สัญญาเลขที่ R2560C059

รายงานฉบับสมบูรณ์

ชื่อโครงการ รูปแบบของระบบบันทึกเหตุการณ์ในสภาพแวดล้อมของการประมวลผลแบบกลุ่มเมฆ

ชื่อหัวหน้าโครงการ วินัย วงษ์ไทย

หน่วยงานที่สังกัด ภาควิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ มหาวิทยาลัยนเรศวร

เดือน ปี ที่เสร็จโครงการ

กุมภาพันธ์ 2561

สัญญาเลขที่ R2560C179

รายงานฉบับสมบูรณ์

ชื่อโครงการ **<u>รูปแบบของระบบบันทึกเหตุการณ์ในสภาพแวดล้อมของการประมวลผลแบบกลุ่มเมฆ</u>**

คณะผู้วิจัย	สังกัด
1. ดร.วินัย วงษ์ไทย	ภาควิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ มหาวิทยาลัยนเรศวร
2. ผู้ช่วยศาสตราจารย์ ดร.จักรกฤษณ์ เสน่ห์นมะหุต	ภาควิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ มหาวิทยาลัยนเรศวร
3. ผู้ช่วยศาสตราจารย์ ดร.เกรียงศักดิ์	ภาควิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ มหาวิทยาลัยนเรศวร

สนับสนุนโดย

งบประมาณรายได้ มหาวิทยาลัยนเรศวร ปีงบประมาณ 2560

Logging System Patterns for Infrastructure as a Service Cloud

1. Executive Summary

Thus this research provided approaches for providing spectrum of patterns for describing how to construct logging systems with varying characteristics. For developers, when building a logging system, the knowledge of characteristics of this system could assist them to get the right design of the system with minimal effort and time commitments. This project intends to define and identify logging system patterns.

2. Abstract (ไทย)

วิจัยนี้ได้เสนอวิธีการในการวิเคราะห์เพื่อสร้างรูปแบบของระบบบันทึกเหตุการณ์ในการประมวลผลแบบกลุ่มเมฆ ซึ่งวิธีการดังกล่าวประกอบด้วยหกกิจกรรมในการสร้างรูปแบบดังกล่าว ซึ่งวิธีการดังกล่าวนี้ไม่เคยนำเสนอที่ใดในงานวิจัยก่อนหน้านี้

3. Abstract (English)

Accountability is one of the keys to the mitigation of risks associated with cloud security. A logging system is an important feature in accountability solutions to anticipate and handle threats in the cloud. However, previous accountability with logging system solutions have been provided without any description of the logging system in the context of a design pattern of the system's components. There are a number of benefits in applying the design steps for patterns customized from the object-oriented software design and development area. Stating a design pattern of the logging system's components also facilitates the analysis of a logging system in terms of, for example, their

quality or characteristics. This can minimize the effort and time commitment necessary for a system's design and development. However, to define and identify a pattern needs appropriate approach, which this paper provides. To the best of our knowledge, the approach have not previously been described in the literature.

4. เนื้อหางานวิจัย

1 Introduction

Leitersdorf and Schreiber [1] state that cloud security is one of five major cybersecurity market trends which will define the cybersecurity budgets of firms for 2015. The Top Threats to Cloud Computing Report [2] by Cloud Security Alliance (CSA) provides examples of threats to cloud security. Accountability can be key to mitigating the risk of these threats. Many researchers [3, 4, 5, 6] have argued, in relation to cloud security, that cloud behaviors are open to inspection by any party for both legitimate or illegitimate purposes. [7, 8, 9, 10, 11, 12] argue that a logging system is an important feature in accountability solutions to assist in dealing with these problems and threats. [12] also states that a logging system is composed of logging processes which focus on logging-related tasks together with log files used to store contents produced by these processes.

However, previous accountability with logging systems solutions [12, 13, 14, 15, 16, 17, 18, 19, 20, 7, 11] have been provided system architectures for the specific logging system without any description of the design pattern upon which the system's components are based. We have identified something in the order of 90 or more such architectures. We have also identified common components within this variety of system architectures, leading us to develop a pattern-based approach to logging system design. The IaaS environment is complex and complicated and each developer designs their architecture without any example or standard to emulate; there are none. Thus, this paper proposes pattern definitions with structures based on those now widely accepted in the OO development environment. We state 6 activities to be used as tools to identify appropriate patterns. Following this, we will state and publish at a later time, a set of patterns structured according to these definitions.

Summary of contributions: the first contribution is a discussion of the need for defining and identifying logging system patterns for logging system design in the IaaS context, together with an associated development environment. The second contribution we are making is a statement of an approach to the definition and identification of the proposed logging system patterns. We believe that our approach can be used to identify and state usable and useful patterns for logging system design and development in the IaaS context. This is our future work. A further advantage envisaged is that the patterns can be used not only to describe any particular logging system, in 'standard' terms, but also to compare different logging systems for their similarities and differences in their characteristics, components and functionality. This can be taken to the further step of identifying the absence of properties considered essential or necessary, and the advantages and disadvantages of the particular system. By applying the standards, the quality of the system can be more easily assessed. This will facilitate the design and development of logging systems, and minimize the effort and time commitment required for system design and development.

2 Background.

2.1. IaaS Architecture.

Fig. 1illustrates the architecture of an IaaS public cloud configuration, which comes from our previous work [12]. The provider side is any organization that publicly offers virtual machines or VMs to the customer. The customer can rent the VMs and access them via the Internet. A hypervisor is software that enables one computer to have more than one VM. The dom0 is a privileged domain VM that is launched by the hypervisor during system boot, where 0 indicates that this VM is physically managed and owned by the provider. The dom0 directly accesses the hardware (hw) and manages multiple domUs. A domU is an unprivileged domain VM that runs on top of the hypervisor, U indicates that this VM is virtually managed and owned by a customer. A domU is a VM and IaaS product.

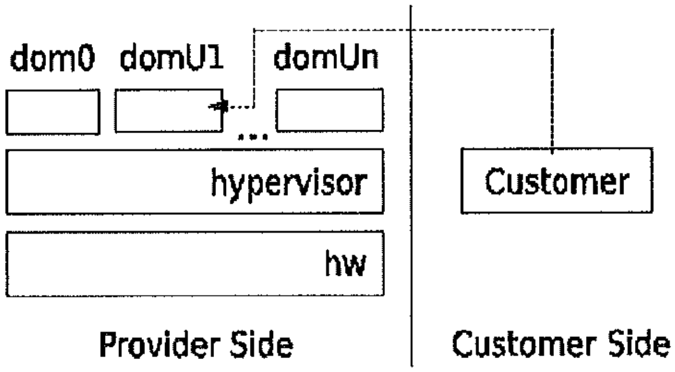

FIGURE 1. The IaaS architecture.

2.2. Generic logging Components of IaaS Cloud.

To describe the approach to defining and identifying logging system patterns for IaaS cloud configurations that exploits the generic logging components of an IaaS cloud configuration, needs an understanding of the definition of those generic components, according to [11]. [11], our previous work, provides information on these generic logging components, which we illustrate in Fig. 2.

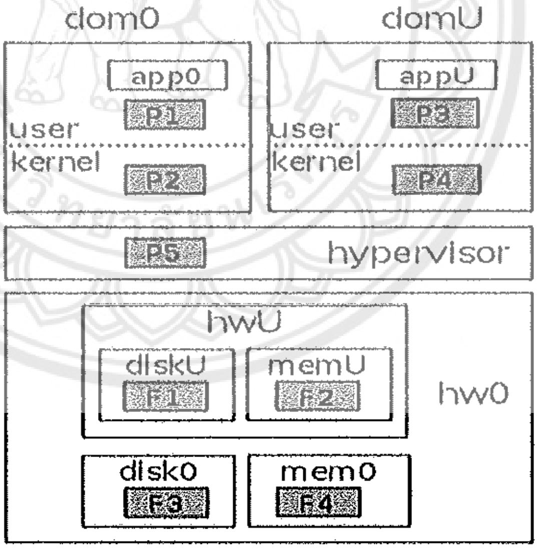

FIGURE 2. The overall view of generic logging components: logging process or Px (P1 to P5), and log files or Fy (F1 to F4).

This current paper further discusses these components which we group under two headings or as two sets, (1) IaaS components and (2) critical components.

The first set of IaaS components are shown as white boxes in Fig. 2. This set includes the hypervisor, dom0, domU, hw0, hwU, disk0, diskU, mem0, memU, app0, and appU. The first four components have already been discussed above. (Note that hw0 is hw in the previous discussion.) HwU is domU's hardware and physically located inside hw0 (which is owned by dom0) although it is virtually owned by domU. Disk0 is a physical disk of the hw0, and diskU is a virtual disk of a domU. Mem0 is the main memory of the hw0, and memU is the virtual main memory of domU. App0 is an application that runs inside dom0 and similarly appU is applications that runs the domU user level.

The elements of the second set are the critical components in the logging processes(Px, x=1,...,5) and log file components (Fy, y=1,...,4)(shaded boxes). The logging processes P1 to P5 perform logging-related tasks, and the log files F1 to F4 are used for storing the data produced by P1 to P5. Full details of P1 and P2 and P3 to P5 were fully described in our previous publications [12] and [11]. All the generic logging components, including and especially the critical components, will be referred to in the following sections in this paper in our descriptions of our approach to defining and identifying logging system patterns for IaaS.

2.3. The needs for defining and identifying Logging System Patterns for IaaS.

Gamma et al [21] discuss a number of benefits of design patterns, stating that "design patterns make it easier to reuse successful designs and architectures". By expressing proven techniques as design patterns makes them more accessible to developers of new systems and enables correct designs to be immediately produced. Design patterns help developers choose design alternatives promoting system and component reusability. As well, design patterns can improve documentation correctness and standardisation and reduce the maintenance cost and effort of existing systems by furnishing an explicit specifications of class and object interactions and their underlying intent. "Put simply, design patterns help a designer get a design right faster".

It is suggested here that design patterns in the logging context can bring the same benefits as enthusiastically discussed by Gamma et al., particularly promoting the reusability of designs and standardizing the development of logging systems. Our ultimate intention is to provide logging design patterns and make them more accessible to developers of new logging systems, thereby enabling these designers to better choose between design alternatives, as well as the other benefits of standard, and complete, documentation and support for the on-going maintenance function.

Our design patterns will provide explicit specifications of the critical logging components (Px and Fy) with their locations in any IaaS infrastructure and describe the underlying intent of each component, especially the logging processes or P1 to P5 and log files or F1 to F4 which are the critical components of any logging system. Our perceived outcome is "building correct systems faster, the first time".

Learning from the object-oriented area, design patterns have also been applied in other environments [22]. This extension of the concept and use of design patterns encouraged our work to apply the design pattern concept to logging systems' design and development in IaaS as well, which we elaborate in the next section.

3. The Approach to Define and Identify Logging System Patterns for IaaS Cloud.

The previous section discussed the need for defining and identifying logging system patterns for IaaS. This section discusses our approach to defining and identifying these logging system patterns. This approach has three steps which are discussed below in sub-sections 3.1 to 3.3. The first step was to identify and define the general concept of design patterns, and more specifically as they are known in object-oriented software design. We then consider our view of the well-found linkage between object-oriented software design patterns and the application of the concepts in logging system design and development, which is the second step described in sub-section 3.2. Finally, Step 3, elaborated in sub-section 3.3 includes the six activities to be used as tools to identify the patterns for logging system design and development in IaaS.

3.1. Design Patterns in general and in object-oriented software design.

A design pattern is generally described in [22] as "a documented best practice or core of a solution that has been applied successfully in multiple environments to solve a problem that recurs in a specific set of situations."Based on simple and elegant solutions to specific problems, a significant number of design patterns for object-oriented software design were created and published by [21]. Specific design patterns were al so discussed in [23] and termed software design patterns, and defined as: descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context. [22] states that patterns typically have the attributes of Name, Purpose, Description of when and why to apply the pattern, Structural diagrams, Examples of use, and Discussion of interactions with other patterns.

3.2. Defining a pattern in logging system design and development environment.

This section discusses what we consider to be a well-founded linkage between patterns in object-oriented software design and in logging design and development. For simplicity, we refer to a design pattern for logging system design an development in IaaS simply as a 'pattern'. In this sub-section we will define the definition of a pattern based on well-known design patterns in object-oriented software design and development created by Gamma et al [21], whose definition of design patterns, as previously cited, is:

descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.

For further discussions, and based on the generic logging components, our general definition pattern in logging system design and development environment is:

descriptions of participating critical components (Px and Fy) and their locations that are customized (the components can be appropriately located in IaaS components including domU, dom0, and hypervisor) to solve a general design problem in a particular context. Note that participating critical components are Px and Fy that are used to form a logging system. From the generic logging components in Fig. 2, the logging processes P1 to P5 and log files F1 to F4 are critical components of a logging system.

We believe that our patterns can be defined based on the definition discussed above. For example, Fig. 3. is an architecture of a logging system. Based on our investigation in [11], this architecture is only one of the 93 possible logging system architectures we have previously identified, and which we discussed in [11]. The patterns can be used as a blueprint to create a concrete software architecture before building the software. In the logging system context, a concrete logging system architecture, which can be any one of the 93, can be derived from a pattern as well. The created patterns can then be used as tools to describe and model a logging system and to compare between logging system models in terms of the systems' characteristics and advantages and disadvantages. This should facilitate the design and development of the systems.

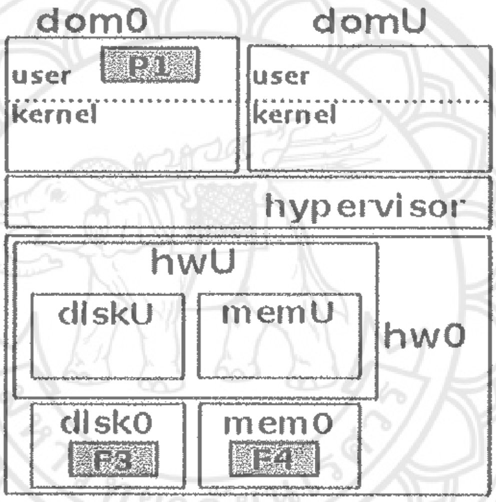

FIGURE 3. A logging system architecture deploying P1, F3, and F4

3.3. The activities to identifying patterns for logging system design and development.

This section discusses six activities to be undertaken to identify the patterns. Heer and Agrawala [23] identify software design patterns for information visualization based upon a review of existing frameworks and their own experiences building visualization software. We have followed their approach for our purposes. In our future work we will also identify specific patterns for IaaS logging environments based on our own experience of building logging systems particularly.

The first activity was to base the patterns on our own experience, that is, a retrospective analysis of our 'experience' from which Fig. 2 is derived, showing the generic logging components identified by us. We are leveraging on our own experience in building a prototype of logging systems. We also draw on the spamming case study from [12]. For mitigating risks associated with threats of malicious activities performed in consumers' virtual machines/VMs which can affect the security of both consumers and providers we cite [13], and in logging solutions to mitigate risks associated with security issues in platform as a service cloud models we have followed the discussion in [10]. Other experiences involving many aspects of logging systems such as the systems' performance [8, 9], especially the systems' quality [7] the systems' architectures [11], all of which gave us better understanding of logging systems in terms of the variety of architectures and the perceived and measurable quality of the various logging systems.

The second activity was to investigate and evaluate all possible forms of distribution of Px and Fy to form a logging system architecture, into either or both the customer side structure and the provider side structure.

Then we investigated and evaluated the distributions of Px and Fy in system architectures of the related work concerning logging systems, as the third activity in this pattern development process.

The fourth activity was to define the meaning and purpose of the logging data. The common abilities of logging systems to capture and store necessary logging data, will be expressed in the patterns we will define. We define the meaning and purpose of this logging data as, first, illustrating and recording the behavior or activities of a process or processes in domU Examples of which are discussed in [12] (the spamming case study). Alternatively, we consider the necessity of the data record of the domU files' life cycles. Examples and discussion of a domU file's life cycle from creation, through accessing and updating, and ultimately destruction of the files, are found in in [16] and in [14].

The fifth activity is the identification of a pattern which is based on the idea that we need a logging system that can be developed with in the shortest time possible and with minimal efforts. Simplicity and cost are two factors of importance in this regard. Secondly, the pattern must be designed in such a way as to ensure the reliability of Px and Fy, and includes the system's ability to facilitate the self-enforcement of its internal security policies. Last, but equally important, is the capability of capturing as much of the necessary logging data as possible, primarily as evidentiary data in any legal proceedings that may eventuate.

In the object oriented software development area, [21] describe their well-known design patterns using a consistent format. The final activity in our process is to emulate this approach in describing our patterns. The elements of the format of patterns described in [21] are: pattern name, intent, motivation, applicability, structure, participants, collaborations, consequences, implementation, known uses, and related patterns. Using this format for our patterns results in a clear and complete description of any pattern.

These six activities in the process of developing the patterns will result in useful and applicable logging system patterns for the IaaS environment, bringing with it an essential level of standardization, and understanding. In following this process, and carrying out these activities fully and in order, we believe that we can identify appropriate patterns for logging system design and development. The actual development of these patterns will be the subject of our future work.

4. Conclusions.

There are a number of benefits of this pattern based approach will deliver the same benefits for logging system design as have been experienced in object oriented software design and development. To achieve these benefits, this paper discusses the need for defining and identifying logging system patterns, and describes an approach to defining and identifying these logging system patterns, as an essential precursor to the actual definition of the patterns. We have drawn on what we might term the Reference Discipline of the pattern-based approach to object-oriented software design, and our own experience in developing logging systems in the IaaS environment.

We are confident that this approach can result in appropriate and well-defined patterns which can provide guidance and standardization for the future development of propriety logging systems, and the basis for analyzing the completeness of any such system. It will further provide the ability to analyze and identify the advantages and disadvantages inherent in any particular logging system design. This is our future work.

5. ภาคผนวก

a. บทความสำหรับการเผยแพร่

i. ตีพิมพ์ผลงานแล้ว คือ :Wongthai , W., van Moorsel , A. (2018). An Approach to Defining and Identifying Logging System Patterns for Infrastructure as a Service Cloud.ICIC Express Letters. Vol.12, No.กิจกรรมที่เกี่ยวข้องกับการนำผลจากโครงการไปใช้ประโยชน์

b. ตารางการเปรียบเทียบวัตถุประสงค์ กิจกรรมที่วางไว้ และกิจกรรมที่ดำเนินการมาและผลที่ได้รับตลอดโครงการ

กิจกรรมที่วางไว้	กิจกรรมที่ดำเนินการมาและผลที่ได้รับตลอดโครงการ
1. To investigate design patterns in object-oriented software design and development area.	obtained investigated design patterns in object-oriented software design and development area
2. To analyze the possible architectures	obtained analyze the possible architectures
3. To map 1 and 2 to define logging system patterns for logging system design and development area	proposed definition of logging system patterns for logging system design and development area
4. To identify and propose logging system patterns for logging system design and development area	Obtained an approach to identify and propose logging system patterns for logging system design and development area
5. To discuss the proposed logging system patterns	discussed the proposed logging system patterns
6. Publishing the project's results in a journal which is appeared in Scopus database	The paper was published.

7. Writing the project summary report	wrote the project summary report

c. Acknowledgment

Many thanks to Mr. Roy Morien of the Naresuan University Language Center for his editing assistance and advice on English expression in this document.

d. References

[1] Y. Leitersdorf and O. Schreiber. (2014) Cybersecurity hindsight and a look ahead at 2015. TechCrunch. [Online]. Available: http://techcrunch.com/2014/12/28/ cyber-security-hindsight-2020-and-a-look-ahead-at-2015/

[2] CSA, "The notorious nine: Cloud computing top threats in 2013," The Cloud Security Alliance (CSA), Tech. Rep., 2013

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[4] A. Haeberlen, "A case for the accountable cloud," SIGOPS Oper. Syst. Rev., vol. 44, no. 2, pp. 52–57, Apr. 2010.

[5] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in Proceedings of the 2009 conference on Hot topics in cloud computing, 2009.

[6] J. Lyle and A. Martin, "Trusted computing and provenance: better together," in Proceedings of the 2nd conference on Theory and practice of provenance, 2010.

[7] W. Wongthai and A. van Moorsel, "Quality analysis of logging system components in the cloud," Lecture Notes in Electrical Engineering, 2016. [8] W. Wongthai and A. Van Moorsel, "Performance measurement of logging systems in infrastructure as a service cloud," ICIC Express Letters, 2016. [9] P. Chan-In and W. Wongthai, "Performance improvement considerations of cloud logging systems," ICIC Express Letters, 2017.

[10] ——, "Logging solutions to mitigate risks associated with security issues in platform as a service cloud models," Information (Japan), 2016.

[11] W. Wongthai and A. Van Moorsel, "Logging system architectures for infrastructure as a service cloud," Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 2017. [Online]. Available:

[12] W. Wongthai, F. L. Rocha, and A. van Moorsel, "A generic logging template for infrastructure as a service cloud," in Proceedings of the 2013 27th International Conference on Advanced Information Networking and Applications Workshops. IEEE Computer Society, 2013.

[13] W. Wongthai, F. Rocha, and A. v. Moorsel, “Logging solutions to mitigate risks associated with threats in infrastructure as a service cloud,” in Proceedings of the 2013 International Conference on Cloud Computing and Big Data, ser. CLOUDCOM-ASIA ’13. IEEE Computer Society, 2013.

[14] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, “Trustcloud: A framework for accountability and trust in cloud computing,” IEEE Congress on Services, pp. 584–588, 2011.

[15] A. Haeberlen, P. Aditya, R. Rodrigues, and P. Druschel, “Accountable virtual machines,” in Proceedings of the 9th USENIX conference on Operating systems design and implementation, 2010.

[16] P. Macko, M. Chiarini, and M. Seltzer, “Collecting provenance via the Xen hypervisor,” in 3rd Workshop on the Theory and Practice of Provenance, June 2011.

[17] B.Dolan-Gavitt,B.Payne,andW.Lee,“Leveragingforensictoolsforvirtualmachineintrospection,” Georgia Institute of Technology, Tech. Rep., 2011.

[18] B. Payne, M. de Carbone, and W. Lee, “Secure and flexible monitoring of virtual machines,” in Annual Computer Security Applications Conference, 2007.

[19] B. Payne, M. Carbone, M. Sharif, and W. Lee, “Lares: An architecture for secure active monitoring using virtualization,” in IEEE Symposium on Security and Privacy, 2008.

[20] S. Sundareswaran, A. C. Squicciarini, and D. Lin, “Ensuring distributed accountability for data sharing in the cloud,” IEEE Transactions on Dependable and Secure Computing, 2012.

[21] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable ObjectOriented Software. Addison-Wesley Professional, 1994.

[22] P. Kuchana, Software Architecture Design Patterns in Java. Boston, MA, USA: Auerbach Publications, 2004.

[23] J. Heer and M. Agrawala, “Software design patterns for information visualization,” IEEE Transactions on Visualization and Computer Graphics, 2006.