Management Studies, July-Aug. 2019, Vol. 7, No. 4, 315-324
doi: 10.17265/2328-2185/2019.04.006

An Approach Towards Developing an Algorithm for Software Application Error Analysis

Hoo Meng Wong, Sagaya Sabestinal Amalathas
UNITAR International University, Selangor Darul Ehsan, Malaysia

The proposed algorithm is a conceptual guide which later helps as a guideline to build an independent software plug-in component sitting at the logic tier with human-like intelligent of analysis activities. This algorithm enables the capability on analyzing software application errors which pulls the software log data and other related data from various databases such as Configuration Database, Production Support Ticketing System Database, and Application and System Monitoring Database, and base on the yield data as input information to apply automated analysis action. In addition, based on the proposed algorithm, it can categorize the repeatably pattern of the software application errors in various categories and proactively apply resolution steps automatically to each category of software application errors. The analyzed information will be stored into a separate database catered for upcoming analysis and fine tuning the logic for better decision making in the near future.

Keywords: application analysis, application log, application debugging, log file analysis, log processing

Introduction

The computing era evolved since data-processing until today, software application is required to host on a server box (either on a physical server box or a virtual server box), connecting with network communication (whether it is a Local Area Network, LAN, or a Wide Area Network, WAN), and the data of software application stores at location either in a file or in a database (which is another software application that focuses on handling and processing data into collection of information under an organized manner for accessing) and this database can be hosted on another server box. Due to the complexity of having hardware (physical and virtual hardware), Operating System, Networking for software application communication, and database to allow the software application execution, it would lead to many concerns arises on sustaining the software application execution up-time to support the business operation. The concerns will be explained in the later paragraphs.

When a business company adopts Information Technology (IT) as a tool to enable its business, many required software applications (which are application software designed or customized for users to perform specific tasks within the computer systems) would be purchased into the organization to sustain the daily business operation. Software applications cannot afford to have downtime as it can cause the business operation to cease and this has been stated clearly at Labels: Data Center, Downtime, www.evolven.com (2014).

Hoo Meng Wong, Ph.D. student in Information Technology, UNITAR International University, Selangor Darul Ehsan, Malaysia.
Sagaya Sabestinal Amalathas, Dr., UNITAR International University, Selangor Darul Ehsan, Malaysia.
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.

Regardless a business company owns an IT production support team to provide support to whichever business-as-usual (BAU) system running in the organization, or even a service provider is having a dedicated IT support team running in different shift timing to provide IT support to its specific customer, the time spent on analyzing and trouble-shooting the software application error, and identifying the root cause of it will be long if the input information such as log files or past event of the error will be taking even longer time to locate it before the decided resolution steps can be applied to fix the software application error. To support this problem statement, there are comments obtained from the Internet regarding to time consuming on application trouble-shooting, which are REDDIT (2015) and StackOverflow (2015).

On the other hand, according to the information contributed by both Mercer (2015) and SuccessFactors (2015), business companies are relying on Information Technology (IT) and software applications (i.e., software which is required for performing daily business tasks and processes in the respective business operation) to sustain, to continue, or even to increase business productivity at their business-as-usual (BAU) in today's IT era. When a software application is adopted, it does have up time (when the software application is working fine) and down time (when the software application fails to be accessible or nonoperative) as this is reality on day-to-day business operation, which implies that the challenges and tension are always on the IT support team shoulder to resolve these software application issues as soon as possible during the software application down time. This is because software application downtime would lead to lost of money in business as per the article published by Bloom (2014).

As of the fact, in order to identify the root cause of software application issue before a resolution can be applied, there are several required actions to conduct, such as collecting related information and performing root cause analysis these activities. However, most of the time collecting input information for root cause analysis is time consuming, which is supported by Management Logic (2012) stating 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". In addition, Horvath (2015) 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, looking for efficient method to conduct root cause analysis activities on software application issues is crucial.

Software application generally has built-in logging ability. The objective of software application logging delivers information recorded the activities and events of software application during execution; this information includes appropriate debugging information, and later the information will be analyzed for software application trouble-shooting (reconstructing events after a problem has occurred) or other purposes. The concern arises to the required information logging, which means how much logging information would be sufficient for software application analysis, and what is the required level for each logging sections such as information, error, debug, and fatal would contribute efficiency to the analysis activity. The logging information would generate excessive logging information and lead to the manual analysis activity becoming difficult and tedious to identify the software application error. Furthermore, software application development would concern how much details to be recorded into the log file and at the same time mitigate the performance impact created to the software application, and these concerns had been highlighted by Loggly (2017) and Panda (2011).

Another scenario is that what if the software application error occurred and the cause is outside the software application boundary, such as recently Operating System patching or hardware configuration change request had been conducted and led to the software application running into issue and breakdown. From the information contributed by Rinnan (2005), IT support team will require 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 error occurred.

In addition, despite the effort of software application logging, as the requirements of software application, it must be hosted on a server, and obtains the required server resources such as Central Processing Unit (CPU), Memory, and Disk Space, as mentioned in Wikimedia (2017b). In the situation that available Disk Space is facing insufficient on the server, software application will be finding difficulty on logging information, and under this situation the log file accuracy is impacted due to certain activities and events of software application cannot be logged into the log file. With this situation occurred, it lowers the accuracy of logging information and leads to software application trouble-shooting activity becoming difficult to identify the exact software application error.

As per the Figure 1, Operating System is the middle person who communicates between software application and server resources. Software application has to interact with Operating System to obtain allocated server resources such as CPU, Memory and Hard Disk Space to handle software application processing, which means software application has highly dependency on server resources. Without the server resources, software application will not execute by itself.

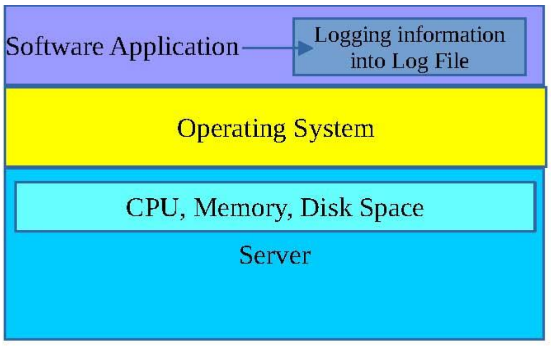


Figure 1. Software application is required sufficient server resources to execute all its functionality.

In another scenario, a server administrator carries out a change request activity during the server maintenance window by installing an Operating System patch, and causes the software application runs with errors. However the software application errors did not specifically indicate the root cause, and eventually the entire software application would require a restoration from the backup tape along with database consistency check to validate the software application functions and data retrieval from its database. The actual of fact (root cause) remains unknown, and therefore the possibility of the same issue will be re-occurred which is high.

With all the mentioned concerns, by depending software application log file alone may not be sufficient for conducting software application analysis whenever error occurred beyond the boundary of software application. Hence a proposed research is required for a new model of process, the log files obtained from various software application databases under the analytical processing logic; this is a research potential to contribute to the business intelligent technology in order to improve the analysis activity when software application error occurred.

Significant of the Study

For the contribution to the business, by lower 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 IT support to face day-to-day software application error challenge in order to provide reliable up-time for the software application utilized in the business organization. On the other hand, business companies can still continue to utilize their existing software applications (without incurring 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.

For the technical beneficial, over the years there were various researches having been done at this area such as consolidating the logs or integrating 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.

Research Objectives

(a) Reduce time spent to conduct analyzing and trouble-shooting activities manually on the software application error.

(b) Reduce human error by automating the analyzing and trouble-shooting activities on the software application error.

(c) Improve accuracy to identify the software application error's root cause.

(d) Propose a new model of analyzing the log files obtained from various software application databases under the analytical processing logic.

(e) Automate on decision making to select the best software application resolution according to the result of analyzing and trouble-shooting activities.

(f) Automate on applying resolution to fix the software application error.

(g) Help in future decision making on software application resolution through historical data trending and past resolution action taken which these information and data are stored in a database for fast retrieval.

Related Work

In the past research, Stewart (2012) is focusing on debugging real-time software application error using logic analyzer debug macros, whereby Eick, Nelson, and Schmidt (1994) are focusing on presenting the error logs in a readable manner. Moreover, Peng and Dolores (1993) suggested focusing on error detection in software application at the time of software development and maintenance, and Salfner and Tschirpke (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, and some literature suggested that the software application error analysis would be better if it is built in the software development process and this approach is still within the same application development boundary without factoring in any other area of concerns which 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 to be recorded into the software application error log which is stored into software application database. 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. On the other hand, Murínová (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. This is a great potential to propose a model to research a new approach towards developing an algorithm for software application error analysis.

Research Proposed

The objective of this research is to propose an algorithm which can be applied to develop a complete analytical logic for analyzing the software application log and the logs retrieved from various software application databases, and to apply the necessary action to resolve the software application errors manually, semi-automatic, or fully automatic based on the preferred settings on the logic.

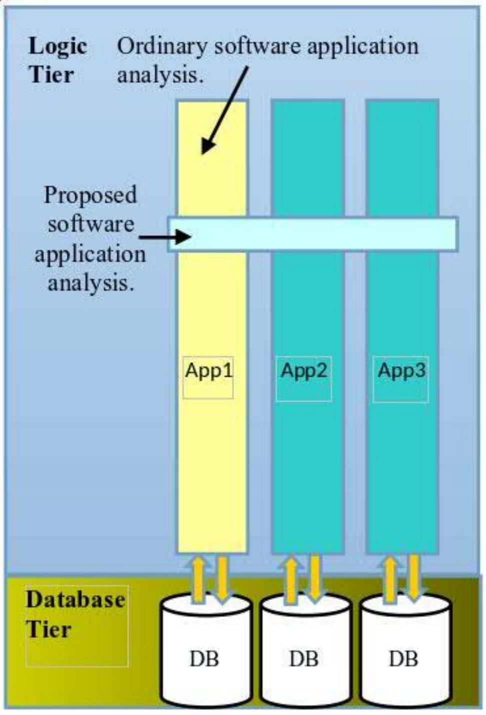

Logic Tier
Ordinary software application analysis.
Proposed software application analysis.
App1
App2
App3
Database Tier
DB
DB
DB

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

Proposed Research Scope

The proposed scope of this research is on defining the algorithm for simple and complex analysis to form a complete analytical logic on conducting analysis and trouble-shooting activities automatically when reacting to software application error. The reliable information can be collected through various databases shown as Figure 2.

From the initial proposed algorithm based on overcoming the current facing problem of software application error, this algorithm includes:

- Integrate various log files obtained from different software application databases.
- Identify possible log data and select the necessary log data for analysis.
- Analyze the selected data and define possible resolution option.
- Allocate weight to each possible resolution option based on *Analytic Hierarchy Process (*AHP).
- Shortlist the preferred resolution option under the highest weight.
- Deploy the preferred resolution option to fix the software application error under the predefined condition.

Simple Analysis

To the best of my knowledge, the proposed algorithm mainly consists of two analysis areas, which are simple and complex analysis to form a complete analytical logic. For simple analysis area, it is to build the pre-defining logic to handle the common software application error, whereby this model is to guide the system builder on answering a set of pre-defining questions on common software application errors and carry out the predefined activities only reacting to these common software application error, for example, restarting the software application process if it is stopped.

Complex Analysis

On the other hand, for complex analysis, it is to build a logic which requires collecting necessary data from three databases, which are Configuration Database, Production Support Ticketing System Database, and Application and System Monitoring Database. 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 combining with the pre-defined 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 who administrate the software applications. This complex analysis would have three different modes which are “manual”, “semi-auto”, and “fully-auto” offered to the IT support team whereby the complex analysis area performs the suggested steps and carries out the action against the software application error by handling predicated system behavior dynamically in order to prevent future application failure based on the permission given on the offered mode by the IT support team.

By focusing on 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, and hence if the validating activities can be predefined into a checklist and base on the combination of the answers (yield from the validating activities in the checklist) to pick up the ultimate predefined solution and then to react to the incident automatically. This would be the logic 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 as the person who handles the incident requires obtaining the software application log files and searching 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 he or she can identify the software application error root cause, and suggest to apply the agreed resolution steps (it may involve peers in the IT support team to discuss the preferred resolution steps or even consult with the application developer, and the resolution steps are approved by the Change Management if required) 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, then based on the outcome of the analysis activities, human expects to see a list down of each possible root cause along with either the suggested further human activities (which involve human to handle this incident by his or her own to perform further investigation based on the given analysis outcome) or the proposed resolution steps (which require human's agreement to proceed with the suggested resolution steps) in a complete list, then the decision is on the human to choose which is the preferred option. If the human choose to precede with the suggested resolution steps, then the human will receive the final question on whether he or she agrees to let the automated activities execute the same suggested resolution steps automatically in the future if the same incident occurs again. This is the logic which has the ability to handle unpredictable software application errors by performing simulated analysis activities comparing with human; we call this logic as *complex analysis*.

The complex analysis will pull the related logs based on specific time frame (duration) before and during the software application failure from various databases such as application logs for software application error evidence, configuration management logs for understanding any recent applied software application patches or Operating System patches, performing and capacity monitoring logs for any hardware resources running insufficient, and production support ticketing tool logs for cross-checking any related issue recently occurred under the predefined database scheme. These log 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, but when the specific number of reoccur incidents hits, the complex analysis will be activated to perform the required analysis activities automatically to produce the complete analysis report and suggestion(s). Based on this suggested design, the complex analysis would have a loosely but it is fairly important relationship with the simple analysis as the complex analysis needs to understand how many times the simple analysis has handled the same incidents in the past and this information would be crucial to make a decision on suggesting the reasonable resolution steps to the human after the complex analysis produces the analysis report.

Analytic Hierarchy Process (AHP)

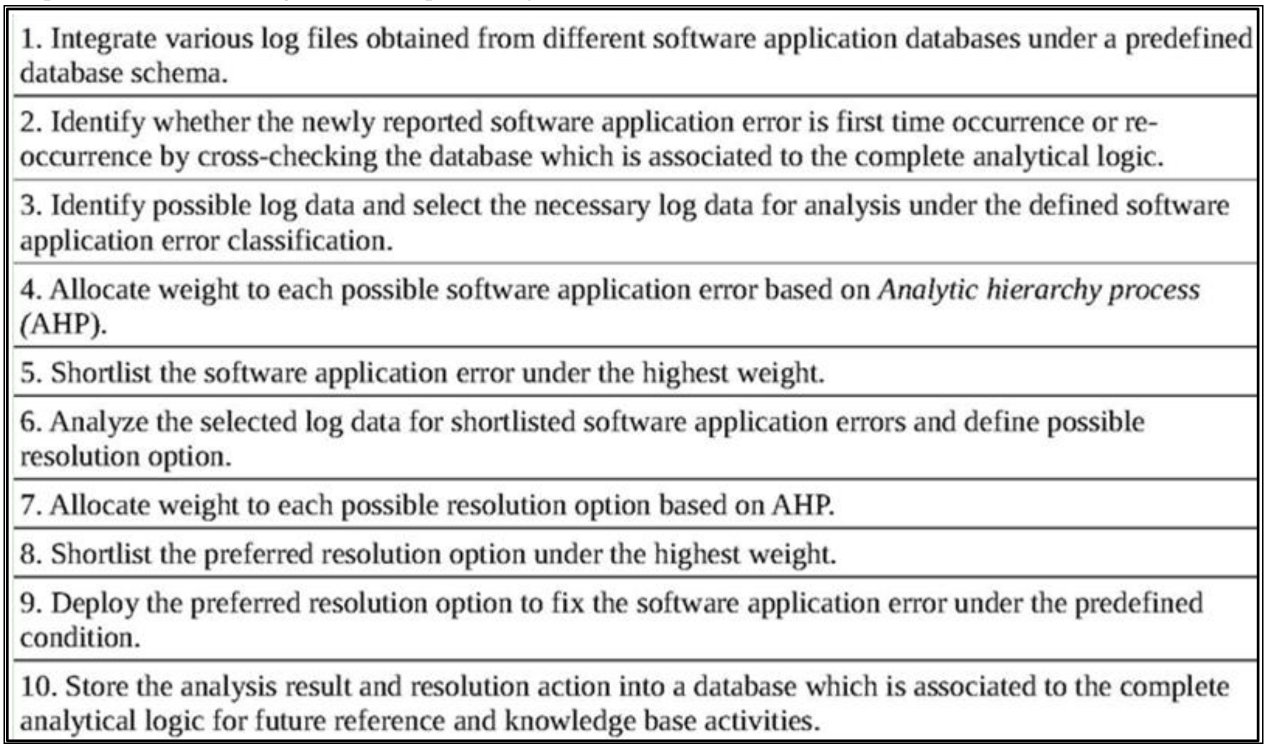
With the proposed model behind the complete analytical logic for conducting software application error root cause analysis activity, the shortlisted proposed resolution action in fact can be processed by applying the Analytic Hierarchy Process (AHP) which was developed by Thomas L. Saaty stated in Wikipedia (2015), and based on the processed data the complete analytical logic shortlists the best solution to be proposed or to be applied against the software application incident automatically. This is because AHP is a decision tool, which can be utilized to weight the resolutions under a hierarchical structure and decide the best resolution among the shortlisted resolutions against the complex software application errors analyzed and identified by the proposed complete analytical logic.

Proposed Algorithm

With the proposed algorithm of the complete analytical logic which consists of both simple and complex analysis, it can be integrated into an independent software plug-in component sitting at the logic tier. Under such design, it helps to avoid any interference of existing software application execution while the plug-in component can still pull the related log information from software application and various databases such as Configuration Database, Production Support Ticketing System Database, and Application and System Monitoring Database. These databases store the logged event information with predefined schema to allow the complete analytical logic to initiate Structured Query Language (SQL) queries in order to retrieve the necessary logged event information as data whenever software application error arises, and based on the yield data as the input information to the complete analytical logic, reactive action will either be responded by the logic automatically, or consolidate the input information into a support ticket raised in the IT support system and alerted the IT support engineer to work on the reported software application error. In addition, based on the given logic, it can categorize the repeatably pattern of the software application error in various categories and proactively apply resolution steps automatically to each category of software application error. At the end of each analysis of the software application error, the analyzed information will be stored into a separate database catered for upcoming analysis and fine tuning the logic for better decision making in the near future.

The proposed algorithm includes the following activities:

Table 1
Proposed Process Activity Under Proposed Algorithm



1. Integrate various log files obtained from different software application databases under 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 complete 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.
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 complete analytical logic for future reference and knowledge base activities.

Environment

In this proposed work, we would require a multiple-tier software application which involves web tier, application 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 information from various databases and store in a separate location for information retrieval later. A new standard database schema will be also proposed on this research, and it will be applied across all the required databases. This will help on retrieving data from various databases when the analysis activity is triggered at the logic tier when software application occurs in the ordinary software application log file.

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.

Conclusion

Today there are many software application error analysis techniques in the market, and mostly these techniques are applied during the software development process for software application. In fact there are tools used on analyzing the software application error log files (which are produced during the software application execution), and these tools mainly analyze based on the collected error logs which is reasonable if the hardware resources of the system are under a healthy stage. Without any doubt whenever the hardware memory as one of the main hardware resources which is running low, or the Central Processing Unit (CPU) resource is saturated, or even Hard Disk Space is running insufficient for the software application, with all these possible causes which would lead to the software application it may not log its error correctly into the error log files. Hence, if the root cause analysis is solely depending on software application error log files may not be accurate. By looking at an algorithm which improves the software analysis technique into a more human-like intelligent, this proposed algorithm will be embedded into a software plug-in component to serve as a logic, and the software plug-in component will be sitting at the logic tier to automate the software application error analysis in a proactive manner for the existing software application. The research of this proposed algorithm would be a great potential which brings benefit to business industries because companies which are heavily depending on software applications for their daily business activities would help to mitigate the software malfunction risk, and improve the confidence of the software application users to reliance on their software applications.

References

Andrews, J. 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

Atwood, J. (December 3, 2008). The problem with logging. Retrieved from https://blog.codinghorror.com/the-problem-with-logging/

Andrews, J. H. (2007). 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

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

Error Handling, Auditing and Logging. (July 17, 2015). Retrieved from https://www.owasp.org/index.php/Error_Handling,_Auditing_and_Logging

Eick, S. G., Nelson, M. C., & Schmidt, J. 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)

Hasan, M. 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)

Horvath, K. (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)

Loggly Inc. (2017). Retrieved from https://www.loggly.com/blog/measuring-the-impact-of-logging-on-your-application/

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

Mercer, E. (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, J. (2015). Application log analysis. Retrieved from http://is.muni.cz/th/374567/fi_m/thesis_murinova.pdf

Ornelas, L. V. (2003). SANS Institute. Important of event logging. Retrieved from https://www.giac.org/paper/gsec/3297/importance-event-logging/105437

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

Peng, W. W., & Dolores, R. (1993). Wallace. Software error analysis. NIST Special Publication. Retrieved from http://www.geocities.ws/itopsmat/SoftwareErrorAnalysis.pdf (accessed on 12th December 2015)

REDDIT. (2015). Experienced Dev’s: How much of your time do you spend troubleshooting? Retrieved from https://www.reddit.com/r/webdev/comments/3sldcc/experienced_devs_how_much_of_your_time_do_y ou/ (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)

Salfner, F., & Tschirpke, S. (2015). Error log processing for accurate failure prediction. *Www.usenix.org*. Retrieved from https://www.usenix.org/legacy/event/wasl08/tech/full_papers/salfner/salfner_html/ (accessed on 12th December 2015)

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, D. B. (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-logi c-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)

WWW.evolven.com, Labels: Data Center, Downtime. (2014). Downtime, outages and failures—Understanding their true costs. *Evolven Software*. Retrieved from http://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html (accessed on 13th November 2015)

Wikimedia Foundation Inc. (2015). Analytic hierarchy process. Retrieved from https://en.wikipedia.org/wiki/Analytic_hierarchy_process (accessed on 13th November 2015)

Wikimedia Foundation, Inc. (2017a). Application server. Retrieved from https://en.wikipedia.org/wiki/Application_server (accessed on 28th June 2017)

Wikimedia Foundation Inc. (2017b). Logfile. Retrieved from https://en.wikipedia.org/wiki/Logfile (accessed on 28th May 2017)