Quality Model for Evaluating Platform as a Service in Cloud Computing

Temitope Olokunde, Sanjay Misra(✉), and Adewole Adewumi

Center of ICT/ICE Research, CUCRID, Covenant University, Ota, Nigeria
{temitope.olokunde, sanjay.misra, wole.adewumi}@covenantuniversity.edu.ng

Abstract. Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. Cloud computing can be offered in three ways namely: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Researchers have begun to investigate quality in SaaS and have proposed quality models in this regard. There is however a dearth of literature investigating quality in PaaS and IaaS which form the motivation for this work. In this paper, therefore a model is proposed for evaluating quality of PaaS. We first define key features of PaaS. And then, we derive quality attributes from the key features, and define metrics for the quality attributes. To validate our quality model for PaaS, we conduct assessment based on case studies using the Analytic Hierarchy Process. By using the proposed PaaS quality model, developers will be able to select suitable PaaS solutions among alternatives.

Keywords: PaaS · Quality · Analytical hierarchy process

1 Introduction

Cloud computing has become an increasingly adopted computing phenomenon. It is defined as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. Cloud computing services can be offered in three ways namely: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Figure 1 shows these service offerings in layers.

SaaS is a model in which software is provided by a vendor [2]. The software being provided is hosted at the vendor's data center under agreed terms of use. The key strength of SaaS is that users do not have to develop their own software but rather can patronize SaaS vendors [3]. An example of this is the Gmail service offered by Google [4]. PaaS is a model where the manufacturers supply services to the users such as development environment, server platform and hardware resources [5]. The users can then use such platforms to develop their own applications and make it available to other customers through their server and Internet. PaaS thus provides the middleware platform, applications development environment, database management systems, and application server for the enterprise and the individual [6]. Examples of PaaS include:

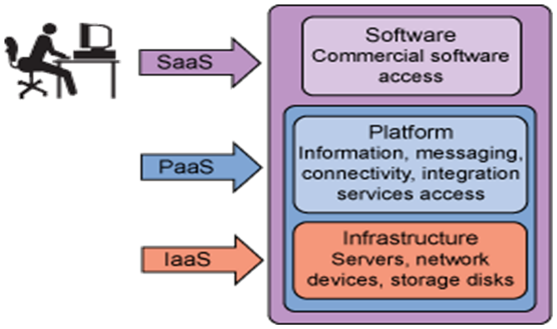
SaaS
PaaS
IaaS
Software
Commercial software access
Platform
Information, messaging, connectivity, integration services access
Infrastructure
Servers, network devices, storage disks

Fig. 1. Cloud computing layered architecture (http://www.ibm.com/developerworks/cloud/library/cl-cloudintro/)

Google App Engine offered by Google, Windows Azure offered by Microsoft. IaaS is a model that provides the consumer with the capability to provision processing, storage, networks, and other fundamental resources. It also allows the consumer to deploy and run arbitrary software, which can include operating systems and application [7].

In recent times, interest has been shown in investigating quality in cloud computing research and this is pioneered by the work of [8] who proposed a quality model for evaluating the quality of SaaS. However, no work was seen in literature proposing a quality model for IaaS and in particular PaaS. This forms the motivation for this present paper. The aim of this study therefore is to propose a model for evaluating quality of PaaS. We first define key features of PaaS. And then, we derive quality attributes from the key features, and define metrics for the quality attributes. To validate our quality model for PaaS, the Analytic Hierarchy Process (AHP) was applied on three known PaaS systems. The AHP is a multi-criteria decision-making approach introduced by [9]. By using the proposed PaaS quality model, PaaS systems can be evaluated by developers. Furthermore, the evaluation results are utilized as an indicator for PaaS quality management. The rest of this study is organized as follows: Sect. 2 examines related work. In Sect. 3 the proposed model is shown along with all the criteria used in its creation. In Sect. 4 the model is evaluated using specific case studies. In Sect. 5 we discuss the results of the evaluation process and conclude the paper in Sect. 6.

2 Related Work

A comprehensive model for evaluating quality of SaaS was proposed by [8]. They achieved this by first defining the key features of SaaS and went on to derive quality attributes for the key features. Metrics were then defined for the quality attributes. They validated their quality model for SaaS by conducting an assessment based on IEEE 106.1. They argued that by using their proposed model, SaaS could be evaluated by service providers. Their evaluation result could also be used as indicator for SaaS quality management. The proposed model however is limited to evaluating quality in SaaS.

Also, a model based on fuzzy logic for assessing SaaS quality was proposed by [10]. They argue that such a model of quality criteria provides a ground for more comprehensive quality model, which may assist a SaaS customer to choose a higher quality service from available services on Cloud. They also argued that the quality model serves as a guideline to SaaS providers enabling them to improve the quality of service provided. Although the study examined quality assessment using an Artificial Intelligence (fuzzy logic) technique, the model proposed does not apply to PaaS.

In addition, researchers in [11] proposed a newer quality model, which addressed the limitation of previously, proposed SaaS quality models. The quality models that existed before their work did not factor in security, quality of service and software quality of the SaaS service. Their proposed quality model is able to clarify SaaS service into four levels, which are: basic level, standard level, optimized level and integrated level. They argued that by using their quality model, a customer could evaluate the provider. The provider on the other hand could use it for quality management. Their quality model however is only suited for SaaS.

All the models reviewed in literature focused on evaluating the quality of SaaS cloud computing delivery model. However, apart from SaaS, there is still the PaaS and IaaS model which both have unique attributes separate from SaaS [12]. As a result, there is need to evaluate quality in these other aspects of cloud computing service delivery models. In this study therefore, we will be confining on PaaS, which is the layer directly below the SaaS layer in the Cloud computing layered architecture (as shown in Fig. 1).

3 The Proposed Model

In order to propose a quality model, there is a need to first identify the factors that affect quality in PaaS. First, developers do not like to be locked in by any technology platform; as such they would go for a platform that allows them a certain level of freedom especially as it relates to programming languages and operating systems. The quality of a PaaS platform therefore is determined by its ability to interact with specified systems (i.e. interoperability) [13]. Also, the degree to which PaaS enables developers to learn its usage affects such developer's perspective about its quality [14]. In addition, the degree to which PaaS can be successfully ported to various operating environment can affect its quality [15]. The model in this section therefore measures the quality of PaaS by examining its quality based on its interoperability, usability and portability. These are collectively referred to as the quality criteria. They are discussed as follows.

3.1 Interoperability

This has been defined by ISO as the attribute of software (PaaS) that bear on its ability to interact with specified systems. A PaaS system would be said to be of more quality if it reduces or eliminates vendor-lock-in [16]. The metrics to measure this in PaaS therefore includes number of operating system supported as well as the number of programming languages supported.

3.2 Usability

This refers to the ease with which a developer is able to master the use of a given platform. Learnability is an important sub-attribute in this regard. It is the degree to which a software product enables users to learn its application [17]. Developers therefore would consider a platform to be of higher quality if such a platform offers documentation and useful learning resources in multiple formats such as wikis, PDF files and video tutorials. Therefore metrics to measure learnability is number of documentation formats available.

3.3 Portability

This refers to the ease with which a system can be transferred from one environment to another. An important sub-attribute in this regard is installability, which refers to the degree of ease with which a software product can be installed and uninstalled in a specified environment. In the context of PaaS, environment can be seen as the operating systems [18]. Metrics to measure installability therefore include installation time on Windows, installation time on Linux and installation time on Mac OS.

The quality model proposed in this section is shown in Fig. 2. The goal of the model is to evaluate PaaS alternatives as depicted at the root of the decision hierarchy. As a result, three quality attributes namely: portability, usability and interoperability (taken from the ISO 25010 product quality model) were considered as evaluation criteria. From Fig. 2, interoperability has two metrics – number of operating systems supported as well as number of programming languages supported. Usability has one sub-criterion called learnability which in turn has a metric – number of documentation formats available. Similarly, portability has one sub-criterion (i.e. installability) as well. Installability being a sub-criteria has three metrics as given in Fig. 2.

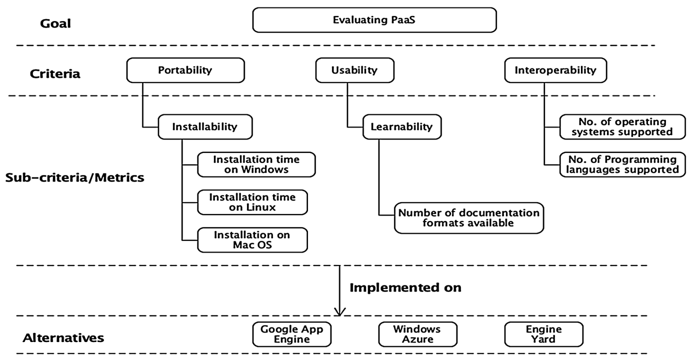

Fig. 2. Proposed model

4 Validation of the Model

There exist quite a number of PaaS offerings today. To evaluate our model, we will be selecting three namely: Google App Engine, Windows Azure and Engine Yard (GWE). These three are well known among developers. We evaluated the model based on data gotten from their respective websites: (cloud.google.com/appengine, azure.microsoft.com, engineyard.com). We applied the Analytic Hierarchy Process technique to the data in order to validate our model. This technique was adopted because it had earlier been used to evaluate quality in the cloud domain [19]. This section discusses the evaluation process.

4.1 Interoperability

This quality criterion that was formerly a sub-attribute of functionality in ISO 9126 was evaluated using two measures. The first measure determines the number of operating systems on which a PaaS solution is able to run [20]. The second determines the number of programming languages supported by the PaaS solution. In order to measure interoperability therefore in GWE the metrics used are given as follows:

Number of operating systems supported: Google App Engine and Windows Azure both support the three major operating system platforms namely: Windows, Mac OSX and Linux/other Unix-based systems. Engine Yard however does not support newer versions of Mac OSX such as 10.9 and 10.10. It only supports the Gentoo Linux distribution[1].

Number of programming languages supported: Windows Azure demonstrated the highest support for six programming languages namely:.NET, Java, Python, Ruby, PHP and Node.js. In addition to the above, we noted that it also provides support for mobile development especially iOS, Android and Windows phone development. Google App Engine and Engine Yard both come in second place as both equally support four programming languages. Google App Engine supports Python, Java, Go and PHP while Engine Yard supports PHP, Ruby, Node.js and Java.

4.2 Usability

The sub-attribute of this quality criterion is learnability as given in ISO 9126. The metric proposed here is the number of documentation formats available which include: wikis, video tutorials and PDF files. For this metric, we observed that all the PaaS platforms considered had just two formats of presenting documentation. The first was as wiki pages on their website and through video. Videos were often used to record tutorials. We discovered that because of these other sources of documentation provided, platform service providers did not see any need to include PDF documentation.

[1] https://support.cloud.engineyard.com/categories/20033681-Engine-Yard-Cloud-Documentation.

4.3 Portability

Installability is the sub-criteria given for this quality criterion. We measure installability through the following metrics namely:

Installation time on Windows: Google App Engine and Engine Yard take the lead in this metric being able to install within 5 min on an average Windows machine running Windows 7. It takes about 30 min however to install on Windows Azure due to the installation of dependencies like SQL Server.

Installation time on Linux: Engine Yard requires only 5 min to install on a typical Linux machine; followed closely by Google App Engine which requires 6 min to install. Windows Azure however takes longer requiring about 25 min to install.

Installation on Mac OSX: In this regard, Google App Engine clearly takes the lead requiring only 3 min to install. Engine Yard takes 15 min to install while Windows Azure takes 25 min.

4.4 Validating the Model with AHP

With reference to Fig. 2, the goal of our metric is to evaluate quality in PaaS and the factors being considered are: Usability, Interoperability and Portability. We now determine the relative ranking of the factors to each other. As given in [22] Usability is more important to the developer followed by Interoperability and then Portability. Hence, Table 1 shows the relative ranking of factors.

Table 1. Relative ranking of the proposed model's criteria

	Usability	Interoperability	Portability
Usability	1/1	2/1	3/1
Interoperability	1/2	1/1	2/1
Portability	1/3	1/2	1/1

From Table 1 we observe a 1/1 relationship (for Usability to Usability, Interoperability to Interoperability and Portability to Portability). The value 2/1 between Usability and Interoperability implies that Usability is twice as important as Interoperability. The value 1/3 between Portability and Usability implies that Usability is thrice as important Portability. Normalizing Table 1 data, we obtain the following matrix (rounding to four decimal places):

$$\begin{bmatrix} 1.0000 & 2.0000 & 3.0000 \\ 0.5000 & 1.0000 & 2.0000 \\ 0.3333 & 0.5000 & 1.0000 \end{bmatrix}$$

We square the normalized result to get the following matrix:

$$\begin{bmatrix} 1.0000 & 2.0000 & 3.0000 \\ 0.5000 & 1.0000 & 2.0000 \\ 0.3333 & 0.5000 & 1.0000 \end{bmatrix} * \begin{bmatrix} 1.0000 & 2.0000 & 3.0000 \\ 0.5000 & 1.0000 & 2.0000 \\ 0.3333 & 0.5000 & 1.0000 \end{bmatrix}$$
$$= \begin{bmatrix} 2.9999 & 5.5000 & 10.0000 \\ 1.6666 & 3.0000 & 5.5000 \\ 0.9166 & 1.6666 & 2.9999 \end{bmatrix}$$

Table 2. Process of obtaining the eigenvector for the first iteration

	Sum of rows	Normalization process	Normalized result
2.9999 + 5.5000 + 10.0000	18.4999	18.4999/34.2490	0.5402
1.6666 + 3.0000 + 5.5000	10.1660	10.1660/34.2490	0.2968
0.9166 + 1.6666 + 2.9999	5.5831	5.5831/34.2490	0.1630
Total	34.2490		1.0000

Summing the rows and normalizing the result we obtain the eigenvector for this first iteration given in Table 2 below.

The values in the normalized matrix are added as shown in the table below which results in the sum of rows. We then normalize the result to by dividing each sum by the sum total; this is the normalization process.

The resulting eigenvector is:

$$\begin{bmatrix} 0.5402 \\ 0.2968 \\ 0.1630 \end{bmatrix}$$

In the next iteration, we square the previous matrix obtained to get the following matrix:

$$\begin{bmatrix} 2.9999 & 5.5000 & 10.0000 \\ 1.6666 & 3.0000 & 5.5000 \\ 0.9166 & 1.6666 & 2.9999 \end{bmatrix} * \begin{bmatrix} 2.9999 & 5.5000 & 10.0000 \\ 1.6666 & 3.0000 & 5.5000 \\ 0.9166 & 1.6666 & 2.9999 \end{bmatrix}$$
$$= \begin{bmatrix} 27.3317 & 49.6655 & 90.2480 \\ 15.0407 & 27.3326 & 49.6655 \\ 8.2770 & 15.0407 & 27.3317 \end{bmatrix}$$

Table 3. Process of obtaining the eigenvector for the second iteration

	Sum of rows	Normalization process	Normalized result
27.3317 + 49.6655 + 90.2480	167.2452	167.2452/309.9334	0.5396
15.0407 + 27.3326 + 49.6655	92.0388	92.0388/309.9334	0.2970
8.2770 + 15.0407 + 27.3317	50.6494	50.6494/309.9334	0.1634
Total	309.9334		1.0000

Summing the rows and normalizing the result we again obtain the eigenvector for the second iteration in Table 3.

Subtracting the eigenvector obtained in the second iteration from the first iteration we get the following, which show that should another iteration be performed, the difference in value will be insignificant.

$$\begin{bmatrix} 0.5402 \\ 0.2968 \\ 0.1630 \end{bmatrix} - \begin{bmatrix} 0.5396 \\ 0.2970 \\ 0.1634 \end{bmatrix} = \begin{matrix} 0.0006 \\ -0.0002 \\ -0.0004 \end{matrix}$$

Thus our resultant eigenvector can be taken as

$$\begin{bmatrix} 0.5396 \\ 0.2970 \\ 0.1634 \end{bmatrix}$$

We now proceed to normalize the proposed metrics.

For the metrics under the Interoperability criteria we have the following as results, which we normalize (see Tables 4 and 5) to get eigenvectors:

The resulting eigenvectors for Interoperability is thus given as:

$$\begin{bmatrix} 0.4286 \\ 0.4286 \\ 0.1429 \end{bmatrix} + \begin{bmatrix} 0.2857 \\ 0.4286 \\ 0.2857 \end{bmatrix} = \begin{bmatrix} 0.7143 \\ 0.8572 \\ 0.4286 \end{bmatrix}$$

For the metrics under usability we have the following Table 6:

The resulting eigenvector for Usability is thus given as:

$$\begin{bmatrix} 0.3333 \\ 0.3333 \\ 0.3333 \end{bmatrix}$$

For the metrics under Portability, Table 7 shows the different installation times on Windows. It takes 5 min for Google App Engine to install, 30 min for Windows Azure and 5 min for Engine Yard. The results are normalized in Table 7.

Table 8 shows the time it takes for each PaaS platforms to be installed on Linux. It takes 6 min for Google App Engine to install, 25 min for Windows Azure and 5 min for Engine Yard. The results are normalized as seen in Table 8.

Table 9 shows the time it takes for each PaaS platforms to be installed on Mac OS. It takes 3 min for Google App Engine to install, 25 min for Windows Azure and 15 min for Engine Yard. The results are normalized as seen in Table 9.

The resulting eigenvector for Portability is given as:

$$\begin{bmatrix} 0.1250 \\ 0.7500 \\ 0.1250 \end{bmatrix} + \begin{bmatrix} 0.1667 \\ 0.6944 \\ 0.1389 \end{bmatrix} + \begin{bmatrix} 0.0697 \\ 0.5814 \\ 0.3488 \end{bmatrix} = \begin{bmatrix} 0.3614 \\ 2.0258 \\ 0.6127 \end{bmatrix}$$

Table 4. Interoperability metric 1

Number of operating systems		
Google App Engine	3/7	0.4286
Windows Azure	3/7	0.4286
Engine Yard	1/7	0.1429
		1.0000

Table 5. Interoperability metric 2

Number of programming languages supported			
Google App Engine	4	4/14	0.2857
Windows Azure	6	6/14	0.4286
Engine Yard	4	4/14	0.2857
	14		1.0000

Table 6. Usability metric

Number of documentation formats			
Google App Engine	2	2/6	0.3333
Windows Azure	2	2/6	0.3333
Engine Yard	2	2/6	0.3333
	6		**1.0000**

Table 7. Portability metric 1

Installation time on windows			
Google App Engine	5 min	5/40	0.1250
Windows Azure	30 min	30/40	0.7500
Engine Yard	5 min	5/40	0.1250
	40		1.0000

Table 8. Portability metric 2

Installation time on Linux			
Google App Engine	6 min	6/36	0.1667
Windows Azure	25 min	25/36	0.6944
Engine Yard	5 min	5/36	0.1389
	36		1.0000

Multiplying the eigenvectors of Interoperability metrics, Usability metrics and Portability metrics with that of the proposed model's criteria gives the following whose interpretation is given in Table 10:

Table 9. Portability metric 1

Installation time on Mac OS			
Google App Engine	3 min	3/43	0.0697
Windows Azure	25 min	25/43	0.5814
Engine Yard	15 min	15/43	0.3488
	43		1.0000

Table 10. Result of validating the proposed model using GWE

PaaS systems considered	Normalized result
Google App Engine	1.4435
Windows Azure	0.8925
Engine Yard	0.4304

$$\begin{bmatrix} 0.7143 & 0.3333 & 0.3614 \\ 0.8572 & 0.3333 & 2.0258 \\ 0.4286 & 0.3333 & 0.6127 \end{bmatrix} * \begin{bmatrix} 0.5396 \\ 0.2970 \\ 0.1634 \end{bmatrix} = \begin{bmatrix} 1.4435 \\ 0.8925 \\ 0.4304 \end{bmatrix}$$

5 Discussion

The challenge of measuring quality in any domain can be regarded as a multiple criteria decision problem. AHP is suited for addressing such problems. It is an approach that allows for objective evaluation and decision making in the midst of multiple criteria. In order to apply AHP, it is important to first define the purpose for adopting it. In this study, the reason was to evaluate quality in PaaS. The next thing to do is to define the quality criteria for PaaS, which include: usability, interoperability and portability. Having performed the aforementioned step, alternatives are selected and in this study, the alternatives that were considered included: Google App Engine, Windows Azure and Engine Yard. These are just three of the several PaaS systems that exist.

The preliminary information specified for AHP is then arranged in a hierarchical tree similar to the structure of our model in Fig. 2. The information is then synthesized to determine the relative rankings of the alternatives. AHP allows for comparison of both qualitative and quantitative criteria and can be compared using informed judgments to derive weights and priorities. The metrics of our model however was strictly quantitative. Pairwise comparison was used to determine the relative importance of one criterion over another. In the PaaS domain, usability is crucial. If the platform were not easy to use (e.g. providing relevant documentation to developers) then it would be difficult to adopt. Interoperability is the next important factor in the PaaS domain and then portability. In order to get the ranking of the priorities from the pairwise matrix, eigenvector solution is used as proposed by Saaty in 1990.

The way to go about obtaining the eigenvector solution is to raise the pairwise matrix to powers that are successively squared each time. The row sums are then calculated and normalized as seen in Tables 2 and 3. The process is discontinued when the difference between these sums in two consecutive calculations is small and insignificant (e.g. 0.0006 or -0.0002). Computing the eigenvector determines the relative ranking of the alternatives under each criterion. Since our model provides only quantitative metrics, the value of each metric is obtained for the alternatives and the result is normalized (as seen in Tables 4, 5, 6, 7, 8 and 9) and used with other rankings.

The result of the validation process using Google App Engine, Windows Azure and Engine Yard as case studies (alternatives) shows that AHP ranks Google App Engine higher (1.4435) in terms of the quality criteria defined in the model. Windows Azure is the next highest ranked of the three alternatives with a score of 0.8925. Engine Yard is ranked with a score of 0.4304 as seen in Table 10.

6 Conclusion and Future Work

In this paper, a model has been proposed for evaluating quality in PaaS. Although such a model has been proposed for SaaS, no literature was seen addressing PaaS. A model was composed by first identifying the factors that affect quality in PaaS. Sub-factors were also identified and metrics proposed drawing from the ISO 25010 model. The proposed model was then evaluated by applying it to measure quality in Google App Engine, Windows Azure and Engine Yard. The results show that Google App Engine is the candidate software based on the evaluation of the proposed model using AHP.

As future work, more factors can be incorporated into the proposed model. Also, more PaaS systems can be evaluated with the resulting model to observe the kind of result it produces compared to this. A tool can also be developed or leveraged to ease the computation process.

Acknowledgements. We acknowledge the support and sponsorship provided by Covenant University through the Centre for Research, Innovation and Discovery (CUCRID).

References

1. Mell, P., Grance, T.: The NIST definition of cloud computing. National Institute of Standards and Technology Special Publication (2011). csrc.nist.gov/publication/nistpubs/800-145/SPSD-1485.pdf
2. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. **34**(1), 1–11 (2011)
3. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **2009**, 599–616 (2009)

4. Chang, V., Wills, G., De Roure, D.: A review of cloud business models and sustainability. In: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 43–50. IEEE (2010)
5. Xu, X.: From cloud computing to cloud manufacturing. Rob. Comput.–Integr. Manuf. **28**(1), 75–86 (2012)
6. Chavan, P., Kulkarni, G.: PaaS cloud. Int. J. Comput. Sci. Inf. Secur. (IJCSIS) **1**, 22–26 (2013)
7. Zissis, D., Lekkas, D.: Addressing cloud computing security issues. Future Gener. Comput. Syst. **28**(3), 583–592 (2012)
8. Lee, J.Y., Lee, J.W., Kim, S.D. A quality model for evaluating software-as-a-service in cloud computing. In: 7th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2009, pp. 261–266. IEEE (2009)
9. Saaty, T.L.: Decision making with the analytic hierarchy process. Int. J. Serv. Sci. **1**(1), 83–98 (2008)
10. Baliyan, N., Kumar, S.: Quality assessment of software as a service on cloud using fuzzy logic. In: IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 1–6. IEEE (2013)
11. Wen, P.X., Dong, L.: Quality model for evaluating SaaS service. In: Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), pp. 83–87. IEEE (2013)
12. Zhang, S., Zhang, S., Chen, X., Huo, X.: Cloud computing research and development trends. In: Second International Conference on Future Networks, pp. 93–97. IEEE (2010)
13. Sheth, A., Ranabahu, A.: Semantic modeling for cloud computing. Part 2 Internet Comput. IEEE **14**(4), 81–84 (2010)
14. Garg, S.K., Versteeg, S., Buyya, R.: A framework for ranking of cloud computing services. Future Gener. Comput. Syst. **29**(4), 1012–1023 (2013)
15. Leavitt, N.: Is cloud computing really ready for prime time? Computer **42**(1), 15–20 (2009)
16. Lewis, G.A.: Role of standards in cloud computing interoperability. In: 46th Hawaii International Conference System Sciences (HICSS), pp. 1652–1661. IEEE (2013)
17. Goyal, P.: Enterprise usability of cloud computing environments: issues and challenges. In: Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), pp. 54–59 (2010)
18. Takabi, H., Joshi, J.B., Ahn, G.J.: Security and privacy challenges in cloud computing environments. IEEE Secur. Priv. **8**(6), 24–31 (2010)
19. Kolb, S.: Towards application portability in platform as a service. In: IEEE 8th International Symposium Service Oriented System Engineering (SOSE), pp. 218–229. IEEE (2014)
20. Godse, M., Mulik, S.: An approach for selecting software-as-a-service (SaaS) product. In: IEEE International Conference on Cloud Computing, CLOUD 2009, pp. 155–158. IEEE (2009)
21. Zhang, Z., Wu, C., Cheung, D.W.: A survey on cloud interoperability: taxonomies, standards, and practice. ACM SIGMETRICS Performance Eval. Rev. **40**(4), 13–22 (2013)
22. Haigh, M.: Research versus practice in software engineering: comparison of expert opinions to measured user priorities (2009)