Modeling End User Performance Perspective for Cloud Computing Systems Using Data Center Logs from Big Data Technology

BY

Anderson RAVANELLO

THESIS PRESENTED TO ECOLE DE TECHNOLOGIE SUPERIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY PH.D.

MONTREAL, OCTOBER 16, 2017

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alain April, Thesis Supervisor
Department of Software Engineering and IT, École de Technologie Supérieure

Mr Abdelaoued Gherbi, Thesis Co-supervisor
Department of Software Engineering and IT, École de Technologie Supérieure

Mr Ambrish Chandra, President of the Board of Examiners
Department of Electrical Engineering, École de Technologie Supérieure

Mr Stephane Coulombe, Member of the jury
Department of Software Engineering and IT,École de Technologie Supérieure

Mrs. Cherifa Mansoura Liamani, external evaluator

Business Architect at TEKsystems

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

SEPTEMBER 5TH, 2017

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENTS

This thesis would not have been possible without the guidance of my research supervisors, Professors Alain April and Abdelouahed Gherbi, which frequently provided constructive feedback and direction on the work to be done. I am also grateful for the assistance provided by Professors Alain Abran and Jean-Marc Desharnais who patiently reviewed my work on the published papers and to Mrs. Rosalia Falco for the careful proofreading.

I owe my deepest gratitude to my family who provided the stable background and environment without which would have made this work even harder to accomplish.

MODÉLISER LA PERSPECTIVE DE L'UTILISATEUR FINAL CONCERNANT LA PERFORMANCE DE LOGICIELS INFONUAGIQUES À L'AIDE DES REGISTRES DE JOURNAUX DU CENTRE DE DONNÉES ET DE TECHNOLOGIES DE DONNÉES MASSIVES (BIG DATA)

Anderson RAVANELLO

RÉSUMÉ

La mesure de la performance des systèmes d'information obsède encore les ingénieurs en logiciel depuis que s'est développé ce champ de recherche. Progressivement, de nombreuses techniques et méthodologies se sont développées en vue d'aider les ingénieurs et les entreprises à mieux comprendre, à gérer et à améliorer les performances que les utilisateurs finaux perçoivent lorsqu'ils utilisent les systèmes d'information lors de leurs opérations quotidiennes. Certaines techniques de mesure de la performance se servent d'enquêtes pour examiner quels aspects de la performance va satisfaire ou ne satisferont pas aux exigences des utilisateurs finaux. D'autres techniques de mesure de la performance proposent de dupliquer les mêmes opérations entre différents services afin de comparer les performances en utilisant un ensemble de tâches similaires. Une autre manière d'aborder le problème a été expérimentée et a permis de modifier légèrement les données échangées entre les clients et les serveurs afin d'inclure des composantes qui vont favoriser le suivi détaillé de la performance en tenant compte ce de qu'a été modifié.

L'approche de mesure à l'aide d'enquêtes n'offre pas d'informations détaillées en ce qui a trait au temps de la journée, à l'humeur du répondeur et à de nombreux autres facteurs humains, dissimulant ainsi une cause fondamentale qui ne favorise pas suffisamment l'interaction qui permettrait des actions préventives et correctives lorsqu'un problème de performance se dessine. D'autres recherches proposent l'utilisation de techniques de simulation à titre de solution potentielle permettant ainsi de produire des informations précises en ce qui concerne les caractéristiques de la performance d'un logiciel, mais la faiblesse de cette approche est qu'elle ne permet pas la participation réelle de l'utilisateur. De plus, les simulations, dans l'état actuel de la technologie, ont du mal à refléter avec précision la complexité du raisonnement

derrière les décisions de l'utilisateur concernant l'emploi ou non d'un système d'information selon le cheminement suivi par chacun des utilisateurs. Finalement, les techniques proposées qui manipulent les données transactionnelles, entre clients et serveurs, ont comme effet secondaire d'influencer de manière négative la performance, la confidentialité et la réfutabilité des données. Si un système de modélisation de la performance modifie des données transactionnelles (même le plus légèrement), ce système pourrait causer des problèmes ce qui en décourage généralement l'utilisation.

Dans cette thèse, la question fondamentale qui se pose est de savoir si la performance telle que perçue par des utilisateurs finaux qui se servent d'applications opérant sur l'infonuagique peut être modélisée de manière fidèle et peu ou non intrusive de sorte que l'information puisse être analysée et utilisée pour la prévention et la correction de problème le plus rapidement possible. Pour modéliser fidèlement la notion de performance du point de vue d'un utilisateur final, la solution considérée inclut la combinaison d'une analyse exhaustive des requêtes interactives qui tient compte de l'état de l'environnement opérationnel et du suivi détaillé de la manipulation des données tout en minimisant les faiblesses de chacune de ces techniques.

Au fur et à mesure que les entreprises mettent en place des infrastructures et des systèmes d´infonuagique, les difficultés de mesure de la performance sont influencées par de nombreux facteurs, mais surtout celui de la complexité accrue de ces systèmes qui se trouvent à être beaucoup plus complexes que les technologies précédentes. L'infonuagique souffre actuellement d'un manque d'outils de mesure de la performance telle que perçu par l'utilisateur; influencé par des aspects sociotechniques tels que les connaissances techniques, la fiabilité, la performance du système, la disponibilité et l'efficacité. (Armbrust, Fox, & Griffith, 2009; Gruschka & Jensen, 2010; Grobauer, Walloschek, & Stocker, 2011)

Afin de pouvoir relever ce défi, la solution proposée et expérimentée lors de cette recherche vise à mieux utiliser les journaux de performance disponibles au sein des centres de données. Ces journaux de performance sont très répandus et contiennent des données textuelles mettant en évidence différentes consommations de ressources ainsi que des activités réalisées dans

différentes composantes opérationnelles des applications opérant sur l'infonuagique. La journalisation est largement utilisée dans l'industrie pour le dépannage et les enquêtes concernant les problèmes de performance ponctuels. Dans cette thèse, les données de ces journaux sont explorées de façon plus approfondie afin de répondre au besoin de précision, de granularité et de réactivité dans un court temps de décision requis de façon à répondre au défi d'une modélisation fidèle de la performance telle que perçue par l'utilisation et potentiellement pour sa prédiction. La quantité et la granularité des données recueillies dans ces journaux sont actuellement massives. Chacune des composantes informatiques et chacune des composantes du réseau utilisé par une application infonuagique analysées pourraient générer jusqu'à 800 Ko de données par minute. Cette quantité massive de données est difficile à traiter et à explorer à l'aide de technologies traditionnelles. Afin de résoudre cette problématique, l'utilisation de technologies Big Data émergentes, tel que Hadoop Distributed File System et Apache Spark sont utilisées afin de collecter et traiter, en temps réel, les données de toutes ces sources et de traiter simultanément ces nombreux journaux.

Cette thèse propose un nouveau modèle de mesure de la performance telle que perçue par les utilisateurs finaux permettant de modéliser la performance d'un système d'information opérant sur l'infonuagique. Ce modèle possède de nombreuses applications pratiques dans le domaine de la mesure du niveau de service et de la prévision de la performance. Il utilise un faible sous-ensemble de données de mesures de journaux (c.-à-d. des mesures directes et des mesures dérivées) qui s'avèrent plus significatives et exploitables afin de modéliser fidèlement et rapidement la performance d'un système d'information telle que perçues par son utilisateur final. Ce nouveau modèle de performance proposé a été inspiré du modèle théorique de la mesure infonuagique et des caractéristiques de qualité préalablement proposée par Bautista (Bautista, Abran, & April, 2012). Ce nouveau modèle de mesure de la performance d'un logiciel infonuagique telle que perçue par l'utilisateur final a été validé à l'aide d'une quantité massive de données provenant de journaux de transactions et fait une extension des principes de mesure suggérés par Bautista en proposant l'utilisation d'un indicateur de performance et en ajoutant une interaction avec les utilisateurs finaux afin de confirmer la perception pressenti par ce modèle.

Une expérimentation à grande échelle sur un cas de figure réel d'une grande entreprise qui possède des logiciels opérant sur une infrastructure infonuagique privée a été réalisée et décrite. Cette expérimentation du modèle proposé, dans laquelle des mesures sont analysées à partir d'une infrastructure Big Data moderne, a permis de confirmer qu'il est possible de modéliser fidèlement la performance perçue par l'utilisateur final; cette performance étant représentée comme un ensemble d'indicateurs de performance basés sur l'accord de niveau de service pour l'infonuagique étudié. Cette recherche permet finalement d'apporter une solution à la question de la modélisation de la performance d'un logiciel telle que perçue par les utilisateurs finaux. Elle offre une avenue de solution pour la mise en œuvre de la mesure lors d'ententes de service pour les applications logicielles opérant sur l'infonuagique de manière à favoriser l'analyse de données de performance dans un court laps de temps. Le modèle ouvre aussi la porte à la mise en place future d'algorithmes de prévision de la performance.

Mots-clès : performance, infonuagique, big data, spark

MODELING END USER PERFORMANCE PERSPECTIVE FOR CLOUD COMPUTING SYSTEMS USING DATA CENTER LOGS FROM BIG DATA TECHNOLOGY

Anderson RAVANELLO

ABSTRACT

Information system performance measurement has been a concern for software engineers since the early days of the field's development. Over time, numerous techniques and methodologies have been developed that help engineers and companies better understand, manage and improve the performance that the end users perceive when using information systems in their daily operations. Some performance measurement techniques employ surveys that investigate which aspects satisfy or do not satisfy end user requirements. Other performance measurement techniques simulate the same operations across different services in order to compare performance given a similar workload. Yet another approach that has been experimented slightly modifies the data exchanged between clients and servers in order to include components that help with tracing the performance of different operation statuses.

When we consider surveys or questionnaires as a performance measurement technique, they do not include detailed information about the sources of problems that may be impacted by the time of day, the responder's mood and many other human factors, thus masking the root cause and they are not sufficiently interactive to allow for a timely reaction when there is a performance problem. Simulation is also proposed as a potential solution where the same operations, over different platforms, correctly report fundamental characteristics of performance. This approach however, removes the user's perspective. It is difficult to assume that a simulation would be able to, with the current state of technology, accurately reflect the complexity of a user's reasoning and decisions regarding the use of a specific information system in a particular way. Finally, the manipulation of the transactional data between client and hosts could affect the confidentially and refutability of the data used to determine the performance; if an information system includes the possibility of data being modified, even slightly, the end user could lose trust in it, negatively affecting the human-machine relation.

The question that is considered here is how can the end user performance perspective of cloud computing-based applications be modeled in a way so that timely analysis can be enacted upon the information? The best possible solution for understanding performance from the end user's perspective could emerge from combining the completeness of interactive surveys with the controlled environment of simulations and the traceability of packet manipulation, while minimizing the weaknesses of each of these techniques. As companies continue to rollout cloud computing infrastructures and systems, the difficulty with performance measurement increases due to a number of factors, most noticeably, the increased complexity of these systems in comparison with their previous versions as well as the unreliability of the performance experience as perceived by the end user, which is influenced by socio-technical aspects such as technical knowledge, trust, system performance, availability and efficacy (Armbrust, Fox, & Griffith, 2009) (Gruschka & Jensen, 2010) (Grobauer, Walloschek, & Stocker, 2011)**.**

In order to be able to address these particular challenges, one possible solution could be to make better use of the ubiquitous industry standard performance logs. Performance logs are textual representations of different resource consumption and activities performed in the various operational cloud system components. Logs have been extensively deployed in the industry and used for both troubleshooting and punctual investigations of performance problems. In this research, logs are explored more extensively in order to address the need for precision, granularity and responsiveness within the decision time required for the current management/prediction challenges. The amount and granularity of the data harvested could potentially be massive. Each of the analyzed hosts or network components can generate as much as 800 KB of data per minute. This could quickly turn into a very large amount of data that is difficult to process and access using traditional SQL-based technologies. One of the possible alternatives for resolving this issue is employing Big Data technologies such as the Hadoop Distributed File System and Apache Spark in order to interactively collect the data from multiple sources and process the individual files simultaneously, which would prove difficult using classic relational database technology.

This research proposes a novel performance measurement model for cloud-based information systems as perceived by end users, with many practical applications in the domain of service level measurement and performance prediction. It identifies meaningful and actionable data center logs of low-level direct and derived measurements to model the end user performance perspective. The cloud computing measurement model and quality characteristics presented by Bautista's framework (Bautista, Abran, & April, 2012) are implemented and experimented. The model for the end user performance perspective for cloud computing systems using data center logs from Big Data technology expands Bautista's original work by proposing the utilization of a performance indicator and including end user response in order to forecast possible performance anomalies.

A large-scale experimentation is described where the measures are analyzed using a modern Big Data infrastructure in order to model the end user performance perspective as an expression of performance indicators based on the service level agreement for the cloud computing services studied.

The experimentation addresses the research question and offers a solution avenue for modelling the end user performance perspective of cloud computing based applications in future service level agreements so that a timely analysis of the data can be expected and a predictive algorithm developed to anticipate upcoming performance issues.

Keywords: performance, cloud computing, big data, spark

TABLE OF CONTENTS

LIST OF TABLES

Page

XVIII

LIST OF FIGURES

LIST OF ALGORITHMS

TABLE OF ABREVIATIONS

Abbreviation	Description
APIs	Application Programming Interface
BSC	Balanced Score Card
CC	Cloud Computing
CCA	Cloud Computing Application
CCS	Cloud Computing System
CIF	Common Industry Format
COTS	Commercial Off-The-Shelf
CSV	Comma Separated Value
HDFS	Hadoop Distributed File System
IaaS	Infrastructure as a Service
ICA	Independent Component Analysis
IT	Information Technology
ITSM	Information Technology Service Management system
KPI	Key Performance Indicators
LLDM	Low Level and Derived Measure
MTBF	Mean Time Between Failures
NAS	Network Area Storage
NIST	National Institute of Standards and Technology
MAC	Machine Address
PaaS	Platform as a Service
PCA	Principal Component Analysis
PDCA	Plan-Do-Check-Act
PMFCCA	Performance Measurement Framework for Cloud Computing Applications

RAM	Random Access Memory
RC	Root Cause
SaaS	Software as a Service
SLA	Service Level Agreement
S.M.A.R.T	Specific-Measurable-Achievable-Relevant-Time Phased.
SPQM-RM	Software Product Quality Measurement Reference Model
SQuaRE	Systems and software Quality Requirements and Evaluation

INTRODUCTION

Performance measurement of information systems, which is the ability to complete a given task measured against known standards of accuracy, completeness, cost, and speed, is a challenging research topic. Measuring the quality of information systems has been a concern for organizations, academia and software engineers since the early days of information technology. In the 1970's, Juran, a renowned quality expert, had already identified that measuring the quality of software, systems and Information Technology (IT) services is a challenging task (Juran & De Feo, 2010). This is, in part, caused by both the immaturity of software engineering as a science and that the industry as well as individual organizations are seldom able to keep up with rapidly evolving technologies (HP, 2013).

The measurement of a software used by an end user can be described from three main perspectives:

1) Internal quality perspective that measures how well built and maintainable is the application system.
2) External quality perspective which focuses on how well its underlying system infrastructure behaves to satisfy its end users.
3) Quality in use which is concerned with the end user perception when using the system to achieve daily tasks.

External measures try to reflect the actual utilization of the system by end users–one of the stakeholders of the software and the ones who use it to perform a task–to achieve their particular business goals (ISO/IEC, 2005). These perspectives and their interrelationships are documented in the ISO/IEC 25000 family of standards as described in Figure 0.1

There is a difference between the ISO software engineering standard definition of the quality perspectives of software performance measurement and the organizational, or business, perspective of software and IT performance. Software engineering performance, according to ISO, is related to the software construction, deployment and operational quality. Sustaining

high internal quality has the potential of offering greater end user (or external) quality, as long as it is well integrated with the operational environment. If this is achieved, it then has a better potential for achieving a high quality in use. It is also reported that if the end users are well trained and comfortable with using the software, the end user satisfaction (reflected in a high quality rating) will be high (Stavrinoudis, 2008). For high quality to be achieved, a number of factors must be controlled and measured to ensure success.

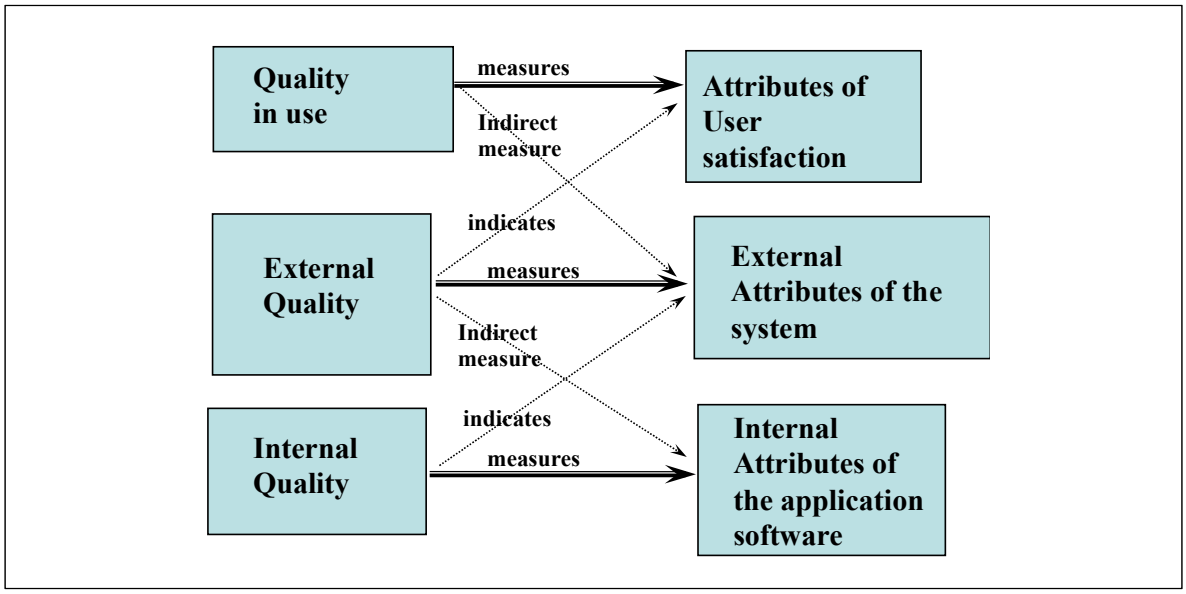

Figure 0.1 – Quality characteristics and attribute association (Bautista, Abran, & April, 2012)

Businesses, on the other hand, consider software to be a part of the service it renders to its customers; it is either useful or not to the organization in fulfilling its business goals (Bundschuh & Dekkers, 2008). This perspective of software system quality is focused mainly on the end results. This means that the utilization of software, and the resulting end user satisfaction, is the most important factor and is influenced by its availability and also by its performance. The usefulness is the ability of a software to solve organizational needs and is reported by Robert Glass as **"**the main criteria that the organizations use to state if a software is useful or not**."** (Glass, 1998)

As presented earlier, measuring end user perception of system performance has been a concern of software engineering researchers since the early 60's (Emery, 1964). Many experiments

about this topic have been designed, tested and validated (Buyya, Yeo, Venugopal, Brober, & Brandic, 2009) (Davis F. D., 1989) (Davis S. &., 2001) (Etezadi-Amoli & Farhoomand, 1996) (Fagan & Neill, 2004) (Law, Roto, Hassenzahl, Vermeeren, & Kort, 2009) (Mahmood, Burn, Gemoets, & Jacquez, 2010) (Tullis & Albert, 2010). Initially, these researchers used surveys with end users to understand the impacts of poor quality on their activities. It has been reported that using surveys in this context has important limitations, such as not being appropriate for following trends in real time, not providing a good source for cause and effect, having poor timing response, demonstrating low response rates and being vulnerable to responder bias (Couper, 2012). To minimize these issues and complement survey data, some form of automated, user-independent system performance measurement has been proposed over the years.

Literature reviewed on this topic describes how system performance measurement is conducted in many ways. One popular approach is to use data center logs as a source of information. This is popular because IT infrastructure (i.e. each component of the IT infrastructure) produces readily available operational data that are reported in the daily logs of its operational systems, applications, computers and telecommunications equipment (all of which we will call components in this thesis). These logs are often composed of binary files that include data from different components comprised in a system (the term "system" includes hardware and software in this thesis). Logs contain large quantities of data and are typically stored in a file or a database for further analysis when needed. Recently, many commercial, open source, and easily accessible log tools are available for collecting, analyzing and generating performance dashboards that present different measures of the IT infrastructure components used by an information system ("information system" is the application used by an end user in this thesis) (Microsoft, 2013) (Kopp, 2011) (Omniti Labs, 2014) (Agendaless Consulting and Contributors, 2017) (Tidelash Inc, 2017) (Massie, 2012) (Munin and colaborators, 2017) (The Cacti Group, 2017) (Nagios, 2013) (Zabbix, 2017) (Observium Limited , 2013) (Zenoss, 2013) (Forster F. , Collectd Open source project, 2017) (Weisberg, 2013). How these log measures are analyzed and interpreted and how these measurement results reflect the organizational goals, especially the end user’s perspective of system performance, are still to be resolved and

are part of the objectives of this research (St-Amour, 2011). One promising theoretical proposal to address this problem was submitted in a recent PhD dissertation by Luis Bautista entitled: "A Performance Measurement Model for Cloud Computing Applications". His theory and limited experimentation is described in greater detail in section 2.3, where we explore what has already been attempted by other researchers and how this research can contribute to help solve this problem.

As early as 1996 (Laguë & April, 1996), research showed that systems performance measurement using internal measures issued from data center logs tend only to measure the internal and very technical quality perspectives of an information system. This is why the end user performance perspective is often inferred, estimated, approximated and even sometimes guessed at based on experience and only sometimes using log data that may or may not directly affect the actual user's perceived performance according to the observer's perspective and experience (Huffman, 2017) (Friedl & Ubik, 2008) (Kufrin, 2005). As an example, data center analysts have observed that whenever a desktop's processor reaches 100% of utilization according to the performance logs, the end user experience, that is, what the end user perceives while using that specific information system at that specific moment, is degraded (Bundschuh & Dekkers, 2008). It has also been reported that a very high level of utilization of a particular component is not always a guarantee that it directly affects the performance experience of the end user at that time. This has also been reported in publications in the field of reliability engineering (Denney, 2005), where a stressed system with different levels of stress applied to each of its individual components (i.e. different components are placed under distinct stress levels, aiming for a balanced cost effectiveness of the software-hardware-communications arrangement) did not necessarily affect the performance of the system as perceived by its end users (Rapoza, 2015) (CA Technologies, 2014). For example, scenarios where the end user isn't interacting with the system but the performance is deemed as "bad" at that time represent a false positive, as the individual was not there to perceive it and consequently was not affected. Other scenarios where the quality is considered as degraded by the end user but was not properly measured by the internal and external measures have also been reported as false negatives (Mahmood, Burn, Gemoets, & Jacquez, 2010) (Tullis & Albert, 2010).

ISO 9141-10 defines end user experience as *"a person's perceptions and responses that result from the use or anticipated use of a product, system or service"* (ISO, 2009). This definition relates to human emotions, evidence that the end user experience is dynamic, context-dependent and subjective (Law, Roto, Hassenzahl, Vermeeren, & Kort, 2009). Information systems performance measurement, on the other hand, focuses on collecting quantic (i.e. quantitative, measurable and scalable) data to determine how the end users employ the system. This data can then be interpreted, compared with the organizations' benchmarks (Castor, 2006), the data center analyst's empirical proof and personal experience and with this system's typical daily operation, in order to give an opinion on the presence or absence of degraded performance *ex post facto*. Law, Roto et al. reported that the many log measures collected by any of the many available automated log data production tools always need to be interpreted by different stakeholders allowing for an interpretation in the organization context, that is, interpreting (the measure) and how it *feels* (the end user experience). This is a great source of debate and research both in academia and in today's organizations (ISO, 2009).

One characteristic of performance logs that should also be highlighted here is that there is little or no control over the quality of the design of the existing measures created by the individual software developers which are in turn used on this research. Software measurement theory insists that the design phase of a measurement process requires that both the direct and indirect measures collected should measure either the measurand, the perceptible portion of the software execution, or a model of its interaction with the real world. Additionally, the measurement process or model should aim to build a consensus on what will or will not be measured, describing the entity and the attribute, and documenting an adequate model that characterizes the attributes and their relationships. In this thesis, the quality of the resulting measurement model will be assessed by following the activities proposed in the recent Software Metrics and Software Metrology book published in 2010 by Dr. Alain Abran (Abran, 2010).

The recent, rapid and broad adoption of cloud computing technology (Weinman, 2009) by organizations presents many operational challenges in measuring system performance. Cloud

computing allows for the development of fragmented systems with multiple distinct components that rely on the performance of complex IT infrastructures that often include components that are dispersed geographically, are shared and distinct, often concurrently executing software (Mirzaei, 2008) (Mei, Liu, Pu, & Sivathanu, 2010). This rapidly emerging technology uses recently developed and emerging hardware and software technologies to deliver ubiquitous, resilient, scalable, billed-by-use, application agnostic systems (Prasad & Choi, 2010). Cloud computing technology is often categorized by three different service models:

1) Infrastructure as a Service (IaaS).
2) Platform as a Service (PaaS).
3) Software as a Service (SaaS).

To add to the complexity of this emerging technology, each of these service models can be hosted within an organization or supplied by third parties.

With this emerging technology, the challenges associated with the collection of data that is physically and logically displaced amongst different service providers and over different hardware is a major concern (Gilbert, 2011) (Trappler, 2011). For example, let's look at a very common infrastructure used by a typical internet-based information system:

1) Has a web page.
2) Runs on a distributed web server.
3) Is hosted on a clustered, multi-homed hardware.
4) Accesses a database that has local and remote content.

This is a very simple example. Organizational reality can get much more complicated. When using this information system on cloud computing technology, issues like the location of the data, the ownership of the servers, the accessibility of the logs, the security and privacy on the shared resources and the quality of the service provided are now pressing concerns for the organizations (Prasad & Choi, 2010) (Dillon, Wu, & Chang, 2010).

These new technologies pose interesting challenges. For example, cloud computing applications and their supporting infrastructures, when measured, generate large amounts of measures (Buyya, Yeo, Venugopal, Brober, & Brandic, 2009). When an end user reports degraded performance of the application software he currently uses on a cloud computing infrastructure, how can the data center analyst diagnose, and potentially prevent, such problems? What are the techniques and technologies that allow for a better understanding of performance monitoring and performance management using these new cloud computing technologies? (Jackson & Ramakrishnan, 2010)

Cloud computing performance measurement is an emerging research topic and is currently addressed by different authors. Some empirical approaches propose that automated software be used to simulate access to services, then measure response times (Suakanto, Supangkat, & Suhardi, 2012). Third-party performance evaluation services propose comparative tests amongst different providers (Pivotal Software, 2013) (Gartner, 2013) (Avran, 2010). When considering these proposals closely, very few details of how this is done are provided. Other approaches suggest the collection of internal measures of different service configurations over the same infrastructure (Meijer, 2012). Finally, Croll suggests that cloud performance should be approached from a business perspective first and the use of internal measures be considered afterwards (Croll, 2013).

Although there are numerous proposals, they all fall short of the goal of our main research question: how can the end user performance perspective of cloud computing based applications be modeled so that a timely analysis of the data can be enacted upon?

The literature review has helped summarize the state of the art in end user perspectives of systems performance. The review concludes that the end user perspective is rarely addressed, is not explained in detail when it is and is still not solved for cloud computing based applications. Beginning with a theoretical and unimplemented model proposed by Bautista, “A Performance Measurement Model for Cloud Computing Applications” (Bautista, Abran, & April, 2012), this research will design a novel model for the end user performance perspective

for cloud computing systems using data center logs from Big Data technology that not only expands Bautista's original theory by enhancing the original proposal, it will test the theory in a large scale private cloud case study, propose the use of a performance indicator and include end user feedback in order to validate and potentially forecast possible performance affecting anomalies.

Additionally, Bautista's research considered that the measures, once associated with a performance concept, would be used in adapted formulas to represent the referred concept. In this research, a particular combination of measures is only considered relevant at an individual point in time and proposes that the particular performance concept be represented as adequate or degraded depending on a combination of not only the associated characteristics, but of the whole application delivery chain. This representation is discussed in detail in section 4.6. The proposed measurement model will include both internal measures, directly collected from performance logs, as well as context-dependent, end user interactive satisfaction measures as suggested by previous researchers (Law, Roto, Hassenzahl, Vermeeren, & Kort, 2009) (Marshall, Mills, & Olsen, 2008) (Etezadi-Amoli & Farhoomand, 1996) (Baer, 2011). Experimentation in a large scale private cloud case study will explore if this proposed approach offers advancement for this problem compared with previous proposals and the state of the art.

As stated earlier, this research proposes that log data be collected during the experimental part of this research to validate the proposed measurement model. This will be performed on an actual private cloud computing information system and consequently will process very large amounts of data in real time. Log data will become increasingly larger as each experimentation iteration will increase the size of our experimental database. One possible solution for processing very large quantities of data, in real-time, is the utilization of recent and emerging Big Data technologies (Cohen, Dolan, Dunlap, & Hellerstein, 2009) (Trelles, Prins, Snir, & Jansen, 2011) such as the Hadoop Distributed File System and Apache Spark. These technologies can process data from multiple sources and individual log files simultaneously. This could prove to be difficult using a classic SQL-based technology (Reeve, 2012). Big Data cluster computing parallel programming approaches have recently been used successfully and

have shown to be useful for processing large performance logs (Rabl, 2012) (Dean & Ghemawat, 2008).

In summary, the objective of this research is to design a novel measurement model that includes a performance management framework that allows for the modelling of the performance, as perceived by the end user, of a cloud computing-based information system. This model shall employ, as much as it is possible, data center log measures currently in use in the industry for the convenience of their wide availability and the technical familiarity for data center analysts. Finally, this model should facilitate the future implementation of some form of performance management technique such as an SLA (Service Level Agreement) audit or continual improvement process. Finally, utilization of the end user feedback in the model will provide additional validation for the proposed anomaly detection model. A test for forecasting anomalies using the model will be attempted using a simple forecasting mechanism to explore if the present research can be further improved upon in the future.

In order to achieve this objective, a research methodology comprised of seven sub-steps is proposed and further explained in section 1.5:

1) Phase 1 – associating end user satisfaction with low level and derived measures (LLDM) extracted from performance logs.
2) Phase 2 – mapping LLDM measures into the Performance Measurement Framework.
3) Phase 3 – Validation of the quality measures using a validation method (Jacquet & Abran, 1998).
4) Phase 4 – Laboratory experiment for end user performance modeling.
5) Phase 5 – Design of an automated mechanism for end user performance modeling and proposition of a performance measurement model.
6) Phase 6 – Experimental validation of the proposed end user performance model.
7) Phase 7 – Discussions of the end user performance model's abilities and shortfalls.

CHAPTER 1

Research Introduction

1.1 Motivation

It has already been stated that managing IT infrastructure has been a challenge since the early days of the implementation of information systems in organizations (e.g. technology, data, and knowledge level of end users) (Laudon & Laudon, 2013). The accelerated adoption of recent technology, such as the emergence of new and highly mobile technologies, distributed knowledge, real time collaboration as well as growing competition have increased, or, more specifically, have constantly increased the complexity of information technology. In order to be able to compete, are companies leveraging their information systems in a way that enables end users to be as productive as possible? Are the investments required to keep these increasingly complex systems and infrastructures efficient really spent in a way that ensures a firm's competitiveness?

As we approach information systems as an ensemble of technology, information, knowledge and people, performance measurement becomes increasingly difficult to precisely define. When performance of an application system is measured, the goal is generally the reporting of a measure, usually mathematical or percentile, that explains how the system performs in the form of 0% - 100% of an N-dimension resource consumption: what does 0% resource consumption mean? What does 30% utilization mean? And what does 100% resource consumption mean? Who is concerned with these measurements being either high or low?

Goodhue and Thompson propose possible answers to these questions. A good management approach states that resources should be applied in such means that end users should be able to fulfill their task based on the "Fitness to Task Theory" (Goodhue & Thompson, 1995). This means consuming the least possible amount of resources with the help of techniques like the "Resource Allocation Matrix Theory" (Martensson, 2006). For example, measuring using an interval, such as from 0 – 100, would be just a quantitative way of measuring if an end user is

capable of completing a specific task using the available resources. On the other hand, Davis highlights that both resource availability and end user capability are directly dependent on the end user's motivation to actually fulfill tasks as described in the "End User Acceptance Theory" (Davis F. D., 1989). This theory is a derivation from two other research results: 1) the "Theory of Reasoned Action" (Fishbein & Ajzen, 1975) which is a widely used model from social psychology that describes performance, for a particular action, as a result of a person's intention, attitude and subjective norms towards that action; and 2) the "Technology Acceptance Model" (Davis F. D., 1989), which describes that the resulting use of a system is a result of the end users intention to use it, weighted by the attitude of perceived usefulness and ease of use, as well as other external variables.

Fundamental to the "End User Acceptance Theory" is the author's use of a seven question "Likert Scale" for measuring time effectiveness, ease of use, improved quality, exclusivity, accessibility, dependability and refutability of the end user's use of a particular information system. This was presented to the end users of a particular information system in scenarios of both brief-exposure (e.g. a one-hour hands-on controlled experiment) as well as direct interviews at the end of the school semester. The outcomes that concern this particular research is that perceived usefulness, which is what the information system does to help the end user to achieve his goal, and perceived ease of use, which is how effortless it is to perform the said actions, were the key factors found to impact the end user's acceptance and thus the perceived quality of an information system. The information system analyzed by this case study was a simple text processing program. In this case, it is easy to understand the task that has to be performed and the increase in performance in comparison to the alternatives: either handwriting, using an old-fashioned typewriter or even the use of a different text processing software. The same parallel is valid for the concept of ease of use; features like auto correction, automatic saving and the ability to work with revisions can be compared to alternatives that would give the end user the *impression* of what is easier to use. It is reported that this early research approach is still a fundamental concept employed by many recent software engineering research approaches to the study of end user acceptance of different information systems (Hambling & van Goethem, 2013). The emergence of cloud computing technologies

adds complexity to this measurement approach. Figures 1.1 and 1.2 demonstrate the difference between an older IT client-server architecture versus a modern cloud computing architecture used by an information system.

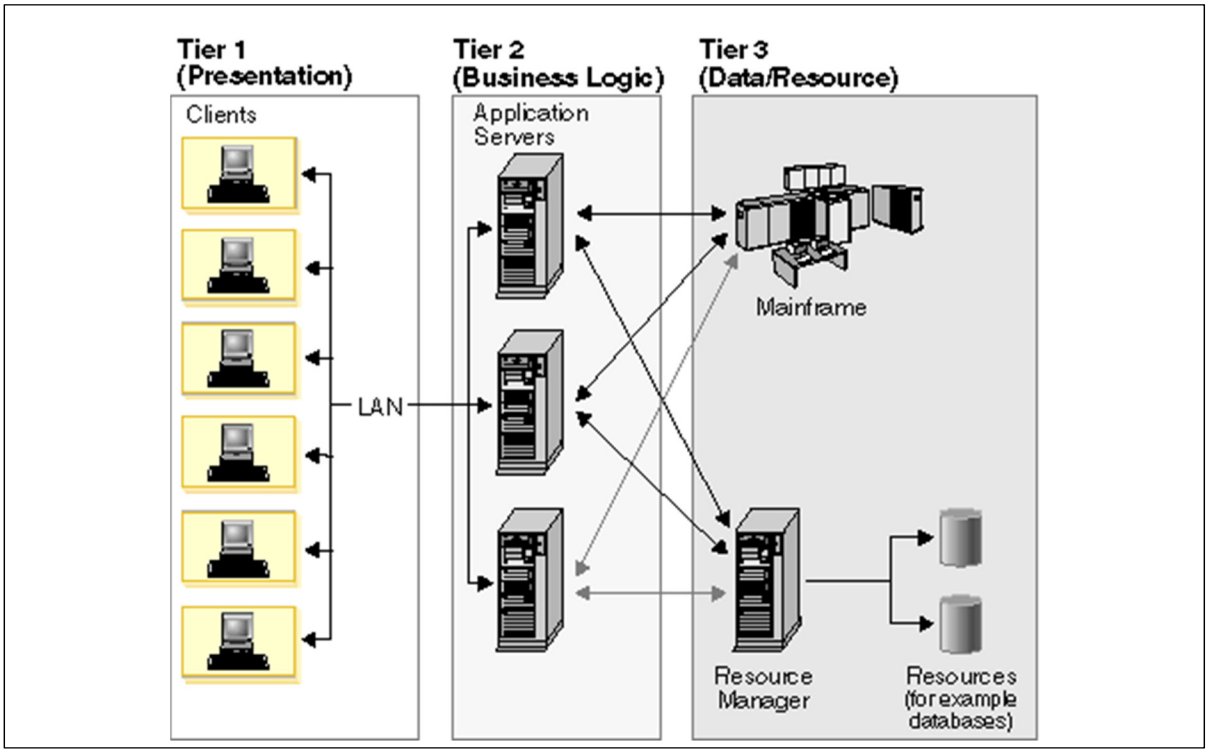

Figure 1.1 – Common three-tiered client-server architecture (IBM, 2013)

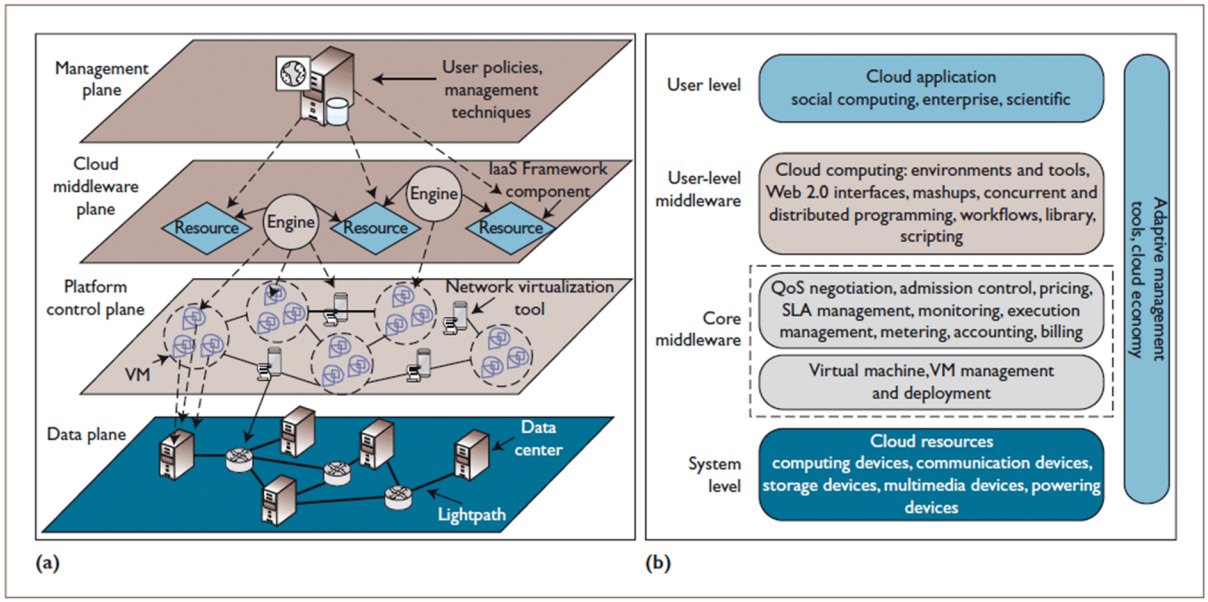

Figure 1.2 - Cloud computing architecture (Martensson, 2006)

In a cloud computing environment, after an end user is trained on a specific information system and engages in its daily execution, what components between his end user interface and the data repositories and processors should be included in order to actually measure (or try to measure as accurately as possible) the end user's perceived performance when using the information system? One of the possible approaches to measuring performance from the end user perspective is the industry default approach of data center log analysis. This practice is already used for numerous applications as both a troubleshooting and a monitoring technique. To cite a few, data center logs are present in different operational systems, types of hardware and applications (also called components in this thesis), and the resulting operating information is created at different granularity levels (Agendaless Consulting and Contributors, 2017) (Kopp, 2011) (Bundschuh & Dekkers, 2008) (The Cacti Group, 2017) (Friedl & Ubik, 2008). Employing log files for modeling the end user perceived performance of the information systems in use could be an approach that reveals itself to be both simpler and more easily automated than performing the end user acceptance theory and interviews of all the end users of a cloud computing application as proposed by the fundamental theory presented in the previous section.

Performance measurement frameworks for cloud computing applications (CCA) are still in the early stages of research (Bautista, Abran, & April, 2012). Adoption of cloud computing technology by the industry is also in its early stages (Phaphoom, Wang, & Abrahamsson, 2012) (US General Service Administration, 2010). The study of cloud computing performance management has the potential for innovative research, particularly in conjunction with the utilization of recent very large volume data processing technologies such as Big Data (Lin & Dyer, 2010).

With this understanding, we now have a possible solution for addressing this research question by designing a performance measurement model and experimenting with it in a real cloud computing world setting, where a variety of complex and interconnected individual IT infrastructure components can be measured using emerging Big Data technologies. The proposed measurement model would include measures from the information system and its IT

infrastructure components to provide an end user perspective of the performance of the information system. Once these measurements are collected and related to each other, there is an opportunity to model the perceived performance from the end user's perspective over time and maybe even predict it. To achieve this goal, many research activities will have to take place. Here is an overview of the proposed research method steps considered in regards to this question. Beginning with Bautista's theoretical and unimplemented model (Bautista, Abran, & April, 2012), we identify specific performance measures currently available from the data center logs of an actual private cloud computing application, expanding the initial model that only used a partial set of controlled measures with the addition of end-to-end measures that represent the complete cloud computing application delivery chain, including the end user performance perspective.

It is important to note that performance log measures may not be sufficient to completely model the end user perceived performance. The following scenario can explain this: it is possible that a component, at in any given moment, is performing an action while unattended. If this action consumes many resources, modeling the performance only by monitoring the logs would create a false-positive. In a similar way, if a problem occurs with a component for which the measure wasn't automatically identified as important but affects the end user, the model wouldn't flag that particular situation as a problem, resulting in a false-negative. Many authors reported the importance of end user feedback as an additional validation for understanding the actual perceived performance of an information system at a given moment.

A possible solution to this problem is correlating candidate measures that are extracted from the performance logs with end user performance degradation reports for the different components in order to identify potential systemic issues that degrade the performance across multiple components. This can be complemented by an anomaly detection process, performed on the analyzed logs, in order to identify if a particular point in time has potential performance degradation. The result of the anomaly detection along with the end user's feedback could then be used to compose a degradation scenario. To some extent, the possibility of the occurrence

of degradation events could potentially be forecast. This solution scenario will be further discussed in section 2.3.3.

1.2 Problem definition

Measuring the performance of an information system from the end user perspective is a complex task. First, internal software performance concepts, measured through a number of internal measures, must be correctly defined, designed, then validated to ensure the measurement correctly produces what it is supposed to measure. Secondly, these internal measures must be transformed or translated into information (e.g. external measures) and applied/communicated to yield results within the decision time specified or required by the organization. Finally, these measurement results need to be exploited/interpreted by some form of intelligent mechanism that may either be machine or human in order to infer significance to the measurement and potentially take preventive actions.

These quantitative measures, when collected with a high level of granularity (for example, one value per second per measure) will quickly accumulate in a large data repository. Initial experimental estimates indicate that each host, network device and server can generate 800 KB of data per minute. For this organizational case-study network, this could reach 1.2 GB and ~800000 columns per minute. By comparison, this is more than 80 times bigger than the highest recommended configuration for the most recent SQL databases (US General Service Administration, 2010). Considering such a challenging experimental scenario, the problem definition of this research can be summarized as: modeling end user experience on cloud computing environment with the proposition of a performance measurement model, using data currently available from data center logs and gathering end user feedback as needed and if possible, and, because of the amount of data, employing emerging Big Data technology such as Spark, for its capture and experimentation. If the use of the data center logs is insufficient, additional feedback mechanisms will be proposed.

1.3 Research question

Given the opportunities for discovery in the field of software performance measurement from an end user perspective using cloud computing technology, this research focuses on the proposition of a performance measurement model considering two main objectives: 1) Is it possible to measure and analyze the performance of an information system operating on the cloud, from an end user perspective, using only data center log data?; 2) What are the useful internal measures among all of the available measures that would reflect the application software performance as perceived by its end users?

The general research question can be formulated as: how can end user perceived performance of an information system be measured in a cloud computing environment? This question is then segmented in the following four specific research questions:

1) What defines a cloud computing environment?
2) What influences the end user performance perspective measurement in a cloud computing environment?
3) Are performance logs sufficient for modeling the end user performance perspective? If not, which other sources are required?
4) Can the theoretical proposal of the performance measurement framework for CCA (Bautista, Abran, & April, 2012) be used for the creation of a performance model using data center logs that represents the end user performance perspective of an application using cloud computing technology in a timely fashion?

1.4 Methodology

In order to answer the research questions outlined in the previous section, the software engineering research methodology proposed by Victor Basili (Basili, Selby, & Hutchens, 1986) is used to plan this research and is described using four main research activity phases: 1) definition of the research, 2) planning, 3) development of theory and experimentation and 4) interpretation of the results which are presented in sections 1.4.1 to 1.4.4.

1.4.1 Definition of the research

This first research phase, presented in Table 1.1, clarifies the research motivation, objective, goal and end users.

Table 1.1 - Research Definition

Motivation	Objective	Goal	Users
The design of a performance measurement model that reflects the end user experience for an information system operating on a cloud computing environment.	. Define/clarify the notion of end user performance perspective; . Define/Clarify the cloud computing technology; . Identify the data center log direct measures that best reflect the end user perspective of an application operating on a cloud; . Design a measurement model and its toolset to support the infrastructure specialist in proactively managing the cloud infrastructure to identify the performance issues from the end user's perspective.	Design a performance measurement model and its prototype that is capable of representing the end user experience of an application operating on a cloud by mainly using data center measures currently available in commercial and open source tools.	Students, researchers, IT professionals and managers.

This next phase presents the specific planning of research activities that have to be achieved in order to meet the objective.

1.4.2 Planning

The planning phase contains the description of deliverables which address each of the four research questions. This research begins with the required literature reviews (see Table 1.3).

Table 1.2 - Research Planning

Planning Steps	Inputs	Outputs
State of the art of the concept of end user quality/performance perception when using an information system	Literature review: . Software Engineering performance; . End user expectation and perception of information system performance; . End user performance perception, and other psychosocial entities that affect end user performance perception.	-Literature review of the state of the art of containing IS performance standards, models, techniques and methods; -State of the art of the end user performance perspective for cloud computing- based systems.
State of the art of cloud computing and Big Data technology for data center log processing	Literature review: . Cloud computing technology, components, types and utilization; . Existing data center log data analysis; . Apache Spark project documentation; . REAP project data.	- Literature review of existing data center log uses and techniques for its analysis, open source Big Data technology and corroboration of the Cloud computing syllabus by matching of components with the experiment's infrastructure; -First publication: proposal on how to measure performance as perceived by the end user that uses cloud applications.

1.4.3 Development of theory and experimentation

The development phase of this research presents activities that support new knowledge and theories. It also describes the definition and preparation required for the experimentations and validations as well as the key research activities that attempt to answer the main research question. In order to address this, we segment the task into the following research sub-steps:

1) Association of end user performance with LLDM measures.
2) Mapping LLDM into the Performance Measurement Framework.
3) Validation of the quality measures using a validation method (Jacquet & Abran, 1997).
4) Laboratory experiment for end user performance modeling.
5) Expanded laboratory experimentation.
6) Design of an automated mechanism for end user performance modeling and proposition of a performance measurement model.
7) Validation of the automated model.
8) Proposition of the model.

1.4.3.1 Association of end user performance with low level and derived measures

Measuring end user perceived performance and satisfaction with the use of an information system has already been presented by several researchers (Baer, 2011) (Buyya, Yeo, Venugopal, Brober, & Brandic, 2009) (Davis F. D., 1989) (Davis & Wiedenbeck, 2001) (Etezadi-Amoli & Farhoomand, 1996) (Fagan & Neill, 2004) (Law, Roto, Hassenzahl, Vermeeren, & Kort, 2009) (Mahmood, Burn, Gemoets, & Jacquez, 2010) (Marshall, Mills, & Olsen, 2008) (Tullis & Albert, 2010). In these publications, end user performance and end user satisfaction were identified as intrinsically interdependent, meaning that whenever end users where satisfied with information systems these proved to be well performing, and vice versa. These research results were mostly based on conducting surveys and interviews with the end users in order to identify factors, determine performance and evaluate information system

quality. One of the challenges of this research is mapping measures to end user performance characteristics. Assuming that a way for the end user to communicate the dissatisfaction with a system is to present a complaint, a survey could be performed on these complaints, and, in this survey, identify the events where the end user was not satisfied with the system's performance. The performance logs of these events could be investigated to look for evidence of which measures were in a degraded state at the time reported for each of the events. This could lead to a non-exhaustive list of measures and states reported for moments of end user dissatisfaction.

1.4.3.2 Mapping low level and derived measures into the Performance Measurement Framework

Measuring the performance of cloud computing-based applications using ISO quality characteristics is a complex activity for various reasons. Among them is the complexity of the typical cloud computing infrastructure on which an application operates. Beginning with the quality concepts proposed in the ISO 25010 standard (maturity, fault tolerance, availability, recoverability, time behavior, resource utilization and capacity) this research maps the collected measures into the performance concepts by associating the influence of each particular measure in regards to the concepts. This is fundamentally different from Bautista's proposition where the measures are manually associated to the performance concepts and the formulae are built depending on the context selected. In the present research, the combination for particular measures is only relevant for that particular moment in time and, for another observation, different measures can fulfill the same concept. This is explained in detail in section 4.

1.4.3.3 Validation of the quality measures using a validation method

Jacquet and Abran (Jacquet & Abran, 1998) propose a validation framework for software quality measures which address three main validation issues: 1) the validation of the design of the measurement method; 2) the application of the measurement method; and 3) the predictive

system. This measurement validation framework is based on a measurement model which is detailed in Figure 2.5 and presented later in this thesis. For this research, we use the results of sub-steps 1.4.3.1 and 1.4.3.2 using this model and conduct 3 experiments: 1) the validation of the representation theorems; 2) the application of different numerical values to these rules in order to simulate the response of the theorem; and 3) the proposition of a quality model.

1.4.3.4 Laboratory experiment for end user performance modeling

This sub-step will consider the measures collected during sub-step 1.4.3.1 and the validated mapping to the measurement framework from sub-steps 1.4.3.3 and 1.4.3.4 to manually create an end user performance model for the experimental case study. The objective is to gather information for the creation of an automated solution that would be able to respond to the information needs of the decision makers in a timely manner. This experiment will also attempt to represent the end user performance perspective in a graphical manner, facilitating the interpretation of results. In this experiment, we will also determine if the log data is sufficient for modeling end user performance perspectives.

1.4.3.5 Expanded laboratory experimentation

Leveraging the outcomes of sub-step 1.4.3.4, this next step will expand the initial population to a larger infrastructure of servers and desktops, aiming to target approximately 500 servers and 30000 end users in North America. The objective of this is to verify the reproducibility and expandability of the earlier findings. If the log data has been found to be insufficient in the previous sub-step, a feedback mechanism will be proposed during this sub-step in order to gather further information about the user's perspective under different information system performance scenarios, such as where there is evidence of degradation, evidence of good performance, lack of end user complaints or increased end user complaints.

1.4.3.6 Design of an automated mechanism for end user performance modeling and proposition of a performance measurement model

With the utilization of emerging Big Data technology, it may be possible to design an experiment that will apply the measurement rules and allow for the analytic functions to model the performance as perceived by the end user in a case study.

1.4.3.7 Validation of the automated model

This sub-step is a repetition of sub-step 1.4.3.3 and aims at validating the automated model using the same process.

1.4.3.8 Proposition of the end user performance model for cloud computing applications

This is the final sub-step of this research that will propose a model for end user perceived performance of the information system operating on a private cloud. This model will be based on the results from the previous sub-step and might include, if necessary, a self-reporting mechanism where the end users can point to a degraded performance. Additionally, this proposed model will include a prototype using a Big Data processing cluster-based on Spark in order to test machine learning algorithms capable of predicting end user behavior given the analysis of the performance time series.

1.4.4 Interpretation of the results

This section contains the planned activities for properly understanding the methods, use cases, scenarios and results that will be obtained during the experimentation of the proposed model, as well as for providing grounds for conducting future research.

Table 1.3 - Interpretation of the results

Interpretation Context	**Extrapolation of results**	**Future research**
Experimentation: Application clusters are assigned according to the specific use cases tested, for example "all Outlook 2010 end users"; Discussion on the validity of the measures identified for the experimentation; Discussion on the resulting performance model utility.	. Five different case studies for 2 population levels (500 servers, 30000 end users) and 4 different physical location arrangements (North America, Asia, Europe, Global-whole world combined) . Different sets of measurement variables . Discovery of related applications in shared workspaces . Machine learning approaches for dynamic work distribution based on end user performance measurement fluctuations	. Further investigations using machine learning to prevent degradation and resource misallocation . Further investigation to locate clusters of related applications (applications that consume different sets of resources, thus optimizing resource utilization) . Is it possible to locate clusters of related end users? . Can machine learning dynamically assign workloads according to related profiles?

1.5 Chapter conclusion

In this chapter, the research steps have been defined and presented along with their motivation, objective and specific questions. The methodological approach has been presented using Basili's software engineering research experiment framework in order to present an overview of the research steps. Research sub-steps, with particular deliverables, have also been presented. In chapter 2, the literature review which covers the topics of performance management and cloud computing will be presented, setting the stage for the clarification of the research problematic that will be presented in chapter 3.

CHAPTER 2

Literature review

This section presents a synthesis of elements concerning performance management from a software engineering and business perspective as well as the literature review of the topic of cloud computing, its architecture, advantages and disadvantages. Software quality models have long been discussed (Mccall, Richards, & Walters, 1977) (Dromey, 1995) (Grady, 1992) (Jacobson, Booch, & Rumbaugh, 1999) (ISO/IEC, 2003) (ISO/IEC, 2005) with researchers and practitioners gravitating towards internal and external performance characteristics that should be satisfied in order to obtain a software product that displays high quality. On the other hand, the business perspective often relies on the concepts of key performance indicators (Kaplan & Norton, 1992) and service level agreements (ISACA, 2012), focusing on efficiency and end user satisfaction. These two perspectives overlap and are complementary, both required for the creation of a broad model for performance measurement that is able to measure end user performance of CCA. Figure 2.1 demonstrates a generic ISO/IEC 25000 measure paired with an equivalent strategic map that contains a KPI. Finally, the cloud computing topic is presented with the review of the relevant literature.

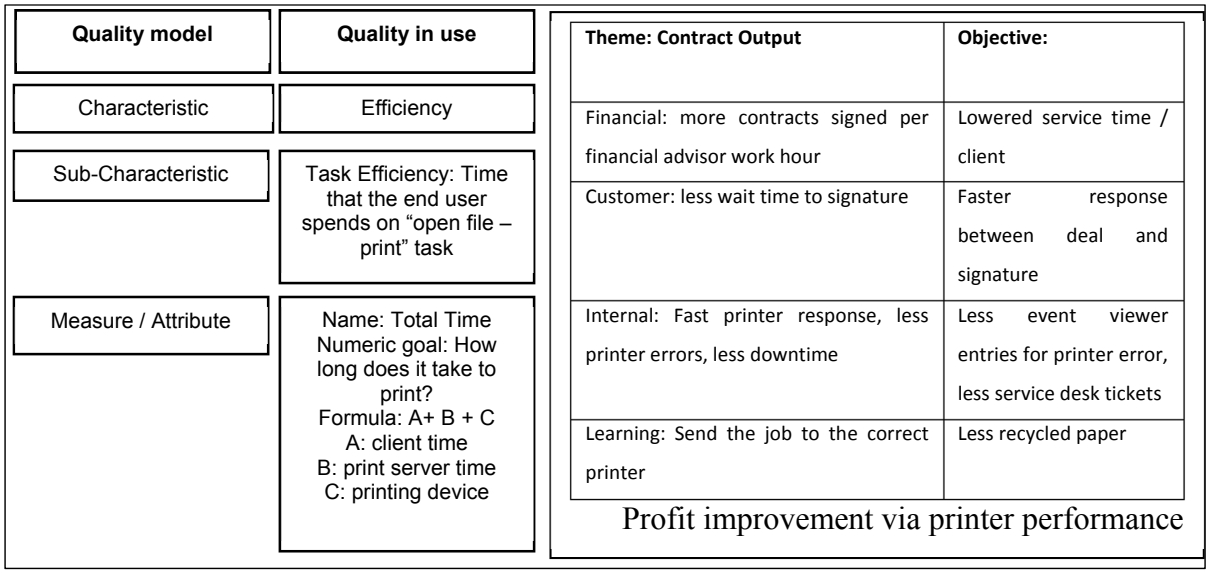

Figure 2.1 - ISO/IEC 25000 compliant measure versus BSC & KPI compliant measure

2.1 Performance management

The software engineering perspective of performance measurement is presented in section 2.1.1. It summarizes a review of the most recent ISO reference models. This review is based on the international standards as well as different issues and limitations published concerning their applications. Then, the business perspective of the end user perception of performance measurement frameworks, when using an information system, is described in section 2.1.2. This topic has been popular since the 80's and its evolution, current trends and performance measurement tools are presented. Methodologies, research conducted and their results are discussed in order to uncover potential research and applicability of the techniques in lieu of the proposed cloud computing-based research. Finally, the limitations and difficulties of using these proposals are discussed.

2.1.1 Performance Measurement – Software Engineering Perspective

This section presents the ISO 25000 family of standards, the ISO 15939 standard, the subject of metrics validation and the difficulties of applying such standards in organizations. The objective is the documentation of the completeness of the contemporary ISO 25000 standard as the confluence of previous standards, the coverage of the ISO 15939 measurement process and the caveats that involve the selection, election and evaluation of the metrics. Finally, an evaluation of the performance measurement process is executed to demonstrate the typical efforts and challenges involved in applying such standards in an organization.

What is quality for a software product? Many authors define and debate quality: (Shewhart, 2015), (Deming, 2000), (Feigenbaum, 1991), (Juran & De Feo, 2010) and others have contributed to the creation of a broad definition, reflected in ISO/IEC 9001, where quality is the characteristic that a product or a service has that defines it as satisfactory to its intentions. Measuring quality then requires validated and widely accepted measurement models like ISO/IEC 9126 (ISO/IEC, 2003) and its superseding ISO/IEC 25000 series (ISO/IEC, 2005) of standards named SQuaRE. Systems and Software Engineering – Systems and software Quality

Requirements and Evaluation (SQuaRE) aims to harmonize many other standards of software quality such as ISO/IEC 9126, 14598 and 15939, complementing and addressing the gaps between them.

SQuaRE has many groups of documents for different audiences. They are: Quality Management (ISO/IEC 2500n), Quality Model (ISO/IEC 2501n), Quality Measurement (ISO/IEC 2502n), Quality Requirements (ISO/IEC 2503n), Quality Evaluation (ISO/IEC 2504n) and the Extensions (ISO/IEC 25050 - 25099). The 5 groupings and their 14 documents are listed in the next section (section 2.1.1.1).

2.1.1.1 ISO 25000 (SQuaRE) Grouping and Documents.

This section briefly describes the 5 groupings and 14 documents that compose the SQuaRE international standard on software quality. Figure 2.2 demonstrates the groups and documents.

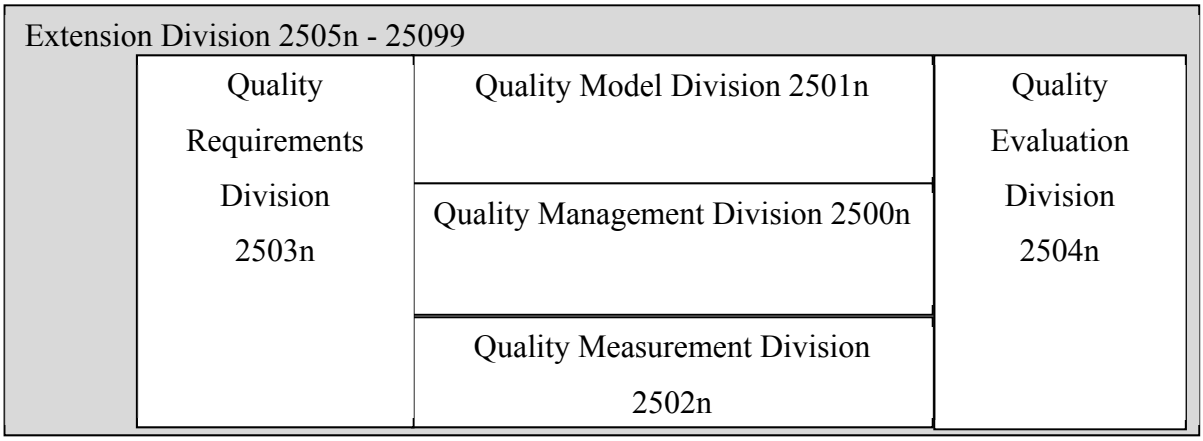

Figure 2.2 - ISO/IEC 25000 - Groups of documents, adapted from (ISO/IEC, 2005)

- **ISO/IEC 2500n – Quality Management.** International Standards for common models, terms and definitions that are referred to by the other documents of the SQuaRE series. It contains only two documents: 1) 25000 Guide to SQuaRE– pertaining to the architecture, terminology overview, parts and references; and 2) 25001 Planning and Management–with the requirements and guidance for supporting the specification and evaluation of software and system products.

- **ISO/IEC 2501n – Quality Model.** Quality models for systems and software products, quality in use and data, including practical guidance for its utilization. It contains only two documents: 1) 25010 Quality model–characteristics and sub-characteristics for product quality and quality in use, derived from ISO/IEC 9126-1 and 14598-1; and 2) 25012 Data Quality model–definitions of general data quality models within computer systems, for data quality requirements, measures, planning and quality evaluations.
- **ISO/IEC 2502n – Quality Measurement.** Reference model, mathematical definitions and practical guidance for quality measurement. The five documents contained in this division are: 1) 25020 Measurement reference model and guide–introductory explanation and reference model for the application of performance measurement from the International Standards; 2) 25021 Quality measure elements–recommended base and derived measures to be used during the system or software development life cycle; 3) 25022 Measurement of quality in use–a set of measures for quality in use; 4) 25023 Measurement of system and software product quality–quantitative measures for system and software products according to the characteristics defined in ISO/IEC 25010; and 5) Measurement of data quality–quantitative measures for utilization with ISO/IEC 25012.
- **ISO/IEC 2503n – Quality Requirements**. Specification of quality requirements to be used in the elicitation for product requirements and inputs for evaluations. It contains only one document: 25030 Quality requirements: guidance–requirements and recommendations for quality requirements based on ISO/IEC 9126-(1-4), 14598-(1, 3-5).
- **ISO/IEC 2504n – Quality Evaluation.** Requirements, guidelines and recommendations for product evaluation. It contains four documents: 1) 25050 Evaluation reference model and guide–requirements and process description for evaluating system or software products; 2) 25041 Evaluation guide for developers, acquirers and independent evaluators–specific recommendations for these 3 types of actors; 3) Evaluation modules–structure and contents for documentation of

evaluation modules; and 4) Evaluation modules for recoverability–external measures for systems and software resiliency and automated recovery.

- **ISO/IEC 25050 to 25099 – SQuaRE extensions.** International Standards and/or technical reports addressing specific application domains or complementary to one or more SQuaRE standards. There are seven document in this series: 1) 25051 Requirements for quality of commercial off-the-shelf (COTS) software products and instructions for testing–quality, documentation, test requirements, conformity and evaluation of COTS software according to the ISO/IEC 12119; 2) 25060 Common Industry Format (CIF) for usability test reports–general framework for usability-related information, potential standards for specification and evaluation of the usability of interactive systems; 3) 25062 Common Industry Format (CIF) for usability test reports–format for reporting measures from usability tests according to ISO 9241-11; 4) 25064 Common Industry Format (CIF) for usability–user needs report CIF for reporting end user needs with specifications for the contents and sample format of end user needs reports; 5) 25063 Common Industry Format (CIF) for usability–context of use description, high level and detailed description format for existing or future systems; 6) 25065 Common Industry Format (CIF) for usability–user requirements specification, CIF for end user requirements specifications with relationship between the specified requirements; and 7) 25066 Common Industry Format (CIF) for usability–evaluation report specifications of the contents of evaluation reports.

2.1.1.2 ISO/IEC 25010 – Quality in use and Product Quality Models

The quality in use of a system is the result of the internal quality of the software, the hardware and its operation environments, as well as the interactions between the end users and the system. It is influenced by the end users, the tasks and the social environment that is created by leveraging the use of the system. The five characteristics that compose the ISO software product quality model are: effectiveness, efficiency, satisfaction, freedom from risk and

context coverage. Figure 2.3 demonstrates some of the characteristics and sub-characteristics in a graphical manner for clearer understanding of the internal and external quality model.

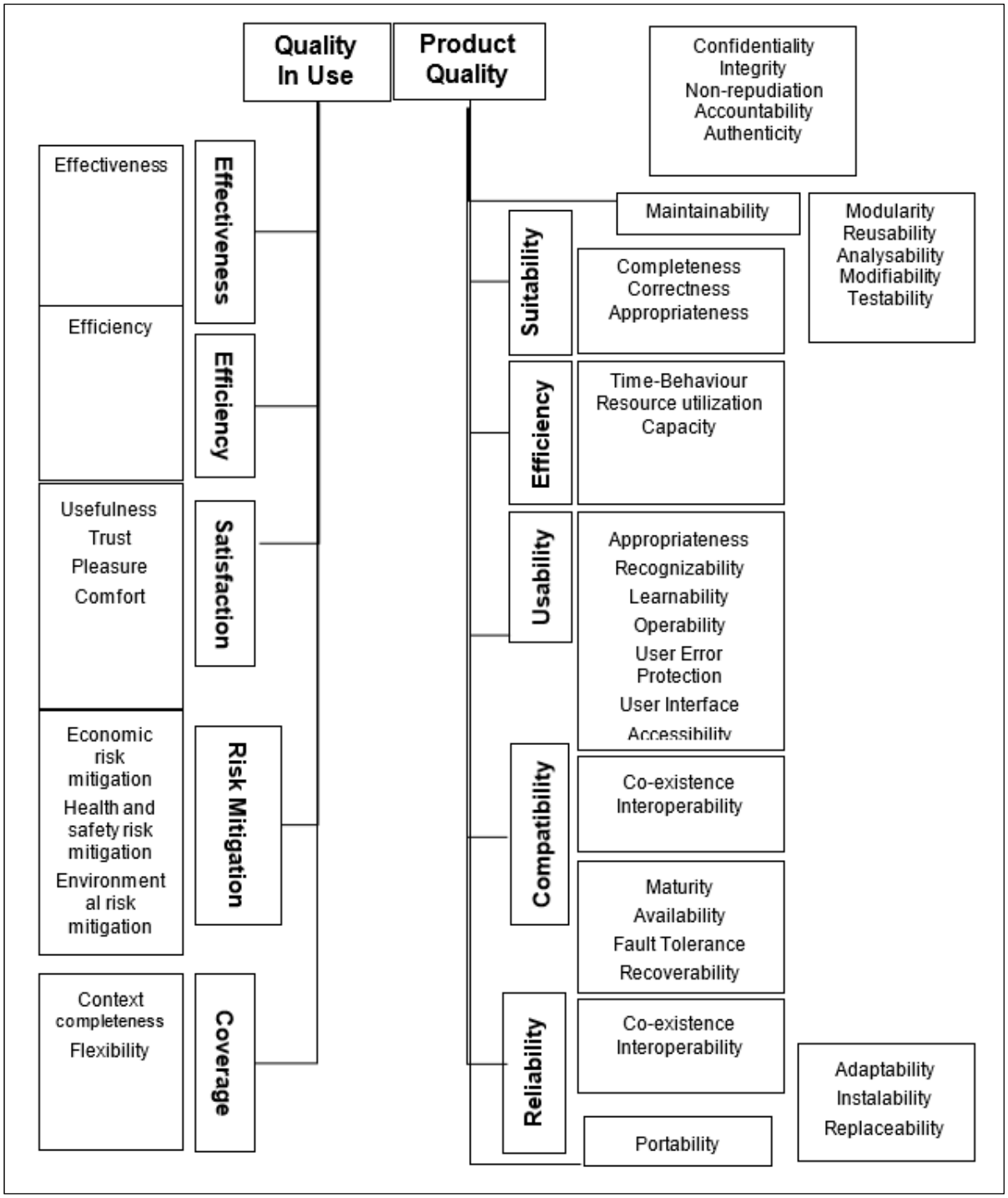

Figure 2.3 - Quality in Use and Product Quality models (ISO/IEC, 2005)

Sub-characteristics are derived from these broader categories. Usefulness, trust, pleasure and comfort are sub-characteristics of *satisfaction*. Economic, health and safety, and environmental risks are sub-characteristics of *freedom from risk*. Flexibility and context completeness are sub-characteristics of *context coverage.*

The product quality model focuses on the intrinsic qualities of the software products, the computer system and the sub-characteristics that integrate the system. The quality in use model focuses on the interaction between the end user and the system and how this interaction affects the outcomes and operation of a system, whereas the product quality model focuses on the software and system components and their interaction that influence the results achieved by the system. One such measure is described in Figure 2.4.

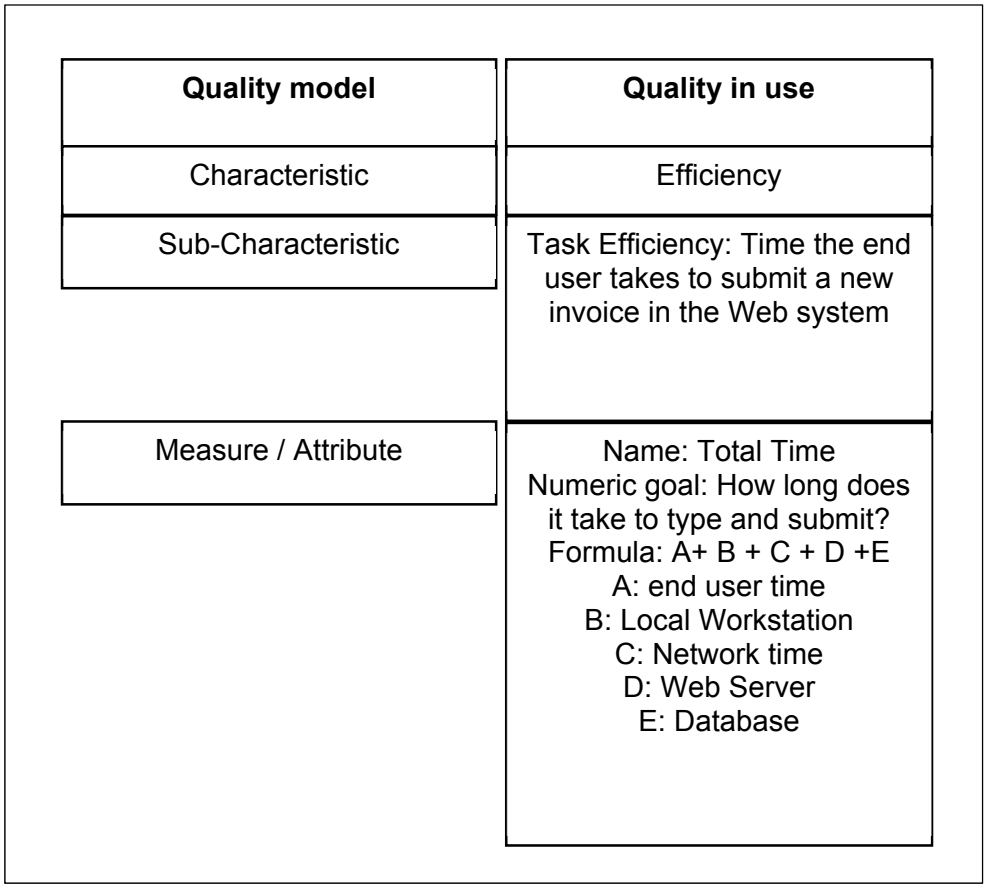

Figure 2.4 - Quality in use: New Invoice Submission efficiency measure

With this approach to quality, it is possible to imagine a model of an information system: the system composed of directly related hardware and software, as well as unrelated software (applications installed on the same machine that are not part of the information system, for example) and unrelated hardware (other machines that use the same network as the target system). The actual information system, composed of machines, information and people, encompasses both the product quality target as well as the scope of utilization, requirements and evaluation by the end users, with the stakeholders directly influencing the perception of a system's quality.

The quality in use model proposed by ISO/IEC 25000 is defined by five characteristics: effectiveness, efficiency, satisfaction, freedom from risk and context coverage. The product quality model is characterized by eight properties: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. These properties are extensively described in the ISO/IEC 25010 document.

In this research, the focus will be placed on efficiency, usability and maintainability (particularly time behavior), task efficiency, resource consumption, end user time and error occurrence. This refines the focus and the objectives of the research.

Different stakeholders have different perspectives of the perceived quality. The stakeholders can be characterized as primary (direct interaction with the system in order to achieve primary goals), secondary (content providers, managers, maintainers and installers) and indirect (output consumers). It is important to differentiate the stakeholders' approach to determining the scope of a quality system because the intrinsic differences between perspectives, knowledge and expectation will define different measures for each one of the characteristics and sub-characteristics. At any given moment, although one user, like a data center support technician, might be satisfied with a server performance, it is not guaranteed that an end user using an application on that same server, will be just as satisfied at that same moment. Table 2.1 describes this relation in regards to the measures and the expected outcomes.

Stakeholder satisfaction can be greatly influenced by external elements such as a user's predisposition towards technology, learning, stress levels, comfort, environment in use, and cooptation levels towards the system's goals. Quality measures, in this case, might be influenced by the effect of external elements on its stakeholders. This "noise level" should be explored at the point of the measurement result step in order to decide if its presence could alter the performance measurement process (Marshall, Mills, & Olsen, 2008).

Table 2.1 - Different stakeholder perspectives for the quality of "Time Effectiveness"

Stakeholder	**Measure**	**Measurand**	**Expected outcome**
Primary user: Direct interaction with the system	Effectiveness: time to complete and submit form	Browser's response time "document done"	Typing, clicking "submit" and receiving confirmation should be completed without errors and delays.
Secondary User: Content Provider or application owner	Effectiveness: time for processing form	Processor time and utilization, process stack	User will provide proper data that will be processed according to previous benchmarks, no extraneous influences on the system.
Secondary User: Maintainer or support technician	Effectiveness: time for processing form	Actively collected logs	There will be no internal errors, crashes, end user errors or exceptions that cause the whole system to be unstable.
Indirect user: Manager	Effectiveness: form processing effectiveness	Number of processed forms versus work hours and infrastructure investment	The number of processed forms must increase whereas work hours and investment in infrastructures lower per processed form.

The quality in use and product quality models are described by interactive quality characteristics. These characteristics can be represented from different stakeholder perspectives. The stakeholder bias and predisposition towards a system can influence the fulfillment of the system's primary goals, thus affecting the quality measure. Quality in use and product quality are both the ability of a system to satisfy the stakeholder's needs as well as the result of the interaction of the aforementioned stakeholders with the system. An end user that is personally unsatisfied with the organization as a whole will often present poor

satisfaction with any aspects of the organization, including its systems (Baer, 2011) (Buyya, Yeo, Venugopal, Brober, & Brandic, 2009) (Davis & Wiedenbeck, 2001) (Etezadi-Amoli & Farhoomand, 1996) (Law, Roto, Hassenzahl, Vermeeren, & Kort, 2009).

2.1.1.3 ISO/IEC 15939:2007 Systems and software engineering – Measurement process.

Measuring is an important part of the quality process. It is the measurement process that determines the objectives and where progress towards the fulfillment of the set requirements may be assessed. It is also with the help of measures that it is possible to observe changes such as “improvement” and “deterioration” of the status of quality measures.

The objective of a measurement process is to collect, analyze and report data for decision making as recommended by the international standards. A successful measurement process should observe the following stages: organizational commitment towards measuring; identification of information needs; identification or development of measure sets; identification of measuring activities; planning for measurement; data collection, storage and analysis; utilization of the information for better decisions and communication; evaluation of the measurement process and communication of the improvements on the measurement process to the process owner. The core activities of the measurement process, as recommended by ISO 15939, are planning and performing the measurement process itself. The other activities establish, sustain measurement commitment, evaluate measurement support and extend the core measurement activities.

Figure 2.5 presents the measurement process proposed by ISO. The driver of the process is the organization’s information needs, whereas the products of the process are the information products that satisfy the said needs, with the aim to support better decision-making. The items numbered 5.1-4 refer to the activities discussed on pages 10-11 of the International Standard under the topic “3.3 – Organization of this International Standard”.

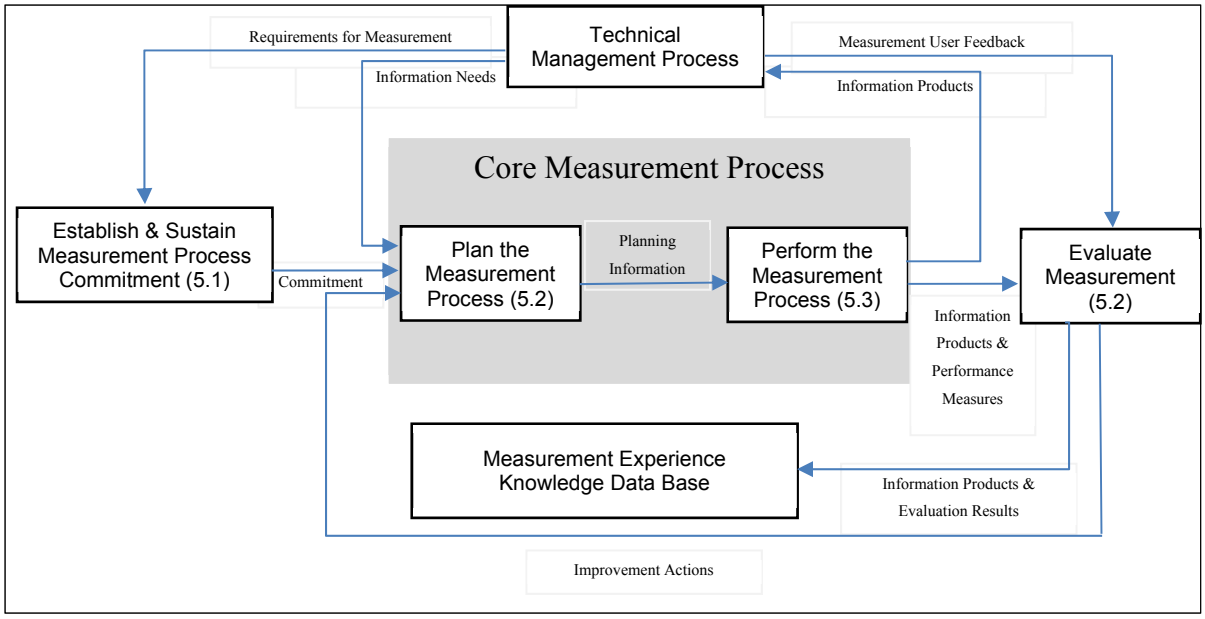

Figure 2.5 - ISO/IEC 15939:2007 - Measurement process

Discovery, creation or selection of measures is a process that requires careful validation. Jacquet and Abran present the validity issues while proposing a process model for software measurement methods (Jacquet & Abran, 1997). The validation is addressed by three different approaches: validation of the design of the measurement method, validation of the application of the measurement method and validation of the use of the measurement results in a predictive system. This validation method is further discussed in section 4.4.

Measurement process is well defined in multiple literature entries and from different perspectives (ISO/IEC, 2003), (Kaplan & Norton, 1992), (Alinezhad, Masaeli, Esfandiari, & Mirhadi, 2010). It is one of the axiomatic components of the Plan-Do-Check-Act (PDCA) cycle defined by the ISO/IEC9000 and is therefore of great importance for any engineering process that follows that standard. Through the application of measurement methods and exploitation of measurement results, it is possible to define improvement points for processes. The design of the measures must be validated in order to guarantee that the measurements yield pertinent and relevant outcomes that relate to what is expected to be measured (Jacquet & Abran, 1998).

2.1.1.4 ISO/IEC 25020 Software product Quality Requirements and Evaluation (SQuaRE) – Measurement reference model and guide

The scope of this standard is the selection and construction of quality measures for software products. Based on the Software Product Quality Measurement Reference Model (SPQM-RM), software product quality is composed of quality characteristics and sub-characteristics that are demonstrated by software quality measures acquired from measurement functions that apply previously defined quality measure elements. Internal, external and quality in use measures are referred to as part of the software product quality life cycle.

Internal software quality measures are defined and implemented during its development. External software quality measures are related to the behavior of the system where the specific software product is inserted. Quality in use measures come from the product's ability to meet the user's needs. All these measures should be applied during the software life cycle to achieve effective software quality management.

Quality measures should contain the following properties: name, corresponding characteristic and sub-characteristic, measurement focus, purpose statement, decision criteria for interpretation and action, and identification of the quality measure elements used to construct it. Performance measurement metrics should be validated and have their reliability assessed. Validation should be inferred from correlations, tracking, consistency, predictability and discrimination. Reliability and repeatability measure the variations in a measurement method, both direct and those caused by external sources.

2.1.1.5 Software Product Measurement and Measure Validation

ISO/IEC 25000 determines that there are 3 forms of quality measurement for software performance: internal, external and quality in use. Each form possesses different, inter-complimentary primitives and measurement methods, and all of them require validation.

Quality measures are explained in detail in ISO/IEC 25022, ISO/IEC 25023 and ISO/IEC 25024 for internal, external and quality in use perspectives.

Internal software quality is related to the intrinsic characteristics of the coding, assembling, testing, project management, documentation and reporting that is present in a system. It can be assessed during the early software lifecycle through numerous software engineering measurement techniques (Haldestead, 1975), (McCabe, 1976), (Tsai, Lopez, Rodriguez, & Volovik, 1986). It does not allow for the inferring of future software quality in use, but it allows for the early discovery of software defects and poor coding practices.

External software quality measures are related to the outcomes of the software development, deployment, learning, operation, maintainability and adaptability. These measures can be acquired by third-party applications and external observations of the operation behavior, often via automated run-time data collection, questionnaires, surveys and interviews. Authors suggest that high internal quality can influence higher external quality, whereas low internal quality will always negatively impact external quality. External quality is only measurable when considering the software as part of a system.

Quality in use measures refer to the user's ability to fulfill their goals by employing the software and can be assessed by observing end users in real or simulated work conditions. This can be achieved by the simulation of a realistic working environment or by observation of the operational use of the product. Whereas internal quality measures can be obtained early in the lifecycle and external quality is measurable on run-time, quality in use can only be approached from a broader perspective that encompasses both the technical elements of the software development, deployment and customization as well as the non-technical human-related factors, such as learning, comfort, satisfaction and trust. An extended list of measures is presented in ISO/IEC 25020, 25022, 25023, 25024.

After analyzing the literature related to metrics validation and scientific measurement, Jacquet and Abran (1997) proposed a process model for software measurement methods that is defined

in 4 steps: design of the measurement method; application of the measurement method rules, measurement result and exploitation of the measurement result. The design of the measurement method and application of measurement method steps are subdivided into sub-steps that contain the required tasks for each step.

The first group of sub-steps relates to the design phase of a measure: definition of the objectives, where what is going to be measured is declared; characterization of the concept to be measured, as well as the definition of the most concrete possible attribute for that concept; design or selection of the meta-model, where it is possible to find the description of the entity types that will be used to describe the software and the rules that allow their identification and definition of the numerical assignment rules which will allow for the definition of a formal relation system.

Sub-steps are also included in the application phase: gathering software documentation for information about the system under study, construction of the software model where the entities for the measurement are referenced according to the meta-model and the application of the numerical assignment rules.

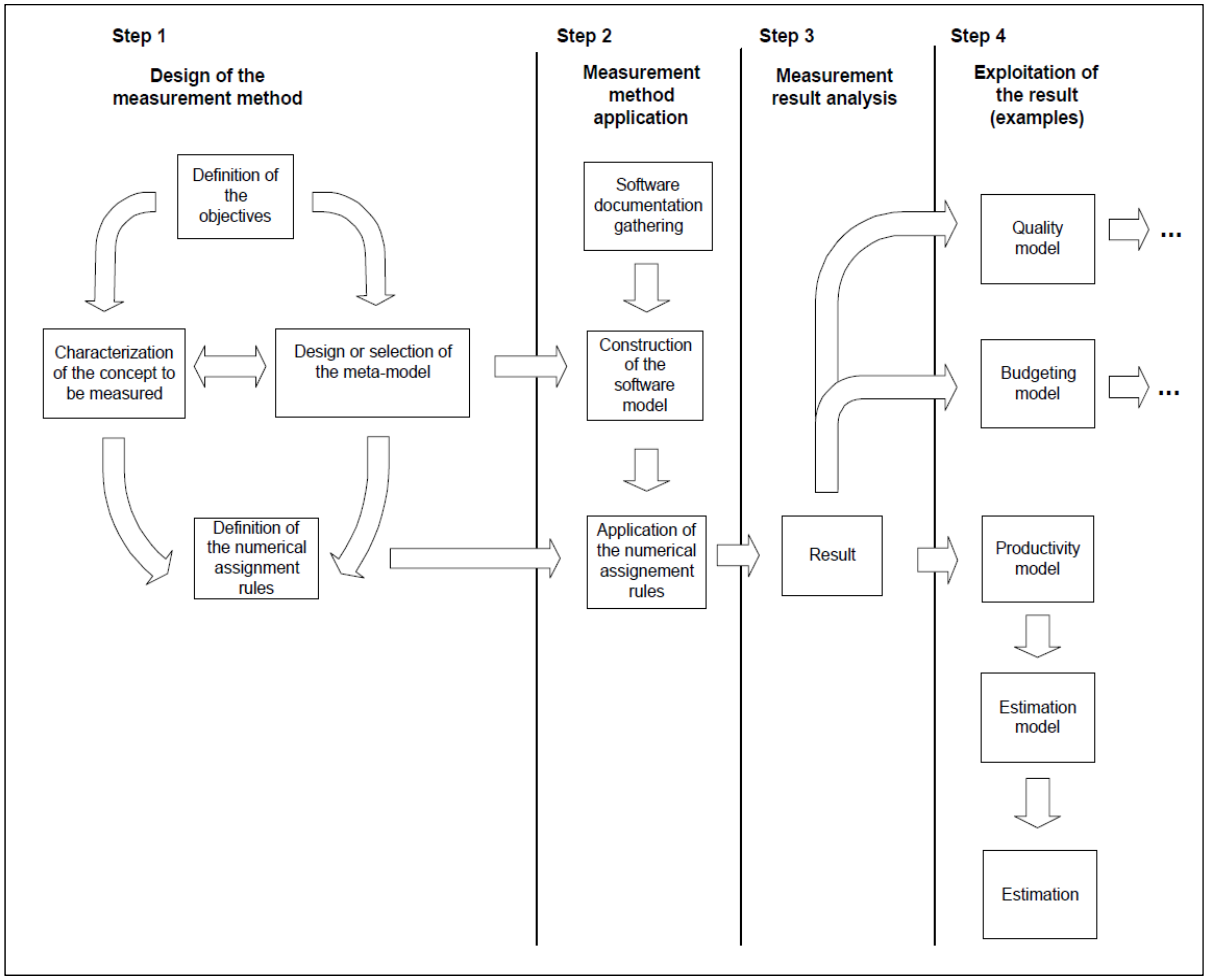

Figure 2.6 - Detailed Model – Measurement Process (Jacquet & Abran, 1997)

Throughout the software measurement definition lifecycle, measures must be validated in each of the different steps of the process, in different ways. The validation of the design of the measurement model is required in order to guarantee that the measurement method is capable of verifying the representation theorem. The validation of the application of a measurement method can be conducted both *a priori* and *posteriori*, relating to steps 2 and 3 of the process with the objective of guaranteeing that there is enough information to carry out the process as well as ensuring we have the technical understanding of the technology and rules applied. Finally, step 4 of the measurement process requires that once a measurement result is available it must be interpreted in specific contexts, for example it could be used in a predictive algorithm or system.

2.1.1.6 Limitations and difficulties of using ISO/IEC software engineering quality models in a typical organization

Many different authors discuss the difficulties of implementing the ISO standards in different industries (Sousa-Poza, Altinkilink, & Searcy, 2009) (Cagnazzo, Taticchi, & Fuiano, 2010), (Poksinska, Kahlgaard, & Antoni, 2002) (Gotzamani, 2005). A number of challenges that permeate across industries and standards are: lack of financial and human resources, inadequate technical knowledge of quality management, lack of knowledge of formalized systems and lack of ability and experience for conducting internal audits. The literature reviewed showed conclusive results that, no matter the effort involved in standardization, the outcomes were positive for the organizations and the associated stakeholders (Lamport, Seetanah, Cohhyedass, & Sannassee, 2010). The difficulty in applying ISO standards related to quality measurement of software are also reported in industries such as energy, mass production and extraction, which are historically the most mature applications of contemporary engineering. It is also stated that such "basic" factors such as knowledge, ability and investment are the recurring factors that affect the use of ISO standards in these industries.

The discipline of software engineering, when considered from an epistemological perspective, presents additional challenges that are mostly related to its immaturity. The term software engineering became prominent following a NATO workshop held in 1968 (Naur & Randell, 1969) where the expression was minted to bring attention to the shortcomings of current software developers. According to Mary Shaw (Shaw, 1990), software engineering has been following the historical evolution of other engineering disciplines, as described in Figure 2.7 (Finch, 1951): starting as an artisanal trade, moving to commercial, scaled, scientific and finally professional engineering. Software engineering came into existence as an ad-hoc approach to solving problems and then progressively became more systematic as more engineers developed/improved/normalized its practices. An artisanal approach to problem solving focuses on implementing a solution based on an individual's experience. It would then become a set of skills and knowledge that each artisan would master differently depending on the specific task. With the passing of time, peers would agree on a set of techniques that, when

used to solve specific problems, would yield known results. These techniques, or "practices", then become part of the practitioners' skills, leading to formal codification that would then be turned into process, models and theories that could, potentially, lead to best practices adopted by the whole industry.

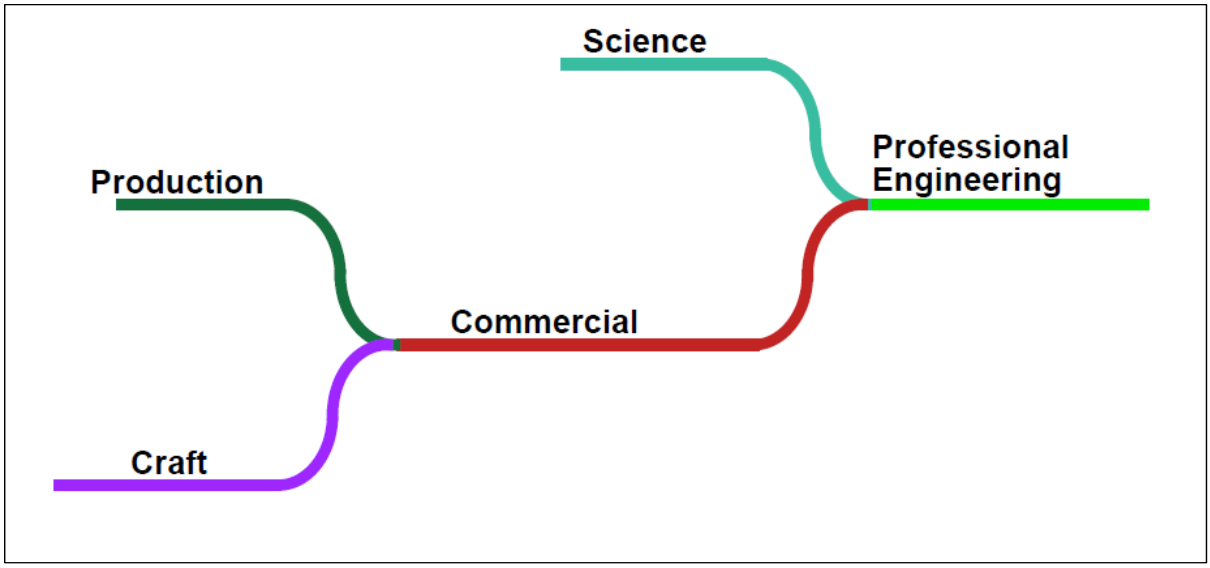

Figure 2.7 - Evolution of engineering disciplines (Finch, 1951)

Software engineers have stated difficulties with utilizing software quality measurement standards like ISO/IEC 25000 including a number of factors related to:

- Lack of knowledge of formalized systems influences and is influenced by the immature state of knowledge in the field; most of the software engineering best practices are not widely adapted because they aren't widely known by practitioners;
- Inadequate knowledge of quality management causes and is caused by the difficulties of creating a high-quality software product; as demonstrated in Table 2.1, different stakeholders have different expectations of the outcomes of the software, so even the measurement of "good enough" is elusive; (Bach, 1997)
- Lack of financial and human resources are a cause-and-consequence of its own; if the software engineering lifecycle cannot make clear how high quality software systems improve the organization's outcomes, there will be less organizational commitment in high quality;

- Lack of ability and experience in performing quality audits is the result of not knowing the standards and norms that already exist or the new model being developed.

This research aims to consider some of these factors in the proposition of the solution. The approach of both the business and the engineering perspectives aim to bridge the gap between business and science, highlighting the value of high quality software engineering standards for the organizations. The utilization of Big Data technology is expected to allow for the prototyping of these concepts as the amount of data to be processed surpasses the capacity of current technology and it could simplify the interpretation of the case study results.

2.1.1.7 Section conclusion

This section has presented the internal, external and quality in use measurement process, steps and validation. Note that the more external the measure, the more complex it is to acquire data for its measurement. Additionally, the validation of measures must be conducted in order to guarantee that the measurands are related to the measures and to the desired outcomes.

The information systems performance measurement process is a complex topic that includes the challenge of understanding the expectations, needs and desires of the organizations. It is not widely understood or employed by organizations. The current software engineering terminology used in this domain does not easily translate into the day to day business reality, which hinders its broader application in organizations. Also, quality improvement and achieving high quality of software products and systems does not always receive the attention needed or the executive commitment from organizations for such a quality system, as presented in ISO/IEC 25000, to be effectively implemented.

It is important to note that different stakeholders performing the same daily functions can have different mental models, in relation to quality, satisfaction and success measures. Considering these perspectives and external influences, modeling end user perceived performance is a

challenge in both technical and non-technical aspects. Complex information systems with components that dynamically self-organize (as in clustering and fail-overs) are operated by different individuals that may aim to achieve the same business result when using the systems, and these different perspectives might influence the expected quality outcome of the information system; one stakeholder could be satisfied concerning a particular result, while another individual might not.

It is a well-known fact that software engineering is a very young domain, and it is evolving in a way that is analogous to other engineering disciplines, experiencing the same standardization challenges and difficulties as other, more mature engineering domains had in the past, such as lack of financial investment, human resources, inadequate technical knowledge of quality management, lack of knowledge of formalized systems and lack of ability and experience for conducting internal audits. Additionally, incomplete and evolving formality, paradigms and sometimes the use of current artisanal practices also delay the acceptance of the importance of standardization in the domain of software engineering.

2.1.2 Performance Measurement – Business perspective

This section presents the business perspective of software performance measurement by contrasting the differences from: 1) a software engineering perspective, and 2) a business perspective. Outside of the intrinsic divergences between the foci of the approaches, the end user performance perspective is impacted by a mix of the available resources for performing a set of tasks, the end user motivation and his engagement (Hutchins, Hollan, & Norman, 1985) (Davis & Wiedenbeck, 2001) and factors such as training (Marshall, Mills, & Olsen, 2008), perceived usefulness, ease of use (Davis F. D., 1989), support, anxiety and experience with technology (Fagan & Neill, 2004) that influence the user's ability to actually perform the task. It is important, in this context, to present the techniques that business employs for managing the qualitative and subjective aspects of performance measurement.

2.1.2.1 Key Performance Indicators and the Balanced Score Card

Section 2.1.1 described how performance measurement from a software engineering perspective focuses mainly on designing/identifying valid measures and measurement methods, collecting relevant data and properly exploiting the result of the measures. Performance measurement from a business perspective, on the other hand, focuses on the ability to provide managers with timely information for decision making; information that allows stakeholders to plan and react accordingly to scenarios that can be unfavorable to the organization. Whereas the software engineering perspective is interested in the intrinsic quality of the software product or service, the business perspective measures the effects of the quality in use and upon the organizations' ability to achieve its goals.

In a similar way as measures presented by the ISO/IEC standard, performance measurement from the business perspective often uses Key Performance Indicators (KPI) as a popular technique for measurement (Chandler, Strategy and Structure: chapters in the history of the American Industrial Enterprise, 1962). Multiple publications address its definition, development, creation, documentation and analysis (Eckerson, 2013) (Marr & Creelman, 2011). KPIs are defined as being an abstract construct, derived from quantitative measures that indicate the proximity of the quality level of a working process to its desired goal. A good KPI specification is said to follow the S.M.A.R.T characteristics, i.e. Specific, Measurable, Achievable, Relevant and Timely (Parmenter, 2010). Table 2.2 presents an example of one KPI for performance management. One of the most popular business performance management techniques and concepts is the Balanced Score Card (BSC). A BSC is a business performance measurement framework that adds strategic non-financial performance measures to traditional financial measures already used by managers. The objective of the BSC is tying in the different measures that, when combined, document and identify an organization’s success while allowing for executive action on the results of individual KPIs (Kaplan & Norton, 1992). Figures 2.8 and Table 2.3 present the classic strategic map from the literature as well as a simplified strategic map for information technology objectives.

Table 2.2 - Generic KPI – Average processor utilization for servers

Significance	High processor utilization causes delays on the processing of new orders
Measurement	Real Time
Expected Behavior	Managing workloads and upgrading processors should reduce the average utilization
KPI	Average Processor utilization, servers
Objective	The measure should be below 80%

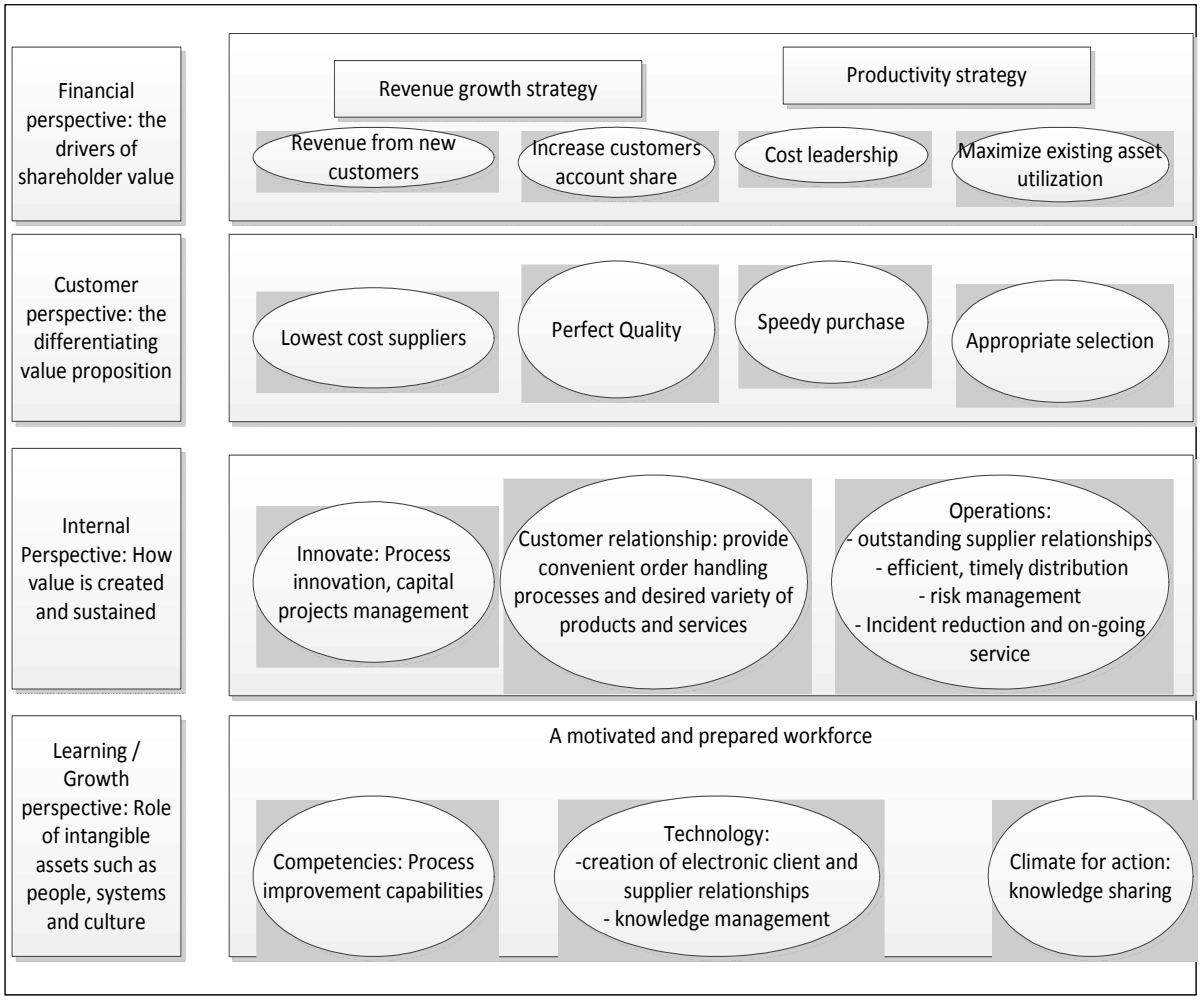

Figure 2.8 - Balanced Scorecard Strategic Map – adapted from (Kaplan & Norton, 1992)

Table 2.3 - Generic Strategic map containing a simple IT objective aligned to the business

Theme: Contract Output	**Objective:**
Financial: more contracts signed per financial adviser work hour	Lowered service time / client
Costumer: less wait time to signature	Faster response between deal and signature
Internal: Fast printer response, less printer errors, less downtime	Less event viewer entries for printer error, less service desk ticket
Learning: Send the job to the correct printer	Less recycled paper
Expected result:	Profit improvement due to improved printer performance

The reason for describing these two specific approaches for performance measurement is based on the fact that whereas ISO 25000 is not yet largely utilized by the industry, the BSC's and KPIs are the de facto standard for IT measurement in organizations from a business perspective (Nagumo & Donion, 2006). In order to foster a better penetration of the ISO standard in organizations, it would be important to provide organizations with a methodology to use such a standard which demonstrates that it can be useful and provide understandable results for managers. Additionally, since end user performance perspective is something that is perceived individually by each end user, it is also important to provide these stakeholders with a simple representation of the performance measurement data that would allow them to readily use it and empower themselves in relation to the utilization of the information system.

The BSC usually employs 4 different, or balanced, perspectives that demonstrate the organization's performance: Business process, Customer, Financial, Learning and Growth. Business process focuses on the internal quality of the processes and how well the outcomes conform to customer needs. The customer perspective relates to the level of customer satisfaction and/or potential for yet undiscovered needs; it represents how big the organization is and its potential growth. Financial perspective is the more orthodox approach to performance management that has been used historically to measure an organization's outcomes. Learning and growth perspective includes employee training and corporate cultural attitudes towards company performance; its objective is to foster the environment where end users – both as

stakeholders and as important corporate resources – are continuously learning and increasing in value. Neither KPI's nor the BSC's have set international standards, but both have well accepted characteristics.

KPIs should be: (Chandler A. D., 2002) (Parmenter, 2010)

1) Non-financial measures.
2) Measured frequently.
3) Acted upon by senior management.
4) Clearly indicate what action is required.
5) Tied to a specific team for action and remediation (are "owned").
6) Have a significant impact on organizational performance.
7) Respond to action and remediation.

A Balanced Score Card should be: (Kaplan & Norton, 1992)

1) Widely adopted in the organization.
2) A source of objective data for business decisions.
3) Adopted and sponsored by top management.
4) Used for employee training.
5) Driver of reward and recognition.
6) Facilitator for implementing change.
7) Analytic sources of information for acting upon corporate problems.
8) Allow for the organization's performance management through performance measurement.

From this information, it is possible to identify that there are notable differences in the software quality measurement philosophy for software products between the software engineering perspective and the business perspective. Software engineering considers internal quality, external quality and quality in use, whereas the business perspective largely ignores internal and external software quality, focusing on the effects of software product quality in use.

It is also important to note that there are no standards for KPI's or Strategic Maps, as they are usually custom tailored tools that help management. There are, on the other hand, well-accepted characteristics that should be present and that can be harmonized to the quality in use model proposed by ISO/IEC.

2.2 Cloud computing

This section reviews the cloud computing literature. Cloud computing applications are part of complex systems which depend on different infrastructures that include components that are often geographically dispersed with shared elements and which are running diverse applications (Mirzaei, 2008), (Mei, Liu, Pu, & Sivathanu, 2010). This technology employs hardware and software to deliver ubiquitous, resilient, scalable, billed-by-use, application agnostic systems (Prasad & Choi, 2010). Cloud computing technology is often categorized by 3 different service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These service models can be hosted and managed in-house or offered by different third party providers. In the scope of this research, the cloud computing infrastructure that is analyzed is described as Private cloud (Iosup, et al., 2010) (Mei, Pu, & Sivathanu, 2010) (Suakanto S. , Supangkat, Saragih, & Saragih, 2012). The advantages and disadvantages of this technology, as well as initiatives for measuring cloud computing performance, are also discussed.

2.2.1 Definition

The standard definition for cloud computing has still not reached consensus, but it can be described as an "*Emerging paradigm of computer systems utilization that assumes the provisioning and usability of any IT service from the internet*". This brings the prospect of computing services acquired on demand in opposition with the historically preemptive acquisition of computer resources (Voss & Zhang, 2009). It is the most recent evolution of computer connectivity and data distribution that displays advantages and disadvantages according to different tasks.

One of the frequently cited sources for the definition of cloud computing is the one by the US National Institute of Standards and Technology (NIST), that proposes that "*Cloud computing is 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*" (NIST - National Institute of Standards and Technology, 2011).

From these definitions, it is possible to identify that cloud computing is a technology that relies on the connectivity provided by the Internet to allow access to shared pools of resources, whose utilization should be easily adhered to and relinquished without much administrative effort. These shared resources would permit a high degree of flexibility for variable workloads and could be managed via SLA that can describe the expected behavior and performance of a cloud-computing offer. The quality in use of a cloud computing offer is directly related to the quality of the network infrastructure as well as the configuration of the pooled resources. This will be further discussed in section 2.2.4.

2.2.2 Service and deployment models

Cloud computing is offered or assembled in different formats to consumers. Three formats are the most prominent: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These services can be deployed in different formats, mostly constraining cost, administrative effort, customization and privacy requirements, being Public, Private or Hybrid. For the purpose of this research, the cloud format studied is defined as a Private SaaS.

Infrastructure as a Service (IaaS) is a format where a provider offers virtual or physical computing resources (CPUs, memory, disk space) over which a customer is free to deploy and manage his own environment. This allows for a greater degree of customization, but causes a

larger overhead in management processes for the client. Amazon Elastic Compute Cloud is one example of such a service.

Platform as a Service (PaaS) is a different offer where a set of computing resources, operational systems and development tools are hosted by the provider and the customer is capable of creating services and applications that are compliant with the offer's characteristics and have a limited degree of customizability. This offers greater stability and control of computational resources, as the customer can focus on developing or hosting the products and services owned without having to spend resources on managing, updating and maintaining the infrastructure. One such offer of this type is the Windows Azure Platform.

Software as a Service (SaaS) is a format where the consumer accesses applications, services and information from a standard interface, with low customizability but no administrative effort. These applications are hosted and completely managed by the provider. One such application is the widely used Gmail application by Google.

Public Clouds are owned, managed, configured and controlled by the service providers who can then offer the cloud to third party clients. **Private clouds** are built for specific organizations, with the possibility of outsourcing its management to third parties. Finally, **Hybrid clouds** contain one or more components that are owned by private and public parties.

These distinct service and deployment models have different advantages and disadvantages that will be further discussed in section 2.2.3. Figure 2.9 presents the Private Software as a Service cloud infrastructure that will be experimented as part of this research.

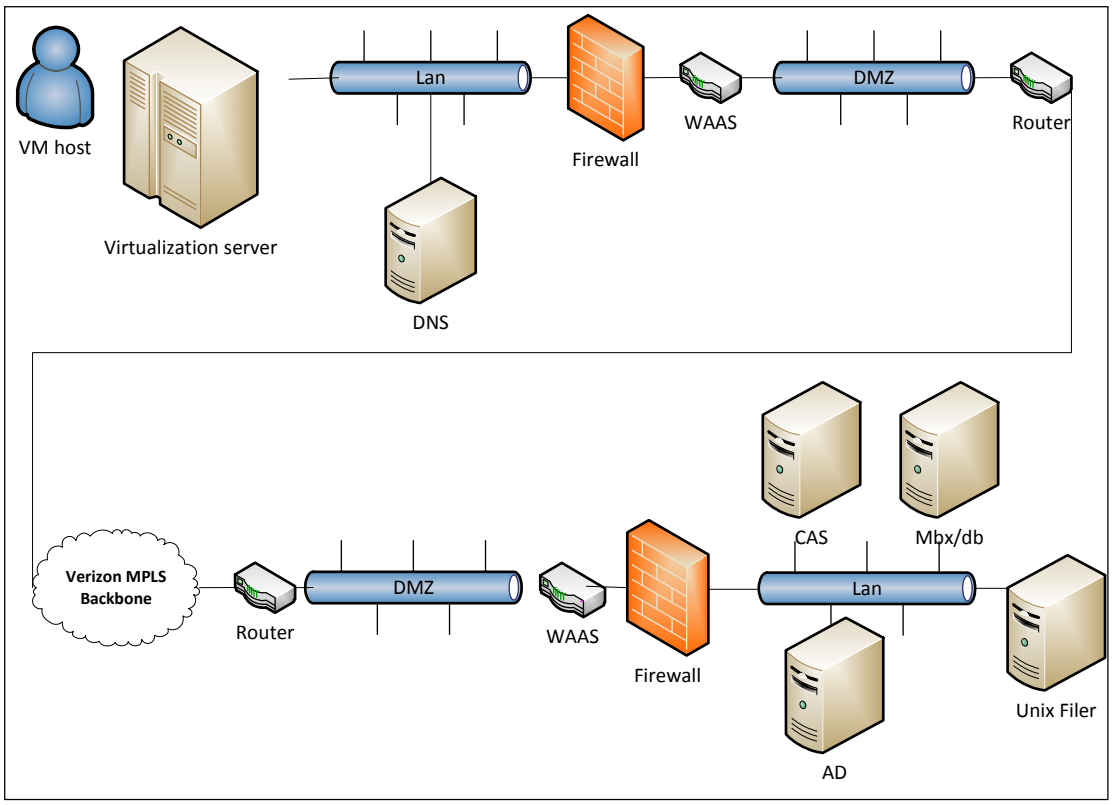

Figure 2.9 - A Private SaaS cloud that will be used in the experimentations

2.2.3 Advantages and disadvantages of cloud computing technology

A number of authors cite the advantages and disadvantages of utilizing cloud computing technology. It is important to identify that cloud computing is not a universal solution for all of the computing problems that exist and that often it is misused in lieu of other technologies. This research aims to address one of the described disadvantages, which is the unreliability of system performance due to the complexity of the infrastructure.

Advantages of utilizing cloud computing technology: (Creeger, 2009)**,** (Phaphoon, Oza, Wang, & Abrahamsson, 2012)

1) Ability to address volatile workload: due to the flexibility in provisioning more resources according to the client's needs, it is possible to quickly address fluctuations in workload size and complexity.

2) Simplification of deployment processes for development and quality assurance: by employing cloud computing technology, customers do not have to spend effort on coding applications that have to be aware of the infrastructure; it is always expected to be available.
3) Decreased time for running back end processes: the ability of pooling resources from different machines allows for the distribution of tasks based on the availability of the resources in the pool and not on specific units.
4) Larger Mean Time Between Failures (MTBF) and less downtime: the availability of a shared pool of resources enables the coexistence of clustering and fault tolerant infrastructures, permitting workloads to be roamed out of any faulty components.
5) Efficient business continuity: for the same reasons as above, when the infrastructure is composed of fault tolerant components, the business benefits from a larger degree of resiliency and continuity.
6) Possibility of shortening the cycle from idea to product: by leveraging the granularity of provisioning, instead of procuring, acquiring and configuring new components and resources, organizations are able to quickly respond to business needs.
7) Centralized auditability and security: even though the infrastructure is naturally distributed, converging points for logging and auditing might be set so that the managing and auditing of the infrastructure can be conducted in a centralized way.

Disadvantages of utilizing cloud computing technology (Armbrust, Fox, & Griffith, 2009) (Gruschka & Jensen, 2010) (Grobauer, Walloschek, & Stocker, 2011)**:**

1) Potential for incompatibility of end user behavior and enterprise processes: with large distributed systems, there are more possibilities for end users to find undocumented features within the infrastructure that may diverge from the company's expectations.
2) Data lock in and system lock in: on cloud computing systems, it is harder to pinpoint the physical location and precise ownership of any given information, exposing consumers to legal and political issues that don't exist with other technologies.
3) Decreased overall performance per processor: despite the possibility of better allocating workloads through the shared pool, a single processing unit will always have

more load than it would as a standalone unit, mostly due to virtualization and other cloud specific services.

4) Complex integration: a cloud computing infrastructure is not a trivial implementation of computer resources, often having to resort to multiple specialists, services and technologies for its assembly.
5) Risk of information leakage: faulty software components might cause the data to leak between the cloud layers (physical, core operational system, virtualization, pooling and shared resources); faulty software in one of the pooled operational systems, for example, might leak data to other machines in the pool. Additionally, misconfigured components might allow the information to leak purposely.
6) Risk of data interception: even though customers and service providers might be bound by confidentiality agreements, there is always the possibility that data can be intercepted between the different layers of the cloud.
7) Risk of security breach in the virtualization layer: for the same reason as above, there is always the risk that the virtualization layer may be accessed by unauthorized individuals.
8) Unreliable system performance due to the complexity of the infrastructure: as many components interact to process the data, it is hard to pinpoint performance issues. This characteristic is the one that is expected to be addressed by this research and that will be further explored in the following sections.

2.2.4 Section conclusion

This section reviewed cloud computing technology, its deployment and service models as well as its advantages and disadvantages. The main characteristic to be addressed during this research concerning “unreliable system performance due to the complexity of the infrastructure” was also presented. The objective is to provide a mechanism for better understanding cloud computing application performance with the use of a performance measurement framework.

2.3 Analysis of the previous research

This research considers two different perspectives for computer systems performance measurement: end user performance perspective and internal quality performance. This section discusses previous research conducted for each of these two perspectives and fundamental principles, the basis for their referencing, as well as propose an initiative for bridging any gaps between these perspectives.

2.3.1 End user performance perspective

The concern with the user's ability to efficiently interact with computer systems has long been raised and discussed (Emery, 1964). In the 60's, the issues where that different end users could have different backgrounds and abilities for exploring different degrees of information, even if the computing systems could be standardized. The same is not true for end user behavior. Much more recently, (Buyya, Yeo, Venugopal, Brober, & Brandic, 2009) suggest that there is still some convergence required for computing to be considered a utility, just like water, electricity, gas and telephone. The first point considers computing a highly technical task that requires intense end user preparation and personal ability; the second point demonstrates that, as with many other technologies, cloud computing can eventually reach the level of a utility, where end users can simply consume its byproducts without having to be concerned with the interaction. In the 80's and 90's, more research was conducted (Davis F. D., 1989) (Etezadi-Amoli & Farhoomand, 1996) (Davis & Wiedenbeck, 2001), reporting that the end user performance perspective is a complex attitudinal construct with 4 principal components being common on systems considered as good, according to the end-users: a) it should improve the average quality of the end user's work; b) it should make the end user's job easier; c) it should save the end user's time; d) it should fulfill the needs and requirements of the end user's job. Fagan & Neill (2004) further present the concept of self-efficacy – being able to fully exploit a technology – with end user anxiety, experience, support and utilization of computer systems. One of the main findings of this research is that whenever the utilized system presented a good technical response (*saving the user's time* and *fulfilling the needs and requirements of the*

user's job), the actual end user performance perspective – the performance as seen by the user - was improved.

The individuality of the performance perception and end user experience has been previously explored (Law, Roto, Hassenzahl, Vermeeren, & Kort, 2009), where the authors identified that different individuals would display variable responses to fluctuating end user experience. The perceived usefulness and task-related expectation of a system has also been positively related to end user satisfaction (Mahmood, Burn, Gemoets, & Jacquez, 2010). Five performance characteristics are proposed as relevant for the end user in other research initiatives: *task success*, *time-on-task*, *errors, efficiency* and *learnability* (Tullis & Albert, 2010). This research reinforces the concept that the end user wants to be able to perform the required tasks in a timely fashion, with the least amount of effort.

Two other researches also provide interesting points of view: a) when the end user *feels* prepared for the task to be performed with the utilization of the given tools, they will frequently present a greater degree of satisfaction and tolerance to failure whilst performing the task (Marshall, Mills, & Olsen, 2008); b) the application delivery chain–the sum of all components that are involved in making an application available to the end user–is becoming increasingly complex; this makes "seeing the big picture" as well as understanding all its elements and potential problems a harder task for technical analysts (Baer, 2011). With these findings, the following points are the reference basis of this research:

1) Cloud computing is not yet a utility; there is a high level of computer knowledge required, both for the end user and for the technical workforce, for the completion of tasks.
2) Well-performing systems must fulfill three main goals: *help end users to complete the required tasks*, *save a user's time*, and be accomplished with *the minimum amount of effort possible*.
3) The delivery of end user performance perspective, given the complexity of the cloud infrastructure, is something that challenges the interpretation of the performance of the whole system.

These three main points might be used as a basis for the next steps of the research: if the end user satisfaction is lower when the performance is degraded, could the inverse be inferred? How do end users report their dissatisfaction with a system or a service to the organization?

One possible solution for these questions is the utilization of incident management, which is the methodical approach of processes, tools and registries that allow for a system to be recovered to a predetermined level of quality (ISACA, 2012) (Adams, 2011) (ISO/IEC, 2011). Incident management can be performed via a process of collecting the incident data, submitting it for analysis, solving, and providing feedback to the user. This process should be able to capture the moment when the end user reported a degraded system performance, which would in turn be used to guide the system performance measurement process.

2.3.2 System measurement process

The problem of measuring a system's performance is not new and has been explored by numerous authors, with many tools available for the measurement and display of the values of low level and derived measures. Different approaches are implemented across the tools: some install agents on the involved components that report the measures back to the performance management database (Omniti Labs, 2014) (Agendaless Consulting and Contributors, 2017) (Tidelash Inc, 2017) (Massie, 2012) (Munin and colaborators, 2017); others monitor the measures via SNMP (Symmetrical Network Monitoring Protocol) (The Cacti Group, 2017) (Nagios, 2013) (Zabbix, 2017) (Observium Limited , 2013) (Zenoss, 2013), collecting the measurements directly and other tools store the measurements locally in performance logs (Microsoft, 2013) (Forster F. , Collectd Open source project, 2017) (Weisberg, 2013). These measures are usually processed locally for monitoring purposes or stored and processed for later use.

The measures that are processed as they are collected usually have some level of pre-defined thresholds that, when surpassed, trigger an action, such as restarting a service, rebooting a computer or generating an alarm. The measures that are processed or analyzed afterwards–

frequently use *secondary data* for troubleshooting, management and decision making purposes. Some tools are capable of conducting both the first level of processing (alerts, restarts, reboots) as well as storing data for even further analysis (Kopp, 2011) (St-Amour, 2011).

Two concerns emerge from these approaches: when the processing is performed at the moment of the data collection, is the actual processing affecting the value of the collected measures, i.e. is the measuring process affecting the measured system? If the processing is done ex post facto, is there a risk of losing data quality and not being able to make decisions in a timely fashion (Huffman, 2017) (Friedl & Ubik, 2008) (Kufrin, 2005)?

As seen in section 2.2, cloud computing increases the difficulty of understanding the service delivery infrastructure. As it is more complicated to identify which paths the service follows within the infrastructure, it is consequently harder to identify points of failure. Additionally, defining quality characteristics and effectively managing such infrastructures is difficult if the measuring process is complex. In order to study cloud computing performance, two particular approaches are of interest for this research:

1) Iosup et al. (2010) employ a set of experiments that deploy a customized application on different commercial cloud computing platforms, thus measuring the performance of the infrastructure.
2) Mei et al. (2010) use stochastic methods to simulate the end user interaction with a customized system, comparing the effects of network I/O on end user response.

The first approach offers a set of equations that allows for the comparison of different systems. The second demonstrates the cause-effect of end user interaction with a system. Certainly both researches have limitations: customizing an application isn't always a possibility for the cloud computing consumer and stochastic simulations are, by definition, non-generalizable. For measuring cloud computing performance, there should be an alternative that offers a good compromise between time to decision and generalization potential.

2.3.3 Big Data and Machine learning

The expression “Big Data” has been a current topic in recent years since, with the emergence of e-commerce and the evolution of scientific applications, businesses and academia have been able to collect larger volumes of information in relation to their pertinent transactions. One of the possible definitions of the term can be: *Big Data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze. This definition is intentionally subjective and incorporates a moving definition of how big a dataset needs to be in order to be considered Big Data* (Manyika, Chui, Brown, Bughin, & Dobbs, 2011). Another definition that is not only related to the data size could be: *Big Data technologies are a new generation of technologies and architectures designed to economically extract value from very large volumes of a wide variety of data by enabling high-velocity capture, discovery and/or analysis* (Gantz & Reinsel, 2012). These definitions for Big Data gravitate around data volume, data velocity and data variety.

Two additional characteristics also help to define what Big Data is: data veracity and data value. With large enough datasets, many statistical correlations can be discovered, but more data also means more bad data, therefore, the larger the datasets, the larger the noise they generate, which in turn increases the reliability of the data. Therefore, to validate the data during the analysis process necessitates that this noise be managed, taking into account the heterogeneity and quality of the data (Helland, 2011) (Taleb, 2013).

Data can be considered a business asset when it is used to create transparency, experimentation space, business details, tailoring and segmentation, support on decision making and improving or enabling business to new performance levels. This is where the data can develop value for the organization.

In this research, the performance data logs that need to be analyzed generate 40 million records per hour (volume, velocity). This data is related to 80,000 components, where some

components have certain counters that are not used on other components, so the data is not completely homogeneous (variety). The performance data that is being analyzed is both the result of machine-to-machine as well as human-machine interaction; sometimes, degraded performance might be explained by technical factors but there may be events where the degradation might be caused by end user activity (veracity). Understanding the performance for CCA would impact correctly sizing the infrastructure, would enable the proactive implementation of solutions before issues hinder end user productivity and provide grounds for evolving the user's perception of the performance offered (value). These characteristics further position this research as a good candidate for employing Big Data tools as a means to achieving the required calculations in timely fashion.

Big Data systems can be implemented using several different frameworks, commonly leveraging from open source projects such as Hadoop Distributed File System (HDFS), Map Reduce, HBase and Hadoop, which are an open source implementation of Google's Big Data technology (Dean & Ghemawat, 2008). Another such implementation is Apache Spark (Apache foundation, 2017), a general-purpose cluster computing system that provides API's in Java, Scala and Python as well as other high level tools such as Shark (Hhive metastore query engine), Spark SQL for structured data, GraphX for graph processing, Spark Stream for processing data from live streams and MLlib for machine learning. The latter is further investigated as part of the objectives of this research.

The MLlib library implements common algorithms that are used for machine learning, such as logistic regression, linear least squares, decision trees, Bayesian networks, clustering using k-means, dimensional reduction and optimization. This research can be particularly interesting to leverage the reduction of dimensions, do data clustering and apply Bayesian analysis as described below:

1) Dimensional reduction via Principal component analysis (PCA): PCA is a statistical method that aims to find matrix rotations where the first columns will demonstrate the largest variability and the subsequent ones will be increasingly smoothed. This would

be useful in this research in order to reduce the number of rows analyzed, allowing for different levels of optimization both on the data analysis and collection.

2) Clustering with k-means calculation: clustering is often utilized in exploratory studies when there are already some notions of similarity for the data. In the present case, clustering could be used to identify the causes for a degradation event, helping the learning phase of the Bayes algorithm.
3) Multinomial Naïve Bayes: this is an implementation of probability training and distribution of vectors that support training and learning. This algorithm could be used to forecast n-next steps for the reduced dimensions calculated with PCA, trained via k-means, in order to predict events of degraded performance.

2.3.4 Section conclusion

In this section, the main points in the literature have been discussed and compared. For measuring end user performance perspective, one recommended approach is the utilization of IT service management processes. These processes should be paired with a form of performance log analysis that allows for management decisions to be taken in a timely fashion while being simple enough for the customers of cloud computing to understand. In order to achieve such a timely and useful data analysis, Big Data and machine learning technologies that can be employed in this research are presented. These findings will be further discussed in the next sections and experimented in a case study.

2.4 Chapter conclusion

In this chapter, the literature review demonstrated the importance of measure validation, the challenge of collecting the data and the differences between the business and the software engineering perspective on systems performance management. Cloud computing is a distributed computation model that is not free from disadvantages and has one particular characteristic: its unreliable performance which is based on the infrastructure characteristics. This is the main focus of this research. Measuring end user performance, in a timely fashion

in such a scenario, can possibly be achieved with the use of both IT service management processes as well as performance log data. Big Data technologies can be employed to process and analyze the large volume of data produced by these data sources.

CHAPTER 3

Research problematic

This chapter presents the research problematic, the originality of the research, possible known solutions for the problem, the proposed experimentation and the research plan. This section expands the ideas discussed in sections 1.2 and 1.3.

3.1 Research Problematic

After a review of the topics of cloud computing and performance measurement from both business and software engineering perspectives, it is possible to state the research problem and questions as follows:

Research Problem:

Modelling end user experience on cloud computing environments with the proposition of a performance measurement model, using only data currently available from data center logs, if possible, and, because of their large size, employing Big Data technology, such as Apache Spark, for its capture and processing. If it is discovered, during experimentation, that the data center log information is insufficient, additional feedback mechanisms will be proposed.

This problem is broken down into the following more specific research questions:

1) What defines a cloud computing environment?
2) What influences end user performance perspective measurement in a cloud computing environment?
3) Are performance logs sufficient for modeling the end user performance perspective? If not, which other sources are required?
4) Can the performance measurement framework for cloud computing applications (Bautista, Abran, & April, 2012) be used for the creation of a performance perspective

model using data center logs that represent the end user performance perspective of an application that uses cloud computing technology in a timely fashion?

The motivation for this research is based on 3 main interests: first, the subject of cloud computing is recent, and as such there is still much to be researched. Additionally, the logs that need to be explored by this research comprise very large amounts of data, in the range of several gigabytes per minute. Lastly, the solution to the problem proposed by this research is different from already published approaches, and the experiments that will be performed are based on live data from real life scenarios, as opposed to simulations or small samples. This will be further discussed in sections 3.2, 3.3 and 3.4.

3.2 Originality of the research

Bautista's model proposed a novel approach for measuring performance on CCA. The present research aims to improve upon this theoretical proposal through the following means:

1) Addition of the end user performance perspective.
2) Addition of a feedback mechanism.
3) Statistical anomaly detection using Big Data tools.
4) Prediction of anomalies using Big Data.

3.3 Planned solution and validation method for the research problem

This section describes the planned solution and the validation method for the solutions proposed for the different research questions. The research questions create a connection between the research proposal, the solution and the validation methods: 1) what defines a cloud computing environment? 2) What influences end user performance perspective measurement in a cloud computing environment? 3) Are performance logs sufficient for modeling the end user performance perspective? If not, which other sources are required? 4) Can the performance measurement framework for cloud computing applications (Bautista, Abran, & April, 2012) be used for the creation of a performance model using data center logs that

represent the end user performance perspective of an application that uses cloud computing technology in a timely fashion?

The first research question is answered by the literature review which categorizes the study of CCA as a Private SaaS CCA. For the second research question, the literature review was inconclusive regarding what influences the end user performance perspective universally, and a research suggestion proposed conducting individual surveys, per application, in order to ascertain the characteristics of each particular case. This survey is described in section 4.2.

The identification of the capacity of performance logs to sufficiently model the end user performance perspective is described in section 4.6. The application of Bautista's Model is discussed in section 4.5. The proposed extension of Bautista's model, which includes the timeliness of the model, is proposed in section 4.6. Each of the steps of the proposed solution is described below with justifications for the particular approaches:

1) Association of end user performance perspective with low level and derived measures: considering that no conclusive set of measurements has been presented as a definitive descriptor of the end user performance perspective in the literature review, one recommendation suggested by the literature is to conduct a survey. A survey investigating end user complaints concerning their application performance degradation, using a transversal approach, was designed. The results were analyzed following the recommendation of Bautista's model in order to assert if the variance of means had any positive relation with the presence of an end user complaint at that same moment.
2) Mapping performance measures for CCA, platform and software engineering concepts: In this step, a manual association of the ISO 25010 performance characteristics with the measure description is conducted. This is done by reviewing each performance measurement provided by the vendors, and then associating each of them with the concept that best reflects them.
3) Validation of quality measures for representing performance from an end user perspective for CCA: In order to validate if the association and mapping of measures

and performance concepts is good, two main approaches have been suggested by the authors, mainly the manipulation of task payloads in order to simulate performance degradation and the creation of controlled disturbances on a running service where the effect of the controlled disturbance is later measured. The introduction of the measures can create disturbances. These were assessed considering that the manipulation of the task payload could affect the end user data. We ensured that the disturbances were assessed and that each end user request did not impact our measures.

4) Laboratory experiments for end user performance modelling: In this step, data from all components is collected in a controlled time window. This data is analyzed as recommended by Bautista's theoretical model. Two difficulties and shortcomings of using this model for our purposes were identified and are discussed in section 4
5) Extension of Bautista's performance measurement model: In order to address the two limitations identified in Bautista's original proposal, specifically the omission towards performance data's time series characteristics and the addition of the end user feedback, an extended model is proposed in Chapter 5. Furthermore, as one of the objectives of this research in particular the timeliness of the modelling of the end user performance perspective, the experiment will use an algorithmic approach to collecting, organizing the data, identifying anomalies and proposing the creation of a performance indicator. End user feedback is described and used as a confirmation of performance issues and is also used for predicting performance issues.

3.4 Chapter Conclusion

This chapter has described the research problematic, the originality of the research and the planned solution along with the proposed experimentation and validation. The following chapter will refer to these descriptions in order to propose the final performance model.

CHAPTER 4

Experiment

4.1 Introduction

This chapter describes the experiment that was conducted in order to test the proposed theories. Following the structure proposed in section 1.4.3, the sub-steps proposed are undertaken with 3 different activities: an initial experiment that responds to the challenges described in sections 1.4.3.1 to 1.4.3.4, a separate experiment for the sub-steps 1.4.3.4 and 1.4.3.5 and a final experiment for the sub-steps 1.4.3.6 to 1.4.3.8. These experiments will help validate the proposed performance model. Figure 4.1 describes the steps and the focus of each particular experiment, which is then explained in further detail in subsequent sections of this chapter.

Figure 4.1 begins with the division of the research problem into 4 research questions. Question 1 uses three findings from the literature review: a) There are no clear or defined relations between resource consumption and the end user performance perspective; b) one of the proposed theories, the Fitness to Task Theory, describes resource consumption as a side effect of task completion, so performance degradation is not a direct effect of resource consumption, as there are cases where the end user may want the CCA to consume as much resources as possible in order to finish a task; c) the cloud infrastructure studied is characterized as a private SaaS CCA, meaning that the research has the underlying limitations inherent to this CCA model.

Question 2 invokes the investigation of what influences the end user experience. Given that the literature does not provide a straight answer to this question, a survey was designed, using service tickets, which established a relation between the fluctuations of the performance measure values observed as a potential cause of degradation. This was validated in sub-section 4.4 where a simulated load increased the time for job turnaround, implying that the task performance would be degraded in cases where the resource utilization increased.

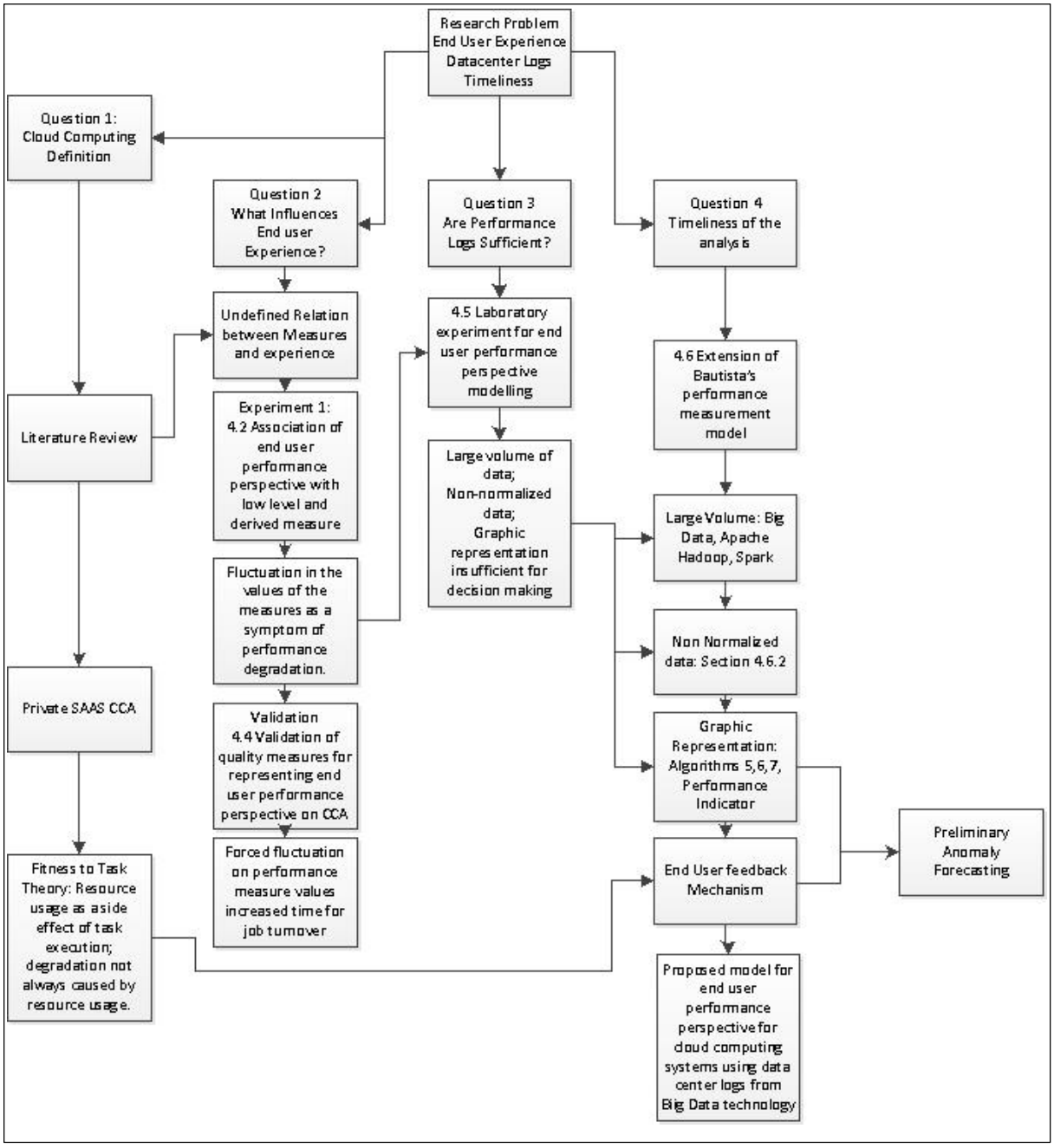

Figure 4.1 - Research and experiments schema

Question 3 examines the idea of identifying whether logs are sufficient to model the end user performance perspective. Starting with the concept that fluctuation in measure values is a symptom of degradation, the laboratory experiment described in section 4.5 was conducted through the collection of data from 38 components. The experiment identified the large volume

of data necessary for representing the end user performance perspective; the non-normalized characteristic of the data and the first attempt at graphically representing the performance was insufficient for decision-making. This led to the final research question which considered the timeliness of the analysis.

The experiment described in section 4.6 is an extension of Bautista's performance measurement model where the large volume of data was considered with the utilization of Big Data tools, namely Apache Hadoop and Spark. For the non-normalized data, the procedure described in section 4.6.2 was used whereas for the graphical representation, algorithms 5, 6, 7 and the Performance Indicator are suggested. The addition of the end user feedback is a response to the characteristics of the Fitness to Task theory, which culminates with the proposition of the performance model for end user performance perspective on CCA. The end user feedback was also used to create an initial approach to forecasting performance anomalies that could serve as a basis for future research.

4.2 Association of end user performance perspective with low level and derived measures

As previously mentioned, the literature describes that systems performance measurement is conducted in many ways. One popular approach is to use data center logs to assess the performance of systems. Many commercial, open source, and easily accessible log tools are available today for collecting, analyzing and generating performance dashboards that present different measures for the Cloud Computing System (CCS) (Microsoft, 2013), (Kopp, 2011), (Omniti Labs, 2014), (Agendaless Consulting and Contributors, 2017), (Tidelash Inc, 2017), (Massie, 2012), (Munin and colaborators, 2017), (The Cacti Group, 2017), (Nagios, 2013), (Zabbix, 2017), (Observium Limited , 2013), (Zenoss, 2013), (Forster F. , Collectd Open source project, 2017), (Weisberg, 2013). How these log tool measures can be analyzed and interpreted and the impact of the measurement results on the organizational goals, especially the end user's perspective, is explored in sections 4.1 to 4.3 of this research (St-Amour, 2011).

The fundamental concept that needs to be tested initially is the assumption that performance log measure values can reflect the end user perception of performance, specifically, whether the end user really perceives a degraded performance whenever a performance log measure reaches a certain threshold. This exploration is described in the following sections: 4.2.1 describes the experiment, 4.2.2 presents the data analysis and 4.2.3 provides findings and a conclusion.

4.2.1 Experiment description

In order to investigate if there is any association between the performance log measure values and the degradation of performance as perceived by an end user, a survey was conducted using a "trouble ticket system" that is used by the end users of a particular CCS to report problems with their application. It is assumed that the end users, when affected by a performance degradation event, will report such an event to the help desk so that their functionalities are recovered. The following methodological protocol is applied during the case study:

Data collection: Data is collected from two different sources: a) the Information Technology Service Management system (ITSM) that is accessed and maintained by the help desk for record keeping and b) the data center logs that are automatically collected. During the case study, we received 30 complaints at the help desk and collected approximately 4 GB of data center logs for this application.

Data organization: For the help desk tickets, the data collected is concentrated in the smallest time segment possible in order to represent the most amount of complaints with as minimal environmental variation as possible. For the performance logs, three different work windows are open: 1) the moment the degradation report was reported at the help desk, 2) the three hours preceding this report, and 3) the preceding week. Once this data is collected, the LLDM are associated with the ISO quality characteristics. Then a data analysis is conducted. Two distinct processes are used for analyzing the data. First, the ticket information is manually read to clearly identify the performance issues. Second, statistical data from the logs is compared for

the three previous work windows as well as between reports in order to identify similarities. Figure 4.2 describes the relative comparison between similar days and referential week data.

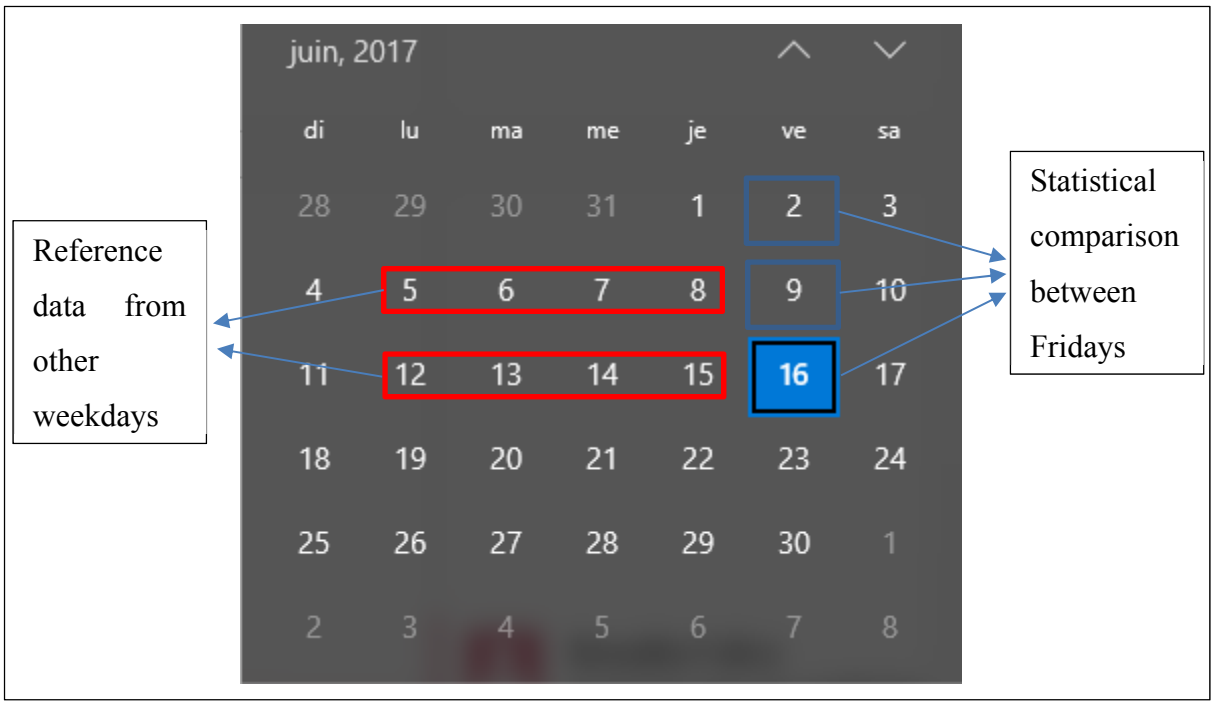

Figure 4.2 - Relative presentation of collected and referenced data

Data interpretation: This is conducted in order to identify the possibility of mapping the end user complaints to the LLDM's and then the LLDM's to the ISO quality characteristics. This would offer a method for monitoring LLDM's to 1) understand the end user perspective and 2) generate quality indicators for the application software under study.

4.2.2 Data Analysis

This section describes the analysis of the end user reports of degradation and performance log data collected as described in section 4.2.1. The main objectives of this analysis is to identify the list of LLDM associated with ISO quality characteristics.

In order to identify the desired list of LLDM, 2 steps are taken, in particular, a discovery of what conditions lead the user to file a degradation report (identification of degradation reports) and then the statistical analysis of the data relevant to such identifications (data extraction and organization).

1 – Identifying degradation reports: The tickets logged at the help desk were manually analyzed for keywords (i.e. “slow”, “hanging”, and “slowness” amongst others). Tickets that contained these keywords were flagged as potential performance degradation issues.

2 – Data extraction and organization process: The performance data associated with the degradation report was extracted for 1 week of time. A total of 63589 data points were collected, with 38 distinct LLDM. We observed 33 high degree correlations (i.e. >+0.74), with 12 representing a strong negative correlation (i.e. <-0.60) from which we reduced the 38 initial measures to 15. These were selected based on the variance and kurtosis of each value and also based on the logical response of being regarded as being available when the value is lower. For example, Memory_Committed_Bytes is selected instead of Memory_Available_Bytes, mainly because both are strongly uncorrelated (i.e. -0.98) and because the smaller the amount of committed bytes, the more bytes will be available. For this case study, the selected list of LLDM is listed in Table 4.1, along with their association to the quality characteristics described in ISO/IEC 25023.

The five-minute average of each of the values of the LLDM monitored during the 3 work windows is described in Table 4.2 (degradation report, “Time 0” ; previous three hours “Time -1” and previous week “Time -2”). These values are discussed further in section 4.2.3.

Table 4.1 - Association of the identified LLDM and ISO 25023 concepts

LLDM	Concept
%_Processor Time Committed_Bytes_MB Disk_Free_MB Process_%Processor_Ut I/O_Read Private_Bytes	Performance Efficiency – Resource Utilization
Page_File_%_Used Avg_Disk_Read_Queue Avg_Disk_Write_Queue Connections_Active Pages/Sec Handle_Count Thread_Count	Performance Efficiency – Capacity
Connections_Failures	Reliability – Maturity
Connections_Reset	Reliability – Fault Tolerance

Table 4.2 - Average LLDM value for the machines identified in the degradation reports.

LLDM	Time-2	Time-1	Time-0
%_Processor Time	39%	34%	78%
Committed_Bytes_MB	4.2 GB	6 GB	8.2 GB
Disk_Free_MB	24%	24%	24%
Process_%Processor_Ut	2%	4%	54%
I/O_Read	150/s	207/s	3512/s
Private_Bytes	204 MB	267 MB	402 MB
Page_File_%_Used	14%	9%	9%
Avg_Disk_Read_Queue	0.00	0.00	2.05
Avg_Disk_Write_Queue	4	16	104
Connections_Active	5	6	6
Pages/Sec	104.00	168.00	1100.00
Handle_Count	60002.00	45123.00	30098.00
Thread_Count	1960.00	1701.00	1801.00
Connection_Failures	0.00	0.00	5.00
Connections_Reset	12.00	14.00	37.00

4.2.3 Experiment conclusion

This section discusses the results presented in Table 4.2 of section 4.2.2. These results are the foundation for subsequent research steps and experiments. Table 4.2 presents the list of the average values of the specified measures for the components in the moments when the end user reports the performance degradation. For 10 of the measures, the Time-0 value (moment of the degradation report) is significantly higher than the Time-1 and Time-2 values. Five measures do not display the same behavior, even though they display high variance and skewedness. The difference ratio between measures is also very disparate; some measures are 15 times larger in Time-0 than the other times listed, whereas processor utilization, for example, is only 2 times higher.

It would be possible to assume that, for this experiment, the values of the measures fluctuate as a symptom of performance degradation. The ratios, frequencies and relevance of each particular measure will be discussed and analyzed as the search for a cause of the degradation event proceeds. Under the assumption that the degradation happens in moments that are also supported by the values of the performance measures, the experiment is further developed in sections 4.3 and 4.4.

4.3 Mapping performance measures for CCA, platform and software engineering concepts

Mapping the performance measures from the CC platform to the quality concepts of Bautista's model is a required step in order to establish a relationship between performance data and ISO quality concepts. This data is typically collected using performance logs. In this list, we would like to have the most granular level of performance data possible that is provided by the log tools. Table 4.3 presents an excerpt of the full list of performance data from which CCS component types are collected. The complete list is provided in Annex 2. It is important to note that while there are only 57 types of log data in the table presented in Annex 2, 24 of them, identified with a "*", are data sources applied to multiple disks, processors, processes and network interfaces and will fluctuate depending on the activity level at any given moment. An

expanded description of each measure can be found in the documentation of the original sources, as created by the owner of the appropriate systems (Microsoft, 2013) (Forster F. , 2017) (Weisberg, 2013).

Table 4.3 - Excerpt of the Data Collected and the location and type of CCS component (where * means affecting multiple components)

Performance Log Data Measure Name	**CCS component type**
\LogicalDisk(*)\Free Megabytes	Client, Server
\Netlogon(*)\Average Semaphore Hold Time	Server
\Memory\Page Faults/sec	Client, Server
\Memory\Available Bytes	Client, Server, network
\Memory\Pages/sec	Client, Server
\Paging File(*)\% Usage	Client, Server
\System\File Read Bytes/sec	Client, Server
\System\File Write Bytes/sec	Client, Server
\System\System Up Time	Client, Server
\System\Processor Queue Length	Client, Server

The data presented in Table 4.4 is an excerpt of the table in Annex 2 and describes the association between the measures above and the quality concepts of ISO 25010 for efficiency (i.e. time behavior, resource utilization, capacity) and for reliability (i.e. maturity, availability, fault tolerance, recoverability) used in the Performance Measurement Framework for Cloud Computing Applications (PMFCCA) describe by Bautista. It is possible to identify some imbalances in the quantity of performance log data types associated with each concept, similar to what has been already reported by Bautista et al. This could lead to a discussion on how to effectively design a CCA, which is not within the scope of the research reported here.

Table 4.4 - Excerpt of the association between performance log data and PMFCCA quality sub-concepts (where * means affecting multiple components)

Performance Log Data Measure Name	**ISO 25000 Quality Concept**
\LogicalDisk(*)\Free Megabytes	capacity
\Netlogon(*)\Average Semaphore Hold Time	maturity
\Memory\Page Faults/sec	maturity
\Memory\Available Bytes	capacity
\Memory\Pages/sec	time behavior
\Paging File(*)\% Usage	time behavior
\System\File Read Bytes/sec	resource utilization
\System\File Write Bytes/sec	resource utilization
\System\System Up Time	availability
\System\Processor Queue Length	time behavior

The asterisks in Table 4.3 refer to the same meaning as for those in Table 4.4, i.e. data sources are applied to multiple disks, processors, processes and network interfaces and will fluctuate depending on the activity level at any given moment. It may be the case where multiple processes or service interactions span different instances of counters, each collecting data for a particular process or service. In this case, multiple counters for the same processes are named with a "#" and a number, according to the order in which each instance of the same process is invoked. For example, if a CCS component such as an end user desktop computer has two Internet Explorer applications running, the performance counters would be \Process(Iexplore)\% Processor Time and \Process(Iexplore#1)\% Processor Time.

4.4 Validation of quality measures for representing performance from an end user perspective on CCA

This step describes an approach for validating whether the measures selected in section 4.2 are related to the problems reported. These measures were selected from the performance degradation reports and represent the end user interaction with specific systems. This step

follows the recommendation of Suakanto et al. (Suakanto, Supangkat, & Suhardi, 2012), where a simulation is conducted on equal workloads.

4.4.1 Validation description

In section 2.3.1, we identified that one of the user's interest is the ability to perform a task, in a timely manner, with the least possible amount of effort. In order to validate the measures collected, it would be necessary to measure the ability of an end user to complete the said tasks, if they are homogeneous, under different configurations of the CCA, which would in turn provide feedback that the measurement is indeed relevant. Only the measures associated with the ISO/IEC 25010 performance concepts of time behavior, capacity and resource utilization where matched with log measures presented in section 4.1 were collected.

In reality though, the question to be asked of the end user would be to repeat the same task, which could bias the end user towards the system utilization. Given that, a simulation is used to perform the automated task akin to what an end user would execute. This same task is repeated across 30 physical components, inserting different signals for testing each of these measures. The objective is not to test the degree of the effect of each particular load, nor the possible combinations of each contribution. At this time, the objective is only to measure the turnaround time and if it is a longer time, it will suffice to indicate that when the particular measure is affected, the end user would be able to identify degradation. Each of the measures was manipulated in a particular way, relative to its particularity. Some manipulations affect more than one particular measure; an observation that is further discussed in sections 4.4 to 4.8. Table 4.4 contains an excerpt of the measures that where manipulated and how.

Table 4.5 - Performance measurement and manipulation technique

Performance Log Data Measure Name	**Manipulation**
\LogicalDisk(*)\Free Megabytes	Local disk filled with random data
\Netlogon(*)\Average Semaphore Hold Time	Local logon server with a high processor utilization
\Memory\Page Faults/sec	Loading and unloading random sets of data
\Memory\Available Bytes	Loading random data on the memory until the memory is exhausted
\Memory\Pages/sec	Forcing context-switching on a list of 1MB random text data files
\Paging File(*)\% Usage	Continuously loading random data on the memory after memory is exhausted
\System\File Read Bytes/sec	Opening and closing a list of 1MB text data files
\System\File Write Bytes/sec	Continuously writing a contiguous block of random data with a 1MB size
\Processor(*)\% Processor Utilization	Generating hash for random data blocks
\System\Processor Queue Length	Opening and closing a list of 1MB text data files

Table 4.5 describes different actions that were employed in this experiment while measuring turnaround time. The experiment was conducted on 30 identical HP Moonshot 700p components with the exact same task: sending and receiving a particularly large message (30MB) that had to be scanned by the server for specific keywords in the body of the message. This was then replicated once for each of the measures, across all components. Algorithm 1 describes one of the scripts used for running the simulation with the different parameters.

Algorithm 4.1 - Performance measurement validation simulation

```
Activate(Manipulation Action);
Start Timer ();
Initiate Send-Message-Requires-Safety-Scan();
Stop Timer;
```

The script presented in Algorithm 1 above was run across all components one time for each manipulation task. This resulted in 1740 simulations. The data is analyzed in section 4.4.2.

4.4.2 Data analysis

The objective of the simulations is to identify the cases where the manipulation of a particular measure increases its utilization to the maximum possible and impacts the job turnaround. The time of each job turnaround was compared to the average of 100 passes of the same workload, for a total of 3000 simulations. The average undisturbed job turnaround is 1 / 138,004 ms (~ 2 minutes per message). Table 4.6 contains an excerpt of the measures, the value of the measurement after the manipulation and the relative effect on the average job turnaround time.

Table 4.6 - Excerpt of Performance Log Measures, the simulation values and the effects on job turnaround.

Performance Log Data Measure Name	**Value**	**Relative effect**
\LogicalDisk(*)\Free Megabytes	0	Increase
\Netlogon(*)\Average Semaphore Hold Time	750 ms	Increase
\Memory\Page Faults/sec	~ 150,000	Increase
\Memory\Available Bytes	~ 175 MB	Increase
\Memory\Pages/sec	~ 19.000	Increase
\Paging File(*)\% Usage	96%	Increase
\System\File Read Bytes/sec	~ 40MBps	Increase
\System\File Write Bytes/sec	~22MBps	Increase
\Processor(*)\% Processor Utilization	100%	Increase
\System\Processor Queue Length	12	Increase

The data collected indicates that affecting the resource utilization, capacity and time behavior measures indeed increases the job turnaround time. Different measures had different contributions to the increase, but the actual contributions are not the focus of this simulation.

4.4.3 Validation conclusion

The measures identified in section 4.2 increased the time of the job turnaround in all simulated events for the measures that are associated with the performance concepts of resource utilization, capacity and time behavior. Different measures reported different contributions to this increase, as described by Bautista. This validates the measures accordingly, which then allows for the construction of the laboratory experiment described in sections 4.5 and 4.6.

4.5 Laboratory experiment for end user performance modeling

From the results of the previous research findings presented in sections 4.2, 4.3 and 4.4, we successfully associated the LLDM with the end user performance perspective, mapping them to ISO/IEC 25010 performance concepts. We validated this process by manipulating the values of the LLDM and measuring the job turnaround times in a simulation of the impacts that would be perceived by the end users. Using this acquired knowledge, we plan to build a laboratory controlled experiment that collects the data for the LLDM and uses the values of these measures to represent the end user performance perspective.

4.5.1 Description

For this initial laboratory experiment, a total of 30 different desktops where selected, in a single physical location, as well as 5 network devices and 3 email servers. The objective was to investigate the possibility of modelling end user performance perspective from the live data generated by these components. Each of the 38 investigated components could generate up to 700 different LLDMs.

4.5.2 Setup

The components were configured in a way that they generated comma-separated-value (CSV) text files, with each time interaction recorded as a new line on the file and each measure

occupying a different column on the file. These files were automatically stored in a shared network area storage (NAS). The NAS was the repository that was used to collect the data and perform the analysis. This particular setup proves to be problematic as discussed in sections 4.2.3 and 4.4.3. In order to process the data generated in this experiment, a single Intel Xeon HT 4 core desktop computer with 32GB of Read-Only Memory (RAM) was used.

4.5.3 Data preparation

In the data preparation phase, it was necessary to clean up the null data and to create a single repository from which the analysis can be performed. This involved 2 different steps: the interpolation of data, i.e., the reading of the multiple files on the NAS and creating a copy of their linear contents to a new file on the same NAS using the time stamp as the main index for this pseudo-database. Algorithm 2 represents the pseudocode for collecting the data and Algorithm 3, the relevant code for organizing it.

Algorithm 4.2 - Experiment 1 data preparation

```
While True()
        For each (file in storage) { tail file >> analysis.csv}
```

Algorithm 2 simply browses the NAS contents, sending the last lines of data from each file to the analysis text file, which is then organized according to Algorithm 3.

Algorithm 4.3 - Experiment 1 data organization

```
For each (line on analysis.csv) order by time, host, column
For each (value in each cell), check if not valid, value in cell = 0, move next.
```

Algorithm 3 recursively orders the lines by time, name of the component and name of the LLDM. Three issues where encountered when performing this data organization: 1) As the components generated files on a shared network storage, there were occurrences of file locks when multiple machines attempted to access the same resources; 2) The algorithm used for

sorting the data is not optimal and recursively runs through all data in order to sort it in a usable format, which could lead to processing times that are impractical for the decision making process; 3) Much of the collected data came in as null, which forced the manipulation of the original values by setting them to 0.

4.5.4 Analysis

As the data was organized, a manual investigation of the analysis file could be performed. This investigation involved describing each line of the file graphically using a barchart representation, so that the performance could be understood in such a way that the higher the bars, the more degraded the performance. Figure 4.3 displays one attempt at representing this data.

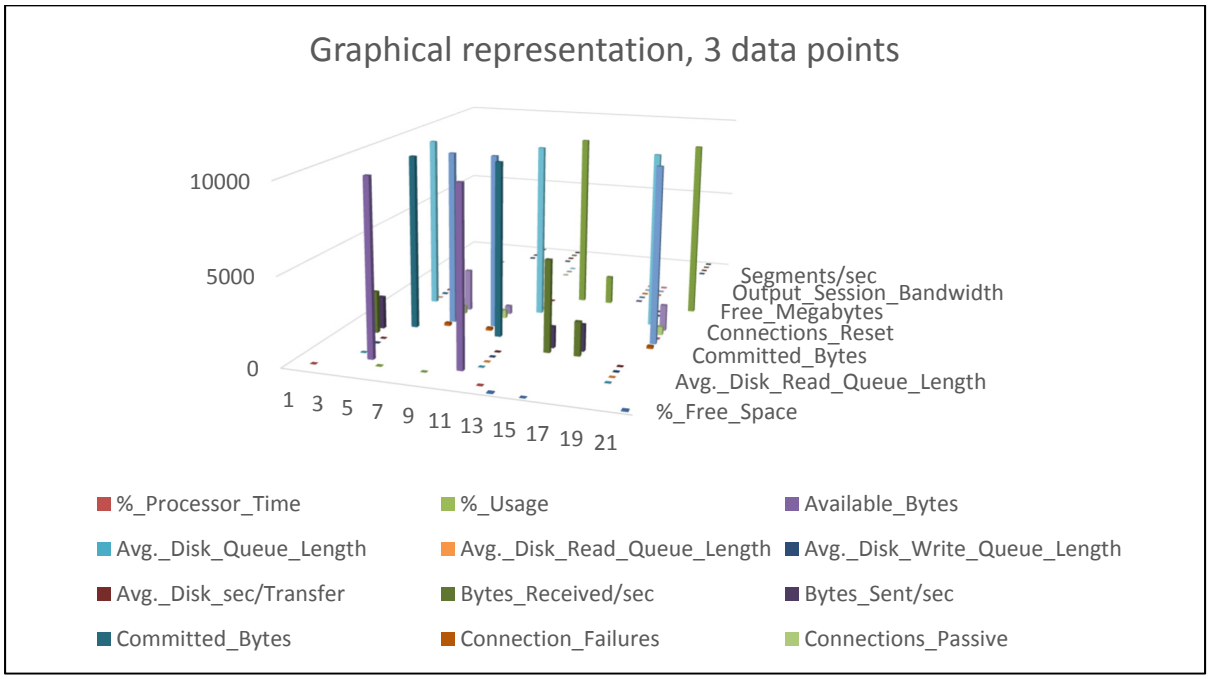

Figure 4.3 - Graphical representation of the data for 3 consecutive points in time

Figure 4.3 shows that it is possible to identify which measures emerge more frequently, indicating possible culprits for degradation of the end user performance perspective.

4.5.5 Experiment conclusion

This experiment applied the concepts presented in sections 4.2, 4.3 and 4.4 in order to provide an initial approach to the solution of the research problem. This first experiment encountered issues such as: processing the large volume of data, the fact that the data is not normalized (i.e. some LLDM will range from 0-100 percent whereas other LLDM are represented in continuums) and the fact that the graphical representation of the data, albeit interesting, is not very easy to interpret. The subsequent experiment will aim to address these issues.

4.6 Extension of Bautista's performance measurement model

In this section, the limitations described in section 4.5.5 will be addressed with the implementation of automated methods of data extraction, normalization and anomaly detection, as well as prediction as earlier proposed in the research problem and definition. This experiment includes interactions with end users and uses their feedback to train the initial performance problem predicting mechanism. The involved cloud computing environment is the same as that which is presented in Figure 2.9.

4.6.1 Setup

In order to be able to use the Big Data technologies chosen for this research, all components already discussed and presented in the previous sections were configured to generate logs with the extension ".csv" in a centralized HDFS configured to be the main data collector. This served as a central repository for the distributed data.

As was described in chapter 3, the scope of the experiment must include the ability to collect data from a real life application. This can be performed by leveraging the OOZIE Coordinator which implements workflow execution on multiple triggers. Algorithm 4 demonstrates, in pseudocode, the high level configuration of the OOZIE Coordinator for this task.

Algorithm 4.4-Oozie coordinator Algorithm

```
For (1){
        If (file exists, extension=csv) Run (http://url:9000/perfman/organization);
        }
```

This configuration will invoke the organization algorithm when a new file with a .csv extension is added on /perfman/. The workflow of the processes is shown in Figure 4.4 and the relevant algorithms are discussed in sections 4.6.3 and 4.6.5.

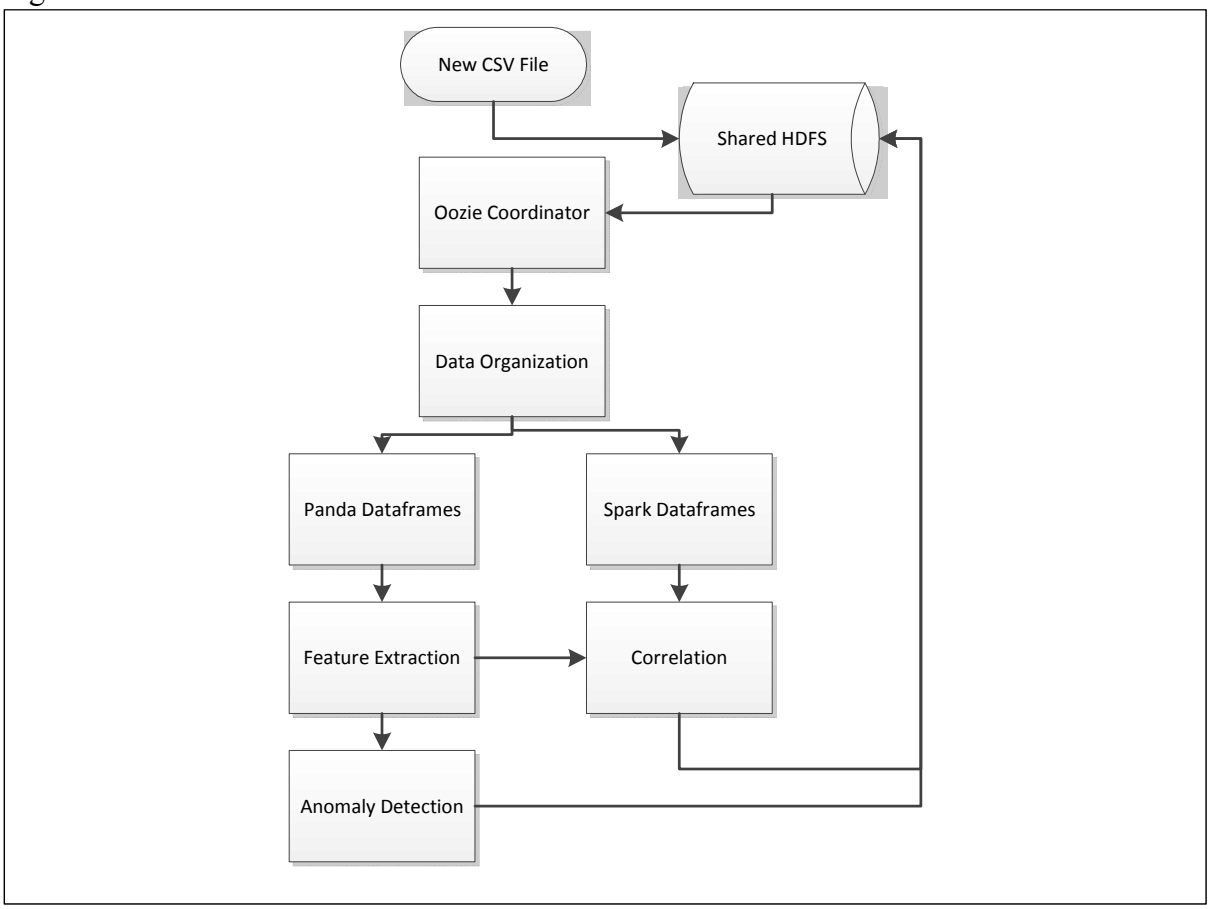

Figure 4.4 - Experiment components and relationships

Figure 4.4 describes the components and relationships of the experiment as a sequence of executions of the algorithms to produce the desired outcomes. It also shows that not all operations could be performed using the Spark dataframes and had to be done using Pandas dataframes because of the limitations of Spark when calculating the kurtosis.

4.6.2 Data preparation

The approach chosen for processing very large amounts of data in real-time (see section 4.1) is through the use of Apache Spark, a recent Big Data technology (Apache foundation, 2017) which is both faster than Hadoop and simpler to learn and program. Spark supports Application Programming Interfaces (API) for Scala, Java and Python. This algorithmic approach allows for processing a very large amount of log data coming from multiple CCS components, within decision time, and includes the ability to analyze anomalies and describe potential sources of problems as early as possible.

Three data preparation activities were carried out:

1) Data organization: the files are loaded in HDFS and shared by the individual components as they are created. When a new file is detected, the workflow coordinator invokes the organization algorithm, which scans for new files and loads them in memory. This begins the data cleanup step.

2) Data cleanup: two main processes occur to a) convert the results to percentiles, so that all measures can be plotted in the same space, and b) convert qualitative measures characterized as H in to Annex II, so that all measures conform to the "lowest is better" qualitative approach. Some dataframes might contain null values; these must also be cleaned up. Once completed, the dataframe segregation algorithm is invoked.

3) Dataframe segregation: the .csv data is converted into either a Pandas or a Spark dataframe. Panda's dataframes are used for the calculation of kurtosis, variance and anomaly detection, and the Spark dataframes are used for vertical correlation. The dataframes do not necessarily have the same format (i.e. the same number of rows and columns) as it depends on the ability of a particular component to log information. This activity returns the Pandas and Spark dataframes to the correlator function for the feature extraction which is described next.

4.6.3 Feature Extraction

The feature extraction activity, represented in Figure 4.4, is necessary in order to provide the 30 most significant features that are extracted via the kurtosis and the variance analysis of each performance data file, identifying the elements in which the highest peaks and variance are located, for the data center analyst to consider.

The original research and experimentation conducted by Bautista uses a combination of particular statistical methods in order to obtain a list of the most relevant measures to be analyzed. In this research, a different approach using the time series analysis where the datasets of values are considered not as a matrix of self-contained values but as an interactive result that considers previous values is implemented. With this type of data, the selection of random samples from the data contained in the matrix of performance log values only makes sense if random times containing all the types of data can be selected. In the experimental setup reported in section 3, the performance log file is considered as potentially endless, with each of the lines of the file reflecting a new interaction of the system.

In order to analyze this potentially endless data, each value is considered as a string of multiplexed values across the time that relates to the same measure, through the time variable, and across the multiple variables that were collected in the same moment, through correlation and covariance.

When this data is represented in a matrix format–such as a spreadsheet, for example–the flow of time represents additional lines on the matrix or the spreadsheet. The LLDMs are the columns in the same representation. That means that a single measure (a column) can correlate with itself throughout the time (vertical perspective) or a single value (a cell in the spreadsheet) correlates to the values at the same time (all other cells on the same line).

An example of such interaction would be “\Paging File(*)\% Usage”. This performance log measure describes how much of the page file is being used by the local operational system. In

practice, this means that a) there is a need for paging (e.g., the main memory is exhausted) and b) the act of reading this file is causing I/O reads and writes on the local disk. Ultimately, the higher the utilization, the more indication will be given that there is a significant time degradation on the task. This time degradation is caused by disk I/O that is slower than memory I/O.

As the counter collects data, the time series will show a "history", its values representing different relevancies as time progresses. For example, a sequence of values such as 5, 6, 50, 100 would suggest that the performance is degrading, whereas the same values, in the reverse order, 100, 50, 6, 5 would represent an improvement in the time behavior, as simulated in figure 4.5. Intrinsically, each of these values would impact the measures collected in the same sampling according to its values. The "100" value for this measure should also contain more disk I/O read/write as long as the values of multiple features had been sampled on the time interval

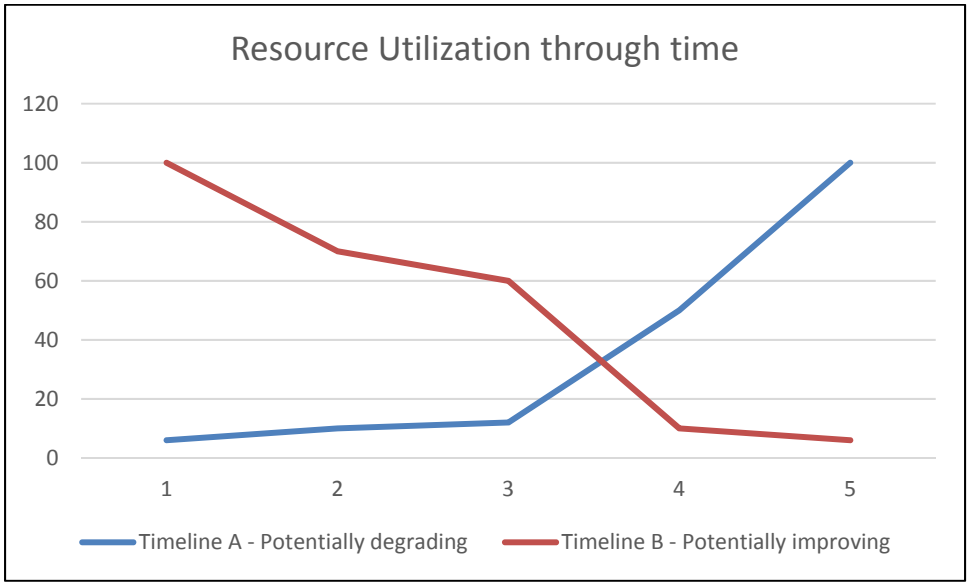

Figure 4.5 - Performance degradation versus improvement through time

In the end user performance perspective theory described by Davis (Davis F. D., 1989), software-based systems have to help the end users to perform the required tasks in a timely fashion. We consider that the features that have a linear correlation and those with more symmetry and reduced variance would induce the end user to expect a standardized response.

Consequently, disturbances to the end user performance perspective can be identified by the peaks or asymmetries in the resulting time series.

Considering these effects, it is possible, using the end user performance perspective theory and Independent Component Analysis (ICA) (Hyvärinen & Oja, 2000), to justify choosing both kurtosis and variance as initial approaches for feature selection. The symmetry and variability fit the description of ICA: in neural networks, amongst many other disciplines, the identification of the multivariate data requires a large amount of calculations. ICA has been recently developed as a way to represent non-Gaussian data in terms of independent, or as independent as possible, components. The variance and kurtosis analysis is a preliminary approach for this technique and leaves room for further improvements of the initial technique in further research.

This process reduces the dataframes to a more manageable size with a fixed width of the 30 most relevant columns. These reduced dataframes are part of the desired outcomes as they contain the most variant performance log measures which could potentially represent the measures that better explain any degraded performance and are passed into the correlator function described in section 4.5. Algorithm 5 contains the pseudocode for the features extraction.

Algorithm 4.5-Feature extraction via Variance and Kurtosis analysis.

```
for column in df:
        temp = (df[column].copy()
        mean = temp.mean()
        if mean is not None and mean!= 0:
                kurt = stats.kurtosis(temp)
                results[column] = [kurt, temp.var()/mean]
        Transpose results by Kurtosis, Variance/Mean
```

The initial approach for the 30 most relevant performance log measures follows the findings of previous research whereas 38 other features explained most of the variance of performance log data samples (Esmael, Arnaout, Fruhwirth, & Thonhauser, 2015). In this particular experiment, the 30 features mentioned above contained at least 98% of the cumulative variance of the dataset. This algorithm is dependent on the Pandas dataframes because the kurtosis and variances formulae are not implemented on Spark as of the writing of this paper. It also means that this particular code cannot be distributed to the child components, which reduces the processing turnaround.

4.6.4 Correlation analysis

The correlation analysis activity of the experiment allows the data center analyst to identify candidate relationships between different performance measures across the CCS components or internally within a particular component. Although not inferring causality, it sheds light on similarity, patterns and equivalences between the values of the performance log measures. The correlation analysis of the most relevant performance measures is executed in two ways, intra-component and trans-component:

1) Intra-component performance data correlation: On a performance log with the .csv format, the performance measures are each displayed in a column. When correlating performance measures on the same time-frame, these performance measures are located side-by-side, referred to here as an intra-component correlation.

2) Trans-component performance data correlation: This activity is comparable to the intra-component correlation, however, in this case, the columns with the same name are correlated across all the components that contain that particular performance log measure. Again, without inferring causality, it is possible to recognize the similarity of loads, resource consumption and even widespread issues which affect more than one component. The computations necessary in order to calculate the trans-component correlations can be distributed to the Spark cluster

and are significantly faster when running in multi-component tenancy than in single component.

Intra-component correlations clarify the relationships between performance measures within the same CCS component, thus helping to identify issues which affect a single component.

Trans-component correlations assist in investigating issues that involve the same performance measure that affects multiple components.

4.6.5 Anomaly detection

The anomaly detection implementation has the potential to provide useful information with regards to solutions to performance and resource utilization incidents. The anomalies are calculated using the Holt-Winters (second order) algorithm, which considers that the most recent values have a larger weight on the current measures than more distant ones (i.e. older events): empirical observations of data center analysts often assume that a recent fluctuation in processor utilization would have a greater impact on end user perceived performance than a time distant one.

Considering that one of the concerns of this research is how the end user perceived performance of CCA could be modeled in a way that timely analysis of the data can be enacted upon, it is important to describe an approach by which "timely analysis" can be done. The objective is to provide, as early as possible, some kind of information to a data center analyst that would help identify issues potentially affecting the end user performance perspective.

One of the key differences between this research and that of Bautista, Alan and April (2012) is the analysis of the view of performance data as a time series: a time series has some well-known properties such as seasonality, momentum, interval and unicity of the data points (Castor, 2006). The idea of interpreting the data as a time series also allows for the possibility of forecasting, cycling and trending.

The concept invoked here, however, is that of the anomaly detection. An anomaly is any feature value that is positioned outside of an expected model. In this research, the Holt-Winters technique of exponential smoothing (Weinman, 2009) is used in the code in Algorithm 2 presented in section 4.6. This statistical method was selected because it is one of the simplest forms of exponential smoothing and thus serves as a starting point for approaching the problem of anomaly detection.

The algorithms for Holt-Winters exponential smoothing are available in multiple languages such as R and Python. This algorithm in particular has 2 important drawbacks. First, it tends to produce lagging results, meaning that burst-type variations take at least one full cycle interaction to compute. Additionally, seasonal data is hard to take into account unless the moving average windows can be fixed around the seasonality. This leads to a poor prediction capability for this particular method, as exposed in the conclusions and is an opportunity for further research.

This implementation uses an anomaly ratio of 25% above the predicted level, as well as a time span of 100 observations. As for any autoregressive method, the longer the data is collected, the greater its forecasting precision will be (ISO, 1994). Algorithm 6 presents the pseudocode.

Algorithm 4.6 - Anomaly Detection employing Holt-Winters second-order algorithm

```
Calculate ratio_series()
        alpha = 2.0 / (1 + span)
        s = np.zeros((N, ))
        b = np.zeros((N, ))
        s[0] = arr[0]
        for i in range(1, N):
                s[i] = alpha * arr[i] + (1 - alpha) * (s[i - 1] + b[i - 1])
                b[i] = beta * (s[i] - s[i-1]) + (1 - beta) * b[i-1]
        return s
        hw = _holt_winters_second_order_ewma(series_clean, 2, 0.5)
        ratio_series = series_clean.apply(_lambda_get_ratio, args=(hw, cpt))
return ratio_series[ratio_series > _ANOMALY_MAX_RATIO]
```

This algorithm reads the list of performance logs and uses the exponential smoothing technique to identify the candidate anomalies, marking that particular feature in time with an X. This could help the data center analyst in identifying potential sources for issues in the time series that need to be focused upon.

4.6.6 Application of the model

In this activity, the measures emerging from the extracted features are mapped back to the concepts described in Table 8. This mapping helps with interpreting the results that follow a textual presentation such as:

Anomaly detected (Time X, Measurement Name Y, Performance concept Z)

The Bautista performance model includes a set of formulae for representing the performance concepts presented in Table 4.5 and the complete list in Annex 2. Table 4.7 provides an excerpt of all the adapted formulae originating from the extensive list of the same Annex. In the next section, these formulae are discussed in more detail. Since this research focuses on the performance measures obtained from different components, and considering that the collected measures are all quantitative values, the resulting expressions are qualitative evaluations of the values of these measures on an ordinal scale. In Table 4.7, L means "Lowest is better" – i.e. a value closer to zero represents a better perceived performance to the end user, whereas H means that the higher the value, the better the performance perceived by the end user.

Table 4.7 - Excerpt of collected measures and qualitative evaluations

Collected Measures	**Evaluation scale**
\LogicalDisk(*)\Free Megabytes	H
\Netlogon(*)\Average Semaphore Hold Time	L
\Memory\Page Faults/sec	L
\Memory\Available Bytes	H
\Memory\Pages/sec	L
\Paging File(*)\% Usage	L
\System\File Read Bytes/sec	L
\System\File Write Bytes/sec	L
\System\System Up Time	H
\System\Processor Queue Length	L

Only 8 performance log measures were evaluated as within the H scale category, whereas 49 others were evaluated within the L category. This experiment had no control on this characteristic of the log measures, given that they are created by the developers of the operational systems and applications, and was only used for the research as provided. A search for a greater balance between the 2 categories could be explored by using the available log tools that use such counters (The Cacti Group, 2017) (Microsoft, 2013) (Munin and colaborators, 2017) (Sandeep, Swapna, Niranjan, Susarla, & Nandi, 2008) (Omniti Labs, 2014) (Zenoss, 2013). In order to simplify the calculation process, a conversion was made whenever

a measure fit the H description so that after being transformed into a percentage, these 8 measures were then rewritten based on a 100-value and sorted so that “the lowest, the better”.

4.6.7 Discussion

This first experiment processed a large amount of data: 2.4 GB, with approximately 500,000 rows and 483,000,000 data points. Processing the full dataset for feature extraction, anomaly detection and correlation required 7 hours on the processing infrastructure utilized.

In order to verify if the 7 hours for the processing of the total data points was a constant, we attempted the segmentation of data into 5 chunks of ~300MB as represented in Figure 4.6, reporting the time that it took to process the data in each case. The base dataset had a size of 983 MB and took 16 minutes to process. Each additional dataset contained the previous ones. The first segment had 983 MB and a processing duration of 16 minutes, whereas the 3rd segment had 1500MB and took 100 minutes to be processed using the algorithms 1 to 6.

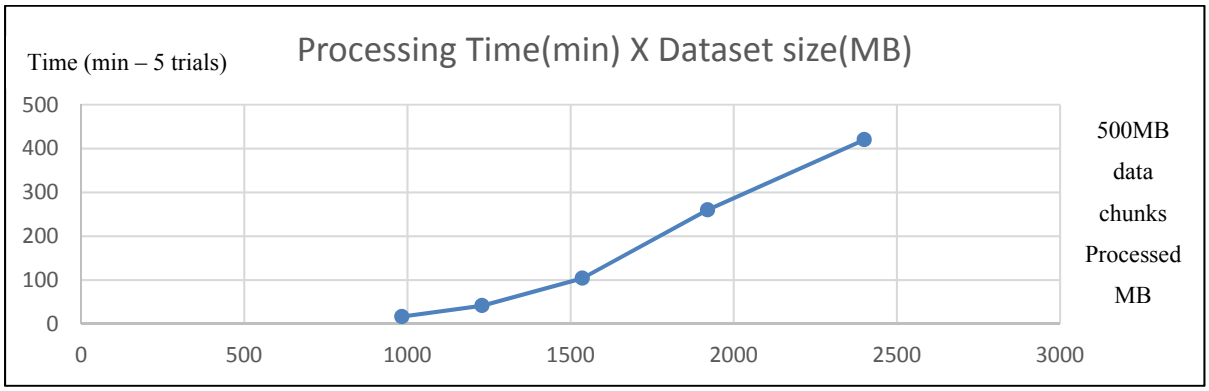

Figure 4.6 - Non-linear processing lengths, 5 trials, 500MB Chunks

Figure 4.6 shows that in this experiment, the relationship between time and data size is not linear. This could be caused by the large number of possible data relations and data size and could be further explored to determine an optimum ratio of processing time, size and time series accuracy. This data was processed using the algorithm in section 4.4 which led to the extraction of the most relevant features. Table 4.8 represents an excerpt of the extracted components and it contains the most frequently extracted features as well as the frequency of their representations.

Table 4.8 - Most Frequently Extracted Features

Extracted Feature Name	**Frequency**
\Process(WmiPrvSE#3)\% User Time	0.41
\Process(EACommunicatorSrv)\IO Read Operations/sec	0.35
\Process(CirratoClient)\% User Time	0.32
\Process(svchost#1)\IO Read Operations/sec	0.32
\Process(sftvsa)\% Privileged Time	0.32
\Process(svchost#3)\IO Read Operations/sec	0.32
\Process(nvvsvc)\% Privileged Time	0.29
\Process(CirratoClient)\% Processor Time	0.29
\Process(csrss)\IO Read Operations/sec	0.29
\Process(concentr)\% Processor Time	0.24
\Process(wfcrun32)\% Processor Time	0.24
\Process(EACommunicatorSrv)\% Privileged Time	0.24
\Process(CentralizedLogPusher)\% User Time	0.24
\Process(CUI)\% Privileged Time	0.24
\Process(wdp)\% Privileged Time	0.24
\Process(AERTSr64)\% Privileged Time	0.21
\Process(concentr)\% Privileged Time	0.21
\Process(CirratoClient)\% Privileged Time	0.21
\Process(wfcrun32)\% User Time	0.21
\Process(winlogon)\% User Time	0.18
\Process(spoolsv)\% Privileged Time	0.18
\Process(idarchive)\IO Read Operations/sec	0.18
\Process(taskhost)\IO Read Operations/sec	0.18
\Process(SCNotification)\% Privileged Time	0.18
\Process(wfcrun32)\% Privileged Time	0.18
\Process(svchost#11)\% Privileged Time	0.18
\Process(svchost#10)\% Privileged Time	0.18
\Process(svchost#8)\IO Read Operations/sec	0.18
\Process(CentralizedLogPusher)\% Privileged Time	0.18
\Process(sftlist)\IO Write Operations/sec	0.18

The measures represented in Table 4.9 are good candidates for an initial investigation of the most frequent sources for end user performance perspective degradation events.

The extracted features have been intra- and trans-correlated, as described in section 4.6. The intra-component correlation of performance measures of one component is presented in Table 4.9. The list of the trans-component correlations between different machines for the same performance measure is presented in Table 4.10.

Table 4.9 - The intra-component correlation of performance measures of one component

	(CirratoClient)\% Processor Time	(CentralizedLogPusher)\%Processor Time	(pnamain)\% Processor Time	(pnamain)\% User Time	(nvvsvc)\IO Read Operations/sec	(WmiPrvSE#3)\% User Time	(SCNotification)\% Processor Time	(AcroRd32)\% Privileged Time	(nvxdsync)\% Privileged Time
CirratoClient)\% Privileged Time	1								
(CentralizedLogPusher)\% User Time		1							
(pnamain)\% User Time			1						
(nvvsvc)\IO Read Operations/sec			1	1					
(SCNotification)\% Processor Time						0.9			
(SCNotification)\% User Time						0.9	1		
(BESClientUI)\% User Time			0.89	0.89	0.89				
(nvxdsync)\% Processor Time									1
(AcroRd32)\% Processor Time								0.95	

The data in Table 14 allows us to identify, for example, that BESClientUI has an 89% correlation ratio to both (pnamain)\% Processor Time, (pnamain)\% User Time and (nvvsvc)\IO Read Operations/sec. This could indicate that whenever troubleshooting issues involve one of these particular processes, the others could potentially be likewise affected.

Table 4.10 - Trans-component correlation ratios, (svchost#1)\IO Read Operations/sec

(svchost#1)\IO Read Operations/sec	node0	node1	node2	node10	node3	node4	node11	node5
node1	1							
node2	1	1						
node3	1	1	1					
node4				1				
node6	1	1	1		1			
node7				.99		.99		
node5							1	
node8							1	1

Table 4.10 shows the trans-component correlations for one particular performance measure. We can see that for the (svchost#1)\IO Read Operations/sec performance measure, there is a high correlation level between components 0,1,2,3 and 6. This means that if an issue was to be found in one of these CCS components which impacted the selected measure, it would be appropriate to also investigate the correlated CCS components for similarities.

Table 4.11 contains an index of all the measures that have been correlated as trans-component for this experiment. Some of them present a large number of correlations, while others correlate only to a single component. When investigating systemic issues or when evaluating distinct end users and locations, this table could be used to guide the identification of possible similarities in otherwise segregated scenarios, aiding with the investigation of performance degradation events.

Table 4.11 - Trans-component correlated performance measures

\Process(CirratoClient)\% Processor Time
\Process(idarchive)\IO Write Operations/sec
\Process(svchost#1)\IO Read Operations/sec
\Process(encsvc)\% Privileged Time
\Process(svchost#5)\IO Write Operations/sec
\Process(csrss)\IO Read Operations/sec
\Process(CirratoClient)\% Privileged Time
\Process(svchost#3)\IO Read Operations/sec
\Process(idarchive)\IO Read Operations/sec
\Process(WmiPrvSE#3)\% User Time

The anomaly detection algorithm was able to detect different occurrences of anomalies when an anomaly is defined as a difference of 25% of the value observed compared with the predicted values obtained using the Holt-Winters algorithm. Table 4.12 presents a list of the measures which displayed anomalies in 3 different CCS components. The anomaly detection, as well as the time stamp of each individual event, can be mapped back to the end user reports of degraded performance to determine the possible causes of the degraded performance events. These can be candidate Root Causes (RC) for the degradation events.

Table 4.12 - Anomaly sources – 3 machine sample.

1\Security Per-Process Statistics(1168)\Context Handles
1\Process(svchost#5)\IO Write Operations/sec
1\Process(chrome#7)\IO Read Operations/sec
1\Process(chrome#8)\IO Write Operations/sec
2\Process(chrome)\IO Read Operations/sec
2\Process(TrustedInstaller)\IO Write Operations/sec
3\Security Per-Process Statistics(3428)\Context Handles
3\Security Per-Process Statistics(3464)\Credential Handles
3\Security Per-Process Statistics(8024)\Context Handles
3\Process(iexplore)\% User Time
3\Process(iexplore)\% Processor Time
3\Process(iexplore)\% Privileged Time
3\Process(System)\IO Read Operations/sec
3\Process(iexplore#2)\% Processor Time
3\Process(iexplore#2)\% Privileged Time
3\Process(iexplore#2)\% User Time

For example, from Tables 4.10 and 4.9 it can be observed that the measure \Process(Svchost) is a possible candidate for an RC on performance degradation events: it is seen as trans-correlated (i.e. affecting multiple CCS components) as well as intra-correlated (for a particular component). It would be possible to narrow the investigation to this particular measure and, upon solving this issue, potentially improving the end user perceived performance.

The application of Bautista's model involves the manual association of the measures with the performance concepts proposed in the ISO 25010 standard which, in this experiment, has produced the following distribution:

- "capacity": 10 measures
- "availability": 5 measures
- "time behavior": 27 measures

- "fault tolerance": 9 measures
- "resource utilization": 780 measures
- "maturity": 58 measures

An imbalance between the quantity of measures exists. This is because the resource utilization measures are multiplied by the number of processes running on a CCS component which, at a macro level, represents more resources as being consumed by a particular application. Given the imbalances in the quantity of components for each concept, it was necessary to create a form of aggregator – an indicator – for each performance concept. In order to construct such an indicator, the following process was selected:

- All the values were initially converted to percentages so that they can be represented on the same scale. For values with a quality evaluation of "H" in Table 12 (Annex 2), the value was further converted into a "100-value" so that the lower the number, the better would be the expected perceived end user performance.

- After normalizing the values to percentages, all the values were plotted on a virtual plane. These points were distributed concentrically over a radial graph, so that the distances between each point and the 0 central point were equivalent to their values (point 10, for example, is 5 units closer to the center than point 15).

Algorithm 4.7 - Circumscribed polygon of N sides area calculation, Python

```
count = len(number_of_columns)
        for i in range(count - 1):
                result += arr[i] * arr[i+1]
                result += arr[0] * arr[count-1]
        result *= 0.5
        rad = ((360.0 / (count *1.0)) * math.pi) / 180.0
 return result * math.sin(rad)
```

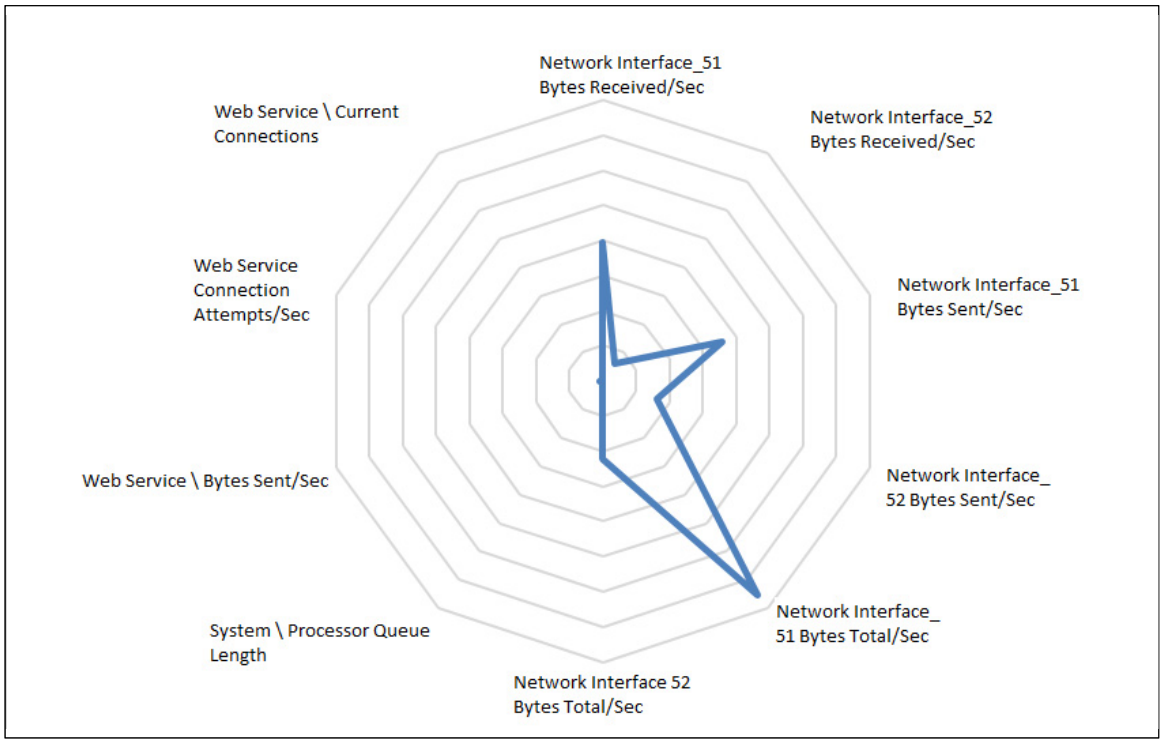

Figure 4.7 - One point, multiple time behavior measures displayed on a virtual plane

Using the formula in algorithm 7 to calculate the area of a circumscribed polygon of N sides, implemented in the Python programming language, the value for the area of each point, as shown in Figure 4.7, was calculated. The values for the calculated areas were then represented across the time for the experiment, representing different levels of capacity, availability, time behavior, fault tolerance, resource utilization and maturity.

The area of this polygon represents the utilization of a given resource. The utilization is represented by a scale with 0 at the center (which means the resource is available) and 100% (at the exterior boundary which means the resource is busy). The figure shows the calculated area of a N-sided polygon. When it is smallest, it means that less resources are used which in turn potentially represents a better time behavior.

Multiple areas can be represented in the form of a single time series that respects the quality evaluation of either L or H where appropriate. Essentially, this new indicator would be able to demonstrate, across a timeline, the events with the highest potential of causing a degraded performance as perceived by the end user.

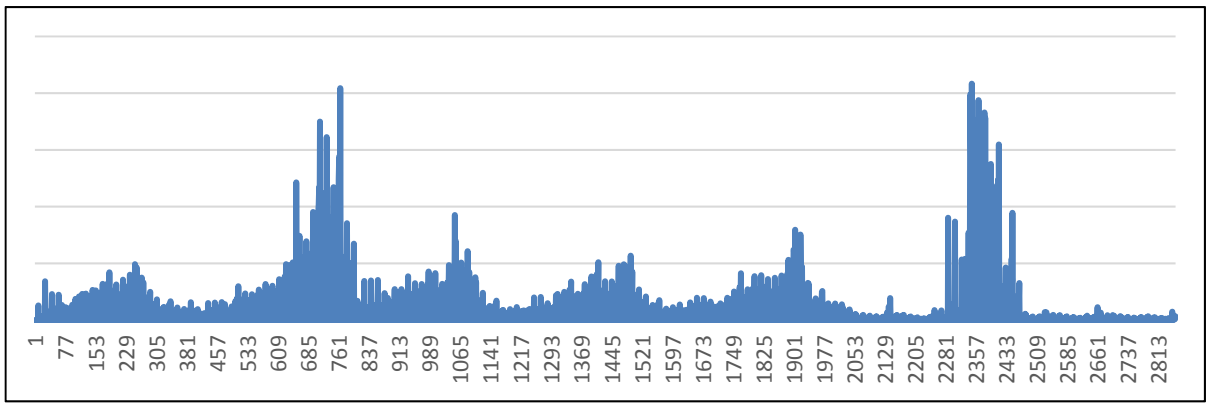

Figure 4.8 - Time behavior representing peaks in occurrence 765 and 2343

Figure 4.8 shows particular points in time where the time behavior concept had high values. This could indicate times where a particular combination of measures presented a degraded behavior as well as displaying points where intervention would be more effective. Additionally, the monitoring of these indicators could be part of the development of a service level agreement for CCA based on the end user perceived indicators as calculated in this experiment.

4.6.8 End user feedback and anomaly forecasting

The implementation of end user feedback is part of the core measurement process demonstrated in Figure 2.5 - ISO/IEC 15939:2007 - Measurement process. In this experiment, we implement the end user feedback with 2 mechanisms: interactive and voluntary. These feedback mechanisms will aid with training the prediction algorithms. The implementation of collecting feedback has two main motivations: first, it expands Bautista's model, improving its completeness in regards to ISO/IEC 15939:2007; second, the end user performance perspective theory described in section 2.3.1 defines the performance of a system for the end user as its ability to consume resources in order to complete tasks. When measuring resource consumption, it is assumed that these will relate directly to end user requested or desired tasks. The end user feedback helps with addressing the assumption that if an anomaly is detected and is corroborated by end user feedback, then it is the convergence of both the statistical method and the user's perception. As stated in section 3.3, it is important to re-visit whether the

research takes into consideration one of the disadvantages of cloud computing technology, i.e., unreliability of system performance due to the complexity of the infrastructure. The end user feedback will also be used to provide information to reduce unreliability, as described next. The mechanisms and statistic data for the end user feedback are presented in annex III.

4.6.8.1 Voluntary end user feedback

The voluntary end user feedback mechanism is implemented in a simple way in order to allow individuals to freely provide examples of degraded performance. It is a simple one-click-action that gathers relevant data in order to build an "anomaly case". Algorithm 8 provides the pseudocode for collecting the data and storing it for the machine learning algorithm.

Collected data:

- Desktop name.
- Destination mail box.
- Time.
- Network configuration.

Algorithm 4.8 - Voluntary end user-feedback

```
Action(on click)
        Gather(hostname, opened mailbox, local machine time, time zone)
        Array(network trace, arp table, MAC address, from hostname to server)
        Store(hostname, mailbox, local machine time, time zone, array of network
configuration).on Shared HDFS as local machine time-hostname-v.euf
```

Algorithm 8 helps to address the disadvantage highlighted in section 2.2.3 by storing a qualitative metadata of the current network configuration as an array of component addresses. The uniqueness of the addresses is achieved by storing of the local media access control (MAC) address, which is unique to each network enabled component as per EUI-48 (Lamber, 2001). This is useful in order to gather only the information for the components involved in this

particular performance degradation case so that not all network configuration changes, failovers and switching is captured. The data is stored on the same shared HDFS to be used by the interactive end user feedback (algorithm 9) and machine learning (algorithm 10). The relative position of these algorithms in relation to the proposed solution's workflow is presented in Figure 4.10.

4.6.8.2 Interactive end user feedback

In order to implement the interactive end user feedback, the workflow presented in Figure 4.4 is slightly modified with the introduction of the feedback loop presented in Figure 4.9. The data collected is the same as described in section 4.5.7.1. Algorithm 9 describes the pseudocode which builds upon the anomaly detection algorithm presented in section 4.5.5 that enables the end user feedback loop:

Algorithm 4.9 - Interactive anomaly detection

```
On(Anomaly Detected)
        Gather(hostname)
        1 minute Timed Popup(Anomaly detected, confirm? Yes/No)
        If confirmed,
                Gather(hostname, opened mailbox, local machine time, time zone)
                Array(network trace, arp table, MAC address, from hostname to server)
                Store(hostname, mailbox, local machine time, time zone, array of network
                configuration).on Shared HDFS as local machine time-hostname-c.euf
        Else
                Gather(hostname, opened mailbox, local machine time, time zone)
                Array(network trace, arp table, MAC address, from hostname to server)
                Store(hostname, mailbox, local machine time, time zone, array of network
                configuration).on Shared HDFS as local machine time-hostname-i.euf
```

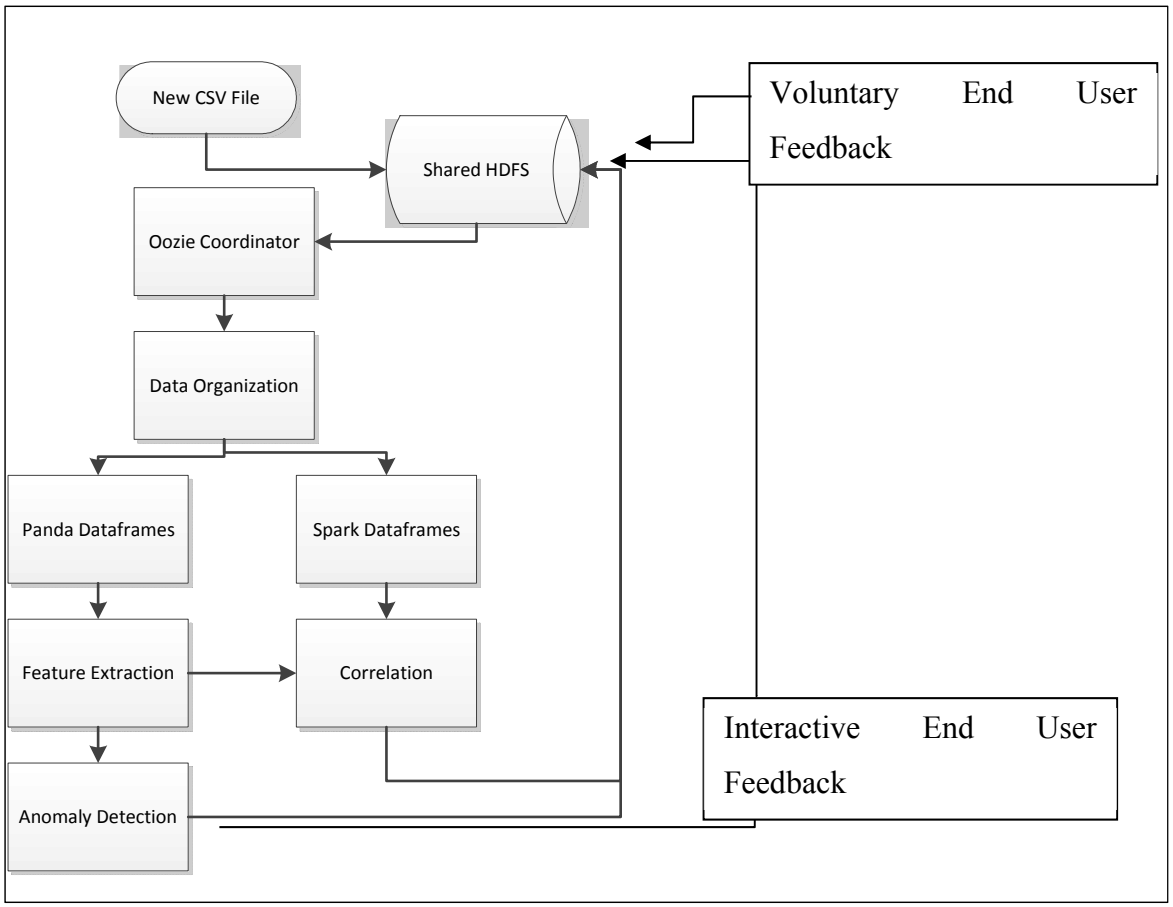

Figure 4.9 - End User feedback mechanism

Three feedback files are collected: Voluntary, Interactive and Confirmed. Confirmed feedback is that which is both a calculated anomaly and voluntarily provided by the end user. Voluntary and interactive are either simply reported by the end user or automatically calculated.

4.6.8.3 Anomaly forecasting

The anomaly forecasting capability is the final step of the experiment, aiming to predict the next occurrence of an anomaly based on the training sets provided by the end user feedback. Figure 4.10 presents the workflow for the automated anomaly forecasting and algorithm 10 contains the pseudocode.

The anomaly forecasting algorithm scans the shared HDFS for a new .EUF file using OOZIE coordinator as described in algorithm 4. This will initiate the data organization algorithm which

browses through the different types of .EUF files, the voluntary, interactive and confirmed feedback. The confirmed feedback (-c.euf files) is automatically loaded into the trainer with an arbitrary weight ratio of 3:1, whereas the voluntary and interactive feedback is loaded with a weight of 1:1 for the training.

From the data organization algorithm, the Spark Dataframe is initialized loading only the database rows that correspond to the unique identification of the components collected in the feedback algorithm, which will then be used with the training of the prediction mechanism. The full data is loaded into the Pandas dataframe and is used for prediction of the particular affected rows using the summary of each row calculated in the trainer in order to predict whether the value belongs to the training class or not.

The accuracy is calculated as a final step. An arbitrary value of 1 minute of data is used as each anomaly window, i.e. if an anomaly is calculated, a chunk of data with the size of 1 minute will be collected for the particular measures, which then is used on both Spark and Pandas dataframes.

We employ a Naïve Bayes algorithm as an initial approach to the analysis of this time series understanding that the chosen method is simple, albeit incomplete, and leaves room for further investigation. Also, Naïve Bayes features a good decoupling ability where not all data must be dimensionally symmetric, reducing the size of the calculated dataset (Russel & Norvig, 2003).

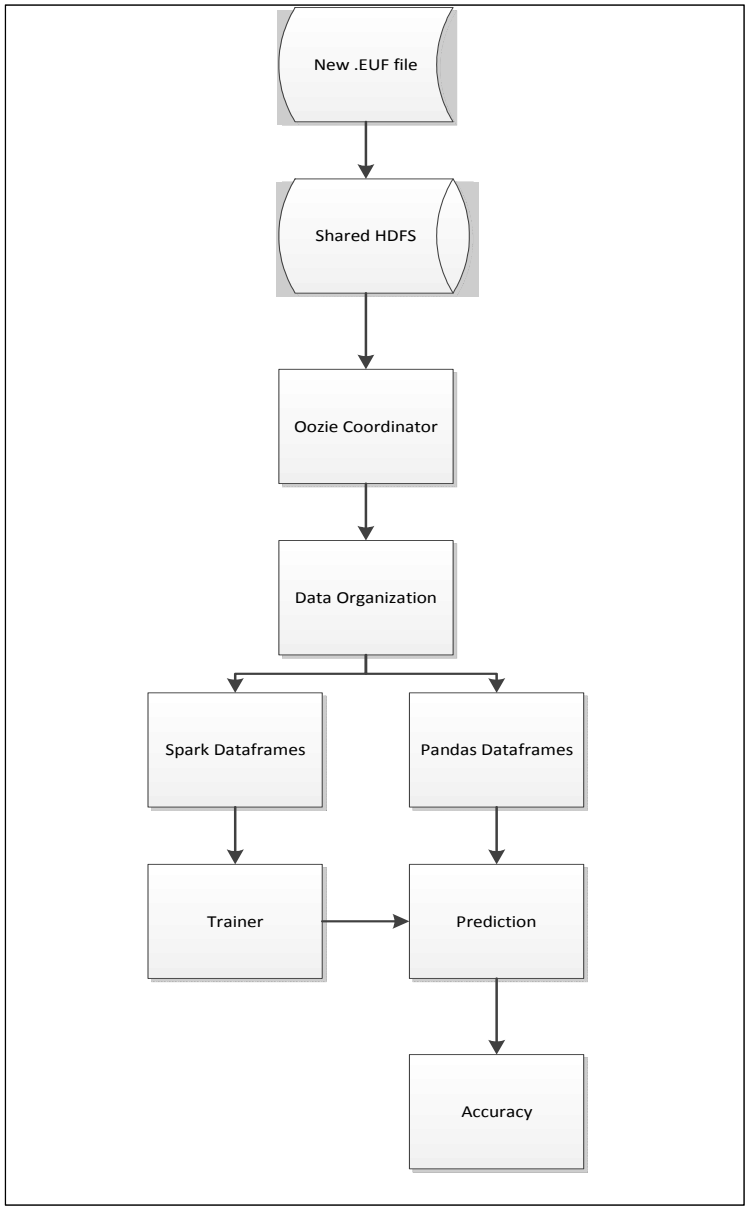

Figure 4.10 - Anomaly forecasting workflow

The accuracy is tested in two different scenarios. First, a group of confirmed anomalies is detected. Then, three training sets are created: confirmed anomalies training set, interactive anomalies training set and full dataset training set. As an independent confirmed anomaly is located, the values encountered for this anomaly are tested as per the summarization algorithm which will aim to locate whether or not the value belongs to the anomalies class.

Algorithm 4.10 - Anomaly forecasting

```
//confirmed summaries – trainer set
For each *-c.euf on HDFS
Load file as CSV
        For each header on CSV
                For each line on *.csv between time-1 and time
                        Load header from HDFS *.CSV as a new dataframe
                Calculate Mean, Stdev(header) as confirmedSummary(header)
//unconfirmed summaries - trainer
For each *-[I,V].euf on HDFS
Load file as CSV
        For each header on CSV
                For each line on *.csv between time-1 and time
                        Load header from HDFS *.CSV as a new dataframe
                Calculate Mean, Stdev(header) as unconfirmedSummary(header)
//confirmed summaries have a higher weight than unconfirmed ones
Weightedsummary=(confirmedsummary(all)*3+unconfirmedsummary(all))/4
//predict if a value belongs to a particular training set.
Tail *-c.euf on HDFS
Load file as CSV
        For each header on CSV
                probConfirmed= (1/(sqrt(2*PI) * confirmedsummary(stdev) * exp(-1(header-
confirmedsummary(mean)/2)/2*pow(confirmedsummary(stdev)/2)
        For each header on CSV
                probUnConfirmed= (1/(sqrt(2*PI) * unconfirmedsummary(stdev) * exp(-
1(header-unconfirmedsummary(mean)/2)/2*pow(unconfirmedsummary(stdev)/2)
```

Algorithm 10 will return the probability of any given value to be either a confirmed anomaly, an unconfirmed one or not an anomaly. This is further discussed in section 4.6.8.4.

4.6.8.4 Analysis

Three main points can be identified from the end user feedback mechanisms and anomaly forecasting experiment: the number of confirmed and voluntary anomalies in relation to the number of unconfirmed anomalies, the processing time for prediction of anomalies and the accuracy of the prediction. The total data size of detected anomalies for the dataset is approximately 2.4 GB of data with 500,000 rows and 483,000,000 data points. The number of detected anomalies is 14,445 with 6,972 confirmed anomalies. Another 1,911 anomalies were marked voluntarily by the end users. (Annex III)

Using the last confirmed anomaly as starting point and back tracking on the time series, the accumulated accuracy of the unconfirmed anomalies was 51% and 72% for the confirmed anomalies. As the end user feedback considerably reduced the datasets analyzed, the processing time for classifying the anomalies was ~4 seconds per occurrence, totaling 48 minutes of processing time.

Regarding the importance of the application delivery chain (Baer, 2011), by storing the anomalies along with the array of MAC addresses of the components involved in the particular detected anomaly, it is possible to describe the anomaly in a given moment T as an interaction of the resource utilization level of the most variant and highest-kurtosis measures of that particular chain. This means that the same utilization level on other components does not necessarily incur another anomaly. This finding ties the measured values with the end user feedback mechanism, creating a new form of data composed of the performance indicator (Figures 4.7 and 4.8), the application delivery chain for each particular measurement and the end user feedback.

4.6.9 Experiment conclusion

This experiment demonstrated that for the case studied, it was possible to extract the most relevant performance measures for identifying performance degradation of CCA (Tables 4.8,

4.9, 4.10) using only performance logs. These measures were correlated intra-component and trans-component in order to expose possible causes of the degradation events. Anomaly calculation using the Holt-Winters algorithm helped to identify the most probable causes of the degradation events (Tables 4.10 and 4.1). These measures should represent the end user perspective of the performance degradation events given that they are associated with the relevant performance concepts.

The experimentation modified the original theoretical proposal with the utilization of time series analysis on the performance data in order to determine the performance of a CCA from the perspective of an end user. An indicator for each performance concept has been introduced using a formula for calculating the area for an N-sided circumscribed polygon.

With the help of the end user feedback, it was possible to train the machine learning algorithms which helped to increase the accuracy of the model from 51% to 72% while reducing the processing time from a few hours to a few minutes, which justifies the use of the feedback mechanisms.

The challenges identified by this experiment are listed below:

1) Theoretical Challenge: The association of performance log measures and performance concepts, inspired from the original work of (Bautista, Abran, & April, 2012) and based on the ISO/IEC 25000 standards, remains manual. It is therefore lengthy and prone to human error. An index would be useful for more easily associating performance log measures and the concepts.
2) Technological Challenge: Spark allows for the distribution of data to be computed on the slave components, aiming to increase performance. Unfortunately, Spark lacks support for the most interesting statistical methods employed in this experiment and, therefore, had to be used in tandem with Pandas, a mathematical package which is very popular amongst Python

developers. The processing time could be significantly reduced with a similar software tool that supports all the required functions.

3) Measure Design Challenge: The research provided evidence of an imbalance in the number of performance log measures associated with each concept. This could be because the discussion of properly designing measures has rarely been used by the industry (Abran, 2010) or perhaps because there is a greater concern with certain aspects of performance. A possible subsequent plan for this would be a discussion with software that highlights that when performance logging is created, it should include a balance of measure types.

4.7 Chapter conclusion

The experiment provided some interesting findings for this research. Section 4.1 showed that the kurtosis and skewedness correlate positively to end user complaints of performance degradation. In section 4.2, it was described that the challenge of manually mapping the performance measures to the ISO performance concepts remains a concern. Section 4.3 discussed that it was possible to validate, through manipulation of the system workload, the measures used to describe the system's ability to process different workloads. Section 4.4 presented the first laboratory experiment where the data was identified as non-normalized and hard to interpret in graphical format.

A second expanded laboratory experiment, described in section 4.5, proposed an indicator in order to represent the end user experience, addressing the concerns raised during the previous experiment, as well as described an automated anomaly detection mechanism. The final findings of the experiment resides in the proposal of a new category of data, or metadata, which is the configuration of the system at the moment of a detected anomaly, the performance indicator and the end user feedback. This metadata represents the end user performance perspective for cloud computing systems using data center logs from Big Data technology for a private cloud application. The algorithms proposed here can be used as a basis for further measurement models and methods.

CHAPTER 5

Proposition of a Model for End User Performance Perspective for Cloud Computing Systems Using Data Center Logs from Big Data Technology

In this chapter, we revisit the research objectives as well as the results of the experiments conducted, presenting a model for the end user perspective of cloud computing performance based on performance logs generated by cloud computing systems with the feedback provided by end users.

Bautista's framework applies the performance concepts of the ISO 25023 standard on a logical process to describe or predict issues that may affect the outcome of a request in a cloud computing application. Figure 5.1 recalls the framework.

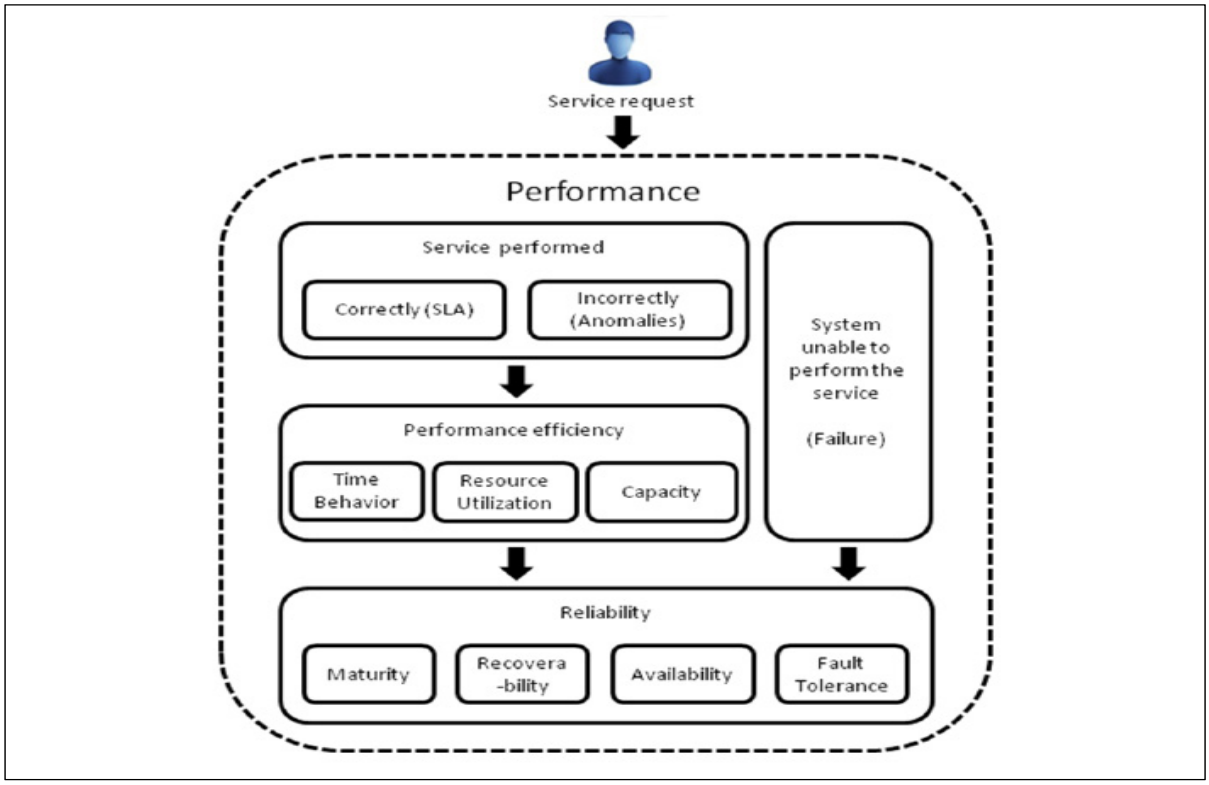

Figure 5.1 - Bautista's framework (Bautista, Abran, & April, 2012)

The research problem was defined as modeling end user experience on cloud computing environments with the proposition of a performance measurement model by only using data currently available from data center logs and gathering end user feedback as needed. The problem was then segmented into research questions: 1) what defines a cloud computing environment? 2) What influences end user performance perspective measurement in a cloud computing environment? 3) Are performance logs sufficient for modeling the end user performance perspective? If not, what other sources are required? 4) Can the performance measurement framework for cloud computing applications (Bautista, Abran, & April, 2012) be used for the creation of a performance model using data center logs that represents the end user performance perspective of an application that uses cloud computing technology in a timely fashion?

The definition of cloud computing that has been used during this research focused on the cloud computing technology, its deployment, its service models and its advantages and disadvantages. The main concept considered is the "unreliable system performance due to the complexity of the infrastructure". The proposal of a performance indicator for the time behavior of a cloud computing application, as presented in section 4.5, attempts to respond to this issue.

The research has been divided into sub-steps, specifically: 1 – association of end user performance with LLDM measures; 2 – mapping LLDM into the Performance Measurement Framework; 3 – Validation of the quality measures using a validation method (Jacquet & Abran, 1997); 4 – Laboratory experiment for end user performance perspective modeling; 5 – Expanded experiment; 6 – Creation of an automated mechanism for end user performance perspective modeling; 7 – Validation of the automated model; 8 – Proposition of the model. These steps have been addressed using 3 separate experiments, one initial experiment that solves the challenges described in section 1.4.3.1 to 1.4.3.4, a second experiment for steps 1.4.3.4 and 1.4.3.5 and a final experiment for the steps 1.4.3.6 to 1.4.3.8.

The first experiment identified that, in accordance with the end users report of degradation, performance measures have different levels of utilization in certain conditions. It is possible to see in experiment 4.1 that for 10 of the measures, the Time-0 value (moment of the degradation report) is significantly higher than the Time-1 and Time-2 values. Five measures do not display the same behavior, even though they display high variance and skewedness. The difference ratio between measures is also very disparate; some measures are 15 times larger at Time-0 than the other timely counterparts, whereas processor utilization, for example, is only 2 times higher. It would be possible to consider that, for this experiment, the values of the measures fluctuate as a symptom of performance degradation.

The association of the performance measures is a manual step that links the LLDM with the quality concepts of ISO/IEC 25010 (i.e. time behavior, resource utilization, capacity) and for reliability (i.e. maturity, availability, fault tolerance, recoverability). It is possible to identify some imbalances in the quantity of performance log data types associated with each concept, similar to what has already been reported by Bautista et al. This could lead to a discussion on how to effectively design a CCA, which is not within the scope of the research reported here. This is represented in Annex 2

With these measures identified, another step for validation was implemented with the construction of an automated mechanism for sending and receiving messages automatically, while manipulating the resource utilization of different performance elements of the system studied. The manipulation of the measures identified in section 4.1 increased the time of the job turnaround in all simulated events for the measures that are associated with the performance concepts of resource utilization, capacity and time behavior. Different measures reported different contributions to this increase, as described by Bautista. This effectively validates the measures as indicative of performance degradation.

The first laboratory experiment for end user performance perspective modeling was created by manually analyzing and representing data from the performance logs in order to provide a preliminary solution for the research problem. This initial experiment encountered issues such

as: the large volume of data; the fact that the data is not normalized, i.e., some LLDM will range from 0-100 percent whereas other LLDM are represented in continuums, and the fact that the graphical representation of the data, albeit interesting, is not very easy to interpret.

A final automated experiment, aimed at analyzing the collected data, identifying anomalies, gathering end user feedback and predicting further anomalies proposes the use of an indicator, based on the data collected from the data center logs, that describes the end user perceived performance. End user feedback has been employed to predict further anomalies with the indicator, as described in Figure 5.2, for the proposed model for end user performance perspective for cloud computing systems using data center logs from Big Data technology.

The proposed model is described by the performance indicator, the application delivery chain and the end user feedback for a given moment in time. In a real world scenario, the resource utilization (represented here by the indicator) or the end user feedback are not enough to describe the performance, especially on cloud computing infrastructures where the application delivery chain has the potential to change.

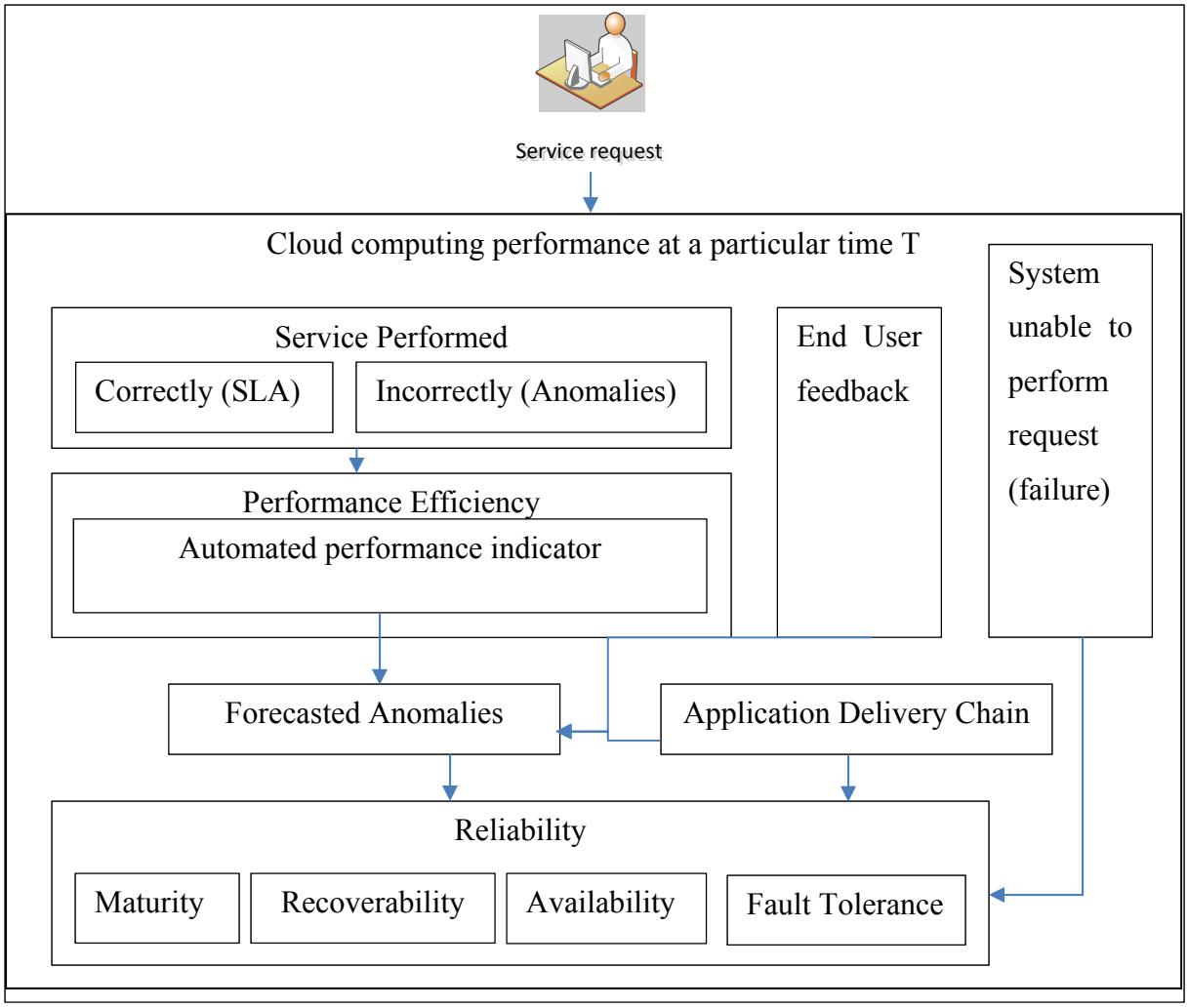

Figure 5.2 - Proposed model for end user performance perspective for cloud computing systems using data center logs from Big Data technology

CHAPTER 6

Conclusion

This chapter discusses the conclusion of this thesis that proposes a model for end user performance perspective for cloud computing systems using data center logs from Big Data technology. This model is based on the premises set out in Bautista's framework and upon practical experiments which improved the model in order to achieve practical results.

The literature review demonstrated the importance of measure validation, the challenge of collecting the data and the differences between business and software engineering perspectives on systems performance management. Cloud computing is a distributed computation model that has some disadvantages and one particular characteristic: unreliable performance which is based on infrastructure characteristics. This is the main focus of this research. Measuring end user performance, in a timely fashion in such a scenario, can possibly be achieved with the use of both IT service management processes and performance log data. Big Data technologies can also be employed to process and analyze the large volume of data produced by these data sources.

The laboratory experiment in section 4 provided some interesting findings for this research. Section 4.1 showed that the kurtosis and skewedness correlate positively to end user complaints of performance degradation. In section 4.2, it was described how the challenge of manually mapping the performance measures to the ISO performance concepts remains a concern. Section 4.3 discussed that it was possible to validate, through manipulation of the system workload, the measures used to describe the system's ability to process different workloads. Section 4.4 presented the first laboratory experiment where the data was identified as non-normalized and hard to interpret in graphical format.

A second expanded laboratory experiment, discussed in section 4.5, proposed an indicator in order to represent the end user experience, addressing the concerns raised during the previous

experiment, as well as described an automated anomaly detection mechanism. The final findings of the experiment reside in the proposal of a new category of data, or metadata, which is the configuration of the system at the moment of a detected anomaly, the performance indicator and the end user feedback. This metadata represents the end user performance perspective for cloud computing systems using data center logs from Big Data technology for a private cloud application. The algorithms proposed here can be used as a basis for further measurement models and methods.

The discussion of the particular research questions are as follows:

1) What defines a cloud computing environment?

   As seen in the literature review, cloud computing has a particular definition, and its advantages and disadvantages are clearly exposed. One particular disadvantage has been investigated in this research, the unreliability of the performance due to the large number of inter connected components that make up the cloud.

2) What influences end user performance perspective measurement in a cloud computing environment?

   Also from the literature, a number of factors have been identified as potentially affecting the end user performance, with greater effect being caused by the performance of the applications in contrast to the expectations of the individual users. This has been explored in section 4.2.

3) Are performance logs sufficient for modeling the end user performance perspective? If not, which other sources are required?

   From the laboratory experiment conducted (section 4.6), the addition of the end user feedback mechanism raised the accuracy of the predictions by 50%. With this result, it is possible to consider that end user feedback is a valid tool to include in the proposed model.

4) Can the performance measurement framework for cloud computing applications (Bautista, Abran, & April, 2012) be used for the creation of a performance perspective model using data center logs that represents the end user performance of an application that uses cloud computing technology in a timely fashion?

   With the revision of the studied framework and its application in real use cases, a few issues were discovered (timeliness, accuracy). The model was then improved with the addition of automated, algorithmic approaches to log analysis and parsing, as well as the proposition of the end user feedback mechanism for initial approaches to prediction of performance anomalies.

Future work will be necessary for improving the anomaly detection, prediction mechanisms, as well as a possible machine learning approach for identifying the conditions that will generate performance impact in the future.

ANNEX I

RESEARCH CONTRIBUTIONS

First, a preliminary research paper was accepted by the IEEE-ISWM Mensura conference, 2014, where we were able to initially apply Bautista's performance measurement framework on test data taken from data center logs. With this first laboratory experiment, we were able to measure the time behavior of production servers during a specific time frame. These early findings suggested that the sub-steps presented in section 1.5 would be required for the completion of this research: 1) we will need to further study if the end user experience can be related directly to the LLDM measures; 2) the base (low level) measures, captured in the logs, will have to be mapped to the framework's performance characteristics, which is the intended topic of the next paper; and finally 3) the measures will be validated using the validation method presented in this report. It is important to note that there were significant difficulties with the utilization of the proposed framework, specifically the challenges of mapping the base measures into the quality characteristics defined by the author. A set of improvements to the framework are being discussed and will be part of another research paper.

A second research paper was presented to IARIA's Cloud Computing 2015. In that paper, the authors applied a measurement procedure to predict the degraded state of a private cloud application using only the available data center log LLDM of an ongoing case study. The intent was to improve the discussion of service level agreements of a widely used private cloud computing application (i.e. 80,000 end users on 600 servers world-wide). In organizations, cloud application performance measuring is often based on subjective and qualitative measures with very little research to address the large-scale private cloud perspective. Furthermore, measurement recommendations from ISO proposals (i.e. ISO 250xx series, ISO/IEC 15939 and more recently ISO/SC38-SLA) are poorly adopted by the industry, mainly due to the high degree of complexity in implementing the measurement concepts described in these international standards. To achieve this, the ISO 25010 performance efficiency characteristics are used with a number of LLDMs to model the utilization state of the private cloud computing application using indicators such as normal, abnormal, adequate or degraded. This paper

applies the approach developed in the previously accepted work presented at IWSM/Mensura, 2014, October 6-8, Rotterdam, for collecting the many cloud measures currently available in the logs of each private cloud component, then reducing the measures using statistical exploration, which has led to some findings involving the relation between the measures. We further conducted calculations for representing the indicators of the quality characteristics described in the ISO standard (Maturity, Fault Tolerance, Availability, Recoverability, Time Behavior, Resource Utilization, and Capacity).

The research journal was published in August 2016 using the approaches of both the previous papers and applying them in the discovery of the most relevant performance measures for root cause analysis of performance degradation events on a private cloud computing application. This journal uses most of the same data as described in chapter 4.

ANNEX II

COMPLETE LIST OF IDENTIFIED MEASURES

Performance Log Data Measure Name	CCS component type	ISO 25000 Quality Concept	Quality Evaluation L=lowest, H=highest is better
\ LogicalDisk(*)\Free Megabytes	Client, Server	capacity	H
\Netlogon(*)\Average Semaphore Hold Time	Server	maturity	L
\Memory\Page Faults/sec	Client, Server	maturity	L
\Memory\Available Bytes	Client, Server, network	capacity	H
\Memory\Pages/sec	Client, Server	time behavior	L
\Paging File(*)\% Usage	Client, Server	time behavior	L
\System\File Read Bytes/sec	Client, Server	resource utilization	L
\System\File Write Bytes/sec	Client, Server	resource utilization	L
\System\System Up Time	Client, Server	availability	H
\System\Processor Queue Length	Client, Server	time behavior	L
\System\Processes	Client, Server	availability	L
\System\Threads	Client, Server	capacity	L
\Processor Information(*)\% Privileged Time	Client, Server, Network	resource utilization	L
\Processor Information(*)\% User Time	Client	resource utilization	L
\Processor Information(*)\% Processor Time	Client, Server, network	resource utilization	L
\LogicalDisk(*)\Current Disk Queue Length	Client, Server	time behavior	L
\PhysicalDisk(*)\Disk Reads/sec	Client, Server	capacity	L
\PhysicalDisk(*)\Disk Writes/sec	Client, Server	capacity	L
\Processor(*)\% Processor Time	Client, Server, Network	time behavior	L
\Processor(*)\% User Time	Client	time behavior	L
\Processor(*)\% Privileged Time	Client, Server, network	time behavior	L
\Search Indexer(*)\Master Index Level	Client	maturity	H
\Client Side Caching\Application Bytes Read From Server (Not Cached)	Server	fault tolerance	L
\Client Side Caching\Application Bytes Read From Server	Server	fault tolerance	L
\Client Side Caching\Application Bytes Read From Cache	Server	recoverability	H
\Client Side Caching\Prefetch Bytes Read From Server	Server	fault tolerance	L
\Client Side Caching\Prefetch Bytes Read From Cache	Server	fault tolerance	H
\Client Side Caching\Prefetch Operations Queued	Server	fault tolerance	L
\Client Side Caching\SMB BranchCache Hash Bytes Received	Server	fault tolerance	H
\Client Side Caching\SMB BranchCache Hashes Received	Server	fault tolerance	H
\Client Side Caching\SMB BranchCache Hashes Requested	Server	fault tolerance	H
\Client Side Caching\SMB BranchCache Bytes Requested From Server	Server	time behavior	L
\Client Side Caching\SMB BranchCache Bytes Published	Server	time behavior	L
\Client Side Caching\SMB BranchCache Bytes Received	Server	time behavior	H
\Client Side Caching\SMB BranchCache Bytes Requested	Server	fault tolerance	L
\Offline Files\Bytes Received/sec	Client	time behavior	L
\Offline Files\Bytes Transmitted/sec	Client	maturity	L

ANNEX II (Continued)

\Offline Files\Bytes Transmitted	Client	maturity	L
\Offline Files\Bytes Received	Client	time behavior	L
\Terminal Services\Total Sessions	Server	availability	L
\Terminal Services\Inactive Sessions	Server	maturity	L
\Terminal Services\Active Sessions	Server	availability	L
\Security System-Wide Statistics\NTLM Authentications	Server	maturity	L
\Security System-Wide Statistics\Kerberos Authentications	Server	maturity	L
\Distributed Transaction Coordinator\Active Transactions	Server	availability	L
\Distributed Transaction Coordinator\Committed Transactions	Server	time behavior	L
\Security Per-Process Statistics(*)\Credential Handles	Server, network	maturity	L
\Security Per-Process Statistics(*)\Context Handles	Server, network	resource utilization	L
\Authorization Manager Applications(*)\Number of Scopes loaded in memory	Network	resource utilization	L
\Authorization Manager Applications(*)\Total number of scopes	Network	resource utilization	L
\Network Interface(*)\Bytes Received/sec	Host, server, network	capacity	L
\Network Interface(*)\Bytes Sent/sec	Host, server, network	capacity	L
\Process(*)\% Processor Time	Host, Server	resource utilization	L
\Process(*)\% User Time	Host	resource utilization	L
\Process(*)\% Privileged Time	Host, Server	resource utilization	L
\Process(*)\IO Read Operations/sec	Host, Server	resource utilization	L
\Process(*)\IO Write Operations/sec	Host, server	resource utilization	L

ANNEX III

ANOMALY DETECTION (SCREENS, UNTRAINED, TRAINED BAYES)

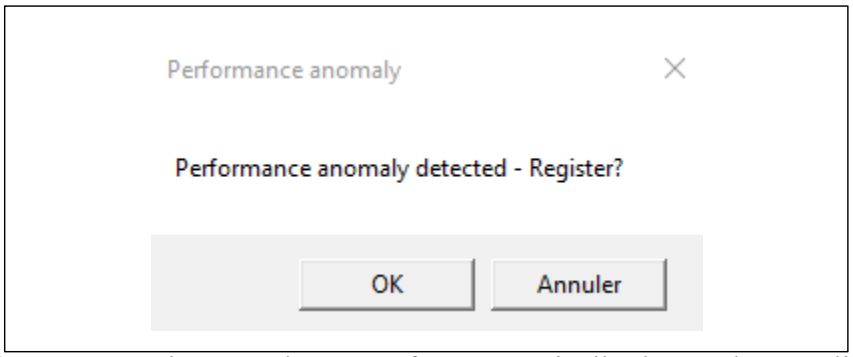

Fig A.1- Sample anomaly screen for automatically detected anomalies

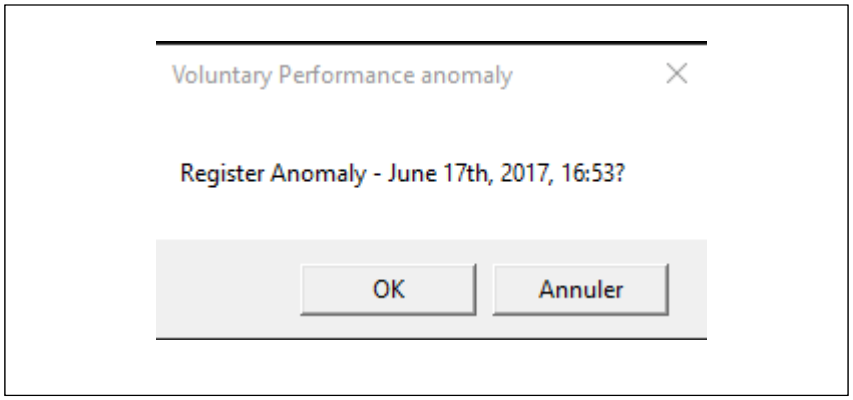

Fig A.2 - Sample anomaly screen for voluntary performance anomaly registration

Naïve Bayes statistics
483,000,000 data points.
Calculated anomalies: 14,445
Confirmed anomalies: 6,972
Voluntary anomalies: 1,911
Data set anomaly (average, stdev): 15.1779; 3.338
Untrained (average, stdev): 21.4436; 6.7712
Trained (average, stdev): 18.5531; 4,6920

Fig A.3 – Naïve Bayes statistics for experiment 4.6.8.3

BIBLIOGRAPHY

CA Technologies. (2014). *Improving Capacity Planning using Application Performance Management*. (CA technologies) Retrieved 02 26, 2016, from CA Technlogies: http://www.ca.com/content/dam/ca/us/files/solution-brief/reliably-predict-infrastructure-needs-with-ca-application-performance-management.PDF

Abran, A. (2010). *Software Metrics and Software Metrology.* New York: John Wiley & Sons Interscience and IEEE-CS Press.

Adams, S. (2011). *ITIL V3 foundation handbook (Vol. 1).* . Stationery Office.

Agendaless Consulting and Contributors. (2017, 06 18). *Supervisor Process Control*. Retrieved from Agendaless Supervisor: http://supervisord.org/

Alinezhad, A., Masaeli, A., Esfandiari, N., & Mirhadi, M. (2010). *"Evaluation of Effectiveness of Implementing Quality Management System (ISO9001:2000) Using BSC Approach in NIGC".* Journal of Industrial Engineering 6, 33-42.

Apache foundation. (2017, June 18). *Apache Spark Overview.* Retrieved 07 04, 2014, from Apache Spark: http://spark.apache.org/

Armbrust, M., Fox, A., & Griffith, A. (2009, February 10). *Above the Clouds: A Berkeley View of Cloud Computing.* Retrieved from Berkeley.edu: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

Avran, A. (2010, August 03). *Benchmarking 5 Cloud Platforms*. Retrieved from Infoq.com: http://www.infoq.com/news/2010/07/Benchmarking-5-Cloud-Platforms.

Bach, J. (1997, August). Good Enough Quality: Beyond the Buzzword. *IEEE Computer*, 96 - 98.

Baer, T. (2011). *Application Performance is in the Eyes of the End-User.* OVUM, CYIT0122.

Basili, V., Selby, R., & Hutchens, D. (1986). Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*, 733-743.

Bautista, L., Abran, A., & April, A. (2012). Design of a Performance Measurement Framework for Cloud Computing. *Journal of Software Engineering and Applications*, Vol. 5 No. 2, pp. 69-75.

Bautista, L., Abran, A., & April, A. (2012). Design of a Performance Measurement Framework for Cloud Computing. *Journal of Software Engineering and Applications*, Vol. 5 No. 2, 2012, pp. 69-75.

Bundschuh, M., & Dekkers, C. (2008). *The IT Measurement compendium.* Berlin: Springer-Verlag.

Buyya, R., Yeo, C., Venugopal, S., Brober, J., & Brandic, I. (2009). *Vision, hype, and reality for delivering computing as the 5th utility.* . FGCS, Volume 25, I 6, June 2009, 599-616.

Cagnazzo, L., Taticchi, P., & Fuiano, F. (2010). Benefits, barriers and pitfalls comming from the ISO 9000 Implementation: the impact on business performances. *WSEAS Transactions on Business and Economics*, 311-321.

Castor, K. (2006, june 1). *Testing and benchmarking methodology*. Retrieved from Castor Testing: http://donutey.com/hardwaretesting.php

Chandler, A. D. (1962). *Strategy and Structure: chapters in the history of the American Industrial Enterprise.* Cambridge: MIT press.

Chandler, A. D. (2002). *The Visible Hand: The Managerial Revolution in American Business.* Cambridge: The Belknap Press.

Cohen, J., Dolan, B., Dunlap, M., & Hellerstein, J. (2009). MAD skills: New analysis practices for bid data. *Proceedings of the VLDB Endowment* , 1481-1492 .

Couper, M. P. (2012). Advantages and Disadvantages of Internet Survey Methods for Official Statistics. *4th International Workshop on Internet Survey Methods.* Ann Harbour.

Creeger, M. (2009, August 1). *CTO roundtable: cloud computing.* Retrieved from Communications of the ACM: https://cacm.acm.org/magazines/2009/8/34495-cto-roundtable/fulltext

Croll, A. (2013, March 20). *How Should we Measure clouds?* Retrieved from Information Week: http://www.informationweek.com/cloud-computing/software/how-should-we-measure-clouds/240151231

Davis, F. D. (1989). *Perceived usefulness, perceived ease of use, and user acceptance of information technology.* MIS Quarterly, 13(3), 319-339.

Davis, S. &. (2001). The mediating effects of intrinsic motivation, ease of use and usefulness perceptions on performance in first-time and subsequent computer users. *Interacting with Computers, 13(5)*, 549-580.

Davis, S., & Wiedenbeck, S. (2001). The mediating effects of intrinsic motivation, ease of use and usefulness perceptions on performance in first-time and subsequent computer users. *Interacting with Computers, 13(5)*, 549-580.

Dean, J., & Ghemawat, S. (2008). *MapReduce: Simplified data processing on large clusters.* communications of ACM 51(1):107-113.

Deming, W. E. (2000). *Out of the Crysis.* Princeton: The MIT PRess.

Denney, R. (2005). *Succeeding with use cases: Working Smart to Deliver Quality.* New Jersey: Addison-Wesley.

Dillon, T., Wu, C., & Chang, E. (2010). *Cloud Computing: Issues and Challenges.* Perth, WA: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference.

Dromey, R. G. (1995). A model for software product quality. *IEE transactions on Software Engineering; 21*, pp. 146-162.

Eckerson, W. (2013, December 29). *Creating Effective KPI's*. Retrieved from Information Management: www.information-management.com

Emery, J. C. (1964, December 29). The Impact of Information Technology on Organizations. *24th Annual Meeting, Academy of Management*, p. 1.

Esmael, B., Arnaout, A., Fruhwirth, R., & Thonhauser, G. (2015). A Statistical Feature-Based Approach for Operations Recognition in Drilling Time Series. *International Journal of Computer Information Systems and Industrial Management Applications, 5*, 454-461.

Etezadi-Amoli, J., & Farhoomand, A. (1996). A structural model of end user computing satisfaction and user performance. *Information & Management 30* , 65-73.

Fagan, M., & Neill, S. (2004). An empirical investigation into the relationship between computer self efficacy, anxiety, experience, support and usage. *Journal of Computer Information Systems, 44(2)*, 95-104.

Fagan, M., & Neill, S. (2004). An empirical investigation into the relationship between computer self efficacy, anxiety, experience, support and usage. *Journal of Computer Information Systems, 44(2)*, 95-104.

Feigenbaum, A. (1991). *Total Quality Control.* McGraw-Hill Companies; 3 Rev Sub edition (January 1, 1991).

Finch, J. K. (1951). *Engineering and Western Civilization.* Mcgraw-Hill.

Fishbein, M., & Ajzen, I. (1975). *Belief, Attitude, Intention and Behavior: An Introduction to Theory and Research.* Reading, MA: Addison-Wesley.

Forster, F. (2017, June 06). *Collectd Open source project*. Retrieved from Collectd - The system statistics collection daemon: http://www.collectd.org

Forster, F. (2017, January 23). *Collectd Open source project*. Retrieved from The system statistics collection daemon: www.collectd.org

Friedl, A., & Ubik, S. (2008). *Perfmon and Servmon: Monitoring Operational Status and Resources of Distributed Computing Systems.* CESNET Technical report 1/2008.

Gantz, J., & Reinsel, D. (2012). *THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East.* IDC.

Gartner. (2013, March 8). *Cloudharmony Project*. Retrieved from CloudHarmony: http://cloudharmony.com

Gilbert, F. (2011, 02). *Cloud computing legal issues: data location*. (IT Law Group) Retrieved 02 2016, from http://searchcloudsecurity.techtarget.com/tip/Cloud-computing-legal-issues-data-location

Glass, R. (1998). *Software runaways: Lessons Learned from Massive software project failures.* Englewood Cliffs, NJ: Prentice Hall.

Goodhue, D. L., & Thompson, R. L. (1995). *Task-technology fit and individual performance.* MIS Quarterly, 19, 2, 213-236.

Gotzamani, K. D. (2005). The implications of the new ISO 9000:2000 standards for certified organizations - A review of anticipated benefits and implementation pitfalls. *International Journal of Productivity and Performance Management*, 645-657.

Grady, R. B. (1992). *Pratical Software Metrics for Project Management and Process Improvement.* Englewood Cliffs, NJ: Prentice Hall.

Grobauer, B., Walloschek, T., & Stocker, E. (2011). Understanding Cloud Computing Vulnerabilities. *IEEE Security and Privacy, vol 9, no.2*, 50-57.

Gruschka, N., & Jensen, M. (2010, july). Attack surfaces: a taxonomy for attacks on cloud services. *Proceedings of the 3rd IEEE International Conference on Cloud Computing*, pp. 276-279.

Haldestead, M. (1975). *Elements of Software Science.* Holland: Elsevier.

Hambling, B., & van Goethem, P. (2013). *User Acceptance Testing: A Step by Step Guide.* BCS Learning and Development Limited, 2013.

Helland, P. (2011, 06). If You Have Too Much Data, then 'Good Enough' Is Good Enough. *Communications of the ACM*, pp. 40-47.

HP. (2013). *HP ITSM Transforming IT organizations into service providers.* Retrieved from http://h20427.www2.hp.com/program/ngdc/cn/zh/file/fuwu/management/ITSMBusinessWP.pdf

Huffman, C. (2017, May 4). *Performance Analysis of Logs Tool.* Retrieved from Codeplex: https://github.com/clinthuffman/PAL

Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1985). Direct manipulation interfaces. *Human-Computer Interaction, 1(4),*, 311-338.

Hyvärinen, A., & Oja, E. (2000). Independent Component Analysis: Algorithms and Applications. *Neural Networks, 4-5*(13), 411-430.

IBM. (2013, 10 14). *Three-Tiered client/server architecture*. Retrieved from Software Information Center: http://pic.dhe.ibm.com/infocenter/txformp/v7r1/topic/com.ibm.cics.tx.doc/concepts/c_wht_is_distd_comptg.html

Iosup, A., Ostermann, S., Nezih, Y., Prodan, R., Fahringer, T., & Epema, D. (2010, November). Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEE TPDS Many-Task Computing*.

ISACA. (2012). *Cobit 5: A business Framework for the Governance and Management of Enterprise IT.* Rolling Meadows: ISACA.

ISO. (1994). *ISO 5725-1:1994(en) - Accuracy (trueness and precision) of measurement methods and results — Part 1: General principles and definitions*. (ISO) Retrieved 04 22, 2016, from https://www.iso.org/obp/ui/#iso:std:iso:5725:-1:ed-1:v1:en

ISO. (2009). *ISO FDIS 9241-210:2009. Ergonomics of human system interaction - Part 210: Human-centered design for interactive systems (formerly known as 13407). International Organization for Standardization (ISO).*

ISO/IEC. (2003). *TR 9126-(1-4) Software Engineering - Product Quality.* Geneva: International Organization for Standardization.

ISO/IEC. (2005). *ISO/IEC 25000:2005 Software Engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, ISO/IEC JTC 1/SC 7.*

ISO/IEC. (2011). *ISO/IEC 20000-1:2011 Information technology -- Service management.*

Jackson, K., & Ramakrishnan, L. (2010). *Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. Cloud Computing Technology and Science (CloudCom)*. 2010 IEEE Second International Conference.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison Wesley.

Jacquet, J. P., & Abran, A. (1997). From Software Metrics to Software Measurement Methods: A Process Model. *Third International Symposium on Software Engineering Standards, ISESS.* Walnut Creek CA.

Jacquet, J.-P., & Abran, A. (1998). Metrics Validation Proposals: A Structured Analysis. *8th International Workshop on Software Measurement.* Magdeburg.

Juran, J., & De Feo, J. (2010). *Juran's Quality Handbook: The Complete Guide to Performance Excellence,*. McGraw-Hill.

Kaplan, R., & Norton, D. (1992). The Balanced Scorecard: Measures That Drive Performance. *Harvard Business Review*.

Kopp, M. (2011, May 11). *Troubleshoot Response Time Problems.* Retrieved from High Scalability: Building Bigger, Faster and more reliable websites: http://highscalability.com/blog/2011/5/11/troubleshooting-response-time-problems-why-you-cannot-trust.html

Kufrin, R. (2005, May 26). Measuring and improving application performance with Perfsuite. *Linux Journal*, pp. 1-5.

Laguë, B., & April, A. (1996). Mapping of the ISO9126 Maintainability Internal MEtrics to an industrial Research Tool. *SESS'96 conference proceedings.* Montreal.

Lamber, K. (2001). *IEEE Guidelines for 48-bit Global Identifier (EUI-48).* Retrieved 12 5, 2016, from IEEE : https://standards.ieee.org/develop/regauth/tut/eui48.pdf

Lamport, M., Seetanah, B., Cohhyedass, P., & Sannassee, R. V. (2010). *The association between ISO 9000 certification and financial performance.* Mauritius: International Research Symposium in Service Management.

Laudon, K., & Laudon, J. (2013). *Management Information Systems.* Princeton: Prentice Hall.

Law, E., Roto, V., Hassenzahl, M., Vermeeren, A., & Kort, J. (2009). *Understanding scoping and defining user experience: a survey approach. Proceedings of Human Factors in computing Systems,.* CHI' 09, pp 719-728. .

Lin, J., & Dyer, C. (2010). *Data-Intensive Text Processing with MapReduce.* University of Maryland, College Park.

Mahmood, M., Burn, J., Gemoets, L., & Jacquez, C. (2010). *Variables affecting information technology end-user satisfaction: a meta-analysis of the empirical literature.* IJHCS, Vol 54, I 4, 751-771.

Manyika, J., Chui, M., Brown, B., Bughin, J., & Dobbs, R. (2011). *Big data: The next frontier for innovation, competition, and productivity.* McKinsey Global Institute.

Marr, B., & Creelman, J. (2011). *Creating and Implementing a Balanced Scorecard: The Case of the Ministry of Works.* Advanced Performance Institute.

Marshall, B., Mills, R., & Olsen, D. (2008). The Role of End-User Training in Technology Acceptance. *Review of Business Information Systems*, 1-8.

Martensson, A. (2006). A resource allocation matrix approach to IT management. *Information Technology and Management, v7 i1*, 21-34.

Massie, M. (2012). *Monitoring with Ganglia.* O'Reilly Media.

McCabe, J. (1976). A complexity measure. *IEE transactions of Software Engineering*, SE-2(4).

Mccall, J. A., Richards, P. K., & Walters, G. F. (1977). *Factors in Software Quality, V I-III.* USA: US Rome Air Development Center Reports.

Mei, Y., Liu, L., Pu, X., & Sivathanu, S. (2010). *Performance Measurements and Analysis of Network I/O Applications in Virtualized Cloud.* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.6742&rep=rep1&type=pdf.

Mei, Y., Pu, X., & Sivathanu, S. (2010). *Performance Measurements and analysis of Network I/O Applications in Virtualized Cloud.* Atlanta, GA: Georgia Institute of Technology.

Meijer, G. (2012, April 19). *How do you Measure Cloud Performance.* Retrieved from Cloud Provider USA: http://www.cloudproviderusa.com/how-do-you-measure-cloud-performance

Microsoft. (2013). *Microsoft Perfmon*. Retrieved from http://technet.microsoft.com/en-us/library/cc749249.aspx

Microsoft. (2013). *Microsoft Perfmon*. Retrieved from Microsoft: http://technet.microsoft.com/en-us/library/cc749249.aspx

Mirzaei, N. (2008). *Cloud Computing.* Institute Report, Community Grids Lab, Indiana.Edu.

Munin and colaborators. (2017, January 21). *Munin Monitoring Open Project*. Retrieved from Munin: http://munin-monitoring.org/

Nagios. (2013). *Nagios IT Infrastructure Monitoring*. Retrieved from Nagios: www.nagios.org

Nagumo, T., & Donion, B. (2006). Integrating the balance scorecard and the COSO ERM frameworks. *Cost Management*(20), 20-30.

Naur, P., & Randell, B. (1969). *Software Engineering: report on a conference sponsored by the NATO science committee.* Garmisch: NATO.

NIST - National Institute of Standards and Technology. (2011). *The Nist Definition of Cloud Computing.* Gaithersburg, MD: CSD - TIL - NIST.

Observium Limited . (2013, April 2). *Observium: autodiscovering SNMP network monitoring*. Retrieved from Observium Network Monitoring and Management: www.observium.org

Omniti Labs. (2014, june 20). *Reconnoiter Fault Detection and Trending.* Retrieved from Omniti Labs: https://labs.omniti.com/labs/reconnoiter

Parmenter, D. (2010). *Key Performance Indicators (KPI): Developing, Implementing and Using Winning KPIs.* Wiley.

Phaphoom, N., Wang, X., & Abrahamsson, P. (2012, December 12). Foundations and technological Landscape of Cloud Computing. *ISRN Software Engineering*, p. 31.

Phaphoon, N., Oza, N., Wang, X., & Abrahamsson, P. (2012). Does Cloud Computing deliver the promised benefits for IT industry? *Proceedings of the WICSA/ECA*, 45-52.

Pivotal Software. (2013). *Cloudsleuth Project*. Retrieved from Spring Cloud Sleuth: https://cloud.spring.io/spring-cloud-sleuth/

Poksinska, B., Kahlgaard, J., & Antoni, M. (2002). The State of ISO 9000 Certification: A study of Swedish Organizations. *THe TQM Magazine*, 297 - 306.

Prasad, R. B., & Choi, E. (2010). *A taxonomy, survey, and issues of Cloud Computing Ecosystems.* London: Springer-Verlag.

Rabl, T. (2012). *solving big data challenges for enterprise application performance management.* Proceedings of the VLDB Endowment, Vol. 5, No. 12.

Rapoza, J. (2015, 01). *Optiomize IT Infrastructure to Maximize Workload Performance*. (Hewlett Packard) Retrieved 2 25, 2016, from http://h20195.www2.hp.com/v2/getpdf.aspx/4AA5-6394ENW.pdf

Reeve, A. (2012, 9 7). *Big Data and NoSQL: The Problem with Relational Databases*. (Dell EMC) Retrieved 12 16, 2016, from https://infocus.emc.com/april_reeve/big-data-and-nosql-the-problem-with-relational-databases/

Russel, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* Prentice Hall.

Sandeep, S. R., Swapna, M., Niranjan, T., Susarla, S., & Nandi, S. (2008, June 18). *CLUEBOX: A Performance Log Analyzer for Automated Troubleshooting*. (Netapp, Inc) Retrieved 03 14, 2016, from Usenix.org: https://www.usenix.org/legacy/event/wasl08/tech/full_papers/sandeep/sandeep_html/

Shaw, M. (1990). Prospects for an Engineering Discipline of Software. *IEEE Software*, 15-24.

Shewhart, W. A. (2015). *Economic Control of Quality of Manufactured Product.* Eastford: Martino Fine Books.

Sousa-Poza, A., Altinkilink, M., & Searcy, C. (2009). Implementing a Functional ISO 9001 Quality Management System in Small and Medium-Sized Enterprises. *International Journal of Engineering*, 220-228.

St-Amour, L. (2011, July 19). *The Complete List of End User Experience Monitoring Tools*. Retrieved from Real User Monitoring: http://www.real-user-monitoring.com/the-complete-list-of-end-user-experience-monitoring-tools/

Stavrinoudis, X. (2008). *"Comparing internal and external software quality measurements"*. (I. Press, Ed.) Piraeus, Greece: Proccedings of the 8th Joint Conference on Knowledge-Based Software Engineering.

Suakanto, S., Supangkat, S. H., & Suhardi, R. S. (2012). *Performance Measurement of Cloud Computing Services*. International Journal on Cloud Computing: Services and Architecture (IJCCSA), April 2012, Volume 2, Number 2.

Suakanto, S., Supangkat, S., Saragih, S., & Saragih, R. (2012, April). Performance Measurement of Cloud Computing Services. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, p. vol 2 no 2.

Taleb, N. N. (2013, 08). *Beware the Big Errors of 'Big Data'*. Retrieved 07 2014, from http://www.wired.com/opinion/2013/02/big-data-means-big-errors-people/

The Cacti Group. (2017, june 11). *Cacti RRDTool*. Retrieved from Cacti: http://cacti.net/

Tidelash Inc. (2017, Jun 07). *Monit process monitor*. Retrieved from Tidelash: http://mmonit.com/monit/

Trappler, T. J. (2011). *Cloud Adviser: Where's your data?* (Computerworld) Retrieved 2016, from: http://www.computerworld.com/article/2500232/cloud-computing/cloud-adviser--where-s-your-data-.html

Trelles, O., Prins, P., Snir, M., & Jansen, R. (2011). *Big data, but are we ready?* Nature Reviews Genetics, 8 February 2011.

Tsai, T., Lopez, A., Rodriguez, V., & Volovik, D. (1986). An approach to Measuring Data Structure Complexity. *COMPSAC86*, 240-246.

Tullis, T., & Albert, B. (2010). *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics.* Morgan Kaufmann.

US General Service Administration. (2010). *Cloud Computing Initiative Vision and Strategy Document.* Washington: US General Press.

Voss, J., & Zhang, J. (2009). Cloud computing: newwine or just a new bottle? *IT professional, vol 11, n 2*, 15-17.

Weinman, J. (2009). *Cloudonomics:The Business Value of Cloud Computing.* Toronto: John Wiley & Sons.

Weisberg, J. (2013, February 13). *Argus TCP Monitor*. Retrieved from http://argus.tcp4me.com

Zabbix. (2017, 03 08). *Zabbix: Enterprise Class Open source Distributed Monitoiring Solution.* Retrieved from Zabbix: www.zabbix.com

Zenoss. (2013). *Zenoss Unified IT Operations Monitor*. Retrieved from Zenoss: www.zenoss.com