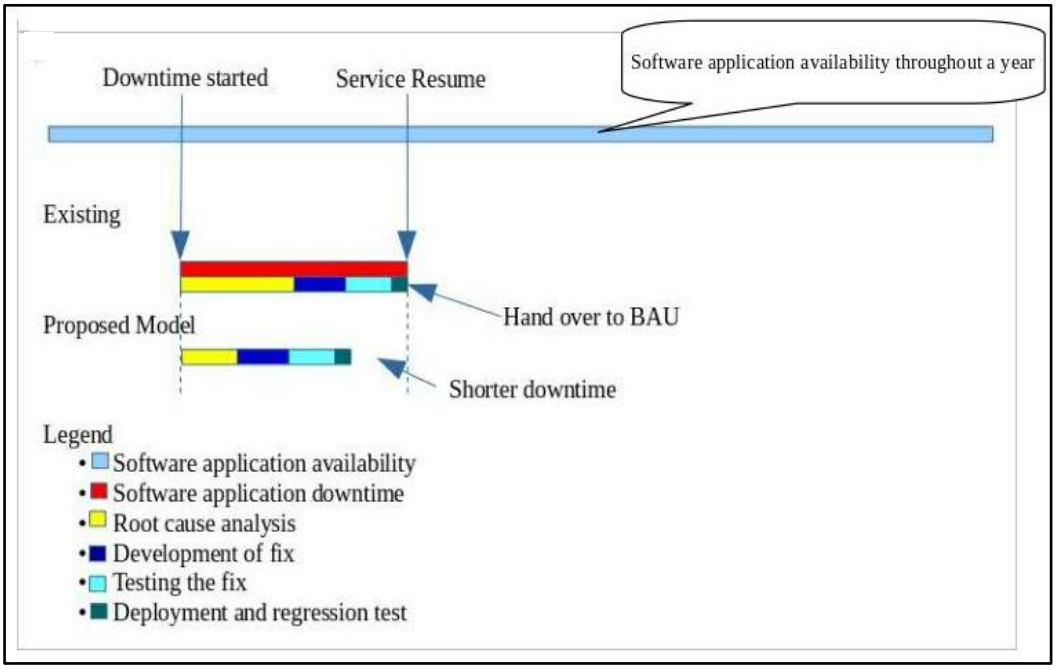

Figure 1. The illustration of time on software application downtime.

In the overall total time taken, it is from the point of time software application encounters error, until the point of time software application error is resolved. The first activity is conducting root cause analysis to identify the actual software application error. There are activities will be conducted after the root course analysis activity. These activities are in the duration of developing, testing, and deploying the resolution. All

these activities are all in a row (sequence) and each activity has its dependency; that is depending the completion of previous activity. Which means how fast the root cause of the software application error can be identified at the duration of conducting root cause analysis activity, and whenever the root cause is identified only then the next activity which is the activity of developing the resolution can be activated.

With a simple mathematics approach, the problem is evaluated and it is a real problem, and therefore the entire proposed logic model is designed to shorten the time spent on root cause analysis activity duration for the problem. The proposed model is the outcome of the objective to achieve, and 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. Please note that under the proposed model, this software application plug-in is not only accessing the software application log file, but at the same time, it also accessing other log files from various databases as input information for software application analysis activities shown as the Figure 2.

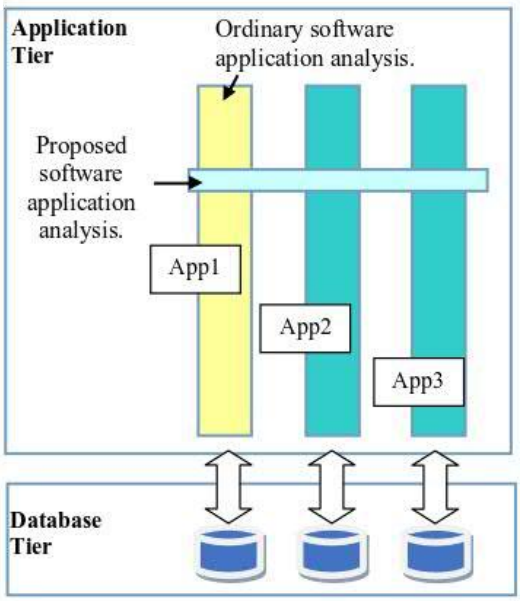

Figure 2. The proposed software application analysis algorithm analyses multiple software application log files.

Under the proposed model for the prescriptive analytical logic, it mainly consists of two logic components for software application analysis activities, which are designed for simple and complex analysis respectively. Each logic area consists of the proposed algorithm to perform the specific software application activities shown as Figure 3. It leads the scope of this research would define the propose of simple and complex analysis to form a prescriptive analytical logic to conduct trouble-shooting activities automatically when react to software application error. Thus, the reliable information as input information for the analysis activities must be collected through various databases shown as Figure 2.

As in the above Figure 4, there are modules had been proposed in the past research. As for this proposed research, the Prescriptive Analytical Logic Model would incorporate the past modules with the proposed modules. Of course, not all the past modules can be directly adopted in the proposed logic model. This is because certain past modules may require enhancement so that it can fit into the proposed logic model to handle the specific process slightly differently with the original module design and implementation in the past research.

Finally, the initial proposed algorithm under the proposed Prescriptive Analytical Logic Model (PAL) includes the main activities presented in Table 2.

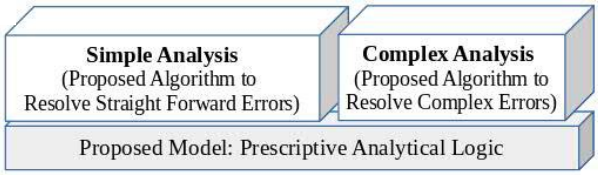

Figure 3. The proposed model consists of two sets of software application analysis algorithm.

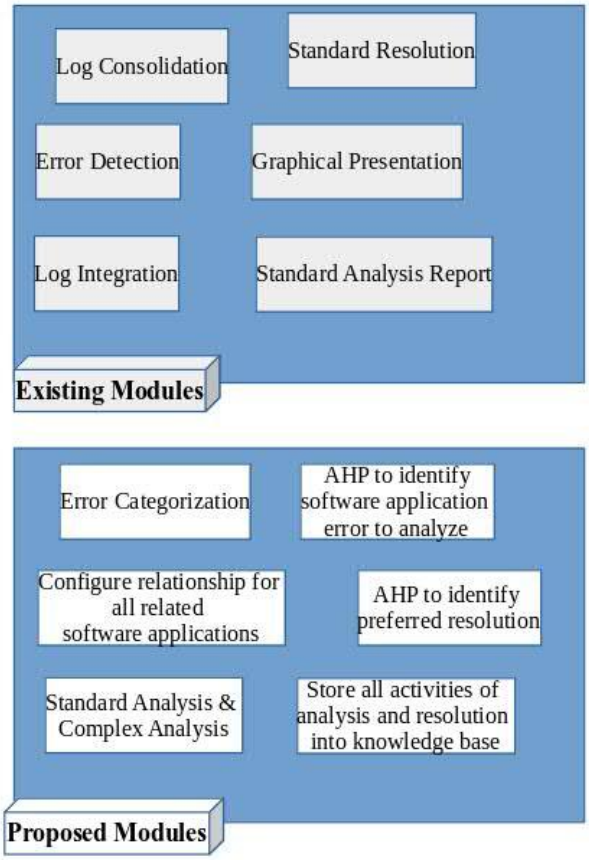

Figure 4. The proposed model introduces new modules to handle specific process respectively.

Table 2

Proposed Process Activity Under Proposed Algorithm

Activity	AHP	Algorithm
1		Collect or Integrate various log files obtained from different software application databases with/without a predefined database schema.
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 Analytic Hierarchy Process (AHP).
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.

Table 2 to be continued

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.

Conclusion

Business operation cannot afford for long downtime. The proposed research is focusing to deliver a proposed Prescriptive Analytical Logic Model along with the AHP inside for decision making. The target is to shorten the overall total time taken of software application downtime by shortening the duration of conducting root cause analysis activity. Although there are also duration on developing, testing, and deploying the resolution under the overall total time taken for resolving the software application error during downtime, this proposed research of the Prescriptive Analytical Logic Model will attempt to provide resolution actions to those software application errors which are not involving software development cycle to develop the resolution. As for those software application errors which are required software development cycle to develop the resolution, the proposed Prescriptive Analytical Logic Model will generate an analysis report and provides advice and suggestions to the identified root cause.

References

Bloom, A. (2014). The cost of application downtime is priceless. *StatusCast.* Retrieved November 28, 2015, from http://www.statuscast.com/cost-application-downtime-pricess/

Gifford, J. (2015). *Data is king: Measuring the impact of logging on your application*. Retrieved February 19, 2017, from https://www.loggly.com/blog/measuring-the-impact-of-logging-on-your-application/

Hoo Meng, W., & Amalathas, S. S. (2019a). An approach towards developing an algorithm for software application error analysis. *Management Studies, 7*(4), 315-324. Retrieved from http://www.davidpublisher.com/index.php/Home/Article/index?id=39805.html

Hoo Meng, W., & Amalathas, S. S. (2019b). A new approach towards developing a prescriptive analytical logic model for software application error analysis. In R. Silhavy, P. Silhavy, and Z. Prokopova (Eds.), *Intelligent systems applications in software engineering* (pp. 256-274). Cham: Springer. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-30329-7_24

Horvath, K. (2015). Using root cause analysis to drive process improvement. *Intland Software.* Retrieved July 2, 2018, from https://content.intland.com/blog/safety-engineering/using-root-cause-analysis-to-drive-process-improvement

Management Logic. (2012). *Root-cause analysis*. Retrieved November 28, 2015, from http://www.management-logic.com/toolbox/sales/Root-Cause%20Analysis/Index.html

Margulius, D. (2006). Tech jobs take stress to whole new levels. *InfoWorld Inc.* Retrieved November 13, 2015, from http://www.infoworld.com/article/2655363/techology-business/tech-jobs-take-stress-to-whole-new-levels.html

Mercer, E. (2015). How technology affects business operations. *OpposingViews.com.* Retrieved November 28, 2015, http://science.opposingviews.com/technology-affects-business-operations-1659.html

Panda, A. (2011). High performance and smarter logging. *DZone.* Retrieved February 20, 2017, from https://dzone.com/articles/high-performance-and-smarter

REDDIT. (2015). *Experienced Dev's: How much of your time do you spend troubleshooting?* Retrieved November 14, 2015, from https://www.reddit.com/r/webdev/comments/3sldcc/experienced_devs_how_much_of_your_time_do_you/

Rinnan, R. (2005). Benefits of centralized log file correlation. *Gjøvik University College.* Retrieved March 16, 2017, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.8787&rep=rep1&type=pdf

Stack Overflow. (2015). *How much time do you spend in production troubleshooting?* Retrieved November 14, 2015, from http://stackoverflow.com/questions/1425069/how-much-time-do-you-spend-in-production-troubleshooting

Success Factors. (2015). *Using technology to increase your business productivity*. Retrieved November 28, 2015, from https://www.successfactors.com/en_us/lp/articles/using-technology-to-increase-your-business-productivity.html

Wong, H. M., Amalathas, S., & 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: https://doi.org/10.24018/ejece.2019.3.5.133. https://www.ejece.org/index.php/ejece/article/view/133

Wikimedia Foundation Inc. (2016). *Electronic data processing*. Retrieved March 4, 2018, from https://en.wikipedia.org/wiki/Electronic_data_processing

Wikimedia Foundation Inc. (2017). *Log file*. Retrieved February 19, 2017, from https://en.wikipedia.org/wiki/Logfile