A Quality Model for Evaluating Software-as-a-Service in Cloud Computing *

Jae Yoo Lee, Jung Woo Lee, Du Wan Cheun†, and Soo Dong Kim
Department of Computer Science
Soongsil University
511 Sangdo-Dong, Dongjak-Ku, Seoul, Korea 156-743
{jylee81, jwlee, dwcheun}@otlab.ssu.ac.kr sdkim777@gmail.com

Abstract

Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. One type of cloud service, SaaS is commonly utilized and it provides several benefits to service consumers. To realize these benefits, it is essential to evaluate the quality of SaaS and manage relatively higher level of its quality based on the evaluation result. Hence, there is a high demand for devising a quality model to evaluate SaaS cloud services. Conventional frameworks do not effectively support SaaS-specific quality aspects such as reusability and accessibility. In this paper, we propose a comprehensive model for evaluating quality of SaaS. We first define key features of SaaS. And then, we derive quality attributes from the key features, and define metrics for the quality attributes. To validate our quality model for SaaS, we conduct assessment based on IEEE 1061. By using the proposed SaaS quality model, SaaS can be evaluated by both service providers. Furthermore, the evaluation results are utilized as an indicator for SaaS quality management.

1. Introduction

Cloud Computing (CC) is emerged as an effective reuse paradigm, where software functionality, hardware computing power, and other computing resources are delivered in the form of *service* so that they become available widely to consumers [1]. *Software-as-a-Service (SaaS)* is one type of cloud services, where software functionality is delivered as a service [2][3]. SaaS provides benefits to service consumers; no initial cost to purchase software, free of maintenance/updates, accessibility through Internet, high availability, and pay-per-use pricing. That is, SaaS should maintain relatively higher level of its quality than conventional system. Consequently, these intrinsic features of SaaS require more rigorous quality measurements.

As shown in Figure 1, conventional frameworks for measuring quality such as ISO 9126 would be limited in evaluating the quality of SaaS, mainly due to the gap between the conventional computing paradigms and CC paradigm. That is, conventional quality measurement frameworks do not effectively evaluate CC-specific quality aspects. However, widely accepted quality model for evaluating SaaS and supported instructions are yet to come.

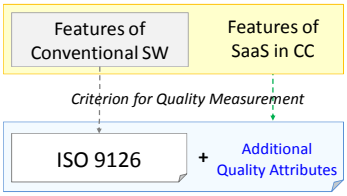

Figure 1. Differences between Conventional Software and Cloud SaaS

Therefore, there is a high demand for devising a quality model to evaluate SaaS cloud services which have characteristics such as *supporting commonality*, *internet-based invocation*, *virtualization* and *data management on server side*. In this paper, we propose a comprehensive quality model for evaluating SaaS. We first define key features of SaaS. And then we derive quality attributes from the features, and define metrics for the quality attributes. To validate our quality model for SaaS, we conduct assessment in terms of *correlation*, *consistency*, and *discriminative power*, defined in IEEE 1061. By using the proposed SaaS quality model, SaaS can be evaluated by service providers. Furthermore, the evaluation results are utilized as an indicator for SaaS quality management.

2. Related Works

* This research was supported by National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development and Experts Education.
† Corresponding Author

978-0-7695-3903-4/09 $26.00 © 2009 IEEE
DOI 10.1109/SERA.2009.43

ISO 9126 is an international standard for the evaluation of product quality [4]. This standard provides three aspects for evaluating software products; internal quality, external quality, and quality in use. And, there are sixteen characteristics for three types of qualities. However, this standard focuses on evaluating quality of conventional products. Hence, it is required that the standard is customized and extended to evaluate the quality of SaaS.

Jureta's work proposes a quality model, called QVDP, to measure the quality of Service-Oriented System [5]. This model consists of four sub models; quality characteristic, characteristic value, quality dependency, quality Priority. These represent dependencies and priorities between quality dimensions and quality characteristics. However, this work considered services-oriented application as a target of quality model and identifies issues related to them at conceptual level.

Kim's work defines a model for web services quality management and quality factors in the process of developing and using web services [6]. This work suggests six quality factors and their several sub-factors. Also, it provides metrics to measure quality factors. Hence, it is required that this model is customized and extended to evaluate the quality of SaaS.

Most of current works are not for SaaS but for certain targets such as a conventional software or SOA based system. Due to the situation, it is hard to evaluate quality of SaaS and judge which SaaS is good. Therefore, our work provides a quality model to evaluate SaaS.

3. Key Features of Software-as-a-Service

To define a quality model for evaluating SaaS, it is a prerequisite to identify key features of SaaS. From our rigorous evaluation on current references in CC [2][7], we identify key features as shown in Figure 2;

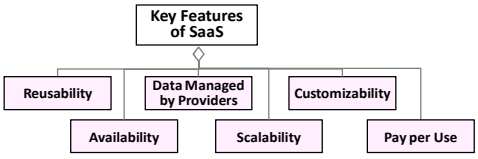

Figure 2. Key Features of SaaS

Reusability: In software engineering, reusability is the ability of software elements to serve for the construction of many different applications.

The fundamental underlying of CC is to reuse various types of internet based services [2]. In case of SaaS, software itself is a target of reuse and it is delivered to service consumers over the Internet. That is, one-to-many relationships are often used when delivering SaaS services. For example, Google map service provides a set of operations to utilize shared information on map and local, which can be used by various customers [9].

Data Managed by Provider: SaaS is a model of software deployment whereby service providers license applications to customers for use as a service on demand. Thus service providers are responsible for service installation and data management on their own server. Therefore, most data which service consumers produce is stored on provider's data center and managed by provider. Thus service customers do not perceive two things in detail; *where* is their data stored and *how* can be the data managed. So that, the service customers may distrust the services and then service utilization becomes lower, if service providers do not provide data security and reliability function [10].

Service Customizability: Service customizability means the ability for services to be changed by service consumers based on the individual requirements. This characteristic allows service providers to meet the different needs of each customer.

According to the characteristics of cloud services, various consumers, who can access the Internet, can become potential users of the cloud services [11]. Because of that, it is impossible that service providers customize their cloud services for all service consumers. Thus service consumers need to customize their services for their own purposes. If service providers do not provide customizable SaaS services, the only thing service consumers can is that they simply utilize the services. This will limit the usage of SaaS services.

Availability: In CC, the service consumers are able to access SaaS service from a Web browser via the Internet. Also, the consumers do not have any ownership for the SaaS which is deployed and runs on the provider's server. Because of these natures, many SaaS vendors focus their attentions to achieve high availability of services. If a SaaS is not available, service consumers cannot use the functionality of the SaaS. For example, Gmail service was down completely at September 1st, 2009. Because of that, many Gmail users could not use the service which the users read or write email.

Scalability: In software engineering, scalability is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work or to be readily enlarged.

Due to the black-box nature of CC services, service consumers cannot control resources which are utilized by the services, such as memory, network, or CPU utilization. That is, a service provider is a responsible for rescaling the resources according to consumer's requests without notifying the consumers in detail.

Pay per Use: The expenses for SaaS are estimated

not about purchasing an ownership of services, but about using services such as the number of service invocations or duration which services are utilized. That is, service consumers can connect and use the service each time they want, and then pay for just amount of usage such as a utility like electricity or water. Therefore, the customer takes an activity for using the service.

4. Quality Attributes

Based on ISO/IEC 9126, *Efficiency* and *Reliability* are extended to cover CC specific features. Derived from the key features, *Reusability*, *Availability*, and *Scalability* are newly defined.

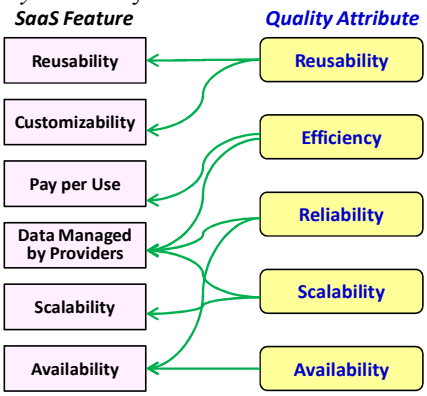

Figure 3. Mapping from Features to Quality Attributes

Figure 3 shows mapping relationship between the key features of SaaS and the quality attributes. For each attribute, its definition and our rationale for defining it are as the following.

Reusability: This attribute measures if functionalities provided by services are common to the requirements defined by service consumers.

The rationale for defining this attribute is explained by both service provider's view and service consumer's view. Typically, SaaS providers have the business model, *pay per use*. If SaaS provides functionalities which meet requirements, then the number of service consumers and return on investment (ROI) is consequently increased. Hence, the reusability is treated as an indicator which can reveal ROI of service providers. Service consumers want to use better services. When service consumers evaluate the services, they typically consider two factors; the number of subscribers and the number of provided features required by service consumers. These factors indirectly show that quantitative values, which reveal *supporting commonality,* are important to service consumers.

Availability: This attribute measures the ratio of the total time to the time which a SaaS service is capable of being operable. Typically availability objectives are specified either in decimal fractions or percentage.

The rationale for defining this attribute is that SaaS to be available is a prerequisite for the successful execution of the SaaS. Especially, a SaaS is deployed and runs on the service provider's server. Service consumers can discover, subscribe, and utilize the SaaS which they want through *internet-based invocation*. Also, the service consumers cannot have the control of the services which are on the service provider side. Unless high availability of services is not ensured, service consumers cannot use the services on demand.

Scalability: This attribute measures the ability to either handle growing amounts of resources, used for SaaS, or to be readily enlarged.

The rationale for defining this attribute is that a cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet [12]. SaaS should support functionality that the amounts of resources used for the SaaS are grown whenever service consumers request, since SaaS is a kind of application which provides a set of functionalities as a whole. Also, service providers cannot predict how much the amount of resources will be utilized. For example, Google docs service provides functionality of word processor. When service consumers finish writing the first page, the service creates the second page as a blank automatically and allocates more resources for the page. Therefore, service providers can get more benefit in terms of ROI, if they provide their application which can scale resources flexibly.

Reliability: This attribute measures the ability of a SaaS to keep operating with specified level of performance over time. Several aspects of reliability are important within a cloud computing, particularly the reliability of the messages that are exchanged between the service consumers and the services, and the reliability of the services themselves.

The rationale for defining this attribute is explained by three folds. First fold is that SaaS are reusable and are used in various users. That is, if an unexpected fault occurs in a SaaS, it propagates failures to service consumers who utilize the SaaS.

Second fold is that most data utilized in SaaS is managed on server side in CC. That is, the reliability of SaaS has an effect on service consumers themselves. Thus the company might have a serious obstacle in terms of economy or opportunity if faults occur in the SaaS. Due to the *data management on server side*, many researchers focus on issues about the unreliable state of SaaS.

Third fold is that running SaaS are performed on the service provider's side. That is, service consumers have less belief due to their limited controllability. This triggers similar problem of the data management.

Efficiency: This attribute measures the amount of

resource used for SaaS when providing the required functionality and the level of performance, under stayed condition. That is, it is a measure of how well SaaS utilizes resources.

The rationale for defining this attribute is that the expense for SaaS is estimated not about purchasing an ownership of the service, but about the amount of usage of the service. That is, a typical business model of cloud computing depends the amount of utilized resource or time-use.

5. Quality Metrics

In this section, we define metrics for each quality attribute and provide a description including formula, value range, and relevant interpretations. Figure 4 shows relationship between quality attributes and their metrics.

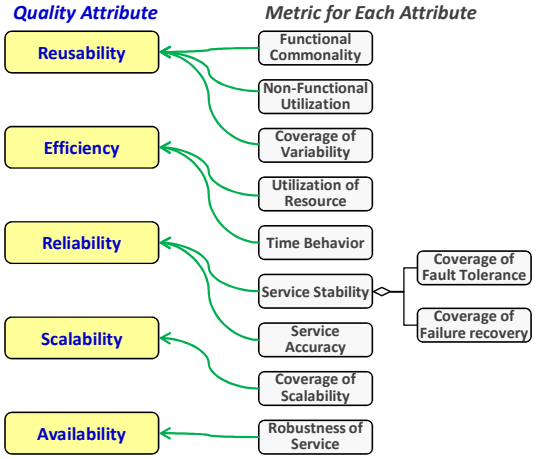

Figure 4. Metrics for Quality Attributes

5.1. Reusability

The reusability is measured by Functional Commonality (FC), Non-functional Commonality (NFC), and Coverage of Variability (CV) metrics.

Functional Commonality: This metric measures an average of commonality of each functional feature defined in a target SaaS. Commonality of each functional feature can be measured by calculating the degree of members who use each functional feature. This can be computed as;

$$FC = \left(\sum_{i=1}^{n} \frac{number\ of\ requirements\ appling\ ith\ functional\ feature}{total\ number\ of\ requirements\ analyzed\ in\ the\ domain} \right) / n$$

where feature is a function or characteristic in a SRS to specify functionality of SaaS. And n is the total number of functional features. Therefore, the number of functional features is identified from SRSs in the same domain.

Hence, the range of *FC* is 0...1. The value 1 denotes all features applied in the SaaS are common in the same domain.

Non-functional Commonality: This metric measures the average of commonality of each non-functional feature. This is similar to *FC* metric and can be computed as;

$$\mathrm{NFC} = \left(\sum_{i=1}^{m} \frac{\text{number of requirements appling ith non - functional feature}}{\text{total number of requirements analyzed in the domain}} \right) / m$$

where *m* is the total number of non-functional features in the domain. Also, non-functional features are identified from the SRSs in the same domain.

The range is 0...1. The value 1 implies that all non-functional features are common in the same domain.

Coverage of Variability: This metric measures how many of the variation points included in the domain are actually realized in the SaaS. This can be computed as;

$$CV = \frac{(\text{number of variation points realized in the SaaS})}{(\text{number of variation points in the domain})}$$

where the denominator is the number of variation points identified for SRSs in the domain.

The range is 0...1. A higher value indicates that the coverage of the SaaS is comprehensive enough to have realized a larger number of variation points.

Finally, the value of ***Reusability*** is derived from there three metrics;

$$Reusabilty = W_{FC} \cdot FC + W_{NFC} \cdot NFC + W_{CV} \cdot CV$$

where $\boldsymbol{W_{FC}}$, $\boldsymbol{W_{NFC}}$, and $\boldsymbol{W_{CV}}$ are the weights for each metric which the sum is 1. The value of weight is set by considering the importance of each metric.

The range of *Reusability* is 0...1 and the higher value indicates the SaaS is more reusable.

5.2. Availability

The availability is measured by Robustness of Service (ROS) metric.

Robustness of Service: This metric measures a ratio of the total operating time to the time which the SaaS is available to be invoked. This can be computed as;

$$\mathrm{ROS} = \frac{(\text{available time for invoking SaaS})}{(\text{total time for operating SaaS})}$$

where the denominator is a total period of time to start operating the SaaS. And the numerator is the specific period of time to be able to invoke the SaaS. The numerator can be calculated as 'total operating time of SaaS – Service Failure Time'. The Service Failure Time is a total time when the service cannot be invoked because of any service failures.

The range is 0...1 and the higher value indicates that the SaaS has higher availability.

5.3. Scalability

The scalability is measured by *Coverage of Scalability (COS)* metric.

Coverage of Scalability: This metric measures the average amount of allocated resources among the amount of requested resources. This can be computed as;

$$COS = \left(\sum_{i=1}^{k} \frac{(amount\ of\ allocated\ resources\ of\ ist\ request)}{(total\ amount\ of\ requested\ resources\ of\ ist\ request)} \right) / k$$

where k is the total number of requests for extending resources used.

The range is 0…1. The value 1 implies that all requested resources are allocated. In general, a SaaS with a higher *COS* value yields a higher acceptance for the customer's request to re-scale the resources used.

5.4. Reliability

The reliability is measured by *Service Stability (SS)* and *Service Accuracy (SA)* metrics. To define these metrics, we precisely define two terms; *fault* and *failure*. *Fault* indicates an abnormal condition of certain system whether it cause a reduction in the capability of the system. *Failure* is an inability of the system to perform its required functions within specified QoS. Therefore, *fault* can trigger *failure*. And, the fault may not reveal observable *failure*.

Service Stability: This metric measures a degree of the service operating without faults or failures.

Coverage of Fault Tolerance (CFT) metric measures the ratio of bearing the occurrence of a fault without a failure. This can be computed as;

$$CFT = \frac{(number\ of\ \ faults\ without\ becoming\ failures)}{(total\ number\ of\ \ faults\ occurred)}$$

where the denominator is the total number of occurring faults identified and the numerator is the number of faults which do not cause failures. From the log file of the SaaS server, we can also identify total number of occurring failures which were caused by occurring faults. Thus, we can derive the numerator from the discrepancy between total number of faults and failures.

The range is 0…1 and higher value indicates that the SaaS has higher durability about various faults.

Coverage of Failure Recovery (CFR) metric measures the ratio of remedying failures the specific period of time. This can be computed as;

$$CFR = \frac{(number of\ \ failures remedied)}{(total\ number of\ \ failures)}$$

where the denominator is the total number of failures caused by faults.

The range is 0...1 and higher value indicates that the SaaS has higher recoverability.

Service Accuracy: This metric measures a degree of correct response for a customer's request. The correct response means that the service provides what a service consumer wants within specific time period. The time period is defined in service level agreement (SLA). This can be computed as;

$$SA = \frac{(number of\ correct responses)}{(total\ number of\ \ requests)}$$

where the denominator is a total number of requests from customers and the numerator is a number of correct responses without failure. The numerator can be acquired as *(total number of request – number of failures)*.

The range is 0...1 and higher value indicates that the SaaS has higher accuracy.

Finally, *Reliability* is derived from two metrics;

$$Reliability = W_{CFT} \cdot CFT + W_{CFR} \cdot CFR + W_{SA} \cdot SA$$

where $\boldsymbol{W_{CFT}}$, $\boldsymbol{W_{CFR}}$, and $\boldsymbol{W_{SA}}$ are the weights for each metric which the sum is 1. The value of weight is set by considering the importance of each metric.

The range is 0...1 and higher value indicates that the SaaS is more reliable.

5.5. Efficiency

The efficiency is measured by *Resource Utilization (RU)* and *Time Behaviour (TB)* metrics.

Resource Utilization: This metric measures a ratio of an amount of allocated resources for the pre-defined resources. The resources include a network bandwidth and other computing powers such as storage capacity, CPU. This can be computed as;

$$RU = \frac{(amount\ of\ \ allocated\ resources)}{(amount\ of\ \ pre\text{-}defined\ resources)}$$

where the denominator is the amount of pre-defined resources for each type of resources and it is defined in the SLA. The numerator is an actual amount of allocated resources from the amount of pre-defined resources for invoking the service.

The range is 0...1 and higher value indicates that the SaaS has higher resource shares.

Time Behaviour: In general, waiting time occurs in the some part of service process, such as calling the data transmission time and preparation time to execute a function of service components. This metric measures a ratio of an execution time for a total invocation time. This can be computed as;

$$TB = \frac{(execution\ time)}{(total\ service\ invocation\ time)}$$

where the denominator is the consumed time period from sending request to receiving response. The numerator is an execution time just for processing functionality. The execution time can be computed as *(total service invocation time – waiting time)*.

The range is 0...1 and higher value indicates that the SaaS has higher efficiency of time.

Finally, ***Efficiency*** is derived from two metrics;

$$Efficiency = W_{SR} \cdot SR + W_{TB} \cdot TB$$

where $\boldsymbol{W_{SR}}$ and $\boldsymbol{W_{TB}}$ are the weights for each metric which the sum is 1. The value of weight is set by considering the importance of each metric.

The range of *Efficiency* is 0...1 and higher value of *Efficiency* indicates that the SaaS is more effective.

6. Assessment

In this section, we assess usefulness and practicability of the five metrics proposed in this paper by using three criteria which are identified from IEEE

std 1061 [13]. The reason for discarding the three criteria, predictability, discriminative power, and reliability, is that our quality model is applied when the SaaS development is finished. Therefore, it is not need to track changes in service quality over the lifecycle.

Correlation: The metrics proposed in this paper are derived from quality attributes. There is a strong linear association between quality attributes and their metrics. For example, *Reusability* is measured by *FC*, *NFC*, and *VC*. According to our formula, the reusability is a sum of FC, NFC, and VC. Therefore, each value can affect a calculated result of reusability.

Consistency: Similar to the criterion 'correlation,' the values among quality attributes also has strong linear association. If quality attribute values A_1, A_2, A_n, have relationship $A_1 > A_2 > A_n$, then the corresponding metric values shall have the relationship $M_1 > M_2 > M_n$. For example, *Scalability* is measured by *COS* metric. If the values of *Scalability* are S_{T1}, S_{T2}, S_{T3} at times *T1, T2, T3* have the rank among the values as $S_{T1} > S_{T2} > S_{T3}$, then the values of *COS* metric are also has same rank $COS_{T1} > COS_{T2} > COS_{T3}$.

Discriminative power: The metric is capable of discriminating between high-quality SaaS (e.g., high MTTF) and low-quality SaaS (e.g., low MTTF). The set of metric values associated with the former should be signiÞcantly higher (or lower) than those associated with the latter. We assume that three values of three SaaSs for *efficiency*, V_1, V_2, V_n, have relationship $V_1 > V_2 > V_n$. Hence, in terms of the *efficiency*, we can conclude that first SaaS can be evaluated as the highest efficient SaaS.

7. Conclusion

SaaS is one type of cloud services is emerged as an effective reuse paradigm. It provides benefits to service consumers; no initial cost to purchase software, free of maintenance/updates, accessibility through Internet, high availability, and pay-per-use pricing. Hence, evaluating the quality of SaaS becomes more important activity to a successful SaaS management.

In this paper, we presented a comprehensive quality model for evaluating SaaS. First, we extracted six features of SaaS from various references. And, we derived five quality attributes from the key features. Then, we defined ten metrics for evaluating these quality attributes. To show validity of the quality model, we conducted an assessment based on IEEE 1061.

We believe that these metrics are meaningful for service providers. They cover essential features of SaaS and provide a way to evaluate the SaaS in quantitative manner. Through our quality model for SaaS, service providers evaluate their services and may predict their ROI. Moreover, service consumers can refer quality evaluation results to discover, subscribe, and utilize SaaS. Furthermore, the evaluation results are utilized as an indicator for SaaS quality management.

References

[1] Weiss, A. "Computing in the Cloud," *netWorker*, Vol. 11, No. 4, pp. 16-26, 2007.

[2] Gillett, F.E., "Future View: New Tech Ecosystems of Cloud, Cloud Services, and Cloud Computing," *Forrester Research Paper,* 2008.

[3] Turner, M., Budgen, D., and Brereton, P., "Turning Software into a Service," *IEEE Computer*, Vol. 36, No. 10, pp. 38-44, 2003.

[4] Software Engineering – Product Quality – Part 1 : Quality Model. ISO/IEC 9126-1, June, 2001

[5] Jureta, I., Herssens, C., and Faulkner, S., "A comprehensive quality model for service-oriented systems," *Software Quality Journal*, to be published.

[6] Kim, E. and Lee, Y., *Quality Model for Web Services*, Working Draft, OASIS, September 2005.

[7] Kim, W., "Cloud Computing: Today and Tomorrow," *Journal of Object Technology*, Vol. 8, No. 1, pp. 65-72, January-February, 2009.

[8] Soo Dong Kim, "Software Reusability ," in *Wiley Encyclopedia of Computer Science and Engineering*, Vol. 4, edited by Benjamin W. Wah, pp. 2679-2689, Wiley-Interscience, Jan. 2009.

[9] Espadas, J., Concha, D., and Molina, A., "Application Development over Software-as-a-Service platforms," *In Proceedings of Software Engineering Advances(ICSEA 2008)*, IEEE, pp.97 - 104, October, 2008.

[10] T. V. Raman, "Cloud computing and equal access for all," In Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A 2008), ACM, April, 2008.

[11] Thomas, D., "Enabling Application Agility – Software as A Service, Cloud Computing and Dynamic Languages," *Journal of Object Technology*, Vol.7, No. 4, May-June, 2008.

[12] Gartner. "Gartner Says Cloud Computing Will Be As Influential As E-business," www.gartner.com. Gartner. http://www.gartner.com/it/page.jsp?id=707508.

[13] IEEE Standard for a software quality metrics methodology, IEEE Std 1061-1998, 1998.