Journal of Software Engineering and Applications, 2012, 5, 69-75
http://dx.doi.org/10.4236/jsea.2012.52011 Published Online February 2012 (http://www.SciRP.org/journal/jsea)

Design of a Performance Measurement Framework for Cloud Computing

Luis Bautista[1,2], Alain Abran[2], Alain April[2]

[1]Department of Electronic Systems, Autonomous University of Aguascalientes, Aguascalientes, Mexico; [2]Department of Software Engineering and Information Technology, ETS University of Quebec, Montreal, Canada.
Email: lebautis@correo.uaa.mx, {alain.abran, alain.april}@etsmtl.ca

Received December 20th, 2011; revised January 21st, 2012; accepted February 5th, 2012

ABSTRACT

Cloud Computing is an emerging technology for processing and storing very large amounts of data. Sometimes anomalies and defects affect part of the cloud infrastructure, resulting in a performance degradation of the cloud. This paper proposes a performance measurement framework for Cloud Computing systems, which integrates software quality concepts from ISO 25010.

Keywords: Cloud Computing; Measurement; Performance; ISO 25010; COSMIC; Maintenance

1. Introduction

Cloud Computing (CC) is an emerging technology aimed at processing and storing very large amounts of data. It is an internet-based technology in which several distributed computers work together to efficiently process large amounts of information, while ensuring the rapid processing of query results to users. Some CC users prefer not to own the physical infrastructure they are using: instead, they rent cloud infrastructure, or a cloud platform or software, from a third-party provider. These infrastructure application options delivered as a service are known as Cloud Services [1].

One of the most important challenges in delivering Cloud Services is to ensure that they are fault tolerant. Failures and anomalies can degrade these services, and impact their quality, and even the availability. According to Coulouris [2], a failure occurs in distributed systems (DS), like CC systems (CCS), when a process or a communication channel departs from what is considered to be its normal or desired behavior. CCS include all the technical resources clouds have in order to process information, like software, hardware, and network elements, for example. An anomaly is different, in that it slows down part of a CCS without making it fail completely, impacting the performance of tasks within nodes and, consequently, of the system itself.

A performance measurement framework (PMF) for CCS should propose a means to identify and quantify "normal cluster behavior", which can serve as a baseline for detecting possible anomalies in the computers (*i.e.* nodes in a cluster) that may impact cloud performance. To achieve this goal, methods are needed to collect the necessary base measures specific to CCS performance, and analysis models must be designed to determine the relationships that exist among these measures.

The ISO International Vocabulary of Metrology (VIM) [3] defines a *measurement method* as a generic description of a logical organization of operations used in measurement, and an *analysis model* as an algorithm or calculation combining one or more measures obtained from a measurement method to produce evaluations or estimates relevant to the information needed for decision making.

The purpose of a measurement process, as described in ISO 15939 [4], is to collect, analyze, and report data relating to the products developed and processes implemented within the organizational unit, to support effective management of the process, and to objectively demonstrate the quality of the products.

ISO 15939 [4] defines four sequential activities: establish and sustain measurement commitment, plan the measurement process, perform the measurement process, and evaluate the measurement. These activities are performed in an iterative cycle that allows for continuous feedback and improvement of the measurement process, as shown in **Figure 1**.

This work presents a PMF in which the two activities recommended by the ISO 15939 measurement process are developed: 1) establish measurement commitment; and 2) plan the measurement process. This framework defines the requirements for the CC performance measurement, the type of data to be collected, and the criteria for evaluating the resulting information. In future work,

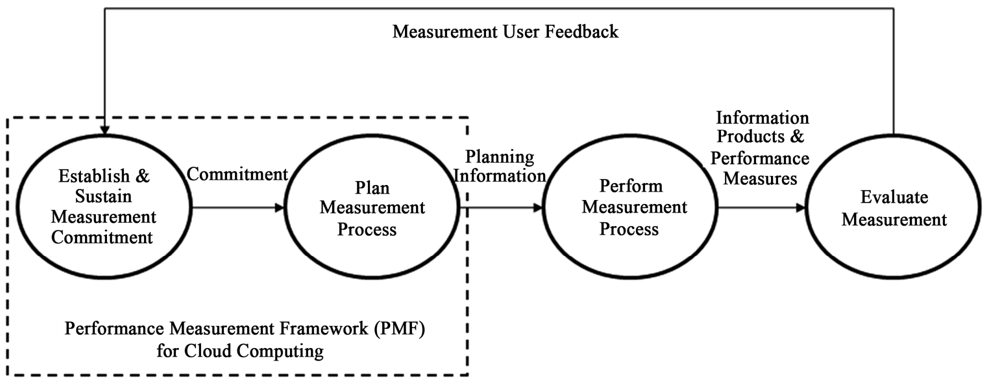
Measurement User Feedback
Establish & Sustain Measurement Commitment
Commitment
Plan Measurement Process
Planning Information
Perform Measurement Process
Information Products & Performance Measures
Evaluate Measurement
Performance Measurement Framework (PMF) for Cloud Computing

Figure 1. Sequence of activities in a measurement process (Adapted from the ISO 5939 measurement process model [4]).

the design of a measurement method and a performance measurement model for CCS will be developed.

This paper is structured as follows. Section 2 presents related work on performance measurement for computer based systems. Section 3 establishes the performance context for CC by defining the basic concepts of performance and developing an overview of the elements involved in the measurement process. Section 4 presents the design of the proposed PMF for CCS using COSMIC concepts. In addition, this section introduces a number of key international standards terms, related to performance, with which we further detail the PMF described in this section. Finally, Section 5 summarizes the contributions of this research and suggests future work.

2. Related Work

2.1. Performance Measurement Approaches for Computer Systems

Currently, the measurement of computer-based system (CBS) performance has been investigated in the computer science literature from the following viewpoints: load balancing, network intrusion detection, and host state maintenance. For example, Burges [5] defines system performance as “normal behavior”, and proposes that this behavior can only be determined by learning about past events and by modeling future behavior using statistics from the past and observing present behavior. According to Burges, modern computing systems are complex: they are composed of many interacting subsystems, which makes their collective behavior intricate and, at the same time, influences the performance of the whole system.

Other authors have tried to predict the performance of complex systems (computer clusters, for example) by simulating cluster behavior using a virtual environment. For instance, Rao [6] estimates the variation of cluster performance through changes in task size, as well as the time taken to solve a particular problem. He has also built a predictive model using regression analysis to investigate the behavior of the system and predict the performance of the cluster.

Other published approaches have focused on the reliability aspects of large, high-performance computer systems in order to measure system performance. Smith [7] observes that failure occurrence has an impact on both system performance and operational costs. He proposes an automatic mechanism for anomaly detection that aims to identify the root causes of anomalies and faults. Smith [7] has also developed an automatic anomaly detection framework that is aimed at processing massive volumes of data using a technique based on pattern recognition. In a case study, Smith identifies health-related variables, which are then used for anomaly detection. Each of these variables is related to a system characteristic (such as user utilization, CPU idle time, memory utilization, I/O volume operations). Once the measurement data have been collected, he proposes clustering categories, where an outlier detector identifies the nodes that potentially have anomalies. Finally, a list of those possible anomalies is sent to a system administrator who has the expertise to quickly confirm whether or not an anomaly exists.

Smith’s research presents interesting avenues for the measurement of system performance from various perspectives. Further work is needed to define an integrated model of performance measurement, which would include the perspectives of users, developers, and maintainers.

2.2. Jain’s System Performance Concepts and Sub Concepts

A well known perspective for system performance measurement is proposed by Jain [8], who maintains that a performance study must first establish a set of performance criteria (or characteristics) to help to carry out the system measurement process. He notes that if a system performs a service correctly, its performance is typically measured using three sub concepts: 1) responsiveness, 2) productivity, and 3) utilization, and proposes a measurement process for each. In addition, Jain notes that for each service request made to a system, there are several possible outcomes, which can be classified in three categories:

the system may perform the service correctly or incorrectly, or it may refuse to perform the service altogether. Moreover, he defines three sub concepts associated with each of these possible outcomes which affect system performance: 1) speed, 2) reliability, and 3) availability. **Figure 2** presents the possible outcomes of a service request to a system and the sub concepts associated with them.

2.3. ISO 25010 Performance Concepts and Sub Concepts

There are several software engineering standards on system and software quality models, such as ISO 25010 [9], which is a revision of the ISO 9126-1 [10] software quality model. The ISO 25010 standard defines software product and computer system quality from two distinct perspectives: 1) a quality in use model, and 2) a product quality model:

1) The quality in use model is composed of five characteristics that relate to the outcome of an interaction when a product is used in a particular context of use. This quality model is applicable to the entire range of use of the human-computer system, including both systems and software.

2) The product quality model is composed of eight characteristics that relate to the static properties of software and the dynamic properties of the computer system.

This product quality model is applicable to both systems and software. According to ISO 25010, the properties of both determine the quality of the product in a particular context, based on user requirements. For example, performance efficiency and reliability can be specific concerns of users who specialize in areas of content delivery, management, or maintenance. The performance efficiency concept proposed in ISO 25010 has three sub concepts: 1) time behavior, 2) resource utilization, and 3) capacity, while the reliability concept has four sub concepts: 1) maturity, 2) availability, 3) fault tolerance, and 4) recoverability. In this research, we have selected performance efficiency and reliability as concepts for determining the performance of CCS. Both Jain's proposal and the ISO 25010 concepts and sub concepts form the basis of our definition of the performance concept in CC.

3. Definition and Decomposition of the Performance Concept for Cloud Computing

3.1. Definition of the Performance Concept for Cloud Computing

Based on the performance perspectives presented by Jain and the product quality characteristics defined by ISO 25010, we propose the following definition of CCS performance measurement:

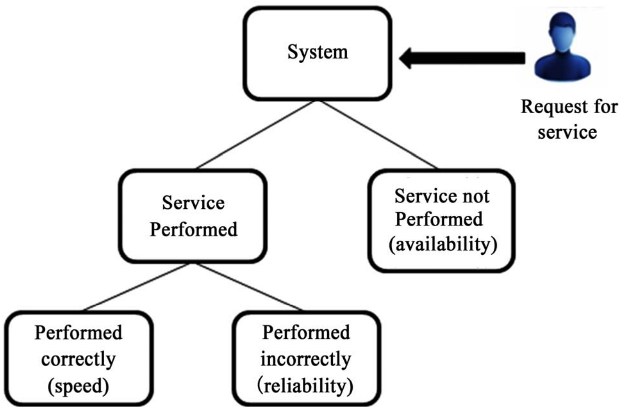
System
Request for service
Service Performed
Service not Performed (availability)
Performed correctly (speed)
Performed incorrectly (reliability)

Figure 2. Possible outcomes of a service request to a system, according to Jain [10].

"*The performance of a Cloud Computing system is determined by analysis of the characteristics involved in performing an efficient and reliable service that meets requirements under stated conditions and within the maximum limits of the system parameters.*"

Although at first sight this definition may seem complex, it only includes the sub concepts necessary to carry out CCS performance measurement from three perspectives: 1) users, 2) developers, and 3) maintainers.

Furthermore, from the literature review, a number of sub concepts have been identified that could be directly related to the concept of performance, such as:

- Performance efficiency: The amount of resources used under stated conditions. Resources can include software products, the software and hardware configuration of the system, and materials.
- Time behavior: The degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
- Capacity: The degree to which the maximum limits of a product or system parameter meet requirements.
- Resource utilization: The degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements.
- Reliability: The degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.
- Maturity: The degree to which a system meets needs for reliability under normal operation.
- Availability: The degree to which a system, product or component is operational and accessible when required for use.
- Fault tolerance: The degree to which a system, product, or component operates as intended, in spite of the presence of hardware or software faults, and,
- Recoverability: The degree to which a product or system can recover data directly affected in the event of an interruption or a failure and be restored to the desired state.

3.2. Definition of a Performance Context Diagram for Cloud Computing

Now that the CCS performance measurement concepts and sub concepts have been identified, a context diagram will be helpful that shows the relationships between the performance sub concepts proposed by ISO 25010 and the performance measurement perspective presented by Jain, as well as the logical sequence in which the sub concepts appear when a performance issue arises in a CCS (see **Figure 3**).

In this figure, system performance is determined by two main sub concepts: 1) performance efficiency, and 2) reliability. As explained previously, when a CCS receives a service request, there are three possible outcomes (the service is performed correctly, the service is performed incorrectly, or the service cannot be performed). The outcome will determine the sub concepts that will be applied for performance measurement. For example, suppose that the CCS performs a service correctly, but, during its execution, the service failed and was later reinstated. Although the service was ultimately performed successfully, it is clear that the system availability (part of the reliability sub concept) was compromised, and this affected CCS performance.

As illustrated above, CCS performance can be based on two main concepts: 1) performance efficiency, and 2) reliability. Performance efficiency will determine the amount of resources used for a period of time, while reliability will determine the degree to which a system successfully performs specified functions during the same period. Resources include all CCS elements, such as: software applications, hardware system, and network system.

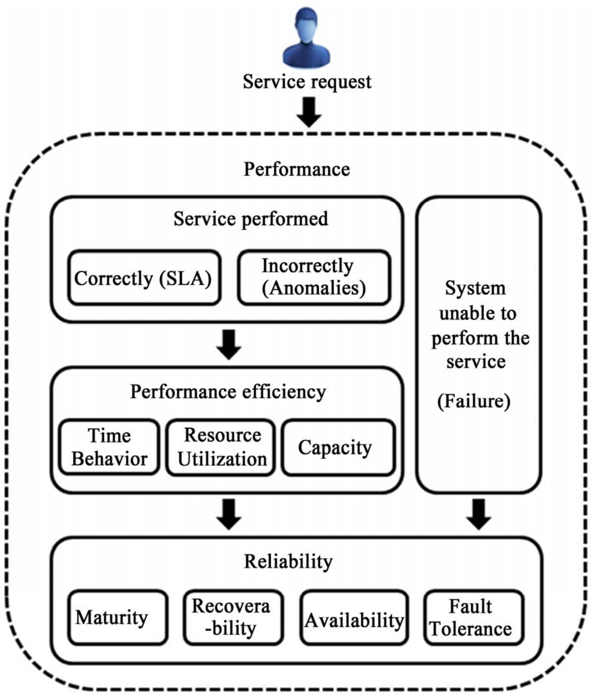

Figure 3. Context diagram for Cloud Computing performance measurement.

4. Design of Performance Measurement Framework for Cloud Computing

4.1. The COSMIC Measurement Method Model

The ISO 19761 COSMIC v 3.0 Functional Size Measurement Method (FSM) [11] defines an explicit model of software functionality derived from the functional user requirements (FUR). FUR describe the functionality that the software or system is to execute (sometimes also known as system capabilities). According to this method, each FUR is represented by one or more functional processes within the piece of software to which it has been allocated. In turn, each functional process is represented by sub processes, which can be of the data movement type or the data transform type.

Based on this explicit model of functionality, four data movement types are recognized (Entry, Exit, Read, and Write). **Figure 4** shows the COSMIC model of generic software adapted from Figure 12.4, p. 256 of [12]).

According to the COSMIC model [12], software is delimited by hardware, as shown on the left-hand side of

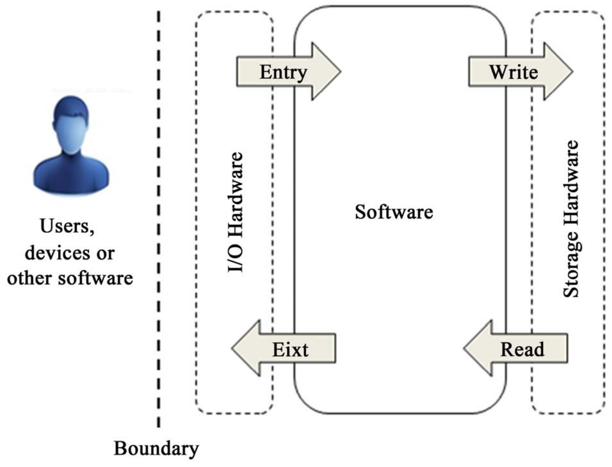

Figure 4. COSMIC model of generic software-adapted from Figure 12.4 of [12].

Figure 4: software can be used by a user, an engineered device, or other software through I/O hardware, such as a keyboard, a printer, a mouse, etc. In addition, as depicted on the right-hand side of **Figure 4**, software is delimited by persistent storage hardware, like a hard disk. Thus, software functionality can be viewed as a flow of data groups characterized Entry, Exit, Read, and Write data movements. The Entry and Exit data movements allow the exchange of data with the user across the I/O hardware/software boundary, and the Read and Write data movements allow the exchange of data between the software and the storage hardware.

4.2. Performance Measurement Framework for Cloud Computing Systems

Sarayreh [13] notes that different abstractions are typically used for different measurement purposes. For example, in real-time software, users are typically replaced by engineered devices that interact directly with the software; that is, the users are I/O hardware. In other domains, like business application software, for example, the abstraction commonly assumes that the users are one or more humans who interact directly with the software, ignoring the I/O hardware/software boundary.

Based on the COSMIC model of generic software and the abstractions mentioned above, our proposed design for the generic PMF for CC is presented in **Figure 5**.

The left-hand side of **Figure 5** presents the CCS. Entries are the detailed attributes that determine CCS performance: for example, system attributes such as memory use, CPU loads, network information, etc., as well as user application attributes, such as successfully performed tasks, error tasks, etc. These attributes are used to quantify the efficiency and reliability concepts through various measurement functions to be able to satisfy functional requirements.

These measurement functions provide an interpretation of the system attributes, *i.e.* they assign values to the properties. A target value of the attribute measure represents a system requirement, *i.e.* the required value of the availability property. Similarly, the actual value of the measurement represents the observed level of satisfaction of the requirement.

So, the above measurement functions share common base measures between the concepts using intermediate Entries and Exits as communication channels. A base measure is the result of the measurement of an attribute obtained through a measurement method. The right-hand side of the figure shows the system that stores the function results to be used to determine CCS performance through, for example, an analysis model.

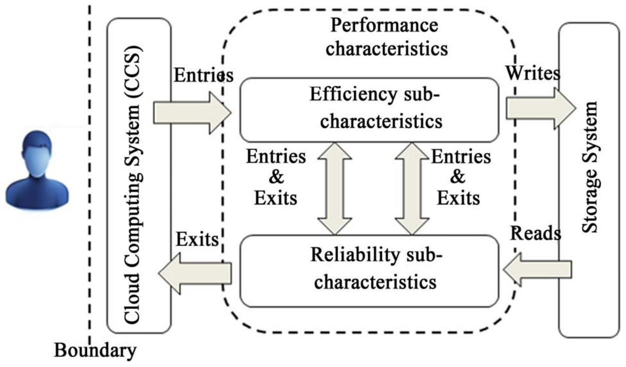
Boundary
Cloud Computing System (CCS)
Entries
Exits
Performance characteristics
Efficiency sub-characteristics
Reliability sub-characteristics
Entries & Exits
Entries & Exits
Writes
Reads
Storage System

Figure 5. Generic performance measurement framework for CC.

4.3. Identification of Terms Associated with Performance

Once the generic PMF for CCS has been defined, the next step is to identify the various terms related to performance concepts (refer to **Figure 3**) that represent the system attributes, and which can be measured to assess whether or not the CCS satisfies the system requirements. We chose the ECSS [14] and ISO 25010 [9] standards to carry out an inventory of requirement terms (both standards are recognized guidelines used in academia and by practitioners to define both hardware and software systems). A number of terms have been identified that represent key aspects of measurement, and these have been included in the proposed performance framework.

These terms are grouped into functions, which are responsible for conducting the measurement process using a combination of base measures. They are associated with the corresponding ISO 25010 quality concepts, as presented in **Table 1**.

Table 1. Terms associated with performance sub concepts.

Term	Functions	ISO 25010 Concepts
Failures avoided Failures detected Failures predicted Failures resolved	Failure function	Maturity Resource utilization Fault tolerance
Breakdowns Faults corrected Faults detected Faults predicted	Fault function	Maturity Fault tolerance
Tasks entered into recovery Tasks executed Tasks passed Tasks restarted Tasks restored Tasks successfully restored	Task function	Availability Capacity Maturity Fault tolerance Resource utilization Time behavior
Continuous resources time Down time Maximum response time Observation time Operation time Recovery time Repair time Response time Task time Time of I/O devices occupied Transmission response time Turnaround time	Time function	Availability Capacity Maturity Recoverability Resource utilization Time behavior
Transmission errors Transmission capacity Transmission ratio	Transmission function	Availability Capacity Maturity Recoverability Resource utilization Time behavior

4.4. Detailed Performance Measurement Framework

Now that the terms related to the performance concepts have been identified and categorized, we position the resulting functions (on the right-hand side of **Figure 6**) in a more detailed view of the PMF for CC. These functions would be interconnected through an intermediate service that shares common base measures, reducing the number of operations in the measurement process at the time of calculation.

The resulting PMF is composed of seven quality concepts which are presented on the left-hand side of **Figure 6**. Each concept is measured using the five basic functions that will share base measures through an intermediate service (IS). This means that the IS will share results from the measurement processes of each function.

It is important to mention that this PMF proposal addresses the generic requirements for capturing the data needed for the context diagram presented earlier, in **Figure 3**. These data can be collected using automated data collection software.

4.5. Example of Measurement of the Availability Concept

Using the proposed PMF for CC, we can demonstrate how to measure the Cloud availability concept. This concept (shown in **Table 1**) is determined by using three functions: 1) the time function, 2) the task function, and 3) the transmission function. For example, a time function can use several different measurements, such as CPU user utilization, job duration, and response time. These measurements are obtained through a measurement method that uses a data collector to obtain the base measures needed. In turn, these base measures are inputted to a time function that calculates a derived measure of the time concept. The intermediate service (IS) combines the derived measures of each function to determine a measurement of the availability concept that contributes to CCS performance, along with the other concept measures defined in the framework.

It is important to mention that this measurement procedure is similar for all the concepts. The measures are then grouped together, depending on the desired perspective: user, developer, or maintainer.

Finally, the performance analysis of the CCS is supported by an analysis model to help interpret the results by relating them to the initial performance requirements.

5. Summary, Contributions, and Future Work

Cloud Computing is an Internet-based technology aimed at processing very large amounts of data in a more efficient way, and one of its most important challenges is to

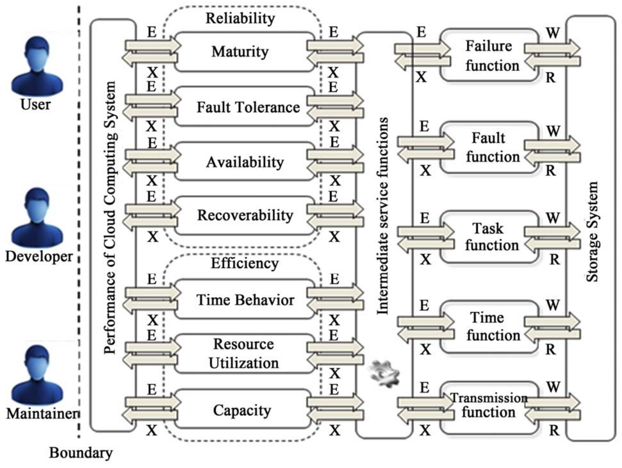

Figure 6. Detailed performance measurement model for CC.

deliver a high level of tolerance to faults and anomalies. This paper proposes a performance measurement framework for Cloud Computing systems. Such framework should define the elements necessary to measure "cluster behavior" using software quality concepts. The design of our framework is based on the concepts of metrology, along with aspects of software quality directly related to the performance concept, which are addressed in the ISO 25010 international standard. We found through our literature search that the performance efficiency and reliability concepts are closely associated with the measurement perspective of Jain. As a result, this research proposal integrates ISO 25010 concepts into Jain's perspective for the performance measurement of information systems. The terminology and vocabulary associated with performance are aligned with many different international standards, such as ECSS. This first performance measurement framework for Cloud Computing systems (see **Figure 5**) was designed based on the COSMIC proposal that models generic software, and includes the functions required to measure performance.

This research work proposes a novel performance measurement framework for Cloud Computing systems. It defines the basis for the design of a measurement method and will make it possible to measure the Cloud Computing concepts that are directly related to performance.

Further research is needed for the design of measurement methods and mechanisms to analyze the performance of a real Cloud Computing application, which could contribute to validating our proposed performance measurement framework. Such evaluation work will include the definition and description of various derived measures which will be mapped to the functions identified during our literature review. We therefore expect that in future research a model will be proposed to analyze the "normal node behavior" of a Cloud Computing Cluster, in order to enable the detection of possible anomalies that affect Cloud Computing system performance.

REFERENCES

[1] H. Jin, S. Ibrahim, T. Bell, L. Qi, H. Cao, S. Wu and X. Shi, "Tools and Technologies for Building Clouds," *Cloud Computing*: *Principles*, *Systems and Applications*, *Computer Communications and Networks*, Springer-Verlag, Berlin, 2010. doi:10.1007/978-1-84996-241-4_1

[2] G. Coulouris, J. Dollimore and T. Kindberg, "Distributed Systems Concepts and Design," Addison-Wesley, 4th Edition, Pearson Education, Edinburgh, 2005.

[3] ISO/IEC Guide 99-12, "International Vocabulary of Metrology—Basic and General Concepts and Associated Terms, VIM," International Organization for Standardization ISO/IEC, Geneva, 2007.

[4] ISO/IEC 15939, "Systems and Software Engineering—Measure," International Organization for Standardization ement Process, Geneva, 2007.

[5] M. Burgess, H. Haugerud and S. Straumsnes, "Measuring System Normality," *ACM Transactions on Computer Systems*, Vol. 20, No. 2, 2002, pp. 125-160. doi:10.1145/507052.507054

[6] A. Rao, R. Upadhyay, N. Shah, S. Arlekar, J. Raghothamma and S. Rao, "Cluster Performance Forecasting Using Predictive Modeling for Virtual Beowulf Clusters," In: V. Garg, R. Wattenhofer and K. Kothapalli, Eds., *ICDCN* 2009, *LNCS* 5408, Springer-Verlag, Berlin, 2009, pp. 456-461.

[7] D. Smith, Q. Guan and S. Fu, "An Anomaly Detection Framework for Autonomic Management of Compute Cloud Systems," *IEEE* 34*th Annual IEEE Computer Software and Applications Conference Workshops*, Seoul, 19-23 July 2010, pp. 376-381.

[8] J. Raj, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling," Wiley-Interscience, New York, 1991.

[9] ISO/IEC 25010:2010(E), "Systems and Software Engineering—Systems and Software Product Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models," International Organization for Standardization, Geneva, 2010.

[10] ISO/IEC 9126-1:2001(E), "Software Engineering—Product Quality—Part 1: Quality Model," International Organization for Standardization, Geneva, 2001.

[11] ISO/IEC-19761, "Software Engineering—COSMIC v 3.0—A Functional Size Measurement Method," International Organization for Standardization, Geneva, 2003.

[12] A. Abran, "Software Metrics and Software Metrology," John Wiley & Sons Interscience and IEEE-CS Press, New York, 2010. doi:10.1002/9780470606834

[13] K. Sarayreh, A. Abran and L. Santillo, "Measurement of Software Requirements Derived from System Reliability Requirements," *Workshop on Advances on Functional Size Measurement and Effort Estimation*, 24*th European Conference on Object Oriented Programming*, Maribor, 20-22 June 2010.

[14] ECSS-E-ST-10C, "Space Engineering: System Engineering General Requirements," European Cooperation for Space Standardization, Requirements & Standards Division, Noordwijk, 2009.