PRIORITIZING AND FULFILLING QUALITY ATTRIBUTES FOR VIRTUAL LAB DEVELOPMENT THROUGH APPLICATION OF FUZZY ANALYTIC HIERARCHY PROCESS AND SOFTWARE DEVELOPMENT GUIDELINES

Chun Yong Chong[1], Sai Peck Lee[2], Teck Chaw Ling[3]

[1, 2, 3] Faculty of Computer Science and Information Technology, University of Malaya, 50603 Lembah Pantai, Kuala Lumpur, Malaysia

cychong@um.edu.my[1], saipeck@um.edu.my[2], tchaw@um.edu.my[3]

ABSTRACT

The evolution in pedagogy and research has introduced new requirements in the educational sector. On-demand computing resource provisioning, often referred as virtual lab, is one of the requirements in great demand. Most current virtual lab systems are proprietary and thus their detailed software architectures are not accessible to developers. The lack of transparency in virtual lab architecture makes it hard for educational institutions to perform experimental testing and prototyping on these systems to determine their effectiveness in meeting certain service level requirements. To lead to an effective virtual lab development, it is important to ensure that all non-functional requirements, or commonly known as quality attributes, stated in the service level agreement (SLA) are realizable in the required order of priority. While a quality attribute can be achieved by applying one or more software development guidelines, software development guidelines may have different effects on different quality attributes and may also conflict with one another. In addition, priority assessment of the quality attributes are needed in order to focus on the higher priority ones while ensuring that the bare minimum expectation of the remaining ones are attainable. This paper aims to apply fuzzy Analytic Hierarchy Process (AHP) during the pre-negotiation stage of SLA to identify quality attributes and rank them based on their priorities. Based on the results, a set of suitable development guidelines are chosen to help realize and achieve the prioritized quality attributes in the virtual lab architecture, thus paving a way to facilitate architecture evaluation. The application of fuzzy AHP in our experiment has shown that stakeholders ranked reliability, usability, efficiency, security, maintainability, and portability in decreasing order of priority, based on which a set of suitable, non-conflicting software development guidelines were determined. The final result can provide a promising reference model for better understanding of virtual lab in the educational sector.

Keywords: quality attributes, priority assessment, fuzzy analytic hierarchy process, software development guideline, virtual lab

1.0 INTRODUCTION

On-demand resource provisioning in the educational sector has changed the way how we perceive computer laboratories. A physical computer laboratory is replaced by virtual machines, which can be accessed by anyone, from anywhere, and at any time. The virtual machine specifications are tailored based on different users' hardware and software requirements. There are several initiatives from higher educational institutions that have successfully integrated the virtual lab concept [1]. Specifically, the Virtual Computing Lab (VCL) system developed by North Carolina State University (NCSU) [1] is aiming at providing on-demand cloud computing services through efficient use of available hardware resources. Users are able to access varieties of computational, storage, and software resources based on their demands.

On-demand resource provisioning is capable of providing numerous benefits to higher educational institutions. Firstly, universities are able to share, provision and aggregate computational power, storage capacities, network resources, and software licenses on-demand. These benefits are achievable through the virtualization technology. Virtualization consolidates all the underlying hardware and creates a unified resource pool (including computing power, storage, and network) where virtual machines are created based on an on-demand basis. Virtual machines are provisioned to the end users and are highly customizable based on the users' requirements. They can be loaded with different operating systems easily. This feature allows students and researchers to perform system testing on different platforms without the hassle of getting a few physical machines. Furthermore, users can configure the specification of the virtual machines including processor speed, storage capacity, and network

bandwidth based on their needs. Moreover, a virtual machine can be easily disabled when it is not needed. The computing power can be returned to the unified resource pool to be reused by others. Users can access to the virtualized resource pool anywhere and at any time as long as there is a stable Internet connection.

Despite the fact that virtual lab has been shown to be beneficial in the educational sector, there are very limited work that focus on the concern of prioritizing quality attributes and balancing their trade-offs in meeting the service level requirements of such a cloud computing environment [2]. Quality attributes often push and constrain the architectural design of a software system in order to ensure that those qualities can be fulfilled in the finalized software system [3]. Software architecture is a vital resource in any software development project because it is the blueprint which dictates the structure, behavior, and relationships between software components. Therefore, software quality and architecture should be analyzed and studied before valuable resources are committed [4]. Until now, there is still no reference or well-established software architectural design that is deemed effective in any problem domain which can be adopted by software developers to lead to reliable software development. In addition, the idea of adopting an established software architecture of virtual lab cannot be applied because such a virtual lab is always proprietary. Besides this, software architectures of existing open-source virtual lab systems are not standardized. Thus, performing experimental testing, system prototyping, and assessing the suitability of virtual lab systems become a difficult task for early adopters.

Standardizing software architecture is an open issue in the field of software engineering that has been actively discussed [5]. In this paper, we aim to provide an alternative solution to the problem of lack of standardized software architecture through an approach by reaching an agreement of service level requirements among all stakeholders over an SLA document for virtual lab development. Through the SLA, a list of quality attributes that are expected to be delivered in the final product can be identified. The software architecture can then be designed based on these quality attributes. If several candidate architectures are to be nominated, architecture evaluation technique can be imposed to choose the most appropriate architecture which reflects the quality requirements of virtual lab system. However, to fulfil all the quality attributes under limited budget and time constraints without any trade-off is impossible. Thus, a methodical way of prioritizing quality attributes and improving the fulfilment of these attributes can help remedy the inadequacy of standardized software architecture for virtual lab development. To help strengthen the fulfilment of quality attributes, we may apply software development guidelines. However, these guidelines might have contradictory effects on different quality attributes and might even be conflicting on one another.

This paper aims to determine a list of prioritized quality attributes from the stakeholders and the weightage of each attribute using fuzzy AHP. Based on these results, software development guidelines are selected to help achieve the quality attributes. The selection of development guidelines is based on [6], where the effects of the development guidelines toward achievement of the related quality attributes, and the relationships among the selected development guidelines are emphasised. The prioritized quality attributes and chosen development guidelines can then be used to facilitate the architecture evaluation process in order to improve the confidence in decision making.

This paper is organized as follows: Section 2 discusses related work on priority analysis of quality attributes in a virtual lab environment as well as methods to achieve the fulfilment of these attributes. Section 3 presents the application of fuzzy AHP to perform priority assessment, and also development guidelines to improve the achievement of quality attributes. Section 4 discusses the results and contribution of the application of fuzzy AHP and software development guidelines. Finally, the conclusion along with the suggested future work is discussed in Section 5.

2.0 RELATED WORK

Quality attributes can be defined as the degree to which a particular software system fulfils stakeholders' requests regarding its functions and behavior [7]. The ISO/IEC 9126 Software Quality standard [8] defines quality attributes as the totality of software features and characteristics of a software product that can satisfy specified or implied stakeholder needs. ISO/IEC 9126 categorized quality attributes into six major criteria, namely Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability.

In order to ensure successful development and implementation of a particular software system, software developers need to comprehend and leverage the dependencies between quality attributes and software architecture [9]. If software developers can capture the essential quality attributes from stakeholders during early stages of software development, they can easily design a software architecture that is capable of satisfying those attributes. Based on the proposed software architecture, software developers can then build prototypes and perform incremental system testing to fine-tune and evaluate the system. Thus, eliciting quality attributes and evaluating the captured attributes are crucial to ensure the success of software development. These statements are even more truthful in the case of virtual lab system because the growth of virtual lab is still in its early stage and there are very limited established references and guidelines.

Without a proper reference, software developers can only depend on their own intuition and experiences to design the software architecture of virtual lab system. If a systematic way to analyze the benefits and trade-off of different software architecture designs can be implemented, this may ensure that the chosen software architecture can always reflect the quality requirements of virtual lab system. Architecture evaluation is one such technique used to evaluate the suitability of a chosen software architecture. A number of architecture evaluation techniques have been proposed, such as Architecture Trade-off Analysis Method (ATAM) [10] and Architecture-Level Maintainability Analysis (ALMA) [11] to perform evaluation during the architectural decision making process. A typical decision making process in the early phase of software development consists of the following activities: problem identification; problem analysis and solution development; selection and evaluation of solution [12]. Architecture evaluation often focuses on the selection and evaluation activities to help improve the confidence in decision making.

In a perfect environment where time and resources are not a limiting factor, stakeholders could demand for a virtual lab system to achieve perfect results in each and every aspect of quality attributes. However, this situation can never be achieved in real life. To fulfil all users' requirements from a service provider's point of view is almost impossible, and therefore, a balance needs to be achieved. As suggested by [13], there should have a way to prioritize quality attributes so that software engineers can focus on achieving higher-priority attributes within a limited time frame. Thus, requirements engineering plays an important role to achieve an agreement among stakeholders upon the expected level of quality attributes to be achieved in the final product. An SLA is one of the outcomes based on requirements engineering [14-17]. The SLA dictates the agreement signed between a service provider and the stakeholders with regards to the quality of services provided.

However, due to the fact that virtual lab is still at its infancy, there is a lack of well-established SLA framework. The work by [18] introduces a conceptual SLA framework for on-demand resource provisioning based on cloud computing environment, focuses on availability, scalability, cost, flexibility and security. The authors presented several scenarios that can be applied to the cloud computing environment when negotiating SLA with cloud providers. Meanwhile, the work by [19] proposes a mechanism for managing the SLA in a cloud computing environment. The authors found that there are no universal sets of SLA metrics that can be implemented across different cloud providers. This is partly due to different kinds of services provided by different providers and the diversity in local legislation. While numerous SLA frameworks have been suggested in cloud computing, few approaches address the prioritization and fulfilment of quality attributes for the SLA in a virtual lab environment.

The work by [20] has shown that potential SLA violations and conflicts are more likely to be identified if software developers can analyze the SLA of cloud service providers and matching them against stakeholders' requirement in the pre-negotiation stage of the SLA. Without reaching a consensus among stakeholders, designing software architecture for virtual lab becomes difficult because software developers cannot comprehend the expectations from stakeholders. Therefore, capturing quality attributes for including in the SLA and ensuring that these attributes can be achieved before delivering the final software system is important.

There are typically two challenges in analyzing and fulfilling quality attributes stated in the SLA. Firstly, software requirements are usually imprecise and defined qualitatively when expressed in verbal form [21]. The usage of imprecise or subjective terms might introduce ambiguities in the requirements which can lead to bad software design. Stakeholders often specify their requirements verbally, making the requirements capturing process less effective. Besides that, quality attributes often affect with each other [22]. For example, designing a system that is extremely secure will inevitably cause negative impact on the performance on the software system. This dilemma has to be identified in the early stage of system design so that stakeholders are aware of the trade-off.

Priority analysis is one of the techniques used to resolve trade-off between conflicting quality attributes [22]. Given a set of quality attributes, priority analysis works by prioritizing and ranking these attributes according to the context given by stakeholders. The relative priority and importance of the given quality attributes plays a significant role, not only allow software developers to focus on the higher-priority ones, but also ensure that the delivered system can attain a certain level of quality. However, to directly assess the priorities of all quality attributes from stakeholders is often difficult because human judgments and preference are often vague and cannot be estimated easily with exact numerical values [23].

The nature of prioritizing quality attributes is a multi-criteria decision-making (MCDM) problem. MCDM is a research of methods and procedures concerning about evaluating multiple conflicting criteria and deriving a way to come out with a trade-off resolution. The given set of evaluation criteria often differ in the degree of importance. Several approaches have been proposed in the area of requirements prioritization to help select and prioritize requirements in software development projects [24], such as AHP [25], Cost-Value [26], and Binary Search Tree [27]. In general, these techniques are applied to devise a set of goals and constraints. The selection of prioritization techniques mainly depends on the domain of problem, the type of input (qualitative or quantitative) and the size of input. The work by [28] applied several requirements prioritization techniques on a small telephony system to evaluate their performance and effectiveness. The author found that AHP is one of the most reliable techniques although it uses more time and does not scale very well with a large number of prioritization criteria. If the evaluation criteria for the MCDM problem are relatively small, the results of AHP can easily outperform other prioritization techniques.

AHP works by taking judgmental inputs from a group of stakeholders and constructs a hierarchy of criteria according to their priorities. The hierarchy of criteria enable all stakeholders to visualize the problem systematically. Additional criteria can also be added based on the input from stakeholders in order to revise the hierarchical structure of the AHP model. The simplest form of prioritizing a particular criterion in AHP is based on a nine-point scale, which expresses preferences between options ranging from equal importance to extreme importance. AHP has been proven to be one of the dominating tools in solving the MCDM problem because it can be integrated with other techniques like linear programming and fuzzy logic easily [29].

Regardless of its popularity, AHP is less effective when handling imprecision and uncertainties associated with human judgment. Natural language tends to be ambiguous and vague. Verbal terms such as “very low”, “low”, “balance”, “high”, and “very high” are easier to describe the desired value and weight of criteria. The fuzziness and ambiguity in the process of AHP can be dealt with using the fuzzy set theory [23]. Fuzzy set theory is capable of accepting crisp input, or in this case, uncertain judgements from stakeholders. Based on the crisp inputs, fuzzy set theory is applied to determine the degree to which these inputs belong to the correspondent fuzzy sets. Then, a defuzzification process which produces a quantifiable result normally in the form of numerical values is applied. By integrating fuzzy set theory, AHP is capable of handling the fuzziness of the data involved in decision making efficiently. However, priority analysis is only the prior work before fulfilling the associated quality attributes.

The next challenge is to attain the stated level of quality defined by the stakeholders in the SLA. Due to the importance of quality attributes, numerous approaches have been proposed by researchers to improve the achievement of quality attributes. One of the popular methods to improve the achievement of quality attributes is the preventive approach. In this approach, the software architecture design phase is conducted in a formal and systematic way to prevent defects associated with quality attributes. Examples of preventive approach include usage of design patterns, software architectural styles, and adherence to other established software development guidelines. This approach helps in reducing effort for system testing and maintenance while resulting in more reusable and higher quality software [30].

Several works [31-33] in the domain of preventive approaches have been introduced to improve the achievement of quality attributes in software systems. However, these approaches focus on individual quality attributes, whereas software systems usually involve multiple attributes. Meanwhile, the work by [34] introduced an approach for evaluating and improving quality attributes by defining soft goals. The authors introduced a method to reject candidate software development guidelines that negatively affect higher-priority quality attributes.

There are mainly three types of software development guidelines, namely design patterns, architectural styles and best practices [35]. Architectural style is normally applied during the early design stage while best practices can be applied throughout the whole software development process if applicable. However, each guideline might have positive or negative effect on certain quality attributes. For example, peer-to-peer architecture style improves the performance and availability of the system but has negative impact on security and maintainability.

Based on the discussion above, it is clear that to successfully develop a virtual lab system, eliciting and prioritizing quality attributes during pre-negotiation stage of SLA is crucial. This context is even more truthful in the case of virtual lab environment because there are very limited references and well established software architectural design available. The lack of references prevents educational institutions to go for early stage prototyping and software architecture evaluation. Even if one chooses to go for an open-source virtual lab solution, customization of the functionalities and software modules are needed to cater for different institutional needs. Without prioritizing and negotiating a proper SLA framework between cloud providers and stakeholders, one might risk delivering a low quality system. As a result, this paper aims to apply fuzzy AHP to capture and prioritize quality attributes from stakeholders during the pre-negotiation stage of SLA. The result of fuzzy AHP will be used as the input to discover complementary software development guidelines based on the work by [6]. The combination of both techniques can be used to facilitate the architecture evaluation process, especially in the problem analysis and solution evaluation stages by revealing how well a chosen architecture satisfies the quality requirements in virtual lab system.

3.0 PRIORITY ASSESSMENT OF QUALITY ATTRIBUTES AND IDENTIFICATION OF DEVELOPMENT GUIDELINES

This section is divided into two parts. Firstly, fuzzy AHP method is introduced to perform priority assessment of quality attributes for the development of virtual lab. Following that, a set of suitable development guidelines that complement the result of priority analysis are identified using the method proposed by [6].

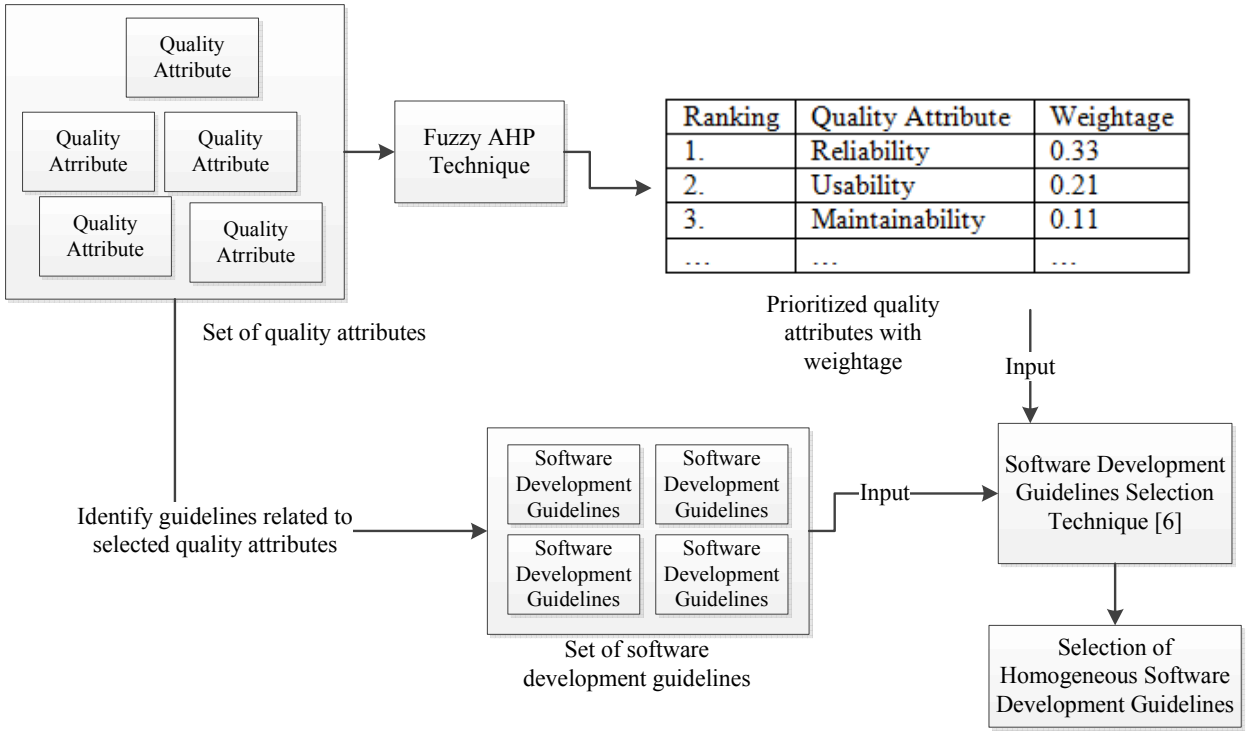

Fig. 1: Workflow of prioritization and fulfilment of quality attributes in a virtual lab environment

Fig. 1 shows the overall workflow in this paper to help improve the achievement of quality attributes for virtual lab development. Firstly, a set of quality attributes associated with the development of virtual lab system are identified. Based on the identified quality attributes, development guidelines which complement those attributes are selected. Next, fuzzy AHP is used to prioritize, rank, and weight the selected quality attributes based on the feedback from stakeholders. The result gathered from the application of fuzzy AHP can act as a baseline model to design the software architecture of the virtual lab system. Quality attributes are ranked according to their priority along with their corresponding weightage. Subsequently, the result of application of fuzzy AHP and the selected development guidelines are used as the input for the development guidelines selection technique introduced by [6]. Finally, a homogeneous set of software development guidelines are identified to help improve the overall achievement of quality attributes for virtual lab development. The definition of homogenous software development guidelines will be explained in detail in Section 3.2.

3.1 Fuzzy Analytic Hierarchy Process

Fuzzy AHP is chosen in this paper because it is capable of handling ambiguous judgmental inputs given by the stakeholders. Fuzzy AHP is also capable of converting qualitative inputs into quantitative results, in the form of weightage and ranking. The weightage and ranking of quality attributes can easily help stakeholders to analyze trade-off and choose the set of software development guidelines that are complementary to the associated quality attributes.

In both traditional and fuzzy AHP, the process starts by modeling a hierarchy of decisions based on the problem domain. The top of the hierarchy consists of the goal for conducting the test, followed by a group of possible choices to achieve that particular goal. The choices can be further divided into sub-criteria if required. The problem domain for this paper is on virtual lab environment for the educational sector. The selection of evaluation criteria is based on ISO/IEC 9126 Software Quality standard [8]. Functionality is not involved because it is expressed as a totality of essential functions that the software product provides. As for the remaining quality attributes, they can only be measured when the functionality of a given system is present. Thus, fulfilment of functionality becomes the pre-requisite for the fulfilment of the remaining attributes. In this paper, we assume that the functionality of the virtual lab system would be fully satisfied before evaluating the fulfilment of the remaining attributes. However, the ISO/IEC 9126 standard does not consider security as one of the major criteria. Numerous researches [36-38] related to cloud computing have indicated that security is one of the main concerns for systems deployed in a cloud computing environment. This is because most cloud-based storage is shared by multiple users and users have limited control over the physical location of their data. Thus, in this paper, the evaluation criteria of quality attributes are extended to fit into the context of security in a cloud-based system. All in all, there are a total of six evaluation criteria. As indicated earlier, the research by [28] has proven that AHP is one of the best prioritization techniques in a small-scale MCDM problem, such as the one presented in this paper. Fig. 2 shows an illustration of AHP hierarchy model for a virtual lab system.

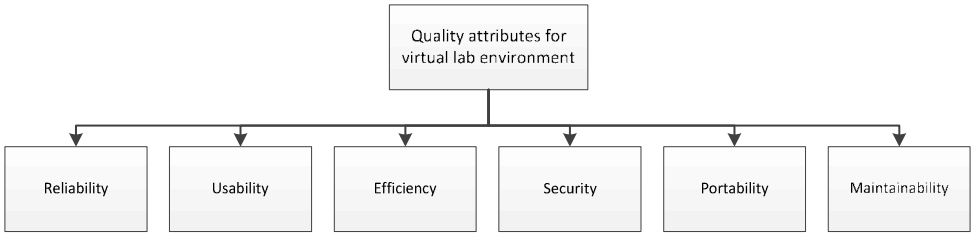

Fig. 2: AHP Hierarchy modeling of virtual lab

A pair-wise comparison among all the possible choices is conducted to justify the importance between them. Each choice is associated with a weightage in order to reflect the priority of the choice toward the ultimate goal. Then, stakeholders perform a pair-wise comparison and give weightage using a nine-point scale ranging from 1-9, where a greater value represents higher importance. In order to reach a consensus among the stakeholders, triangular fuzzy number (TFN) is used. TFN is capable of aggregating the subjective judgments of all the

stakeholders through fuzzy set theory. Fig. 3 shows an example of TFN denoted as (*L,M,H*) which represents the highest possible value, most ideal value, and lowest possible value respectively.

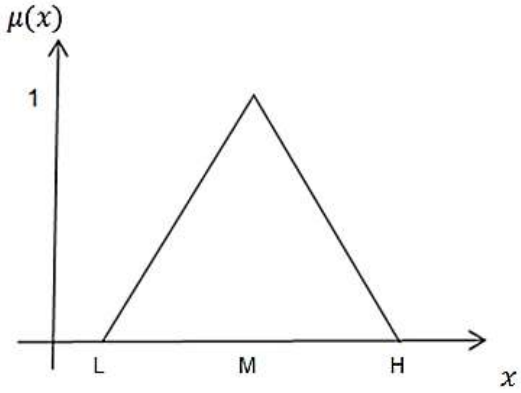

Fig. 3: Triangular fuzzy numbers

The triangular fuzzy number T_{xy} is constructed using the following formulas:

$$T_{xy} = (L_{xy}, M_{xy}, H_{xy}) \quad (1)$$

$$L_{xy}, M_{xy}, H_{xy} \in (1/9, 9) \quad (2)$$

$$M_{xy} = \sqrt[n]{J_{xya} \cdot J_{xyb} \cdot J_{xyc} \cdots J_{xyn}} \quad (3)$$

where *x* and *y* represent a pair of criteria being judged by stakeholders. Given *n* number of stakeholders, J_{xya} represents an opinion of stakeholder *"a"* toward the relative importance for criteria C_x- C_y.

Value M_{xy} is calculated based the geometric mean of stakeholders' scores for a particular comparison. The geometric mean is capable of accurately aggregating and representing the consensus of stakeholders [25]. L_{xy} and H_{xy} represent the lowest and highest scores respectively toward the relative importance for criteria C_x- C_y. After getting the TFN value for every pair of comparison, a fuzzy pair-wise comparison matrix is established in the form of *n x n* matrix. Table 1 illustrates an example of the matrix.

Table 1: Fuzzy pair-wise comparison matrix

$\tilde{F}_{xy} =$

	C_a	C_b	.....	C_n
C_a	*1*	p_{ab}	.....	p_{an}
C_b	$1/p_{ab}$	*1*	.....	p_{bn}
⋮	⋮	⋮	⋮	⋮
C_n	$1/p_{an}$	$1/p_{bn}$	.....	*1*

p_{ab} represents the TFN for the comparison between criteria C_a and C_b. Comparison between criteria C_b and C_a is the reverse of C_a and C_b, thus making the TFN value for C_b and C_a to be represented as $1/p_{ab}$. $\tilde{F}_{xy}$ denotes the TFN values derived from formulas (1)-(3).

The stakeholders who participated in this evaluation include university students and developers who have experiences in cloud computing development. However, not the selected participants have adequate knowledge in virtual lab environment. Thus, it is common to have some outliers where the judgemental inputs of these individuals might deviate a lot from the average group's judgments. These outliers are commonly referred as

inconsistent individual judgements in AHP. If the level of inconsistency is very high, one might need to revise the whole AHP process. The work by [39] found that consistency of AHP in group decision making is not a problem when the number of participants exceeds a threshold value and the geometric mean is used to aggregate individual judgments. The threshold value to achieve an acceptable level of inconsistency is twenty participants. This means that as long as the number of participants is equal or more than the threshold value, one can ensure that the aggregated results will be consistent and revision of judgements is not needed. Thus, twenty participants were chosen in this paper to ensure the consistency of results.

Each participant was instructed to perform pair-wise comparison and give their judgement based on the relative scores of 1-9. The triangular fuzzy numbers were established using formulas (1)-(3) and tabulated into a comparison matrix shown in Table 2.

Table 2: Fuzzy pair-wise TFN values for quality attributes in virtual lab environment

	Reliability	Usability	Efficiency	Security	Portability	Maintainability
Reliability	**1**	0.5, 1.488, 4	0.33, 1.675, 7	0.14, 1.304, 5	0.25, 1.16, 7	0.2, 0.861, 3
Usability		**1**	0.2, 1.062, 5	0.11, 1.508, 4	0.33, 1.13, 7	0.2, 0.861, 3
Efficiency			**1**	0.25, 1.16, 6	0.6, 1.203, 5	0.12, 0.98, 4
Security				**1**	0.14, 1.218, 6	0.12, 0.98, 4
Portability					**1**	0.11, 0.437, 3
Maintainability						**1**

Following the construction of comparison matrix, defuzzification take places to produce a quantifiable value based on the calculated TFN values. The defuzzification method adopted in this paper was derived from [40] as formulated in (4) which is commonly referred as the alpha cut. Alpha cut of a fuzzy set is the set of all elements which have its membership value greater than or equal to an alpha threshold value, represented by α. Alpha cut enables one to describe a fuzzy set as a composition of crisp sets. Crisp sets simply describe whether an element is either a member of the set or not. Formula (4) shows the algorithm of alpha cut.

$$\mu_{\alpha,\beta}\left(\tilde{F}_{xy}\right) = \left[\beta \cdot f_{\alpha}\left(L_{xy}\right) + (1-\beta) \cdot f_{\alpha}\left(H_{xy}\right)\right] \qquad (4)$$

and

$$0 \leq \alpha \leq 1, \qquad 0 \leq \beta \leq 1$$

such that $f_{\alpha}\left(L_{xy}\right) = \left(M_{xy} - L_{xy}\right) \cdot \alpha + L_{xy}$, which represents the left-end boundary value of alpha cut for $\tilde{F}_{xy}$. On the other hand, $f_{\alpha}\left(H_{xy}\right) = H_{xy} - (H_{xy} - M_{xy}) \cdot \alpha$ represents the right-end boundary value of alpha cut for $\tilde{F}_{xy}$.

α and β in this context carry the meaning of preferences and risk tolerance of stakeholders. These two values range between 0 and 1, in such a way that a lesser value indicates greater uncertainty in decision making. Since preferences and risk tolerance are not the focus of this paper, value of 0.5 for α and β is used to represent a balanced environment. This indicates that stakeholders are neither extremely optimistic nor pessimistic about their judgments.

The results from Table 2 were then synthesized using formula (4) with α and β at 0.5. An example of calculation is shown below for the pair Functionality-Reliability

$f_{0.5}(L_{Reliability-Usability}) = (1.488 - 0.5) \cdot 0.5 + 0.5 = 0.994$

$f_{0.5}(H_{Reliablity-Usability}) = 4 - (4 - 1.488) \cdot 0.5 = 2.744$

$\mu_{05,0.5}(\tilde{F}_{Reliablity-Usability}) = [0.5 \cdot 0.909 + (1 - 0.5) \cdot 3.244] = 1.86$

$\mu_{05,0.5}(\tilde{F}_{Usability-Reliability}) = 1/1.86 = 0.54$

Table 3 represents the result of fuzzy pair-wise comparison after defuzzification process.

Table 3: Aggregated fuzzy pair-wise comparison matrix using alpha cut method

	Reliability	Usability	Efficiency	Security	Portability	Maintainability
Reliability	1	1.86	2.67	1.94	2.4	1.2
Usability	0.54	1	1.83	1.71	2.44	1.2
Efficiency	0.37	0.55	1	2.2	2	1.51
Security	0.52	0.58	0.45	1	2.08	1.51
Portability	0.42	0.41	0.50	0.48	1	0.99
Maintainability	0.84	0.84	0.66	0.66	1.01	1

The next step is to determine the eigenvalue and eigenvector of the fuzzy pair-wise comparison matrix. The purpose of calculating the eigenvector is to determine the aggregated weightage of particular criteria. Assume that δ denotes the eigenvector while λ denotes the eigenvalue of fuzzy pair-wise comparison matrix $\tilde{F}_{xy}$.

$$[(\mu_{\alpha,\beta}(\tilde{F}_{xy}) - \lambda I] \cdot \delta = 0 \qquad (5)$$

Formula (5) is based on the linear transformation of vectors, where *I* represents the unitary matrix. By applying formulas (1)-(5), the weightage of particular criteria with respect to all other possible criteria can be acquired.

The eigenvectors of associated quality attributes for virtual lab were then calculated using formula (5).

$$[(\mu_{\alpha,\beta}(\tilde{F}_{xy}) - \lambda I] = \begin{bmatrix} 1 & 1.86 & 2.67 & 1.94 & 2.4 & 1.2 \\ 0.54 & 1 & 1.83 & 1.71 & 2.44 & 1.2 \\ 0.37 & 0.55 & 1 & 2.2 & 2 & 1.51 \\ 0.52 & 0.58 & 0.45 & 1 & 2.08 & 1.51 \\ 0.42 & 0.41 & 0.50 & 0.48 & 1 & 0.99 \\ 0.84 & 0.84 & 0.66 & 0.66 & 1.01 & 1 \end{bmatrix}$$

Multiplying eigenvalue λ with unitary matrix *I* produces an identity matrix that cancels out each other. Thus, the notation λI is discarded in this case. Applying formula (5) results in

$$\begin{bmatrix} 1 & 1.86 & 2.67 & 1.94 & 2.4 & 1.2 \\ 0.54 & 1 & 1.83 & 1.71 & 2.44 & 1.2 \\ 0.37 & 0.55 & 1 & 2.2 & 2 & 1.51 \\ 0.52 & 0.58 & 0.45 & 1 & 2.08 & 1.51 \\ 0.42 & 0.41 & 0.50 & 0.48 & 1 & 0.99 \\ 0.84 & 0.84 & 0.66 & 0.66 & 1.01 & 1 \end{bmatrix} \begin{bmatrix} \delta_{Reliability} \\ \delta_{Usability} \\ \delta_{Efficiency} \\ \delta_{Security} \\ \delta_{Portabililty} \\ \delta_{Maintainability} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \delta_{Reliability} \\ \delta_{Usability} \\ \delta_{Efficiency} \\ \delta_{Security} \\ \delta_{Portabililty} \\ \delta_{Maintainability} \end{bmatrix} = \begin{bmatrix} 0.275 \\ 0.201 \\ 0.168 \\ 0.136 \\ 0.090 \\ 0.130 \end{bmatrix}$$

The aggregated result in terms of weightage is tabulated in Table 4. The results obtained are ordered as follows: Reliability (0.275), Usability (0.201), Efficiency (0.168), Security (0.136), Maintainability (0.130), and Portability (0.090).

Table 4: Weightage and priority of selected criteria

Priority	Quality Attribute	Weightage
2	Reliability	0.275
3	Usability	0.201
4	Efficiency	0.168
5	Security	0.136
6	Maintainability	0.130
7	Portability	0.090

The results gathered from fuzzy AHP served as a baseline to model the system architecture for a virtual lab system. Therefore, software developers can easily identify the most important quality attributes and invest more efforts to ensure that those attributes are fulfilled in the final product. Next, a method to identify software development guidelines based on the results gathered in Table 4 is presented.

3.2 Identification of Complementary Guidelines to Improve Quality Attributes

The results from the priority assessment are then mapped with established software development guidelines that complement the associated quality attributes. The selection of guidelines are based on the technique proposed by Hneif and Lee [6]. These guidelines are categorized into three groups, namely architectural styles, best practices and design patterns.

In order to improve achievement of quality attributes, the selected software development guidelines should have two characteristics. Firstly, the guidelines must have positive effects on the higher-priority quality attributes. Typically, software development guidelines will have one or more effects on quality attributes. These effects can either be positive, negative or no effect toward the associated attributes. Mapping of relationship between software development guidelines and quality attributes is needed. These relationships can help in identifying conflict among the chosen development guidelines toward higher-priority quality attributes.

Secondly, the chosen software development guidelines should only have homogeneous relationships with each other. This implies that the selected guidelines should not have contradictory effects among themselves when applied together. The relationships among all chosen software development guidelines need to be identified in order to obtain a set of non-overlapping guidelines.

Using the comprehensive dataset of software development guidelines, one can identify a set of homogeneous development guidelines to help improve the achievement of quality attributes when developing the software architecture of a virtual lab system. The steps below describe the basic idea on how to produce a set of homogeneous guidelines:

Input : Quality attributes Output : Set of homogeneous guidelines that complement the associated quality attributes

1. For each required quality attribute:
    a. Collect the software development guidelines that positively affect the associated attribute.
    b. Reject any guideline that has negative effect on the higher-priority quality attributes.
    c. For two overlapping or conflicting guidelines of two different quality attributes:
        i. If the two quality attributes have different priorities, then the guideline that supports the lower-priority quality attributes is rejected
        ii. If the two quality attributes have the same priority, one of the development guidelines is rejected, according to the choice of the software engineer.
2. For two overlapping guidelines for the same quality attribute, one of them is rejected, according to the choice of the software engineer.

Repeat the same step for each required quality attribute.

3.3 Case Study

In this section, we describe the result based on a case study conducted to evaluate and identify the set of homogeneous software development guidelines for a virtual lab environment is illustrated.

The required quality attributes are extracted from Table 4. Reliability, Usability, Efficiency, Security, Maintainability, and Portability are used for evaluation in this example. Table 5 shows the effects of sample software development guidelines toward the chosen quality attributes. Table 6 shows the inter-relationship between the selected sample guidelines. Using the information gathered in Table 5 and Table 6, the dataset are applied into the software development guidelines technique introduced by [6].

Table 5: Effects of software development guidelines toward the chosen quality attributes

Guideline	Guideline Type	Quality Attribute					
		Reliability	Usability	Efficiency	Security	Maintainability	Portability
G1: Lazy initiation	Best Practice	No effect	Negative	Positive	No effect	No effect	No effect
G2: Single access point	Design Pattern	Negative	No effect	Positive	Positive	No effect	No effect
G3:Visibility	Best Practice	No effect	Positive	No effect	No effect	No effect	No effect
G4: Active redundancy	Best Practice	Positive	No effect	No effect	No effect	No effect	No effect
G5: Multi-tier	Architectural Style	No effect	No effect	Negative	Positive	Positive	No effect
G6: Prototype	Design Pattern	No effect	No effect	Positive	No effect	No effect	No effect
G7: Factory	Design Pattern	Positive	No effect	No effect	Positive	No effect	No effect
G8: Chain-of-Responsibilities	Design Pattern	Negative	No effect	No effect	No effect	Positive	No effect
G9: Model-View-Controller	Architectural Style	Negative	No effect	Negative	No effect	Positive	Positive

Table 6: Relationships between selected software development guidelines

	G1	G2	G3	G4	G5	G6	G7	G8	G9
G1									
G2	No relation								
G3	No relation	No relation							
G4	No relation	Conflicts	No relation						
G5	No relation	No relation	No relation	No relation					
G6	Conflicts	No relation	No relation	No relation	No relation				
G7	Conflicts	Comple ment	No relation	No relation	No relation	No relation			
G8	No relation	No relation	No relation	No relation	Comple ment	No relation	Conflicts		
G9	No relation	No relation	No relation	No relation	Comple ment	No relation	No relation	No relation	

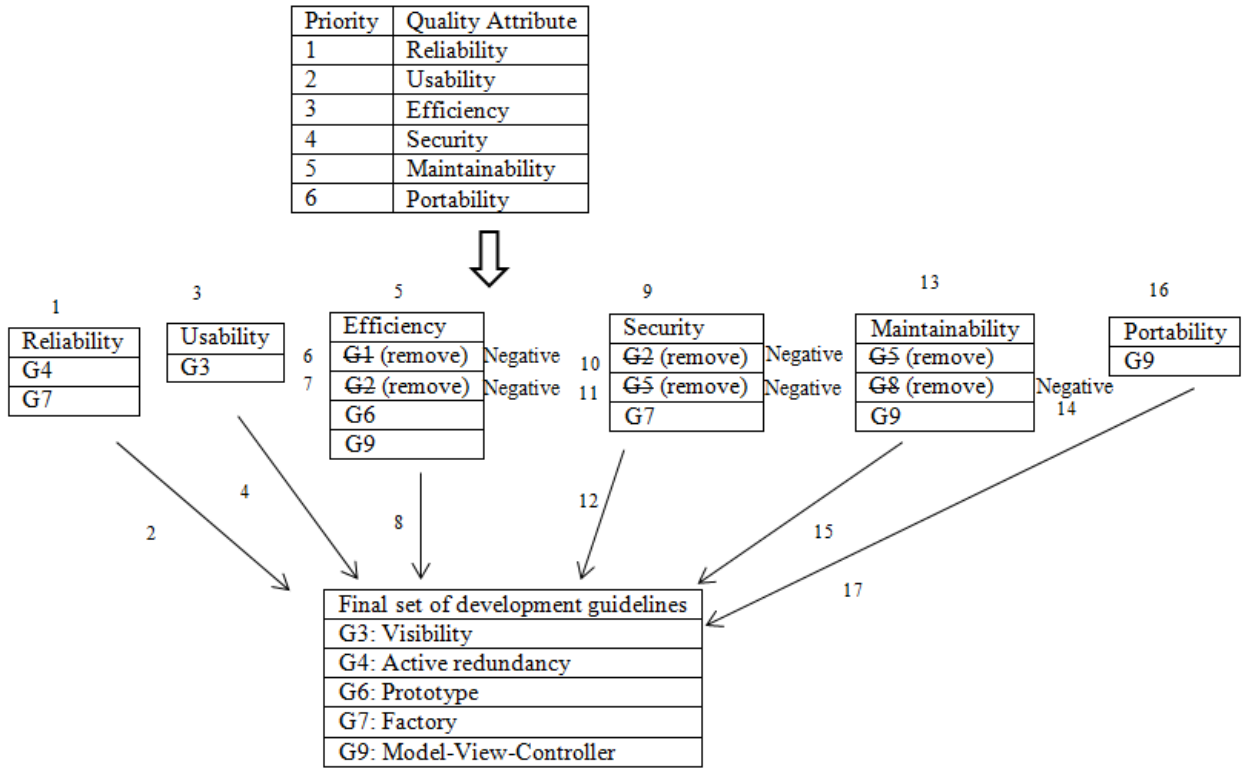

Fig. 4: Example of selecting homogeneous software development guidelines for virtual lab environment

The working steps (1-17) to select a set of homogeneous software development guidelines are shown in Fig. 4, and are descriptively given below.

1) The software development guidelines that have positive effect on the highest priority quality attribute, which in this case, reliability, are gathered.
2) <G4, G7> have no overlapping or conflicting effect between each other, thus <G4, G7> are included into the *Final set of development guidelines*.
3) <G3> is identified as the complementary guideline for usability and has no conflicting effect with <G4, G7>.
4) <G3> is then added into *Final set of development guidelines* as well.
5) There are a total of four guidelines, which are <G1, G2, G6, G9>, that improve efficiency.
6) <G1> has negative impact on usability, which is of higher importance compared to efficiency. Thus, <G1> is removed.
7) <G2> has negative impact on reliability, which is of higher importance compared to efficiency. Thus, <G2> is removed.
8) <G6> and <G9> with no negative and conflicting relationship are added into the *Final set of development guidelines*.
9) The set of development guidelines that complement security are <G2, G5, G7>.
10) <G2> has been excluded earlier thus will not be considered.
11) <G5> has negative impact on efficiency, which is of higher priority compared to security. Thus, <G5> is removed.
12) <G7> has been added into *Final set of development guidelines*.
13) The set of development guidelines that complement maintainability are <G5, G8, G9>.
14) <G5> has been excluded earlier thus will not be considered. <G8> has negative impact on reliability, which is of higher importance compared to maintainability. Thus, <G8> is removed.
15) <G9> has negative impact on reliability and efficiency, which is of higher importance compared to maintainability. Thus, <G9> is removed.
16) No guideline that complements maintainability is added into *Final set of development guidelines*.
17) <G9> has been identified as the guideline that complements portability. <G9> has been excluded earlier thus will not be considered.

The final set of software development guidelines thus consists of <G3, G4, G6, G7>. These guidelines are said to be homogeneous in the sense that they do not have overlapping, duplicate and conflicting relationships.

3.4 Research Contribution in Facilitating Architecture Evaluation Through Priority Assessment and Software Development Guidelines

The result of fuzzy AHP and selection of development guidelines can then be used to facilitate the overall architecture evaluation process, specifically ATAM. ATAM consists of 9 methodical steps which involve all stakeholders to evaluate software architectures relative to quality attribute requirements.

The main steps in ATAM with respect to quality attribute prioritization and software development guidelines are the following.

- In Step 2, the business drivers and quality attributes are discussed and defined.
- In Step 4, the potential architectural approaches are discussed and defined
- In Step 5, a utility tree is defined for quality attributes which capture the importance of quality requirements and the value of achieving a certain level of quality. The utility tree is a form of quality requirements prioritization [41].
- In Step 6, possible architectural approaches are evaluated with respect to the utility tree defined in Step 5.

Fig. 5 depicts how the results from fuzzy AHP priority assessment and software development guidelines can help facilitate those steps, namely steps 2, 4, 5, and 6. The left side of Fig. 5 mimics the overall workflow of prioritization and fulfillment of quality attributes shown in Fig. 1. The right side of Fig. 5 shows the nine ATAM steps. Meanwhile, the arrows connecting the left and right sides of the figure show the steps that can be facilitated and complemented.

- In Step 2 of ATAM, business drivers and quality attributes are identified, which is similar to the first step of quality attribute prioritization. Thus, the set of quality attributes identified during the prioritization process can be reused in ATAM.

- Similarly, the set of software development guidelines that complement the associated quality attributes can be used to supplement Step 4 in ATAM.
- Meanwhile, the result of fuzzy AHP can be reused as the utility tree in Step 5 of ATAM. Compared to a typical utility tree in ATAM, the result of fuzzy AHP carries even more information such as the ranking and weightage of each associated quality attribute.
- Finally, the set of homogenous software development guidelines identified for virtual lab development can be used to assist in Step 6 of ATAM to provide more insight about the chosen architectural approaches.

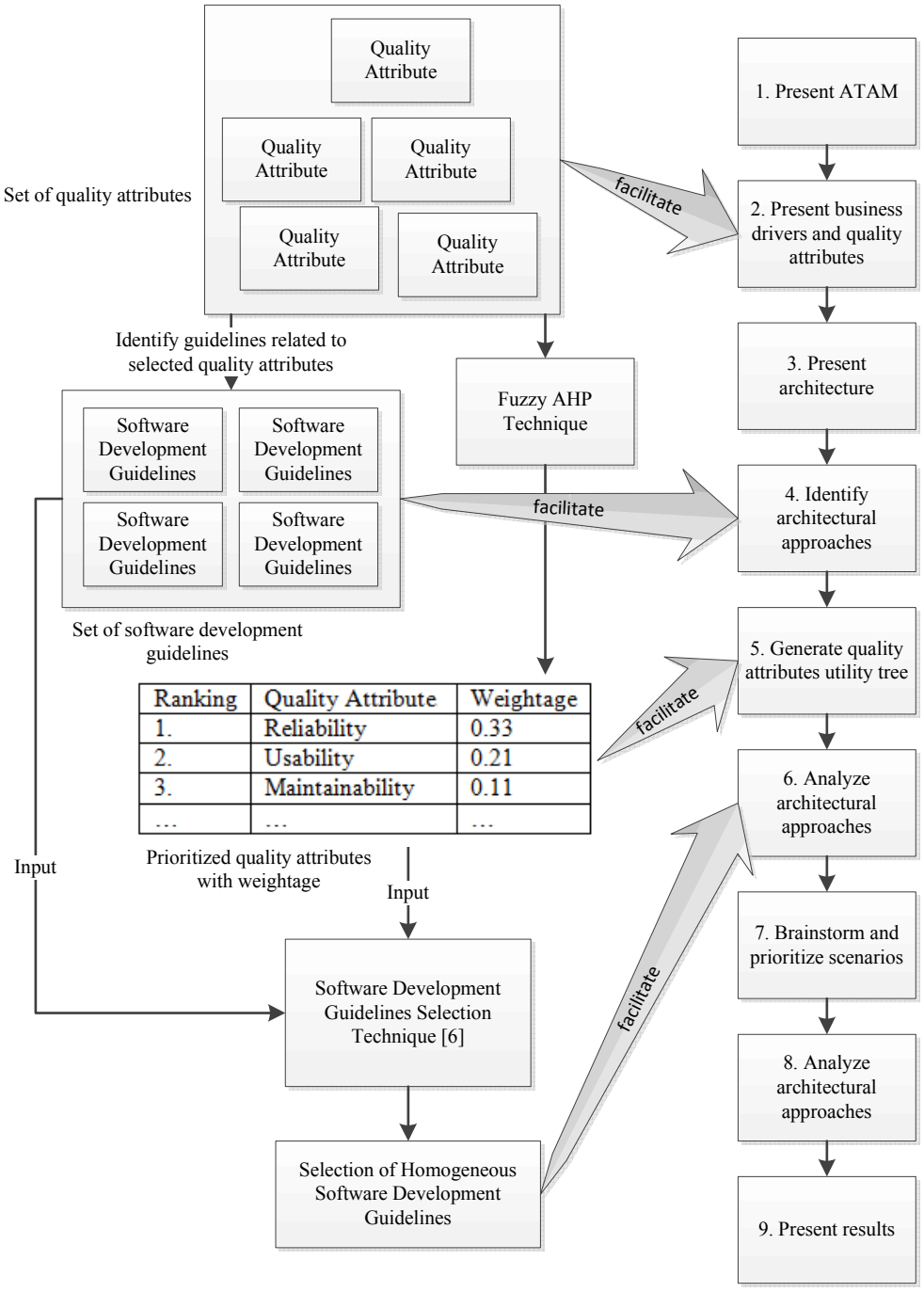

Fig. 5: Illustration of how priority assessment and software development guidelines can facilitate ATAM

As mentioned earlier, development of virtual lab is still at its early stages and there are high risks associated with improper software architecture design. The ATAM technique, facilitated by the prioritization of quality attributes and software development guidelines, can help minimize the aforementioned risk in a virtual lab environment.

4.0 DISCUSSION

The priority assessment is based on fuzzy AHP, which works effectively in aggregating stakeholders' decisions. Fuzzy AHP is a suitable evaluation framework which is capable of handling MCDM problem and uncertain inputs. Applying fuzzy AHP in the virtual lab environment not only helps stakeholders to prioritize quality attributes, but also aids in dealing with disagreement among stakeholders.

The aggregated results obtained from the priority assessment of quality attributes are based on the assumption that the values α and $\beta = 0.5$ use alpha-cut method. The values α and β are dependent on environmental uncertainties. These values will directly affect the weights of individual criteria and priority ranking. If the participants involved in priority assessment have strong background knowledge on virtual lab, the values of α and β can be readjusted to indicate confident judgments. In-depth research and analysis is needed in order to determine the values of α and β accurately. The accuracy of fuzzy AHP can be further improved by investigating the impact of α and β values toward the final results.

The result from priority assessment using fuzzy AHP shows that reliability ranked highest in the virtual lab environment. The concept of virtual lab allows end users to remotely access computing resources using a stable Internet connection all the time regardless location and time constraints. This is an advantage over traditional computer laboratories where students can only access to computing resources during office hours. Thus, the stakeholders anticipate that the virtual lab system is reliable and capable of maintaining high availability to fully serve the requests of students and researchers in the educational sector.

Security, on the other hand, ranked fourth in the list because virtual lab systems deployed in educational institutions usually do not involve any monetary transactions. Besides that, stakeholders in this context are less concern about possible leak or loss of confidential data. Typically, students and researchers who visit a computer laboratory mainly use it to perform software development, software testing on different platforms, document editing, and data analysis using specialized software tools that are not commonly available. In a virtual lab perspective, virtual machines need to be returned to the unified resource pool when they are not needed, and users can easily transfer all their confidential data into a removal disk. The data stored in the removal disk can be copied into other virtual machines if one wishes. Transferring data from one virtual machine to another is more robust, and users have more control over the location of their data.

The result shown in Table 4 can serve as a reference model for SLA in virtual lab environments. Developers can focus on the higher-priority quality attributes to ensure that the final system can meet the expectation of stakeholders. The selected software development guidelines are homogeneous in such a way that they do not have conflicting and overlapping relationships among themselves. By applying this set of homogeneous guidelines, software developers will less likely need to reason about the complexity behind the associated quality attributes.

However, there are exceptional cases where two overlapping development guidelines improve the same quality attribute. Applying both development guidelines will cause redundancy and overhead toward the overall system architecture. In this case, software engineers will need to decide which guideline is more appropriate to be implemented based on their skills and experiences. One option is assigning a weightage to each and every software development guideline, depending on the effectiveness of the guideline towards particular quality attributes. For example, a weightage of 0.9 is given to active redundancy in improving system availability, while 0.8 for voting. Selection of software development guidelines toward availability can be made based on this weightage. Thus, software development guidelines can be chosen more effectively regardless of a software engineer's experience.

5.0 CONCLUSION AND FUTURE WORK

Prioritization of quality attributes in virtual lab plays an important role to help software developers to focus on fulfilling higher-priority attributes within the limited budget and time. Because the development of virtual lab is still at its infancy, there are very limited references and established architecture design that can be adopted by software developers. Software developers need to focus on capturing and prioritizing essential quality attributes so that the software architecture designed based on the captured requirements can fulfill the stakeholders' requests. However stakeholders often specify requirements using natural language which are often imprecise. Judgments of stakeholders are usually subjective and ambiguous. Even if quality attributes can be prioritized, most software developers only look into one quality attribute at one time, neglecting possible contradicting effects among other attributes. The fact is that most software systems are required to have multiple quality attributes at the same time. This paper addresses the problem of capturing stakeholders' requirements and improving achievement of multiple quality attributes in a virtual lab environment. A homogenous set of software development guidelines are identified to help complement the achievement of quality attributes in a virtual lab environment. Software developers can then design the software architecture of virtual lab systems based upon the prioritized quality attributes and chosen software development guidelines.

This paper has made several contributions toward the establishment of virtual lab in educational institutions. Firstly, fuzzy AHP is used during the pre-negotiation stage of SLA in order to prioritize and weight the associated quality attributes for virtual lab development. Based on the results of fuzzy AHP, software developers can design an appropriate software architecture which is capable of fulfilling those quality attributes stated in the SLA. This will not only solve the conflicting relationships among quality attributes but also enable software developers to concentrate on the most important attributes within the given time and budget constraints.

Besides that, the homogeneous set of software development guidelines selected using the technique introduced by [6] can be treated as guidance for early adopters of virtual lab system. The selected software development guidelines can help improve the overall achievement of quality attributes and ensure that conflicting effects among development guidelines and higher-priority quality attributes can be avoided.

The combination of fuzzy AHP and software development guidelines can help facilitate the ATAM architectural evaluation technique. Several results from prioritization of quality attributes and software development guidelines can be reused in ATAM not only to save development effort but also provide more insight on resolving trade-off among the associated quality attributes.

Further work can be considered by involving subject matter experts from different higher educational institutions. The aggregated opinions from different experts can be used to model a system architecture for a virtual lab through attribute-driven design. Eventually, the system architecture can act as a reference to introduce a virtual lab solution in the educational sector.

ACKNOWLEDGEMENT

This work is carried out within the framework of a research project supported by eScienceFund with reference 01-01-03-SF0851, funded by Ministry of Science, Technology and Innovation (MOSTI), Malaysia.

REFERENCES

[1] H. E. Schaffer, S. F. Averitt, M. I. Hoit, A. Peeler, E. D. Sills, and M. A. Vouk, "NCSU's Virtual Computing Lab: A Cloud Computing Solution," *Computer,* vol. 42, pp. 94-97, 2009.

[2] S. Zardari, R. Bahsoon, and A. Ekart, "Cloud Adoption: Prioritizing Obstacles and Obstacles Resolution Tactics Using AHP," in *Requirements Engineering Track, The 29th ACM Symposium On Applied Computing, Gyeongju, Korea*, 2014.

[3] L. Chung and J. Prado Leite, "On Non-Functional Requirements in Software Engineering," in *Conceptual Modeling: Foundations and Applications*. vol. 5600, A. Borgida, V. Chaudhri, P. Giorgini, and E. Yu, Eds., ed: Springer Berlin Heidelberg, 2009, pp. 363-379.

[4] R. Kazman, L. Bass, M. Klein, T. Lattanze, and L. Northrop, "A Basis for Analyzing Software Architecture Analysis Methods," *Software Quality Journal,* vol. 13, pp. 329-355, 2005/12/01 2005.

[5] L. Dobrica and E. Niemela, "A survey on software architecture analysis methods," *Software Engineering, IEEE Transactions on,* vol. 28, pp. 638-653, 2002.

[6] M. Hneif and S. P. Lee, "Using Guidelines to Improve Quality in Software Nonfunctional Attributes," *Software, IEEE,* vol. 28, pp. 72-77, 2011.

[7] G. Redig and M. Swanson, "Total quality management for software development," in *Computer-Based Medical Systems, 1993. Proceedings of Sixth Annual IEEE Symposium on*, 1993, pp. 301-306.

[8] "Software Engineering - Product Quality, {ISO/IEC} 9126-1," 2001.

[9] C.-H. Jane. (2013) The Twin Peaks of Requirements and Architecture. *IEEE Software*. 24-29.

[10] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The architecture tradeoff analysis method," in *Engineering of Complex Computer Systems, 1998. ICECCS '98. Proceedings. Fourth IEEE International Conference on*, 1998, pp. 68-78.

[11] P. Bengtsson, N. Lassing, J. Bosch, and H. van Vliet, "Architecture-level modifiability analysis (ALMA)," *Journal of Systems and Software,* vol. 69, pp. 129-147, 2004.

[12] L. Zhu, A. Aurum, I. Gorton, and R. Jeffery, "Tradeoff and Sensitivity Analysis in Software Architecture Evaluation Using Analytic Hierarchy Process," *Software Quality Journal,* vol. 13, pp. 357-375, 2005/12/01 2005.

[13] F. Bachmann, L. Bass, M. Klein, and C. Shelton, "Designing software architectures to achieve quality attribute requirements," *Software, IEE Proceedings -,* vol. 152, pp. 153-165, 2005.

[14] E. Hull, K. Jackson, and J. Dick, *Requirements engineering*. London ; New York: Springer, 2002.

[15] A. M. Davis and H. Pei, "Giving voice to requirements engineering," *Software, IEEE,* vol. 11, pp. 12-16, 1994.

[16] J. Siddiqi, "Requirement Engineering: The Emerging Wisdom," *Software, IEEE,* vol. 13, p. 15, 1996.

[17] C. Chong, S. Lee, and T. Ling, "Development of virtual lab system through application of fuzzy analytic hierarchy process," in *Information Science and Digital Content Technology (ICIDT), 2012 8th International Conference on*, 2012, pp. 207-211.

[18] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," in *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, 2010, pp. 606-610.

[19] P. Patel, A. Ranabahu, and A. Sheth, "Service level agreement in cloud computing," in *Cloud Workshops at OOPSLA*, 2009.

[20] S. Zardari, F. Faniyi, and R. Bahsoon, "Using Obstacles for Systematically Modeling, Analysing, and Mitigating Risks in Cloud Adoption," *Aligning Enterprise, System and Software Architectures.},* 2012.

[21] G. Génova, J. Fuentes, J. Llorens, O. Hurtado, and V. Moreno, "A framework to measure and improve the quality of textual requirements," *Requirements Engineering,* vol. 18, pp. 25-41, 2013/03/01 2013.

[22] X. F. Liu, "A quantitative approach for assessing the priorities of software quality requirements," *Journal of Systems and Software,* vol. 42, pp. 105-113, 1998.

[23] C.-C. Chou, L.-J. Liu, S.-F. Huang, J.-M. Yih, and T.-C. Han, "An evaluation of airline service quality using the fuzzy weighted SERVQUAL method," *Applied Soft Computing,* vol. 11, pp. 2117-2128, 2011.

[24] J. Karlsson, C. Wohlin, and B. Regnell, "An evaluation of methods for prioritizing software requirements," *Information and Software Technology,* vol. 39, pp. 939-947, 1998.

[25] T. L. Saaty, *The analytic hierarchy process : planning, priority setting, resource allocation*. New York ; London: McGraw-Hill International Book Co., 1980.

[26] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *Software, IEEE,* vol. 14, pp. 67-74, 1997.

[27] D. A. Heger, "A disquisition on the performance behavior of binary search tree data structures," *European Journal for the Informatics Professional,* vol. 5, pp. 67-75, 2004.

[28] J. Karlsson, *A systematic approach for prioritizing software requirements*: Linköpings universitet, 1998.

[29] O. S. Vaidya and S. Kumar, "Analytic hierarchy process: An overview of applications," *European Journal of Operational Research,* vol. 169, pp. 1-29, 2006.

[30] R. G. Dromey, "Software Quality—Prevention versus Cure?," *Software Quality Journal,* vol. 11, pp. 197-210, 2003.

[31] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*: Addison-Wesley Professional, 2012.

[32] B. Andreopoulos, "Satisficing the conflicting software qualities of maintainability and performance at the source code level," in *WER-Workshop em Engenharia de Requisitos*, 2004, pp. 176-188.

[33] D. G. Rosado, C. Gutiérrez, E. Fernández-Medina, and M. Piattini, "Security patterns and requirements for internet-based applications," *Internet Research,* vol. 16, pp. 519-536, 2006.

[34] B. Andreopoulos, "Achieving Software Quality Using the NFR Framework: Maintainability and Performance," in *Proc. 3rd Int'l Conf. Computer Science, Software Eng., In formation Technology, e-Business, and Applications, 2004*.

[35] D. Budgen, *Software design*, 2nd ed. Harlow, England ; New York: Addison-Wesley, 2003.

[36] L. M. Kaufman, "Data Security in the World of Cloud Computing," *Security & Privacy, IEEE,* vol. 7, pp. 61-64, 2009.

[37] H. Takabi, J. B. D. Joshi, and G. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," *Security & Privacy, IEEE,* vol. 8, pp. 24-31, 2010.

[38] A. di Costanzo, M. D. de Assuncao, and R. Buyya, "Harnessing Cloud Technologies for a Virtualized Distributed Computing Infrastructure," *Internet Computing, IEEE,* vol. 13, pp. 24-33, 2009.

[39] R. Aull-Hyde, S. Erdogan, and J. M. Duke, "An experiment on the consistency of aggregated comparison matrices in AHP," *European Journal of Operational Research,* vol. 171, pp. 290-295, 2006.

[40] T.-S. Liou and M.-J. J. Wang, "Ranking fuzzy numbers with integral value," *Fuzzy Sets Syst.,* vol. 50, pp. 247-255, 1992.

[41] A. Koziolek, "Research Preview: Prioritizing Quality Requirements Based on Software Architecture Evaluation Feedback," in *Requirements Engineering: Foundation for Software Quality*. vol. 7195, B. Regnell and D. Damian, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 52-58.

BIOGRAPHY

Chun Yong Chong obtained his MSc degree in Computer Science from University of Malaya, Malaysia, in 2012. He is currently a PhD candidate in the Software Engineering Department at University of Malaya. He can be contacted at cychong@um.edu.my.

Sai Peck Lee is a professor at the University of Malaya. She obtained her PhD in Computer Science from Université Panthéon-Sorbonne (Paris I). Her current research interests include Object-Oriented Techniques and CASE tools, Software Reuse, Requirements Engineering and Software Quality. She is a member of the IEEE and a founding member of the Informing Science Institute. She is currently serving as the executive editor of Malaysian Journal of Computer Science and in several expert review panels, both locally and internationally. She can be contacted at saipeck@um.edu.my.

Teck Chaw Ling is an associate professor at University of Malaya, where he obtained his PhD from the same university. His research areas include core network research, Inter-domain Quality of Service, Software Defined Network, Voice over IP, Grid and Cloud Computing, and Network Security. He can be contacted at tchaw@um.edu.my.