CloudMeasure: A Platform for Performance Analysis of Cloud Computing Systems

Luis Eduardo Bautista Villalpando
Department of Electronic Systems
Autonomous University of Aguascalientes
Aguascalientes, Mexico
E-mail: lebautis@correo.uaa.mx

Alain April
Department of Software Engineering and Information Technology
ETS – University of Quebec
Montreal, Canada
E-mail: alain.april@etsmtl.ca

Alain Abran
Department of Software Engineering and Information Technology
ETS – University of Quebec
Montreal, Canada
E-mail: alain.abran@etsmtl.ca

Abstract— **Concepts such as price, performance, time to completion (availability), probability of failure and non-repudiation are key to developing a comparison service in order to establish Service Level Agreements (SLA) or design better mechanisms to improve performance in Cloud Computing Systems (CCS). This work presents our design of a performance analysis platform that includes an infrastructure, a framework for performance measurement and tools to facilitate the design, validation, and comparison of performance models and algorithms for CCS. The purpose of the platform is to help establish attribute–performance relationships relating to specific applications with relatively well-known demands on systems in order to be able to determine how comparison services may be formulated. The design of the CloudMeasure platform is based on a framework for implementing big data science in organizations (DIPAR) and the three-dimensional performance measurement model for CCS which defines the basis for the analysis of Cloud Computing concepts directly related to performance as identified in international standards such as ISO 25010.**

Keywords— cloud computing; performance; analysis; model; platform; framework; ISO 25010 quality model, maintenance

I. INTRODUCTION

Cloud Computing is defined by ISO and IEC as the paradigm for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable cloud resources accessed through services which can be rapidly provisioned and released with minimal management effort or service provider interaction [1].

Cloud services are categorized into three service models: 1) Infrastructure as a Service (IaaS), 2) Platform as a Service (PaaS), and 3) Software as a Service (SaaS) [2]. Each of these service models includes all the technical resources that clouds need to process information, including software, hardware, and network elements. For example, the service model that most relates to the software engineering community is the SaaS model, while the IaaS model is most related to hardware architectures and virtualization. Software engineers focus on software components, and customers use IT provider applications running on a cloud infrastructure to process information according to their processing and storage requirements. One of the main characteristics of the IaaS model is that customers do not manage or control the underlying Cloud infrastructure (including network, servers, operating systems, and storage), except for limited user-specific application configuration settings.

One of the most important challenges in delivering Cloud services is to ensure they are fault tolerant and minimizes anomalies which can degrade the services or impact quality and even availability. According to Coulouris [3], a failure occurs in a distributed system like a Cloud Computing System (CCS) when a process or a communication channel departs from what is considered to be its normal or desired behavior. An anomaly is different, in that it slows down a part of a CCS without making it fail completely, impacting tasks within nodes and, consequently, affecting system performance.

Performance analysis models (PAMs) for CCS must provide a means to identify and quantify "normal cluster behavior," which can serve as a baseline for detecting possible failures and anomalies in the computers (i.e. nodes in a cluster) that may impact overall cloud performance. To achieve this goal, measurement 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) [4] 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.

An important aspect in the creation of the above models is the data requirement. Data are necessary to carry on experiments by simulating different scenarios in order to select the models that best fit the requirements. Therefore, it is important to have access to performance data repositories to help determine the performance models that subsequently can be implemented in live CCS. In addition, it is necessary to have measurement frameworks which can assist the design process of PAMs for CCS. Currently, however, there are no CCS performance platforms that can help in the design and evaluation of such models.

This paper presents the CloudMeasure platform process, which aims to develop and make available a public platform for CCS performance analysis. Its purpose is to provide a framework, data and infrastructure to facilitate the design, validation, and comparison of performance analysis models and algorithms for CCS. One of the most important aspects of CloudMeasure is integration of the above components within the platform. This integration is very important because it determines the efficiency and reliability of the results obtained from the PAM. In addition, such integration helps to define the group of CCS elements and attributes that determine CCS performance. For instance, one group of elements might be data related to an application running on the CCS such as "Job history," with some of the attributes being "number of successful task executions" or "time taken to process them."

This paper is structured as follows. Section 2 presents related work on platforms for data analysis in software engineering and Cloud Computing. Section 3 presents the proposed CloudMeasure platform architecture and describes its operation and the elements upon which it is based. Section 4 presents the CloudMeasure DataFlow (CMDF) process and defines the steps for designing, developing and validating CCS performance analysis models. Finally, section 5 summarizes the contributions made in this work and suggests future work.

II. Related Work

A. *Software project platforms and repositories*

The CloudMeasure platform provides a conceptual model for PAM design as well as the definition of a standard format for CCS performance attributes and tools for data analysis. In addition, the platform offers a collaborative data loading mechanism that addresses the different participant roles (for example, performance data providers, PAM contributors and infrastructure users).

There are several data repositories publicly available for sharing information related to software projects, web systems or trace data sets. For example, the International Software Benchmarking Standard Group (ISBSG) [5] maintains the largest publicly available repository of software projects. The ISBSG mission is to help to improve software engineering practices and the management of information technology resources by providing project and application data.

Other platforms, such as web metrics repositories [6], define catalogues of web metrics which allow evaluators and stakeholders to have a service and a consultation mechanism to support different phases of the development of the software life cycle in web development. The web metrics repository is used to support different quality assurance processes such as definition and specification of non-functional requirements, understanding and selection of metrics, definition of quality tests, and the development or maintenance phase. In addition, the web metrics repository provides data collection and cataloguing tools for analyzing the web system and improving its performance.

Other data platforms record failure traces to allow comparison and cross validation of a fault-tolerant model or algorithms across identical trace data sets. The Failure Trace Archive (FTA) [7] defines a standard format for failure traces and provides a toolbox that facilitates comparative trace analysis for its users. A failure trace is a data set collected from different distributed systems and containing records of their availability represented in time series. The FTA presents uniform statistical analysis and failure models for nine diverse distributed systems. According to Kondo [7], this can be used to assess whether a model can be generalized to another context (for example in terms of reliability or user base), or to determine which trace data set is better suited or applicable to a given algorithm or model.

Although the above projects present some interesting designs of data repositories, none of them cover Cloud Computing technologies.

B. *Performance Measurement Framework for Cloud Computing*

The Performance Measurement Framework for Cloud Computing (PMFCC) [8] integrates software quality concepts from ISO 25010. This framework defines CCS performance as determined by two main subconcepts: 1) performance efficiency and 2) reliability. In addition, it establishes three possible outcomes to a CCS service request: the service is performed correctly, the service is performed incorrectly, or the service cannot be performed. The outcome will determine which sub-concepts will be applied for performance measurement.

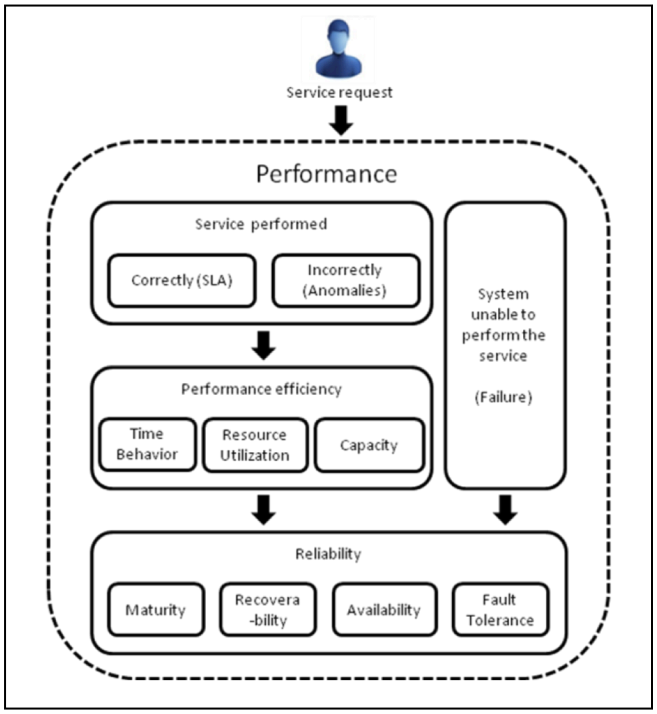

Fig. 1. Quality concepts and sub-concepts associated with CCS performance measurement (adapted from [8]).

For example, suppose the CCS performed 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 system availability (part of the reliability sub-concept) was compromised, and this affected CCS performance. Figure 1 presents the quality concepts and sub-concepts associated with CCS performance measurement.

The performance measurement framework does not define the type of data format related to the CCS. It only defines the concepts that best represent the types of attributes, and which of these can be measured to assess whether or not from a quantitative viewpoint the CCS satisfies the stated requirements. These types of attributes are grouped into performance concepts which serve to conduct the measurement process using a combination of base measures through a data collector. They are associated with the corresponding ISO 25010 quality derived measures, as presented in Table I.

The types of attributes presented in Table I are grouped into performance concepts. These concepts were designed to share intermediate results from common performance measures, reducing the number of operations in the measurement process at the time of calculation.

TABLE I. FUNCTIONS ASSOCIATED WITH CLOUD COMPUTING PERFORMANCE CONCEPTS

Attribute Type	*Performance Concept*	*ISO 25010 Quality Characteristic*
Failures avoided Failures detected Failures predicted Failures resolved	Failure concept	Maturity Resource utilization Fault tolerance
Breakdowns Faults corrected Faults detected Faults predicted	Fault concept	Maturity Fault tolerance
Tasks entered into recovery Tasks executed Tasks passed Tasks restarted Tasks restored Tasks successfully restored	Task concept	Availability Capacity Maturity Fault tolerance Resource utilization Time behavior
Continuous resource utilization time Down time Maximum response time Observation time Operation time Recovery time Repair time Response time Task time Time I/O devices occupied Transmission response time Turnaround time	Time concept	Availability Capacity Maturity Recoverability Resource utilization Time behavior
Transmission errors Transmission capacity Transmission ratio	Transmission concept	Availability Capacity Maturity Recoverability Resource utilization Time behavior

In addition, the framework determines how to measure a quality characteristic. For example, how would one measure the CCS availability characteristic (as presented in Table I)? In this case, three performance concepts are needed: 1) the time concept, 2) the task concept, and 3) the transmission concept. The time concept may use several different measurable attributes, including CPU utilization by the user, job duration, and response time. These measures are obtained using a data collector and then input to the time concept, which then calculates a derived measure of the time. The combination of results for each concept determines a derived measure of availability, which is an aspect of CCS performance as defined in the framework.

One important aspect of the PMFCC is that the type of attribute only defines the group in which performance data are classified. Performance data usually come from a number of sources, such as application logs, database logs, monitoring system tools, etc. This makes it very difficult to know what type of data will be ingested and then used in the PAM. One of the main problems that arises following the ingestion of performance data is its cleanliness. This problem calls for the quality of the data to be verified prior to the performance analysis. Among the most important data quality issues to consider during data cleaning are corrupted records, inaccurate content, missing values and formatting inconsistencies, to name a few.

The quality of the performance data can affect the results of the PAM and, as a consequence, the decision-making process in the various development stages such as organizational requirements definition, process evaluation, interoperability aspects and application design, to name a few.

III. THE CLOUD MEASURE PLATFORM

CloudMeasure is a platform for the design, development and validation of PAM for CCS. Its purpose is to provide a PMFCC, performance data sets and an infrastructure to facilitate the design, validation and comparison of CCS performance models and algorithms. One of the main reasons for the creation of the CloudMeasure platform was the current lack of information to assist the definition and understanding of how to measure availability, reliability and non-repudiation. Actual measurement of concepts such as price, performance, time to completion (availability), likelihood of completion (probability of failure) and penalty (non-repudiation) are key to being able to compare services and to establish Service Level Agreements (SLA) for CCS.

According to Li [9], commercial CCS currently enable the capture of price performance information related to specific applications with relatively well-known demand patterns. This allows the user to gain information useful for comparing services between suppliers. Comparisons can become complex as they may depend on both the user's performance requirements and the current availability of the system, as well as the price the user can afford. According to Gangadharan [10], the pricing of Cloud Computing services is currently associated with differentiated levels of service based on storage capacity, computing units used, and type of platform. The pricing also varies depending on the operating system and the geographical location of the user. Pricing criteria can also be based on hourly usage, CPU cycle usage, or other usage approaches. Gangadharan further mentions that the pricing of Cloud Computing services depends upon levels of use, layers of service, or a mix of these. Maintainers, users and developers should find CloudMeasure useful for defining the performance data to be used in performance models that can help them understand an SLA and analyze CCS performance.

More specifically, the CloudMeasure platform contains the following:

- A PMFCC that defines attributes, concepts and a measurement method for performance analysis.
- Performance data sets, differing in scale and granularity, used to create models for analyzing concepts such as availability, non-repudiation, capacity, etc.
- An infrastructure that consists of a Hadoop cluster and can be used for developing and testing PAMs.
- The CloudMeasure DataFlow (CMDF) process, which defines the workflow for designing and validating PAMs.

A. The CloudMeasure conceptual model

The CloudMeasure platform is based on a conceptual model that defines and describes the data involved in CSS performance analysis. This allows documentation of each measure, resulting in a comprehensive catalogue of attributes than can be used by different stakeholders. The conceptual model is organized hierarchically (see Figure 2) and includes:

- **Performance data design**, i.e., defining the types of performance-related data that constitute the baseline of the performance platform. This stage ensures data quality, which is the basis for the creation of data sets and analysis models.
- **Performance data collection**, i.e., describing the source and procedure for the data collection process. This stage involves identifying the type of CCS architecture, platform, application development frameworks and so on.
- **Performance data analysis**. Among the most important data quality issues to consider during the data analysis stage is data cleaning, i.e., removal of corrupted records, inaccurate content, missing values, and formatting inconsistencies. An important issue in data integration is formatting inconsistencies caused by the different forms that data can take as result of data collection from different architectures, platforms and applications, or the combination of those forms in order to figure out relationships.
- **Development of PAMs**: Once the performance data have been integrated and analyzed, it is necessary to develop models that can be used by stakeholders during the decision-making process. Such models will facilitate understanding of CCS behavior and performance.

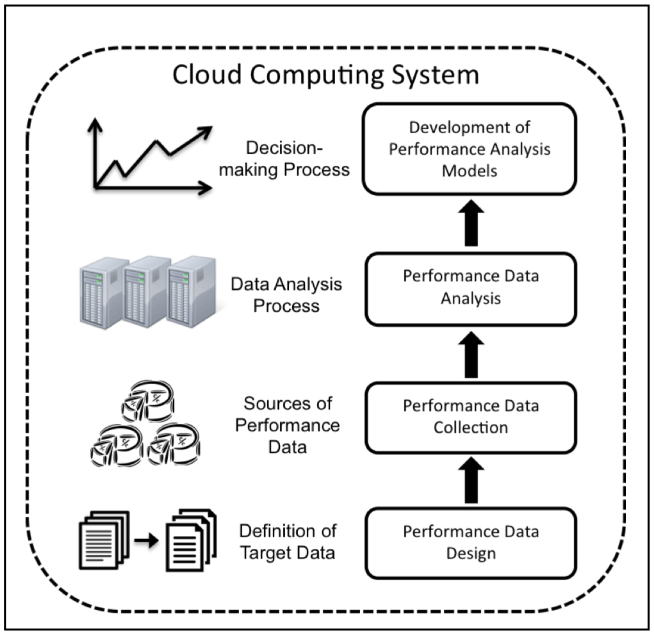
Cloud Computing System
Decision-making Process
Development of Performance Analysis Models
Data Analysis Process
Performance Data Analysis
Sources of Performance Data
Performance Data Collection
Definition of Target Data
Performance Data Design

Fig. 2. CloudMeasure conceptual model.

One important aspect of the CloudMeasure platform is that users can query specific performance data to create models or simply analyze content in the data analysis infrastructure. To facilitate platform design, a template was defined to build a catalogue of performance data that allowed populating and updating a data performance repository.

B. The CloudMeasure template data content

Performance data quality is a key aspect of the CloudMeasure platform. Enhancing the quality of performance data is important for a number of reasons, including:

- Defective data will produce erroneous results in the PAM, contributing to an unsatisfactory decision-making process.
- The dispersion of performance data among different sources of CCS, such as hypervisors, individual virtual machines (VMs), VM information scheduling across multiple hardware cores, etc., does not provide a coherent and integrated vision for performance analysis.
- The coexistence of legacy architectures, Cloud platforms and application development frameworks is normally governed by different standards.

One of the goals of CloudMeasure is to improve the quality of performance data regardless of the Cloud architecture, platform or application development framework used in the CCS. This improvement can be achieved by identifying certain data quality requirements to ensure the quality of the information used in PAM design and validation.

The CloudMeasure data platform is based on ISO 25012: Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) – Data Quality

Model [11]. This standard defines a data quality model that focuses on the quality of the data as part of a computer system and defines quality characteristics for target data used by users and operators of the system. According to ISO 25012, target data are data that the organization chooses to be analyzed and validated through the model; in this case, data related to CCS performance. Moreover, ISO 25012 describes characteristics that may be used as a term of reference to define data quality requirements and data quality measures to establish criteria for the assessment and evaluation of the quality of the data managed by a computer system according to the organization's objectives.

One of the main aspects considered for design of a standard template was to categorize each attribute to be measured under the different ISO quality concepts described in the PMFCC (see Figure 1). For this case: the design of a Three-Dimensional Performance Measurement Model for Cloud Computing (P2M2C-3D), which defines data types and a measurement method that allows measurement of the CC concepts directly related to performance from different perspectives.

C. *A Three-Dimensional Performance Measurement Model for Cloud Computing*

In the Three-Dimensional Performance Measurement Model for Cloud Computing (P2M2C-3D), performance concepts were reviewed and updated according to different types of perspectives as shown in Table II where the different sub-concept measurements are defined according to each perspective.

For example, the measurement of the sub-concept of time behavior from perspective 1 ($TBMP_1$) corresponds to the provider, while the same type of measurement from perspective 2 ($TBMP_2$) corresponds to the customer perspective, and so on.

Each group of sub-concept measurements was combined according to the same perspective number to obtain its respective concept indicator (see Figure 3). For example, the measurements for the sub-concepts MMP_1, RMP_1, AMP_1 and $FTMP_1$ were combined to obtain the indicator for the reliability concept (RI_1). Similarly, the measurements for the sub-concepts $TBMP_1$, $RUMP_1$ and CMP_1 were combined to obtain the indicator of the performance efficiency concept (PEI_1). Finally, the indicators related to reliability and performance efficiency concepts, were integrated to obtain a Key Performance Indicator (KPI_1) from perspective 1, also called provider perspective.

The Key Performance Indicator values (KPI_1, KPI_2 and KPI_3) shown in Figure 4 represent the values of the three dimensions, each placed on a side of a tetrahedron describing a sloped plane section (or hyper plane) in the space providing a quantitative assessment. The key indicators KPI_1, KPI_2 and KPI_3, representing the perspective of provider, customer and user, are shown as different dimensions of the regular tetrahedron.

TABLE II. PERSPECTIVES FOR PERFORMANCE MEASUREMENT CONCEPTS IN CLOUD COMPUTING

Performance Measurement Concept	*Sub concept measurement*	*Description*
Performance efficiency		
	$TBMP_1$	Time behavior measurement - provider perspective
	$TBMP_2$	Time behavior measurement - costumer perspective
	$TBMP_3$	Time behavior measurement - user perspective
	$RUMP_1$	Resource utilization measurement - provider perspective
	$RUMP_2$	Resource utilization measurement - costumer perspective
	$RUMP_3$	Resource utilization measurement - user perspective
	CMP_1	Capacity measurement - provider perspective
	CMP_2	Capacity measurement - costumer perspective
	CMP_3	Capacity measurement - user perspective
Reliability		
	MMP_1	Maturity measurement - provider perspective
	MMP_2	Maturity measurement – costumer perspective
	MMP_3	Maturity measurement - user perspective
	AMP_1	Availability measurement - provider perspective
	AMP_2	Availability measurement - costumer perspective
	AMP_3	Availability measurement - user perspective
	$FTMP_1$	Fault tolerance measurement - provider perspective
	$FTMP_2$	Fault tolerance measurement - costumer perspective
	$FTMP_3$	Fault tolerance measurement - user perspective
	RMP_1	Recoverability measurement from provider perspective
	RMP_2	Recoverability measurement from costumer perspective
	RMP_3	Recoverability measurement from user perspective

IV. CLOUDMEASURE DATAFLOW

CloudMeasure DataFlow (CMDF) is a process for performance data collection, data analysis, analysis model processing and delivery of results. It is designed to organize the diverse and complex dataflows generated by the different measures contributing to CCS performance. The CMDF is based on the Framework for Implementing Big Data Science in Organizations (DIPAR) [12].

A. *DIPAR: A Framework for Implementing Big Data Science in Organizations*

The DIPAR framework is a means to implement Big Data Science (BDS) in organizations and defines its requirements and elements. The framework consists of five

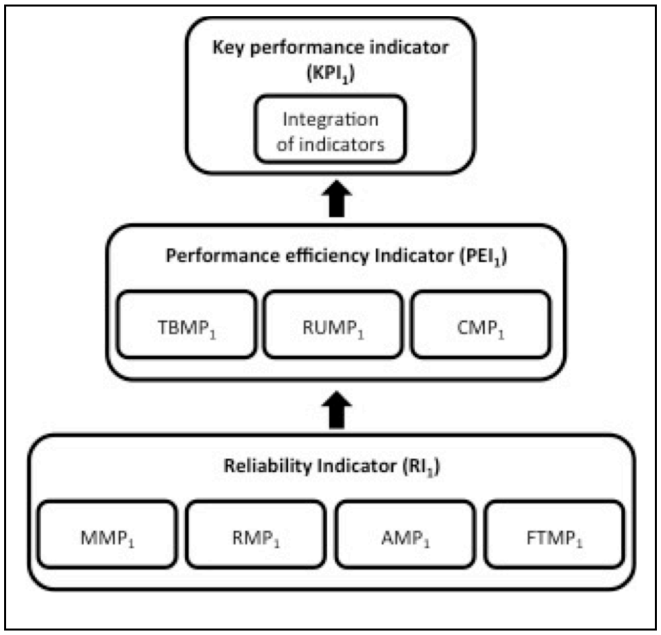
Key performance indicator (KPI1)
Integration of indicators
Performance efficiency Indicator (PEI1)
TBMP1
RUMP1
CMP1
Reliability Indicator (RI1)
MMP1
RMP1
AMP1
FTMP1

Fig 3. Key Performance Indicator from the provider perspective.

stages: define, ingest, preprocess, analyze, and report. Based on the ISO 15939 Systems and software engineering – Measurement process standard [13], its purpose is to collect, analyze, and report data relating to product development.

The DIPAR framework integrates the four activities described in ISO 15939. Its main objective is to design big data products that have a high impact on organizational performance. Figure 5 depicts the order of the five stages to be executed during implementation of the DIPAR framework.

The stages of the DIPAR framework and the elements involved are:

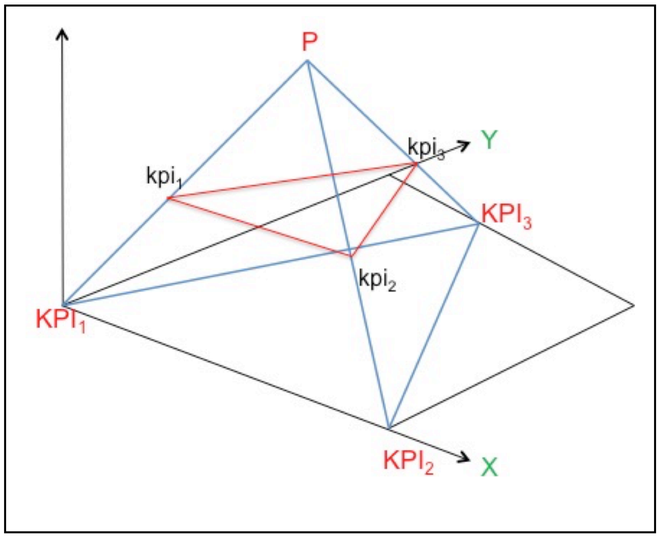

Fig. 4. Regular tetrahedron with KPI1, KPI2 and KPI3 dimensions representing the perspectives of provider, customer and user.

- **Define**: The first step is to determine whether or not a new product is necessary. If not, all the analytical work developed to create the product will be a waste of time and resources. For our case study, it was necessary to define the data needed to develop the product such as a new PAM.

- **Ingest:** One of the main challenges of ingesting the system is to define the ingestion sources. Frequently, data come from a number of sources, such as web logs, databases, different types of applications, etc., which makes it difficult to know what type of data will be ingested by the system. One solution is to use Big Data (BD) software designed specifically to collect and aggregate data from different sources. Projects like Flume [14] and Scribe [15] allow large amounts of log data to be collected, aggregated, and moved from many different sources to a centralized data store.

- **Processing:** One of the main issues that arises following ingestion is cleanliness of the data. Data quality needs to be verified prior to performing data analysis. Consequently, one of the main challenges at the preprocessing stage is how to structure data in standard formats so as to be analyzed more efficiently. This is often easier said than done. During the process of structuring and merging data into common formats, there is a risk of losing valuable information.

- **Analysis:** Once the data have been preprocessed, they are analyzed to obtain relevant results. For this, it is necessary to develop models that can be used in the creation of new products. One of the main problems arising during the design of such models is to recognize which of the available data are the most relevant to an analysis task. Once it becomes feasible to develop complex models and algorithms for data analysis, products with added value for the organization can be created.

- **Report:** Once data are ingested, processed, and analyzed, users need to be able to access and evaluate the results, which must be presented in a way that is readily understood. Here, analysis model results are evaluated as part of a decision-making process in order to arrive at relevant conclusions or design of new analysis models.

B. *CloudMeasure DataFlow*

CloudMeasure DataFlow (CMDF) was designed to define the workflow to be performed during PAM design, development and validation. One of the objectives of CMDF was to accelerate the definition of performance data, data collection, processing, analysis and reporting of results, where different data sources are distributed over a geographically disperse network. Figure 6 shows the stages and elements that constitute the CMDF process.

Initially, performance measures from several different CCS (CCS_A, CCS_B and CCS_C) are extracted to the

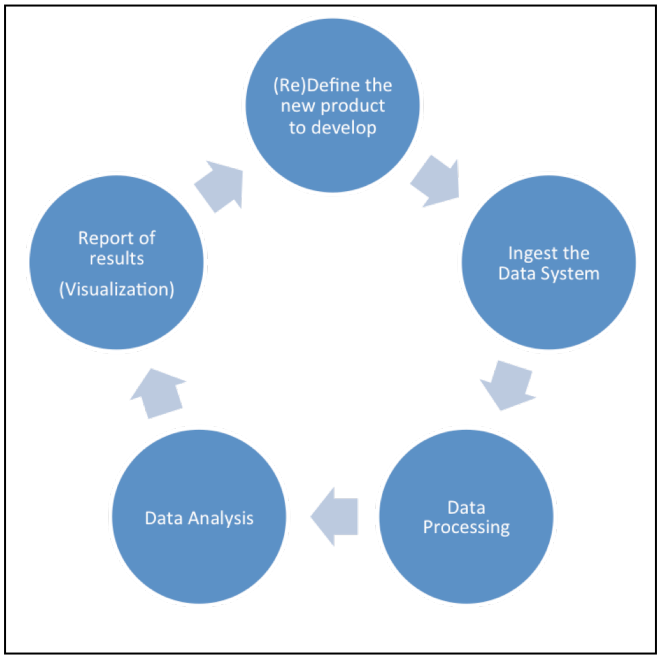

Fig. 5 Stages of development during implementation of the DIPAR framework (adapted from [12]).

CloudMeasure platform by means of different Application Program Interfaces (APIs) and are the baseline for PAM design and development. It is important to mention that such APIs are developed for third-party entities such as CCS providers.

The first stage of the CMDF process is *performance data definition*. In this stage, performance data measures extracted from various CCS providers are organized according to the performance concepts and sub-concepts defined in P2M2C-3D (see Table II). For example, the performance measure *CPU utilization* is categorized as a measure of the *Resource Utilization* sub-concept from one of the perspectives. The categorization of all extracted performance measures is necessary in order to design, develop and validate different PAMs.

Once the performance data have been defined, the next step is storage, which is performed in the *data collection* stage. Here, data are stored in some type of distributed storage such as the Hadoop Distributed File System (HDFS) [16] or some type of NoSQL database such as Hbase [17]. This type of storage is important in order to reduce the cost of data processing.

The next stage is *data processing*, which consists in verifying data cleanliness, or quality, prior to data analysis. As mentioned, one of the main challenges at this stage is how to structure data in standard formats so that they can be analyzed more efficiently.

In the next step, *data analysis*, the PAM is designed, developed and validated. Here, statistical methods and machine learning techniques are used to analyze CCS performance. This is done using data processing tools available in the CloudMeasure platform such as MapReduce [18], Hive [19] or Mahout [20].

Finally, the last stage is *reporting of the results* obtained from the PAM designed in the previous stage. These results will be useful for creation of SLAs, design and development of new analysis models, or improving CCS performance. It is important to mention that CMDF is an iterative process, that is, each stage depends on the previous one and, as a consequence, it will sometimes be necessary to go back to some specific stage in order to ensure the quality of the data input and output.

V. Conclusions and Future Work

The CloudMeasure platform was developed to contribute to the design, development and validation of Performance Analysis Models for CCS. CloudMeasure supports analysis of the performance characteristics of a CCS during the development, maintenance and operational stages.

Using information contained in the platform allows end users to better understand and compare the relationships between performance attributes of different CCS. Moreover, the platform assists the design, validation and comparison of various PAMs (and algorithms), which can then be used in the design of SLA.

One of the main issues in formulating an SLA is how to capture price-performance information relating to specific applications with relatively well-known demands on systems so as to determine how such a comparison service may be formulated. In this way, the CloudMeasure platform is an efficient tool that helps to create models for price-performance comparisons.

Further work is needed to validate the CMDF process within the CloudMeasure platform by reviewing and increasing the number of performance data sets. In addition, new tools for creating Performance Analysis Models need to be developed and tested, which would contribute to CCS performance analysis as well as validating the proposed methodologies.

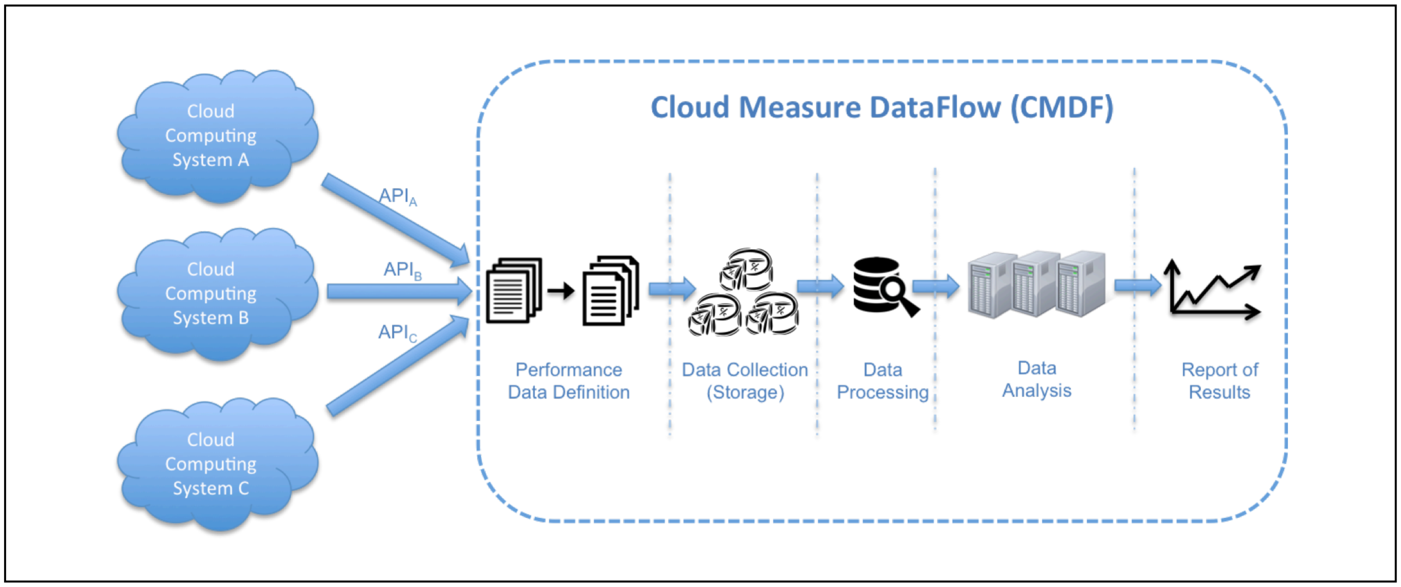

Fig. 6 Stages and elements that constitute the CMDF process.

REFERENCES

[1] ISO/IEC, "ISO/IEC JTC 1 SC38:Cloud Computing Overview and Vocabulary," ed. International Organization for Standardization, Geneva, Switzerland, 2012.

[2] ISO/IEC, "ISO/IEC JTC 1 SC38:Study Group Report on Cloud Computing," ed. International Organization for Standardization, Geneva, Switzerland, 2011.

[3] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems Concepts and Design*, 5th ed. Edinburgh: Addison Wesley, 2011.

[4] I. I. G. 99-12, "International Vocabulary of Metrology – Basic and General Concepts and Associated Terms, VIM," International Organization for Standardization ISO/IEC, Geneva, Switzerland, 2007.

[5] ISBSG, " ISBSG dataset Release 11," ed. International Software Benchmarking Standards Group Limited, Australia, 2012.

[6] L. Olsina, G. Lafuente, and O. Pastor, "Towards a reusable repository for web metrics," *J. Web Eng.*, vol. 1, pp. 61-73, 2002.

[7] D. Kondo, B. Javadi, A. Iosup, and D. Epema, "The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, 2010, pp. 398-407.

[8] L. Bautista, A. Abran, and A. April, "Design of a Performance Measurement Framework for Cloud Computing," *Journal of Software Engineering and Applications*, vol. 5, pp. 69-75, February 2012 2012.

[9] B. Li, L. Gillam, and J. O'Loughlin, "Towards Application-Specific Service Level Agreements: Experiments in Clouds and Grids," in *Cloud Computing: Principles, Systems and Applications, Computer Communications and Networks*. vol. I, ed London: Springer-Verlag, 2010, pp. 361-372.

[10] G. R. Gangadharan and D. M. Parrilli, "Service Level Agreements in Cloud Computing: Perspectives of Private Consumers and Small-to-Medium Enterprises," in *Cloud Computing for Enterprise Architectures, Computer Communications and Network*. vol. I, Z. Mahmood and R. Hill, Eds., ed London: Springer-Verlag, 2011, pp. 207-225.

[11] ISO/IEC, "ISO/IEC 25012: Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) - Data Quality Model," ed. International Organization for Standardization, Geneva, Switzerland , 2008, p. 18.

[12] L. E. Bautista Villalpando, A. April, and A. April, "DIPAR: A Framework for Implementing Big Data Science in Organizations," in *Continued Rise of the Cloud: Advances and Trends in Cloud Computing*. vol. I, Z. Mahmood, ed. Springer-Verlag London, London, England, 2014, pp. XIX, 410.

[13] ISO/IEC, "ISO/IEC 15939:2007 Systems and software engineering — Measurement process," ed. International Organization for Standardization, Geneva, Switzerland, 2008.

[14] A.F.S. (2012, June 13). *Apache Flume*. Available: http://flume.apache.org/

[15] Facebook. (2012, June 13). *Scribe*. Available: https://github.com/facebook/scribe/wiki

[16] B. Dhruba. (2010). *Hadoop Distributed File System Architecture*. Available: http://hadoop.apache.org/docs/r0.20.2/hdfs_design.htm l

[17] A.S.F. (2013, June 6th). *Apache HBase, the Hadoop database, a distributed, scalable, big data store*. Available: http://hbase.apache.org/

[18] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107-113, January 2008 2008.

[19] A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, *et al.*, "Hive-a petabyte scale data warehouse using Hadoop," in *26th International Conference on Data Engineering*, Long Beach, California, USA, 2010, pp. 996-1005.

[20] A.S.F. (2012). *What is Apache Mahout?* Available: https://cwiki.apache.org/confluence/display/MAHOUT /Overview