Building the Monitoring Systems for Complex Distributed Systems: Problems and Solutions

Olga Korableva[1,2], Olga Kalimullina[1] and Ekaterina Kurbanova[1,3]
[1]*ITMO University, S-Petersburg, Russian Federation*
[2]*St. Petersburg State University, S-Petersburg, Russian Federation*
[3]*"CS Information Technology" Ltd, S-Petersburg, Russian Federation*

Keywords: Monitoring Systems, Complex Distributed Systems, Big Data, System Approach, Innovations, Strategic Management.

Abstract: Complex distributed systems are of more significance nowadays, due to a broader range of its use and because of provision of better services to users. It is clear, that system health needs continuous monitoring, while running software apps that are ensuring the implementation of the business processes, working with Big Data, etc. In the course of this study a monitoring system has been developed. It meets all modern requirements, such as scalability, flexibility, comprehensiveness of necessary data and ease of use. In order to identify unified problems in development of monitoring systems for complex distributed systems and respectively - the solutions for their elimination, the data regarding IT-architecture most common types used in modern companies, related to fault-points of business apps has been gathered and analysed. All of identified problems and optimal solutions to eliminate them were aggregated in line with the three development stages of monitoring system, such as: development of servicing model for system-of-interest, implementing tools to detect objects of monitoring, generating a health map of system-of-interest. In order to develop monitoring systems for complex distributed systems taking into account the architecture of these systems, all of the gathered data was analysed, and we articulated all problems and optimal solutions for their elimination as well.

1 INTRODUCTION

The result of this study is the development of monitoring system of software package, designed for production records and further turnover for all of company's manufactured products. The serviceability of the software package was crucial for the company's business, because in the case of its failure, the shipment of manufactured products would become impossible. To get up-to-date information on the status of the software package, it was necessary to design a monitoring system.

The study identified algorithm, enabling to make this system integrated, scalable, and easy-to-use. The implementation of the designed monitoring system onto one of the largest Russian company gave the option for testing of the developed algorithm.

Now, the company is already successfully interacts with the system for 7 months. The developed algorithm, articulation of problems and optimal solutions of their elimination, formulated concepts – all of it contributes to the theory and practice of building monitoring systems, including working with such systems as Big Data.

"Big Data" is a term, used for the combination of complex algorithms, approaches and techniques for gathering, processing and display of massive volume of data (Russom, 2013), which are rapidly emerging as a result of technological and business innovations (Thakuriah, et al., 2015). These methods should be not only effective in the environment of continuous growth, but shall also possess a high level of fault tolerance and availability (Toporkov et al., 2015). Many algorithms are multithreaded, volumes of information are enormous, and therefore, even sporadic failures can lead to the loss of huge amounts of data. Monitoring system gives you the option to inform the administrator about detected negative trends, what in turn, speeds up the reaction of the administrator, and therefore increases the level of system sounded (healthy) functioning (Yang and Chang, 2011; Korableva and Litun, 2014).

The term "Monitoring system" refers to a system, intended to track health (sound) status, as

Korableva, O., Kalimullina, O. and Kurbanova, E.
Building the Monitoring Systems for Complex Distributed Systems: Problems and Solutions.
DOI: 10.5220/0006271002210228
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2*, pages 221-228
ISBN: 978-989-758-248-6

well as displaying warnings for troubleshooting on the monitored object (Guo et al., 2010; Dourish et al., 2000; Coulouris, 2013). Monitoring of distributed systems includes the gathering, interpretation and display of data, related to the interactions between processes, including running in parallel (Joyce et al. 1987).

There is IT- architecture in any kind of business, monitoring of serviceability of which is necessary for early detection of failures and having solutions for their elimination (Latyshev and Akhmetshin, 2015; Korableva and Kalimullina, 2016a). A malfunction of IT-infrastructure for some companies leads to downtime, which often involve material losses.

Therefore, there is a burning reason to arrange the system, which would allow checking on status of all company's IT-architecture components, as well as timely sending alerts regarding impending failures in work (Becker et al., 2016; Korableva and Kalimullina, 2014).

This article describes a design concept for monitoring systems of complex distributed systems, developed in the course of this study and which meets the following requirements:

- Scalability of solution
- System usability
- Comprehensiveness of developed fault-adaptive map of system-of-interest

Based on gathered and analyzed data, have been defined solutions for problems, which would be faced during the process of building monitoring system.

2 METHODOLOGY

Any system is a set of components, each of which is subdivided into modules, which in turn can be represented as the sum of even smaller components. To monitor such hierarchical structures, we shall be able to identify each component of the system. This feature provides the following concept:

1. A class to identify the specific component of the system is created
2. The conditional description helps clearly determine, whether existing objects in the monitoring system are the desired components or not. For example, one of the most common types of verification is the search for key in the registry.
3. The scope of existing objects for verification in the monitoring system is determined.
4. Based on preset parameters, the monitoring system performs search of specified criterion on the specified scope of objects. In case of successful validation – an instance of a Class is created on the object and identifies a component of the system-of-interest.

Thus, all components of the company's IT-architecture identified in the monitoring system as playing a specific role. If the same architectural object plays multiple roles (for example, one server is used to handle company's database and some user application), then this object will be identified in monitoring system as multitasking.

Identification of roles of monitored objects allows to get list of the system key components, serviceability of which is crucial for this company. To build a servicing model, it is also essential to specify the relationship between used components. The most common are the following types of relationships:

- Hosting relationships
- Containment relationships
- Reference relationships

Servicing model, built in such way for the system-of-interest, reflects all key components, their levels and interoperations, but does not reflect the state of the system healthy functioning. To ensure serviceability of monitored components, a series of validations allowing identify the health status shall be performed In general such validations are called monitors, and are split into 3 categories, such as: 1) Unit monitors; 2) Dependency monitors; 3) Aggregate monitors.

3 RESULTS

3.1 Summary of Contributions

Building monitoring systems for complex distributed systems investigated in a number of excellent previous articles, among which are (Sigelman et al., 2010; Sloman, 1994; Ferdowsi et al., 2014). There are a number of studies in the field of monitoring of distributed systems, part of which mainly focused at troubleshooting (Phuc and Son, 2012, 2013; Kshemkalyani and Singhal 2008), while others focused at performance optimization (Sharma et al., 2013, Sambasivanm et al., 2014; Korableva and Guseva, 2015).

This study aims to identify the problems encountered in the process of building monitoring system, as well as to search for solutions and their testing during all three-development stages, such as - development of servicing model for system-of-

interest, implementing tools to detect objects of monitoring, generating a health map of system-of-interest. The study proposed the solutions for the some issues. A concept, allowing minimizing the load on objects of the company infrastructure has been introduced. It have been also introduced some principles, following of which allows you to obtain a complete, concise information about system's health. These concepts also focus at minimizing administrator's response time. The identified best practices of using dependency and aggregate monitors allow you to get a comprehensive health scheme of the system, taking into account all of existing, both explicit and hidden relationships between objects of monitoring system. We have also formulated principles for system's faultfinding.

The study found the answers to the following questions: exactly what kind of components are the key components of the system; what is required level of detail for key components; what values should be set as Class variables; and whether this setting may have a harmful impact on the monitoring system activity. All these issues can be combined into a single overall objective - an implementation of scalable monitoring system, which would not require drastic revisions due to new input data. Such requirements are also relevant when dealing with Big Data (Son and Phuc, 2010; Korableva and Kalimullina, 2016b).

Completing of this task requires thorough comprehensive approach to the analysis of the system-of-interest. This article will describe such approach from the perspective of the following three scale steps:

- Development of servicing model for the system-of- interest.
- Implementing tools to detect objects of monitoring
- Generating a health map of system-of-interest.

3.2 Development of the System Servicing Model

The system-servicing model is a set of Classes, as well as a set of the relationship between them, which as a whole is cloning the structure of the system itself.

3.2.1 Issue 1: Choosing an Approach to Build Servicing Model

The main dilemma when choosing an approach for building servicing model is to define the necessary scope of monitoring. It could be either the entire complex distributed system or just its specific module, serviceability of with is crucial. Moreover, a scope of monitoring may not only be restricted by system-of-interest, but also include the pool of all systems that are integrated with it.

This study exposed two approaches for building servicing model, which are in line with planned scope of monitoring. Table 1 reflects all the advantages and disadvantages of both approaches. However, you cannot select one of them as all-in-one approach to build servicing models for all kind of systems. Each system-of-interest requires its own optimal approach. Therefore, the best approach for the analyzed software package is a comprehensive approach and, in the case of monitoring for Big Data, it is better to choose a modular approach. Table 1 shows advantages and disadvantages of comprehensive and modular approaches.

Table 1. Advantages and disadvantages of comprehensive and modular approaches.

	Advantages	**Disadvantages**
Comprehensive approach	Reflection of information about: 1. the whole system under consideration 2. systems integrated with system consideration	Great efforts for the implementation of all stages of the project
Modular approach	Saving time at the following stages: • Planning of Architecture • Development of documentation •Phase matching •System Development	• Narrowly specialized data • More complex process of system scaling •Inability to obtain information on the status of related systems • Lack of information on the premises of negative trends • A large number of blind spots

3.2.2 Issue 2: Definition of Level of Detail in Servicing Model

The next issue of developing servicing models for the system-of-interest is to define a list of its key components for monitoring. Even though there is a need to obtain full and comprehensive data regarding monitored object, the designed monitoring system shall be concise and user-friendly. Based on experience in building monitoring systems for complex distributed systems, the following algorithm has been developed to set the level of detail in the servicing model. This innovative

algorithm is taking into account all the key components of the system and avoiding cluttering in the design of servicing model (see Figure 1):

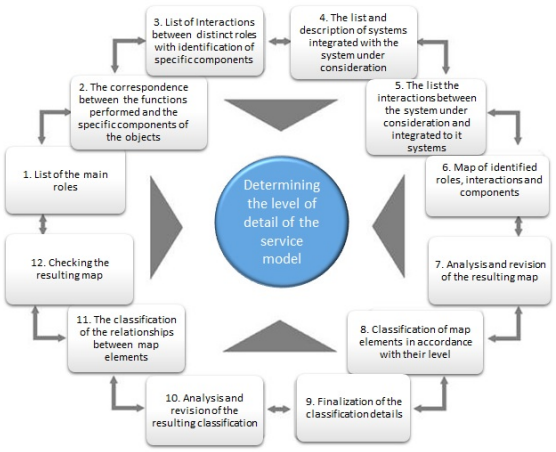

Figure 1: Algorithm to set the level of detail in the servicing model.

1. Making list of essential roles, played by the system, description of its intended use and a list of functionalities.
2. Setting up correspondence between executing functions and specific object components.
3. Identifying interactions between selected roles, specifying specific components to play these roles, involved in each of the interactions. When it is done, one can see an outline of future servicing model, especially a list of the essential roles, its mandatory components, as well as operational interrelations.
4. Definition of the list of other systems, integrated with this system-of-interest, as well as a description of their intended use.
5. Identification of all interactions between the system-of-interest and other systems integrated with it, taking into account the specific components involved in these interactions.
6. Mapping detected roles, components and interactions.
7. Make the analysis of the obtained map, as well as a study of the system-of-interest for missing components, used for system functions execution. Making adjustments for existing map.
8. Make classification of elements on the map in line with their levels. A good example of such a classification is a clustered SQL Server. The top level is the cluster itself, the next level consists of nodes (servers) of the cluster, each of which runs a number of utilities, ensuring the integrity of SQL Server. In this way, the following chain represents the classification of levels for SQL Server in a simplified form: cluster – server – utilities. It is also possible to make allotment into separate classes, database roles, tasks, and other components that play a key role for serviceability of the system-of-interest.
9. Finalize details of the obtained classification in line with functional tasks, performed by the mapped elements. A good example of it might be the scope of utilities running on the server, when SQL Server is in place. In addition to those utilities, that provide operability of the SQL Server, some other Agents utilities of contiguously allocated system can also be used any related systems. In such case, it makes sense to deepen classification of Class services, making it more detailed.
10. Make analysis of the obtained classification, refinement of the details.
11. Make classification of relationships between mapped elements, which is based on three types of existing relations (Hosting, Containment relations, Reference relationships).
12. Do verification of the obtained map, which is a framework of the servicing model under development.

The above-described approach for choosing level of detail, as well as identification of the scope of monitored objects, permits to take into account all of employed system components, and at the same time - not overloading the design with irrelevant data.

3.2.3 Issue 3: Choosing Concept of Setting Objects Variables

A good practice in building a servicing model is to set key data about objects of monitoring as variables (parameters). This information either would be used for health checking or would be information of interest for system administrator and for other users. A server name, AD site, regional location, software version and many others that describe the given object could serve like such parameters.

However, use of large number of parameters may overload monitoring system and make it less convenient to use. To avoid such situation, this study suggested innovative concepts of setting up parameters of objects. Figure 2 displayed their descriptions and advantages of using.

Use of the described innovative concepts in design of a servicing model helps to complement the map of system-of-interest seamlessly, without overloading database and not complicating

operational process of monitoring system, which is under development.

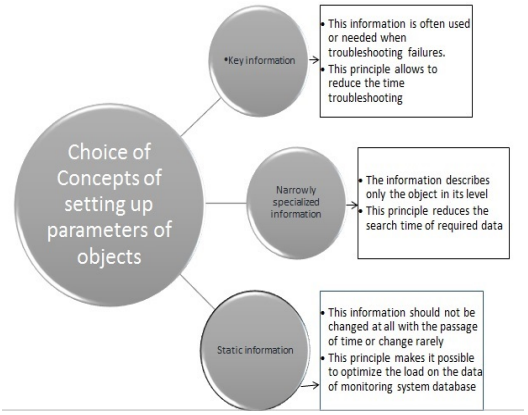

Figure 2: Concepts of setting up parameters of objects.

3.2.4 Issue 4: Definition of Scope for Subsidiary Objects

An aggregation of existing Classes on any grounds is required in order to have an easy review on the structure of a distributed application, as well as for configuring a number of additional functions, when setting up the monitoring system.

This can be configuring of various alerts and warning messages to system administrators of specific brunch, regarding the failures of the system components that belong to this Office. In addition, the optimal solution for monitoring of complex distributed system, that has several branches, is a logical separation of monitored objects with respect to branches involved.

For additional logical classification of existing objects, it is necessary to create an additional scope for subsidiary objects. Such subsidiary objects are Groups and Containers. The design intent of these objects, as well as examples of their successful use reflected in Figure 3 - "Definition of scope for subsidiary objects".

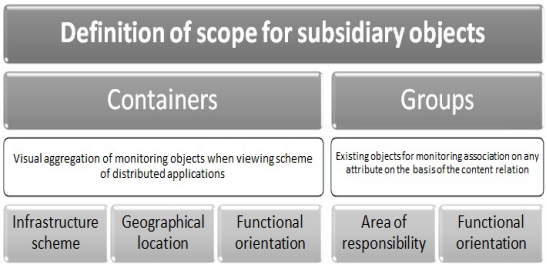

Figure 3: Definition of scope for subsidiary objects.

3.3 Implementing Tools to Detect Objects of Monitoring

To generate a health map of system-of-interest, the monitoring system has to detect the required objects. The development of this aspect of monitoring system requires special attention, because sloppy implementation of it could lead to a performance downturn not only of all components and monitoring area, but of the entire company infrastructure as a whole. Within the framework of this study, we identified the reasons for this negative impact and articulated the following solutions and key principles for implementing detection tools.

We can articulate the following key points in the implementation of detection tools for monitored objects:

1. To set up the optimal time-out value between two search iterations of objects.
2. To select the lowest possible scope of existing objects for search for the required objects.
3. To make an intense use of the detection tools, consistent with registry data.

3.4 Generating a Health Map of System-of-Interest

Making of health map for any system is always based on the data, received upon verifications that system components are compliant with some preset condition. These verifications are called monitors, and are divided into 3 categories: unit, aggregate and dependency. Experience has shown that a number of practices allow to use such categories of monitors to build a complete and easy-to-use monitoring system.

3.4.1 Best Practices of using Unit Monitors

Unit monitors are designed to check on any state aspect of the system-of-interest. Given aspect could be described by an execution of just one task or by a number of such tasks. Within the framework of this study, the following best practices of using unit monitors have been articulated:

1. Allocating separate monitors to carry out of individual tests intended for checking on the same state aspect of the system
2. Using the smallest number of monitor states
3. Making use of warning condition, providing that:
   a. there is a possibility of preventive measures
   b. absence of error does not ensure health aspect of the system
4. Making adjustments for timeout duration

between the verification activities based on criticality state aspect.

All of the above-specified practices are focused on getting of such picture healthy functioning of monitored object, which contains complete but concise information about occurring faults, substantially minimizing, therefore, administrators' response.

3.4.2 Best Practices of using Aggregate Monitors

Usage of aggregate monitors allows you to create a hierarchical map of health, consequently simplifying the process of handling the monitoring system.

There are 3 the most common guidelines to aggregate monitors:

- Accessibility
- Configuration
- Performance capacity

Further development of the monitoring system may lead to increase in the number of unit monitors to such an extent, that even split of conducted verification into these 3 categories will not permit to get rid of unreadable huge list of carried out verifications. In such a case, if it is possible, it is essential to set up additional subcategories that should be a satisfactory to the only principle: aggregating of several monitors in one aggregate monitor should be based on any criteria, which is common to all of the above-mentioned unit monitors.

3.4.3 Best Practices of using Dependency Monitors

Dependency monitors are used to verify state of any object in the state of another object. Figure 4 shows Code of dependency monitor in xml.

```
<DependencyMonitor ID="
USAIS.SQL.Cluster.Dependency.SQL.Filial
Watchers"
Accessibility="Internal" Enabled="true"
Target="BEL!USAIS.SQL.Cluster"
ParentMonitorID="Health!System.Health.A
vailabilityState" Remotable="true"
Priority="Normal"
RelationshipType="BEL!USAIS.SQL.Cluster
.Contains.USAIS.SQL.FilialWatchers"
MemberMonitor="Health!System.Health.Ava
ilabilityState">
<Category>AvailabilityHealth</Category>
<Algorithm>WorstOf</Algorithm>
</DependencyMonitor>
```

Figure 4: Code of dependency monitor in xml.

The essence of the design is to detect, state of what element should be duplicated in the state of another element. The presented section of code bears not only identity code of dependency monitor, but also code of the target object and source-object of the state. Figure 5 shows the illustrative example of dependency monitor usage – it represents a hierarchical model of dependency monitor relating to workflow. The client, from whom a check-up of the database availability is carried out, plays the role of source-object of the state.

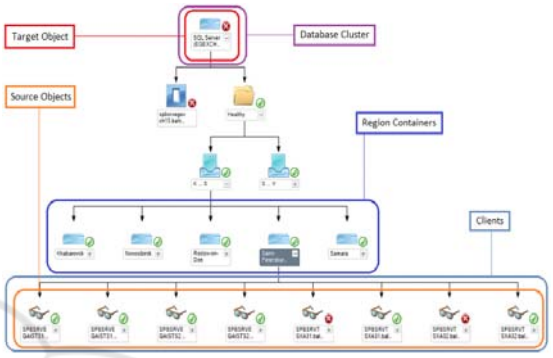

Figure 5: The hierarchical model of dependent monitor.

The cluster of database itself plays the role of the target object. In this case, the duplication of the state is needed, due to specifics of verification: even though database availability is carried out on the client side, this aspect of health refers not to the client, but to the database itself.

Figure 5 shows the structure of monitoring system for under review software package, used to display testing procedure of system workflow availability. All clients are distributed across regions and are subordinated to the cluster. Quantity of dependency monitors is equal to the number of clients monitors are set for the cluster itself. Each dependency monitor of the cluster corresponds to one of the clients.

Therefore, we can mark the following best practices of using dependency monitors:

1. Using dependency monitors, if health state of one object depends on health of another object.
2. Using dependency monitors for estimation of subsidiary objects health status.

3.4.4 Creating a Summary Report on Negative Trends in Running Software Package

Thanks to all conducted work, we have a chance of getting a summary report regarding negative trends in running of the whole software package. This

innovative report is presented in the form of monthly statistics, where aggregation was carried out by month, by region, by computers that have failed, as well as by a specific date. Moreover, statistics was done just for those regions, where faults in system functioning have been registered.

Figure 6 shows monthly statistics on the negative trends in running analyzed software package.

Figure 6: Monthly statistics on the negative trends in running analyzed software package.

4 CONCLUSIONS

In the course of this study, we managed to articulate issues (problems) and solutions, with which developer will be facing while building of monitoring systems for complex distributed systems. These concepts provide means for covering of all the key system-of-interest components, provide ease of handling of designed monitoring system, as well as contribute to building of relevant, complete and concise health map.

The described innovative concepts also facilitate ability to make the system more flexible in the course of its further modernization. In the process of identifying the described concepts, all the major stages of development for monitoring system, have been reviewed, specifically:

1. Development of servicing model.
2. Implementing tools to detect objects of monitoring
3. Generating a health map of system-of-interest.

All data, obtained in the study, serve the possibility to deepen into the task of building of visualization maps for status of the system-of-interest, as well as to explore the possibility of implementing automatic solution for emerging failures in operation. That is also very important when working with Big Data, for it is a perfect example of an integrated system of many components and transactions with high availability demands.

However, for applying of preventive measures is often require to get some statistics for the reporting period. Such data can be formed, if an optimal monitoring system, which is not overloaded with redundant information, but contains a description of all key aspects of operation for system-of-interest is in place.

ACKNOWLEDGMENTS

The research is supported by the RFBR grant 16-29-12965.

REFERENCES

Becker J., Kozyrev O., Babkin E., V. Taratoukhine, N. Aseeva. 2016. *Emerging Trends in Information Systems: Recent Innovations, Results and Experiences.* Springer Int. Publ., Switzerland, ISBN 978-3-319-23929.

Coulouris G., Dollimore J., Kindberg T., Blair G. 2013. *Distributed Systems: Concepts and Design*. Fifth edition. Addison-Wesley.

Dourish P., Swinehart D., and Theimer M.. 2000. The Doctor Is In: Helping End Users Understand the Health of Distributed Systems. *In Proc. of the IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, December 2000.

Ferdowsi M., Benigni A., Löwen A., McKeever P., Monti A., and Ponci F.. 2014. New Monitoring Approach for Distribution Systems. *Proc. of 2014 IEEE International Instrumentation and Measurement Technology Conference* (I2MTC), 12-15 May 2014.

Ferdowsi M., Zargar B., Ponci F. and Monti A.. 2014. Design Considerations for Artificial Neural Network-based Estimators in Monitoring of Distribution Systems, 2014 *IEEE International Workshop on Applied Measurements for Power Systems* (AMPS 2014), (24-27 September 2014, Aachen, Germany).

GE Intelligent Platforms. http://www.ge-ip.com.

Guo, C.G., Zhu, J., Li, X.L. 2010. A Generic Software Monitoring Model and Features Analysis. *Proc. of the 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, Washington DC, 2010, pp. 61–64.

Joyce, J., Lomow, G., Slind, K., Unger, B. 1987. Monitoring Distributed System. *ACM Transactions on Computer Systems*, Vol. 5(2), pp. 121–150.

Korableva, O., Guseva, M. 2015. Activation of innovation processes in banks as a result of the implementation of basic basel accord provisions. Ikonomicheski Izsledvania, Volume 24, No 3, pp. 108-128.

Korableva O., Kalimullina O. 2014. The Formation of a single legal space as a prerequisite for overcoming systemic risk. Asian Social Science, Vol. 10 (21), 256-260.

Korableva, O., Kalimullina, O. 2016 a. An Innovative Approach to Strategic Risk Management in Banking: Russian Banks Case Study. WSEAS Transactions on Business and Economics, Volume 13, Art. #25, pp. 269-282.

Korableva O., Kalimullina O. 2016 b. Strategic Approach to the Optimization of Organization Based on the BSC SWOT Matrix. Proceedings of the International Conference on Knowledge Engineering and Applications. ICKEA, 2016. Singapore, September 28-30, 2016. p. 212-215.

Korableva O., Litun V. 2014. The potential of transitive economies' growth based on innovative strategy. WSEAS Transactions on Business and Economics. 11 (68), 725-736.

Kshemkalyani, A.D., Singhal, M. 2008. *Distributed Computing Principles, Algorithms, and Systems*. Cambridge University Press.

Latyshev, I. O., & Akhmetshin, E. M. 2015. Methodological approaches to analyzing the indicators of human capital management in the interests of innovation development of enterprise. *International Business Management*, 9(6), 1565-1570. doi:10.3923/ibm.2015.1565.1570.

Yang, S.Y., Chang, Y.Y. 2011. An active and intelligent network management system with ontology based and multi agent techniques. *Expert Systems with Applications,* Vol. 38(8).

Phuc, T.N.H., Son, L.V. 2012. Monitoring of large scale distributed systems based on SNMP development. *The Journal of Science and Technology*. Danang University, Vol. I(8), 2012, pp. 79–84.

Phuc, T.N.H, Son, L.V. 2013. An Online Monitoring Solution for Complex Distributed Systems Based on Hierarchical Monitoring Agents. Springer Int. Publ. Proc. of the Fifth International Conference KSE 2013, Volume 1, pp. 187-198.

Russom Ph.. 2013. Operational intelligence. Real-Time Business Analytics from Big Data. *TDWI Checklist Report.* http://www.splunk.com/pdfs/white-papers/real-time-business-analytics-from-big-data.pdf-3.

Sambasivanm R., Fonseca, R., Shafer, I., and Ganger, G. 2014. So, you want to trace your distributed system? Key design insights from years of practical experience. *CMU PDL Technical Report*, CMU-PDL-14-102, April 2014.

Sharma A., Chen H., Ding M., Yoshihira K., Jiang G.. 2013. Fault Detection and Localization in Distributed Systems using Invariant Relationships. *43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (DSN), 24-27 June 2013, pp. 1 – 8.

Sigelman, B., Barroso, L., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., Jaspan, S., and Shanbhag, C. 2010. Dapper, a large-scale distributed systems tracing infrastructure. *Technical Report dapper-2010*-1, Google, April.

Sloman, M. 1994. *Network and Distributed Systems* Management. Addison Wesley.

Son, L.V., Phuc, T.N.H. 2010. Researching on an online monitoring model for large-scale distributed systems. *Proc. of the 13th National Conference in Information and Communication Technology*, (Hungyen, Vietnam, 2010).

Thakuriah, P., N. Tilahun and M. Zellner (2015). Big Data and Urban Informatics: Innovations and Challenges to Urban Planning and Knowledge Discovery. In Proc. of NSF Workshop on Big Data and Urban Informatics, pp. 4-32.

Toporkov V., Toporkova A., Tselishchev A., Yemelyanov D., Potekhin P.. 2015. Metascheduling and heuristic co-allocation strategies in distributed computing. *Computing and Informatics*, Vol. 34 (1), pp. 45–76.