UNIVERSITY OF NICOSIA

Debt-Aware Resource Adaptation in Cloud Elasticity Management

Georgios Skourletopoulos

PhD (Doctor of Philosophy) in Computer Science

November 2020

UNIVERSITY of NICOSIA

Georgios Skourletopoulos

NICOSIA

PhD

2020

UNIVERSITY of NICOSIA

Debt-Aware Resource Adaptation in Cloud Elasticity Management

Georgios Skourletopoulos

A thesis submitted to the University of Nicosia
in accordance with the requirements of the degree of
PhD (Doctor of Philosophy) in Computer Science
Department of Computer Science
School of Sciences and Engineering

November 2020

UNIVERSITY of NICOSIA

Copyright © 2020 Georgios Skourletopoulos

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author except as permitted by law.

Debt-Aware Resource Adaptation in Cloud Elasticity Management

Georgios Skourletopoulos

Abstract

Cloud computing is a well-consolidated paradigm for on-demand provisioning of computing and storage resources that is highly motivated by economies of scale. Elasticity is a fundamental property of cloud computing that enables to maintain, grow or shrink virtual resources while handling the arbitrary workload variation. This research thesis motivates the need for resource adaptation decisions valued from an innovative economics-driven methodology that considers the trade-offs of compromising long-term benefits for short-term gains as a dimension to mitigate imperfections when adapting resource provisioning under the uncertainty of the cloud environment. In this sense, the technical debt metaphor is mapped in the context of cloud elasticity management and service composition. The topic of technical debt has gained significant traction in the software engineering community and it is used to encapsulate numerous long-term quality problems as a result of the acceleration of the velocity when developing and releasing software. Drawing the analogy between technical debt and cloud elasticity, the *elasticity debt* term is formed as the disruptive solution concept for optimal resource adaptations with value creation and strategy-driven incentive mechanisms in multi-tenant software as a service cloud environments. Elasticity debt refers to the suboptimal adaptations or conflicting trade-offs responsible for affecting the utility of the system or application owner. One of the aspects of dynamic resource adaptation is that, due to workload variations, the composition encounters underutilization or overutilization on the composite service components. In the underutilization state, the pay-off is significant in the long run with far-sighted benefits, while in the overutilization state, there is no pay-off and, if exists, it is trivial. Such a definition matches the intentional and unintentional technical debt encountered in software engineering.

In this context, this research work elaborates on a complete solution to debt-aware dynamic resource adaptation in cloud elasticity management with value creation- and strategy-oriented reasoning. The goal of this work is to optimize the resource provisioning and utilization in cloud-centric systems under balanced, well-compromised trade-off decisions. The novel

UNIVERSITY of NICOSIA

contributions and impact of this research project on the theory and practice of cloud elasticity management are the following:

1. A theoretical, economics-driven framework to support value creation in cloud elasticity management and service composition leveraging the technical debt metaphor.
2. Novel debt-aware models and algorithms with value creation and strategy-driven considerations to reason about elasticity resource adaptations, embracing the approach of unavoidable gaps in resource provisioning between the resource supply and demand and quantifying the interest during the overutilization or underutilization states of the composite service component utility as a result of the dynamic fluctuations in request workload to drive elasticity adaptations.
3. Novel game theoretic control mechanisms with debt-aware elasticity considerations, formation of dynamic coalitions and optimization of trade-off decisions for optimal resource utilization and exchange of resource capacity in multi-tenant software as a service cloud environments. The analysis of the utility-driven state of debt minimization and profit maximization is presented from the service provider viewpoint based on a novel Hidden Markov Model, followed by a novel mobile opportunistic offloading model in the mobile cloud computing framework for task offloading on the cloud after denial by two mobile device nodes in close proximity due to resource restrictions.
4. Implementation of a prototype tool for debt-aware quantification at cloud service utility level through the integration of the designed models and algorithms, allowing the provision of quantitative reports, dashboards and insights into the overutilization and underutilization states of service components in multi-tenant applications hosted in the cloud.

UNIVERSITY of NICOSIA

Keywords—cloud computing, cloud elasticity management, debt-aware, elasticity debt, game theory, hidden markov model, multi-tenant software as a service, resource adaptation, resource provisioning, resource utilization, service composition, service utility.

Acknowledgements

This research work was carried out during my stay as a doctoral researcher at the University of Nicosia, Cyprus. With a debt of gratitude, I would like to thank my supervisor Prof. Constandinos X. Mavromoustakis for his advice and generous support during my research at the Mobile Systems Laboratory and throughout my Ph.D. candidature. His practical and sharp vision in research has not only been invaluable for my work on this doctoral thesis, but also for developing my taste in research and my development as a researcher. I extend my sincere gratitude to my second supervisor Prof. John N. Sahalos for his support during the period of the Ph.D., and my third supervisor Prof. George Mastorakis for his insightful suggestions and stimulating discussions, which have trained me in research, as well as his encouragement during various stages of my Ph.D. I am deeply indebted to my Master's thesis supervisor Prof. Rami Bahsoon from the University of Birmingham, UK for inducing me in research and his overall support and guidance since 2012. I would also like to thank the Ph.D. committee members and the external examiners for their excellent reviews and constructive suggestions on improving this research thesis.

My sincere thanks to the people I met during my research visits. In particular, I would like to thank Prof. Ciprian Dobre from the Department of Computer Science and Engineering at the University Politehnica of Bucharest, Romania for the useful discussions. Also, I thank Prof. Nuno M. Garcia from the Assisted Living Computing and Telecommunications Laboratory, Instituto de Telecomunicações at the University of Beira Interior, Portugal for the support.

I acknowledge the University of Nicosia for providing me with the research scholarship to pursue my doctoral studies. The research visit to the University Politehnica of Bucharest was supported by the EU ICT COST Action IC1406 on 'High-Performance Modelling and Simulation for Big Data Applications (cHiPSet)' research grant, while the visit to the University of Beira Interior was supported by the EU ICT COST Action IC1303 on 'Algorithms, Architectures and Platforms for Enhanced Living Environments (AAPELE)' research grant.

Georgios Skourletopoulos

Nicosia, Cyprus

November 2020

"Work as hard as if you were having fun. You can then create something from nothing, and change the history. A man can achieve greatness with such a mentality."

— Georgios Skourletopoulos

UNIVERSITY of NICOSIA

"Excellence is never an accident. It is always the result of high intention, sincere effort, and intelligent execution; it represents the wise choice of many alternatives – choice, not chance, determines your destiny."

— Aristotle

Declaration

I declare that the work in this thesis was carried out in accordance with the regulations of the University of Nicosia. It is a product of original research work of my own towards the Ph.D. and due acknowledgment has been made in the text to all other material used through references, notes, or any other statements.

Georgios Skourletopoulos

Georgios Skourletopoulos, November 2020

UNIVERSITY of NICOSIA

Scientific Publications Resulting from Thesis

Archival International Peer-Reviewed Journal Articles

[1] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, H. Song, J. N. Sahalos, and E. Pallis, "Elasticity Debt Analytics Exploitation for Green Mobile Cloud Computing: An Equilibrium Model", in *IEEE Transactions on Green Communications and Networking Journal (TGCN)*, IEEE Communications Society, IEEE Signal Processing Society, and IEEE Vehicular Technology Society, vol. 3, Issue 1, 19 March, 2019, pp. 122-131. DOI: 10.1109/TGCN.2018.2890034.

[2] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, and J. N. Sahalos, "An Evaluation of Cloud-Based Mobile Services with Limited Capacity: A Linear Approach", in *Soft Computing Journal*, vol. 21, Issue 16, Germany: Springer-Verlag Berlin Heidelberg (Springer Nature), 5 August, 2017, pp. 4523-4530. DOI: 10.1007/s00500-016-2083-4.

Archival Proceedings of International Peer-Reviewed Conference Papers

[3] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, H. Song, J. N. Sahalos, and E. Pallis, "Elasticity Debt Analytics Exploitation for Green Mobile Cloud Computing: An Equilibrium Model", in *Proceedings of the 2018 52nd IEEE International Conference on Communications (ICC), Communications QoS, Reliability and Modelling Symposium (CQRM)*, IEEE Communications Society, Kansas City, Missouri, USA, 20-24 May, 2018, pp. 1-6. DOI: 10.1109/ICC.2018.8422956.

[4] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. N. Sahalos, J. M. Batalla, and C. Dobre, "A Game Theoretic Formulation of the Technical Debt Management Problem in Cloud Systems", in *Proceedings of the 2017 14th IEEE International Conference on Telecommunications (ConTEL), 4th International Workshop (Special Session) on Enhanced Living Environments (ELEMENT 2017)*, IEEE Communications Society, Zagreb, Croatia, 28-30 June, 2017, pp. 7-12. DOI: 10.23919/ConTEL.2017.8000012.

[5] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. N. Sahalos, J. M. Batalla, and C. Dobre, "Cost-Benefit Analysis Game for Efficient Storage Allocation in Cloud-Centric Internet of Things Systems: A Game Theoretic Perspective", in *Proceedings of the 2017 15th IFIP/IEEE International Symposium on Integrated Network Management*

(IM), 2017 First International Workshop on Protocols, Applications and Platforms for Enhanced Living Environments (PAPELE 2017), IEEE Communications Society, Lisbon, Portugal, 8-12 May, 2017, pp. 1149-1154. DOI: 10.23919/INM.2017.7987453.

[6] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, P. Chatzimisios, and J. M. Batalla, "A Novel Methodology for Capitalizing on Cloud Storage through a Big Data-as-a-Service Framework", in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Fifth IEEE International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA)*, IEEE Communications Society, Washington, District of Columbia, USA, 4-8 December, 2016, pp. 1-6. DOI: 10.1109/GLOCOMW.2016.7848821.

[7] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, E. Pallis, J. M. Batalla and G. Kormentzas, "Quantifying and Evaluating the Technical Debt on Mobile Cloud-Based Service Level", in *Proceedings of the 2016 50th IEEE International Conference on Communications (ICC), Communications QoS, Reliability and Modelling Symposium (CQRM)*, IEEE Communications Society, Kuala Lumpur, Malaysia, 23-27 May, 2016, pp. 1-7. DOI: 10.1109/ICC.2016.7510995.

[8] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, E. Pallis, P. Chatzimisios, and J. M. Batalla, "Towards the Evaluation of a Big Data-as-a-Service Model: A Decision Theoretic Approach", in *Proceedings of the 2016 35th IEEE International Conference on Computer Communications (INFOCOM), First IEEE International Workshop on Big Data Sciences, Technologies, and Applications (BDSTA)*, IEEE Communications Society, San Francisco, California, USA, 10-15 April, 2016, pp. 877-883. DOI: 10.1109/INFCOMW.2016.7562202.

[9] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. J. P. C. Rodrigues, P. Chatzimisios, and J. M. Batalla, "A Fluctuation-Based Modelling Approach to Quantification of the Technical Debt on Mobile Cloud-Based Service Level", in *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), Fourth IEEE International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA)*, IEEE Communications Society, San Diego, California, USA, 6-10 December, 2015, pp. 1-6. DOI: 10.1109/GLOCOMW.2015.7413999.

[10] **G. Skourletopoulos**, R. Bahsoon, C. X. Mavromoustakis, G. Mastorakis, and E. Pallis, "Predicting and Quantifying the Technical Debt in Cloud Software Engineering", in *Proceedings of the 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, IEEE Communications

Society, Athens, Greece, 1-3 December, 2014, pp. 36-40. DOI: 10.1109/CAMAD.2014.7033201.

Books (Authored and Edited)

[11] **G. Skourletopoulos**, G. Mastorakis, C. X. Mavromoustakis, C. Dobre, and E. Pallis, (Eds.) "Mobile Big Data: A Roadmap from Models to Technologies", in *Lecture Notes on Data Engineering and Communications Technologies Book Series (LNDECT)*, 1st edition, vol. 10, Cham, Switzerland: Springer International Publishing AG (Springer Nature Switzerland AG), 1 November, 2017, pp. 1-347. DOI: 10.1007/978-3-319-67925-9, Print ISBN: 978-3-319-67924-2, Online ISBN: 978-3-319-67925-9, Series Print ISSN: 2367-4512, Series Online ISSN: 2367-4520.

International Peer-Reviewed Book Chapters

[12] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, C. Dobre, J. N. Sahalos, R. I. Goleva, and N. M. Garcia, "Game Theoretic Approaches in Mobile Cloud Computing Systems for Big Data Applications: A Systematic Literature Review", in *Mobile Big Data: A Roadmap from Models to Technologies, Lecture Notes on Data Engineering and Communications Technologies Book Series (LNDECT)*, 1st edition, vol. 10, G. Skourletopoulos, G. Mastorakis, C. X. Mavromoustakis, C. Dobre, and E. Pallis, (Eds.), Cham, Switzerland: Springer International Publishing AG (Springer Nature Switzerland AG), 1 November, 2017, pp. 41-62. DOI: 10.1007/978-3-319-67925-9_3.

[13] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, C. Dobre, S. Panagiotakis, and E. Pallis, "Big Data and Cloud Computing: A Survey of the State-of-the-Art and Research Challenges", in *Advances in Mobile Cloud Computing and Big Data in the 5G Era, Studies in Big Data Book Series (SBD)*, 1st edition, vol. 22, C. X. Mavromoustakis, G. Mastorakis, and C. Dobre, (Eds.), Cham, Switzerland: Springer International Publishing AG (Springer Nature Switzerland AG), 20 November, 2016, pp. 23-41. DOI: 10.1007/978-3-319-45145-9_2.

[14] **G. Skourletopoulos**, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, C. Dobre, S. Panagiotakis, and E. Pallis, "Towards Mobile Cloud Computing in 5G Mobile Networks: Applications, Big Data Services and Future Opportunities", in *Advances in Mobile Cloud Computing and Big Data in the 5G Era, Studies in Big Data Book Series (SBD)*, 1st edition, vol. 22, C. X. Mavromoustakis, G. Mastorakis, and C. Dobre, (Eds.), Cham,

Switzerland: Springer International Publishing AG (Springer Nature Switzerland AG), 20 November, 2016, pp. 43-62. DOI: 10.1007/978-3-319-45145-9_3.

[15] **G. Skourletopoulos**, R. Bahsoon, C. X. Mavromoustakis, and G. Mastorakis, "The Technical Debt in Cloud Software Engineering: A Prediction-Based and Quantification Approach", in *Resource Management of Mobile Cloud Computing Networks and Environments, Advances in Systems Analysis, Software Engineering, and High Performance Computing Book Series (ASASEHPC)*, 1st edition, G. Mastorakis, C. X. Mavromoustakis, and E. Pallis, (Eds.), Hershey, Pennsylvania, USA: IGI Global (Idea Group, Inc.), March, 2015, pp. 24-42. DOI: 10.4018/978-1-4666-8225-2.ch002.

International Peer-Reviewed Scientific and Research Reports

[16] **G. Skourletopoulos**, "Cost-Benefit Analysis Game for Efficient Storage Allocation in Cloud-Centric Internet of Things Systems: A Game Theoretic Perspective", *Short-Term Scientific Mission (STSM) Scientific Report COST-STSM-IC1303-37094 (ECOST-STSM-IC1303-270317-084686), EU ICT COST Action IC1303 on 'Algorithms, Architectures and Platforms for Enhanced Living Environments (AAPELE)'*, Brussels, Belgium, 10 April, 2017, pp. 1-4.

UNIVERSITY of NICOSIA

Table of Contents

UNIVERSITY of NICOSIA

UNIVERSITY of NICOSIA

List of Tables

UNIVERSITY of NICOSIA

List of Figures

UNIVERSITY of NICOSIA

List of Appendices

UNIVERSITY of NICOSIA

Abbreviation Index

SaaS	Software as a Service
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
BDaaS	Big Data as a Service
NFaaS	Network Functions as a Service
MCC	Mobile Cloud Computing
QoS	Quality of Service
QoE	Quality of Experience
SLO	Service Level Objective
SLA	Service Level Agreement
EDQ	Elasticity Debt Quantification
EDA	Elasticity Debt Analytics
EDO	Elasticity Debt Optimization
PRO	Profit Optimization
CA	Cost Analysis
C_D	Cost Difference
B	Benefits
HMM	Hidden Markov Model
ROI	Return on Investment
TD	Technical Debt
DWH	Data Warehouse
CPU	Central Processing Unit
I/O	Input/Output
RAM	Random-Access Memory
VM	Virtual Machine
TB	Terabytes

UNIVERSITY of NICOSIA

COCOMO	Constructive Cost Model
$KLOC$	Thousands (kilo) of lines of code
UML	Unified Modeling Language
ERD	Entity-Relationship Diagram
IT	Information Technology
W3C	World Wide Web Consortium
XML	Extensible Markup Language
JPA	Java Persistence API
UI	User Interface
POJO	Plain Old Java Object
JSP	JavaServer Pages
SQL	Structured Query Language
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
NFV	Network Functions Virtualization
MVNO	Mobile Virtual Network Operator
VR	Virtual Router

UNIVERSITY of NICOSIA

Chapter 1

Introduction

Cloud computing has revolutionized the information and communication technology industry as a paradigm that enables the on-demand provisioning and release of virtual computing resources [1]. It is deemed an economics-driven model promising cost savings, as organizations can either outsource their computational needs under a pay as you go model or build a private cloud data centre for their resource management and provisioning purposes [2].

The large-scale cloud data centres facilitate the dynamic sharing of a pool of computing resources among multiple tenants at runtime; a fact that increases the resource utilization [3], evolving the advantages of economies of scale [4]. In this context, monitoring and controlling the status and availability of the of the physical resources and services present to the cloud infrastructure is essential for cloud providers to design better provisioning strategies and avoid service level agreement (SLA) violations [5]. However, the challenge here lies in the difficulty of managing this information in a reliable way due to the nature of the cloud-centric environments where multiple tenants interact for shared resources and services with dynamic changes in requests' workload [6].

Elasticity is a fundamental feature of cloud computing that enables adaptations in resource provisioning at runtime, i.e., maintain, grow or shrink computing resources in a scalable way, in order to handle the dynamic resource demand, such that the match between the resource supply and demand guarantees the Quality of Service (QoS) requirements are met under specific SLAs [7]. In this sense, application re-dimensioning can be implemented effortlessly by adapting the resources assigned to an application to the incoming user demand. However, the challenge here lies in the identification of the right and tolerable amount of virtual resources to lease off in order to meet the required SLA, while keeping the overall cost at the lowest level [8].

Although the elasticity facilitates the economies of scale in the cloud context, an elasticity management framework in cloud environments is imperative to analyse and coordinate adaptation decisions at runtime while considering value creation and strategic aspects in accordance with far-sighted goals. However, the challenge here has to do with the uncertainty of the cloud environment, which allows multiple trade-off decisions to be taken between conflicting strategies, e.g., quality-driven, cost-driven or energy-driven. In the presence of

these uncompromised trade-off decisions, the perfect elasticity in cloud seems illusionary as a perfect match between resource supply and demand cannot be achieved [9].

To cope with the above problems, this research thesis focuses on the problem of debt-aware resource adaptation in cloud elasticity management with value creation- and strategy-oriented considerations, i.e., ensuring that resources are efficiently allocated and utilized to serve multi-tenant software as a service application request workloads, while optimizing the trade-off decisions in the context of cloud resource provisioning. The reminder of this chapter is organized as follows: in the next section, the research gaps that this research work fills are analysed. Section 1.2 investigates the research questions associated with the debt-aware resource adaptation problem in cloud elasticity management for multi-tenant software as a service cloud environments, covering also the research methodology developed in Section 1.3. Section 1.4 discusses the novel research contributions of this thesis on the theory of cloud elasticity management. The chapter is concluded with the thesis organization in Section 1.5.

1.1. Debt-Aware Resource Adaptation Problem in Cloud Elasticity Management

Dynamic adaptation decisions trade off short-term against long-term benefits and involve risks due to the uncertainty in cloud environments. Elasticity debt is defined as the suboptimal adaptations or conflicting trade-offs under the uncertain context of the cloud environment that are able to affect the system's or application's utility. An innovative economics-driven methodology is therefore recommended that considers the trade-offs of compromising long-term benefits for short-term gains when adapting resource provisioning. One of the aspects of dynamic resource adaptation is that due to workload variations, the composition encounters underutilization or overutilization on the composite service components. In the underutilization state, the pay-off is significant in the long run with far-sighted benefits, while in the overutilization state, there is no pay-off and, if exists, it is trivial. Such a definition matches the intentional and unintentional technical debt met in the software engineering research field. In this sense, the technical debt metaphor [10] is mapped in the context of cloud elasticity management and service composition as the disruptive solution concept for optimal resource

adaptations with value creation incentive mechanisms in multi-tenant software as a service applications.

In this context, this thesis presents a complete solution to debt-aware dynamic resource adaptation in cloud elasticity management with value creation and strategy considerations. The goal is to optimize resource provisioning and utilization in the cloud under dynamic balanced, well-compromised trade-off decisions in cloud service composition. This work focuses on multi-tenant software as a service cloud environments where multiple tenants submit requests for provisioning of VMs, resulting in arbitrary workload, and deploy various types of applications. Since multiple types of applications can coexist in the system and share physical resources, there is a challenge of defining QoS requirements in a workload independent manner to avoid any application performance degradation. The multiple tenants establish SLAs with the cloud provider to formalize the QoS requirements and the provider pays a penalty in case of SLO violations.

Different from the related work presented in chapter 2, underutilization or overutilization states on a composite service component utility are critical when developing this novel elasticity management framework. The amount of profit not earned due to the underutilization of a cloud service component utility is advocated, embracing the approach of unavoidable gaps in resource provisioning (i.e., unused service capacity) between ideal and actual adaptation decisions; these evident gaps (called debts) are recognised between resource supply and demand – as a match between these two is imperfect – and result from the partial usage waste after one machine is released but still charged until the end of the billing cycle. This interest is quantified as a side effect of the unused service capacity, SLO violation and lack of balanced, well-compromised trade-off decisions when auto-scaling in cloud. In this approach, the elasticity debt and profit optimization analytic methods are essential technical paradigms to solve the resource provisioning problem. The hypothesis is that the adaptation decision is driven by the resource capacity after applying value creation-driven techniques, determining when it is best to dynamically re-allocate resources from an overutilized composite service utility due to the workload fluctuations.

Another aspect distinguishing the work presented in this thesis compared to related research in cloud elasticity management is the adoption of game theoretic control mechanisms to mitigate the unavoidable utility gaps and automatically re-allocate resources from an overutilized service component utility that will directly influence the resource utilization and Quality of Service delivered by the systems. This game theoretic incentive scheme with debt-aware considerations for value creation is based on an equilibrium model and motivates the

optimal auto-scaling of resources. The control mechanisms optimally solve the composite service component utility overutilization problem by forming dynamic coalitions and optimizing trade-off decisions, while analysing the status of debt minimization and profit maximization and preserving the overall utility of the system or the application owner.

1.2.Research Problems and Solutions

This thesis tackles the research challenges associated with the debt-aware resource adaptation in cloud elasticity management for multi-tenant software as a service cloud environments. In particular, the following research problems and questions are investigated:

1. How the technical debt metaphor can be exploited to support the economics-driven resource adaptation decisions in cloud elasticity management with value- and strategy-oriented considerations.
2. How the unavoidable gaps in resource provisioning between ideal and actual adaptation decisions can be measured as an aspect of long-term value creation to mitigate the uncertainty of cloud environments.
3. How an analytic technique, i.e., game theory, can be leveraged to mitigate the unavoidable gaps in resource provisioning for multi-tenant software as service applications.
4. How debt-aware resource adaptations can consider the diversity of tenants and be used to form dynamic coalitions in multi-tenancy without affecting the overall utility of the system or the application owner.

To deal with the challenges associated with the above research problems, the following solutions are outlined:

1. Analyse and classify the economics-driven research in the area of cloud elasticity management to gain a systematic understanding of the underlying considerations in the existing mechanisms.
2. Analyse the gap of debt-aware strategies in cloud elasticity management and provide a theoretical analysis of the potential applicability of the technical debt metaphor in the context of cloud elasticity management and service composition.

3. Propose debt- and profit-aware models and algorithms that can be used in quantifying unavoidable gaps in resource provisioning between ideal and actual adaptation decisions.
4. Propose an approach to designing game theoretic control mechanisms with debt-aware considerations for re-allocation of resources from an overutilized service component utility, as part of the optimization of the resource utilization in multi-tenant software as a service cloud environments.
5. Design and implement a prototype tool that can be used for debt-aware quantification of the cloud service utility in multi-tenant software as a service cloud environments, integrating the developed models and algorithms.

1.3. Research Methodology

The research methodology developed and followed in this research project consists of the following sequential steps inspired by Peffers et al. [11]:

1. *Problem identification*. A survey is conducted guided by a literature review to gain an understanding of the cloud elasticity management research area and identify open challenges and issues.
2. *Research motivation*. Based on the survey findings, the elasticity-centric mechanisms in the cloud are classified in accordance with the underlying considerations during their development – i.e., quality-driven, cost-aware, energy-aware, intercloud-oriented and economics-driven – and a research gap is witnessed and spotted in the proposed economics-driven initiatives as far as it concerns value- and strategy-driven considerations in resource adaptation decisions in the cloud.
3. *Solution definition*. The objective of developing an economics-driven elasticity management framework in the cloud that supports value-driven adaptation decisions motivates the need for a theoretical analysis to investigate the potential applicability of the technical debt metaphor in the context of cloud elasticity management and service composition. One of the aspects of dynamic resource adaptation is that due to changes on the request workloads, the composition encounters underutilization or

overutilization on the composite service components. In the underutilization state, the pay-off is significant in the long run with far-sighted benefits, while in the overutilization state, there is no pay-off and, if exists, it is trivial. Such a definition matches the intentional and unintentional technical debt encountered in the software engineering research field. In this sense, the technical debt metaphor is applied in the context of cloud elasticity management as the disruptive solution concept for resource adaptations with value creation considerations.

4. *Model and algorithm design and development*. The unavoidable gaps in resource provisioning and utilization (i.e., unused service capacity) and conflicting trade-offs between ideal and actual adaptation decisions are quantified in the context of multi-tenant software as a service (SaaS) cloud environments. Control mechanisms are designed exploiting game theory to enable auto-scaling in multi-tenant SaaS applications and optimize resource utilization.
5. *Hypothesis testing and evaluation*. Quantitative evaluation and experimentation with diverse scenarios are performed to assess the efficiency of debt-aware models and algorithms, and validate the theorem based on the game theoretic incentive scheme.
6. *Tool development and testing*. A prototype, debt-aware quantification tool is implemented that provides quantitative reports, dashboards and insights into the underutilization / overutilization states of service components in multi-tenant cloud-supported services.

A flowchart with the research methodology processes that are followed in this research thesis and described above is shown in Figure 1.1.

UNIVERSITY of NICOSIA

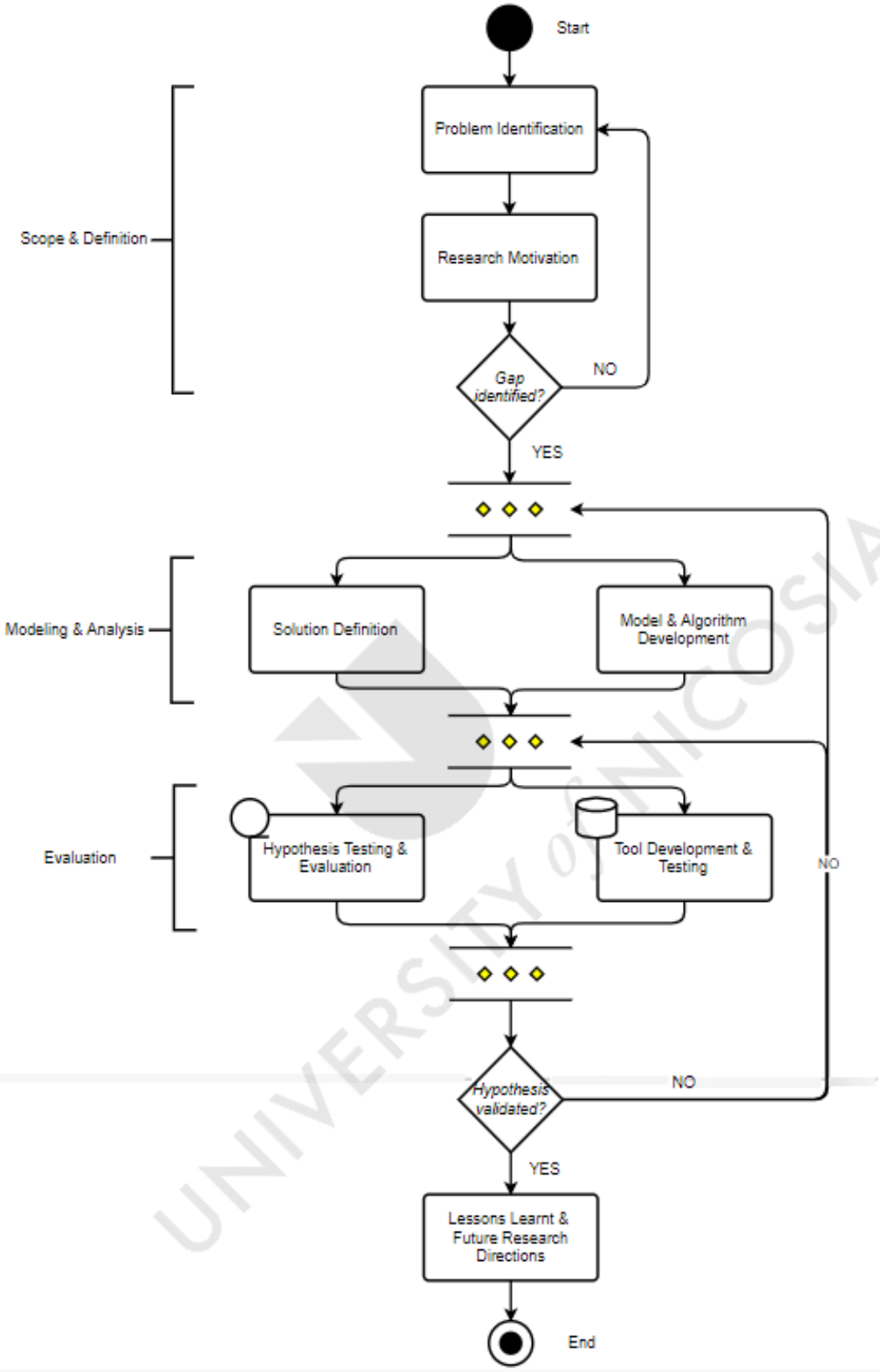


Figure 1.1. Research methodology process flowchart

1.4.Research Contributions

The novel contributions and impact of this research thesis on the theory of cloud elasticity management can be divided into the following five categories:

1. Classification and analysis of the cloud elasticity management research area.
2. Gap analysis towards an economics-driven approach to support value creation in cloud elasticity management.
3. Novel debt-aware models and algorithms with value creation and strategy considerations to reason about elasticity resource adaptations in the context of cloud resource provisioning.
4. Novel game theoretic control mechanisms with debt-aware elasticity considerations for optimal resource utilization and management in multi-tenant software as a service cloud environments.
5. Implementation of a prototype tool for debt-aware quantification at cloud service utility level in multi-tenant applications hosted in the cloud.

To further elaborate, the key contributions of this research project are:

1. A taxonomy and survey of the state of the art in cloud elasticity management, reviewing also the multi-tenancy and network virtualization levels.
2. Gap analysis towards an economics-driven approach to support value creation in cloud elasticity management:
    a. A gap analysis of debt-aware strategies in cloud elasticity management.
    b. A theoretical analysis towards an innovative methodology and framework in cloud elasticity management and service composition leveraging the technical debt metaphor to effectively solve the resource provisioning and utilization research problems in cloud and mobile cloud computing (MCC) systems (for the latter, in case of cloud-supported mobile applications that facilitate decisions for computational offloading of user tasks to the cloud).
    c. A cost estimation model for software as a service implementation in the cloud, exploiting the COCOMO estimations and applying a tolerance value for prediction.
    d. A preliminary debt estimation model from the multi-tenant software as a service application viewpoint in the context of cloud service utility, with the estimation being subject to the service capacity.

UNIVERSITY of NICOSIA

e. Evaluation and analysis of the cost and debt estimation models.

3. Novel debt-aware models and algorithms with value creation and strategy considerations to reason about elasticity resource adaptations in the context of cloud resource provisioning:
   a. The introduction of a debt-aware research approach to dynamic resource adaptation after mapping the technical debt metaphor in the context of cloud elasticity management under uncertainty.
   b. Novel cloud-inspired quantitative models and algorithms based on the elasticity debt and profit optimization analytic methods that measure the amount of profit not earned due to the underutilization of cloud resources, embracing the approach of unavoidable gaps in resource provisioning between ideal and actual adaptation decisions (i.e., overutilization or underutilization problem of the composite service component utility due to the dynamic fluctuations in request workloads and the gaps witnessed between resource supply and demand). This interest is quantified as a side effect of the unused service capacity, service level objective (SLO) violation and lack of effective, well-compromised trade-off decisions when auto-scaling in cloud.
   c. Experimental evaluation and analysis of the debt-aware models and algorithms for scenarios with different cloud service capabilities and storage capacities in the context of multi-tenant software as a service.
4. Novel game theoretic control mechanisms with debt-aware elasticity considerations for optimal resource utilization and management in multi-tenant software as a service cloud environments:
   a. The introduction of a game theoretic incentive scheme for optimal resource provisioning with debt-aware considerations, formation of dynamic coalitions and optimization of trade-off decisions.
   b. Novel control mechanisms that optimally solve the problem of composite service utility overutilization detection as a part of the dynamic resource adaptations in cloud based on an equilibrium model, motivating the optimal auto-scaling and exchange of resources in multi-tenant software as a service applications. The exchange of resource capacity among tenants based on their debt status boosts the fairness between consumers and providers in cloud elasticity management.

UNIVERSITY of NICOSIA

c. Analysis of the utility-driven state of debt minimization and profit maximization from the service provider viewpoint based on a novel Hidden Markov Model in the context of cloud resource scheduling.
d. A novel mobile opportunistic offloading model in the mobile cloud computing framework for task offloading on the cloud after denial by two mobile device nodes in close proximity due to resource restrictions, considering the computation processing time, energy of mobile device user for offloading the input data and overhead of cloud computing in terms of processing time and energy.
e. Experimental evaluation and analysis of the theorem for scenarios with different cloud service capabilities and storage capacities in the context of multi-tenant software as a service.

5. Implementation of a prototype tool for debt-aware quantification at cloud service utility level in multi-tenant applications hosted in the cloud:
   a. The integration of the models and algorithms designed and evaluated in the previous chapters.
   b. The web application implementation in Java – targeted to be deployed in the Google Cloud Platform supported by the Google App Engine – with an analysis on the architecture, design, requirements, technologies, product evaluation and testing plan.
   c. Provision of computations about different case scenarios and insights into the:
      i. Underutilization / overutilization of service components in multi-tenant cloud-supported applications, the gradual payoff of the elasticity debt and the threshold point at which it will be totally cleared out.
      ii. Cost estimations for software as a service implementation in the cloud exploiting the Constructive Cost Model (COCOMO).

1.5. Thesis Organization

The core chapters of this doctoral thesis are structured as shown in Figure 1.2. Finally, the remainder of the doctoral dissertation is organized as follows: Chapter 2 presents a taxonomy and survey of elasticity-centric design approaches in the context of cloud resource

management, as well as the scope of this research and development project and its positioning within the research field. This chapter is derived from [12]–[14]. Chapter 3 presents the research motivation to estimate debts in cloud elasticity management and a gap analysis of disruptive debt-aware strategies in the context of cloud service composition under multi-tenant software as a service applications. This chapter is derived from [15], [16]. Chapter 4 proposes novel debt-aware quantitative models, techniques and algorithms for efficient resource adaptation in cloud resource provisioning. This chapter is derived from [9], [17]–[20]. Chapter 5 proposes novel game theoretic control mechanisms for optimal resource utilization in cloud and mobile cloud computing environments, adopting a multi-tenant software as a service approach. This chapter is derived from [21]–[25]. Chapter 6 describes the architecture, technologies and implementation of the prototype tool, a novel framework for debt-aware quantification in cloud computing environments. Chapter 7 concludes the thesis with a summary of the main findings, discussion of future research directions and final remarks.

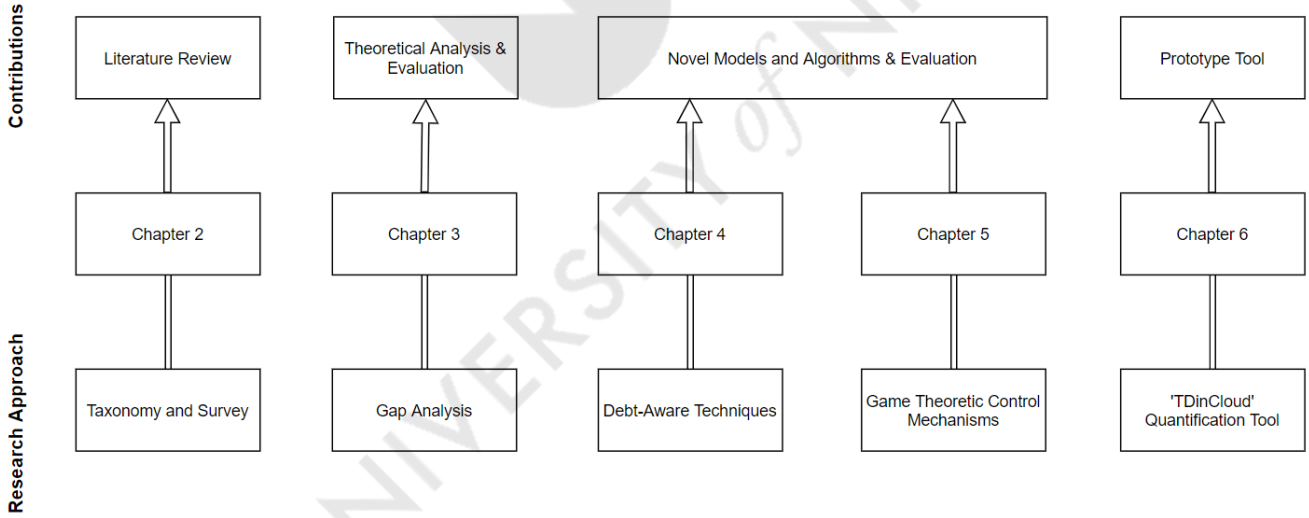


Figure 1.2. Ph.D. thesis organization

UNIVERSITY of NICOSIA

Chapter 2

Taxonomy and Survey of Elasticity-Centric Design in Cloud Resource Management

Cloud elasticity enables the dynamic acquisition and release of shared computational resources on demand towards optimal adaptation decisions to a dynamically changing environment. This characteristic facilitates the efficient adjustment of resource provisioning constrained by QoS and operating costs as the resource requests are diverse and fluctuate dynamically. To identify open challenges and research gaps in the area and facilitate further advancements, it is essential to classify the research on elasticity management in cloud computing environments. This chapter discusses the problems of existing elasticity implementations in cloud-centric systems and presents a taxonomy of elasticity-centric mechanism design in cloud resource management, reviewing also the multi-tenancy and network virtualization levels. The key, economics-driven works in the area are surveyed and mapped onto the taxonomy to guide future design and development efforts. The chapter concludes with a discussion of the scope of this research project and its positioning within the research area.

2.1.Introduction

Cloud computing is a well-consolidated paradigm for on-demand resources and services provisioning on a pay as you go basis [26] and elasticity is a fundamental property that enables to maintain, grow or shrink resources real time in order to handle the dynamic load variation [7]. Elasticity facilitates the advantages of economies of scale [4] in the cloud context, contributing to a decrease in the average cost of the computing and storage resources, and has recently witnessed major developments [27]. In this sense, several research efforts have been devoted to examining economics-related aspects of cloud, i.e., from comparing the monetary cost-benefits of cloud computing with desktop grids [28] and investigating cost-benefit perspectives of using cloud in order to extend the capacity of clusters [29] to measuring the total cost of ownership and utilization cost [30].

In addition, the objective of the admission control and resource scheduling algorithms is to manage the resources effectively while ensuring the Quality of Service requirements of aggregate requests, adhering to the Service Level Agreement guarantees and improving the resource providers' competitiveness and profitability [31]. From a multi-tiered resource scheduling perspective, efficient resource allocation in multi-application virtualized data centres is imperative towards improving the resource utilization as the resources are allocated to applications proportionally according to the application priorities [32]. For example, cloud-supported mobile applications that facilitate decisions for computational offloading of user tasks to the cloud [33] or significant features of workflow systems – such as multi-tenancy for a wide range of analytics tools and back-end data sources, user group customization and web collaboration [34] – can be deployed using several VMs instantiated on different physical nodes. As far as it concerns the resource management decisions in the cloud, in cases of limited resources, the performance of a low-priority application is intentionally degraded and the resources are allocated to critical applications [35], [36]. In [32], resource scheduling at three levels has been proposed: (i) the application-level scheduler dispatches requests across the application's VMs; (ii) the local level scheduler allocates resources to VMs running on a physical node according to their priorities; and (iii) the global-level scheduler controls the resource flow between applications.

The objective of this chapter is to give an overview of the recent research advancements in the area of cloud elasticity management – an intrinsic factor for efficient resource management and provisioning. In this context, the key contributions of this chapter are the following:

1. A taxonomy of elasticity-centric mechanism design approaches and classification for the underlying awareness that drives the adaptation decisions.
2. A set of problems and challenges concerning the elasticity management research from an economics-driven perspective.
3. A survey on related economics-driven research works, covering also the multi-tenancy and network virtualization levels.
4. Position the research work presented in this thesis within the research field.

The reminder of this chapter is organized as follows: in the next section, the classification of the existing elasticity-centric design approaches is presented. Section 2.3 discusses the problems of the elasticity-centric mechanisms. Section 2.4 presents a literature review of the research in cloud elasticity management context, covering also the network virtualization level in Section 2.5. The chapter is concluded with the scope and positioning of this research project within the research area in Section 2.6, followed by a summary in Section 2.7.

2.2. Classification of Elasticity-Centric Design

Elasticity is an intrinsic characteristic of cloud computing for effective provisioning of virtual resources to the dynamic fluctuations for computing capacity to satisfy the resource demand [37]. However, its management is imperative for coordinating the resource-related decisions based on economic incentives and maximizing the value of resource adaptations. Some classifications on elasticity-centric mechanisms have been proposed in the literature [38]–[40], but still an autonomous, economics-oriented resource provisioning driven by the resource demand on time is a challenge [7]. Elasticity-centric design is a set of processes that their objective is to evaluate the dynamic trade-offs in elasticity-driven resource adaptation decisions [41]. The related works for such schemes are reviewed towards the classification of the existing elasticity-centric design approaches. To classify these approaches, the following types of design-inspired perspectives are identified according to the findings from the literature review:

1. *Quality-driven design*: A focus is given on the Quality of Service (QoS) parameters (e.g., response time, throughput, latency) or other performance metrics (e.g., utilization level).
2. *Cost-aware design*: A focus is given on the explicit considerations (direct indicators rather than proxies) for the consumption prices of virtual resources and penalties caused by Service Level Objective (SLO) violations.
3. *Energy-aware design*: A focus is given on the energy consumption, carbon footprint and CO_2 emissions.
4. *Intercloud-oriented design*: A focus is given on solutions associated with federated clouds or multi-cloud [42].
5. *Economics-driven design*: A focus is given on economics-oriented parameters, i.e., value creation, profitability, strategic decisions (e.g., proactive adaptations, long-term objectives), uncertainty awareness (e.g., probability-based methods) and trade-offs [43].

Based on the above findings, Figure 2.1 presents the high-level taxonomy of the elasticity-centric design approaches and Table 2.1 provides the classification of the related research works that were surveyed in each category.

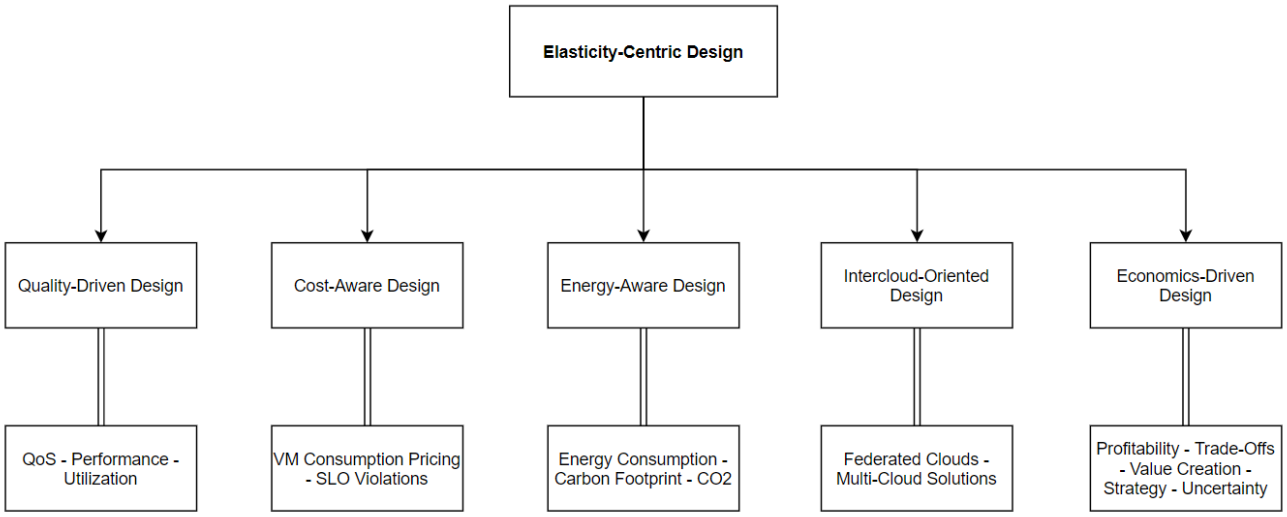


Figure 2.1. High-level taxonomy of elasticity-centric design

Table 2.1. Classification of elasticity-centric design

Elasticity-Centric Design	**Related Research Works**
Quality-Driven Design	[44], [45], [46], [47], [37], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68]
Cost-Aware Design	[69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96]
Energy-Aware Design	[97], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109]
Intercloud-Oriented Design	[110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121]
Economics-Driven Design	[27], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135]

UNIVERSITY of NICOSIA

Quality-Driven Design

In quality-driven design, the elasticity adaptation decisions are taken with respect to guaranteeing the expected SLOs and avoiding potential violations. These mechanisms do not consider cost-related aspects, such as pricing or billing, but mainly performance-related metrics of the virtual resources, such as CPU utilization, memory usage and bandwidth. Therefore, parameters like the response time, throughput and latency are monitored as part of the meeting the SLO expectations. Some related works studied design approaches using custom metrics coupled with performance parameters for over- and under-provisioning states [44], [56].

Cost-Aware Design

In cost-aware design, the elasticity adaptation decisions are taken with respect to cost implications while avoiding SLO violations. These mechanisms consider the cost minimization problem as a multidimensional issue, in which there is a non-linear relation between prices and computing resource granularity. The cost-related aspects include budget constraints, pricing schemes of virtual resources, billing cycles or penalties caused by non-adherence to expected SLOs. The majority of the related works considered monetising penalties or incorporating them in the trade-off analysis of elasticity adaptations by means of utility functions (i.e., trade-off between cost minimisation and quality conformance). These schemes made runtime decisions that pursued immediate rewards and cost minimisation at task scheduling level or when increasing, maintaining or reducing the resource provisioning. In particular, they made visible the trade-off between quality of service and operating costs.

Energy-Aware Design

In energy-aware design, the elasticity adaptation decisions aim to lower the power consumption of physical nodes that allocate the running virtual machines. These mechanisms are built on the fact that energy consumption has a non-linear relationship with cost and SLOs enforcement [136]. Although this kind of mechanisms focus on minimising costs while enforcing SLO adherence, similar to cost-aware design, their cost savings focus on reducing

energy consumption. This reduction may produce further savings in terms of taxes related to carbon footprint. Some of these design approaches consider aspects such as VM live migration, in which a VM migrates from one node to another either to search for additional capacity or consolidate VMs in a lower number of physical nodes. Other related works examined the reduction of power dissipation through a dynamic scaling of CPU frequency and voltage prior to switching nodes off [137], [138].

Intercloud-Oriented Design

In intercloud-oriented design, the elasticity adaptation decisions are made across different clouds. An intercloud is either a federation, in which cloud providers interconnect their infrastructures to exchange resources, or a multi-cloud, in which a cloud service assumes the responsibility to manage its resource provisioning across an aggregation of multiple clouds [2]. Quality of service [113], operating costs [115] and energy consumption [100] are still parameters to be evaluated in this design approach, besides guaranteeing the functional operation among clouds.

Economics-Driven Design

In economics-driven design, the elasticity adaptation strategies considered the uncertainty in the cloud as an opportunity to create value if certain conditions materialise, or trading off the immediate against far-sighted benefits of an elasticity adaptation. An adaptation decision is considered to create value if:

1. It takes advantage of opportunities or counterbalances threats in an environment.
2. It enables a player to boost the satisfaction of their needs.
3. It enables a player to devise strategies that enhance efficiency and effectiveness.

Strategy and economics are intrinsically intertwined for achieving long-term objectives and optimal use of resources [43], performing a holistic strategic analysis (e.g., cost efficiency, cost effectiveness) in which an adaptation strategy is deemed as an investment that may even discard immediate rewards to pursue its far-sighted benefits. Throughout the literature review, the adopted economics-oriented criteria are summarized as follows:

1. Economics-inspired frameworks [27], [123], [124].
2. Adaptation strategies deemed as investments with long-term rewards under uncertainty [122], [133].
3. Pricing techniques and yield management for profit maximisation [107], [131].
4. Analysis of the cost efficiency and cost effectiveness of adaptation decisions [129], [130].

2.3. Problems of Elasticity-Centric Mechanisms in Cloud Computing

In this section, the challenges and issues in cloud elasticity management research are summarised focusing on the economics-driven viewpoint. In this context, the key problems of the proposed elasticity-centric mechanisms are the following:

1. *Lack of debt-aware considerations in runtime resource adaptations*. There is little research focusing on economics-driven frameworks to make explicit value creation considerations when reasoning about elasticity adaptations in cloud [124].
2. *Lack of opportunity cost-aware considerations in runtime resource adaptations*. Related research works tend to estimate the value of elasticity adaptations through utility functions capturing the uncertainty partially [27], [123], [127]. However, considering retrospective valuation mechanisms in runtime adaptation decisions – in the sense of opportunity cost-aware mechanisms [139] – can be proved useful as some decisions might not be the appropriate in the long run due to the context of the scenario at the time of the analysis.
3. *Lack of incentive schemes for strategy-driven resource adaptations*. Existing research efforts have proposed economics-driven elasticity management schemes with incentives to low prices [27] or disincentives to minimize SLOs violations [94]. However, these works do not consider strategy-driven mechanisms that are able to monitor the trade-off decisions between conflicting objectives, i.e., immediate versus far-sighted rewards or individual versus global benefits, with an aim to maintain the interoperability of cloud elasticity management.
4. *Lack of utility-driven considerations in the context of unavoidable gaps in resource provisioning and utilization (i.e., unused service capacity)*. In existing research efforts

[27], the potential utility of unavoidable gaps in resource utilization has been neglected, increasing the effects of uncertainty and conflicting trade-offs at runtime.

5. *Lack of schemes with dynamic coalition considerations in multi-tenancy*. Existing research efforts do not consider widely the diversity of tenants within clusters to enable them re-adjust their initial SLOs based on the arbitrary workflows and workloads due to the unpredicted variations in the resource demand. Forming dynamic coalitions can enable the dynamic sharing of virtual resources and preserve the diversity of the tenants' SLOs at the same time.
6. *Lack of control-oriented incentive schemes driven by game theoretic techniques to mitigate unavoidable gaps in resource provisioning in the context of multi-tenancy*.
7. *Lack of energy-aware considerations in the economics-driven analysis*. The energy consumption of cloud data centres is constantly raising alongside the increasing service demand [140]. In this context, it is imperative to consider energy-aware decisions in cloud elasticity management measuring the impact of energy consumption when provisioning virtual resources [141]. In particular, a research gap is witnessed in the quantification of the value extracted from the waste of energy hidden in elasticity adaptations by estimating the trade-offs between energy savings and the latency in adaptation decisions [142].
8. *Lack of sensitivity analysis aspects in runtime resource adaptations*. There is little research that considers the usefulness of sensitivity analysis [143] when dealing with uncertainty in resource provisioning due to the workload variation or the SLOs re-adjustments [56], [88], [144].
9. *Lack of a knowledge base for value creation in resource adaptations*. Cloud elasticity management allows the monitoring, analysis, scheduling and execution of resource adaptations in real time. The knowledge that can be extracted from these jobs at runtime is essential to be used in order to create value. Limited research efforts have considered the complexity of modeling knowledge from previous elasticity decisions as a reasoning tool, so that future adaptation decisions with strict time constraints can benefit from previous knowledge and increase the accuracy of elasticity in the context of cloud resource provisioning [145], [146].

UNIVERSITY of NICOSIA

2.4.State of the Art in Cloud Elasticity Management

In this section, closely related research in cloud elasticity management level is surveyed. To achieve the desired economics and elasticity gains, the authors in [134] provided a detailed analysis of the economics and elasticity challenges of business applications targeted to be deployed and operate on public cloud infrastructure. The analysis considered aspects, such as public cloud infrastructure service offerings (CISOs), modeling of CISOs economics and elasticity, application workload patterns and its impact on achieving elasticity and economics, economics-driven elasticity decisions and policies, and SLA-driven monitoring and elasticity of cloud-based applications. This research work motivates cloud consumers and business application owners to be more elasticity-aware by analysing important economics and elasticity capabilities of different CISOs subject to meeting the requirements. Najjar et al. [147] presented a survey of the cloud elasticity concept and proposed a classification of the applied mechanisms and techniques that exist in the literature to manage elasticity (i.e., cost- and goal-aware). A resource elasticity analytical model for cloud resource management has been considered imperative in [148]. This work introduces the Controlling Elasticity (ControCity) framework for controlling cloud resources through buffer and elasticity management. Two essential components – a buffer manager and elasticity manager – are provided in the application and middleware layers, respectively. The former controls the queue of the input user requests, while the latter the elasticity of the cloud platform using learning automata techniques. The experimental results indicated that the response time is reduced up to 3.7%, while increasing the resource utilization and elasticity up to 8.4% and 5.4%, respectively, compared to existing approaches.

To optimize the QoS and cost objectives for the cloud-based services, the challenges lie in the fact that these objectives can be conflicted, e.g., throughput and cost, while the QoS requirements often interfere due to the shared cloud infrastructure. Chen and Bahsoon [78] proposed a self-adaptive, dynamic decision-making approach for auto-scaling in the cloud with well-compromised trade-offs. The authors leveraged on ant colony-inspired multi-objective optimization for searching and optimizing the trade-offs decisions and the result is then filtered by compromise-dominance, a mechanism that extracts the decisions with balanced improvements in the trade-offs. The work in [149] examined admission control and resource scheduling algorithms to maximize profits while elastically providing cloud resources to execute queries guaranteeing service level agreements across a range of Quality of Service

requirements. To enable timely responses, the algorithms utilize data splitting-based query admission and resource scheduling offering parallel processing on split datasets. The experimental results have shown that the proposed algorithms perform significantly better in terms of increasing query admission rates and creating higher profits, while supporting efficient resource configurations. Then, Fokaefs et al. [27], [135] presented an economics-driven management approach analysing the value of scaling resources in the cloud to decide adaptations; nonetheless, it was purely reactive and only considered costs in the decision valuation to adapt the resource provisioning, excluding quality considerations in terms of penalties due to SLA violations.

Mera-Gómez et al. [122], [124] discussed the challenge of achieving the perfect match between resource demand and provision in autonomous elasticity management. To address this problem, a technical debt-aware learning approach has been proposed based on reinforcement learning of elasticity debts in resource provisioning as the adaptation pursues strategic decisions that trades off economics against performance. The authors extended CloudSim [150] and Burlap [151] to evaluate the approach, showing that a reinforcement learning of debts in elasticity obtains a higher utility for a cloud customer, while conforming expected levels of performance. To the best of our knowledge, in contrast to previous efforts, the research work presented in this thesis contributes to positioning an economics-driven management process as a new perspective to guide the implementation of elasticity solutions with debt-aware value and strategy considerations.

Alzaghoul and Bahsoon [126] motivated the need for managing the technical debt on cloud-based service selection using real options; in this work, it is stated that the web service selection decision may incur a technical debt that is necessary to be managed. The debt of substitution decisions is examined in support for scaling up scenarios; a debt that is necessary to be managed, cleared out and transformed to value-added. An option-based approach is proposed to inform the selection of candidate web services with varying costs, utilities and system value along with a quantification approach for measuring the extent to which the debt is cleared out and provide future options. An economics-driven viewpoint is also adopted in [125] from a cloud-based architecture perspective, where the problem of web service substitution and the debt valuation is formulated as an option problem. Two types of options are considered – i.e., option to switch between services and option to defer the decision of substitution – and an options analysis approach is used to manage and clear the technical debt.

To find the right balance for trade-offs between timeliness and solution optimality, Pandey et al. [127] proposed a hybrid planning approach to support decision-making in self-adaptive

systems by means of a utility-driven decision valuation and combining more than one planner to obtain the benefits of each. A hybrid planner is triggered through deterministic planning with Markov Decision Process (MDP) planning to obtain: (i) quick provision of plans when timeliness is critical, and (ii) generation of optimal plans when the system has sufficient time to perform this action. The authors validated this hybrid planning approach using a realistic workload pattern in a simulated cloud-based self-adaptive system. Elaboration on the evaluation of the debt in cloud-based architectures using real options is also presented in additional related works [152]–[154]. Narrowing in, the option to defer the decision of substitution under uncertainty is investigated, exploiting binomial options to the formulation. The time-value of the architecture decisions of switching web services and the debt that they imply on the structure are quantified, while the proposed method builds on Design Structure Matrix (DSM) and introduces time- and complexity-aware propagation cost metrics to assess the value of deferral decisions relative to changes in the structure. Additionally, Nallur and Bahsoon [155] suggested a marketplace that allows applications to select services in a decentralized manner. In this work, a cloud-based multi-agent system is presented that uses multiple double auctions to enable applications to self-adapt based on their Quality of Service (QoS) requirements and budgetary constraints.

Regarding the existing classifications for elasticity mechanisms in the literature, Galante and De Bona [38] presented a comprehensive study with the related work in the area in order to define a classification based on scope, policy, purpose and method while discussing open issues and challenges. Likewise, Coutinho et al. [39] addressed different aspects of elasticity, such as definitions, metrics and tools for measuring and evaluation of the elasticity, and proposed a classification based on policy and method. Lorido-Botran et al. [8] discussed the challenge of identifying effectively the amount of resources to lease in order to meet the required Service Level Agreement (SLA), while keeping the overall cost low. The authors proposed a classification into five categories: static threshold-based rules, reinforcement learning, control theory, queuing theory and time series analysis. Furthermore, Al-Dhuraibi et al. [40] provided a detailed analysis of elasticity mechanisms addressing also the elasticity of containers, a technological trend in lightweight virtualization. The authors elaborated on a taxonomy based on characteristics, such as configuration, scope, purpose, mode, method, provider and architecture.

Existing research has also proposed economics-driven elasticity management mechanism design for multi-tenant and intercloud-oriented environments [156]. The challenge here has to do with the limitations of the multi-tenant cloud software as a service applications to the

diversity of tenants by clustering them in a few categories (e.g., premium, standard) with pre-defined workflows, workloads and service level objectives (SLOs). This coarse-grained clustering problem reduces the advantage of these multi-tenant ecosystems over single tenant architectures to share dynamically virtual resources between tenants based on their own workload profile and elasticity adaptation decisions. To address this issue, Mera-Gómez et al. [123] proposed a debt-aware multi-agent elasticity management where each tenant is represented by a reinforcement learning agent that performs elasticity adaptations to form autonomous coalitions that minimise the effect of the unavoidable imperfections in any elasticity management approach. The simulation results indicated that this approach preserved the diversity of tenants and reduced SLO violations without affecting the aggregate utility of the application owner.

Elastic resource provisioning is a critical element in the context of multi-tenant software as a service applications, leveraging sharing and economies of scale to provide cost efficient hosting while guaranteeing the service level objectives [157]. However, performance interference in the hosting platform introduces uncertainty in the performance guarantees of provisioned services. The work in [158] discussed software architectural concerns for implementing multi-tenant applications by pointing out characterizing features and differentiating aspects from related concepts, such as system virtualization. Rameshan et al. [159] mentioned that the existing elasticity controllers are unaware of the performance interference in hosting platforms or overprovision resources to meet the SLOs. In this effort, the elasticity controller Hubbub-Scale is proposed, while modeling interference to control the different sources of unpredictability in the performance of VMs. The evaluation results with Redis and Memcached have shown that the proposed controller conforms to the SLO requirements under scenarios where the standard modeling approaches fail. Kumar et al. [160] proposed an approach for identifying the technical debt involved in service composition under cloud software as a service, combining the ARFIMA time series forecasting method [161] and a model to estimate the future debt and utility in the service composition. The approach is evaluated through a real-world case study, demonstrating successful identification of both good and bad debts, while producing satisfactory accuracy on estimations. In [162], a multi-tenant middleware is proposed for dynamic service composition in cloud software as a service. The authors presented encoding representation and fitness functions that model the service selection and composition as an evolutionary search.

In the same direction, Domingo et al. [6] presented Cloudio, a cloud-based metadata-powered multi-tenant architecture backed with a proof of concept J2EE implementation. On

the contrary, Povedano-Molina et al. [5] proposed DARGOS, a distributed cloud monitoring architecture to disseminate monitoring information for physical and virtual resources in the cloud, while keeping a low overhead. The proposed monitoring architecture has been integrated into a real cloud deployment based on OpenStack [163]. The experimental results have shown that DARGOS introduced a low and scalable monitoring overhead. Finally, Mudigonda et al. [164] elaborated on the multi-tenant network architecture NetLord that exploited commodity equipment to scale the network to several thousands of tenants and millions of virtual machines. The authors implemented NetLord on a testbed and demonstrated its scalability, while achieving order-of-magnitude goodput improvements over existing approaches.

2.5. Virtualization Level

2.5.1. Network Functions Virtualization

Cloud providers of infrastructure as a service (IaaS) need data center networks that support multi-tenancy, scale and ease of operation at low cost. Most existing network architectures cannot meet all of these needs simultaneously. A number of research efforts has been devoted considering the issues of cloud computing associated with Network Functions (NFs) [165]. The emergence of Network Functions Virtualization (NFV) enabled the transformation of network architectures by implementing Network Functions in software that can run on commodity hardware and, thus, placing them upon generic computing resources [166]. Authors in [167] elaborated on an architecture that can insert such network functions dynamically in high performance cloud computing environments. The network functions are deployed on a virtual machine (VM) and the network nodes distributed in the network transfer packets matched to the pre-configured policy to an arbitrary network function. Consequently, packets can traverse across network functions distributed over multiple data centres. Additionally, network functions are provided in an arbitrary sequence among VMs and user clients. This allows users to create network design with network functions in cloud computing environments. Another research work [168] examined two resource allocation algorithms for virtualized network functions based on genetic algorithms for the initial placement of VNFs and the scaling of VNFs to support traffic changes. The authors compared the performance of the proposed algorithms with a traditional integer linear programming resource allocation technique,

combining data from previous empirical analyses to generate realistic VNF chains and traffic patterns. A dynamic resource provisioning algorithm for VNFs to utilize both edge and cloud resources is proposed in [169]. Adapting to dynamically changing network volumes, the algorithm automatically allocates resources in both the edge and the cloud for VNFs. The algorithm considers the latency requirement of different applications in the service function chain, which allows the latency-sensitive applications to reduce the end-to-end network delay by utilizing edge resources over the cloud. The results show that the proposed algorithm reduces the end-to-end response time by processing 77.9% more packets in the edge nodes compared to the application non-aware algorithm.

The design of the Cloud4NFV management and orchestration platform has been discussed in [170] for streamlining Network Functions as a Service (NFaaS) over virtualized network / IT infrastructures, showing that part of its foundations lay on cloud infrastructure management and Software Defined Networking (SDN) platforms. Das et al. [171] introduced FlowComb, a network management framework that helps big data processing applications achieve high utilization with low data processing times. FlowComb uses software agents installed on application servers to predict application network transfers even before they start. A centralized decision engine collects data movement information from the agents and schedules upcoming flows on paths such that the network does not become congested. The authors in [172] proposed the Network-as-a-Service (NaaS) framework that integrates current cloud computing offerings with direct and secure tenant access to the network infrastructure. Tenants can easily deploy custom routing and multicast protocols and, by modifying the content of packets on-path, they can efficiently implement advanced network services, such as in-network data aggregation, redundancy elimination and smart caching. The simulation results have shown that, even with limited processing capability at network switches, NaaS can significantly increase application throughput and reduce network traffic. The work in [173] described how NFV was applied to a virtualized network function, the Session Border Controller, and its integration into an SDN-based network. Finally, CloudNaaS [174] is a cloud networking system where end-users are able to deploy applications augmented with a rich and extensible set of network functions, such as virtual network isolation, custom addressing, service differentiation and flexible interposition of various middleboxes. CloudNaaS primitives are directly implemented within the cloud infrastructure using high-speed programmable network elements, making CloudNaaS highly efficient.

UNIVERSITY of NICOSIA

2.5.1.1. System Architecture of NFV Marketplace

To facilitate the involvement of diverse actors in the NFV scene and attract new market entrants, the authors in [175] proposed the design of a novel NFV marketplace where network services and functions can be published, brokered and traded through a brokerage module. In this research work, it is mentioned that the end-users are able to browse and select the services and virtual appliances that best match their requirements, as well as negotiate the associated SLAs and be charged under various billing models. From an architecture level viewpoint, three core functionalities are identified in the main building blocks as follows:

1. *Resources publication and NF advertisement*. Through a customer front-end client, infrastructure providers announce their available assets. In the sequel, third-party NF developers declare their functions that may be hosted by the Function Store and, finally, customers place their requests for services and virtual appliances.
2. *NF discovery, resource trading and service matching*. Through a brokerage module, customers can place their requests for services and declare the requirements for the corresponding NFs, receive offerings and make the appropriate selections taking into account the offered SLAs. Trading and billing policies, such as long-term lease, scheduled lease, short-term lease or spot markets, are based either on fixed-price or action-based strategies.
3. *Customer-side monitoring and configuration of the offered services and functions*. Through a service dashboard, end-users are able to interact with the orchestrator platform for monitoring of the status of established services and associated NFs, as well as for performing management operations based on the granted rights.

End-user requests provided the customer front-end client are received by a brokerage module, which performs the following jobs:

1. Analyse the requirements.
2. Match the analysis results with the available network resources maintained by the orchestrator and the functions aggregated at the Function Store.
3. Initiate a bargain process for all valid solutions under various merchandise policies and expected SLAs.

Upon successful SLA establishment and resource trading, the orchestrator establishes the virtualized infrastructure with the associated NFs, maintaining the control, customization and administration. In this context, the proposed marketplace acts as an intermediary party among

all actors (i.e., service provider, third party NF developers and customers) by utilizing appropriate optimization algorithms for service establishment as a matter of technical and economic requirements. Prior to any resource allocation, the economics of such transactions are analysed and an optimized solution is obtained, enabling third party NF developers to assign the available NFs under several trading policies. Such a trading framework allows players to deal directly for network functions, thereby establishing a secondary market for network resource leasing or resource auction. This model has the potential to enable small companies and individuals to enter the networking market and trade their virtualized appliances under several pricing schemes, thus boosting competition and innovation. Additionally, third party NFs can be included in the Function Store and rapidly introduced to the market, avoiding the delay and risk of hardware integration and prototyping.

2.5.1.2. Trading Policies

Resource pricing schemes are classified as pay as you go, subscription and spot market [131]. Pay as you go consists in a fixed price per billing cycle. A customer needs to analyse a pricing scheme depending on the workload pattern and volume [134] and a provider considers maintenance costs, such as those of starting or shutting down a VM and additional overheads related to a fine-grained pricing. On the other hand, subscription tends to be deterministic as far as price is concerned [176]. Subscription is not elastic though as customers subscribe beforehand for resources for a definite period of time and the model might not be optimal if the level of demand fluctuates, which is often the case on multi-tenant cloud environments. Finally, in spot market, the users can bid for resources and get available resources when their offers are higher than the spot price [177]. In this context, the exploited trading policies of the brokerage module are analysed in this section in order to lease virtual functions or resources to end-users and provide related costs. The below models are adopted for the provisioning of peak demands:

1. *Pay as You Go Model*: Service providers pay for the aggregate sum of bytes exchanged exploiting the pay as you go model of the cloud [178]. It is a utility-driven charging model executed in distributed computing and intended for enterprises and end clients. Each client is charged for secured – instead of real – registering assets inferred from the utility processing, enabling to scale, tweak and procure registering assets. Asset

charges are dependent upon utilized administrations in a whole base. Software as a Service (SaaS) works in a similar way, where clients lease software and custom functions. Storage as a Service (StaaS) billing is rotated on a regular basis, because the storage requirements are subject to the gradually increased rates [179].

2. *Resource Provisioning for Peak Demand*: Video-on-demand (VoD) suppliers gain a month-to-month plan from ISPs in which a settled data transmission limit, e.g., 1 Gbps, is ensured to guarantee the expected requests. In distributed computing [180], processing assets are versatile and scale effortlessly by the cloud administration supplier when and wherever needed, relying on the underlying asset necessities on runtime without actually upsetting the operations [181].
3. *Auto-Scaling Model*: Auto-scaling is an intrinsic characteristic of distributed cloud computing that enables clients to automatically re-allocate the amount of computational resources in a server farm based on the load [8].

2.5.1.3. Resource Utilization for Peak Demand

To provide the best Quality of Experience (QoE), there is a need to implement a resource utilization prediction engine that provides the ability to predict future demands of network resources. This engine has to be developed based on novel models, able to efficiently predict and plan the needed bandwidth capacity (based on auto-scaling functions) that will automatically accommodate the fluctuations for network resource requests for certain events, such as multimedia event provisioning or streaming sessions. In this context, Niu et al. [178] presented some of the issues of demand forecast and performance prediction in peer-assisted Video-on-Demand (VoD) services. The exploitation of the Box-Jenkins approach [182] is adopted to predict the future population of each video channel based on a given time series in the past. Using regression methods, periodicity is avoided as well as the seasonal ARIMA (autoregressive integrated moving average) model. The authors infer the initial population of a new released video channel, using machine learning techniques, and pass data from newly released video channels is used as training data to make a prediction, based on statistical models and the channels release time. The ARMA (autoregressive moving-average) model has been exploited to predict the server bandwidth demands by a video channel at future time. In other related work [183], a predictive cloud bandwidth auto-scaling system for VoD providers has

been proposed. Based on the historical data of bandwidth demand in each video channel derived by cloud monitoring services, the expectations for near future demands are estimated, deciding the minimum bandwidth reservation to satisfy the demand.

Wu and Lui [184] presented a system architecture and model for optimization of replication strategy in P2P-VoD systems. In this work, it is shown that conventional proportional replication strategy is not optimal and the authors proposed a passive replacement strategy for the decision of the content that should be deleted when a local storage is full. They also proposed an active replication algorithm to aggressively push data to peers, in order to achieve the desired replication ratios. A similar approach to the optimal content placement for a large-scale VoD system has been presented in [185]. RPS [186] is a publicly available toolkit that allows the creation of online and offline resource prediction systems. It includes its own monitoring facilities, but also provides the ability to use monitoring data that comes from other sources. Part of the system is the wavelet toolkit [187] that supports analysis of the signals. In [188], the Push-to-Peer approach was presented that proactively pushes content to peers in order to increase content availability, improving the use of peer uplink bandwidth. It allows performance analysis of the push policies, distributed load balancing strategies for the initial selection of serving peers and distributed strategies to cope with dynamic uplink bandwidth. The work in [189] illustrated the role of algorithmic research in the design of complex networked systems and explained how it has been adapted to balance the load in server clusters, manage the caches on servers, select paths through an overlay routing network, and elect leaders in the context of content delivery.

The authors in [190] proposed a network architecture that predicts the future content delivery demands and network usage, performing all the necessary optimal resource adaptations to deliver the content. In this research work, a collaboration between different actors involved in the delivery process (i.e., network operators, service providers and end-users) is motivated and, therefore, an upper layer that coordinates such a collaboration environment is introduced. More specifically, the entities of this plane interchange information with the current management and control planes of the CDN and Cloud providers, as well as with the so-called Media Home Gateway Cloud provider, which is an ecosystem of user gateways (e.g., set-top-boxes with advanced functionalities) with a distributed management. The network architecture consists of the following conceptual entities: Media Distribution Middleware (MDM), Media QoE Meter (MQM), Media Services Manager (MSM), an Enhanced Home Gateway (EHG) and a Media Advanced Streamer (MAS). These components cooperate together, creating different Management and Control (M&C) planes. The MDM is

the main component of the M&C plane as it executes all necessary operations and determines all data required for optimal allocation of the available resources at each Resource Provider's domain. Consequently, the MDM returns guidelines, which resources should be used for handling given user's request, to achieve the best (in terms of efficiency) resource exploitation. Another important component of the M&C plane is the MQM that is responsible for continuous monitoring of network metrics at the user's and service provider's domain access points, as well as the user's context and preferences. Based on the data gathered by the set of the MQM probes, distributed all over the domain, this entity provides to the MDM the related data about the current network conditions and the estimated value of QoE available for a user. Moreover, the MQM sends alerts to the MDM, only if any of the monitored QoS / QoE parameters declines below the allowed level.

2.5.2. Network Virtualization

Network virtualization brings several benefits to the network providers due to its flexibility, programmability and elasticity [191]. In the area of mobile telephony [192], [193], more than 800 MVNOs (Mobile Virtual Network Operators) offer services to the end-users, even though they do not own the wireless network infrastructure or they have not enough network resources [194]–[196]. When deploying a virtual infrastructure, the operator should take into consideration the flexibility and performance of the virtualization platform in addition to the cost [197]. Providing a virtual environment usually requires some overhead, which is necessary for managing resources between network devices that share the same node (real hardware). Overhead depends on the deployed virtualization software (i.e., platform) and has a significant impact on the performance (especially on throughput) and the packet losses of the routers.

The influence of the virtualization platform on the performance of virtualized network interconnect devices has been elaborated on several research works adopting different comparison approaches [198]–[200]. However, these comparisons are not accurate due to the exploited methodology of benchmarking Virtual Routers (VRs), as the comparison is performed at only one work point of the virtualized device [201]. The term 'work point' refers to the distribution of offered traffic load between the virtual routers into the device. Authors in other published works take measurements only when all the virtual routers suffer equal traffic load, ignoring other work points of the device. Egi et al. [202] analysed the differences between

Xen platforms with bridged and routed setup concluding that the same platform has different behavior – which depends on the setup parameters – and it privileges some domains in the virtualization platform, resulting in differences in throughput of the VRs. Furthermore, authors in [203], [204] draw different conclusions about the performance of the Xen platform, presenting analyses at two different working points. Rathore et al. [203] presented results by measuring the forwarding performance of a single virtual router in the presence of increasing number of VRs, which do not forward packet streams. In this case, the work point exists when one VR receives all the offered load. On the contrary, Mattos et al. [204] indicated scalability issues and provided a Xen Virtual Router Evaluation, however all VRs receive the same offered load.

The benchmarking test methodology presented in [205] considered several measurements related to packet forwarding performance of IP routers, including throughput, latency, loss rate, back-to-back frames, system recovery and reset. The throughput is defined as the maximum offered traffic load that can be forwarded by the device with no packet loss; a fact that indicates the usability limit of a particular device. In the case of a virtualized network device, the total offered traffic load consists of the sum of the loads offered to each VR. Even though the total offered traffic load is a simple scalar value, there are many options for partitioning the load across particular VRs. The mathematical model of the presented issue considers the dataset of loads offered to each of the N virtual routers $O = \{O_i : i = 1, \dots, N\}$, which represents the work point of the virtualized device. Subsequently, the total load offered to the network device O_{tot} is the sum of the N offered loads. The maximum number N of virtual routers in the device depends on the device itself, the virtualization platform (capacity) as well as the aimed function of the virtualized device. For measuring the distribution of the traffic load among different VRs, the fairness index of Jain et al. [206] has been exploited as follows

$$J = \frac{O_{tot}^2}{N \times \sum_{i=1}^{N} O_i^2} \qquad (2.1)$$

The above fairness index is a measure of the second moment for the dataset that, unlike the standard deviation, takes values from a limited range $J \in (0, 1]$. In the case of traffic load sharing among finite number of VRs, it ranges from $1/N$ (single VR receives the whole offered load and the remaining VRs are not loaded) to 1 (all VRs receive the same offered load). It is worthy to notice that other research papers focusing on virtualized network devices, usually assume that the same traffic is offered to all the VRs (i.e., $J = 1$). In this direction, it is critical

to examine if two different O sets (with different distributions of traffic load among VRs) with the same O_{tot} will result in the same forwarding performance. Experiments have been performed with 12 VRs deployed on a HP ProLiant DL360G6 server, running Xen 4.1.2 with host kernel 2.6.56 (64bit). Each VR (guest system) features 2 virtual network interfaces, while the host system maps the traffic to VR with VLAN tags. The traffic has been generated by Spirent TestCenter C1 (equipped with CM-1G-D4 card), whereas the tester and the Device Under Test were connected by two 1 Gbps Ethernet links in ring topology as proposed in [205].

2.6. Thesis Scope and Positioning

This research project investigates the debt-aware dynamic resource adaptation research problem in cloud elasticity management with value and strategy considerations. The goal is to optimize the resource provisioning and utilization in the cloud under dynamic optimal, well-compromised trade-off decisions in cloud service composition. This work focuses on multi-tenant software as a service cloud environments where multiple tenants submit requests for provisioning of VMs, resulting in arbitrary workload, and deploy various types of applications, such as multi-tenant software as a service and web applications that utilize the resources simultaneously. The cloud provider is not aware of the workloads that are executed and, consequently, the resource management system has to be application agnostic, i.e., able to handle arbitrary workloads. Since multiple types of applications can coexist in the system and share physical resources, there is a challenge of defining QoS requirements in a workload independent manner to avoid any application performance degradation. The multiple tenants establish SLAs with the cloud provider to formalize the QoS requirements and the provider pays a penalty in case of SLO violations.

One of the aspects of dynamic resource adaptation is that due to the unpredicted variations in the resource demand, the composition encounters underutilization or overutilization on the composite service components. The proposed approach in this thesis formally defines the problem of debt estimation in the context of cloud service utility adopting an economics-driven viewpoint with the estimation being subject to the service capacity. In this research work, the amount of profit not earned due to the underutilization of a cloud service component utility is measured, embracing the approach of unavoidable gaps in resource provisioning between an ideal and actual adaptation decision. This interest is caused due to the unused service capacity

or SLO violation, and lack of balanced, well-compromised trade-off decisions when auto-scaling in cloud. In this approach, the elasticity debt and profit optimization analytic methods are essential technical paradigms to solve the resource allocation problem in cloud and mobile cloud computing systems. The hypothesis is that the adaptation decision is driven by the resource capacity after applying value creation-driven techniques, determining when it is best to dynamically re-allocate resources from an overutilized composite service utility due to the dynamic changes in requests workload (aggregate requests).

Another aspect distinguishing the work presented in this thesis compared to related research in cloud elasticity management [27], [122]–[124], [160], [207] is the adoption of game theoretic control mechanisms to automatically re-allocate resources from an overutilized service component utility that will directly influence the resource utilization and Quality of Service delivered by the systems. This game theoretic incentive scheme with debt-aware considerations for value creation is based on an equilibrium model and motivates the optimal auto-scaling of resources in cloud and mobile cloud computing environments. The control mechanisms optimally solve the composite service component utility overutilization problem (without affecting the overall utility of the system or the application owner) by forming dynamic coalitions, optimizing trade-off decisions and scheduling resources while analysing the profile of debt minimization and profit maximization based on a Hidden Markov Model. The scope of this research project is summarized in Table 2.2.

Table 2.2. Thesis scope

Characteristic	**Thesis Scope**
Target Systems	Multi-tenant software as a service clouds
Architecture	Multi-tenant service architecture
System Resources	Multiple resources: CPU, memory, bandwidth, storage, throughput, I/O processor
Goal	Optimize resource provisioning and utilization under balanced, well-compromised trade-off decisions in cloud elasticity management
Techniques	Dynamic debt- and profit-aware resource adaptation with value and strategy considerations; Game theoretic control mechanisms
Workload	Arbitrary workloads

UNIVERSITY of NICOSIA

2.7.Conclusions

To facilitate further developments in the cloud elasticity management area, it is essential to review the existing body of knowledge. This chapter presented a taxonomy and survey of elasticity-centric design approaches in the context of cloud resource management, covering also the open research challenges faced by the existing mechanisms. Recent research advancements have been discussed and classified into quality-driven, cost-aware, energy-aware, intercloud-oriented and economics-driven design approaches. In particular, the economics-driven mechanisms adopt criteria, such as economics-inspired frameworks, adaptation strategies deemed as investments with long-term rewards under uncertainty, pricing techniques and yield management for profit maximisation, and analysis of the cost efficiency and cost effectiveness of adaptation decisions. The chapter has concluded with a discussion of the scope and positioning of this research thesis in the context of the presented taxonomy and reviewed research. The proposed research direction is dynamic debt-aware resource adaptation in cloud elasticity management with value and strategy considerations under optimal trade-off decisions.

UNIVERSITY of NICOSIA

Chapter 3

Gap Analysis of Debt-Aware Strategies in Cloud Elasticity Management

Prior to designing new models and algorithms for dynamic cloud resource adaptation, it is important to provide a theoretical analysis of the potential applicability of the technical debt metaphor in the context of cloud elasticity management and service composition. One of the aspects of dynamic resource adaptation is that due to changes on the workload, the composition encounters underutilization or overutilization on the composite service components. This chapter formally defines the problem of debt estimation at cloud service utility level from an economics-driven viewpoint. To understand the nature of the problem, the need to investigate a cost estimation model for software as a service implementation in the cloud is motivated, exploiting the COCOMO estimations and applying a tolerance value for prediction. In the sequel, a debt estimation model is proposed from the multi-tenant software as a service application viewpoint in the context of cloud service utility with the estimation being subject to the service capacity.

3.1. Introduction

This chapter motivates the need to estimate debts in cloud elasticity and provides a gap analysis of disruptive debt-aware strategies in the context of cloud service composition. A theoretical analysis is presented towards the potential applicability of the technical debt paradigm from the software engineering field as a solution concept for the development of a conceptual model of elasticity to support resource adaptation decisions in the cloud. It is envisioned that the elasticity debt quantification perspective will motivate cloud providers to consider debt as a dimension for value creation in elasticity management. In this context, this work differs from the prior analytic literature in the way the system, workload, service utility and capacity are modelled. The problems to be solved are described as follows:

1. *Lack of a conceptual model of elasticity with technical debt considerations to support cloud resource adaptation decisions at runtime*. Cloud elasticity management depends on interrelated factors that have an impact on the value of resource adaptation decisions

and, thus, long-term utility. The results of the survey reveal that, although there are elasticity initiatives that consider the economics of elasticity management, there is little research focusing on economics-driven frameworks to make explicit value creation considerations when reasoning about elasticity adaptations.

2. *Neglecting the potential utility of the unavoidable gaps in resource provisioning and utilization*. There is no elasticity management capable to create a perfect match between resource demand and supply at runtime due to the dynamic variations over time. The results of the survey indicate that the potential utility of unavoidable gaps in resource utilization has been neglected in existing research, increasing the effects of uncertainty and conflicting trade-offs at runtime.
3. *Compromising conflicting perspectives in elasticity management for multi-tenant software as a service applications*. Multi-tenant software as a service applications limit the diversity of tenants by clustering them in categories with pre-defined service level objectives (SLOs). This clustering favours application owners through an aggregate resource provisioning, but forces tenants to adjust their initial SLOs.
4. *Lack of optimization methods to mitigate unavoidable gaps in resource provisioning in the context of multi-tenant software as a service*.

In the analysis of the problem, it is assumed that future events cannot be predicted based on the knowledge of past events. Although this assumption may not be satisfied for all types of real-world workloads, it enables the theoretical analysis of models and algorithms that do not rely on predictions of the future workload. Moreover, the higher the variability of the workloads, the closer they are to satisfying the unpredictability assumption. Since cloud-supported services and applications usually experience highly dynamic workloads, the unpredictability assumption is justifiable. The key contributions of this chapter are the following:

1. A survey on technical debt in the software engineering research area.
2. A cost estimation model for software as a service implementation in the cloud, exploiting the COCOMO model and applying a tolerance value for prediction.
3. A preliminary debt estimation model at cloud service utility level using a multi-tenant software as a service application approach.
4. An evaluation and analysis of the proposed models.

The remainder of this chapter is organized as follows: Section 3.2 provides background information on technical debt in software engineering. Sections 3.3 and 3.4 present a cost estimation approach for software as a service implementation in cloud and a debt-aware model

for cloud service utility in multi-tenant software as service applications, respectively. The chapter is concluded with a summary in Section 3.5.

3.2. Background on Technical Debt in Software Engineering

The technical debt is a metaphor used in the software engineering research field that was initially coined by Cunningham in 1992 [10]. In this metaphor, it is explained how the technical debt correlates the software development with the financial debt [208] and the extra work that has to be held in the future aiming to improve some of the features or non-functional requirements of the software, such as the readability or complexity [209]–[211]. Related works [212], [213] described an analogy between several financial debt attributes (e.g., amnesty, principal, leverage) and their meaning in the technical debt metaphor. Narrowing in, the term refers to the acceleration of velocity for releasing software products that might lead to implications, extra effort for fixing issues, additional cost and likely interest payments [214]. It can be a result of reduced testing, bugs shipped with the software or deflated investment [215]. The re-work that should be held when technical debt occurs can range from a trivial with few amendments to changes that affects the core system's functionality [216]. The applied software development lifecycle is an element that affects the size of the technical debt as, for example, agile processes are able to create less technical debt than waterfall models because of the more flexible structure they have [217], [218]. In this context, the technical debt has been described in different domains such as [219], [220]:

1. Design / code debt, referring to poor quality of code.
2. Testing debt, indicating tests that were not run.
3. Documentation debt, pointing out missing or inadequate documentation.
4. Defect debt, indicating defects that were not fixed.
5. Configuration management debt.
6. Infrastructure debt.

Delivering high quality, defect-free software is the goal of all software projects. To ensure the delivery of high-quality software, software projects often plan their development and maintenance efforts [221]. However, in many cases, developers are rushed into completing tasks for various reasons. There are several causes that create technical debt; the most common reasons are cost reduction, satisfying customers and business / market pressures from

competition (e.g., strict deadlines) with release of software products sooner than the initial schedule [222], without considering the implications of the non-systematic verification of the quality [223], [224]. It is not easy to estimate how much time and effort is needed in order to reinforce the technical debt, thus the result of missing important deadlines always lurks. Consequently, the debt incurs when making technical compromises that are expedient in short-term, but that create a technical context that increases complexity and cost in the long run [225]. If these technical compromises are not paid back, then technical debt can be incurred and degrade the system quality or the development team productivity. However, incurring technical debt is not always bad, if the organization makes informed decisions or has strategic reasons about to incur debt as mentioned in [226]. McConnell [227] classified the technical debt into intentional and unintentional. Intentional debt is taken to optimize the present value in the software project rather than future value or to make informed decisions for gaining short-term benefits. On the other hand, unintentional debt is incurred unknowingly by making non-strategic decisions in a software project. Although it is shown that the debt has an adverse effect on the quality of the software, there is a dearth of work to validate the relationship between technical debt scores against theoretical quality models. Authors is [228] conducted a case study across 10 releases of 10 open-source systems to evaluate three proposed methods of technical debt principal estimation against external quality models. It was noted that, in a multiple linear regression analysis, a different estimation technique had a significant relationship to the quality attributes effectiveness and functionality.

Another cause is parallel development, where changes into a codebase from different branches are made in isolation without been merged into a single source codebase effectively [229]. Technical debt is a result of the poor software architecture and development within a codebase [230]. Future improvements on the software will be a result of the initial poor design process; in this direction, the technical debt can be expressed as a loan that should be paid off as future improvement on the software, otherwise it will result in interests with rates difficult to be managed [231]. Li et al. [232] evaluated architectural decisions from a value-oriented perspective and used the debt to monetise the gap between an optimal and suboptimal architecture when a change scenario occurs. Delayed refactoring is another element that leads to technical debt; the term points out that parts of the code need to be cleaned up in such a way that the external behavior of the code is not altered yet improves its internal structure [233]. Deferred refactoring will result to more code that should be written in the future and, subsequently, to accumulated technical debt that will be cleared out when the refactoring process will be done [234]. There are two possible ways to encounter technical debt when it

occurs; either to pay interest lifetime or proceed to refactoring of the bugs in order to improve the initial poor design [235]. It is important to mention that proceeding with refactoring is more beneficial long-term, because the interest payments will be decreased in the future [236]. In addition, the software is more robust, reliable and easy to maintain, can be fixed from bugs easier – as the source code is more readable – and be more customizable when extending its features [237]. Kazman et al. [238] attempted to quantify the architecture debts in large-scale software projects by modeling the software architecture as a set of design rule spaces. The exploited data was extracted from development artifacts with the aim of identifying the files implicated in architecture flaws and suggest refactoring strategies.

Leitch and Stroulia [239] proposed a novel method for a refactoring plan based on predicting the return on investment (ROI) as the increased adoption of refactoring practices in new agile methodologies and the lack of any prescriptive theory on when to refactor are witnessed over the years. Brown et al. [240] attempted to define the technical debt issue and establish a framework; in this research work, the authors argued that the long-term effects of short-term expedients need to be managed effectively to successfully deliver complex software-reliant systems. Zazworka et al. [241] elaborated on the impact the design debt has on the quality of the software products in terms of god classes. The accumulated debt affects a software product and causes delay in the overall software development process as re-work should be held in order to fix the bugs. Henceforth, god classes are changed more often than non-god classes, ending up to the conclusion that the technical debt has a negative impact on software quality and should therefore be identified during the development process. A broad definitional framework of the technical debt from the stakeholder perspective is proposed in [242], describing how technology gaps affect the quality within the technical environment. Establishing such a framework will enable the technology stakeholders to build a conceptual model better aligning their concerns [243].

The work of Wiklund et al. [244] indicated that the test-driven development methods have become increasingly popular and elaborated on the fact that the development of test automation tools encounter problems due to unpredictable issues. Furthermore, a test automation approach linked to the technical debt is adopted, performing a case study on a telecommunication sub-system and suggesting improvements in the fields of interaction design and general software design principles. Nugroho et al. [245] elaborated on a quantification approach for debts and interest, which can lead to better quality-oriented IT investments, taking into account an empirical assessment method of software quality. The technical debt quantification is based on empirical data monitored by the Software Improvement Group (SIG). The approach is applied

to a real system, providing insights into the IT investment (i.e., return on investment in software quality improvement). On the other hand, Letouzey [246] presented the SQALE method for effectively monitoring, analysing and estimating the quality and technical debt of the source code. Narrowing in, a quality and analysis model is proposed towards providing recommendations and aiming to analyse the structure and the impact of technical debt. Finally, authors in [214] introduced a metric for effectively managing the technical debt and optimizing the cost of development from an architecture-related viewpoint. A demonstration is presented examining the metric's applicability to an ongoing system development effort.

To address the problem of documentation debt, one possible source to detect it is using adequate source code comments, also referred to as self-admitted technical debt. However, what types of debt can be detected using adequate source code documentation remains an open question [247]. The work of Maldonado and Shihab [213] examined the code comments in order to determine different types of technical debt. Four filtering heuristics are proposed to eliminate comments that are not likely to contain technical debt. The analysis concludes that self-admitted technical debt can be classified into five main types: design debt, defect debt, documentation debt, requirement debt and test debt. The most common type of self-admitted technical debt is design debt, making up between 42% to 84% of the classified comments. On the contrary, the relation between self-admitted technical debt and software quality has been investigated in [248], conducting an empirical study using five open source projects, i.e., Hadoop [249], Chromium, Cassandra [250], Spark [251] and Tomcat [252]. More specifically, it was attempted to measure the difficulty of a change using well-known measures, such as the amount of churn, the number of files, the number of modified modules in a change and the entropy of a change. The findings of this work are summarized as follows:

1. There is no clear trend when it comes to defects and self-admitted technical debt, although the defectiveness of the technical debt files increases after the introduction of technical debt.
2. Self-admitted technical debt changes induce less future defects than none technical debt changes.
3. Self-admitted technical debt changes are more complex.

The work in [253] introduced a dataset with project measurement data from 33 Java projects from the Apache Software Foundation. To identify the fault-inducing and -fixing commits, all commits were analysed from separately defined time frames with SonarQube [254] to collect technical debt information and with Ptidej [255] to detect code smells. Ramasubbu and Kemerer [256] developed an evolutionary model of technical debt accumulation to facilitate a

rigorous and balanced analysis of its benefits and costs. The theory focuses on identifying different trade-off patterns between software quality and customer satisfaction under early and late adopter scenarios at different lifecycle stages of the software package. A ten-year longitudinal dataset drawn from 69 customer installations of the software package is used to evaluate the approach.

A lack of significant work related to the symptoms of technical deb interest in the form of increased defect- and change-proneness is also identified in the literature. To address this research question, authors in [257] selected four different identification techniques (i.e., code smells, automatic static analysis issues, grime buildup, modularity violations) and applied them to 13 versions of the Apache Hadoop open source software project [258]. The aggregation of statistical measures was motivated to investigate whether the different techniques identified debt indicators in the same or different classes and whether those classes in turn exhibited high interest. Among the findings, dispersed coupling and modularity violations were co-located in classes with higher defect-proneness, in addition to a strong relationship between modularity violations and change-proneness.

Finally, forecasting the technical debt effectively is another challenge of great importance over recent years. Authors in [259] evaluated the applicability of machine learning methods to model and predict the technical debt evolution. An empirical study has been conducted based on a custom dataset that included weekly snapshots of fifteen open-source software projects over a three-year period and using two popular static analysis tools to extract software-related metrics as predictors. The results indicated that linear regularization models are able to fit and provide meaningful forecasts of technical debt evolution in shorter forecasting horizons, while non-linear random forest regression models perform better in longer forecasting horizons.

3.3. The Cost Estimation Problem in Cloud-Based Software Engineering

In this section, the need to investigate a cost estimation model for software as a service (SaaS) implementation in the cloud is motivated. Three different scenarios (optimistic, most likely and pessimistic) are examined, exploiting the COCOMO estimation in man-months and after applying a tolerance value for prediction. The occurrence of technical debt in the future and its estimation is researched as a result of budget constraints. Estimating effectively the cost

for cloud-based software as a service implementation, a number of advantages are encountered as follows:

1. Identify the technical debt for different scenarios and estimate its occurrence to develop mitigation strategies.
2. Reduce uncertainty as far as it concerns the actual (optimal) budget.
3. Provide optimistic and pessimistic scenarios driven by budgetary constraints.

3.3.1. COCOMO Model Exploitation for Debt Estimation

Several software development cost estimation models have been analysed in the scientific literature using a range of techniques from function points to wavelet neural networks [260]–[264]. The Constructive Cost Model (COCOMO) [265] is exploited to predict the technical debt in cloud implementations by casting the applied effort into cost and providing a range of secureness after a tolerance value for prediction is applied. The risk of entering into a technical debt in the future is high outside this range of secureness. COCOMO is a software cost estimation model and is applied to three software development modes (i.e., organic, semi-detached and embedded). The mathematical formulae are given as

$$Effort\ Applied\ (E) = a_b(KLOC)^{b_b}\ [man - months] \quad (3.1)$$

$$Development\ Time\ (D) = c_b(Effort\ Applied)^{d_b}\ [months] \quad (3.2)$$

$$People\ Required\ (P) = Effort\ Applied/Development\ Time\ \ [count] \quad (3.3)$$

where $KLOC$ is the number of source lines of code (in thousands). The coefficients a_b, b_b, c_b and d_b for each development mode are shown in Table 3.1.

Table 3.1. Coefficients for different development modes

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

3.3.2. Cost Estimation Model for Software as a Service Implementation in Cloud

Oza et al. [266] mentioned that the technical debt might be a potential risk when using the cloud in distributed software development, concluding that the risks may outweigh the benefits associated with it. The total cost for a software as a service (SaaS) implementation in cloud is estimated by casting the applied effort – given by the COCOMO model – into cost for three different case scenarios (i.e., optimistic, most likely and pessimistic). The hypothesis is that the technical debt is associated with deflated investment due to budgetary constraints. In this direction, the gap between the predicted optimal budget and the actual budget is investigated, as well as the probability of entering into a technical debt status due to this gap. The following modeling assumptions are considered:

1. The implementation process is separated into five sub-processes: development, configuration, deployment, licenses and infrastructure.
2. The COCOMO most likely value declares the optimal condition.
3. There is a tolerance value for prediction that reflects:
    a. Optimistic scenario: -10% of the most likely estimation for cost and effort.
    b. Pessimistic scenario: +10% of the most likely estimation for cost and effort.
4. Product customizability (e.g., add features) might be affected from budget constraints.

Towards more accurate future debt estimations, the existence of the actual budget within the specified range – optimistic, most likely, pessimistic – is examined. The risk of entering into a technical debt status lurks out of this range of secureness with inevitable results, such as the options of abandoning or termination that result in accumulated debt difficult to be managed. On the contrary, in case that the actual budget is within this range of secureness, the risk of entering into a debt status in the long run is lower and depends on other factors, such as the levels of experience and productivity of the development team. In this context, the sum of the cost for each of the five sub-processes corresponds to the cost estimation modeling for software as a service implementation in cloud, which is given in the following equation 3.4 as

UNIVERSITY of NICOSIA

$$
\begin{aligned}
Cost = C_{development} + C_{configuration} + C_{deployment} + C_{licences} + C_{infrastructure} \\
= \left(weight_{develoment} + weight_{configuration} + weight_{deployment}\right. \\
\left. + weight_{licenses} + weight_{infrastructure}\right) * Effort\ applied\ in\ man \\
- months * Average\ monthly\ salary\ per\ employee \\
= Effort\ applied\ in\ man - months \\
* Average\ monthly\ salary\ per\ employee
\end{aligned} \tag{3.4}
$$

Table 3.2. Cost Estimation Model: Variable Metrics & Definitions

Variable	Variable Definition	Metric
$C_{development}$	Development cost (implementation, migration and maintenance)	Monetary units
$C_{configuration}$	Software configuration cost	Monetary units
$C_{deployment}$	Deployment cost (including testing)	Monetary units
$C_{licences}$	Software license cost	Monetary units
$C_{infrastructure}$	Infrastructure cost (including hardware)	Monetary units
$weight_{develoment}$	Weighted priority rating for development sub-process	Percentage
$weight_{configuration}$	Weighted priority rating for configuration sub-process	Percentage
$weight_{deployment}$	Weighted priority rating for deployment sub-process	Percentage
$weight_{licenses}$	Weighted priority rating for license sub-process	Percentage
$weight_{infrastructure}$	Weighted priority rating for infrastructure sub-process	Percentage

3.3.3. Evaluation and Analysis

An embedded software development mode is examined for a product size of twelve thousand (12,000) source lines of code. The obtained estimations are shown in Table 3.3 after exploiting the COCOMO model.

Table 3.3. Estimations for effort, development time and required people

Metric	Estimation
$Optimistic\ effort\ applied$	64 man-months
$Most\ likely\ effort\ applied$	71 man-months
$Pessimistic\ effort\ applied$	78 man-months
$Development\ time$	9.8 months
$People\ required$	7

To further elaborate on the obtained estimations, the tolerance value for prediction is applied that reflects -10% and +10% of the most likely effort applied for the optimistic and pessimistic scenarios, respectively. A range of secureness is defined with the required man-months of effort applied to be from 64 to 78; out of this range of secureness, the risk of entering into a technical debt status in the future is high.

Based on the modeling assumptions, the most likely value (71 man-months) declares the optimal condition. The weighted priority ratings for each sub-process are set as follows: development = 40%, configuration = 10%, deployment = 20%, licenses = 10%, and infrastructure = 20%. The average monthly salary per employee is set to 1,500 (in monetary units). For this software as a service implementation in the cloud, the decomposed cost estimation for each sub-process and scenario is shown in Table 3.4.

Table 3.4. Optimistic, Most Likely & Pessimistic Scenarios: Cost estimations for each sub-process

Sub-process	Optimistic	Most Likely	Pessimistic
Development	38,340	42,600	46,860
Configuration	9,585	10,650	11,715
Deployment	19,170	21,300	23,430
Licenses	9,585	10,650	11,715
Infrastructure	19,170	21,300	23,430
Total Cost	95,850	106,500	117,150

The tolerance value for prediction is applied that reflects -10% and +10% of the most likely cost estimation (106,500) for the optimistic and pessimistic scenarios, respectively. In this direction, the actual budget should be in the range from 95,850 to 117,150. Out of this range of secureness, it is more likely to enter into a technical debt status in the future. It is now examined the case of budget constraints – provided budget is 80,000 – that requires a substantial decrease in the estimated man-months to 53; in this context, the development sub-process should be accelerated. Hence, there is a risk of entering into a technical debt status in the future for that scenario, which means re-work and additional cost due to the decrease in the estimated man-months by 18. This decrease in the required effort constitutes the difference between the predicted budget against the actual and could be applied in any of the other sub-processes and tasks that would enhance the product's features and customizability – e.g., testing in deployment phase, documentation in development phase or configuration. In such a scenario, the option of abandoning the software in the long run will create accumulated technical debt and any positive technical debt to be further incurred can be hardly managed.

On the contrary, the scenario where the provided budget is 99,000 is examined. In this case, the budget belongs within the estimated range of secureness, so technical debt will not necessarily occur in the future. To further elaborate, the estimated man-months should be now decreased to 66, creating an impression that technical debt tends to occur as the requirements for either the most likely effort applied or the provided budget are not fulfilled. However, creating such an "intentional" technical debt [267] could be proved beneficial if we assume that the development team members are very experienced and highly productive and, hence, they need less effort in man-months to complete the tasks.

3.4. Debt-Aware Disruption in Cloud Service Composition

Despite the fact that there is related work in the technical debt area in different domains within the software engineering research field as presented in Section 3.2, there is a significant gap for optimal debt-aware strategies in the context of cloud elasticity or service composition level in cloud software as a service (SaaS). In this section, a theoretical analysis is presented towards the applicability of the technical debt metaphor in the context of cloud elasticity management and service composition. It is attempted to quantify the debt from the lease of

cloud software as a service perspective and its estimation is subject to the service capacity. Identifying and estimating effectively the debt at cloud service utility level, the following advantages are encountered:

1. Optimize the resource utilization in cloud environments to reduce energy consumption.
2. Create efficient resource adaptation strategies in the cloud inspired by the gradual payoff of the debt.
3. Inform and predict the underutilization or overutilization states of a cloud-supported service.

3.4.1. Technical Debt Model for Cloud Service Utility in Multi-Tenant Software as a Service

The debt is attempted to be measured from the cost that derives from the unused service capacity in the cloud (i.e., interest due to unavoidable gaps in resource provisioning). The hypothesis is that the selection decision is subject to the variation in the demand for cloud resources and the way the debt is gradually paid off without risking a service utility overutilization status. Billing cycles apply through monthly subscriptions for using cloud services as well as service provisioning costs; both pricing schemes vary throughout the modeling period λ that is declared as the years of return on investment (ROI) and for paying off the debt. The following modeling assumptions are adopted for any candidate cloud-supported service to be leased off:

1. Service adaptability to market demands, i.e., capability for resource adaptations caused by a linear variation (increase or decrease) in the resource demand.
2. The level of service adaptability affects the billing cycles and service provisioning costs accordingly.
3. The multi-tenant cloud-supported services have comparable functional requirements; however, the non-functional requirements can be either comparable or not, but with distinct service capacities.
4. Two types of debt are estimated: a) positive debt, declaring the service utility underutilization and demonstrating the capability to satisfy future dynamic variations in the resource demand; b) negative debt, declaring the service utility overutilization

and indicating the need for abandoning the existing service and/or switching to an upgraded one.

In this context, the debt estimation modeling from the lease of cloud software as a service viewpoint (i.e., multi-tenant approach) in the context of cloud service utility is given in the following equation 3.5 as

$$TD_i = 12 * \left\{ \left(1 + \frac{\Delta\%}{\lambda}\right)^{i-1} * ppm * \left[U_{max} - (1+\beta\%)^{i-1} * U_{curr}\right] - \left(1 + \frac{\delta\%}{\lambda}\right)^{i-1} * Cu_{/m} * \left[U_{max} - (1+\beta\%)^{i-1} * U_{curr}\right] \right\}$$

$$= 12 * \left[U_{max} - (1+\beta\%)^{i-1} * U_{curr}\right] * \left[\left(1 + \frac{\Delta\%}{\lambda}\right)^{i-1} * ppm - \left(1 + \frac{\delta\%}{\lambda}\right)^{i-1} * Cu_{/m} \right], where\ i = 1, 2, \dots \lambda \quad (3.5)$$

Table 3.5. Technical Debt Model: Variable Metrics & Definitions

Variable	Variable Definition	Metric
TD	Technical debt estimation	Monetary units
i	Index of the year	
U_{max}	Maximum service capacity in terms of active users	
U_{curr}	Number of current active users	
$\beta\%$	Average yearly variation in demand	Percentage
ppm	Price per monthly subscription	Monetary units
$\Delta\%$	Average variation in the monthly subscription price	Percentage
$Cu_{/m}$	Monthly service provisioning cost in cloud	Monetary units
$\delta\%$	Average variation in the monthly service provisioning cost in cloud	Percentage

UNIVERSITY of NICOSIA

3.4.2. Evaluation and Analysis

Two different cloud-supported services – Golden and Silver – are evaluated; Golden is a corporate service with high-level Quality of Service (QoS) guarantees in terms of non-functional requirements (i.e., availability, performance), while Silver is a regular service with less features than Golden. Given their distinct service capacities, the debt formation is investigated for two case scenarios over a 5-year modeling period ($\lambda = 5$), in addition to the probability of underutilization or overutilization of either the Golden or Silver service. Assuming a linear yearly increase in the demand, the first case scenario examines a 15% increase in the resource demand, while a 45% increase is forecasted in the second scenario. The data input towards the debt estimation is presented in Table 3.6.

Table 3.6. Data Input for Formula in Section 3.4.1

Variable Definition	Golden Service	Silver Service
Maximum service capacity in terms of active users	$U_{max} = 10{,}000$	$U_{max} = 4{,}000$
Number of current active users	$U_{curr} = 1{,}500$	$U_{curr} = 1{,}500$
Price per monthly subscription	$ppm = 10$	$ppm = 10$
Average variation in the monthly subscription price	$\Delta\% = 3\%$	$\Delta\% = 3\%$
Monthly service provisioning cost in cloud	$Cu_{/m} = 6$	$Cu_{/m} = 6$
Average variation in the monthly service provisioning cost in cloud	$\delta\% = 1\%$	$\delta\% = 1\%$

The obtained results for the two case scenarios are presented in Tables 3.7 and 3.8, while the debt estimation flow is shown in Figures 3.1 and 3.2, respectively. For the first scenario, the Golden service is always underutilized over the 5-year period as indicated by the positive technical debt estimation results. The debt is gradually paid off, which is witnessed by the continuous decrease in the number of monetary units as a result of the linear increase in the demand. Likewise, the Silver service is underutilized with gradual payoff of the debt throughout the modeling period. Despite the fact that the Golden service has high-level QoS guarantees and is more likely to adapt to the evolving market demands, the proper service selection decision is

the Silver service for that case scenario for a number of reasons, such as the low risk of entering into a debt status in the future and the problem of service utility overutilization does not lurk over the 5-year modeling period, avoiding accumulated debt that would motivate the need for abandoning the existing service and/or switching to an upgraded one.

Table 3.7. Case scenario 1 – 15% yearly increase in demand: Technical debt estimations

Term	Golden Service	Silver Service
Year 1	408,000	120,000
Year 2	401,966.4	110,510.4
Year 3	394,047.04	99,110.85
Year 4	383,928.45	85,487.72
Year 5	371,246.19	69,276.42

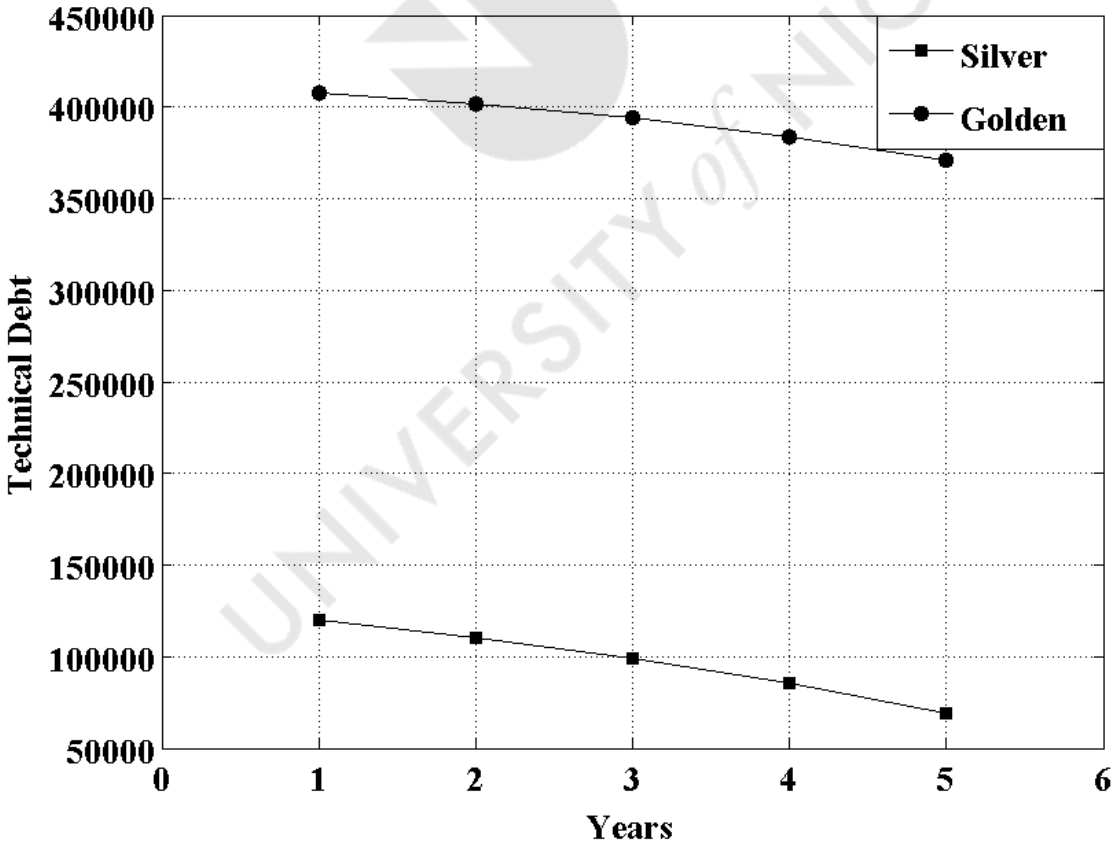


Figure 3.1. Case scenario 1 – 15% yearly increase in demand: Technical debt estimation flow

UNIVERSITY of NICOSIA

Table 3.8. Case scenario 2 – 45% yearly increase in demand: Technical debt estimation

Year	Golden Service	Silver Service
Year 1	408,000	120,000
Year 2	380,107.2	88,651.2
Year 3	336,534.48	41,598.29
Year 4	269,942.75	-28,497.98
Year 5	169,568.13	-132,401.63

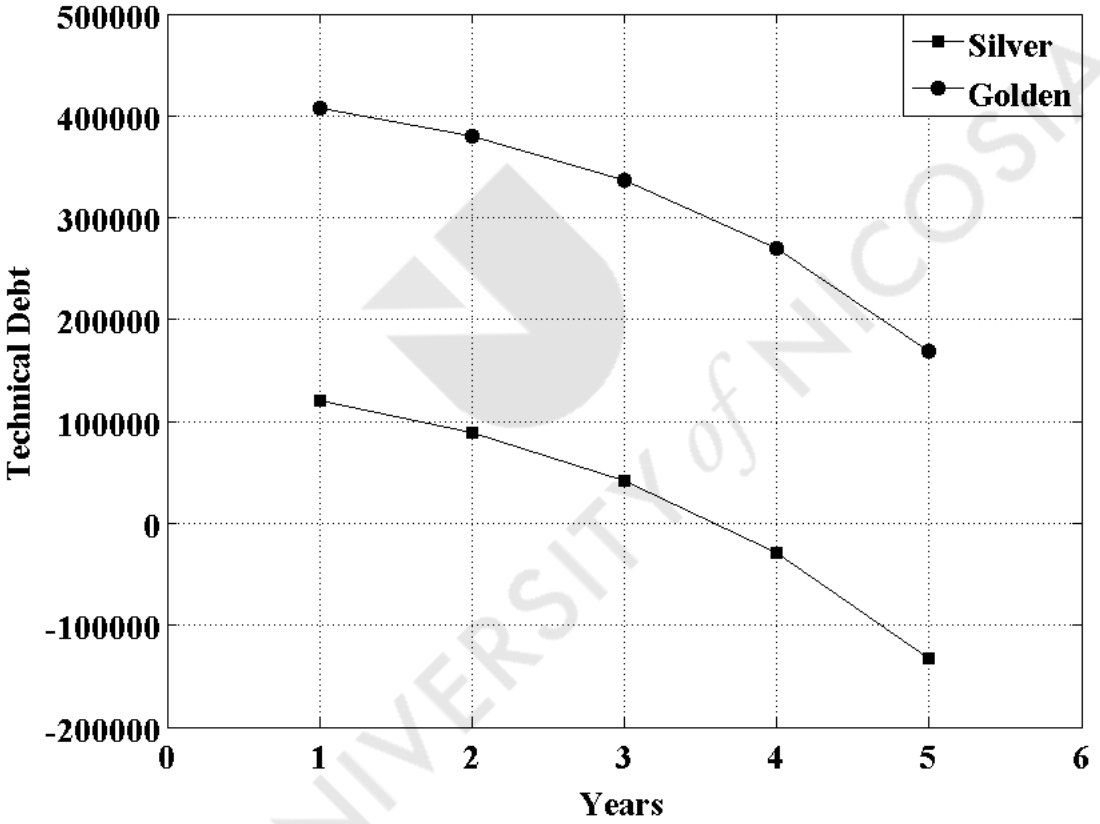


Figure 3.2. Case scenario 2 – 45% yearly increase in demand: Technical debt estimation flow

For the second case scenario, the Golden service is underutilized and the debt is gradually paid off, despite the sheer increase in the demand. As far as it concerns the Silver service, similar behavior is witnessed during the first three years. In year 4, the debt becomes zero at some point, indicating that it is totally paid off (optimal condition). However, due to this increase in the resource demand, the debt estimations become negative from the fourth year

until the end of the modeling period, pointing out that the service is overutilized. The need for abandoning the existing service and/or switching to an upgraded one will be faced, requiring re-work and additional cost. The options of abandoning and switching will create a new debt and any positive debt to be further incurred can be hardly managed. Hence, there is a high risk of entering into an accumulated debt status, which is overburdened by the additional cost of abandoning and switching. Consequently, the Golden service is the proper service selection decision for that scenario, as the problem of service utility overutilization will not lurk, the debt will be gradually paid off and the risk of entering into a new and accumulated debt is low. Additionally, the service utility specifications are better than the Silver and more likely to adapt to the resource demands, not to mention that the end-users will enjoy a better service in terms of QoS guarantees.

3.5.Conclusions

This chapter presented a theoretical analysis of the potential applicability of the technical debt metaphor in the context of cloud elasticity management and service composition in multi-tenant software as a service applications. As the composition encounters underutilization or overutilization on the composite services, a research gap is witnessed in the scientific literature for disruptive debt-aware estimation techniques from the multi-tenant software as a service perspective. To further understand the nature of the problem, the need to investigate a cost estimation model for software as a service implementation in the cloud is motivated, exploiting the COCOMO estimations and applying a tolerance value for prediction. The evaluation results prove the effectiveness of the proposed debt-inspired modeling for different cloud service capabilities in multi-tenant software as a service applications.

The next chapter presents debt-aware quantitative models and algorithms for dynamic resource adaptation in cloud elasticity management, embracing the approach of unavoidable gaps in cloud resource provisioning.

UNIVERSITY of NICOSIA

Chapter 4

Debt-Aware Techniques in Cloud Resource Provisioning: Quantitative Model and Algorithm Formulation

This chapter elaborates on a debt-aware research approach to dynamic resource adaptation in cloud elasticity management as the technical debt metaphor is applied in the context of cloud elasticity. The novel, cloud-inspired quantitative modeling and algorithm formulation measures the amount of profit not earned due to the underutilization of cloud resources, embracing the approach of unavoidable gaps in resource provisioning between ideal and actual adaptation decisions. This interest is caused due to the unused service capacity or SLO violation, and lack of effective, well-compromised trade-off decisions when auto-scaling in cloud. In this approach, the elasticity debt and profit optimization analytic methods are investigated as technical paradigms to solve the resource allocation problem in cloud and mobile cloud computing systems. The hypothesis is that the adaptation decisions are driven by the resource capacity after applying value-added and strategy considerations, determining when it is best to dynamically re-allocate resources from an overutilized composite service component utility due to the dynamic changes in requests workload. The high efficiency of the proposed models and algorithms is shown by extensive scientific experiments using different cloud service capabilities and storage capacities in the context of multi-tenant software as a service.

4.1. Introduction

This chapter presents a set of debt- and profit-aware quantitative models and algorithms to solve the resource adaptation problem in cloud elasticity management. The problem formulation applies statistical variation in the demand for cloud resources and services (either non-linear or linear) and examines the system behavior to infer potential future states. The proposed quantification models and algorithms have the below major advantages:

1. Measure and minimize the size of elasticity debt in a cloud resource management context under QoS constraints.

UNIVERSITY of NICOSIA

2. Predict the risk of entering into a new and accumulated elasticity / technical debt in the long run.
3. Optimize profits and costs at cloud storage level for big data analytics as a service platforms.
4. Auto-scaling of systems (i.e., cloud and mobile cloud systems) when new resource capacity requests are added, which is essential for large-scale cloud providers towards unrestricted functionality, storage and mobility to serve a multitude of (mobile) device users anywhere and anytime [268], [269].

The proposed approach to dynamic cloud resource adaptation considers the following limitations:

1. A multi-tenant, cloud-supported service utility is considered to be underutilized.
2. A multi-tenant, cloud-supported service utility is considered to be overutilized, motivating the option of abandoning/termination.
3. Selecting the active service subject to resource capacity in order to swich from an overutilized service utility and avoid violating the SLO objectives (e.g., QoS requirements).

The research approach can improve resource provisioning and scheduling by eliminating performance degradation and adhering effectively to strict SLA and QoS requirements, as well as handling multi-core CPU architectures, heterogeneous infrastructure and VMs. The models and algorithms are evaluated by extensive scientific experiments using a quantification tool (proof of concept prototype). According to the experimental results, the proposed approach can effectively measure the size of the technical / elasticity debt when variations in the demand requests for cloud resources are applied and provide the point threshold at which overutilization of cloud resources and service utilities are observed throughout the modeling period.

The key contributions of this chapter are the following:

1. The introduction of a debt- and profit-aware approach to efficient and dynamic resource adaptation in cloud elasticity management.
2. Novel quantitative models and algorithms to solve the resource provisioning problem in cloud and mobile cloud systems, following the elasticity debt and profit optimization viewpoints in the context of multi-tenant software as a service applications.
3. An extensive simulation-based evaluation and performance analysis of the proposed models and algorithms.

The remainder of the chapter is organized as follows: Section 4.2 introduces the system model used in the design of the models and algorithms for dynamic resource adaptation,

followed by an evaluation and analysis of the obtained experimental results in Section 4.3. The chapter is concluded with Section 4.4 providing a summary of the results and scientific contributions.

4.2.System Model

4.2.1. Problem Formulation and Parameter Setting

The target system is a multi-tenant software as a service (SaaS) cloud environment provided by an Infrastructure as a Service (IaaS) cloud provider. The underlying IaaS environment is represented by a large-scale data centre consisting of N heterogeneous physical nodes. Each node i is characterized by the CPU performance, amount of RAM and network bandwidth. The type of the environment implies no knowledge of application workloads and time for which virtual machines (VMs) are provisioned. Multiple tenants submit requests for provisioning of VMs resulting in arbitrary workload, which is formed by combining various types of applications, such as high-performance computing, multi-tenant software as a service and web applications that utilize the resources simultaneously. The multiple tenants (e.g., large-scale enterprises, organizations that serve several end-users) establish SLAs with the resource provider to formalize the QoS requirements and the provider pays a penalty in case of SLO violations.

The approach to dynamic resource adaptation follows a distributed model and considers problems, such as composite service utility underutilization and overutilization detection, VM selection and placement. The mitigation of these problems improves the scalability of the system, since the service utility underutilization / overutilization detection and VM placement algorithms are executed locally by each compute host. The objective is the continuous monitoring of the CPU utilization and detecting service utility underutilization and overutilization conditions. In case of overutilization, the local manager running on the overutilized service utility initiates the selection algorithm to determine which VMs to offload a task. The global manager resides on the master node and collects information from the local managers to maintain the overall view of the system's resource utilization. Based on the decisions made by the local managers, the global manager issues resource allocation commands

UNIVERSITY of NICOSIA

to optimize the VM placement and, therefore, the application's performance avoiding the possibility of degradation.

The elasticity debt quantification models are introduced on the mobile cloud-based service utility level when a task is offloaded, scheduled and executed on cloud. A set of collocated mobile device users $N = \{1,2, \dots, N\}$ is considered on cloud-supported, always-on mobile services, which are leased off. The end-users are also classified into segments, e.g., corporate, premium and basic users. Service subscription, resource selection and allocation are subject to:

1. quality of service (QoS) restrictions in terms of delay-sensitivity, network bandwidth, latency and throughput [270],
2. predicted number of active end-users,
3. fluctuations in demand that might lead to service-level objectives (SLO) violations and accumulated elasticity debt.

In the view of this assumption, adaptation decisions are taken to adjust the resource provisioning when new user requests occur, requiring a standard level of QoS even during mobility. The set of users N remains unchanged during a computational offloading period, while it might change across different periods due to fluctuations in the number of mobile device users. We illustrate the model formulation in Figures 4.1 and 4.2 using two UML (Unified Modeling Language) activity and sequence diagrams, respectively. The numerical results are classified into two classes – i.e., positive and negative values as monetary units – while the elasticity debt metrics on mobile cloud-based service level are proposed from non-linear (section 4.2.2.) and linear (section 4.2.3) perspectives.

The proposed quantitative models aim to predict, estimate and quantify the elasticity debt when leasing cloud-supported mobile services under the assumption that fluctuations or a linear growth in the resource demand occur. The hypothesis is that the service selection decision is related to the service utility capacity and it is made with respect to the predicted non-linear or linear growth in the number of end-users over the period of λ-years and the way the elasticity debt is gradually paid off. The model formulation is based on a cost-benefit analysis, measuring the amount of profit not earned due to the underutilization of a given service and considering the probability of a service utility overutilization that would lead to accumulated elasticity debt difficult to be managed.

UNIVERSITY of NICOSIA

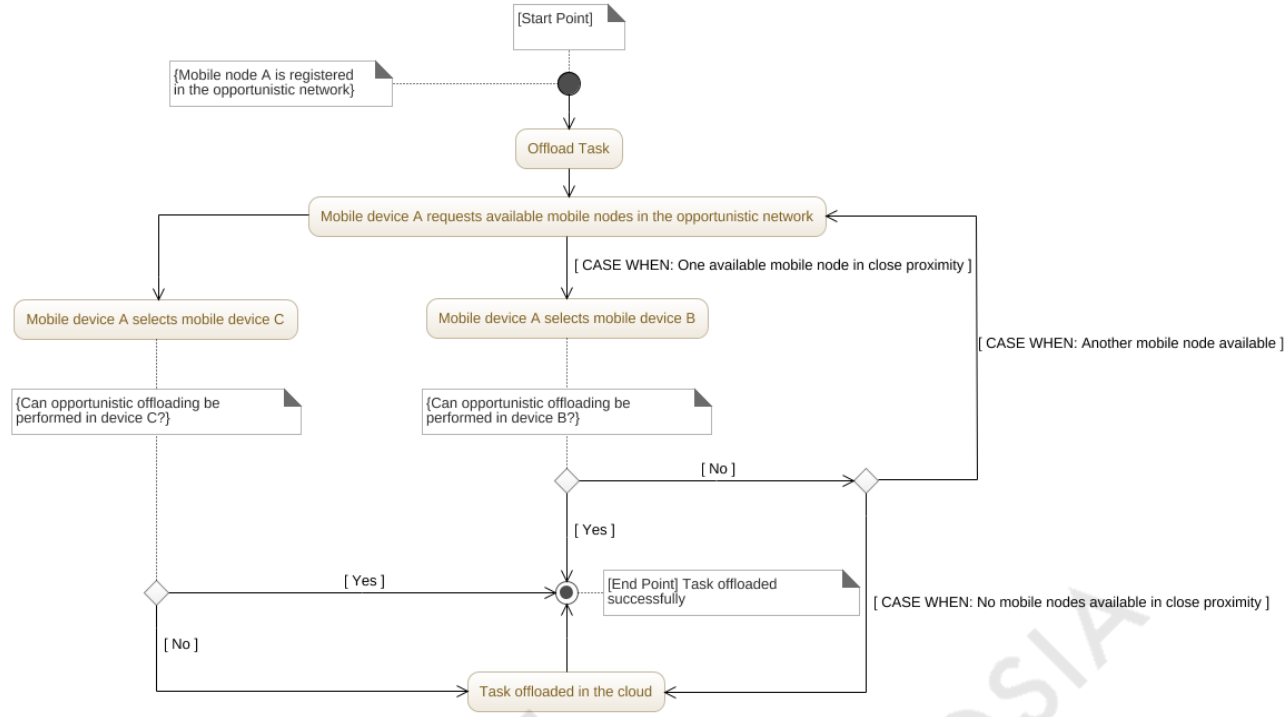


Figure 4.1. UML activity diagram: Mobile opportunistic offloading

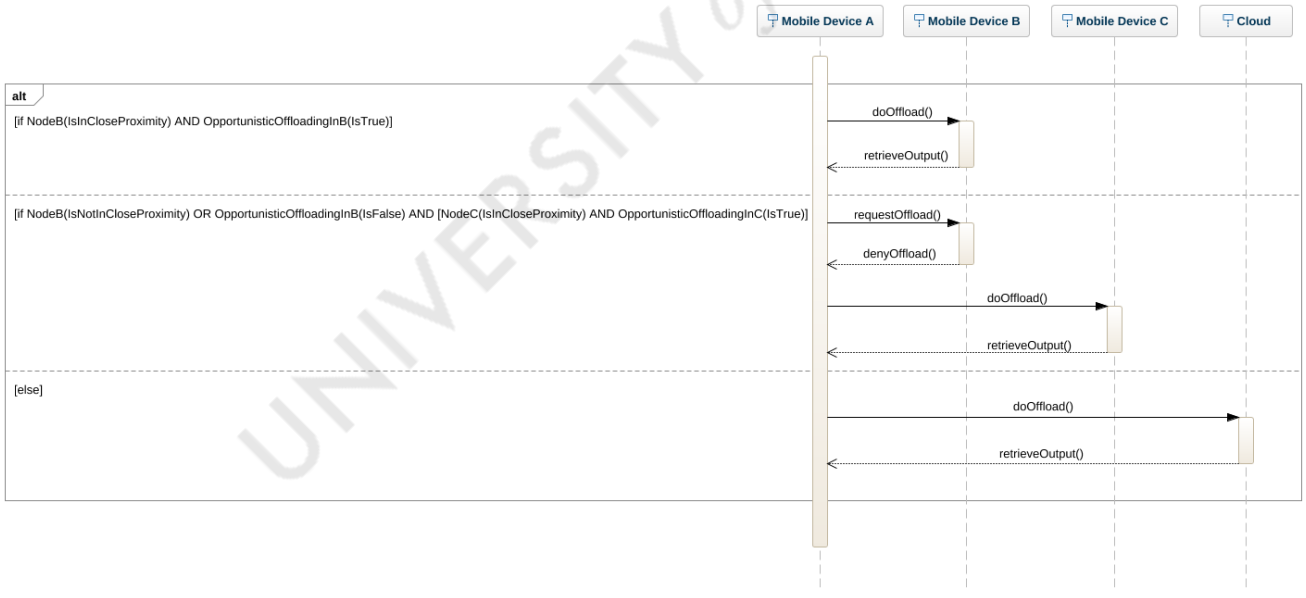


Figure 4.2. UML sequence diagram: Mobile opportunistic offloading

UNIVERSITY of NICOSIA

Any service to be leased off is evaluated with respect to the following modeling assumptions and statements:

1. The modeling period of λ-years is examined, aiming to provide insights into the ROI and the time the elasticity debt will be totally paid off. The optimal condition is zero (zero monetary units as the costs are balanced against the benefits), where the service capacity fully meets the evolving market needs.
2. Any candidate cloud-supported mobile service is subscription-based, while the SLA contract between the involved parties (i.e., cloud provider and client) includes charges for servicing an end-user in the cloud. The pricing schemes and billing cycles vary over the period of λ-years due to the variation in the demand (cloud scalability and elasticity).
3. Fluctuations or a linear growth in the number of active users are predicted over the modeling period of λ-years, increasing the total cost for servicing an end-user in the cloud in order to guarantee the QoS and QoE requirements. The cost variations resulting from the annual variation in the demand are composed of document and data storage, technical support, maintenance service, network bandwidth and server costs.
4. The flexibility of a mobile service requires its adaptability to meet evolving market needs, such as the fluctuations or the linear increase in the number of users.
5. The cloud-based, always-on mobile services are usually sensitive to network bandwidth and latency. The additional network bandwidth cost, which is a result of the increase in the number of users, is expected to satisfy the outbound network traffic demands in order to avoid delays. Likewise, the additional server cost includes those costs that derive from the additional CPU cores and the amount of memory required for processing, as more processing capacity is added to handle anticipated transaction volumes.
6. The offered web services have comparable functional requirements, while the non-functional ones can be either comparable or not (i.e., capacity, flexibility and maintainability); however, the cloud-based mobile services are differentiated with respect to the capacity in terms of the maximum number of active users that can be supported.

Two possible types of elasticity debt calculation results are encountered, when selecting a cloud-based mobile service to lease off:

1. Positive elasticity debt results: It reveals the service underutilization and the probability to satisfy a possible future increase in the number of active end-users against the unused capacity.

2. Negative elasticity debt results: It points out the service overutilization and, hence, possible SLO violations. The need to abandon/terminate the existing service and switch to a more flexible service – in terms of capacity, QoS and QoE restrictions – is motivated.

The optimization of costs and profits in cloud storage for big data analytics as a service platforms is additionally examined. Novel cost and benefit analysis models are introduced towards the evaluation of a big data as a service (BDaaS) model against data warehouse appliances (DWH). The model formulation is based on the lease of cloud storage capacity under the assumption that fluctuations in the demand for storage capacity occur due to the increasingly large amount of data, adopting a non-linear and asymmetric approach (section 4.2.4.). Also, a novel methodology for capitalizing earnings on cloud storage level is presented in section 4.2.5. In this case, the cloud-inspired quantitative modeling is based on a cost-benefit appraisal under the assumption that the demand curves for storage capacity are linear. The size of the additional costs is explicitly affected by the selection decision and the maximum cloud storage capacity, examining the necessity of possible storage upgradation in the long run.

The cloud-inspired, quantification models aim to reveal the benefits of the adoption of a big data as a service model against the conventional high-performance data warehouse appliances. The lease of cloud storage is considered, when developing the mathematical formulas, and the model formulation is based on cost and benefits analyses, measuring the amount of profit not earned due to the underutilization of the storage capacity under the assumption that fluctuations (non-linear and asymmetric approach) or linear (symmetric) growth in the requests for storage capacity occur. The benefits stemming from the selection decision motivate the evaluation of different data-driven models, examining the option of reinvesting the earnings on cloud storage level (i.e., additional storage required in the long run) through a big data-as-a-service framework.

Since the cloud storage and computing capacity are resources to be leased off, the following facts and modeling assumptions are taken into account:

1. The cloud storage is subscription-based and the billing cycles vary over the period of *l*-years due to the fluctuations or linear increase in the demand for storage capacity (i.e., gigabyte per month). A pricing scheme is witnessed in [271].
2. Fluctuations (non-linear) or linear increase in the demand for storage capacity are predicted over the period of *l*-years, increasing the total cost in order to guarantee the Quality of Service (QoS) and Quality of Experience (QoE) requirements.

3. The scalability and elasticity provided by the cloud are taken into account when developing the model, as the predicted cost variations – resulting from the annual variation in the demand for storage capacity – consist of data and document (unstructured) storage, maintenance service, network, on-demand I/O, operations (i.e., service requests), server and technical support costs.
4. The total network cost consists of costs related to bandwidth usage, egress and data transfer between regional and multi-region locations. As the cloud-based, always-on mobile services are usually sensitive to network bandwidth and latency [190], the additional network cost is expected to satisfy the outbound network traffic demands in order to avoid delays. The custom metadata headers are accounted for in the monthly storage and bandwidth usage.
5. The additional on-demand I/O cost intends to increase the throughput [201] when the content retrieval from a bucket should be faster than the default.
6. The additional server cost includes those costs associated with the additional CPU cores and the amount of memory required for processing.

Two possible types of benefits / cost difference numerical results are encountered, when leasing cloud storage:

1. Positive numerical results: Underutilization of the storage capacity and probability to meet the needs of a possible increase in the demand in long run.
2. Negative numerical results: Immediate need for upgradation. This need stimulates additional costs; however, the total amount of accumulated cost in data warehouse appliances is not comparable, as the earnings by adopting a big data as a service model can be reinvested on the additional storage required, maximizing the return on investment.

4.2.2. Elasticity Debt Quantification: Non-Linear Mathematical Modeling

Given the unavoidable gaps in resource provisioning between ideal and actual adaptation decisions, the interest is associated with the lack of effective, well-compromised trade-off decisions. The quantification of this interest is motivated as a result of the unused service capacity or SLO violation during the overutilization or underutilization states of the composite

service component utility, adopting non-linear variations in request workload. In this direction, the elasticity debt (ED) is computed from an asymmetric quantitative viewpoint, examining the lease of a cloud-supported mobile service under the assumption that fluctuations in the number of active end-users occur. The modeling for quantifying the elasticity debt during the first year (i.e., equation 4.1) and from the second year and onwards respectively (i.e., equation 4.2), takes the form as shown below. The proposed elasticity debt minimization incentive mechanism (non-linear approach) uses the algorithm shown in Table 4.1. The metrics and descriptions for the variables used to develop these mathematical models are presented in Table 4.3.

$$ED_1 = 12 * \left[ppm * (U_{max} - U_{curr}) - Cu_{/m} * (U_{max} - U_{curr})\right]$$
$$= 12 * (U_{max} - U_{curr}) * \left(ppm - Cu_{/m}\right) \quad (4.1)$$

$$ED_i = 12 * \{K_{i-2} * [U_{max} - L_{i-2}] - M_{i-2} * [U_{max} - L_{i-2}]\}$$
$$= 12 * (U_{max} - L_{i-2}) * (K_{i-2} - M_{i-2}), \text{s.t.} \, i > 1 \quad (4.2)$$

s.t.

$$K_0 = (1 + \Delta_1\%) * ppm$$

$$K_i = K_{i-1} * (1 + \Delta_{i+1}\%), \text{s.t.} \, i > 0$$

$$L_0 = (1 + \beta_1\%) * U_{curr}$$

$$L_i = L_{i-1} * (1 + \beta_{i+1}\%), \text{s.t.} \, i > 0$$

$$M_0 = (1 + VoC_1\%) * Cu_{/m}$$

$$M_i = M_{i-1} * (1 + VoC_{i+1}\%), \text{s.t.} \, i > 0$$

$$VoC_i\% = a_i\% + \gamma_i\% + \theta_i\% + \mu_i\% + \sigma_i\% + \eta_i\%, \text{s.t.} \, i > 0$$

Table 4.1. Algorithm Used in Elasticity Debt Minimization Mechanism (Non-Linear)

Algorithm 4.1. Pseudocode Implementation of Elasticity Debt Quantification (EDQ)

// This algorithm monitors and provides an analysis of the current state of the elasticity
// debt and predicts the accumulated state in case of parallelization of requests for cloud
// resources.

1: procedure EDQ (i, U_{max}, U_{curr}, ppm, $Cu_{/m}$, n, λ, $\Delta\%$, $\beta\%$, $VoC\%$)
2: sequential input (i, U_{max}, U_{curr}, ppm, $Cu_{/m}$, n, λ, $\Delta\%$, $\beta\%$, $VoC\%$)
3: if ($i = 1$) then
4: $ED[i] \leftarrow 12 * (U_{max} - U_{curr}) * \left(ppm - Cu_{/m}\right)$
5: end if
6: for ($i = 2, \dots, \lambda$) do // increasing the index of year to provide the EDQ output
7: if ($n = 1$) then
8: $K[0] \leftarrow (1 + \Delta_1\%) * ppm$
9: $L[0] \leftarrow (1 + \beta_1\%) * U_{curr}$
10: $M[0] \leftarrow (1 + VoC_1\%) * Cu_{/m}$
11: $ED[i] \leftarrow 12 * (U_{max} - L[0]) * (K[0] - M[0])$
13: else if ($n > 1$) then
14: $K[n-1] \leftarrow K[n-2] * (1 + \Delta_n\%)$
15: $L[n-1] \leftarrow L[n-2] * (1 + \beta_n\%)$
16: $M[n-1] \leftarrow M[n-2] * (1 + VoC_n\%)$
17: $ED[i] \leftarrow 12 * (U_{max} - L[n-2]) * (K[n-2] - M[n-2])$
18: end if
19: end for
20: return $ED[i]$
21: end procedure

UNIVERSITY of NICOSIA

4.2.3. Elasticity Debt Quantification: Linear Mathematical Modeling

Considering the unavoidable gaps in resource provisioning between ideal and actual adaptation decisions, the quantification of this interest is motivated as a result of the unused service capacity or SLO violation during the overutilization or underutilization states of the composite service component utility, adopting linear variations in request workload. In this section, a linear increase/growth in the number of active end-users is researched. The mathematical formulas for quantifying the elasticity debt when leasing cloud-supported mobile services (from a symmetric quantitative perspective) is given below (i.e., equation 4.3). The proposed elasticity debt minimization incentive mechanism (linear approach) uses the algorithm shown in Table 4.2. The metrics and descriptions for the variables used to develop these models are presented in Table 4.3.

$$
\begin{aligned}
ED_i = 12 * \Bigg\{ & \left(1 + \frac{\Delta\%}{\lambda}\right)^{i-1} * ppm * \left[U_{max} - (1+\beta\%)^{i-1} * U_{curr}\right] \\
& - \left(1 + \frac{\alpha\%}{\lambda} + \frac{\gamma\%}{\lambda} + \frac{\theta\%}{\lambda} + \frac{\mu\%}{\lambda} + \frac{\sigma\%}{\lambda} + \frac{\eta\%}{\lambda}\right)^{i-1} * Cu_{/m} \\
& * \left[U_{max} - (1+\beta\%)^{i-1} * U_{curr}\right] \Bigg\} \\
& = 12 * \left[U_{max} - (1+\beta\%)^{i-1} * U_{curr}\right] \\
& * \left[\left(1 + \frac{\Delta\%}{\lambda}\right)^{i-1} * ppm - \left(1 + \frac{\alpha\% + \gamma\% + \theta\% + \mu\% + \sigma\% + \eta\%}{\lambda}\right)^{i-1} \right. \\
& \left. * Cu_{/m}\right] \qquad (4.3)
\end{aligned}
$$

s.t. $1 \leq i \leq \lambda$.

UNIVERSITY of NICOSIA

Table 4.2. Algorithm Used in Elasticity Debt Minimization Mechanism (Linear)

Algorithm 4.2 Pseudocode Implementation of Elasticity Debt Quantification (EDQ)

// This algorithm provides an analysis of the current state of elasticity debt in mobile cloud
// computing environments, once a linear growth in the number of active users occurs.

1: procedure EDQ (U_{max}, $\beta\%$, i, U_{curr}, $\Delta\%$, λ, ppm, $\alpha\%$, $\gamma\%$, $\theta\%$, $\mu\%$, $\sigma\%$, $\eta\%$, $Cu_{/m}$)

2: sequential input (U_{max}, $\beta\%$, i, U_{curr}, $\Delta\%$, λ, ppm, $\alpha\%$, $\gamma\%$, $\theta\%$, $\mu\%$, $\sigma\%$, $\eta\%$, $Cu_{/m}$)

3: for ($i = 1, \dots, \lambda$) do // increasing the index of the year to provide EDQ output

$$\mathrm{ED}[i] \leftarrow 12 * \left[\mathrm{U_{max}} - (1 + \beta\%)^{\mathrm{i}-1} * \mathrm{U_{curr}}\right] * \left[\left(1 + \frac{\Delta\%}{\lambda}\right)^{\mathrm{i}-1} * \mathrm{ppm} - \left(1 + + \frac{\alpha\%+\gamma\%+\theta\%+\mu\%+\sigma\%+\eta\%}{\lambda}\right)^{\mathrm{i}-1} * \mathrm{Cu_{/m}}\right]$$

4: end for

5: return $ED[i]$

6: end procedure

Table 4.3. Elasticity Debt Quantitative Models: Variable Metrics & Definitions

Variable	Variable Definition	Metric
ED	The elasticity debt computations.	Monetary units
i	The λ-year index.	
λ	The modeling period.	Years
U_{max}	The maximum number of active mobile device users that can be supported.	

UNIVERSITY of NICOSIA

Variable	Variable Definition	Metric
K_0	The forming of the new subscription price in the second year, once the variation in the monthly subscription price is applied.	Monetary units
$\Delta_1\%$	The variation in the monthly subscription price in the second year.	Percentage
ppm	The initial monthly subscription price.	Monetary units
K_i	The forming of the new subscription price from the third year and onwards, once the variation in the monthly subscription price is applied.	Monetary units
$\Delta\%$	The average increase in the monthly subscription price over the period of λ-years.	Percentage
$\Delta_i\%$	The variation in the monthly subscription price from the third year and onwards.	Percentage
L_0	The forming of the number of mobile device users in the second year, once the variation in the demand is applied.	
$\beta\%$	The average annual increase in the number of mobile device users.	Percentage
$\beta_1\%$	The variation in the number of mobile device users in the second year.	Percentage
U_{curr}	The initial number of active mobile device users.	
L_i	The forming of the number of mobile device users from the third year and onwards, once the variation in the demand is applied.	

UNIVERSITY of NICOSIA

Variable	Variable Definition	Metric
$\beta_i\%$	The variation in the number of mobile device users from the third year and onwards.	Percentage
M_0	The forming of the cost for servicing an end-user in the cloud in the second year, once the monthly cost variation is applied.	Monetary units
$VoC_1\%$	The cost variation for servicing an end-user in the cloud in the second year.	Percentage
$Cu_{/m}$	The initial monthly cost for servicing a device user in the cloud.	Monetary units
M_i	The forming of the cost for servicing an end-user in the cloud from the third year and onwards, once the monthly cost variation is applied.	Monetary units
$VoC_i\%$	The total cost variation for servicing a mobile device user in the cloud from the third year and onwards.	Percentage
$a_i\%$	The variation in the monthly document storage cost.	Percentage
$\gamma_i\%$	The variation in the monthly data storage cost.	Percentage
$\theta_i\%$	The variation in the monthly technical support cost.	Percentage
$\mu_i\%$	The variation in the monthly maintenance service cost.	Percentage
$\sigma_i\%$	The variation in the monthly network bandwidth cost.	Percentage
$\eta_i\%$	The variation in the monthly server cost.	Percentage

UNIVERSITY of NICOSIA

4.2.4. Profit Optimization in Cloud Storage: Non-Linear Mathematical Modeling

The cost analysis (CA) modeling from the data warehouse appliance (DWH) viewpoint takes the form as shown in equation 4.4. The metrics and descriptions for the variables used to develop these mathematical models are presented in Table 4.5.

$$CA_i = 12 * \left(Cs_{/m} * S_{max}\right), 0 < i \leq l \; and \; S_{curr} \leq S_{max} \quad (4.4)$$

s.t. $Cs_{/m} = Cs_{/m_{(max)}} = C\alpha_{/m_{(max)}} + C\gamma_{/m_{(max)}} + C\eta_{/m_{(max)}} + C\theta_{/m_{(max)}} + C\kappa_{/m_{(max)}} + C\lambda_{/m_{(max)}} + C\mu_{/m_{(max)}} + C\sigma_{/m_{(max)}}$

As the benefits of cloud computing and big data as a service technology (i.e., scalability and elasticity) do not stand in data warehouse appliances, the adopted cost analysis approach does not examine the storage capacity currently used (S_{curr}), which involves metering usage and charging based on actual use. As a result, no cost variations apply due to the fluctuations in the demand for storage capacity as long as $S_{curr} \leq S_{max}$. The benefits and cost difference are always zero ($C_D = 0$) over the period of l-years. It is important to mention that in case of such an increase in the demand for storage capacity that $S_{curr} > S_{max}$, then incremental capacity should be added to the storage systems with overhead and downtime, triggering accumulated costs, not to mention that there are charges for more storage capacity than the actual usage.

On the contrary, the profit that is associated with the unavoidable gaps in cloud storage between ideal and actual adaptation decisions is investigated, adopting non-linear variations in request workload. Therefore, the non-linear cost analysis (CA) and benefits/cost difference (C_D) modeling from the big data as a service (BDaaS) perspective is given during the first year (i.e., equations 4.5 and 4.7) and from the second year and onwards (i.e., equations 4.6 and 4.8) as

$$CA_1 = 12 * \left(Cs_{/m} * S_{curr}\right) \quad (4.5)$$

$$CA_i = 12 * \left(\Delta_{i-2} * B_{i-2}\right), i \geq 2 \quad (4.6)$$

$$C_{D_1} = 12 * \left[Cs_{/m} * (S_{max} - S_{curr})\right] \quad (4.7)$$

$$C_{D_i} = 12 * [\Delta_{\iota-2} * (S_{max} - B_{i-2})], i \geq 2 \quad (4.8)$$

s.t. $Cs_{/m} = Cs_{/m_{(curr)}} = C\alpha_{/m_{(curr)}} + C\gamma_{/m_{(curr)}} + C\eta_{/m_{(curr)}} + C\theta_{/m_{(curr)}} + C\kappa_{/m_{(curr)}} + C\lambda_{/m_{(curr)}} + C\mu_{/m_{(curr)}} + C\sigma_{/m_{(curr)}}$

$$\Delta_0 = (1 + \delta_1\%) * Cs_{/m}$$

$$\Delta_i = (1 + \delta_{i+1}\%) * \Delta_{i-1}, i \geq 1$$

$$\delta_i\% = a_i\% + \gamma_i\% + \eta_i\% + \theta_i\% + \kappa_i\% + \lambda_i\% + \mu_i\% + \sigma_i\%, i \geq 1$$

$$B_0 = (1 + \beta_1\%) * S_{curr}$$

$$B_i = (1 + \beta_{i+1}\%) * B_{i-1}, i \geq 1$$

4.2.5. Profit Optimization in Cloud Storage: Linear Mathematical Modeling

On the contrary, the linear cost analysis (CA) and benefits (B) modelling from the big data as a service (BDaaS) point of view is given in the below equations 4.9 and 4.10, respectively. The benefits calculation procedure (linear variation approach) is also presented in algorithm 4.2 (in Table 4.4), derived from the profit associated with the unavoidable gaps in cloud storage between ideal and actual adaptation decisions. The metrics and descriptions for the variables used to develop these mathematical formulas are presented in Table 4.5.

UNIVERSITY of NICOSIA

$$CA_i = 12 * \left[\left(1 + \frac{\Delta\%}{l}\right)^{i-1} * Cs_{/m} * (1 + \beta\%)^{i-1} * S_{curr}\right] \quad (4.9)$$

$$B_i = 12 * \left\{\left(1 + \frac{\Delta\%}{l}\right)^{i-1} * Cs_{/m} * \left[S_{max} - (1 + \beta\%)^{i-1} * S_{curr}\right]\right\} \quad (4.10)$$

$with\ 0 < i \leq l\ and,$

$$Cs_{/m} = Cs_{/m_{(curr)}} = C\alpha_{/m_{(curr)}} + C\gamma_{/m_{(curr)}} + C\eta_{/m_{(curr)}} + C\theta_{/m_{(curr)}} + C\kappa_{/m_{(curr)}} + C\lambda_{/m_{(curr)}} + C\mu_{/m_{(curr)}} + C\sigma_{/m_{(curr)}}$$

$$\Delta\% = \alpha\% + \gamma\% + \eta\% + \theta\% + \kappa\% + \lambda\% + \mu\% + \sigma\%$$

Table 4.4. Algorithm Used for Profit Optimization in Cloud Storage (Linear)

Algorithm 4.2. Pseudocode Implementation for Benefits Modeling

```
// This algorithm aims to calculate the benefits at cloud storage level from a BDaaS
// perspective under the prediction that the demand curves for storage capacity increase in
// a linear manner.

1: procedure BenefitsCalculation (S_max, S_curr, i, Δ%, β%, l, Cs_/m, α%, γ%, η%, θ%,
κ%, λ%, μ%, σ%)
2:   sequential input (α%, γ%, η%, θ%, κ%, λ%, μ%, σ%)
       Δ% ← α% + γ% + η% + θ% + κ% + λ% + μ% + σ%
3:   return Δ%
4:   sequential input (S_max, S_curr, i, Δ%, β%, l, Cs_/m)
5:     for i = 1 to l do  // Increasing the index of the year to get the output element with
                          // respect to the year
       B[i] ← 12 * {(1 + Δ%/l)^(i-1) * Cs_/m * [S_max − (1 + β%)^(i-1) * S_curr]}
6:     end for
7:   return B[i]
8: end procedure
```

Table 4.5. Cost & Profit Optimization Quantitative Models: Variable Metrics & Definitions

Variable	Variable Definition	Metric
CA	The cost analysis calculation results.	Monetary units
C_D	The cost difference calculations.	Monetary units
B	The benefits calculations.	Monetary units
l	The modeling period.	
i	The index of the year.	
$Cs_{/m}$	The initial monthly cost for leasing cloud storage.	Monetary units
S_{max}	The maximum storage capacity.	Terabytes
S_{curr}	The storage currently used.	Terabytes
$\Delta\%$	The variation regarding the cost for leasing cloud storage for the *l*-year period of time.	Percentage
Δ_0	The cost formation for leasing cloud storage regarding the second year of the period of *l*-years, once the respective variation in the monthly cost is applied.	Monetary units
$\delta_1\%$	The variation regarding the cost for leasing cloud storage for the second year of the period of *l*-years.	Percentage
Δ_i	The cost formation for leasing cloud storage from the third year until the end of the period of *l*-years, once the respective variation in the monthly cost is applied.	Monetary units
$\delta_i\%$	The variation regarding the cost for leasing cloud storage from the third year until the end of the modeling period of *l*-years.	Percentage
B_0	The storage used during the second year of the modeling period of *l*-years, once the corresponding variation in the demand is applied.	
$\beta\%$	The increase in the demand for cloud storage capacity per year.	Percentage
$\beta_1\%$	The variation in the demand for storage capacity regarding the second year of the period of *l*-years.	Percentage
B_i	The storage used from the third year until the end of the modeling period of *l*-years, once the respective variation in the demand is applied.	
$\beta_i\%$	The variation in the demand for storage capacity from the third year until the end of the period of *l*-years.	Percentage
C_α	The data storage cost.	Monetary units
C_γ	The document storage cost.	Monetary units
C_η	The maintenance service cost.	Monetary units
C_θ	The network cost.	Monetary units
C_κ	The on-demand I/O cost.	Monetary units
C_λ	The operations cost.	Monetary units
C_μ	The server cost.	Monetary units
C_σ	The technical support cost.	Monetary units
$\alpha_i\%$	The variation in the monthly data storage cost.	Percentage
$\gamma_i\%$	The variation in the monthly document storage cost.	Percentage
$\eta_i\%$	The variation in the monthly maintenance service cost.	Percentage
$\theta_i\%$	The variation in the monthly network cost.	Percentage

UNIVERSITY of NICOSIA

Variable	Variable Definition	Metric
$\kappa_i\%$	The variation in the monthly on-demand I/O cost.	Percentage
$\lambda_i\%$	The variation in the monthly operations cost.	Percentage
$\mu_i\%$	The variation in the monthly server cost.	Percentage
$\sigma_i\%$	The variation in the monthly technical support cost.	Percentage

4.3. Performance Evaluation: Experimental Results, Analysis and Discussion

4.3.1. Elasticity Debt Quantification: Non-Linear Mathematical Modeling

Throughout the evaluation stage, a 5-year technical debt prediction period ($\lambda = 5$) has been examined, enabling to do a what-if analysis on different case scenarios, web services and lease options. Insights into the return on investment and the time the technical debt will be totally paid off are also provided. The following three mobile services are offered by the same provider and they are investigated with respect to their different features:

1. Corporate Service (C): High-capacity service – the maximum number of active users that can be supported is 18,000. The service is considered better than Premium and Basic in terms of QoS, QoE and non-functional requirements, elaborating on the high-priced strategy that is adopted – the estimated monthly cost for servicing an end-user in the cloud is high. The variations in the monthly cost (e.g., network bandwidth or server cost) due to the fluctuations in the number of users are different from services that provide lower quality and capacity.
2. Premium Service (P): Medium-capacity service – the maximum number of active users that can be supported is 11,000. This service is better than Basic in terms of QoS, QoE and non-functional requirements.
3. Basic Service (B): Low-capacity service – the maximum number of users that can be supported is 5,000.

The elasticity debt estimates are made over the 5-year modeling period for two different case scenarios. The probability of underutilization or overutilization of any of the three services is examined. The yearly variations in the demand for case scenarios 1 and 2 are presented in Table 4.6.

Table 4.6. Case Scenarios 1 & 2: Yearly Variation in Demand

Term	Variation in Demand
Case Scenario 1	
Year 1 to 2	$\beta_1\% = 12\%$
Year 2 to 3	$\beta_2\% = 40\%$
Year 3 to 4	$\beta_3\% = -18\%$
Year 4 to 5	$\beta_4\% = 35\%$
Case Scenario 2	
Year 1 to 2	$\beta_1\% = 20\%$
Year 2 to 3	$\beta_2\% = -10\%$
Year 3 to 4	$\beta_3\% = 60\%$
Year 4 to 5	$\beta_4\% = 65\%$

Towards a better understanding of the elasticity debt quantification, the calculations for the first two years would take the following form:

$$ED_1 = 12 * \left[ppm * (U_{max} - U_{curr}) - Cu_{/m} * (U_{max} - U_{curr})\right] = 12 * (U_{max} - U_{curr}) * \left(ppm - Cu_{/m}\right)$$

$$ED_2 = 12 * \left\{(1 + \Delta_1\%) * ppm * [U_{max} - (1 + \beta_1\%) * U_{curr}] - (1 + VoC_1\%) * Cu_{/m} * [U_{max} - (1 + \beta_1\%) * U_{curr}]\right\} = 12 * [U_{max} - (1 + \beta_1\%) * U_{curr}] * \left[(1 + \Delta_1\%) * ppm - (1 + VoC_1\%) * Cu_{/m}\right]$$

Having set the elasticity debt quantification rules, the data input in Tables 4.7, 4.8 and 4.9 are applied to formulas presented in section 4.2.2. The choice of these specific values and case scenarios enables to obtain accurate and comparable results, which reveal the overutilization of a service along with the appropriate web services to lease off in order to increase the return on investment. Tables 4.8 and 4.9 include the variations for the monthly subscription price and the

cost for servicing an end-user in the cloud, which are dependent on the variations in the number of active users, the service capacity and the QoS / QoE benefits. The obtained evaluation results are shown explicitly in Tables 4.10 and 4.11, while the flow and comparisons of the technical debt calculations over the 5-year modeling period are observed in Figures 4.3 and 4.4.

Table 4.7. Data Input for Formulas in Section 4.2.2

Variable Definition	Corporate (C)	Premium (P)	Basic (B)
Maximum number of active end-users that can be supported	$U_{max} = 18{,}000$	$U_{max} = 11{,}000$	$U_{max} = 5{,}000$
Initial number of active end-users	$U_{curr} = 3{,}000$	$U_{curr} = 3{,}000$	$U_{curr} = 3{,}000$
Initial monthly subscription price (USD)	$ppm = 12$	$ppm = 8$	$ppm = 6$
Monthly cost for servicing a user in the cloud (USD)	$Cu_{/m} = 7$	$Cu_{/m} = 4$	$Cu_{/m} = 3$

Table 4.8. Case Scenario 1 – Data Input: Variations in monthly subscription price and service cost

Variable Definition	Corporate (C)	Premium (P)	Basic (B)
Variation in the monthly subscription price	$\Delta_1\% = 0.3\%$ $\Delta_2\% = 1\%$ $\Delta_3\% = -0.4\%$ $\Delta_4\% = 1.8\%$	$\Delta_1\% = 0.4\%$ $\Delta_2\% = 1.2\%$ $\Delta_3\% = -0.5\%$ $\Delta_4\% = 2\%$	$\Delta_1\% = 0.5\%$ $\Delta_2\% = 1.5\%$ $\Delta_3\% = -0.4\%$ $\Delta_4\% = 2.5\%$
Variation in cost for servicing a user in cloud	$VoC_1\% = 2\%$ $VoC_2\% = 4\%$ $VoC_3\% = -2\%$ $VoC_4\% = 4.5\%$	$VoC_1\% = 3\%$ $VoC_2\% = 6\%$ $VoC_3\% = -3.1\%$ $VoC_4\% = 7\%$	$VoC_1\% = 5\%$ $VoC_2\% = 10\%$ $VoC_3\% = -4\%$ $VoC_4\% = 12\%$

UNIVERSITY of NICOSIA

Table 4.9. Case Scenario 2 – Data Input: Variations in monthly subscription price and service cost

Variable Definition	Corporate (C)	Premium (P)	Basic (B)
Variation in the monthly subscription price	$\Delta_1\% = 0.5\%$ $\Delta_2\% = -0.2\%$ $\Delta_3\% = 1.2\%$ $\Delta_4\% = 2.5\%$	$\Delta_1\% = 0.6\%$ $\Delta_2\% = -0.3\%$ $\Delta_3\% = 1.4\%$ $\Delta_4\% = 2.8\%$	$\Delta_1\% = 0.7\%$ $\Delta_2\% = -0.4\%$ $\Delta_3\% = 1.6\%$ $\Delta_4\% = 3\%$
Variation in cost for servicing a user in cloud	$VoC_1\% = 2.3\%$ $VoC_2\% = -1\%$ $VoC_3\% = 5\%$ $VoC_4\% = 10\%$	$VoC_1\% = 3.4\%$ $VoC_2\% = -1.5\%$ $VoC_3\% = 6\%$ $VoC_4\% = 12\%$	$VoC_1\% = 5.8\%$ $VoC_2\% = -2\%$ $VoC_3\% = 11\%$ $VoC_4\% = 20\%$

The first case scenario indicates that the Corporate and Premium services are always underutilized over the 5-year modeling period due to the positive elasticity debt results. Despite the fluctuations in the demand, a gradual payoff of the elasticity debt is observed as the number of monetary units for both services are less in the end of the period than in the beginning. The interpretation of the elasticity debt results for the Basic service reveals the underutilization of the service throughout the first four years. During the fifth year, the elasticity debt becomes zero – which constitutes the optimal condition – as it is totally cleared out. However, the elasticity debt result becomes negative until the end of the 5-year modeling period, pointing out that the service is overutilized due to the variation in the number of active end-users. The need for abandoning/terminating the existing service and switching to a more flexible one (i.e., in terms of service capacity) will be faced in the long run in order to meet the evolving market needs. The options of abandoning/termination and switching would create additional costs; the risk of entering into a new and accumulated elasticity debt in the future is high, having a significant impact on the return on investment. Having explained that any positive elasticity debt to be further incurred can be hardly managed, the lease of the Premium service constitutes the best selection decision for that case scenario in terms of gradual payoff of the elasticity debt, as the calculation results have the minimum positive values and the problem of overutilization does not lurk throughout the modeling period.

Table 4.10. Case Scenario 1: Elasticity Debt Results for Three Services

	Year 1	Year 2	Year 3	Year 4	Year 5
Corporate	900,000	860,129.28	754,802.22	819,821.78	724,748.39
Premium	384,000	358,652.16	284,164.97	330,501.54	258,685.88
Basic	72,000	56,678.4	9,432.16	37,978.09	-6,276.56

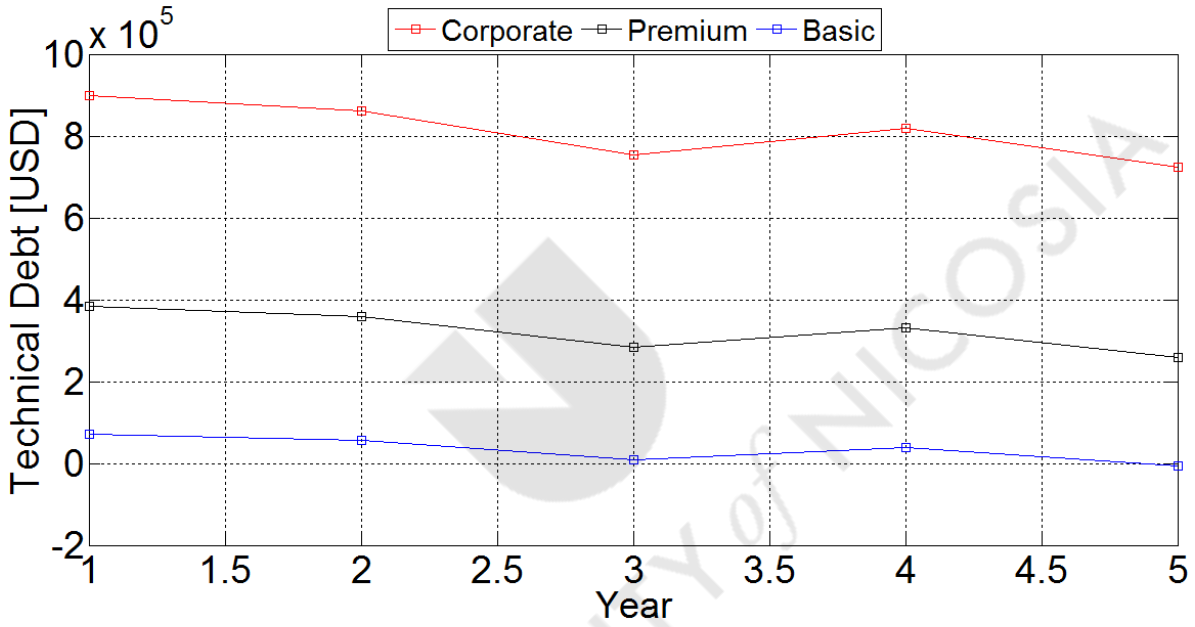


Figure 4.3. Case scenario 1: Flow of elasticity debt

The second case scenario points out that the Corporate and Premium services are always underutilized due to the positive elasticity debt results and the gradual payoff. On the contrary, although the Basic service is underutilized throughout the first three years, the elasticity debt calculation results become negative during the last two years. The elasticity debt is totally cleared out during the fourth year, achieving the optimal condition; however, the variation in the demand affect the flexibility and adaptability of the service, as it is overutilized until the end of the 5-year modeling period. The need for abandoning/terminating the existing service and switching to a more flexible one in terms of capacity will be faced again in the future. Likewise, the options of abandoning/termination and switching entail a new and accumulated elasticity debt. The lease of the Premium service constitutes the most appropriate selection

UNIVERSITY of NICOSIA

decision for that case scenario in terms of gradual payoff of the elasticity debt, as the calculation results have the minimum positive values. It is significant to mention that in case that all three services were overutilized, the options of abandoning and switching would be inevitable. Hence, the need to examine other more flexible services in terms of service capacity/utility from the same or different cloud service providers would be motivated to meet the demand requirements and avoid entering into a new and accumulated elasticity debt in the long run.

Table 4.11. Case Scenario 2: Elasticity Debt Results for Three Services

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
Corporate	900,000	846,547.2	876,122.31	728,428.28	487,045.75
Premium	384,000	347,385.6	367,814.32	266,451.37	103,553.12
Basic	72,000	48,182.4	61,402.43	-5,876.46	-91,867.48

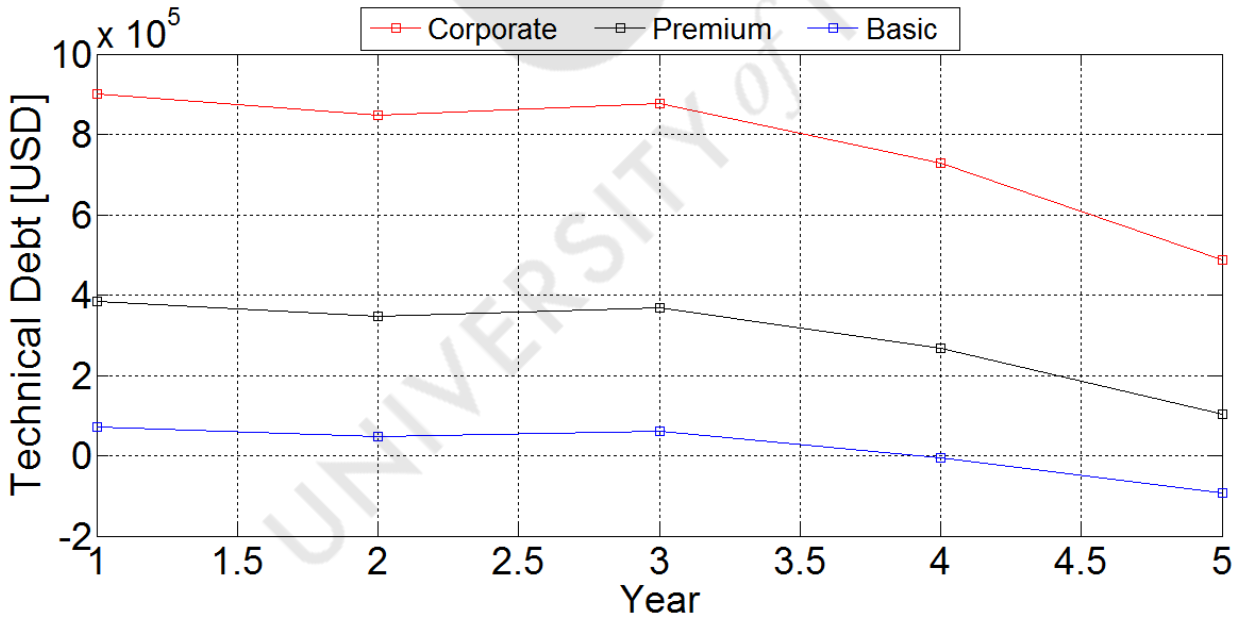


Figure 4.4. Case scenario 2: Flow of elasticity debt

UNIVERSITY of NICOSIA

4.3.2. Elasticity Debt Quantification: Linear Mathematical Modeling

A 4-year elasticity debt prediction period is examined, performing a what-if analysis on different case scenarios, web services and lease options. The following three cloud-supported mobile services are offered by the same provider and they are investigated with respect to the different features they have:

1. Service A: Maximum number of active users that can be supported is 15,000. More flexible than services B and C below in terms of QoS, QoE and non-functional requirements, such as the performance and maintainability. High-priced strategy is adopted as the monthly cost for servicing an end-user in the cloud is high. Once a linear increase in the number of users occurs, small-scale variations in the monthly cost (for instance, network bandwidth or server costs) are observed compared to those variations witnessed in services that provide lower quality and service capacity.
2. Service B: Maximum number of active users that can be supported is 10,000. More flexible than service C in terms of QoS, QoE and non-functional requirements.
3. Service C: Maximum number of active users that can be supported is 5,000. Lower-quality service compared to the other offered services.

The technical debt estimates are made against three different case scenarios over a 4-year modeling period, investigating the probability of overutilization of any of the three services under the assumption of an annual and linear growth in the number of users (i.e., variable $\beta\%$). The first case scenario forecasts a 10% annual increase in the number of users, the second one adopts a 55% annual increase in the demand, while the third one examines an 80% annual increase in the number of end-users.

Towards the elasticity debt quantification when leasing cloud-based mobile services, the data input shown in Table 4.12 are applied to the formulas presented in section 4.2.3. The choice of these specific values and case scenarios enables to obtain accurate and comparable results, which reveal the overutilization of a service and the most cost-effective services to lease off in order to avoid accumulated costs and the risk of entering into new elasticity debt in the long run. The evaluation results are presented in Tables 4.13, 4.14 and 4.15, whereas the flow and comparisons of the elasticity debt over the 4-year modeling period are indicated in Figures 4.5, 4.6 and 4.7, respectively.

Table 4.12. Data Input for Formulas in Section 4.2.3

Variable Definition	Service A	Service B	Service C
Modeling period of λ-years	$\lambda = 4$	$\lambda = 4$	$\lambda = 4$
Maximum number of active end-users that can be supported	$U_{max} = 15{,}000$	$U_{max} = 10{,}000$	$U_{max} = 5{,}000$
Initial number of end-users	$U_{curr} = 2{,}000$	$U_{curr} = 2{,}000$	$U_{curr} = 2{,}000$
Initial monthly subscription price (USD)	$ppm = 15$	$ppm = 10$	$ppm = 8$
Average increase in the monthly subscription price over the modeling period of λ-years	$\Delta\% = 2\%$	$\Delta\% = 2\%$	$\Delta\% = 2\%$
Initial monthly cost for servicing an end-user in cloud (USD)	$Cu_{/m} = 6$	$Cu_{/m} = 3$	$Cu_{/m} = 2$
Average increase in the monthly document storage cost over the modeling period of λ-years	$\alpha\% = 1\%$	$\alpha\% = 1.5\%$	$\alpha\% = 2\%$
Average increase in the monthly data storage cost over the period of λ-years	$\gamma\% = 0.5\%$	$\gamma\% = 1\%$	$\gamma\% = 2\%$
Average increase in the monthly technical support cost over the period of λ-years	$\theta\% = 0.5\%$	$\theta\% = 0.5\%$	$\theta\% = 1\%$
Average increase in the monthly maintenance service cost over the period of λ-years	$\mu\% = 0.5\%$	$\mu\% = 1\%$	$\mu\% = 2\%$
Average increase in the monthly network bandwidth cost over the modeling period of λ-years	$\sigma\% = 0.5\%$	$\sigma\% = 1\%$	$\sigma\% = 2\%$
Average increase in the monthly server cost over the period of λ-years	$\eta\% = 1\%$	$\eta\% = 2\%$	$\eta\% = 2\%$

The first case scenario indicates that services A, B and C are always underutilized over the 4-year modeling period due to the positive elasticity debt results. The elasticity debt is gradually paid off, as the number of monetary units is constantly decreasing due to the mild linear yearly increase in the number of users (10%). Despite the fact that services A and B are

considered better than C in terms of QoS and non-functional requirements and more flexible to meet the evolving market needs, the lease of service C is recommended for that case scenario, as the market needs are met over the 4-year modeling period and there are also reduced costs compared to services A and B. There would not also be a risk of entering into a new accumulated elasticity debt in the future, as the problem of overutilization along with the options of abandoning/terminating and switching do not lurk.

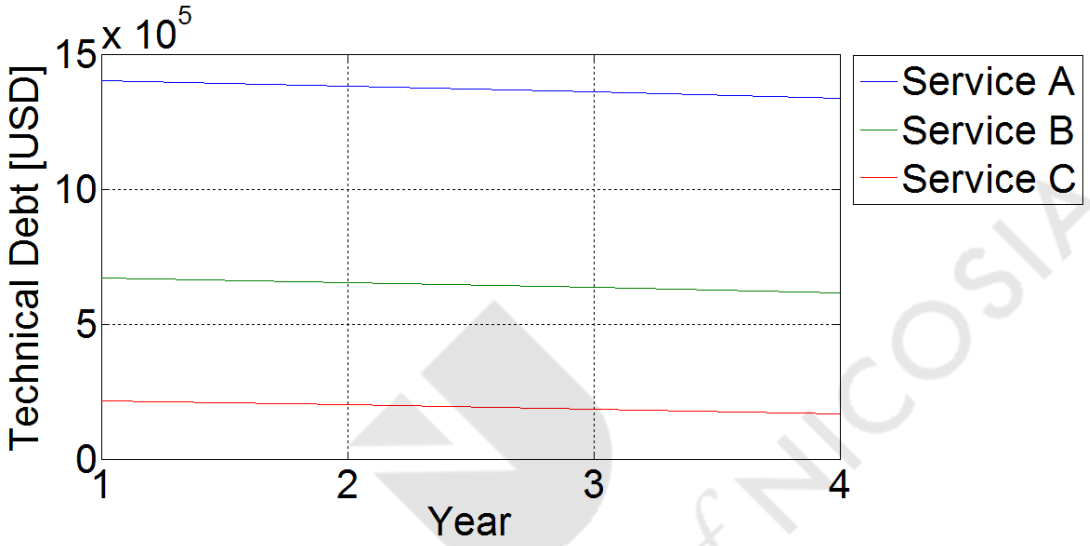


Figure 4.5. Case Scenario 1 – 10% annual increase in the demand: Flow of elasticity debt results over the 4-year modeling period

Table 4.13. Case Scenario 1 – 10% annual increase in the number of active users: Elasticity debt results over the 4-year modeling period

	Year 1	**Year 2**	**Year 3**	**Year 4**
Service A	1,404,000	1,384,704	1,363,135	1,339,066
Service B	672,000	654,966	636,204.4	615,553.6
Service C	216,000	201,096	184,790.6	166,961.9

The second case scenario points out that services A and B are always underutilized over the 4-year modeling period, because of the positive elasticity debt calculations that are observed.

For both services, a gradual payoff of the elasticity debt is witnessed as a result of the continuous decrease in the number of monetary units. The interpretation of the elasticity debt results concerning service C reveals underutilization for the first three years. During the fourth year, the elasticity debt becomes zero (optimal condition), pointing out that it is totally cleared out. The elasticity debt results become negative until the end of the modeling period, indicating that the service is overutilized due to the assumption of the linear growth in the number of active end-users. Hence, the need for abandoning/terminating the existing service and switching to a more flexible service in terms of capacity will be faced in the long run to meet the evolving market needs. The options of abandoning/termination and switching create additional costs and the risk of entering into a new and accumulated elasticity debt in the future is high. Having explained that any positive elasticity debt to be further incurred can be hardly managed, the lease of service B is the most cost-effective option for that case scenario in terms of return on investment and gradual payoff of the elasticity debt, as the calculation results have the minimum positive values, not to mention that the problem of overutilization does not lurk over the 4-year modeling period.

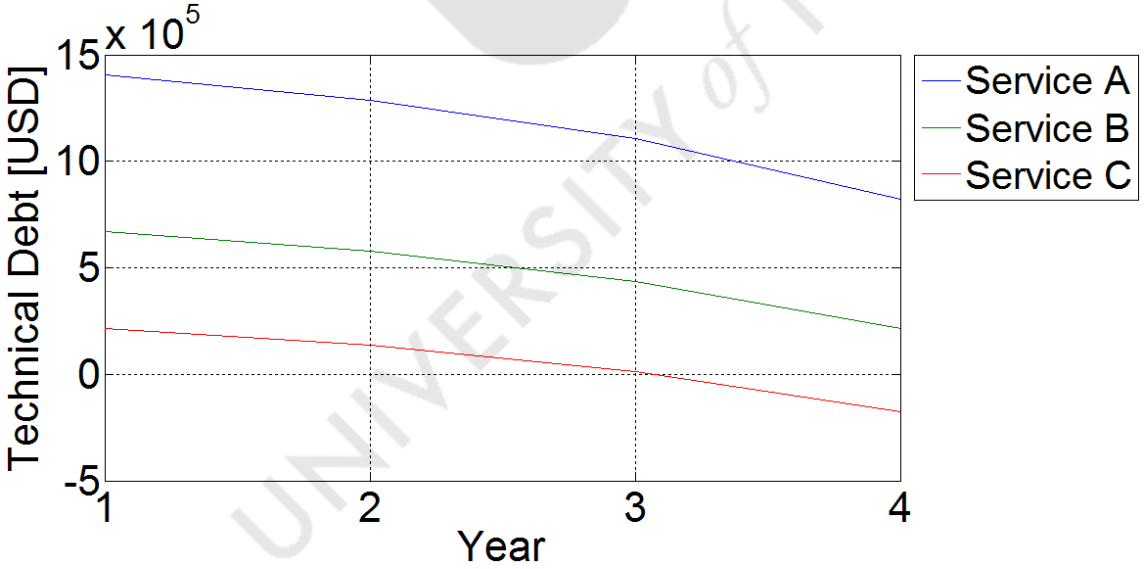


Figure 4.6. Case Scenario 2 – 55% annual increase in the demand: Flow of the elasticity debt results over the 4-year modeling period

UNIVERSITY of NICOSIA

Table 4.14. Case Scenario 2 – 55% annual increase in the number of active users: Elasticity debt results over the 4-year modeling period

	Year 1	**Year 2**	**Year 3**	**Year 4**
Service A	1,404,000	1,287,342	1,104,703	819,659.7
Service B	672,000	579,393	436,026.6	214,097.4
Service C	216,000	136,458	13,966.7	-174,799

The third case scenario demonstrates that service A is still underutilized over the modeling period and the elasticity debt is gradually paid off, despite the sheer annual increase in the number of active users (80%). As far as it concerns service B, the analysis of the elasticity debt results shows underutilization of the service from the first until the end of the third year. During the fourth year, the elasticity debt becomes zero (i.e., optimal condition), revealing that it is totally cleared out. However, the elasticity debt results become negative until the end of the modeling period as a result of the linear growth in the number of active end-users, disclosing that the service is overutilized. On the other hand, service C is underutilized the first two years. During the third year, the elasticity debt is totally paid off, while the elasticity debt results become negative until the end of the modeling period. The negative results imply the overutilization of the service; hence, a sheer increase in the demand (80%) will motivate the need to abandon/terminate either service B or C and switch to a more flexible capacity service. The risk of entering into a new and accumulated elasticity debt in the future is high, as these options would create additional costs. The analysis of the results for this case scenario points out that the most cost-effective cloud-supported mobile service to lease off is A, as the problem of overutilization does not lurk over the 4-year modeling period, the elasticity debt is gradually paid off and the risk of entering into a new elasticity debt in the long run is low. The flexibility of this particular service to adapt to the evolving market needs and the fact that the users will be able to enjoy a better service in terms of QoS / QoE should be also considered during the selection decision process.

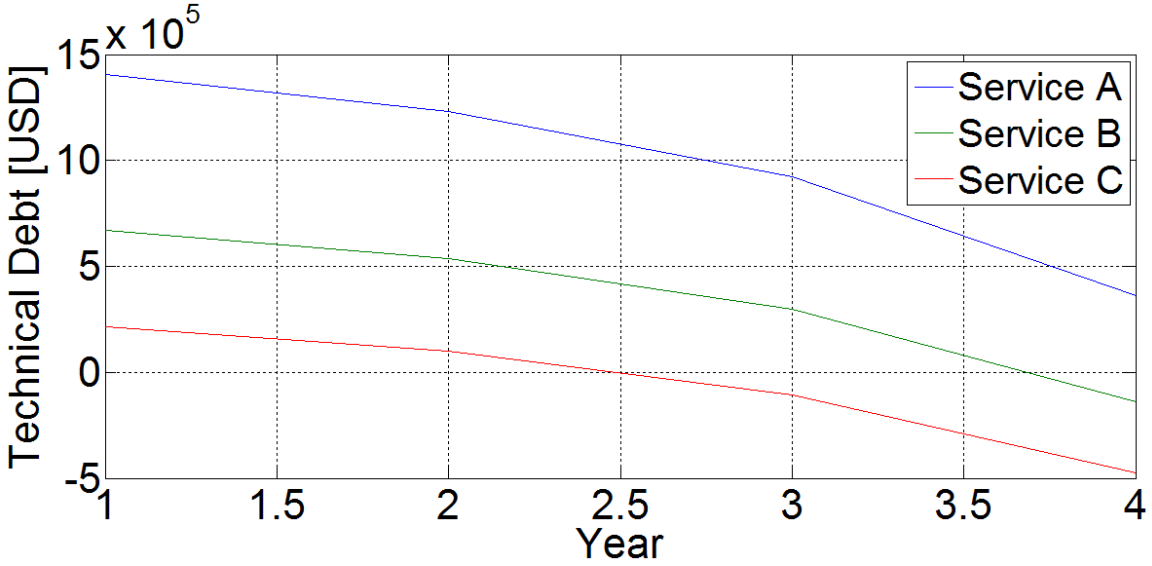


Figure 4.7. Case Scenario 3 – 80% annual increase in the demand: Flow of the elasticity debt results over the 4-year modeling period

Table 4.15. Case Scenario 3 – 80% annual increase in the number of active users: Elasticity debt results over the 4-year modeling period

	Year 1	**Year 2**	**Year 3**	**Year 4**
Service A	1,404,000	1,233,252	923,204.2	362,062.3
Service B	672,000	537,408	295,440.6	-139,585.9
Service C	216,000	100,548	-106,003.9	-475,891.3

4.3.2.1. Cost-Benefit Quantification: Linear Modeling Approach

The calculations are necessary towards the investigation whether a linear growth in the number of active end-users could risk the incurrence of accumulated costs in the long run, leading to options such as those of abandoning/termination and switching. They also provide a deeper understanding of the progress of different case scenarios when leasing cloud-based services. The cost-benefit data can be exploited in order to make strategic lease decisions as it predicts the underutilization or overutilization of a service and reveals the optimal condition (i.e., zero monetary units) by comparing benefits and costs for each case scenario.

UNIVERSITY of NICOSIA

In this direction, a 5-year cost-benefit forecast ($\lambda = 5$) has been made prior to mobile cloud adoption, enabling to do a what-if analysis on four different case scenarios and lease options. The following three different services are examined:

1. Corporate Service (C): High-capacity service – maximum number of active users that can be supported is 12,000. This service is considered better and more flexible than the Premium and Basic in terms of QoS, QoE and non-functional requirements. The monthly cost for servicing an end-user in the cloud is higher, while the variations in the monthly cost, such as the network bandwidth or server costs, are lower compared to services that have lower quality and capacity.
2. Premium Service (P): Medium-capacity service – maximum number of active users that can be supported is 9,000. This service is better than the Basic in terms of QoS, QoE and non-functional requirements.
3. Basic Service (B): Low-capacity service – maximum number of users that can be supported is 6,000.

The cost-benefit estimates are calculated for four different case scenarios, researching the probability of overutilization of any of the three services. The assumption is based on a predicted linear growth in the number of active end-users (variable $\beta\%$) and the adopted case scenarios are presented in detail below:

1. Case Scenario 1: Low-density annual increase in the demand – 20%.
2. Case Scenario 2: Medium-density annual increase in the number of users – 50%.
3. Case Scenario 3: High-density annual increase in the demand – 60%.
4. Case Scenario 4: High-density annual increase in the number of end-users – 70%.

Towards quantifying the cost-benefit appraisal, the data input shown in Tables 4.16 and 4.17 are applied to the algorithm presented in section 4.2.3. The choice of these case scenarios enables to obtain comparable results, which may reveal the overutilization state of a cloud-supported mobile service and point out those services to lease off in order to increase the return on investment. The numerical results are presented thoroughly in Tables 4.18, 4.19, 4.20 and 4.21, while the flow and comparisons of the cost-benefit estimates over the 5-year modeling period are observed in Figures 4.8, 4.9, 4.10 and 4.11, respectively.

UNIVERSITY of NICOSIA

Table 4.16. Data Input to Algorithm in Section 4.2.3

Variable Definition	Corporate (C)	Premium (P)	Basic (B)
Maximum number of active end-users to be supported	$U_{max} = 12{,}000$	$U_{max} = 9{,}000$	$U_{max} = 6{,}000$
Initial number of active users	$U_{curr} = 1{,}500$	$U_{curr} = 1{,}500$	$U_{curr} = 1{,}500$
Initial monthly subscription price (USD)	$ppm = 10$	$ppm = 7$	$ppm = 5$
Average increase in the monthly subscription price over the modeling period of λ-years	$\Delta\% = 2\%$	$\Delta\% = 2\%$	$\Delta\% = 2\%$
Initial monthly cost for servicing a user in cloud (USD)	$Cu_{/m} = 4$	$Cu_{/m} = 2$	$Cu_{/m} = 1$

Table 4.17. Data Input: Variations for service cost in cloud

Variable Definition	Corporate (C)	Premium (P)	Basic (B)
Average increase in the monthly document storage cost over the modeling period of λ-years	$\alpha\% = 1.5\%$	$\alpha\% = 2\%$	$\alpha\% = 2\%$
Average increase in the monthly data storage cost over the modeling period of λ-years	$\gamma\% = 0.5\%$	$\gamma\% = 1\%$	$\gamma\% = 1.5\%$
Average increase in the monthly maintenance service cost over the modeling period of λ-years	$\mu\% = 1\%$	$\mu\% = 2\%$	$\mu\% = 3\%$
Average increase in the monthly network bandwidth cost over the period of λ-years	$\sigma\% = 1\%$	$\sigma\% = 1.5\%$	$\sigma\% = 2.5\%$

Variable Definition	Corporate (C)	Premium (P)	Basic (B)
Average increase in the monthly server cost over the modeling period of λ-years	$\eta\% = 1\%$	$\eta\% = 1.5\%$	$\eta\% = 3\%$

The first case scenario points out that the Corporate, Premium and Basic services are always underutilized over the 5-year modeling period due to the positive cost-benefit calculation results. The mild annual increase in the demand (20%) enables the simultaneous increase in the return on investment, as the monetary units are constantly decreasing over the period for the three services. The Basic service constitutes the most cost-effective lease option for that case scenario as the calculation results have the minimum positive values and they are closer to the optimal condition (zero threshold). Hence, the costs that derive from the unused capacity are more balanced against the benefits. The lease of the Basic service entails that there are reduced costs compared to the other two candidate services and there would not also be a risk of entering into new accumulated costs in the future as the problem of service utility overutilization does not lurk over the 5-year modeling period.

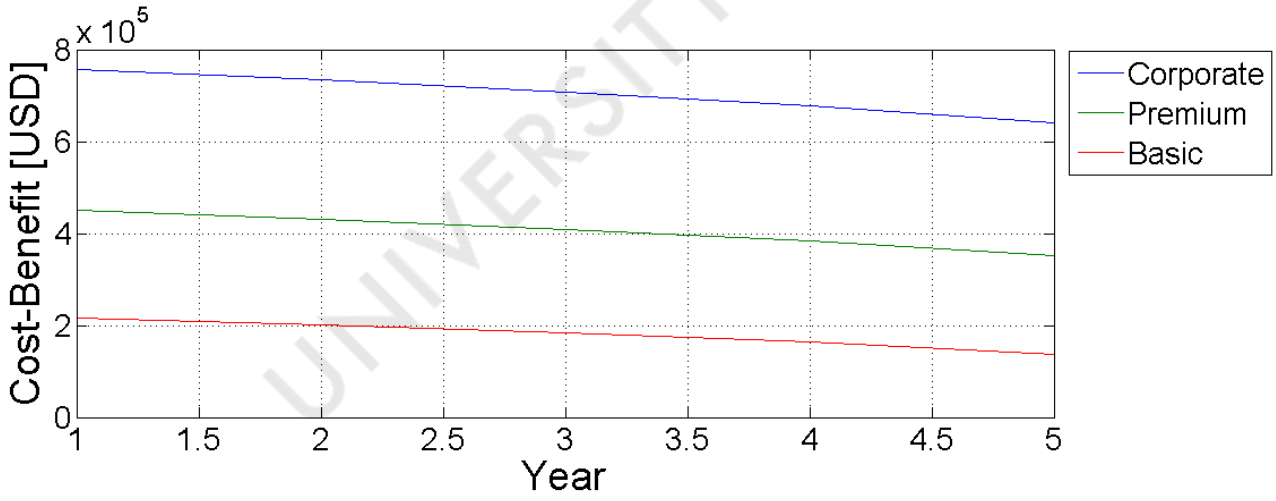


Figure 4.8. Case Scenario 1 – 20% annual increase in the demand: Cost-benefit flow

UNIVERSITY of NICOSIA

Table 4.18. Case Scenario 1: Cost-benefit numerical results

	Year 1	Year 2	Year 3	Year 4	Year 5
Corporate	756,000	734,400	708,451.66	677,294.34	639,896.15
Premium	450,000	431,654.4	409,710.53	383,464.38	352,073.38
Basic	216,000	201,398.4	183,928.5	163,031.84	138,040.92

On the contrary, the second case scenario indicates that the Corporate and Premium services are always underutilized over the 5-year modeling period. An increase in the return on investment is also observed due to the continuous decrease in the number of monetary units. For the Basic service, underutilization is also noticed for the first four years. During the fifth year, the optimal condition is achieved as the calculation result becomes zero at some point; however, the cost-benefit numerical results become negative until the end of the 5-year period due to the continuous increase in the number of active users, revealing that the service is overutilized. The need for abandoning/terminating the existing service and switching to a more flexible capacity one will be faced; these options would create new and accumulated costs difficult to be managed. As a result, the Premium service is the most cost-effective lease option for that case scenario, as the cost-benefit results are closer to the minimum positive values and the problem of the service utility overutilization does not lurk.

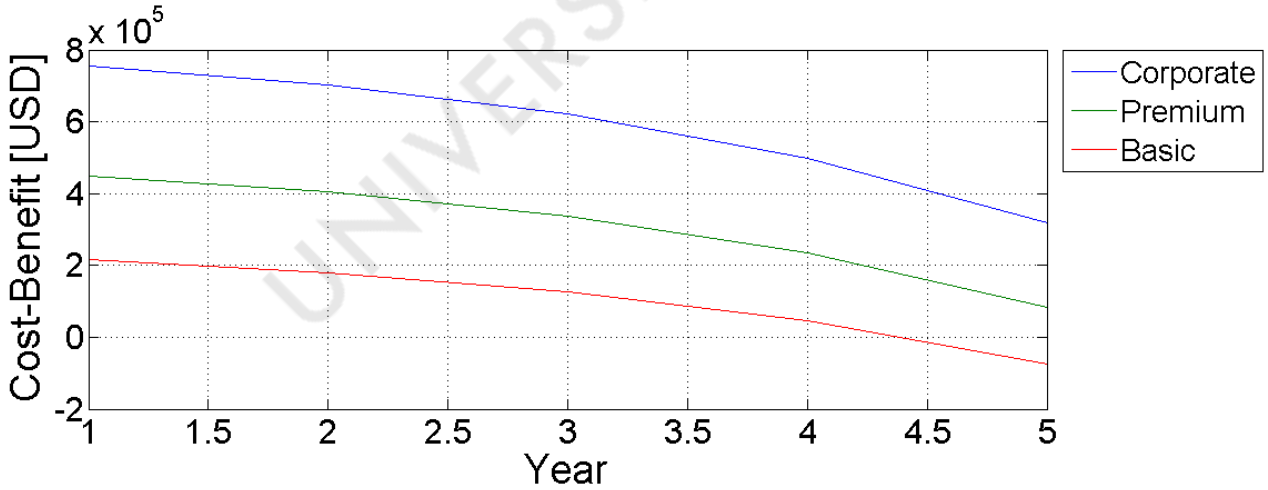


Figure 4.9. Case Scenario 2 – 50% annual increase in the demand: Cost-benefit flow

UNIVERSITY of NICOSIA

Table 4.19. Case Scenario 2: Cost-benefit numerical results

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
Corporate	756,000	702,000	620,975.16	499,439.78	317,173.15
Premium	450,000	404,676	336,933	235,625.93	84,063.98
Basic	216,000	179,820	125,732.38	44,848.11	-76,136.04

The third case scenario reveals that the Corporate service is always underutilized throughout the 5-year modeling period. An increase in the return on investment is also witnessed due to the positive numerical results. On the other hand, the Premium service is underutilized the first four years. However, overutilization of the service is witnessed until the end of the 5-year modeling period, although the optimal condition (i.e., zero monetary units, indicating the perfect match between resource demand and supply) is achieved during the fifth year at some point. Likewise, the Basic service becomes overutilized just before the end of the fourth year and remains in this mode until the end of the modeling period due to the assumption that a linear growth in the number of end-users occurs. The lease of the Corporate service therefore constitutes the most cost-effective selection decision for that case scenario, as the numerical results are closer to the minimum positive values and the problem of service utility overutilization does not lurk over the 5-year modeling period.

UNIVERSITY of NICOSIA

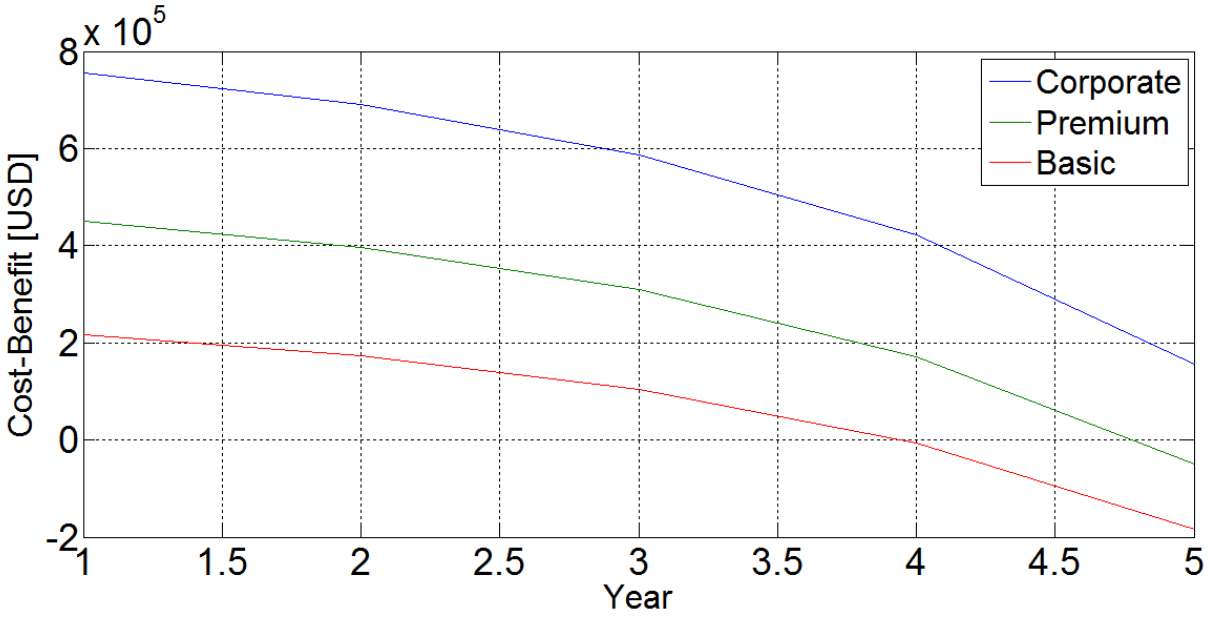


Figure 4.10. Case Scenario 3 – 60% annual increase in the demand: Cost-benefit flow

Table 4.20. Case Scenario 3: Cost-benefit numerical results

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
Corporate	756,000	691,200	587,496.5	421,581.17	156,173.36
Premium	450,000	395,683.2	309,079.87	170,907.34	-49,640.34
Basic	216,000	172,627.2	103,459.78	-6,888.67	-182,984.47

The fourth case scenario shows that the Corporate and Premium services are always underutilized the first four years. During the fifth year, the optimal condition (i.e., zero monetary units' threshold) is achieved for both services at some point; however, the results become negative until the end of the 5-year period, revealing that both services are overutilized. On the contrary, the Basic service is underutilized the first three years due to the positive cost-benefit numerical results. The number of monetary units is constantly decreasing, witnessing a simultaneous increase in the return on investment. However, overutilization of the service

occurs during the last two years, although the optimal condition is achieved during the fourth year. This sheer annual increase in the demand (70%) motivates the need to abandon any of the three candidate services and switch to other more flexible services in terms of capacity from the same or different cloud provider in order to satisfy this linear growth in the number of active users and avoid entering into new and accumulated costs in the long run.

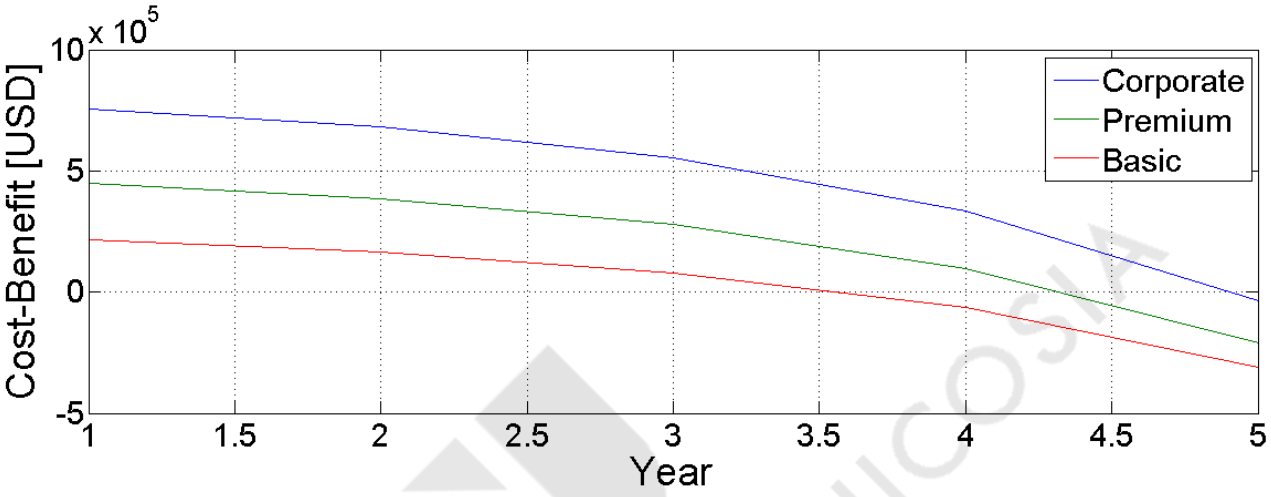


Figure 4.11. Case Scenario 4 – 70% annual increase in the demand: Cost-benefit flow

Table 4.21. Case Scenario 4: Cost-benefit numerical results

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
Corporate	756,000	680,400	551,875.92	333,355.81	-38,017.59
Premium	450,000	386,690.4	279,429.77	97,571.58	-210,908.67
Basic	216,000	165,434.4	79,750.25	-65,514.12	-311,860.4

UNIVERSITY of NICOSIA

4.3.3. Profit Optimization in Cloud Storage: Non-Linear Mathematical Modeling

A 5-year modeling period ($l = 5$) is examined prior to adoption of either a data warehouse or a big data as a service (BDaaS) model, performing a what-if analysis on two different case scenarios under the assumption that fluctuations in the demand for storage occur. The variations in the demand for storage capacity concerning the two case scenarios are presented in Table 4.22.

Table 4.22. Case Scenario 1 & 2: Variations in demand for storage

Term	Case Scenario 1	Case Scenario 2
Year 1 to 2	$\beta_1\% = 5\%$	$\beta_1\% = 10\%$
Year 2 to 3	$\beta_2\% = 15\%$	$\beta_2\% = 22\%$
Year 3 to 4	$\beta_3\% = 20\%$	$\beta_3\% = 35\%$
Year 4 to 5	$\beta_4\% = 23\%$	$\beta_4\% = 40\%$

Towards a better understanding of the cost and benefits analysis quantification, the calculations for the first two years are given as

$$CA_1 = 12 * \left(Cs_{/m} * S_{curr}\right)$$

$$C_{D_1} = 12 * [Cs_{/m} * (S_{max} - S_{curr})]$$

$$CA_2 = 12 * [(1 + \delta_1\%) * Cs_{/m} * (1 + \beta_1\%) * S_{curr}]$$

$$C_{D_2} = 12 * \{(1 + \delta_1\%) * Cs_{/m} * [S_{max} - (1 + \beta_1\%) * S_{curr}]\}$$

Having explained the quantification rules, the data input in Tables 4.23 and 4.24 are applied to the formulas presented in section 4.2.4. Table 4.24 includes the cost variations for leasing additional cloud storage for the two case scenarios, which are dependent on the variations in the demand for storage. The evaluation results are shown thoroughly in Tables 4.25 and 4.26, while the cost analysis flows and the cost difference/benefits comparisons over the 5-year modeling period are witnessed in Figures 4.12, 4.13, 4.14 and 4.15, respectively.

Table 4.23. Data Input for Formulas in Section 4.2.4

Variable Definition	Data Input
Maximum storage capacity (terabytes)	$S_{max} = 4$
Storage currently used (terabytes)	$S_{curr} = 2$
Initial monthly cost for leasing cloud storage (USD)	$Cs_{/m} = 390$

Table 4.24. Case Scenario 1 & 2: Cost Variations for Leasing Cloud Storage

Variable Definition	Case Scenario 1	Case Scenario 2
Cost variation for leasing additional cloud storage	$\delta_1\% = 2\%$ $\delta_2\% = 5\%$ $\delta_3\% = 18\%$ $\delta_4\% = 20\%$	$\delta_1\% = 5\%$ $\delta_2\% = 10\%$ $\delta_3\% = 25\%$ $\delta_4\% = 18\%$

In this direction, the first case scenario points out that adopting a big data as a service model is more cost-effective than a traditional data warehouse appliance, as the cost analysis results reveal always the least positive values over the 5-year modeling period, despite the increase in the demand for storage. The benefits calculations are always positive in big data as a service (the decline in the results is due to the increase in the resource demand for storage capacity), while the cost difference results are always zero in traditional data warehouse approaches. It is worthy to mention that the cost analysis and cost difference/benefits results regarding the traditional data warehouse appliances remain the same throughout the period, because there are

charges for the full storage capacity and not the actual one used, as it is provided by cloud computing.

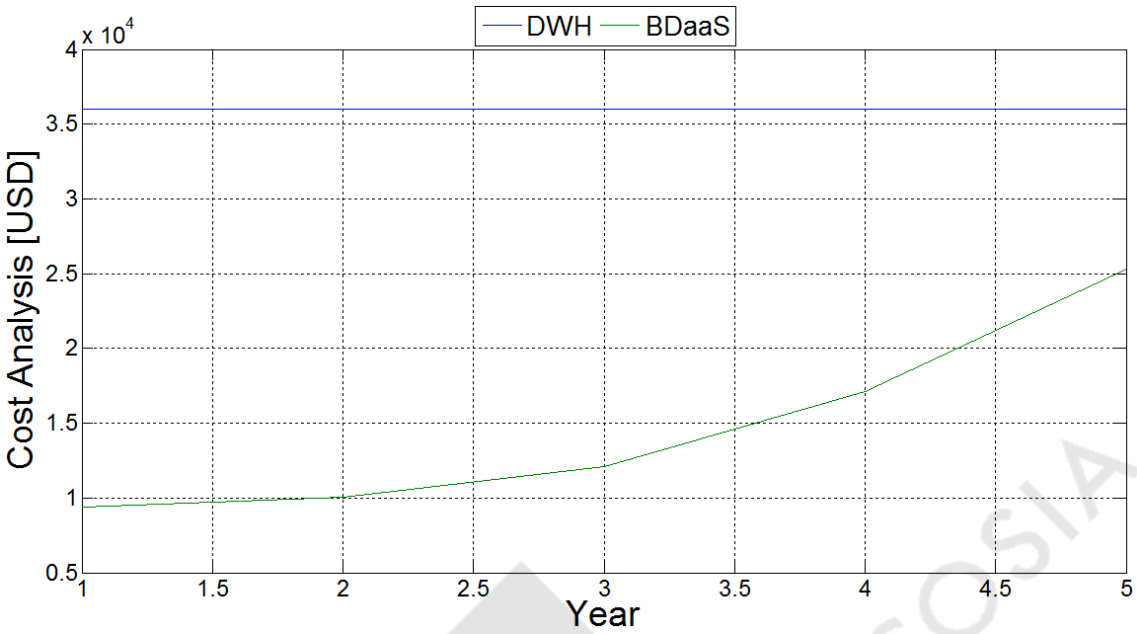


Figure 4.12. Case Scenario 1: Cost analysis flow

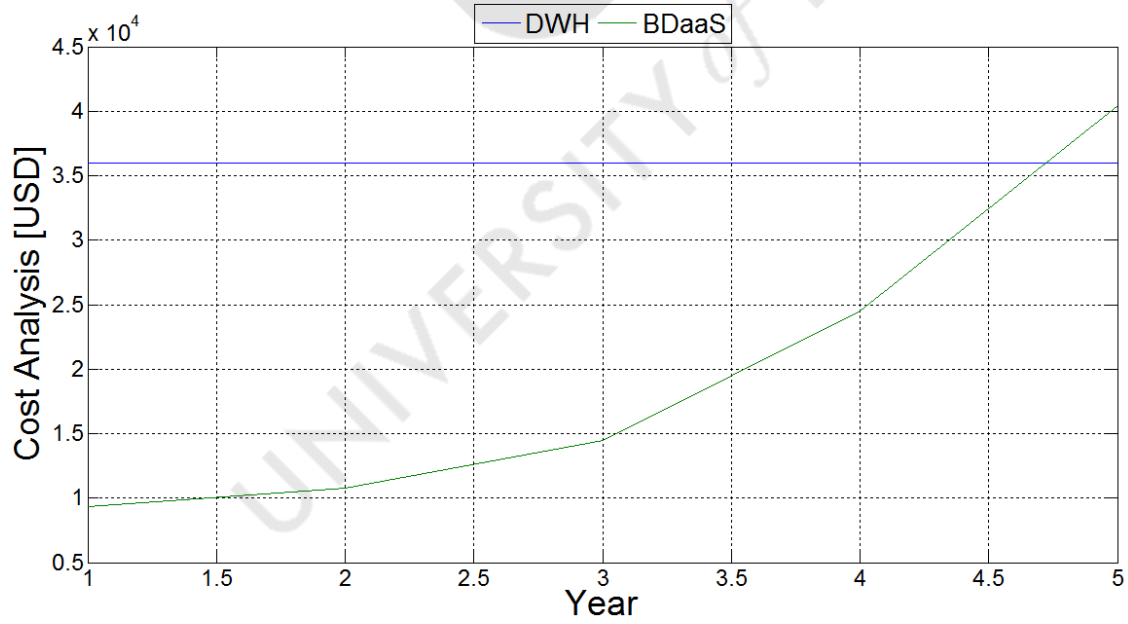


Figure 4.13. Case Scenario 2: Cost analysis flow

On the contrary, the second case scenario demonstrates the cost-effectiveness and the benefits gained by adopting the big data as a service model during the first four years. However, the cost difference/benefits results become negative during the fifth year, which reveal the need

for immediate upgradation that will be motivated in order to meet the demand requirements. The necessity for upgradation can be also observed at the increased costs compared to those in the conventional data warehouse approach. In this case, the earnings gained throughout the period, due to the selection of the dig data as a service model, will be reinvested on the additional storage required, maximizing the return on investment.

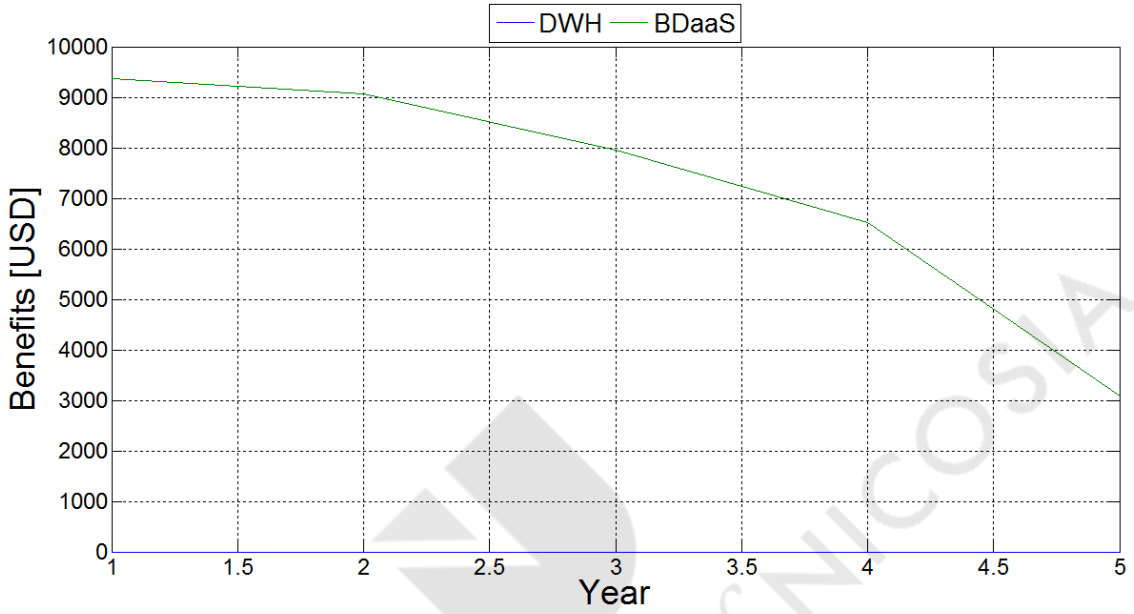


Figure 4.14. Case Scenario 1: Cost difference/Benefits comparison

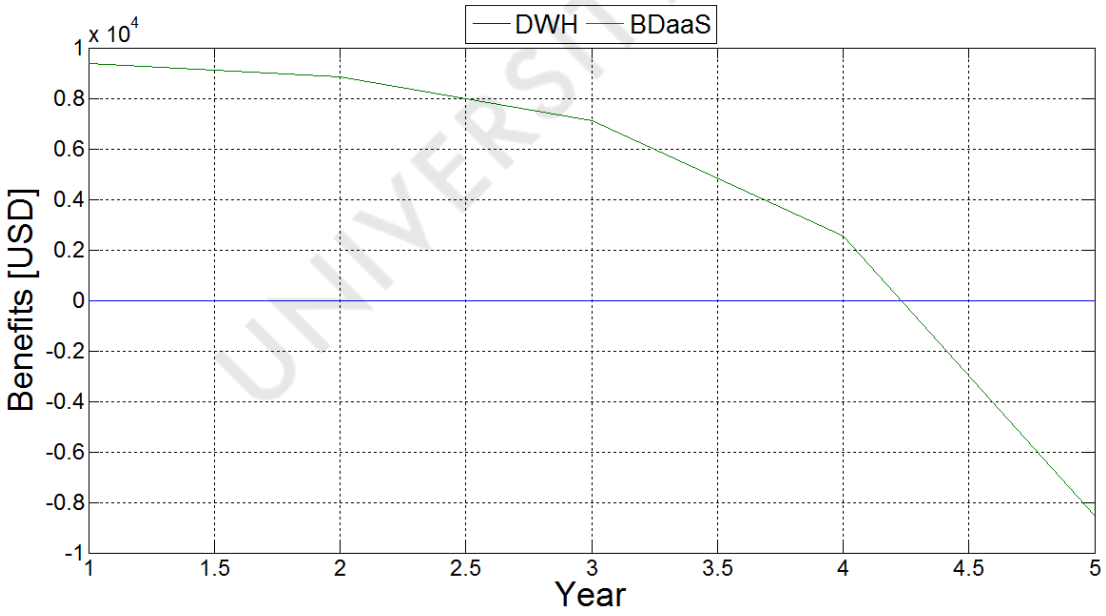


Figure 4.15. Case Scenario 2: Cost difference/Benefits comparison

UNIVERSITY of NICOSIA

Table 4.25. Case Scenario 1: Cost & Benefits Analysis Results (Big Data as a Service)

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
CA	9,360	10,024.56	12,104.66	17,140.19	25,298.93
C_D	9,360	9,069.84	7,944.46	6,517.77	3,090.63

Table 4.26. Case Scenario 2: Cost & Benefits Analysis Results (Big Data as a Service)

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
CA	9,360	10,810.8	14,508.09	24,482.41	40,444.94
C_D	9,360	8,845.2	7,113.51	2,544.59	-8,553.08

4.3.4. Profit Optimization in Cloud Storage: Linear Mathematical Modeling

The cost analysis and benefits comparisons are performed in a 5-year modeling period (l = 5), analyzing three different case scenarios under the prediction that the demand curves for cloud storage increase linearly. The variations in the demand for storage capacity and the total cost variations for leasing additional cloud storage for the three case scenarios are given in Table 4.27. The data input in Table 4.28 are applied to the formulas presented in section 4.2.5. The evaluation results are shown in Tables 4.29 to 4.31, while the cost analysis and benefits flow over the 5-year modeling period are witnessed in Figures 4.16 to 4.21, respectively.

Table 4.27. Variations in Demand for Storage & Cost for Leasing Cloud Storage (3 Scenarios)

	Linear Increase in the Demand for Storage Capacity (yearly)	**Cost Variation for Leasing Additional Cloud Storage (l-year modeling period)**
Case Scenario 1	$\beta_1\% = 18\%$	$\Delta_1\% = 10\%$
Case Scenario 2	$\beta_2\% = 36\%$	$\Delta_2\% = 18\%$
Case Scenario 3	$\beta_3\% = 48\%$	$\Delta_3\% = 24\%$

UNIVERSITY of NICOSIA

Table 4.28. Data Input for Formulas in Section 4.2.5

Variable Definition	Data Input
Maximum storage capacity (terabytes)	$S_{max} = 7$
Storage currently used (terabytes)	$S_{curr} = 3$
Initial monthly cost for leasing cloud storage (USD)	$Cs_{/m} = 400$

Towards the analysis of the evaluation results, the first case scenario points out the cost-effectiveness of the big data as a service model as the cost analysis calculations hold the least positive values over the 5-year modeling period. The benefits numerical results are also positive strengthening the aforementioned argument; the decline in the numbers is subject to the predicted linear demand curves for storage capacity. It is worthy of mention that the cost analysis and benefits numerical results in data warehouse appliances remain the same throughout the period, as there are charges for the full storage capacity not based on the actual use.

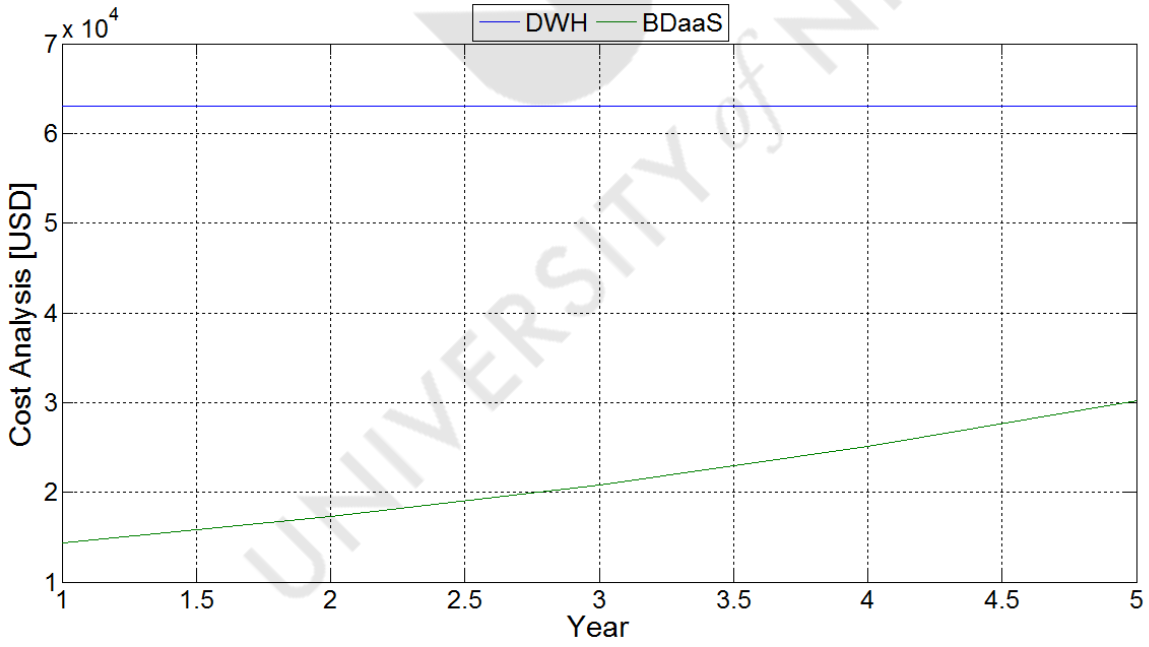


Figure 4.16. Case scenario 1: Cost analysis flow

UNIVERSITY of NICOSIA

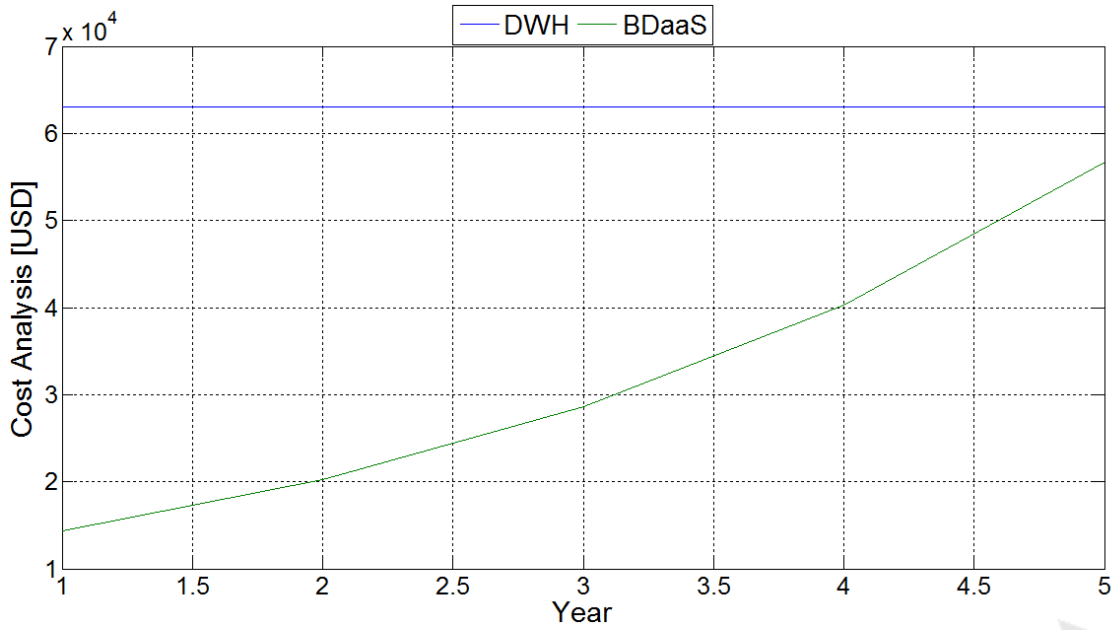


Figure 4.17. Case scenario 2: Cost analysis flow

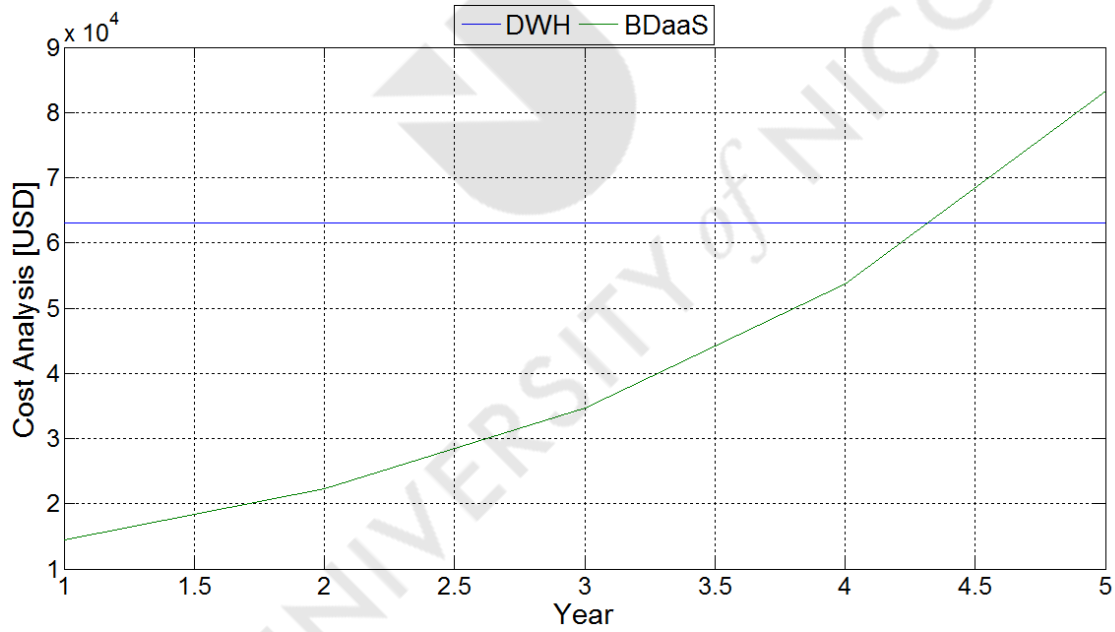


Figure 4.18. Case scenario 3: Cost analysis flow

On the contrary, the second and third case scenarios point out the cost-effectiveness and the benefits gained by adopting big data as a service models during the first two years. However, the benefits calculations become negative during the third year as far as it concerns the second case scenario, motivating the need for upgradation to meet the demand needs and requirements. Storage upgradation is also observed to be necessary when examining the third case scenario (i.e., negative benefits numerical results are witnessed during the third year) due to the

UNIVERSITY of NICOSIA

increased costs that the big data as a service model selection decision brings. In this context, the proposed methodology proves that the earnings gained due to the selection of the big data as a service model should be capitalized on the reinvestment of the additional storage needs in the long run, achieving to maximize the return on investment.

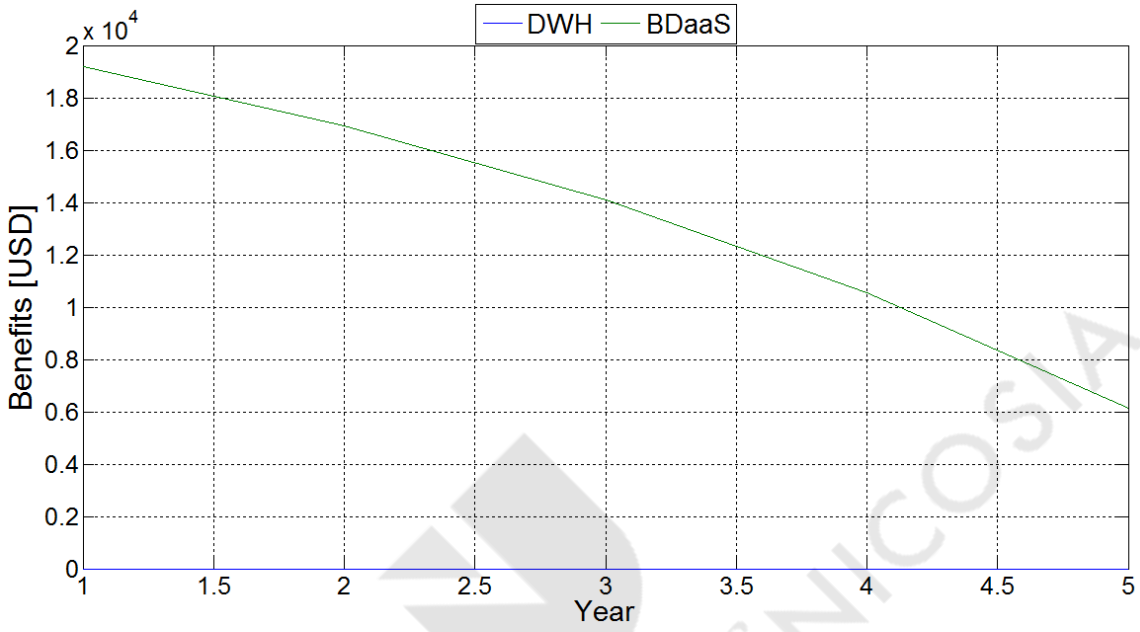


Figure 4.19. Case scenario 1: Benefits analysis flow

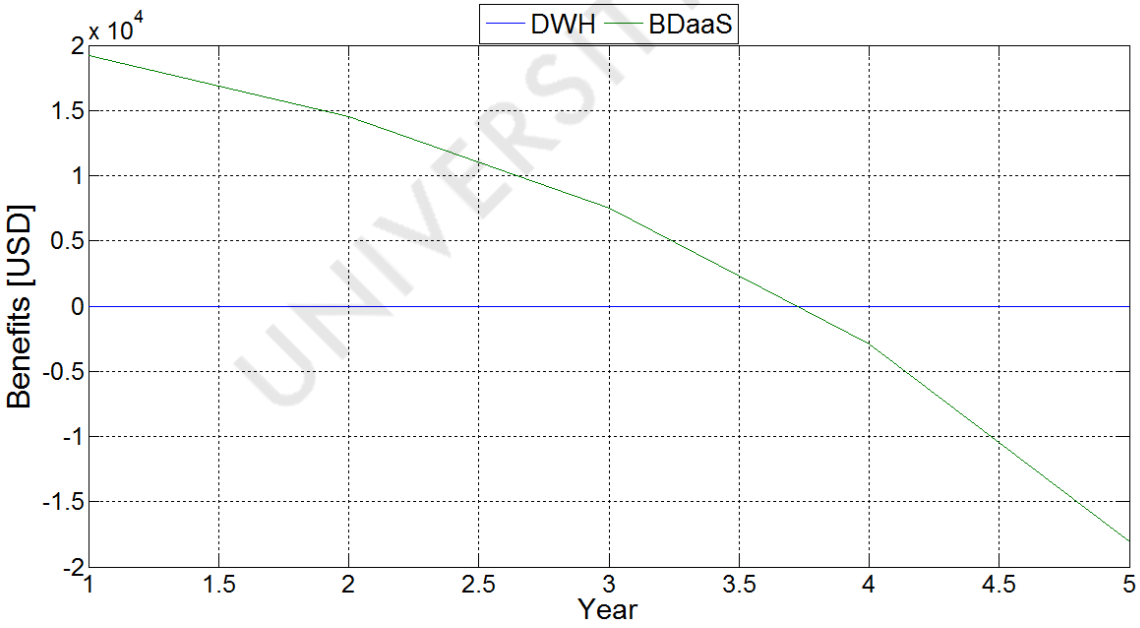


Figure 4.20. Case scenario 2: Benefits analysis flow

UNIVERSITY of NICOSIA

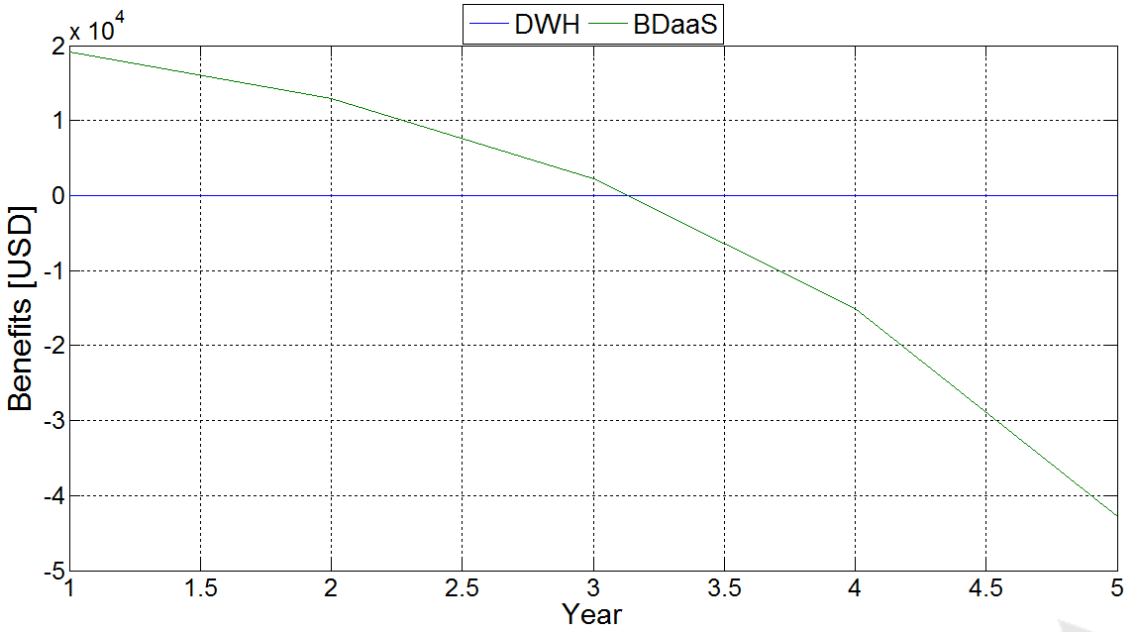


Figure 4.21. Case scenario 3: Benefits analysis flow

Table 4.29. Case Scenario 1: Cost & Benefit Analysis Results (Big Data as a Service)

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
CA	14,400	17,331.84	20,860.6	25,107.82	30,219.77
B	19,200	16,940.16	14,096.84	10,548.77	6,149.95

Table 4.30. Case Scenario 2: Cost & Benefit Analysis Results (Big Data as a Service)

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
CA	14,400	20,289.02	28,586.42	40,277.13	56,748.86
B	19,200	14,520.58	7,476.32	-2,916.12	-18,042.86

Table 4.31. Case Scenario 3: Cost & Benefit Analysis Results (Big Data as a Service)

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
CA	14,400	22,334.98	34,642.44	53,731.81	83,340.19
B	19,200	12,877.82	2,260.57	-15,057.45	-42,809.46

UNIVERSITY OF NICOSIA

4.4.Conclusions

To maximize their return on investment, cloud providers should apply value creation-driven elasticity management strategies, such as dynamic debt- and profit-aware resource adaptation while optimizing the trade-off decisions. However, such cloud resource adaptations are not trivial, as they might result in SLO violations negotiated with service providers. In this context, this chapter proposed novel quantification models and algorithms that follow an elasticity/technical debt and profit optimization approach to leverage the predictability of the dynamic variations in resource capacity/capabilities requests workload (either non-linear or linear) and solve the resource provisioning problem in cloud and mobile cloud systems. The experimental results have shown that the designed models and algorithms quantify effectively the size of the elasticity debt when dynamic variations in the demand requests for cloud resources are applied, providing also insights about the selection decision and the time at which overutilization states for cloud resources and services are encountered throughout the modeling period in order to avoid options like abandoning/termination, service switching and possible SLO violations.

The designed models and algorithms quantify the size of the elasticity debt and measure the profits not earned due to the service utility underutilization of cloud-supported resources (i.e., interest due to cost of unused service capacity). Towards a value- and strategy-oriented resource utilization framework, the next chapter investigates a game theoretic incentive mechanism based on an equilibrium model, which allows to adapt and allocate resources in cloud and mobile cloud computing environments for optimal auto-scaling.

UNIVERSITY of NICOSIA

Chapter 5

Game Theoretic Control Mechanisms for Optimal Resource Utilization in Multi-Tenant Software as a Service

This chapter investigates the problem of composite service utility overutilization detection as a part of dynamic resource adaptation in distributed cloud computing environments. Determining those control mechanisms to re-allocate resources from an overutilized service component utility is an aspect of dynamic resource adaptation that directly influences the resource utilization and Quality of Service (QoS) delivered by the systems. This chapter proposes a novel game theoretic incentive scheme for efficient resource provisioning with debt-aware considerations based on an equilibrium model, which motivates the optimal auto-scaling and exchange of resources in cloud and mobile cloud computing systems. The control mechanisms optimally solve the composite service component utility overutilization problem by forming dynamic coalitions, optimizing trade-off decisions and scheduling resources while analysing the given state of debt minimization and profit maximization based on a Hidden Markov Model. The high efficiency of the proposed theorem and mechanisms is shown by specialized experiments using different cloud service capabilities and storage capacities in the context of multi-tenant software as a service.

5.1. Introduction

This chapter focuses on the problem of composite service utility overutilization detection as a part of dynamic resource adaptation in cloud. Detecting when a service utility becomes overloaded directly influences the QoS, since if the resource capacity is completely utilized, it is likely that the cloud-supported services and applications will experience resource shortage and performance degradation. Different from previous research works [27], [124], this chapter introduces the novel elasticity debt analytics paradigm to support the adaptation decisions when tasks are offloaded, scheduled and executed on highly dynamic cloud and mobile cloud computing environments, including effective mechanisms for balanced improvements in the trade-offs. A definition for *elasticity debt analytics* (EDA) is given as 'the concept that reflects the debt-aware measurement and interpretation of meaningful patterns in cloud and service

UNIVERSITY of NICOSIA

resource data to gain insights for efficient resource provisioning, scheduling and guarantee QoS requirements'. To date, most existing state of the art techniques in the scientific literature, which are related to dynamic trade-offs for resource provisioning, rarely take into account this scientific problem, because they tend to examine the applications' quality of service without addressing adequately issues related to cost minimization in multi-tenant cloud-supported mobile services. To address this scientific problem, determining those control mechanisms to re-allocate resources from an overutilized composite service utility is an aspect of dynamic resource adaptation that directly influences the system's resource utilization. The proposed theorem uses the necessary metrics to overcome these challenges and automate the resource allocation process based on the elasticity debt quantitative results. The equilibrium model ensures also the consideration of the non-cooperative behavior of end-users, service providers and infrastructure suppliers towards the minimization of the elasticity debt and maximization of profits of both the resource supply and the resource on demand. The proposed game theoretic incentive scheme guarantees the following:

1. Adaptation decisions for auto-allocation of incoming resource requests workload to high elasticity debt-level services.
2. Increased elasticity wealth for the service providers or infrastructure suppliers.
3. Accurate predictions on the elasticity debt and mitigation of the risk of service utility overutilization.
4. Efficient scalability as the number of mobile device user requests for cloud resources increases or decreases accordingly.
5. Optimization of cloud resources and consumption rates to prevent unsustainable consumption, considering the current state of elasticity debt or profits.
6. Effective elasticity debt and profit monitoring when parallelization of requests workloads for cloud resources occurs.

The experimental results prove the effectiveness of this theorem in terms of:

1. Allocation of multiple resources simultaneously by applying resource allocation metrics based on elasticity debt.
2. High efficiency when the number of active end-users is large in the context of multi-tenant software as a service and the number of service providers in cloud environments varies.

The scientific contributions of this chapter are the following:

1. Modeling for the computation of the processing time, energy and overhead in mobile opportunistic offloading.

2. A novel utility-driven elasticity debt and profit optimization approach for efficient cloud resource allocation based on the Hidden Markov Model.
3. Applicability of novel debt minimization- and profit maximization-aware game theoretic control mechanisms for optimal resource utilization with well-compromised trade-offs in a multi-tenant software as a service architecture.

The remainder of the chapter is organized as follows: the next section discusses the related work, followed by the applicability of debt- and profit-inspired game theoretic control mechanisms as a solution concept for optimal resource utilization with well-compromised trade-offs in cloud and mobile cloud computing (MCC) environments based on a Hidden Markov Model as presented in Section 5.3. In section 5.4, the experimental evaluation of the theorem is provided. The chapter is concluded with section 5.5 discussing the results.

5.2. Related Work

Cloud elasticity enables the dynamic acquisition and release of shared computational resources on demand and one of the major advantages is the adaptation decisions taken to adjust the resource provisioning constrained by QoS and operating costs. Efficient allocation of resources is critical, because the end-user demands in the cloud are diverse and fluctuate dynamically [24], [272]. The resource allocation problem has been examined using game theory [273], investigating the usage of resources across a cloud-based network and indicating that the cost depends on the amount of computation. In this context, the Nash equilibrium has been exploited to analyse resource allocation-related problems in cloud and mobile cloud computing [274]. The Nash equilibrium is a game involving a group of rational decision makers in which each decision maker knows the equilibrium strategies of the other decision makers and the outcome from this interaction depends on the decisions of the others [275].

Cloud resource provisioning problems are classified into optimization of resource utilization – especially in the data centres of cloud infrastructure [276] – and maximization of profits [17], [149], which can be either the overall profit or the profits of the infrastructure suppliers. The service provisioning problem has been attempted to be solved as a Nash game in [277], improving the overall efficiency of the evaluated cloud systems in terms of Price of Anarchy. On the contrary, the cloud storage service selection strategy has been investigated in [278] from a game theoretic point of view, towards the promotion of truth-telling among different service

providers. Other research works dealt with the application of Nash equilibrium in mobile cloud computing systems and architectures [279], examining the creation of a resource pool to support mobile applications coupled with a game model for the adoption of optimal strategies on capacity expansion of the pool. Another study [280] deals with the cooperative behavior of multiple cloud providers divided into groups from a resource and revenue sharing perspective in a resource pool. The authors attempt to develop a stochastic linear programming game that considers the uncertainty of internal users.

From an energy minimization perspective, the problem of reduction of the energy consumption has been examined as a congestion game in [281] where each player selects one of the servers to offload the computation tasks, while in [282] the computation offloading decision problem in mobile cloud computing environments has been formulated as a decentralized computation offloading game. Game theoretic channel access strategies are adopted in [283] by estimating equilibrium points to achieve balance between conserving energy and completing the data dissemination.

The work in [284] examines a QoS constrained resource allocation problem in which service demanders intend to solve the sophisticated parallel computing problem by requesting the usage of resources across a cloud-based network. A two-step solution is presented: initially, each participant solves its optimal problem independently through a Binary Integer Programming method, without considering the multiplexing of resource assignments. Secondly, an evolutionary mechanism is designed that changes multiplexed strategies of the initial optimal solutions of different participants with minimizing their efficiency losses. Authors in [285] discuss the issue of QoS-guaranteed bandwidth shifting and re-distribution as a utility maximization problem. A descending bid auction mechanism is proposed where each gateway aggregates the demands of all connecting mobile nodes and makes a bid for the required amount of bandwidth.

A green cloudlet network architecture has been introduced in [286] to provide low end-to-end delay between a user equipment and its avatar in the cloudlets and facilitate the workload offloading process. The resource allocation problem in cloud has been also researched in [287], focusing on the usage of resources across a cloud-based environment. This work examines a model based on a Stackelberg game to increase the profits of both cloud resource suppliers and applicants. Decisions for economically optimal scaling at cloud infrastructure level has been explored in [27]. The scaling decisions consider the cost of infrastructure, the revenue from service delivery and the profit of the service provider. An adaptive elastic scaling perspective has been discussed in [92] using cost-aware criteria to detect and analyse the bottlenecks within

multi-tier cloud-based applications. The algorithm achieves to reduce the costs incurred by the users of cloud infrastructure services, allowing them to scale the applications at the bottleneck tiers.

Game theoretic mechanisms have been also used in cloud-centric internet of things (IoT) systems [288]. Performance evaluations of Bayesian coalition games among objects and devices in the IoT landscape have been performed in [289]; the players have variable learning rates in the coalition game, opposed to the constant learning rates provided in the majority of other solutions, based on a utility function.

5.3. Game Theoretic Incentive Scheme for Cloud Resource Provisioning

5.3.1. Mobile Opportunistic Offloading Model

A mobile opportunistic offloading perspective for virtualized and on-demand service provisioning is proposed where a task K_n is finally offloaded to the cloud after denial by two mobile device nodes in close proximity due to resource restrictions. The elasticity debt optimization model can be used to dynamically adapt the loads on different subsystems of the mobile opportunistic system. The computation processing time ($T_{n,proc}^{mc}$) and energy ($E_{n,proc}^{mc}$) of mobile device user n for offloading the input data of size B_n are computed as,

$$T_{n,proc}^{mc} = \frac{B_n}{R_n^{mc}(a)} + \frac{D_n}{F_n^{mc}} + \frac{L_n}{R_n^{mc}(a)} + T_{n,wait\ A\to B}^{l} + T_{n,wait\ B\to C}^{l} + T_{n,wait}^{mc} + \left(P_n^c + T_{n,switch}^{c}\right) \quad (5.1)$$

$$E_{n,proc}^{mc} = \frac{P_n * B_n}{R_n^{mc}(a)} + v_n * D_n + \frac{A_n * L_n}{R_n^{mc}(a)} + E_{n,wait\ A\to B}^{l} + E_{n,wait\ B\to C}^{l} + E_{n,wait}^{mc} + E_{n,switch}^{mc} \quad (5.2)$$

where R_n^{mc} is the uplink data transfer for transmission, D_n the total number of CPU cycles required (i.e., CPU usage) to accomplish the computation task K_n, F_n^{mc} the computation capability (i.e., CPU cores per second), L_n the data output, $T_{n,wait}^{l}$ the tolerable waiting time for queue in order to receive response, P_n^c the sudden shutdown, $T_{n,switch}^{c}$ the intermediate

switches and routers in the cloud that fluctuate the network delay, A_n the transmission power from the cloud, v_n the cloud coefficient denoting the consumed energy per CPU cycle, and $E_{n,wait}^{mc}$ the mobile device user requests based on localization.

According to equations (5.1) and (5.2), the overhead of cloud computing approach (Z_n^{mc}) in terms of processing time and energy is computed as follows, i.e.,

$$Z_n^{mc}(a) = \gamma_n^T * T_{n,proc}^{mc} + \gamma_n^E * E_{n,proc}^{mc} + \gamma_n^M * M_{n,proc}^{mc} \quad (5.3)$$

where $0 \leq \gamma_n^T, \gamma_n^E, \gamma_n^M \leq 1$ denote the weighting factors of processing time, energy and memory usage for mobile device user n's decision making, respectively.

5.3.2. Hidden Markov Model Exploitation for Cloud Resource Scheduling

In this section, the resource allocation strategy is examined from the infrastructure supplier and service provider viewpoint. A utility-driven elasticity debt quantification approach is proposed in the context of cloud resource scheduling for optimization of the resource utilization based on the Hidden Markov Model (HMM). In this context, the cloud resource scheduling model is formulated as a four-tuple of $H =< A, B, N, S >$, where A is the set of N service providers. $A =< a1, a2, \dots, at >$ and $B =< b1, b2, \dots, bt >$ is the price set of the service provider. For $\forall b_i, b_i =< \lambda_i^{CPU}, \lambda_i^{MEM}, \lambda_i^{BW}, \lambda_i^{SR}, \lambda_i^{TP}, \lambda_i^{IOP} >$ denotes the service provider i's bid of CPU, memory, bandwidth, storage, throughput and I/O processor, respectively. $N =< n1, n2, \dots, nt >$ indicates the combinatorial demand of all types of resources requested by the service providers. For $\forall n_i, n_i =< k_i^{CPU}, k_i^{MEM}, k_i^{BW}, k_i^{SR}, k_i^{TP}, k_i^{IOP} >$ denotes the service provider i's demand for all types of resources. $S = (s^{CPU}, s^{MEM}, s^{BW}, s^{SR}, s^{TP}, s^{IOP})$ shows the resources of CPU, memory, bandwidth, storage, throughput and I/O processor managed by the infrastructure provider. The modeling of elasticity debt optimization (*EDO*) and profit optimization (*PRO*) in the context of cloud resource scheduling is given as follows in equations (5.4) and (5.5) respectively, i.e.,

UNIVERSITY of NICOSIA

$$EDO = \max \sum_{i=1}^{N} \left[s^{CPU} - (1 + \beta_i\%) * \delta_i * k_i^{CPU} \right] + \left[s^{MEM} - (1 + \beta_i\%) * \delta_i * k_i^{MEM} \right] + \left[s^{BW} - (1 + \beta_i\%) * \delta_i * k_i^{BW} \right] + \left[s^{SR} - (1 + \beta_i\%) * \delta_i * k_i^{SR} \right] + \left[s^{TP} - (1 + \beta_i\%) * \delta_i * k_i^{TP} \right] + \left[s^{IOP} - (1 + \beta_i\%) * \delta_i * k_i^{IOP} \right] \quad (5.4)$$

$$PRO = \max \sum_{i=1}^{N} \left[\delta_i * (\lambda_i^{CPU} + \lambda_i^{MEM} + \lambda_i^{BW} + \lambda_i^{SR} + \lambda_i^{TP} + \lambda_i^{IOP}) \right] \quad (5.5)$$

s.t. $\sum_{i=1}^{N} \delta_i k_i^{CPU} \leq s^{CPU}$, $\sum_{i=1}^{N} \delta_i k_i^{MEM} \leq s^{MEM}$, $\sum_{i=1}^{N} \delta_i k_i^{BW} \leq s^{BW}$, $\sum_{i=1}^{N} \delta_i k_i^{SR} \leq s^{SR}$, $\sum_{i=1}^{N} \delta_i k_i^{TP} \leq s^{TP}$, $\sum_{i=1}^{N} \delta_i k_i^{IOP} \leq s^{IOP}$

5.3.3. Non-Cooperative Elasticity Debt Quantification Game: An Equilibrium Model

The dynamic resource provisioning in the cloud is an economics-driven ecosystem in which cloud customers and providers behave as self-interested and rational agents [147]. The concept of elasticity debt is examined as a methodological model to reason about elasticity decisions in cloud from a utility-driven viewpoint. The rationale is based on the fact that the mobile devices are owned by different individuals and the decision to offload a user's task to the cloud has a significant impact on the level of the elasticity debt minimization for the provided cloud-based mobile services. Let the number of active users for a service be satisfactory enough such that the elasticity debt is minimized holding the least positive values. In such a case, underutilization of this service is guaranteed, avoiding the incurrence of accumulated elasticity debt. The new mobile device user requests are encountered as adaptation decisions to adjust the resource provisioning for the remaining services. The level of adaptation of the new user requests, which imply additional cloud resources, is based on the elasticity debt results obtained for each service. The elasticity debt minimization mechanism is triggered for the remaining services in accordance to the different interests of each individual-user. The subscription and charging policies are also subject to the lease decision. The adaptation decision problem is

examined based on the number of active mobile device users for each multi-tenant SaaS-centric mobile service within an elasticity debt prediction period of time.

In this direction, let $a_{-n} = (a_1, \dots, a_{n-1}, a_{n+1}, \dots, a_N)$ denote the adaptation decisions by all other mobile device users except new device user n. Given the other user's decisions a_{-n}, user n selects a proper decision $a_n \in \{0, 1\}$ to minimize the elasticity debt, i.e.,

$$\min_{a_n \in \{0,1\}} ED_n(a_n, a_{-n}), \forall\, n \in N$$

The elasticity debt minimization problem is resolved in a different manner between the SaaS-based mobile services due to the distinction between the number of active end-users each one can host [23], which is witnessed on the different elasticity debt numerical results. The elasticity debt function is formed with respect to the new mobile device user n as follows,

$$ED_n(a_n, a_{-n}) = \begin{cases} ED_1, & if\ a_n = 0 \\ ED_2, & if\ a_n = 1 \end{cases} \qquad (5.6)$$

where ED_1 is the elasticity debt quantification formula to be triggered when the basic service is selected according to the new user n's request for resources, while ED_2 refers to the elasticity debt equation that is triggered when the premium service is selected.

Towards the proof of the theorem, for a game G, a function ED is constructed over the set of strategy profiles such that the optimal behavior of ED yields a Nash equilibrium in strategies of G. The adaptation decision problem is formulated as a game $G = (N, \{A_n\}_{n \in N}, \{ED_n\}_{n \in N})$, where N is the set of mobile device users during a computational offloading period, $A_n \triangleq \{0, 1\}$ is the set of strategies for the new user n's request, and the elasticity debt function $ED_n(a_n, a_{-n})$ of each new user n is the cost-oriented function to be minimized by user n. The game G is the elasticity debt quantification game and the concept of Nash equilibrium [290] is now introduced.

Definition 1. A strategy profile $a^* = (a_1^*, \dots, a_N^*)$ is a Nash equilibrium of the elasticity debt quantification game if at the equilibrium a^*, no new player can be allocated to a cloud-based mobile service to further minimize the elasticity debt by unilaterally changing its strategy, i.e.,

$$ED_n(a_n^*, a_{-n}^*) \leq ED_n(a_n, a_{-n}^*), \forall\, a_n \in A_n, n \in N \qquad (5.7)$$

The Nash equilibrium achieves to minimize the elasticity debt by organizing the mobile device users into a mutually satisfactory condition and no user is motivated to deviate unilaterally. The adaptation decisions associated with each service's elasticity debt quantitative result are leveraged towards the optimization of cloud resource provisioning. The number of active users per service or the option of allocating current device users are considered due to the non-linear demand as a result of the new resource requests workload. In the sequel, the game property investigates the existence of Nash equilibrium in the elasticity debt game, introducing the concept of best response [290].

Definition 2. Given the strategies a_{-n} of the other users, user n's strategy $a_n^* \in A_n$ is a best response if

$$ED_n(a_n^*, a_{-n}) \leq ED_n(a_n, a_{-n}), \forall\, a_n \in A_n \qquad (5.8)$$

According to equations (5.7) and (5.8), we observe that all users play the best response strategies towards each other at the Nash equilibrium, concluding with the following lemma.

Lemma 1. Given the strategies a_{-n} of the other mobile device users in the elasticity debt quantification game, the best response of a new user n is given as the following elasticity debt status strategy, i.e.,

$$a_n^* = \begin{cases} 1, & if\ ED_i > 0 \\ 0, & if\ ED_i \leq 0 \end{cases}$$

The elasticity debt status of a multi-tenant cloud-based mobile service is significant both before and after the adaptation decision of a new user n's request for additional resources. Lemma 1 indicates that in the event of elasticity debt results greater than zero, the service remains under-utilized and the new user n's requests are satisfied. In addition, the elasticity debt is further minimized without risking to enter into an accumulated elasticity debt in the long run. However, in case that the elasticity debt results are less than or equal to zero, the composite service utility is overutilized and no resource capabilities are available to satisfy new requests workload. In the latter case, we achieve to avoid accumulated elasticity debt by adapting the requests workload to those services where the elasticity debt could be further

minimized, achieving also improved trade-offs. This theory enables the controlling, prediction and auto-scaling of the elasticity debt in mobile cloud computing environments when multiple adaptation decisions for resource provisioning are taken and the outcome depends on the current or past adaptation decisions.

5.3.4. Non-Cooperative Cost-Benefit Analysis Game: An Equilibrium Model

The proposed game theoretic approach is introduced towards the maximization of the profits evoked by the storage allocation in cloud systems, taking into account that specific storage capacity is assigned to each tenant-player. For a given game *G*, a potential function *B* is constructed over the set of strategy profiles in such a way that the optimal behavior of *B* yields a Nash equilibrium in pure strategies of *G*. Leveraging the storage allocation options with respect to the benefit numerical results delivered by each storage system, we achieve to organize the players into a mutually satisfactory condition while maximizing the profits. For example, we assume two storage systems, i.e., storage system 1 and storage system 2. The number of tenants currently hosted in storage system 1 is satisfactory enough such that the profits are maximized, holding the least positive values. In this case, the additional storage capacity requests should be allocated in storage system 2. This optimal allocation strategy is based on the benefit results delivered by the system's storage capacity and guarantees that storage upgradation will not occur in the long run, avoiding the minimization of the profits. Hence, the benefits maximization is constantly triggered in system 2 always in compliance with the different interests of each tenant. In this direction, the charging policy is restructured according to the additional storage capacity requests and the overall resources to be leased off.

As per the game formulation, the storage allocation problem is examined with respect to the tenants-players and data size within a benefit prediction period of time. Let $a_{-n} = (a_1, \dots, a_{n-1}, a_{n+1}, \dots, a_N)$ denote the storage allocation selection decisions by all other tenants except new tenant n. Given the other tenant's decisions a_{-n}, tenant n selects a proper decision $a_n \in \{0, 1\}$ (i.e., storage system 1 or 2) towards the maximization of the profits and overall benefits, i.e.,

$$\min_{a_n \in \{0,1\}} B_n(a_n, a_{-n}), \forall\, n \in N$$

The benefit experimental results are being proved to differ between the storage systems due to the different pool of tenants and data size that each one can host. Therefore, the benefit maximization problem is not resolved in the same manner for all systems. The benefit function is formed (i.e., either non-linear or linear viewpoint) with respect to the new tenant n as

$$B_n(a_n, a_{-n}) = \begin{cases} B_1, & \text{if } a_n = 0 \\ B_2, & \text{if } a_n = 1 \end{cases} \qquad (5.9)$$

where B_1 refers to the benefits equation for system 1 when selected by the new tenant n, and B_2 indicates the benefits mathematical formula for system 2.

The storage allocation selection decision problem is formulated as a game $G = (N, \{A_n\}_{n \in N}, \{B_n\}_{n \in N})$, where N is the set of tenants-players, $A_n \triangleq \{0, 1\}$ denotes the set of strategies for the new tenant n, and the benefit function $B_n(a_n, a_{-n})$ of each new tenant n is the cost-oriented function to be minimized by player n. The game G constitutes the cost-benefit analysis game and the concept of Nash equilibrium [275] is now introduced.

Definition 1. A strategy profile $a^* = (a_1^*, \dots, a_N^*)$ is a Nash equilibrium of the cost-benefit analysis game if at the equilibrium a^*, no new player can be allocated to a storage system to further maximize the benefits by unilaterally changing its strategy, i.e.,

$$B_n(a_n^*, a_{-n}^*) \leq B_n(a_n, a_{-n}^*), \forall\, a_n \in A_n, n \in N \qquad (5.10)$$

The Nash equilibrium achieves to organize the increasing requests for storage capacity into a mutually satisfactory condition while maximizing the benefits and no tenant-player is motivated to deviate unilaterally. Henceforth, the lease of cloud storage and computing capabilities occurs at an optimal level, considering the current pool of tenants per storage system, the option of allocating a current tenant to another system and estimating the non-linear demand by new requests. In the sequel, the game property analysis investigates the existence of Nash equilibrium in the cost-benefit analysis game, introducing the concept of best response [275].

Definition 2. Given the strategies a_{-n} of the other tenants, tenant n's strategy $a_n^* \in A_n$ is a best response if

$$B_n(a_n^*, a_{-n}) \leq B_n(a_n, a_{-n}), \forall\ a_n \in A_n \qquad (5.11)$$

According to (5.10) and (5.11), we observe that all tenants play the best response strategies towards each other at the Nash equilibrium. Given that statement, we may conclude to the following lemma.

Lemma 1. Given the strategies a_{-n} of the other tenants in the cost-benefit analysis game, the best response of a new tenant n is given as the following benefits status strategy, i.e.,

$$a_n^* = \begin{cases} 1, & \text{if } B_i > 0 \\ 0, & \text{if } B_i \leq 0 \end{cases}$$

Since the current benefits status of a storage system is of primary importance as well as the status after the adoption of the request by the new tenant n, Lemma 1 points out that in case of benefits result greater than zero for a specific system, the potential selected system has storage capacity left and the new tenant n's request can be satisfied always in compliance with the current pool of tenants and data size. In this context, the benefits and profits are further maximized without risking to enter in a storage upgradation status in the long run. Otherwise, once the benefits result is less than or equal to zero, there is no storage capacity left in the selected system to accommodate new requests (i.e., the current pool of tenants and data size are satisfactory enough to maximize the return on investment). In this case, we avoid accumulated costs by allocating the tenant's request workload to other available systems in the data centre where the benefits can be further maximized and the trade-off decisions further improved due to the proposed strategy. This technique enables the benefits controlling and prediction from a cloud storage perspective, when multiple tenants are making decisions simultaneously and the outcome depends on the current or past decisions of the other tenants-players.

UNIVERSITY of NICOSIA

5.4.Evaluation of Theorem: Experimental Results, Analysis and Discussion

5.4.1. Non-Cooperative Elasticity Debt Quantification Game

The proposed theorem attempts to address the resource scheduling and optimization problem for each service on mobile cloud-based service level. The formulae and algorithms were implemented in a MATLAB-based testbed to quantify the elasticity debt and take proper adaptation decisions with optimised trade-offs for resource provisioning where necessary. The proper adaptation decisions enable the allocation of new user requests to high elasticity debt-level services to avoid accumulated elasticity debt. The adaptation strategy to adjust resource provisioning triggers a game theoretic, cost-optimal and control mechanism when new user requests occur. The Nash equilibrium game tends towards elasticity debt optimization with threshold for auto-scaling be $ED_i = 0$ (included). Three cloud-supported mobile services are studied in three different use case scenarios, considering non-linear demand curves (see Table 5.1) in a 5-year elasticity debt period ($\lambda = 5$) and the group composition to minimize the elasticity debt. The key attributes of each service are presented in Table 5.2, which constitute the data input for the elasticity debt formulae. The variations in the monthly subscription price and service cost are presented in Table 5.3.

Table 5.1. Use Case Scenarios: Non-Linear Demand Variations

	Variation in Demand		
Term	**Case Scenario A**	**Case Scenario B**	**Case Scenario C**
Year 1 to 2	$\beta_1\% = 5\%$	$\beta_1\% = 18\%$	$\beta_1\% = 28\%$
Year 2 to 3	$\beta_2\% = 5\%$	$\beta_2\% = 22\%$	$\beta_2\% = 25\%$
Year 3 to 4	$\beta_3\% = 32\%$	$\beta_3\% = 35\%$	$\beta_3\% = 30\%$
Year 4 to 5	$\beta_4\% = 40\%$	$\beta_4\% = 25\%$	$\beta_4\% = 10\%$

UNIVERSITY of NICOSIA

Table 5.2. Key Service Attributes

Variable Definition	**Corporate (C)**	**Premium (P)**	**Basic (B)**
Maximum number of active mobile device users	$U_{max} = 14{,}000$	$U_{max} = 9{,}000$	$U_{max} = 6{,}000$
Initial number of active device users	$U_{curr} = 6{,}000$	$U_{curr} = 4{,}000$	$U_{curr} = 3{,}000$
Initial monthly subscription price	$ppm = 8$	$ppm = 5$	$ppm = 2$
Initial monthly cost for servicing a mobile device user in cloud	$Cu_{/m} = 4$	$Cu_{/m} = 3$	$Cu_{/m} = 1$

Table 5.3. Variations in monthly subscription price and service cost in cloud

	Corporate (C)	**Premium (P)**	**Basic (B)**
Variable Definition	**Case Scenario A**		
Variations in subscription price	$\Delta_1\% = 0.3\%$	$\Delta_1\% = 0.4\%$	$\Delta_1\% = 0.5\%$
	$\Delta_2\% = 0.4\%$	$\Delta_2\% = 0.4\%$	$\Delta_2\% = 0.6\%$
	$\Delta_3\% = 1.6\%$	$\Delta_3\% = 1.7\%$	$\Delta_3\% = 2\%$
	$\Delta_4\% = 1.9\%$	$\Delta_4\% = 2.1\%$	$\Delta_4\% = 2.2\%$
Variations in cost	$VoC_1\% = 0.8\%$	$VoC_1\% = 0.9\%$	$VoC_1\% = 1.2\%$
	$VoC_2\% = 0.9\%$	$VoC_2\% = 1\%$	$VoC_2\% = 1.4\%$
	$VoC_3\% = 2\%$	$VoC_3\% = 2.2\%$	$VoC_3\% = 2.8\%$
	$VoC_4\% = 2.5\%$	$VoC_4\% = 2.8\%$	$VoC_4\% = 3.2\%$
	Case Scenario B		
Variations in subscription price	$\Delta_1\% = 0.7\%$	$\Delta_1\% = 0.8\%$	$\Delta_1\% = 1\%$
	$\Delta_2\% = 0.8\%$	$\Delta_2\% = 0.9\%$	$\Delta_2\% = 1.2\%$
	$\Delta_3\% = 0.9\%$	$\Delta_3\% = 1.1\%$	$\Delta_3\% = 1.3\%$
	$\Delta_4\% = 0.7\%$	$\Delta_4\% = 1\%$	$\Delta_4\% = 1.1\%$
Variations in cost	$VoC_1\% = 0.9\%$	$VoC_1\% = 1.1\%$	$VoC_1\% = 1.6\%$
	$VoC_2\% = 1\%$	$VoC_2\% = 1.2\%$	$VoC_2\% = 1.7\%$
	$VoC_3\% = 1\%$	$VoC_3\% = 1.3\%$	$VoC_3\% = 1.9\%$

UNIVERSITY OF NICOSIA

	Corporate (C)	Premium (P)	Basic (B)
	$VoC_4\% = 1.2\%$	$VoC_4\% = 1.5\%$	$VoC_4\% = 1.6\%$
	Case Scenario C		
Variations in subscription price	$\Delta_1\% = 0.5\%$	$\Delta_1\% = 0.8\%$	$\Delta_1\% = 1.2\%$
	$\Delta_2\% = 0.4\%$	$\Delta_2\% = 0.7\%$	$\Delta_2\% = 1.1\%$
	$\Delta_3\% = 0.8\%$	$\Delta_3\% = 1\%$	$\Delta_3\% = 1.5\%$
	$\Delta_4\% = 0.2\%$	$\Delta_4\% = 0.3\%$	$\Delta_4\% = 0.5\%$
Variations in cost	$VoC_1\% = 0.8\%$	$VoC_1\% = 1.1\%$	$VoC_1\% = 1.7\%$
	$VoC_2\% = 0.7\%$	$VoC_2\% = 1\%$	$VoC_2\% = 1.6\%$
	$VoC_3\% = 1\%$	$VoC_3\% = 1.3\%$	$VoC_3\% = 2\%$
	$VoC_4\% = 0.4\%$	$VoC_4\% = 0.5\%$	$VoC_4\% = 0.8\%$

The experiments aim at identifying whether accumulated elasticity debt occurs during the examined period due to fluctuations in the number of mobile device users. The research findings are summarized in the quantification results obtained before the elasticity debt control mechanism is triggered (Table 5.4) and those obtained after the mechanism is triggered (Table 5.5). Likewise, the plots of the elasticity debt flow for the three case scenarios are presented in Figures 5.1 to 5.6 both before and after the mechanism is triggered. In case scenario A, the obtained results for the basic service demonstrate overutilization of the service utility during year 5. Towards the elasticity debt optimization as an effect of the equilibrium model, the estimated increase in the demand during year 5 should have been up to 37.4 per cent, i.e.,

$$6{,}000 - (1 + x) * 4{,}365.9 = 0 \Longrightarrow x = 0.37428 \approx 37.4\%$$
$$\approx 1{,}634.1 \; new \; mobile \; device \; users$$

The actual 40 per cent increase in the demand that was adopted in our initial scenario corresponds to approximately 1,746.36 new users. The elasticity debt control mechanism achieves to automatically allocate the remaining 112.26 new users to the premium service, helping towards the elasticity debt minimization. The pool of users for the premium service during year 5 is approximately $8{,}149.68 + 112.26 = 8{,}261.94$. The increase in the demand for the premium service is now restructured to approximately 41.9 per cent estimated as follows

$$(1 + y) * 5{,}821.2 = 8{,}261.94 \Longrightarrow y = 0.41928 \approx 41.9\%$$

The variations in the subscription price per month and cost for servicing an end-user in the cloud for the premium service are restructured to $\Delta_4\% = 2.2\%$ and $VoC_4\% = 2.9\%$, respectively. The elasticity debt quantification result for the premium service during year 5 is finally formed to approximately $ED_5 \approx 17{,}960.81$ monetary units, contributing to a further minimization of the elasticity debt. Likewise, the variations in the subscription price and cloud service cost for the basic service are now restructured to $\Delta_4\% = 2.1\%$ and $VoC_4\% = 3.1\%$. The elasticity debt calculation during year 5 is now formed to approximately $ED_5 \approx 15.31$ monetary units, tending towards optimality in terms of minimizing the elasticity debt. In case scenario B, the results for all services indicate overutilization of the service utility during year 5. The estimated increase in the resource demand should have been as follows for the three different services in order to achieve elasticity debt optimization, i.e.,

$$\boldsymbol{Corporate}: 14{,}000 - (1 + x) * 11{,}660.76 = 0 \Longrightarrow x = 0.20060 \approx 20\%$$
$$\approx 2{,}339.24\ new\ device\ users$$

$$\boldsymbol{Premium}: 9{,}000 - (1 + x) * 7{,}773.84 = 0 \Longrightarrow x = 0.15772 \approx 15.7\%$$
$$\approx 1{,}226.16\ new\ device\ users$$

$$\boldsymbol{Basic}: 6{,}000 - (1 + x) * 5{,}830.38 = 0 \Longrightarrow x = 0.02909 \approx 2.9\%$$
$$\approx 169.62\ new\ device\ users$$

In case scenario C, the results we obtain for the basic service demonstrate overutilization of the service utility during year 4. To achieve optimization of the elasticity debt, the estimated increase in the demand during year 4 should have been up to 25 per cent, i.e.,

$$6{,}000 - (1 + x) * 4{,}800 = 0 \Longrightarrow x = 0.25 = 25\% \Longrightarrow 1{,}200\ new\ mobile\ device\ users$$

UNIVERSITY of NICOSIA

The actual 30 per cent increase in the demand in our initial scenario corresponds to approximately 1,440 new users. The elasticity debt control mechanism achieves to automatically allocate the remaining 240 new users to the premium service, when triggered. The pool of users for the premium service during year 4 is approximately $8{,}320 + 240 = 8{,}560$. The increase in the demand for the premium service is now restructured to approximately 33.7 per cent estimated as,

$$(1 + y) * 6{,}400 = 8{,}560 \Longrightarrow y = 0.3375 \approx 33.7\%$$

Once the restructured variations in the subscription price and the cost for servicing an end-user in the cloud are applied, the elasticity debt result for the premium service during year 4 is formed to $ED_4 \approx 10{,}769.18$, while complete optimization is achieved for the basic service with $ED_4 = 0$. Likewise, the estimated increase in the demand during year 5 for the premium service should have been up to 5.1 per cent, i.e.,

$$9{,}000 - (1 + x) * 8{,}556.8 = 0 \Longrightarrow x = 0.0517 \approx 5.1\% \Longrightarrow 443\ new\ mobile\ device\ users$$

Furthermore, the estimated increase for the corporate service during year 5 is 12.1 per cent, i.e.,

$$14{,}000 - (1 + x) * 12{,}480 = 0 \Longrightarrow x = 0.1217 \approx 12.1\%$$

Once the re-estimated variations in the subscription price and service cost are applied, the elasticity debt result for the premium service is formed to $ED_5 \approx 165.13$ during year 5, while for the corporate service is $ED_5 = 480.90$. These results hold the least positive values, minimizing the elasticity debt to the lowest level. Based on the elasticity debt incentive mechanism, the corporate service will satisfy 272 new user requests allocated from the premium and basic services, while the remaining 740 will be automatically allocated to other cloud-based mobile services.

UNIVERSITY of NICOSIA

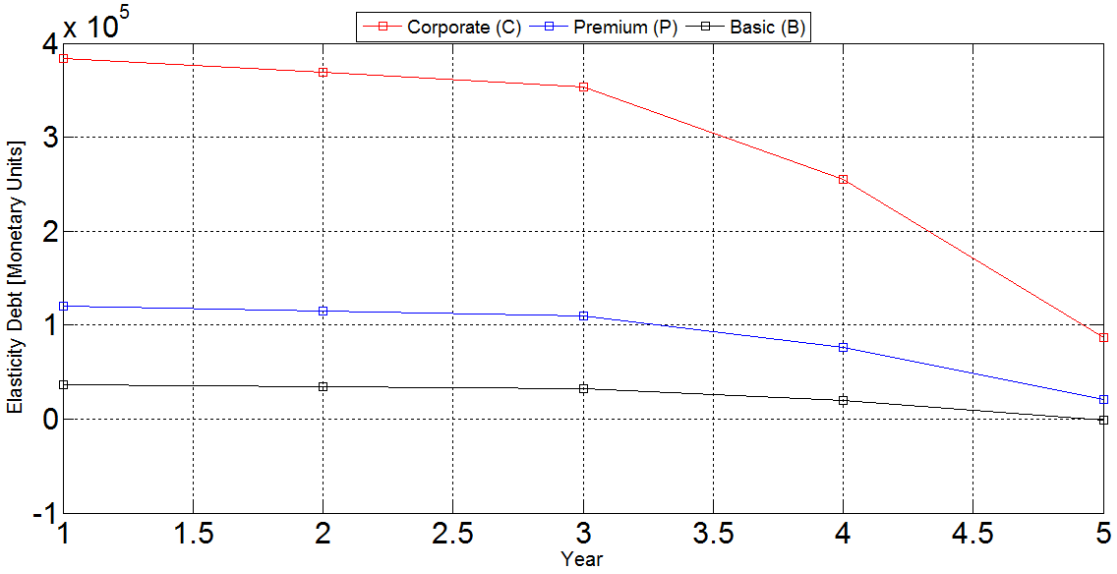


Figure 5.1. Plot for case scenario A: Flow of elasticity debt before the elasticity debt control mechanism is triggered

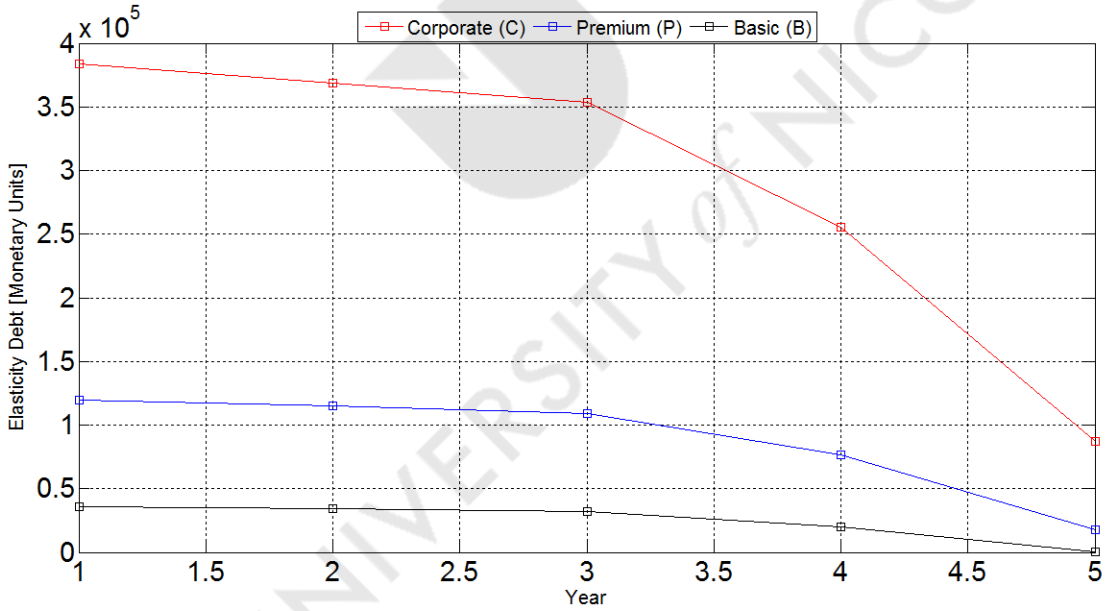


Figure 5.2. Plot for case scenario A: Flow of elasticity debt after the elasticity debt control mechanism is triggered

UNIVERSITY of NICOSIA

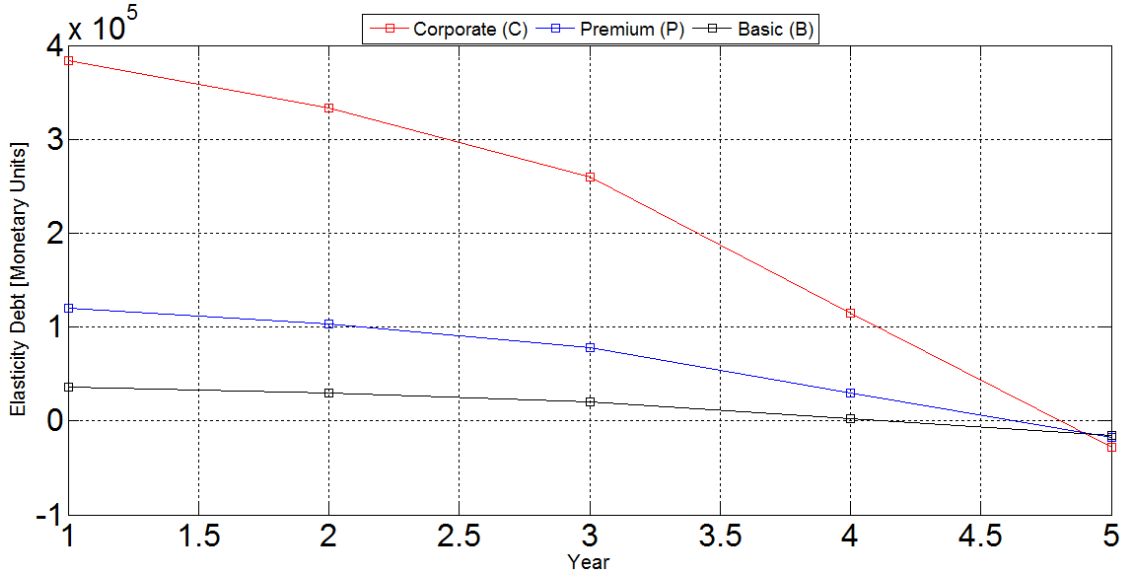


Figure 5.3. Plot for case scenario B: Flow of elasticity debt before the elasticity debt control mechanism is triggered

Table 5.4. Elasticity debt quantification results before the control mechanism is triggered

	Year 1	Year 2	Year 3	Year 4	Year 5
	Case Scenario A				
Corporate	384,000	368,860.80	353,399.55	255,107.72	87,079.06
Premium	120,000	114,796.80	109,213.18	76,338.17	20,625.90
Basic	36,000	34,131.60	32,177.27	19,758.24	-1,373.01
	Case Scenario B				
Corporate	384,000	333,820.80	260,232.21	114,428.37	-28,228.63
Premium	120,000	103,079.52	78,419.39	29,898.70	-17,531.74
Basic	36,000	29,638.08	20,395.67	2,071.90	-15,824.22
	Case Scenario C				
Corporate	384,000	303,966.72	211,830.22	73,615.02	13,172.79
Premium	120,000	93,445.92	62,772.84	16,506.70	-3,689.48
Basic	36,000	26,101.44	14,587.08	-2,946.30	-10,626.93

UNIVERSITY of NICOSIA

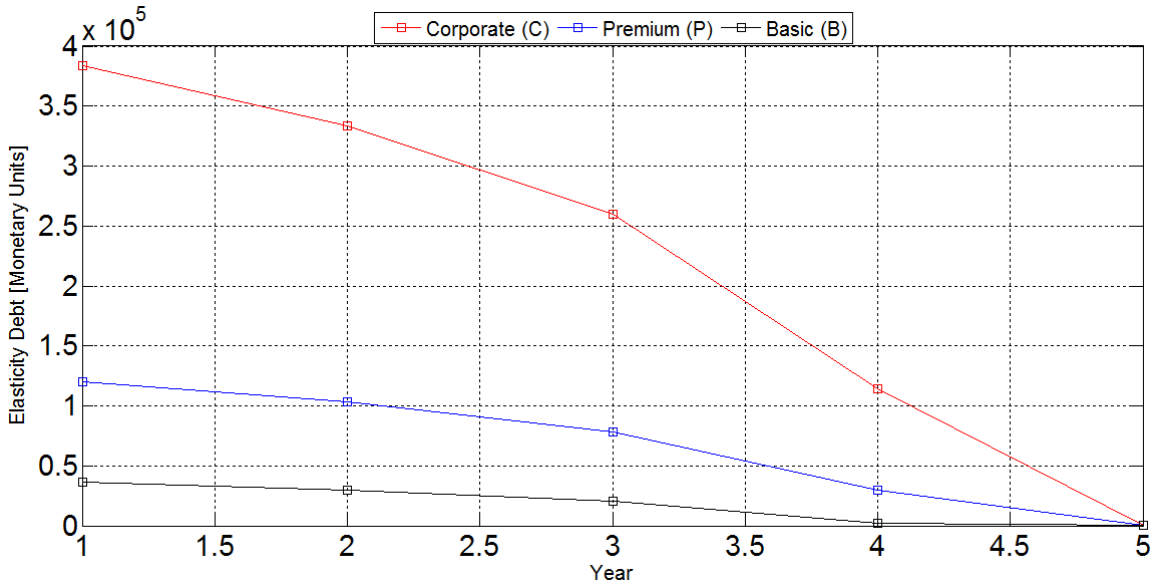


Figure 5.4. Plot for case scenario B: Flow of elasticity debt after the elasticity debt control mechanism is triggered

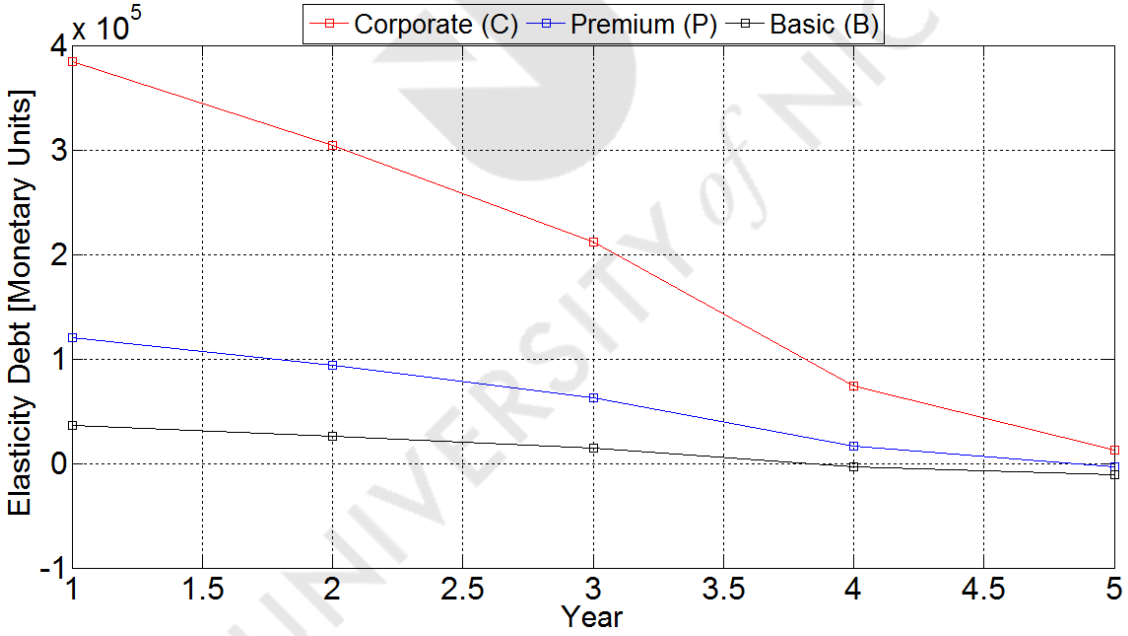


Figure 5.5. Plot for case scenario C: Flow of elasticity debt before the elasticity debt control mechanism is triggered

UNIVERSITY of NICOSIA

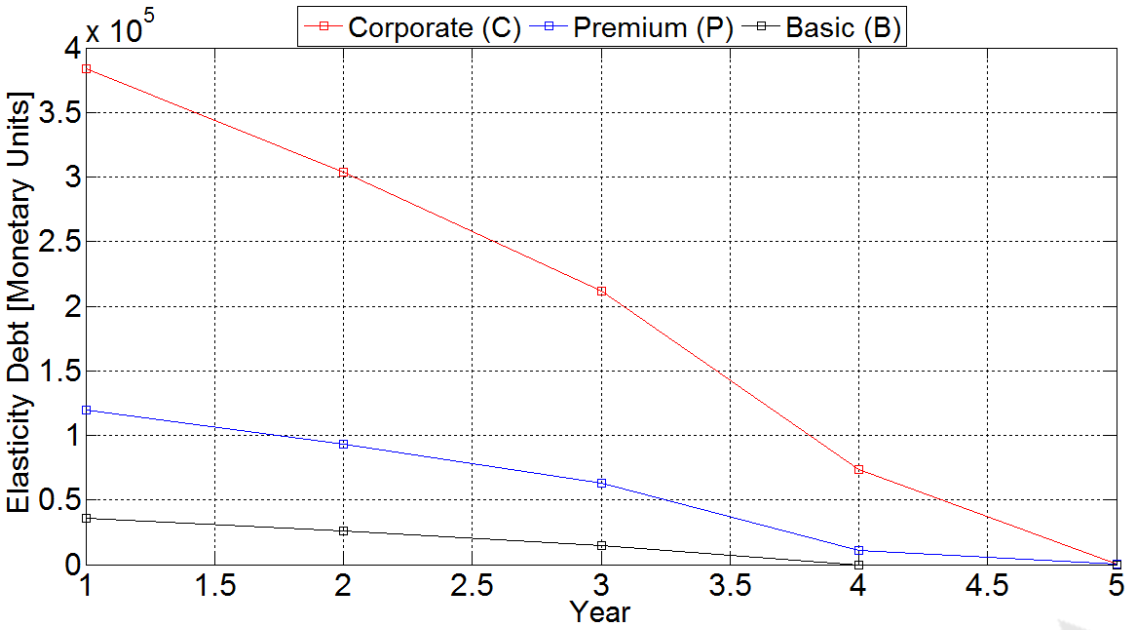


Figure 5.6. Plot for case scenario C: Flow of elasticity debt after the elasticity debt control mechanism is triggered

Table 5.5. Elasticity dent quantification results after the control mechanism is triggered

	Year 1	**Year 2**	**Year 3**	**Year 4**	**Year 5**
	Case Scenario A				
Corporate	384,000	368,860.80	353,399.55	255,107.72	87,079.06
Premium	120,000	114,796.80	109,213.18	76,338.17	17,960.81
Basic	36,000	34,131.60	32,177.27	19,758.24	15.31
	Case Scenario B				
Corporate	384,000	333,820.80	260,232.21	114,428.37	347.05
Premium	120,000	103,079.52	78,419.39	29,898.70	138.37
Basic	36,000	29,638.08	20,395.67	2,071.90	6.58
	Case Scenario C				
Corporate	384,000	303,966.72	211,830.22	73,615.02	480.90
Premium	120,000	93,445.92	62,772.84	10,769.18	165.13
Basic	36,000	26,101.44	14,587.08	0.00	0.00

UNIVERSITY of NICOSIA

5.4.2. Technical Debt-Inspired Throttling Mechanism at Cloud Service Utility Level

As presented in Chapter 4, the technical debt paradigm is applied in the context of cloud elasticity to develop a debt-aware research approach to dynamic resource adaptation in cloud elasticity management. In this direction, throughout the experimentation, two cloud-based services (i.e., premium and basic) are considered – assuming non-linear demand predictions (Table 5.6) in a 4-year technical debt prediction period ($l = 4$) – towards the validation of the efficiency of the mechanism. The service attributes are values close to real-world scenarios (Table 5.7). The variations in the subscription price per month and the service cost for an end-user at cloud service utility level are provided in Table 5.8. The variations in pricing and cost result from the constantly increasing demand for resource usage. The validation analysis focuses on identifying and measuring the technical debt status on different multi-tenant software as a service both before and after new end-user requests workload occur throughout the technical debt modeling period.

Table 5.6. Non-Linear Demand Variations

Term	**Variation in Demand**
Year 1 to 2	$\beta_1\% = 10\%$
Year 2 to 3	$\beta_2\% = 15\%$
Year 3 to 4	$\beta_3\% = 35\%$

Table 5.7. Service attributes

Variable Definition	**Premium Service**	**Basic Service**
Maximum number of active users	$U_{max} = 6{,}000$	$U_{max} = 3{,}000$
Current number of active users	$U_{curr} = 3{,}000$	$U_{curr} = 2{,}000$
Monthly subscription price	$ppm = 6$	$ppm = 3$
Monthly service cost in cloud	$Cu_{/m} = 3$	$Cu_{/m} = 2$

Table 5.8. Variations in subscription price and service cost

Variable Definition	**Premium Service**	**Basic Service**
Monthly subscription price variation	$\Delta_1\% = 0.5\%$ $\Delta_2\% = 0.8\%$ $\Delta_3\% = 2\%$	$\Delta_1\% = 0.6\%$ $\Delta_2\% = 1\%$ $\Delta_3\% = 2.5\%$
Service cost in cloud variation	$VoC_1\% = 1\%$ $VoC_2\% = 1.5\%$ $VoC_3\% = 4\%$	$VoC_1\% = 1.5\%$ $VoC_2\% = 3\%$ $VoC_3\% = 6\%$

Table 5.9. Technical debt measurement results before the throttling mechanism is triggered

	Year 1	**Year 2**	**Year 3**	**Year 4**
Premium Service	108,000	97,200	79,453.82	31,577.06
Basic Service	12,000	9,484.8	5,399.06	-4,527.44

Towards the interpretation of the obtained results in Table 5.9, the premium cloud-supported service remains underutilized throughout the technical debt modeling period due to the positive numerical results. The technical debt is gradually decreasing/optimized as the non-linear demand predictions help towards this state. On the contrary, the results for the basic cloud-supported service indicate that overutilization in the long run is witnessed (i.e., negative results obtained in year 4). The non-linear demand predictions do not motivate the technical debt optimization in this case; in this context, the technical debt-inspired throttling mechanism is motivated such that the increase for new user requests to be fully satisfied in year 4 (in terms of resource availability) is up to approximately 470 requests, i.e.,

$$3{,}000 - (1 + x) * 2{,}530 = 0 \Longrightarrow x = 0.18577 \approx 18.6\% \approx 470\ new\ end - user\ requests$$

To further explain the results, the new requests correspond to 18.6 per cent increase in the demand as shown above, when the actual predicted one corresponded to 885 new users. The proposed throttling mechanism achieves to allocate the remaining 415 new users in the premium service; the mechanism motivates the technical debt optimization for this cloud-based service. As a result, the pool of current end-users for the premium service is $5{,}123 + 415 = 5{,}538$ end-users in year 4 and the increase in the demand is now formed to 45.9 per cent, i.e.,

$$(1 + y) * 3{,}795 = 5{,}538 \Longrightarrow y = 0.45929 \approx 45.9\%$$

The latter increase in the demand will be considered during the next variation in the demand and the new, upcoming requests. As far as it concerns the premium cloud-based service, the monthly subscription price variations as well as the variations in service cost at cloud service utility level are formed to 2.1 and 4.2 per cent accordingly ($\Delta_3\% = 2.1\%$ and $VoC_3\% = 4.2\%$). The technical debt measurement calculation is approximately 16,638.81 monetary units in year 4 after the allocation of the new user requests in that service instead of the basic one, motivating the technical debt minimization. Concerning the basic cloud-based service, the monthly subscription price and service cost variations are formed to 2.2 and 5 per cent, respectively. The measurement result is approximately 0.02 monetary units in year 4, which is a very satisfying scenario in terms of achieving technical debt optimization, avoiding accumulated technical debt and motivating resource stability in cloud-centric systems. Once the throttling mechanism is triggered, the technical debt flow for both cloud-based services is witnessed in Figure 5.7.

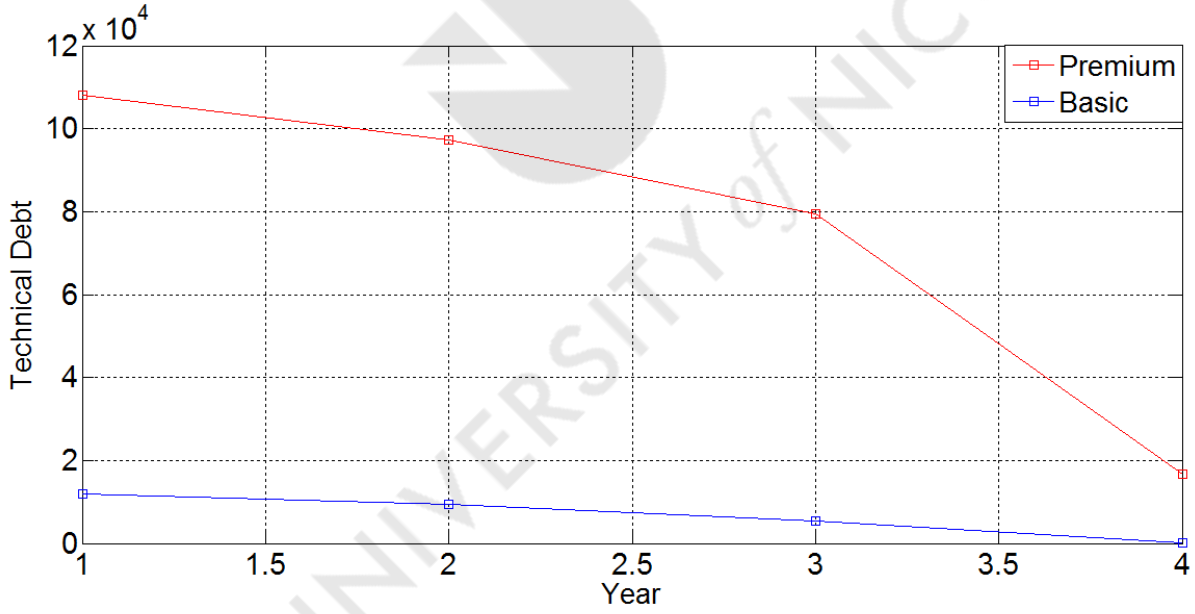


Figure 5.7. Technical debt flow after the throttling mechanism is triggered

5.4.3. Non-Cooperative Cost-Benefit Analysis Game

The validation of the proposed theorem is investigated towards the optimization of the benefits output when examining different storage systems. A cost-optimal storage allocation

incentive mechanism is elaborated, achieving optimal leasing conditions in the cloud storage and computing capacity level, parameterizing the storage currently used per tenant and auto-scaling efficiently as non-linear or linear demand for storage capacity occurs. The experimental analysis considers two storage systems in the cloud-centric environment, where non-linear demand curves are predicted in a 4-year benefits prediction period (l = 4) as shown in Table 5.10. The theorem is sufficient towards the benefits and return on investment maximization through an optimal storage allocation and control mechanism. It is important to mention that achieving optimality at the benefits level is almost illusionary as debts and interests always occur (i.e., optimal condition is achieved when $B_i = 0$); however, the Nash equilibrium game tends towards storage allocation optimality while maximizing the benefits. Each storage system's characteristics are shown in Table 5.11, whereas the variations in the total cost for leasing additional storage capacity on cloud service utility level is witnessed in Table 5.12 as a consequence of the constantly increasing demand. Throughout the evaluation analysis, an effort was made to identify whether storage upgradation will be necessary during the 4-year period of time. The obtained experimental and numerical results indicate that the necessity for storage upgradation in system 1 will not occur due to the positive values, while the benefits are constantly maximizing (as seen in Table 5.13).

Table 5.10. Use case scenario: Non-linear demand variations for cloud storage

Term	**Variation in Demand**
Year 1 to 2	$\beta_1\% = 5\%$
Year 2 to 3	$\beta_2\% = 25\%$
Year 3 to 4	$\beta_3\% = 30\%$

Table 5.11. Storage System Characteristics

Variable Definition	**Storage System 1**	**Storage System 2**
Maximum storage capacity (terabytes)	$S_{max} = 8$	$S_{max} = 5$
Storage currently used	$S_{curr} = 4$	$S_{curr} = 3$
Initial monthly cost for leasing cloud storage	$Cs_{/m} = 420$	$Cs_{/m} = 400$

Table 5.12. Variations in cost for leasing additional cloud storage

Variable Definition	**Storage System 1**	**Storage System 2**
Total cost variation for leasing additional cloud storage	$\delta_1\% = 2\%$ $\delta_2\% = 8\%$ $\delta_3\% = 9\%$	$\delta_1\% = 3\%$ $\delta_2\% = 10\%$ $\delta_3\% = 12\%$

Table 5.13. Benefits numerical results under the big data as a service framework before the storage allocation control mechanism is triggered

	Year 1	**Year 2**	**Year 3**	**Year 4**
Storage System 1	20,160	19,535.04	15,268.18	7,110.81
Storage System 2	9,600	9,146.4	5,778.3	-723,31

On the other hand, the immediate need for upgradation will be faced in the long run regarding storage system 2 (as shown in Table 5.13). Towards achieving optimality through our game theoretic formulation, the estimated increase in the demand in year 4 should have been up to 26.9 per cent, i.e.,

$$5 - (1 + x) * 3.9375 = 0 \Longrightarrow x = 0.26984 \approx 26.9\% \approx 1.06\ terabytes$$

More specifically, the actual 30 per cent that was adopted in the initial scenario corresponds to approximately 1.18 terabytes (TB). According to the storage allocation control mechanism, the remaining 0.12 TB is allocated automatically in storage system 1, helping towards the benefits maximization for this system. Therefore, since the storage allocation control mechanism is initiated, the storage currently used for system 1 is approximately $6.825 + 0.12 = 6.837$ TB in year 4 and the true increase in the demand is now formed to approximately 30.2 per cent, which is estimated as follows

$$(1 + y) * 5.25 = 6.837 \Longrightarrow y = 0.30229 \approx 30.2\%$$

This true increase in the demand will be considered in the next variation in the demand and not the actual 30 per cent that was initially adopted when investigating this case scenario. Furthermore, the variation in the total cost for leasing additional storage for storage system 1 is now restructured to 9.1 per cent (i.e., $\delta_3\% = 9.1\%$) and, thus, the benefits numerical result for the storage system 1 will be finally formed to approximately 7,053.73 monetary units ($B_4 \approx 7{,}053.73$) in year 4. Likewise, the variation in the total cost for the storage system 2 will be reformed to 11 per cent (i.e., $\delta_3\% = 11\%$) and, therefore, the benefits calculation is now formed to approximately 0.03 monetary units ($B_4 \approx 0.03$) in year 4, tending towards optimality in terms of maximizing the benefits. Once the storage allocation mechanism is triggered, the final flow of the benefits for both systems is observed in Figure 5.8. In this context, the proposed game theoretic formulation motivates the need to minimize the risk for storage upgradation in the long run that would lead to service level objective violations.

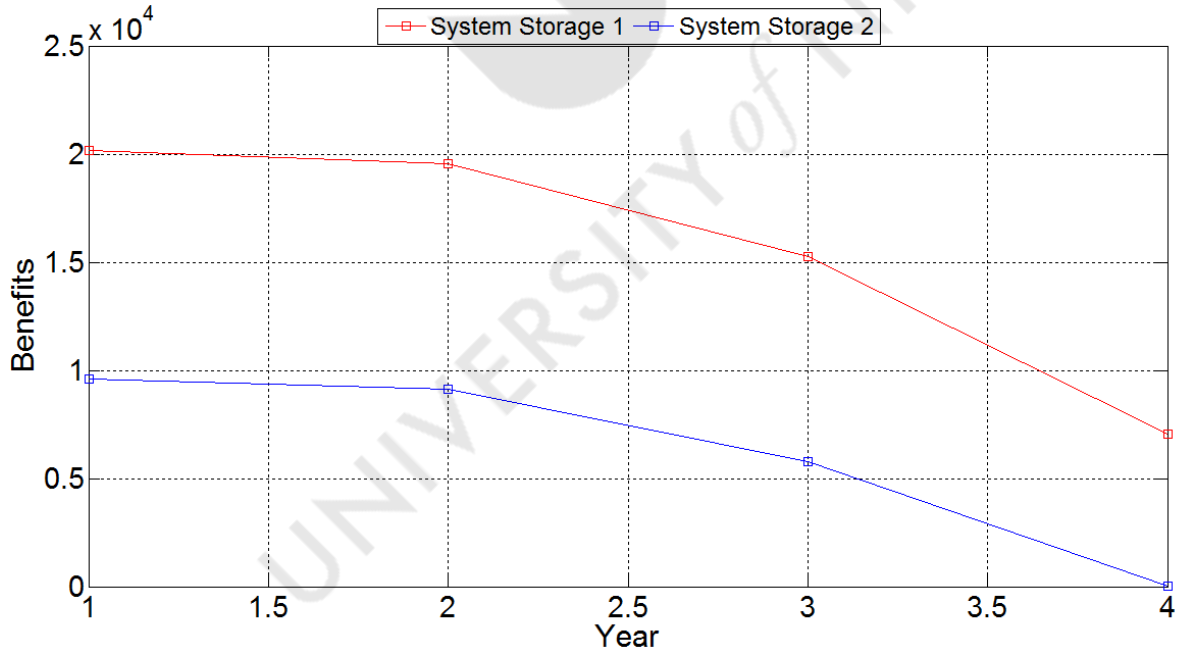


Figure 5.8. Flow of benefits after the storage allocation control mechanism is triggered

UNIVERSITY of NICOSIA

5.5.Conclusions

In this chapter, a novel game theoretic incentive scheme has been proposed for the problem of optimal and dynamic resource adaptation in cloud and mobile cloud computing systems. The proposed control mechanisms optimally solve the service utility overutilization problem by optimizing the trade-off decisions, forming dynamic coalitions and scheduling resources while analysing the given state of debt minimization and profit maximization based on a Hidden Markov Model. The modeling for the computation of the processing time, energy and overhead in mobile opportunistic offloading is also discussed. The experimental study proves that the theorem is efficient, investigating the applicability of debt minimization- and profit maximization-aware game theoretic control mechanisms for optimal resource utilization with well-compromised trade-offs using different cloud service capabilities and storage capacities. The proposed game theoretic technique can be applied in further future research on more complex scenarios with dynamic QoS / QoE analysis and resource allocation in a second scale for short-term predictions. In the next chapter, the mechanisms are integrated into a tool for debt- and cost-aware quantification at cloud service utility level in the context of multi-tenant software as a service.

UNIVERSITY of NICOSIA

Chapter 6

Prototype Tool for Debt-Aware Quantification in Cloud Computing Environments

The previous chapters have introduced a debt-aware quantification approach to dynamic resource adaptation in cloud elasticity management. This chapter presents 'Technical Debt in Cloud', a novel framework for debt-aware quantification at cloud service utility level in multi-tenant software as a service. The implementation of the quantification tool includes the models and algorithms proposed in the previous chapters. The architecture, technologies and implementation of the proof of concept prototype are described in detail, followed by the product evaluation through the testing plan.

6.1. Introduction

This chapter introduces the requirements, architecture, technologies and implementation of 'Technical Debt in Cloud' (TDinCloud), a simulation tool for debt-aware quantification in cloud computing environments. The tool is designed following the debt-aware research approach to optimal resource provisioning and utilization at cloud service utility level introduced and evaluated in Chapters 3, 4 and 5, providing computations about different case scenarios and insights into the following:

1. Underutilization / overutilization of service components in multi-tenant cloud-supported services, the gradual payoff of the elasticity debt and the threshold point at which it will be totally cleared out.
2. Cost estimations for software as a service (SaaS) implementation in cloud exploiting the Constructive Cost Model (COCOMO) [265].

From a technical point of view, the web application is targeted to be deployed in the Google Cloud Platform supported by the Google App Engine [291] and it was implemented using the Java programming language. The end-users that can be benefited by using the quantification tool are cloud experts – e.g., software project managers, cloud architects, cloud service evaluators, software engineers / developers or analysts – that focus on the evaluation of cloud services for selection. Two UML activity diagrams are presented in Figures 6.1 and 6.2, presenting a workflow of operations in the tool (sequential, branched or concurrent), such as

how to create / view a new project from a project management viewpoint (project start-up phase – workflow 1) or create / view debt estimations (workflow 2). The analysis motivates debt-awareness to manage effectively the elasticity debt and avoid the options of abandoning/termination and switching that lead to SLO violations and new accumulated debt difficult to be managed.

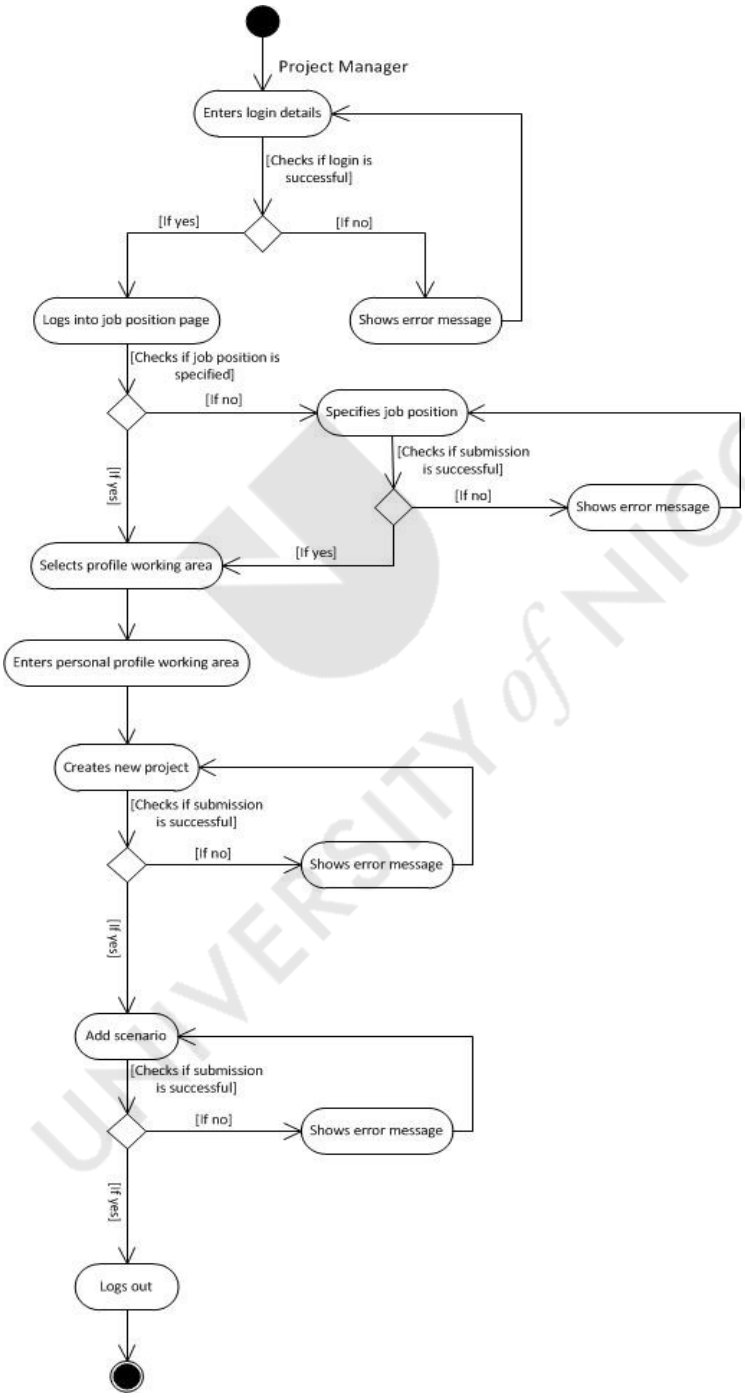


Figure 6.1. UML Activity Diagram – Workflow 1

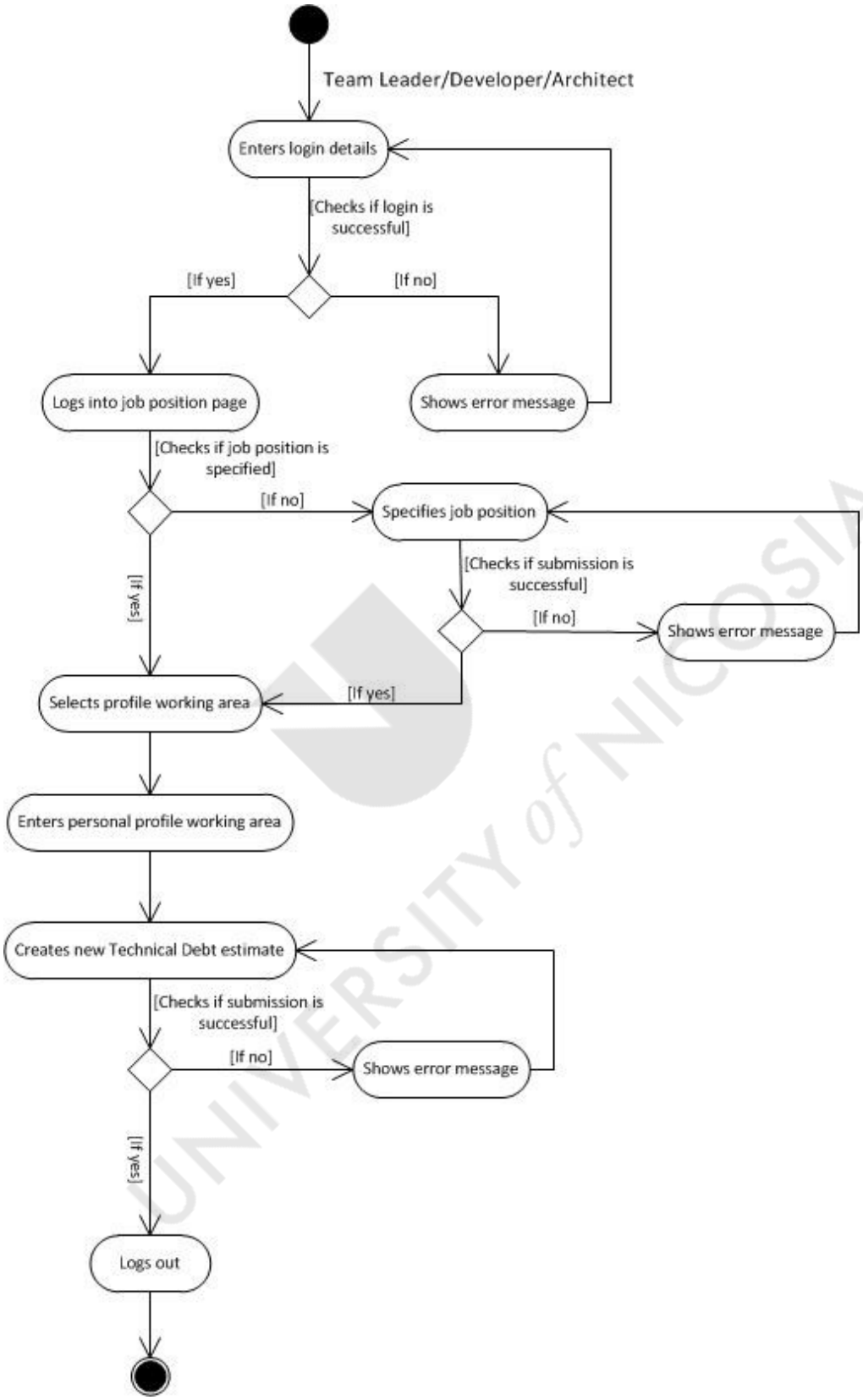


Figure 6.2. UML Activity Diagram – Workflow 2

The key contributions of this chapter are the following:

1. Requirements elicitation and analysis from an end-user perspective.
2. Architecture of the developed quantification tool, providing a detailed analysis and solution design.
3. Implementation of models and algorithms for debt-aware resource adaptation in cloud proposed and evaluated by experiments in the previous chapters.
4. Testing and product evaluation.

The remainder of the chapter is organized as follows: the next section analyses the system requirements from a key stakeholder viewpoint. Section 6.3 presents the overall design and architecture of each component of the quantification tool, following by a testing plan for the product evaluation in Section 6.4. The chapter is concluded with a summary in Section 6.5.

6.2.Requirements Elicitation and Analysis

The key requirements of the tool functionality have been divided into functional and non-functional (the latter for the evaluation of the tool operation) and each requirement is further divided into sub-requirements where necessary. The non-functional requirements have been classified according to Sommerville's methodology [292] for non-functional requirement types.

6.2.1. Functional Requirements

6.2.1.1 Create and view COCOMO estimation

6.2.1.1.1 The user can create a new COCOMO estimation.

6.2.1.1.1.1 The user should provide a COCOMO estimation name.

6.2.1.1.1.1.1 The user should provide a unique estimation name that does not already exist in the database to avoid duplications and redundancies.

6.2.1.1.1.2 The user should provide a software development mode statement.

UNIVERSITY of NICOSIA

6.2.1.1.1.3 The user should provide the software product size in thousands source lines of code (KLOC).

6.2.1.1.1.4 The user should be able to make calculations based on the provided elements.

6.2.1.1.1.4.1 The user should be able to view the optimistic, most likely and pessimistic applied effort for a software product.

6.2.1.1.1.4.2 The user should be able to view the required development time for a software product.

6.2.1.1.1.4.3 The user should be able to view the required resources (number of people) for a software product.

6.2.1.1.1.5 The user should provide a confidence statement for the estimation.

6.2.1.1.1.6 The user should provide a justification.

6.2.1.1.1.7 The user should be able to reset the COCOMO estimation details.

6.2.1.1.1.8 The user should fill in all the necessary fields in the COCOMO estimation form without leaving any blank fields.

6.2.1.2 Create and view cost estimation for software as a service (SaaS) implementation in cloud

6.2.1.2.1 The user can create a new cost estimation.

6.2.1.2.1.1 The user should provide an estimation name.

6.2.1.2.1.1.1 The user should provide a unique cost estimation name that does not already exist in the database to avoid duplications and redundancies.

6.2.1.2.1.1.2 The user should provide a weighted priority rating for the Development process as percentage.

6.2.1.2.1.1.3 The user should provide a weighted priority rating for the Configuration process as percentage.

6.2.1.2.1.1.4 The user should provide a weighted priority rating for the Deployment process as percentage.

6.2.1.2.1.5 The user should provide a weighted priority rating for the License process as percentage.

6.2.1.2.1.6 The user should provide a weighted priority rating for the Infrastructure process as percentage.

6.2.1.2.1.7 The user should provide the most likely applied effort estimation in man-months.

6.2.1.2.1.8 The user should provide an average monthly salary per employee.

6.2.1.2.1.9 The user should be able to make calculations based on the provided elements.

6.2.1.2.1.9.1 The user should not be able to make calculations if the sum of all weighted priority ratings is not 100%.

6.2.1.2.1.9.2 The user should be able to view the optimistic, most likely and pessimistic Development cost for software implementation in cloud.

6.2.1.2.1.9.3 The user should be able to view the optimistic, most likely and pessimistic Configuration cost for software implantation in cloud.

6.2.1.2.1.9.4 The user should be able to view the optimistic, most likely and pessimistic Deployment cost for software implementation in cloud.

6.2.1.2.1.9.5 The user should be able to view the optimistic, most likely and pessimistic License cost for software implementation in cloud.

6.2.1.2.1.9.6 The user should be able to view the optimistic, most likely and pessimistic Infrastructure cost for software implementation in cloud.

6.2.1.2.1.9.7 The user should be able to view the optimistic, most likely and pessimistic total cost for software implementation in cloud.

6.2.1.2.1.10 The user should provide a confidence statement for the cost estimation.

6.2.1.2.1.11 The user should provide a product flexibility statement.

6.2.1.2.1.12 The user should provide a market flexibility statement.

6.2.1.2.1.13 The user should provide a risk statement for entering into a debt in the future.

6.2.1.2.1.14 The user should provide a real options valuation statement if debt tends to occur in the long run.

6.2.1.2.1.15 The user should provide a justification.

6.2.1.2.1.16 The user should be able to reset the cost estimation details.

6.2.1.2.1.17 The user should fill in all the necessary fields in the cost estimation form without leaving any blank fields.

6.2.1.3 Create and view debt estimations at cloud service utility level

6.2.1.3.1 The user can create a new debt estimation at cloud service utility level.

6.2.1.3.1.1 The user should provide a debt estimation name.

6.2.1.3.1.1.1 The user should provide a unique estimation name that does not already exist in the database to avoid duplications and redundancies.

6.2.1.3.1.2 The user should provide the modeling period in years (return on investment-inspired).

6.2.1.3.1.3 The user should provide the maximum capacity of the cloud service (active end-users).

6.2.1.3.1.4 The user should provide the current number of active end-users.

6.2.1.3.1.5 The user should provide an average variation in the yearly resource demand as percentage.

6.2.1.3.1.6 The user should provide a monthly subscription price (monetary units).

6.2.1.3.1.7 The user should provide an average variation in the monthly subscription price for the declared modeling period as percentage.

6.2.1.3.1.8 The user should provide a monthly service cost in cloud (monetary units).

6.2.1.3.1.9 The user should provide an average variation in the monthly service cost in the cloud for the declared modeling period as percentage.

6.2.1.3.1.10 The user should be able to make calculations based on the provided elements.

6.2.1.3.1.10.1 The user should be able to view the yearly debt quantifications.

6.2.1.3.1.11 The user should provide a confidence statement for the debt estimation.

6.2.1.3.1.12 The user should provide a scalability/market adaptability statement.

6.2.1.3.1.13 The user should provide a Quality of Service (QoS) statement according to non-functional requirements (i.e., availability, performance).

6.2.1.3.1.14 The user should provide a risk statement for entering into a debt status in the future.

6.2.1.3.1.15 The user should provide a real options valuation statement if debt tends to occur in the long run.

6.2.1.3.1.16 The user should provide a justification.

6.2.1.3.1.17 The user should be able to reset the details of the debt estimation.

6.2.1.3.1.18 The user should fill in all the necessary fields in the debt estimation form without leaving any blank fields.

6.2.1.4 Create debt graph, dashboard, data visualization and quantitative report

6.2.1.4.1 The user should be able to select created or shared debt estimations to create graphs, dashboards and tables (with quantitative data).

Rationale: To view the flow of debt quantifications and compare different estimations at cloud service utility level (dashboard and report functionality presented in Figure 6.3, while share functionality in Figures 6.4 and 6.5).

6.2.1.4.1.1 The user should have and select at least two debt estimations to create a dashboard.

6.2.1.4.1.2 The user should be able to compare estimations with different modeling periods.

6.2.1.4.1.3 The user should be able to view a table / quantitative report, containing the yearly debt quantification results for different estimations.

6.2.1.4.1.4 The user should be able to view a table / quantitative report with the most important features and attributes of each estimation.

UNIVERSITY of NICOSIA

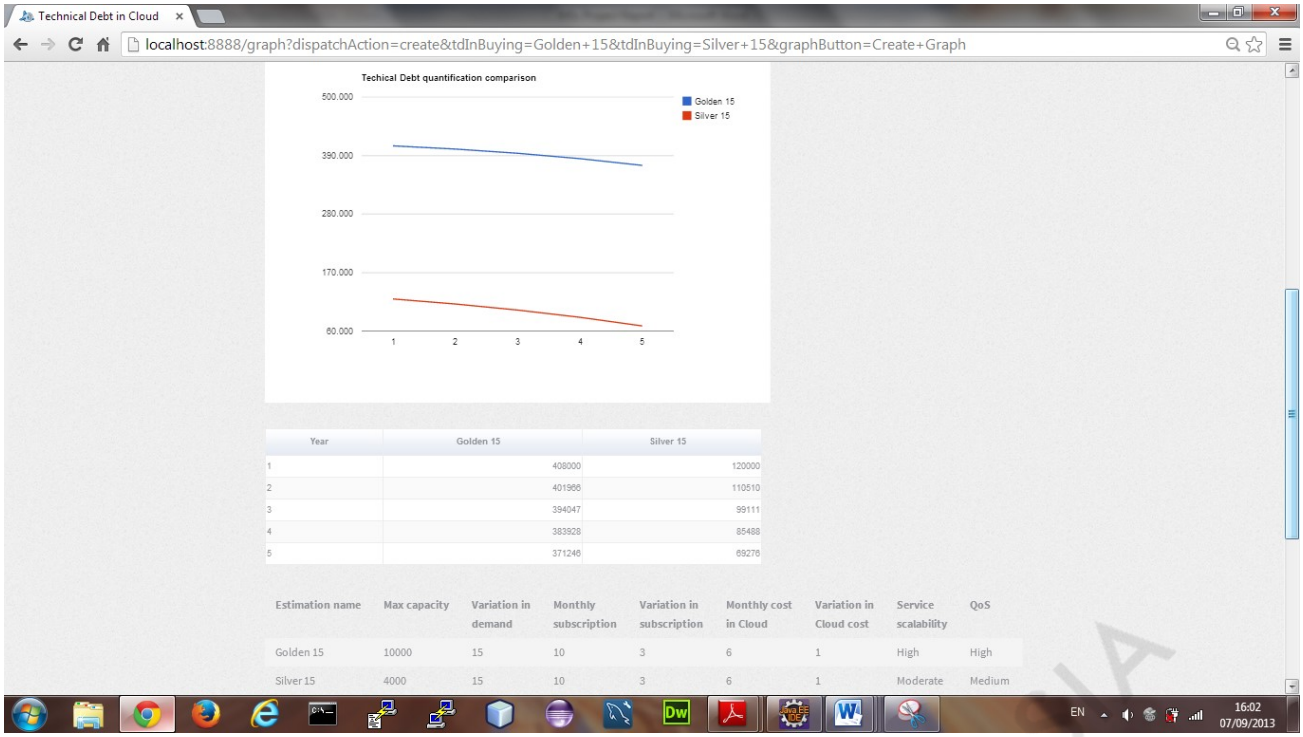


Figure 6.3. Dashboard and report functionality with debt quantifications

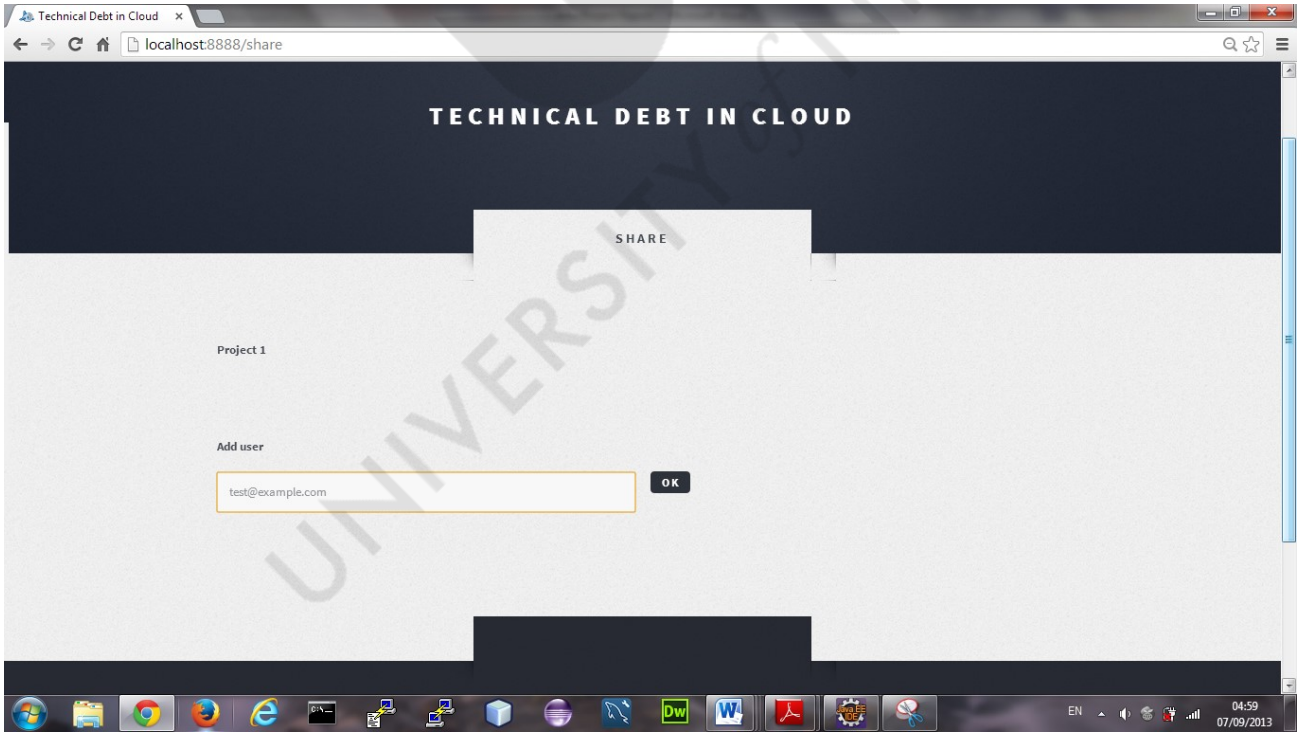


Figure 6.4. Share functionality: Add user

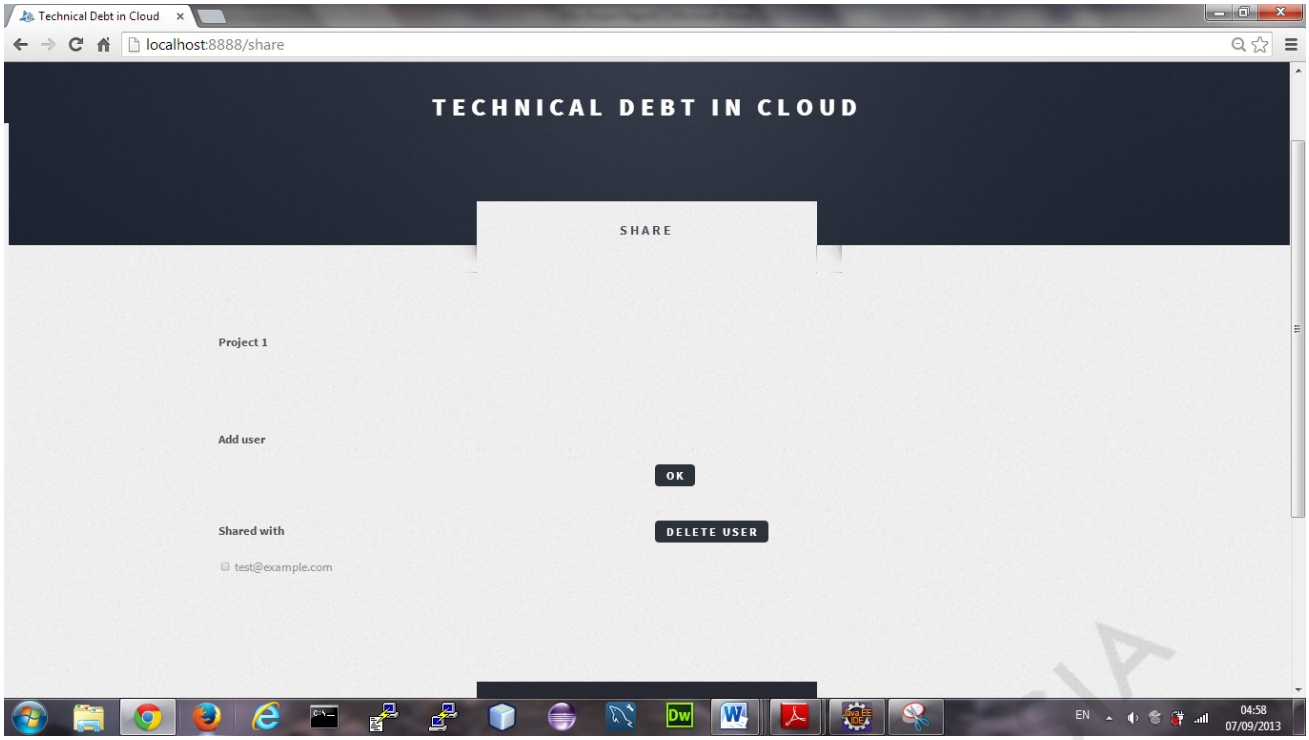


Figure 6.5. Share functionality: Receiver added

6.2.2. Non-Functional Requirements

6.2.2.1 Product requirements

Efficiency (Performance)

6.2.2.1.1 The users should be able to operate the tool with a 10 second maximum response time.

Rationale: To ensure that the tool will operate quickly during the working day.

6.2.2.1.2 The tool should be able to avoid SQL joins.

Rationale: To ensure that the tool will respond quickly during the working day.

Efficiency (Space)

6.2.2.1.3 The tool should have an on-demand frontend instances capacity for up to 28 instance hours per day according to the Google App Engine free account limitations.

UNIVERSITY of NICOSIA

6.2.2.1.4 The cloud-based system/tool should allow network transmissions for outgoing and incoming bandwidth for up to 1 GB per day each according to the Google App Engine free account limitations.

6.2.2.1.5 The cloud-based system/tool should have a database size that fits up to 1 GB.

Usability (Ease of Use)

6.2.2.1.6 The tool should allow the user to customize different projects and scenarios or estimations and calculations.

Rationale: To ensure the tool's adaptability for different operations.

6.2.2.1.7 An experienced user, who is familiar with other project or quantification tools, should be able to understand and operate all features of the web application within 30 minutes.

6.2.2.1.8 The prototype tool should have a standard "look and feel" in compliance with the World Wide Web Consortium's (W3C) design standards.

Rationale: To ensure that the tool is appealing and easy for the user to use.

6.2.2.1.9 The interface of the web application should be simple, recognizable and understandable by the user.

Rationale: To ensure that the application is appealing and easy for the user to use.

6.2.2.1.9.1 The fonts will be clear and of medium size to ensure readability.

Usability (Error Tolerance)

6.2.2.1.10 The users should be informed about the success or failure of any operation or computation.

Rationale: To ensure high-level interaction between the tool and the user informing about the outcome of the performed operations.

Dependability (Availability, Reliability and Robustness)

6.2.2.1.11 The tool should be available for update and enquiry 99.9% of the time during the working day according to Google Cloud Service Level Agreement.

Rationale: To ensure reliability, as several downtimes can discourage the users.

6.2.2.1.11.1 A breakdown of any of the sub-systems should not affect other parts of the tool.

6.2.2.1.11.2 The administrator should be able to maintain the system while this is running.

6.2.2.1.11.3 In case of a system crash, the basic functionalities should be maintained by the Google's backup system.

Dependability (Maintainability and Repairability)

6.2.2.1.12 The coding process should follow the Java coding and Google Cloud SQL conventions.

Rationale: All future changes and modifications should be done according to these conventions.

6.2.2.1.12.1 The different parts of the system should be encapsulated in such a way that changes in one part are unlikely to have a negative impact on other parts.

Security (Resistance to Deliberate or Accidental Intrusion)

6.2.2.1.13 The system will ensure that only authenticated users should be able to perform operations on that.

Rationale: To avoid possible deliberate or accidental attacks that would put into question the system's reliability.

6.2.2.1.14 Inputs from the users will be validated.

Rationale: To prevent SQL injection attacks and ensure that strong validation is present in the system.

6.2.2.1.14.1 The system should ensure that the users will fill in the correct typed inputs in each form through prepared queries and only the correct inputs will be stored in the database.

6.2.2.2 Organizational requirements

Environmental (Operating Platform)

6.2.2.2.1 The web application should operate on Linux, Microsoft Windows and Mac OS platforms.

Rationale: To attract as many users as possible.

Operational (Interoperability)

6.2.2.2.2 The system should fully work and be compatible with Google Chrome, Mozilla Firefox, Internet Explorer, Opera and Safari web browsers (as a server- and platform-independent application).

Rationale: To attract as many users as possible.

6.2.2.2.2.1 The system should employ technology that is supported by all web browsers.

6.2.2.2.2.2 All browsers can easily remove the cookies.

6.2.2.2.3 There are no additional requirements for other interfaces with other systems.

Development (Process Standards)

6.2.2.2.4 The system should be developed in compliance with the TCP/IP communication standards.

6.3. Architectural Analysis and Design

This subchapter provides information on how the quantification tool was conceptually designed. It is given an account on how the problem was approached and the process followed in order the final tool to support the analysis and quantification of the elasticity debt. Entity-Relationship (ERD) and UML diagrams are provided to give a blueprint for the design. From an implementation decision point of view, the web application was developed to be compatible with desktop computers, tablet computers and mobile phones. A proof of this respect is the non-existence of pop-up windows in the tool. The fact that mobile and tablet devices have become powerful is an opportunity to create a tool that maximizes the use of their features, not to mention that the possibility of a larger market share is higher. The web application enables many users to have online access, working in a distributed manner and allowing stakeholders to give estimations from different locations (localization-based approach).

6.3.1. UML Class Diagram and Database Design

A UML class diagram is provided in Appendix A: UML Class Diagram and Database Design, presenting the core design of the adopted object-oriented method, and giving information about the static structure of the tool, including classes, attributes, methods and class relationships. In addition, the database design is critical part of the development phase and the overall tool implementation; in this direction, the database was designed to operate quantities of information and data by inputting, storing, retrieving and managing that information and data. An Entity-Relationship Diagram (ERD) presenting the entities, relationships and attributes of the quantification tool's database is provided in Appendix A: UML Class Diagram and Database Design.

6.3.2. Implementation

This subchapter provides information about the implementation techniques and technologies used for the design, classes, database and user interface (UI). It focuses on the front-end, such as the interfaces, forms and screens, and back-end classes involving the implementation related to the web server and database. The source code of the prototype tool is available online in the GitHub repository[1]. To further elaborate on the technologies used, the web application has been developed using the following:

1. *Java Servlet and JavaServer Pages (JSP) technology on the Java Web Development Server*. The web server provided by Google App Engine and simulates the App Engine Java runtime environment and all of the services including the datastore. All servlets extend the `HttpServlet` class and an XML file (`web.xml`) maps the servlets to the corresponding URLs.
2. *Hibernate framework with JPA and XML specifications*. The framework is used to map Java classes to database tables and Java data types to SQL data types. The combination of Hibernate and JPA technologies guarantees the connectivity and transactional operations with the data layer. JPA generates a file (`persistence.xml`), which keeps track of the classes that represent the model in order to ensure the mapping between the POJOs and the database tables. The database tables are consistent and transparent with

[1] The source code of the prototype tool. https://github.com/Georgios-Skourletopoulos/prototype-tool

the POJOs; the `EntityManagerFactory` opens the database connection with the properties specified in the `persistence.xml` file, while the `EntityManager` initiates the connection to the database.

3. *JavaScript with HTML and CSS.*
4. *Client- and server-side validations*. The validations are used to communicate processing deviations, validate the existing HTML forms, enforce restrictions that the right data input is inserted in the fields according to the datatypes, and avoid duplications and redundancies in the database. Validations for different events are provided in Appendix B: Client- and Server-Side Validations.
5. *MySQL Database Management System*. The database tables with relevant descriptions are provided in Table 6.1. The database schema is provided in Appendix C: Database Management System.

The tool has been designed using a three-tier web architecture consisting of the client, application and data tiers. A deployment diagram for the adopted three-tier architecture is provided in Figure 6.6.

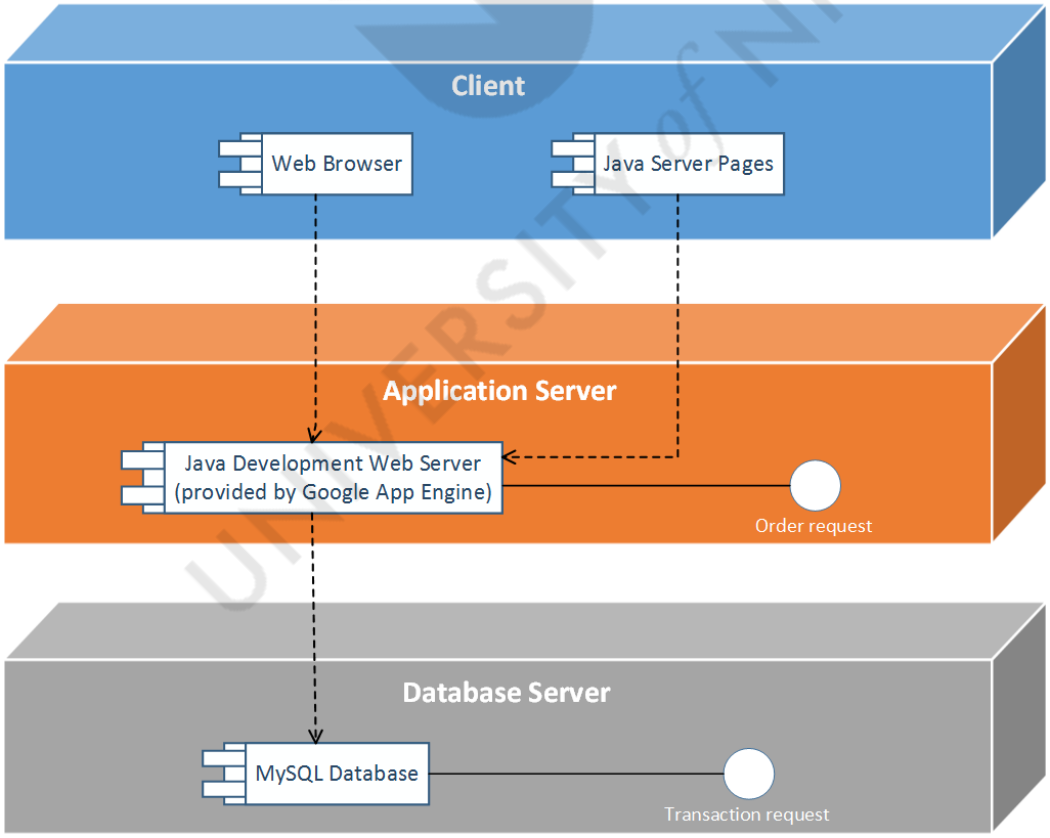


Figure 6.6. Three-Tier Web Architecture – Deployment Diagram

UNIVERSITY of NICOSIA

Table 6.1. Database Tables

Database Table	Database Table Description
`cocomo`	Store data of new COCOMO estimations.
`costforimplementing`	Store data of new cost estimations for software as a service implementation in cloud.
`jobtitle`	Store user job positions.
`project`	Store data of new projects.
`scenario`	Store data of new case scenarios.
`sharedbuying`	Store shared debt estimation IDs, sender's and receiver's IDs.
`sharedcocomo`	Store shared COCOMO estimation IDs, sender's and receiver's IDs.
`sharedimplementing`	Store shared cost estimation IDs, sender's and receiver's IDs.
`sharedproject`	Store shared project IDs, sender's and receiver's IDs.
`tdinbuying`	Store data of new debt estimations at cloud service utility level.
`userlogin`	Store user login details.

6.3.2.1. Error Handling

All errors were handled and, where needed, they redirect to the error page which displays the relevant message. The internal errors of the servlets and any thrown errors redirect to the mentioned page. In the case of problems from a web point of view (e.g., cannot find the required resources to use), the user shall be presented with the error page that has details about the corresponding error code (i.e., 400, 403, 404, 500). Below it is shown how the `web.xml` file maps the error codes to the error page (Figure 6.7).

UNIVERSITY of NICOSIA

web.xml

```
    </servlet>
    <servlet-mapping>
        <servlet-name>errorServlet</servlet-name>
        <url-pattern>/error</url-pattern>
    </servlet-mapping>

    <error-page>
        <error-code>400</error-code>
        <location>/error</location>
    </error-page>
    <error-page>
        <error-code>403</error-code>
        <location>/error</location>
    </error-page>
    <error-page>
        <error-code>404</error-code>
        <location>/error</location>
    </error-page>
    <error-page>
        <error-code>500</error-code>
        <location>/error</location>
    </error-page>
    <error-page>
        <exception-type>java.lang.Throwable</exception-type>
        <location>/error</location>
    </error-page>
    <welcome-file-list>
        <welcome-file>HibTDQ</welcome-file>
    </welcome-file-list>
</web-app>
```

Figure 6.7. Mapping error codes to error page

6.4. Testing and Product Evaluation

Communicating processing deviations, validating the HTML forms, enforcing restrictions that the right data input is inserted in the relevant fields according to the corresponding datatypes, and avoiding duplications and redundancies in the database can be achieved by implementing validations that were discussed in Section 6.3.2. In this section, the testing plan is elaborated running JUnit tests for the methods, database and servlets. The detailed testing is provided in Appendix D: Testing and Product Evaluation. Integration testing is also included in this appendix for different use cases to evaluate the functionality, combinations and different modules that make up a process operate efficiently, followed by a product evaluation plan to assess if the initial requirements are fully or partially met as discussed in Section 6.2.

To test the database, a new `EntityManagerFactory` was created with connection details to the local MySQL database. A representation of the class and the implemented source code for that purpose is shown in Figure 6.8. Also, the correctness of the connection to the local MySQL database is guaranteed including the persistence unit in the `persistence.xml` file as shown in Figure 6.9. Two phases of database testing were adopted:

1. Testing for the methods included in the `DBService` class (`DBServiceTest` class).
2. Testing for the basic level of persisting (i.e., insert, delete, update) the entities represented by the POJOs (`PersistenceLevelTest` class).

EMF.java

```
import javax.persistence.EntityManagerFactory;

/**
 * EMF is a class that deals with creating a singleton for the EntityManagerFactory.
 * @author Georgios Skourletopoulos, by adapting code from Google (2012), "Getting Started with JPA
 * facets and Cloud SQL", https://developers.google.com/appengine/articles/using_jpa_tool [accessed 9 Aug 2013]
 * @version 1 August 2013
 */
public class EMF {

    /**
     * For running the application.
     */
    private static final EntityManagerFactory emfInstance = Persistence
            .createEntityManagerFactory("transactions-optional");    //Singleton EMF field

    /**
     * For running JUnit tests.
     */
    //    private static final EntityManagerFactory emfInstance = Persistence
    //            .createEntityManagerFactory("testing");

    private EMF() {}    //the constructor

    /**
     * Getter for singleton.
     * @return emfInstance is the result
     */
    public static EntityManagerFactory get() {
        return emfInstance;
    }
}
```

Figure 6.8. Creation of new `EntityManagerFactory`

```
<!-- Persistence unit for the JUnit testing. It contains the properties
    to connect to a local MySQL testing database. -->
<persistence-unit name="testing">
    <description>JPA testing Persistence</description>
```

Figure 6.9. Persistence unit for connecting to local MySQL testing database

To test the servlets, the Mockito framework was used. As part of the testing process, the web application requires a user to be logged in. The Google App Engine provides such a method to overcome this issue, allowing to setup and declare a user. The method in the source code is presented in Figure 6.10.

UNIVERSITY of NICOSIA

```
private final LocalServiceTestHelper helper =
        new LocalServiceTestHelper(new LocalUserServiceTestConfig())
.setEnvIsLoggedIn(true)
.setEnvAuthDomain("example.com")
.setEnvEmail("testUser1@example.com");
```

Figure 6.10. User Setup

6.5.Conclusions

This chapter presented the architectural design and implementation of a quantification tool for debt-aware quantification in cloud computing environments, called 'Technical Debt in Cloud'. The tool provides debt-aware analysis to optimal resource utilization at cloud service level that was discussed in the previous chapters. It provides computations about different case scenarios and insights into the underutilization / overutilization states of cloud-supported services, gradual payoff of the elasticity debt, and the threshold point at which it will totally cleared out. It also provides cost estimations for software as a service (SaaS) implementation in cloud, exploiting the Constructive Cost Model (COCOMO). The tool has been designed following an initial requirements elicitation and analysis phase from a key stakeholder viewpoint, as the end-users that can be benefited are cloud experts and architects that focus on the evaluation of cloud services. The tool can be applied in further future research on the debt-aware resource adaptation field to improve the utilization of cloud resources in multi-tenant software as a service applications.

UNIVERSITY of NICOSIA

Chapter 7

Conclusions and Future Directions

This chapter summarizes the research work on debt-aware resource adaptation in cloud elasticity management presented in this research thesis and highlights the main findings. Open research problems in the area are discussed and future research directions are outlined.

7.1. Conclusions and Discussion

This research thesis presents a solution to the debt-aware dynamic resource adaptation problem in cloud elasticity management with value creation and strategy-driven incentive schemes. The principles of the technical debt metaphor are mapped into the context of cloud elasticity management and service composition to develop this innovative economics-driven framework that considers optimal trade-offs of compromising long-term benefits for short-term gains when adapting resource provisioning, without affecting the system or application utility and under the uncertainty of the cloud environment. A major aspect of dynamic resource adaptation is that due to arbitrary workload variations, the composition encounters underutilization or overutilization on the composite service components. In the underutilization state, the pay-off is significant in the long run with far-sighted benefits, while in the overutilization state, there is no pay-off and, if exists, it is trivial. Such a notion perfectly matches the intentional and unintentional technical debt encountered in the software engineering research area.

The goal is to optimize resource provisioning and utilization in multi-tenant software as a service cloud environments under balanced trade-off decisions in cloud service composition. The value of the gaps in resource provisioning between ideal and actual adaptation decisions – recognised between resource supply and demand as a match between these two is imperfect – is driven by the underutilization or overutilization states on a composite service component utility and result from the partial usage waste after one machine is released but still charged until the end of the billing cycle.

The novelty of this research thesis on the theory and practice of cloud elasticity management is summarized as follows:

1. A classification, analysis and survey of the state of the art in the cloud elasticity management research area.
2. An economics-driven approach to support value creation and strategy incentive mechanisms in cloud elasticity management.
3. A cost estimation model for software as a service implementation in the cloud, exploiting the COCOMO estimations and applying a tolerance value for prediction.
4. An elasticity debt-aware research approach to reason about elasticity resource adaptations after mapping the technical debt metaphor into the context of cloud elasticity management under uncertainty.
5. A game theoretic incentive scheme with debt-aware elasticity considerations to optimally solve the problem of composite service utility overutilization detection as a part of the dynamic resource adaptations in cloud. The formation of dynamic coalitions and optimization of trade-off decisions motivates the exchange of resource capacity among tenants based on their debt status, boosting the fairness between consumers and providers in multi-tenant software as a service environments.
6. The implementation of a prototype tool for debt-aware quantification at cloud service utility level in multi-tenant applications hosted in the cloud.

7.2.Future Research Directions

Despite substantial contributions of this thesis in debt-aware dynamic resource adaptation in cloud elasticity management, there is a number of open research challenges that need to be addressed to further advance the area. In particular, the following key issues are encountered:

1. *Investigation of more complex debt-aware scenarios for elasticity resource adaptation with dynamic QoS / QoE analysis and resource allocation in a second scale for short-term predictions*.
2. *Consideration of opportunity cost-aware mechanisms in runtime resource adaptations*. Retrospective valuation mechanisms in runtime adaptation decisions in the context of opportunity cost-aware are essential as some decisions might not be the appropriate in the long run due to the context of the scenario at the time of the analysis.
3. *Consideration of complex dynamic coalitions in multi-tenancy*. Forming dynamic coalitions can enable the dynamic sharing of virtual resources even in arbitrary workloads while preserving the diversity of a tenant's SLOs.

4. *Consideration of energy-aware mechanisms in the economics-driven analysis*. Energy-aware decisions in cloud elasticity management are imperative to measure the impact of energy consumption when provisioning virtual resources [141]; the quantification of the value extracted from the waste of energy hidden in elasticity adaptations is still a challenge as the trade-off decisions between energy savings and the latency in adaptation decisions are not adequate addressed [142].
5. *Consideration of sensitivity analysis aspects in runtime resource adaptations*.
6. *Exploitation of debt-aware resource adaptation patterns and development of a knowledge base for value creation in resource adaptations*. Future adaptation decisions with strict time constraints can benefit from previous knowledge and increase the accuracy of elasticity [145], [146].

7.3. Final Remarks

This research work motivates the need for resource adaptation decisions in cloud elasticity management using debt-aware mechanisms with value creation and strategy-driven considerations. The imperfections met in cloud service utility are encountered by quantifying the utility gaps in resource utilization as a dimension to drive elasticity adaptation decisions. Debt-aware dynamic resource adaptation will enable cloud resource providers to successfully offer scalable service provisioning, adopting debt analysis in cloud elasticity management. An aspect distinguishing the work presented in this thesis compared to related research in cloud elasticity management is the adoption of game theoretic control mechanisms to automatically re-allocate resources from an overutilized service component utility that will directly influence the resource provisioning in multi-tenant software as a service cloud environments. This game theoretic incentive scheme with debt-aware considerations for value creation is based on an equilibrium model and motivates the optimal auto-scaling of resources. The control mechanisms optimally solve the composite service component utility overutilization problem by optimizing trade-off decisions and scheduling resources while analysing the state of debt minimization and profit maximization based on a Hidden Markov Model. Research, such as presented in this thesis, will drive further innovation in cloud elasticity management and the development of a comprehensive framework for value creation in cloud elasticity management.

UNIVERSITY of NICOSIA

Bibliography

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, NIST Special Publication 800-145, Sep. 2011.

[2] N. Grozev and R. Buyya, "Inter-Cloud architectures and application brokering: taxonomy and survey," *Softw. Pract. Exp.*, vol. 44, no. 3, pp. 369–390, 2014.

[3] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] A. O'sullivan and S. M. Sheffrin, *Economics: Principles in Action*. Pearson-Prentice Hall, 2007.

[5] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini, "DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 8, pp. 2041–2056, 2013.

[6] E. J. Domingo, J. T. Nino, A. L. Lemos, M. L. Lemos, R. C. Palacios, and J. M. G. Berbís, "CLOUDIO: a cloud computing-oriented multi-tenant architecture for business information systems," in *2010 IEEE 3rd International Conference on Cloud Computing*, Miami, FL, USA, 2010, pp. 532–533.

[7] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," in *10th International Conference on Autonomic Computing (ICAC'13)*, 2013, pp. 23–27.

[8] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *J. Grid Comput.*, vol. 12, no. 4, pp. 559–592, 2014.

[9] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, J. J. P. C. Rodrigues, P. Chatzimisios, and J. M. Batalla, "A fluctuation-based modelling approach to quantification of the technical debt on mobile cloud-based service level," in *2015 IEEE Global Communications Conference (GLOBECOM 2015), Fourth IEEE International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA 2015)*, San Diego, California, USA, Dec. 2015, pp. 1–6, doi: 10.1109/GLOCOMW.2015.7413999.

[10] W. Cunningham, "The WyCash portfolio management system," *ACM SIGPLAN OOPS Messenger*, vol. 4, no. 2, pp. 29–30, 1992.

[11] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.

[12] G. Skourletopoulos *et al.*, "Game Theoretic Approaches in Mobile Cloud Computing Systems for Big Data Applications: A Systematic Literature Review," in *Mobile Big Data: A Roadmap from Models to Technologies*, 1st ed., vol. 10, Cham, Switzerland: Springer International Publishing AG, 2017, pp. 41–62.

[13] G. Skourletopoulos *et al.*, "Big data and cloud computing: a survey of the state-of-the-art and research challenges," in *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, 1st ed., vol. 22, Cham, Switzerland: Springer International Publishing AG, 2016, pp. 23–41.

[14] G. Skourletopoulos *et al.*, "Towards mobile cloud computing in 5G mobile networks: applications, big data services and future opportunities," in *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, 1st ed., vol. 22, Cham, Switzerland: Springer International Publishing AG, 2016, pp. 43–62.

[15] G. Skourletopoulos, R. Bahsoon, C. X. Mavromoustakis, G. Mastorakis, and E. Pallis, "Predicting and quantifying the technical debt in cloud software engineering," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2014)*, Athens, Greece, Dec. 2014, pp. 36–40.

[16] G. Skourletopoulos, R. Bahsoon, C. X. Mavromoustakis, and G. Mastorakis, "The technical debt in cloud software engineering: a prediction-based and quantification approach," in *Resource Management of Mobile Cloud Computing Networks and Environments*, 1st ed., Hershey, Pennsylvania, USA: IGI Global, 2015, pp. 24–42.

[17] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, and J. N. Sahalos, "An evaluation of cloud-based mobile services with limited capacity: a linear approach," *Soft Comput. J.*, vol. 21, no. 16, pp. 4523–4530, Aug. 2017, doi: 10.1007/s00500-016-2083-4.

[18] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, P. Chatzimisios, and J. M. Batalla, "A novel methodology for capitalizing on cloud storage through a big data-as-a-service framework," in *2016 IEEE Global Communications Conference (GLOBECOM 2016), Fifth IEEE International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA 2016)*, Washington, DC, USA, Dec. 2016, pp. 1–6.

UNIVERSITY of NICOSIA

[19] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, E. Pallis, J. M. Batalla, and G. Kormentzas, "Quantifying and evaluating the technical debt on mobile cloud-based service level," in *2016 IEEE International Conference on Communications (ICC 2016), Communications QoS, Reliability and Modelling Symposium (CQRM)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–7, doi: 10.1109/ICC.2016.7510995.

[20] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, E. Pallis, P. Chatzimisios, and J. M. Batalla, "Towards the evaluation of a big data-as-a-service model: A decision theoretic approach," in *35th IEEE International Conference on Computer Communications (INFOCOM 2016), First IEEE International Workshop on Big Data Sciences, Technologies, and Applications (BDSTA 2016)*, San Francisco, California, USA, Apr. 2016, pp. 877–883.

[21] G. Skourletopoulos *et al.*, "Elasticity Debt Analytics Exploitation for Green Mobile Cloud Computing: An Equilibrium Model," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 1, pp. 122–131, Mar. 2019, doi: 10.1109/TGCN.2018.2890034.

[22] G. Skourletopoulos *et al.*, "Elasticity Debt Analytics Exploitation for Green Mobile Cloud Computing: An Equilibrium Model," in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, Missouri, USA, May 2018, pp. 1–6, doi: 10.1109/ICC.2018.8422956.

[23] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, J. N. Sahalos, J. M. Batalla, and C. Dobre, "A Game Theoretic Formulation of the Technical Debt Management Problem in Cloud Systems," in *Proceedings of the 2017 14th IEEE International Conference on Telecommunications (ConTEL 2017), 4th International Workshop (Special Session) on Enhanced Living Environments (ELEMENT 2017)*, Zagreb, Croatia, Jun. 2017, pp. 7–12.

[24] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, J. N. Sahalos, J. M. Batalla, and C. Dobre, "Cost-Benefit Analysis Game for Efficient Storage Allocation in Cloud-Centric Internet of Things Systems: A Game Theoretic Perspective," in *Proceedings of the 15th IFIP/IEEE International Symposium on Integrated Network Management (IFIP/IEEE IM 2017), 2017 First International Workshop on Protocols, Applications and Platforms for Enhanced Living Environments (PAPELE 2017)*, Lisbon, Portugal, May 2017, pp. 1149–1154.

[25] G. Skourletopoulos, "Cost-Benefit Analysis Game for Efficient Storage Allocation in Cloud-Centric Internet of Things Systems: A Game Theoretic Perspective," Brussels,

Belgium, Short-Term Scientific Mission (STSM) Scientific Report COST-STSM-IC1303-37094 (ECOST-STSM-IC1303-270317-084686), Apr. 2017.

[26] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, WA, Australia, 2010, pp. 27–33.

[27] M. Fokaefs, C. Barna, and M. Litoiu, "Economics-driven resource scalability on the cloud," in *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'16)*, Austin, Texas, USA, May 2016, pp. 129–139.

[28] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," in *2009 IEEE International Symposium on Parallel & Distributed Processing*, Rome, Italy, 2009, pp. 1–12.

[29] M. D. de Assunção, A. di Costanzo, and R. Buyya, "A cost-benefit analysis of using cloud computing to extend the capacity of clusters," *Clust. Comput.*, vol. 13, no. 3, pp. 335–347, 2010.

[30] X. Li, Y. Li, T. Liu, J. Qiu, and F. Wang, "The method and tool of cost analysis for cloud computing," in *2009 IEEE International Conference on Cloud Computing*, Bangalore, India, 2009, pp. 93–100.

[31] Y. Zhao, R. N. Calheiros, G. Gange, K. Ramamohanarao, and R. Buyya, "SLA-based resource scheduling for big data analytics as a service in cloud computing environments," in *2015 44th International Conference on Parallel Processing (ICPP)*, Beijing, China, 2015, pp. 510–519.

[32] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, "Multi-Tiered On-Demand Resource Scheduling for VM-Based Data Center," presented at the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai, China, 2009.

[33] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Trans. Mob. Comput.*, vol. 17, no. 12, pp. 2868–2881, 2018.

[34] F. Zulkernine, P. Martin, Y. Zou, M. Bauer, F. Gwadry-Sridhar, and A. Aboulnaga, "Towards cloud-based analytics-as-a-service (claaas) for big data analytics in the cloud," in *2013 IEEE International Congress on Big Data*, Santa Clara, California, USA, 2013, pp. 62–69.

[35] S. Shekhar and A. Gokhale, "Dynamic resource management across cloud-edge resources for performance-sensitive applications," in *2017 17th IEEE/ACM International*

Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 2017, pp. 707–710.

[36] J. Patel, V. Jindal, I.-L. Yen, F. Bastani, J. Xu, and P. Garraghan, "Workload estimation for improving resource management decisions in the cloud," in *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, Taichung, Taiwan, 2015, pp. 25–32.

[37] N. R. Herbst, S. Kounev, A. Weber, and H. Groenda, "BUNGEE: an elasticity benchmark for self-adaptive IaaS cloud environments," in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Florence, Italy, 2015, pp. 46–56.

[38] G. Galante and L. C. E. de Bona, "A survey on cloud computing elasticity," in *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, Chicago, IL, USA, 2012, pp. 263–270.

[39] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in cloud computing: a survey," *Ann. Telecommun.*, vol. 70, no. 7–8, pp. 289–309, 2015.

[40] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: state of the art and research challenges," *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 430–447, 2017.

[41] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong, "Principles of Elastic Processes," *IEEE Internet Comput.*, vol. 15, no. 5, pp. 66–71, 2011.

[42] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *International Conference on Algorithms and Architectures for Parallel Processing*, 2010, pp. 13–31.

[43] D. Besanko, D. Dranove, M. Shanley, and S. Schaefer, *Economics of strategy*. John Wiley & Sons, 2009.

[44] A. Bauer, N. Herbst, S. Spinner, A. Ali-Eldin, and S. Kounev, "Chameleon: A hybrid, proactive auto-scaling mechanism on a level-playing field," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 800–813, 2018.

[45] L. Baresi, S. Guinea, A. Leva, and G. Quattrocchi, "A discrete-time feedback controller for containerized cloud applications," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 217–228.

UNIVERSITY of NICOSIA

[46] J. Liu, Y. Zhang, Y. Zhou, D. Zhang, and H. Liu, "Aggressive resource provisioning for ensuring QoS in virtualized environments," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 119–131, 2014.

[47] G. Moltó, M. Caballer, E. Romero, and C. De Alfonso, "Elastic memory management of virtualized infrastructures for applications with dynamic memory requirements," *Procedia Comput. Sci.*, vol. 18, pp. 159–168, 2013.

[48] B. Liu, R. Buyya, and A. N. Toosi, "A fuzzy-based auto-scaler for web applications in cloud computing environments," in *International Conference on Service-Oriented Computing*, 2018, pp. 797–811.

[49] E. B. Lakew, C. Klein, F. Hernandez-Rodriguez, and E. Elmroth, "Towards faster response time models for vertical elasticity," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014, pp. 560–565.

[50] S. Khatua, M. M. Manna, and N. Mukherjee, "Prediction-based instant resource provisioning for cloud applications," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014, pp. 597–602.

[51] S. Farokhi, P. Jamshidi, E. B. Lakew, I. Brandic, and E. Elmroth, "A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach," *Future Gener. Comput. Syst.*, vol. 65, pp. 57–72, 2016.

[52] C. G. Ralha, A. H. Mendes, L. A. Laranjeira, A. P. Araújo, and A. C. Melo, "Multiagent system for dynamic resource provisioning in cloud computing platforms," *Future Gener. Comput. Syst.*, vol. 94, pp. 80–96, 2019.

[53] M. Mohamed, M. Amziani, D. Belaïd, S. Tata, and T. Melliti, "An autonomic approach to manage elasticity of business processes in the cloud," *Future Gener. Comput. Syst.*, vol. 50, pp. 49–61, 2015.

[54] L. R. Moore, K. Bean, and T. Ellahi, "Transforming reactive auto-scaling into proactive auto-scaling," in *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*, 2013, pp. 7–12.

[55] M. A. Netto, C. Cardonha, R. L. Cunha, and M. D. Assunçao, "Evaluating auto-scaling strategies for cloud computing environments," in *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, Paris, France, 2014, pp. 187–196.

[56] A. V. Papadopoulos, A. Ali-Eldin, K.-E. Årzén, J. Tordsson, and E. Elmroth, "PEAS: A performance evaluation framework for auto-scaling strategies in cloud applications,"

UNIVERSITY of NICOSIA

ACM Trans. Model. Perform. Eval. Comput. Syst. TOMPECS, vol. 1, no. 4, pp. 1–31, 2016.

[57] G. Moltó, M. Caballer, and C. De Alfonso, "Automatic memory-based vertical elasticity and oversubscription on cloud platforms," *Future Gener. Comput. Syst.*, vol. 56, pp. 1–10, 2016.

[58] A. Gambi, G. Toffetti, C. Pautasso, and M. Pezze, "Kriging controllers for cloud applications," *IEEE Internet Comput.*, vol. 17, no. 4, pp. 40–47, 2012.

[59] G. Galante and L. C. E. Bona, "Constructing elastic scientific applications using elasticity primitives," in *International Conference on Computational Science and Its Applications*, 2013, pp. 281–294.

[60] R. L. Cunha, M. D. Assunçao, C. Cardonha, and M. A. Netto, "Exploiting user patience for scaling resource capacity in cloud services," in *2014 IEEE 7th International Conference on Cloud Computing*, Anchorage, AK, USA, 2014, pp. 448–455.

[61] S. Farokhi, E. B. Lakew, C. Klein, I. Brandic, and E. Elmroth, "Coordinating CPU and memory elasticity controllers to meet service response time constraints," in *2015 International Conference on Cloud and Autonomic Computing*, Boston, MA, USA, 2015, pp. 69–80.

[62] S. Spinner *et al.*, "Proactive memory scaling of virtualized applications," in *2015 IEEE 8th International Conference on Cloud Computing*, New York, NY, USA, 2015, pp. 277–284.

[63] Z. Yin, H. Chen, J. Sun, and F. Hu, "EAERS: An Enhanced Version of Autonomic and Elastic Resource Scheduling Framework for Cloud Applications," presented at the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, CA, USA, 2017.

[64] K. Salah, K. Elbadawi, and R. Boutaba, "An Analytical Model for Estimating Cloud Resources of Elastic Services," *J. Netw. Syst. Manag.*, vol. 24, no. 2, pp. 285–308, 2016.

[65] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "An autonomic prediction suite for cloud resource provisioning," *J. Cloud Comput.*, vol. 6, no. 3, 2017.

[66] V. Persico, D. Grimaldi, A. Pescapè, A. Salvi, and S. Santini, "A Fuzzy Approach Based on Heterogeneous Metrics for Scaling Out Public Clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 8, pp. 2117–2130, Aug. 2017.

[67] A. Gandhi, P. Dube, A. Karve, A. Kochut, and L. Zhang, "Providing Performance Guarantees for Cloud-Deployed Applications," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 269–281, 2020.

[68] P. D. Kaur and I. Chana, "A resource elasticity framework for QoS-aware execution of cloud applications," *Future Gener. Comput. Syst.*, vol. 37, pp. 14–25, 2014.

[69] A. Al-Shishtawy and V. Vlassov, "ElastMan: elasticity manager for elastic key-value stores in the cloud," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, Aug. 2013, pp. 1–10.

[70] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Using genetic algorithms to find optimal solution in a search space for a cloud predictive cost-driven decision maker," *J. Cloud Comput.*, vol. 7, no. 20, 2018.

[71] J. A. Pascual, J. A. Lozano, and J. Miguel-Alonso, "Effects of Reducing VMs Management Times on Elastic Applications," *J. Grid Comput.*, vol. 16, pp. 513–530, 2018.

[72] M. Ficco, C. Esposito, F. Palmieri, and A. Castiglione, "A coral-reefs and game theory-based approach for optimizing elastic cloud resource allocation," *Future Gener. Comput. Syst.*, vol. 78, no. 1, pp. 343–352, 2018.

[73] R. Khorsand, M. Ghobaei-Arani, and M. Ramezanpour, "FAHP approach for autonomic resource provisioning of multitier applications in cloud computing environments," *Softw. Pract. Exp.*, vol. 48, no. 12, pp. 2147–2173, 2018.

[74] S. M. R. Nouri, H. Li, S. Venugopal, W. Guo, M. He, and W. Tian, "Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications," *Future Gener. Comput. Syst.*, vol. 94, pp. 765–780, May 2019.

[75] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *2010 11th IEEE/ACM International Conference on Grid Computing*, Brussels, Belgium, 2010, pp. 41–48.

[76] M. S. Aslanpour, M. Ghobaei-Arani, and A. N. Toosi, "Auto-scaling web applications in clouds: A cost-aware approach," *J. Netw. Comput. Appl.*, vol. 95, pp. 26–41, 2017.

[77] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Autonomic Vertical Elasticity of Docker Containers with ELASTICDOCKER," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, Honolulu, CA, USA, 2017, pp. 472–479.

[78] C. Tao and R. Bahsoon, "Self-Adaptive Trade-off Decision Making for Autoscaling Cloud-Based Services," *IEEE Trans. Serv. Comput.*, vol. 10, no. 4, pp. 618–632, 2017.

[79] M. A. Rodriguez and R. Buyya, "Budget-Driven Scheduling of Scientific Workflows in IaaS Clouds with Fine-Grained Billing Periods," *ACM Trans. Auton. Adapt. Syst.*, vol. 12, no. 2, 2017.

UNIVERSITY of NICOSIA

[80] N. Roy, A. Dubey, and A. Gokhale, “Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting,” in *2011 IEEE 4th International Conference on Cloud Computing*, Washington, DC, USA, 2011, pp. 500–507.

[81] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, “A Cost-Aware Elasticity Provisioning System for the Cloud,” in *2011 31st International Conference on Distributed Computing Systems*, Minneapolis, MN, USA, 2011, pp. 559–570.

[82] S. Islam, K. Lee, A. Fekete, and A. Liu, “How a consumer can measure elasticity for cloud platforms,” in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, 2012, pp. 85–96.

[83] J. Jiang, J. Lu, G. Zhang, and G. Long, “Optimal Cloud Resource Auto-Scaling for Web Applications,” in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, Delft, Netherlands, 2013, pp. 58–65.

[84] G. Copil, D. Moldovan, H.-L. Truong, and S. Dustdar, “SYBL: An Extensible Language for Controlling Elasticity in Cloud Applications,” in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, Delft, Netherlands, 2013, pp. 112–119.

[85] R. da Rosa Righi, V. F. Rodrigues, C. A. da Costa, G. Galante, L. C. E. Bona, and T. Ferreto, “AutoElastic: Automatic Resource Elasticity for High Performance Applications in the Cloud,” *IEEE Trans. Cloud Comput.*, vol. 4, no. 1, pp. 6–19, 2016.

[86] A. G. García, I. Blanquer Espert, and V. Hernández García, “SLA-driven dynamic cloud resource management,” *Future Gener. Comput. Syst.*, vol. 31, pp. 1–11, 2014.

[87] R. G. Martinez, A. Lopes, and L. Rodrigues, “Automated generation of policies to support elastic scaling in cloud environments,” in *Proceedings of the Symposium on Applied Computing*, 2017, pp. 450–455.

[88] P. Jamshidi, C. Pahl, and N. C. Mendonça, “Managing uncertainty in autonomic cloud elasticity controllers,” *IEEE Cloud Comput.*, vol. 3, no. 3, pp. 50–60, 2016.

[89] R.-H. Hwang, C.-N. Lee, Y.-R. Chen, and D.-J. Zhang-Jian, “Cost Optimization of Elasticity Cloud Resource Subscription Policy,” *IEEE Trans. Serv. Comput.*, vol. 7, no. 4, pp. 561–574, 2014.

[90] A. H. Mahmud, Y. He, and S. Ren, “BATS: Budget-Constrained Autoscaling for Cloud Performance Optimization,” in *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Atlanta, GA, USA, 2015, pp. 232–241.

[91] P. Jamshidi, A. Ahmad, and C. Pahl, "Autonomic resource provisioning for cloud-based software," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2014, pp. 95–104.

[92] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," *Future Gener. Comput. Syst.*, vol. 32, pp. 82–98, 2014.

[93] E. De Coninck, T. Verbelen, B. Vankeirsbilck, S. Bohez, P. Simoens, and B. Dhoedt, "Dynamic auto-scaling and scheduling of deadline constrained service workloads on IaaS clouds," *J. Syst. Softw.*, vol. 118, pp. 101–114, 2016.

[94] E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," *Concurr. Comput. Pract. Exp.*, vol. 25, no. 12, pp. 1656–1674, 2013.

[95] R. S. Shariffdeen, D. T. S. P. Munasinghe, H. S. Bhathiya, U. K. J. U. Bandara, and H. M. N. Dilum Bandara, "Workload and Resource Aware Proactive Auto-scaler for PaaS Cloud," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, San Francisco, CA, USA, 2016, pp. 11–18.

[96] J. Shi, J. Luo, F. Dong, J. Zhang, and J. Zhang, "Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints," *Clust. Comput.*, vol. 19, no. 1, pp. 167–182, 2016.

[97] T. Knauth and C. Fetzer, "Scaling Non-elastic Applications Using Virtual Machines," in *2011 IEEE 4th International Conference on Cloud Computing*, Washington, DC, USA, 2011, pp. 468–475.

[98] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011, pp. 1–14.

[99] R. da Rosa Righi, C. A. da Costa, V. F. Rodrigues, and G. Rostirolla, "Joint-analysis of performance and energy consumption when enabling cloud elasticity for synchronous HPC applications," *Concurr. Comput. Pract. Exp.*, vol. 28, no. 5, pp. 1548–1571, 2016.

[100] C. De Alfonso, M. Caballer, A. Calatrava, G. Moltó, and I. Blanquer, "Multi-elastic Datacenters: Auto-scaled Virtual Clusters on Energy-Aware Physical Infrastructures," *J. Grid Comput.*, vol. 17, pp. 191–204, 2019.

[101] K. Guo *et al.*, "On the performance and power consumption analysis of elastic clouds," *Concurr. Comput. Pract. Exp.*, vol. 28, no. 17, pp. 4367–4384, 2016.

UNIVERSITY of NICOSIA

[102] D. Guyon, A.-C. Orgerie, and C. Morin, "Energy-Efficient User-Oriented Cloud Elasticity for Data-Driven Applications," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, Sydney, NSW, Australia, 2015, pp. 376–383.

[103] Y. Zhang, Y. Wang, and C. Hu, "CloudFreq: Elastic Energy-Efficient Bag-of-Tasks Scheduling in DVFS-Enabled Clouds," in *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, Melbourne, VIC, Australia, 2015, pp. 585–592.

[104] J. Yang, X. Xu, W. Tang, C. Hu, W. Dou, and J. Chen, "A Task Scheduling Method for Energy-Performance Trade-Off in Clouds," in *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Sydney, NSW, Australia, 2016, pp. 1029–1036.

[105] S. K. Tesfatsion, E. Wadbro, and J. Tordsson, "A combined frequency scaling and application elasticity approach for energy-efficient cloud computing," *Sustain. Comput. Inform. Syst.*, vol. 4, no. 4, pp. 205–214, 2014.

[106] V. F. Rodrigues *et al.*, "Towards Enabling Live Thresholding as Utility to Manage Elastic Master-Slave Applications in the Cloud," *J. Grid Comput.*, vol. 15, no. 4, pp. 535–556, 2017.

[107] I. S. Moreno and J. Xu, "Customer-aware resource overallocation to improve energy efficiency in realtime Cloud Computing data centers," presented at the 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), Irvine, CA, USA, 2011.

[108] M. Tighe and M. Bauer, "Topology and Application Aware Dynamic VM Management in the Cloud," *J. Grid Comput.*, vol. 15, no. 2, pp. 273–294, 2017.

[109] E. Casalicchio, L. Lundberg, and S. Shirinbab, "Energy-aware auto-scaling algorithms for Cassandra virtual data centers," *Clust. Comput.*, vol. 20, no. 3, pp. 2065–2082, 2017.

[110] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, "Elastic Load Balancing for Dynamic Virtual Machine Reconfiguration Based on Vertical and Horizontal Scaling," *IEEE Trans. Serv. Comput.*, vol. 12, no. 2, pp. 319–334, 2019.

[111] D. Trihinas, G. Pallis, and M. D. Dikaiakos, "Monitoring Elastically Adaptive Multi-Cloud Services," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 800–814, 2018.

UNIVERSITY of NICOSIA

[112] R. Simões and C. Kamienski, "Elasticity Management in Private and Hybrid Clouds," in *2014 IEEE 7th International Conference on Cloud Computing*, Anchorage, AK, USA, 2014, pp. 793–800.

[113] F. Paraiso, P. Merle, and L. Seinturier, "Managing elasticity across multiple cloud providers," in *Proceedings of 2013 International Workshop on Multi-Cloud Applications and Federated Clouds*, 2013, pp. 53–60.

[114] K. Konstanteli, T. Cucinotta, K. Psychas, and T. A. Varvarigou, "Elastic Admission Control for Federated Cloud Services," *IEEE Trans. Cloud Comput.*, vol. 2, no. 3, pp. 348–361, 2014.

[115] Y.-H. Lee, K.-C. Huang, M.-R. Shie, and K.-C. Lai, "Distributed resource allocation in federated clouds," *J. Supercomput.*, vol. 73, no. 7, pp. 3196–3211, 2017.

[116] R. Landa, M. Charalambides, R. G. Clegg, D. Griffin, and M. Rio, "Self-Tuning Service Provisioning for Decentralized Cloud Applications," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 2, pp. 197–211, 2016.

[117] P. Hoenisch, C. Hochreiner, D. Schuller, S. Schulte, J. Mendling, and S. Dustdar, "Cost-Efficient Scheduling of Elastic Processes in Hybrid Clouds," in *2015 IEEE 8th International Conference on Cloud Computing*, New York, NY, USA, 2015, pp. 17–24.

[118] R. Moreno-Vozmediano, R. S. Montero, E. Huedo, and I. M. Llorente, "Orchestrating the Deployment of High Availability Services on Multi-zone and Multi-cloud Scenarios," *J. Grid Comput.*, vol. 16, no. 1, pp. 39–53, 2018.

[119] J. Choi, Y. Ahn, S. Kim, Y. Kim, and J. Choi, "VM auto-scaling methods for high throughput computing on hybrid infrastructure," *Clust. Comput.*, vol. 18, no. 3, pp. 1063–1073, 2015.

[120] N. Grozev and R. Buyya, "Multi-Cloud Provisioning and Load Distribution for Three-Tier Applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 9, no. 3, 2014.

[121] G. Galante, L. C. E. De Bona, A. R. Mury, B. Schulze, and R. da Rosa Righi, "An Analysis of Public Clouds Elasticity in the Execution of Scientific Applications: a Survey," *J. Grid Comput.*, vol. 14, pp. 193–216, 2016.

[122] C. Mera-Gómez, F. Ramírez, R. Bahsoon, and R. Buyya, "A debt-aware learning approach for resource adaptations in cloud elasticity management," in *International Conference on Service-Oriented Computing*, 2017, pp. 367–382.

[123] C. Mera-Gómez, F. Ramirez, R. Bahsoon, and R. Buyya, "A multi-agent elasticity management based on multi-tenant debt exchanges," in *2018 IEEE 12th International*

Conference on Self-Adaptive and Self-Organizing Systems (SASO), Trento, Italy, 2018, pp. 30–39.

[124] C. Mera-Gómez, R. Bahsoon, and R. Buyya, "Elasticity debt: a debt-aware approach to reason about elasticity decisions in the cloud," in *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, Shanghai, China, Dec. 2016, pp. 79–88.

[125] E. Alzaghoul and R. Bahsoon, "Economics-driven approach for managing technical debt in cloud-based architectures," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, Dresden, Germany, 2013, pp. 239–242.

[126] E. Alzaghoul and R. Bahsoon, "CloudMTD: Using real options to manage technical debt in cloud-based service selection," in *2013 4th International Workshop on Managing Technical Debt (MTD)*, San Francisco, CA, USA, 2013, pp. 55–62.

[127] A. Pandey, G. A. Moreno, J. Cámara, and D. Garlan, "Hybrid planning for decision making in self-adaptive systems," in *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, Augsburg, Germany, 2016, pp. 130–139.

[128] Y. Kouki, M. S. Hasan, and T. Ledoux, "Delta Scaling: How Resources Scalability/Termination Can Be Taken Place Economically?," in *2015 IEEE World Congress on Services*, New York, NY, USA, 2015, pp. 55–62.

[129] C. Qu, R. N. Calheiros, and R. Buyya, "A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances," *J. Netw. Comput. Appl.*, vol. 65, pp. 167–180, 2016.

[130] D. Perez-Palacin, R. Mirandola, and R. Calinescu, "Synthesis of Adaptation Plans for Cloud Infrastructure with Hybrid Cost Models," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, Verona, Italy, 2014, pp. 443–450.

[131] H. Jin, X. Wang, S. Wu, S. Di, and X. Shi, "Towards Optimized Fine-Grained Pricing of IaaS Cloud Platform," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 436–448, 2015.

[132] J. Chen, C. Wang, B. Bing Zhou, L. Sun, Y. Choon Lee, and A. Y. Zomaya, "Tradeoffs Between Profit and Customer Satisfaction for Service Provisioning in the Cloud," in *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, 2011, pp. 229–238.

[133] H. Arabnejad, P. Jamshidi, G. Estrada, N. El Ioini, and C. Pahl, "An Auto-Scaling Cloud Controller Using Fuzzy Q-Learning - Implementation in OpenStack," in *European Conference on Service-Oriented and Cloud Computing*, 2016, pp. 152–167.

UNIVERSITY of NICOSIA

[134] B. Suleiman, S. Sakr, R. Jeffery, and A. Liu, "On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure," *J. Internet Serv. Appl.*, vol. 3, pp. 173–193, 2012.

[135] M. Fokaefs, C. Barna, and M. Litoiu, "From DevOps to BizOps: Economic Sustainability for Scalable Cloud Applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 12, no. 4, 2017.

[136] R. Buyya, R. N. Calheiros, and X. Li, "Autonomic Cloud computing: Open challenges and architectural elements," presented at the 2012 Third International Conference on Emerging Applications of Information Technology, Kolkata, India, 2012.

[137] Z. Lu, J. Hein, M. Humphrey, M. Stan, J. Lach, and K. Skadron, "Control-theoretic dynamic frequency and voltage scaling for multimedia workloads," in *Proceedings of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, 2002, pp. 156–163.

[138] A. R. Hummaida, N. W. Paton, and R. Sakellariou, "Adaptation in cloud resource configuration: a survey," *J. Cloud Comput.*, vol. 5, no. 1, 2016.

[139] J. Sloman and D. Garratt, *Essentials of economics*. Pearson Education, 2010.

[140] A. Khosravi and R. Buyya, "Energy and carbon footprint-aware management of geo-distributed cloud data centers: A taxonomy, state of the art, and future directions," in *Sustainable Development: Concepts, Methodologies, Tools, and Applications*, IGI Global, 2018, pp. 1456–1475.

[141] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, Melbourne, VIC, Australia, May 2010, pp. 577–578.

[142] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID '10)*, Melbourne, VIC, Australia, 2010, pp. 826–831.

[143] A. Saltelli *et al.*, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[144] A. Filieri, G. Tamburrelli, and C. Ghezzi, "Supporting self-adaptation via quantitative verification and sensitivity analysis at run time," *IEEE Trans. Softw. Eng.*, vol. 42, no. 1, pp. 75–99, 2015.

[145] M. Szvetits and U. Zdun, "Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime," *Softw. Syst. Model.*, vol. 15, no. 1, pp. 31–69, 2016.

[146] R. Bahsoon, “Dynamic and adaptive management of technical debt: managing technical debt @runtime,” in *Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162)*, vol. 6, P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, Eds. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016, p. 118.

[147] A. Najjar, X. Serpaggi, C. Gravier, and O. Boissier, “Survey of Elasticity Management Solutions in Cloud Computing,” in *Continued Rise of the Cloud*, London: Springer, 2014, pp. 235–263.

[148] M. Ghobaei-Arani, A. Souri, T. Baker, and A. Hussien, “ControCity: An Autonomous Approach for Controlling Elasticity Using Buffer Management in Cloud Computing Environment,” *IEEE Access*, vol. 7, pp. 106912–106924, 2019.

[149] Y. Zhao, R. N. Calheiros, A. V. Vasilakos, J. Bailey, and R. O. Sinnott, “SLA-aware and deadline constrained profit optimization for cloud resource management in big data analytics-as-a-service platforms,” in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, Milan, Italy, 2019, pp. 146–155.

[150] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.

[151] J. MacGlashan, *The brown-umbc reinforcement learning and planning (burlap) library*. 2014.

[152] E. Alzaghoul and R. Bahsoon, “Evaluating technical debt in cloud-based architectures using real options,” in *2014 23rd Australian Software Engineering Conference*, Milsons Point, NSW, Australia, 2014, pp. 1–10.

[153] R. Bahsoon and W. Emmerich, “ArchOptions: A real options-based model for predicting the stability of software architectures,” Portland, Oregon, USA, 2003, pp. 38–43.

[154] R. Bahsoon and W. Emmerich, “Applying ArchOptions to value the payoff of refactoring,” in *Proceedings of the 6th International Workshop on Economic-Driven Software Engineering Research (EDSER-6)*, Edinburgh, UK, 2004, pp. 66–70.

[155] V. Nallur and R. Bahsoon, “A decentralized self-adaptation mechanism for service-based applications in the cloud,” *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 591–612, 2012.

[156] A. N. Toosi, R. K. Thulasiram, and R. Buyya, “Financial option market model for federated cloud environments,” in *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, Chicago, IL, USA, 2012, pp. 3–12.

[157] D. Betts, A. Homer, A. Jezierski, M. Narumoto, and H. Zhang, *Developing Multi-tenant Applications for the Cloud on Windows Azure*. Microsoft patterns & practices, 2013.

[158] R. Krebs, C. Momm, and S. Kounev, "Architectural Concerns in Multi-tenant SaaS Applications," in *Proceedings of the 2nd International Conference on Cloud Computing and Services Science (CLOSER-2012)*, 2012, pp. 426–431.

[159] N. Rameshan, Y. Liu, L. Navarro, and V. Vlassov, "Hubbub-Scale: Towards Reliable Elastic Scaling under Multi-tenancy," presented at the 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, Colombia, 2016.

[160] S. Kumar, R. Bahsoon, T. Chen, and R. Buyya, "Identifying and estimating technical debt for service composition in SaaS cloud," in *2019 IEEE International Conference on Web Services (ICWS)*, Milan, Italy, 2019, pp. 121–125.

[161] K. Liu, Y. Chen, and X. Zhang, "An evaluation of ARFIMA (autoregressive fractional integral moving average) programs," *Axioms*, vol. 6, no. 2, p. 16, 2017.

[162] S. Kumar, R. Bahsoon, T. Chen, K. Li, and R. Buyya, "Multi-tenant cloud service composition using evolutionary optimization," in *24th IEEE International Conference on Parallel and Distributed Systems*, Singapore, Singapore, 2018, pp. 972–979.

[163] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: toward an open-source solution for cloud computing," *Int. J. Comput. Appl.*, vol. 55, no. 3, pp. 38–42, 2012.

[164] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, "NetLord: a scalable multi-tenant network architecture for virtualized datacenters," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 62–73, 2011.

[165] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, 2015.

[166] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 1, pp. 236–262, 2015.

[167] T. Miyamoto, M. Hayashi, and H. Tanaka, "Customizing network functions for high performance cloud computing," in *2009 Eighth IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, USA, 2009, pp. 130–133.

[168] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 2, pp. 343–356, 2017.

[169] J. Son and R. Buyya, "Latency-aware Virtualized Network Function provisioning for distributed edge clouds," *J. Syst. Softw.*, vol. 152, pp. 24–31, 2019.

[170] J. Soares, M. Dias, J. Carapinha, B. Parreira, and S. Sargento, "Cloud4NFV: A platform for Virtual Network Functions," in *2014 IEEE 3rd International Conference on Cloud Networking*, Luxembourg, Luxembourg, 2014, pp. 288–293.

[171] A. Das, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and C. Yu, "Transparent and flexible network management for big data processing in the cloud," presented at the 5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'13), San Jose, CA, USA, Jun. 2013.

[172] P. Costa, M. Migliavacca, P. Pietzuch, and A. L. Wolf, "NaaS: Network-as-a-Service in the Cloud," presented at the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE'12), San Jose, CA, USA, Apr. 2012.

[173] G. Monteleone and P. Paglierani, "Session Border Controller Virtualization Towards 'Service-Defined' Networks Based on NFV and SDN," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, Italy, 2013, pp. 1–7.

[174] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: a cloud networking platform for enterprise applications," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011, pp. 1–13.

[175] G. Mastorakis, E. Markakis, E. Pallis, C. X. Mavromoustakis, and G. Skourletopoulos, "Virtual network functions exploitation through a prototype resource management framework," in *2014 IEEE 6th International Conference on Telecommunications and Multimedia (TEMU)*, Heraklion, Crete, Greece, Jul. 2014, pp. 24–28.

[176] G. Laatikainen, A. Ojala, and O. Mazhelis, "Cloud services pricing models," in *International Conference of Software Business*, 2013, pp. 117–129.

[177] X. He, P. Shenoy, R. Sitaraman, and D. Irwin, "Cutting the cost of hosting online services using cloud spot markets," in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC'15)*, 2015, pp. 207–218.

[178] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *2012 Proceedings IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 460–468.

[179] J. Madhavan *et al.*, "Web-scale data integration: You can only afford to pay as you go," *CIDR*, 2007.

[180] A. J. Macartney and G. S. Blair, “Flexible trading in distributed multimedia systems,” *Comput. Netw. ISDN Syst.*, vol. 25, no. 2, pp. 145–157, 1992.

[181] D. Nurmi *et al.*, “Eucalyptus: A technical report on an elastic utility computing architecture linking your programs to useful systems,” presented at the UCSB Technical Report, 2008.

[182] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[183] D. Niu, Z. Liu, B. Li, and S. Zhao, “Demand forecast and performance prediction in peer-assisted on-demand streaming systems,” in *2011 Proceedings IEEE INFOCOM*, Shanghai, China, 2011, pp. 421–425.

[184] W. Wu and J. C. Lui, “Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1492–1503, 2011.

[185] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, “Optimal content placement for a large-scale VoD system,” *IEEE-ACM Trans. Netw.*, vol. 24, no. 4, pp. 2114–2127, 2015.

[186] P. A. Dinda, “Design, implementation, and performance of an extensible toolkit for resource prediction in distributed systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 2, pp. 160–173, 2006.

[187] J. Skicewicz, J. A. Skicewicz, and P. A. Dinda, “Tsunami: A wavelet toolkit for distributed systems,” *Pa. State Univ.*, 2003.

[188] K. Suh *et al.*, “Push-to-peer video-on-demand system: Design and evaluation,” *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, pp. 1706–1716, 2007.

[189] B. M. Maggs and R. K. Sitaraman, “Algorithmic Nuggets in Content Delivery,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, 2015.

[190] Y. Kryftis, C. X. Mavromoustakis, J. M. Batalla, G. Mastorakis, E. Pallis, and G. Skourletopoulos, “Resource usage prediction for optimal and balanced provision of multimedia services,” in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2014)*, Athens, Greece, Dec. 2014, pp. 255–259.

[191] M. Melo, J. Carapinha, S. Sargento, U. Killat, and A. Timm-Giel, “A re-optimization approach for virtual network embedding,” in *International Conference on Mobile Networks and Management*, 2012, pp. 271–283.

[192] D. Posnakides, C. X. Mavromoustakis, G. Skourletopoulos, G. Mastorakis, E. Pallis, and J. M. Batalla, "Performance analysis of a rate-adaptive bandwidth allocation scheme in 5G mobile networks," in *20th IEEE Symposium on Computers and Communications (ISCC 2015), 2nd IEEE International Workshop on A 5G Wireless Odyssey: 2020*, Larnaca, Cyprus, Jul. 2015, pp. 955–961.

[193] M. Papadopoulos, C. X. Mavromoustakis, G. Skourletopoulos, G. Mastorakis, and E. Pallis, "Performance analysis of reactive routing protocols in mobile ad hoc networks," in *2014 IEEE 6th International Conference on Telecommunications and Multimedia (TEMU 2014)*, Heraklion, Crete, Greece, Jul. 2014, pp. 104–110.

[194] Federal Communications Commission, "Annual report and analysis of competitive market conditions with respect to mobile wireless, including commercial mobile services," *WT Docket*, no. 10–133, 2011.

[195] P. Makris, D. N. Skoutas, and C. Skianis, "A Survey on Context-Aware Mobile and Wireless Networking: On Networking and Computing Environments' Integration," *IEEE Commun. Surv. Tutor.*, vol. 15, no. 1, pp. 362–386, 2013.

[196] D. N. Skoutas, N. Nomikos, D. Vouyioukas, C. Skianis, and A. Antonopoulos, "Hybrid resource sharing for QoS preservation in virtual wireless networks," in *Cloud and Fog Computing in 5G Mobile Networks: Emerging Advances and Applications*, IET, 2017, pp. 303–324.

[197] S. Zhang, Z. Qian, J. Wu, and S. Lu, "An opportunistic resource sharing and topology-aware mapping framework for virtual networks," in *2012 IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 2408–2416.

[198] B. Parreira, M. Melo, J. Soares, J. Carapinha, R. Monteiro, and S. Sargento, "Network Virtualization-A Virtual Router Performance Evaluation," in *CRC 2012: 12a Conferência sobre Redes de Computadores*, 2012, pp. 23–29.

[199] D. M. Mattos, L. H. G. Ferraz, L. H. M. Costa, and O. C. M. Duarte, "Evaluating virtual router performance for a pluralist future internet," in *Proceedings of 3rd International Conference on Information and Communication Systems*, 2012, pp. 1–7.

[200] M. J. Schultz and P. Crowley, "Performance analysis of packet capture methods in a 10 gbps virtualized environment," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, Munich, Germany, 2012, pp. 1–8.

[201] J. M. Batalla, M. Kantor, C. X. Mavromoustakis, G. Skourletopoulos, and G. Mastorakis, "A novel methodology for efficient throughput evaluation in virtualized routers," in *2015 IEEE International Conference on Communications (ICC 2015),*

Communications Software, Services and Multimedia Applications Symposium (CSSMA), London, UK, Jun. 2015, pp. 6899–6905.

[202] N. Egi *et al.*, "A platform for high performance and flexible virtual routers on commodity hardware," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 127–128, 2010.

[203] M. S. Rathore, M. Hidell, and P. Sjödin, "Data plane optimization in open virtual routers," in *International Conference on Research in Networking*, 2011, pp. 379–392.

[204] D. M. F. Mattos, C. Fragni, M. D. D. Moreira, L. H. G. Ferraz, L. H. M. K. Costa, and O. Duarte, "Evaluating virtual router performance for the future internet," *Universidade Fed. Rio Jan. Grupo Teleinformatica E Autom.*, 2010.

[205] S. Bradner, "Benchmarking methodology for network interconnect devices," *RFC 2544*, 1999.

[206] R. Jain, A. Durresi, and G. Babic, "Throughput fairness index: An explanation," in *ATM Forum Contribution*, 1999, vol. 99, no. 45.

[207] S. Kumar, T. Chen, R. Bahsoon, and R. Buyya, "DATESSO: Self-Adapting Service Composition with Debt-Aware Two Levels Constraint Reasoning," presented at the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '20), Seoul, Republic of Korea, 2020.

[208] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, "The financial aspect of managing technical debt: A systematic literature review," *Inf. Softw. Technol.*, vol. 64, pp. 52–73, 2015.

[209] E. Allman, "Managing technical debt," *Commun. ACM*, vol. 55, no. 5, pp. 50–55, 2012.

[210] B. Curtis, J. Sappidi, and A. Szynkarski, "Estimating the principal of an application's technical debt," *IEEE Softw.*, vol. 29, no. 6, pp. 34–42, 2012.

[211] T. Klinger, P. Tarr, P. Wagstrom, and C. Williams, "An enterprise perspective on technical debt," in *Proceedings of the 2nd Workshop on Managing Technical Debt*, 2011, pp. 35–38.

[212] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *J. Syst. Softw.*, vol. 86, no. 6, pp. 1498–1516, 2013.

[213] E. da S. Maldonado and E. Shihab, "Detecting and quantifying different types of self-admitted technical debt," in *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, Bremen, Germany, 2015, pp. 9–15.

[214] R. L. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, "In search of a metric for managing architectural technical debt," in *2012 Joint Working IEEE/IFIP Conference on*

Software Architecture and European Conference on Software Architecture, Helsinki, Finland, 2012, pp. 91–100.

[215] P. Kruchten, R. Nord, and I. Ozkaya, *Managing Technical Debt: Reducing Friction in Software Development*. Addison-Wesley Professional, 2019.

[216] E. Lim, N. Taksande, and C. Seaman, "A balancing act: What software practitioners have to say about technical debt," *IEEE Softw.*, vol. 29, no. 6, pp. 22–27, 2012.

[217] W. N. Behutiye, P. Rodríguez, M. Oivo, and A. Tosun, "Analyzing the concept of technical debt in the context of agile software development: A systematic literature review," *Inf. Softw. Technol.*, vol. 82, pp. 139–158, 2017.

[218] C. Sterling, *Managing Software Debt: Building for Inevitable Change*. Boston: Addison-Wesley Professional, 2010.

[219] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *J. Syst. Softw.*, vol. 101, pp. 193–220, 2015.

[220] C. Seaman and Y. Guo, "Measuring and monitoring technical debt," in *Advances in Computers*, vol. 82, Elsevier, 2011, pp. 25–46.

[221] J. Yli-Huumo, A. Maglyas, and K. Smolander, "How do software development teams manage technical debt? – An empirical study," *J. Syst. Softw.*, vol. 120, pp. 195–218, 2016.

[222] A. Martini, J. Bosch, and M. Chaudron, "Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study," *Inf. Softw. Technol.*, vol. 67, pp. 237–253, 2015.

[223] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, and I. Gorton, "Measure it? manage it? ignore it? software practitioners and technical debt," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 50–60.

[224] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Softw.*, vol. 29, no. 6, pp. 18–21, 2012.

[225] Z. Li, P. Liang, and P. Avgeriou, "Architectural technical debt identification based on architecture decisions and change scenarios," presented at the 2015 12th Working IEEE/IFIP Conference on Software Architecture, Montreal, QC, Canada, May 2015.

[226] R. Marinescu, "Assessing technical debt by identifying design flaws in software systems," *IBM J. Res. Dev.*, vol. 56, no. 5, 2012.

[227] S. McConnell, "Managing technical debt," *Construx Softw. Build. Inc*, 2008.

[228] I. Griffith, D. Reimanis, C. Izurieta, Z. Codabux, A. Deo, and B. Williams, "The correspondence between software quality models and technical debt estimation

approaches," in *2014 Sixth International Workshop on Managing Technical Debt*, Victoria, BC, Canada, 2014, pp. 19–26.

[229] J. D. Morgenthaler, M. Gridnev, R. Sauciuc, and S. Bhansali, "Searching for build debt: Experiences managing technical debt at Google," in *2012 Third International Workshop on Managing Technical Debt (MTD)*, Zurich, Switzerland, 2012, pp. 1–6.

[230] A. Martini, J. Bosch, and M. Chaudron, "Architecture technical debt: Understanding causes and a qualitative model," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, Verona, Italy, 2014, pp. 85–92.

[231] G. Suryanarayana, G. Samarthyam, and T. Sharma, *Refactoring for software design smells: managing technical debt*. Morgan Kaufmann, 2014.

[232] Z. Li, P. Liang, and P. Avgeriou, "Architectural debt management in value-oriented architecting," in *Economics-Driven Software Architecture*, Elsevier, 2014, pp. 183–204.

[233] T. Mens and T. Tourwé, "A survey of software refactoring," *IEEE Trans. Softw. Eng.*, vol. 30, no. 2, pp. 126–139, 2004.

[234] M. Fowler, *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.

[235] F. Buschmann, "To pay or not to pay technical debt," *IEEE Softw.*, vol. 28, no. 6, pp. 29–31, 2011.

[236] J. Kerievsky, *Refactoring to patterns*. Pearson Deutschland GmbH, 2005.

[237] M. Alshayeb, "Empirical investigation of refactoring effect on software quality," *Inf. Softw. Technol.*, vol. 51, no. 9, pp. 1319–1326, 2009.

[238] R. Kazman *et al.*, "A case study in locating the architectural roots of technical debt," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Florence, Italy, 2015, vol. 2, pp. 179–188.

[239] R. Leitch and E. Stroulia, "Understanding the economics of refactoring," in *5th International Workshop on Economic-Driven Software Engineering Research (EDSER-5)*, 2003, p. 44.

[240] N. Brown *et al.*, "Managing technical debt in software-reliant systems," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, 2010, pp. 47–52.

[241] N. Zazworka, M. A. Shaw, F. Shull, and C. Seaman, "Investigating the impact of design debt on software quality," in *Proceedings of the 2nd Workshop on Managing Technical Debt*, 2011, pp. 17–23.

[242] T. Theodoropoulos, M. Hofberg, and D. Kern, “Technical debt from the stakeholder perspective,” in *Proceedings of the 2nd Workshop on Managing Technical Debt*, 2011, pp. 43–46.

[243] P. Conroy, “Technical debt: Where are the shareholders’ interests?,” *IEEE Softw.*, vol. 29, no. 6, pp. 88–88, 2012.

[244] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, “Technical debt in test automation,” in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, Montreal, QC, Canada, 2012, pp. 887–892.

[245] A. Nugroho, J. Visser, and T. Kuipers, “An empirical model of technical debt and interest,” in *Proceedings of the 2nd Workshop on Managing Technical Debt*, Waikiki, Honolulu, HI, USA, 2011, pp. 1–8.

[246] J.-L. Letouzey, “The SQALE method for evaluating technical debt,” in *2012 Third International Workshop on Managing Technical Debt (MTD)*, Zurich, Switzerland, 2012, pp. 31–36.

[247] G. Bavota and B. Russo, “A large-scale empirical study on self-admitted technical debt,” in *Proceedings of the 13th International Conference on Mining Software Repositories*, 2016, pp. 315–326.

[248] S. Wehaibi, E. Shihab, and L. Guerrouj, “Examining the impact of self-admitted technical debt on software quality,” in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Suita, Japan, 2016, vol. 1, pp. 179–188.

[249] The Apache Software Foundation, “Apache Hadoop.” https://hadoop.apache.org/ (accessed Sep. 18, 2019).

[250] The Apache Software Foundation, “Apache Cassandra.” https://cassandra.apache.org/ (accessed Sep. 18, 2019).

[251] The Apache Software Foundation, “Apache Spark™ - Unified Analytics Engine for Big Data.” https://spark.apache.org/ (accessed Sep. 18, 2019).

[252] The Apache Software Foundation, “Apache Tomcat®.” http://tomcat.apache.org/ (accessed Sep. 18, 2019).

[253] V. Lenarduzzi, N. Saarimäki, and D. Taibi, “The technical debt dataset,” in *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, 2019, pp. 2–11.

[254] SonarSource S.A., “SonarQube: Code Quality and Security.” https://www.sonarqube.org/ (accessed Sep. 18, 2019).

[255] Ptidej, “Ptidej: Design Smells.” http://www.ptidej.net/research/designsmells/ (accessed Sep. 18, 2019).

[256] N. Ramasubbu and C. F. Kemerer, “Managing technical debt in enterprise software packages,” *IEEE Trans. Softw. Eng.*, vol. 40, no. 8, pp. 758–772, 2014.

[257] N. Zazworka, C. Izurieta, S. Wong, Y. Cai, C. Seaman, and F. Shull, “Comparing four approaches for technical debt identification,” *Softw. Qual. J.*, vol. 22, no. 3, pp. 403–426, 2014.

[258] D. Eadline, *Hadoop 2 Quick-Start Guide: Learn the Essentials of Big Data Computing in the Apache Hadoop 2 Ecosystem*. Addison-Wesley Professional, 2015.

[259] D. Tsoukalas, D. Kehagias, M. Siavvas, and A. Chatzigeorgiou, “Technical debt forecasting: An empirical study on open-source repositories,” *J. Syst. Softw.*, vol. 170, Dec. 2020.

[260] B. Boehm, C. Abts, and S. Chulani, “Software development cost estimation approaches—A survey,” *Ann. Softw. Eng.*, vol. 10, no. 1–4, pp. 177–205, 2000.

[261] R. Jeffery, M. Ruhe, and I. Wieczorek, “A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data,” *Inf. Softw. Technol.*, vol. 42, no. 14, pp. 1009–1016, 2000.

[262] B. A. Kitchenham and N. R. Taylor, “Software project development cost estimation,” *J. Syst. Softw.*, vol. 5, no. 4, pp. 267–278, 1985.

[263] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, “Software development cost estimation using function points,” *IEEE Trans. Softw. Eng.*, vol. 20, no. 4, pp. 275–287, 1994.

[264] K. V. Kumar, V. Ravi, M. Carr, and N. R. Kiran, “Software development cost estimation using wavelet neural networks,” *J. Syst. Softw.*, vol. 81, no. 11, pp. 1853–1867, 2008.

[265] B. W. Boehm, “Software engineering economics,” *IEEE Trans. Softw. Eng.*, no. 1, pp. 4–21, 1984.

[266] N. Oza, J. Münch, J. Garbajosa, A. Yague, and E. G. Ortega, “Identifying potential risks and benefits of using cloud in distributed software development,” in *International Conference on Product Focused Software Process Improvement*, 2013, pp. 229–239.

[267] A. Potdar and E. Shihab, “An exploratory study on self-admitted technical debt,” in *2014 IEEE International Conference on Software Maintenance and Evolution*, Victoria, BC, Canada, 2014, pp. 91–100.

UNIVERSITY of NICOSIA

[268] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A survey of mobile cloud computing: architecture, applications, and approaches,” *Wirel. Commun. Mob. Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.

[269] G. Skourletopoulos, G. Mastorakis, C. X. Mavromoustakis, C. Dobre, and E. Pallis, *Mobile Big Data: A Roadmap from Models to Technologies*, 1st ed., vol. 10. Cham, Switzerland: Springer International Publishing AG, 2017.

[270] X. Sun, N. Ansari, and R. Wang, “Optimizing resource utilization of a data center,” *IEEE Commun. Surv. Tutor.*, vol. 18, no. 4, pp. 2822–2846, 2016.

[271] Google Cloud Platform, “Google Cloud Storage Pricing,” Dec. 2015. https://cloud.google.com/storage/pricing (accessed Nov. 27, 2015).

[272] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.

[273] F. Teng and F. Magoulès, “A new game theoretical resource allocation algorithm for cloud computing,” in *5th International Conference on Grid and Pervasive Computing (GPC 2010)*, Hualien, Taiwan, May 2010, pp. 321–330.

[274] X. Chen, L. Jiao, W. Li, and X. Fu, “Efficient multi-user computation offloading for mobile-edge cloud computing,” *IEEE ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, 2016.

[275] J. Nash, “Non-cooperative games,” *Ann. Math.*, vol. 54, no. 2, pp. 286–295, 1951.

[276] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[277] D. Ardagna, B. Panicucci, and M. Passacantando, “Generalized nash equilibria for the service provisioning problem in cloud systems,” *IEEE Trans. Serv. Comput.*, vol. 6, no. 4, pp. 429–442, 2013.

[278] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, “Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory,” *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2348–2362, 2015, doi: 10.1109/TC.2015.2389952.

[279] D. Niyato, P. Wang, E. Hossain, W. Saad, and Z. Han, “Game theoretic modeling of cooperation among service providers in mobile cloud computing environments,” in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, 2012, pp. 3128–3133.

[280] D. Niyato, A. V. Vasilakos, and Z. Kun, “Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach,” in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Newport Beach, CA, USA, May 2011, pp. 215–224.

[281] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, “A game theoretic resource allocation for overall energy minimization in mobile cloud computing system,” in *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, Redondo Beach, CA, USA, 2012, pp. 279–284.

[282] X. Chen, “Decentralized computation offloading game for mobile cloud computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, 2015.

[283] A. Antonopoulos and C. Verikoukis, “Multi-player game theoretic MAC strategies for energy efficient data dissemination,” *IEEE Trans. Wirel. Commun.*, vol. 13, no. 2, pp. 592–603, 2014.

[284] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, “A game-theoretic method of fair resource allocation for cloud computing services,” *J. Supercomput.*, vol. 54, no. 2, pp. 252–269, 2010.

[285] S. Misra, S. Das, M. Khatua, and M. S. Obaidat, “QoS-guaranteed bandwidth shifting and redistribution in mobile cloud environment,” *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 181–193, 2014.

[286] X. Sun and N. Ansari, “Green Cloudlet Network: A Distributed Green Mobile Cloud Network,” *IEEE Netw.*, vol. 31, no. 1, pp. 64–70, 2017.

[287] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, “Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing,” *IEEE Trans. Serv. Comput.*, 2016.

[288] J. Duan, D. Gao, D. Yang, C. H. Foh, and H.-H. Chen, “An energy-aware trust derivation scheme with game theoretic approach in wireless sensor networks for IoT applications,” *IEEE Internet Things J.*, vol. 1, no. 1, pp. 58–69, 2014.

[289] N. Kumar, N. Chilamkurti, and S. C. Misra, “Bayesian coalition game for the internet of things: an ambient intelligence-based evaluation,” *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 48–55, 2015.

[290] M. J. Osborne and A. Rubinstein, *A course in game theory*. Cambridge, Massachusetts: MIT press, 1994.

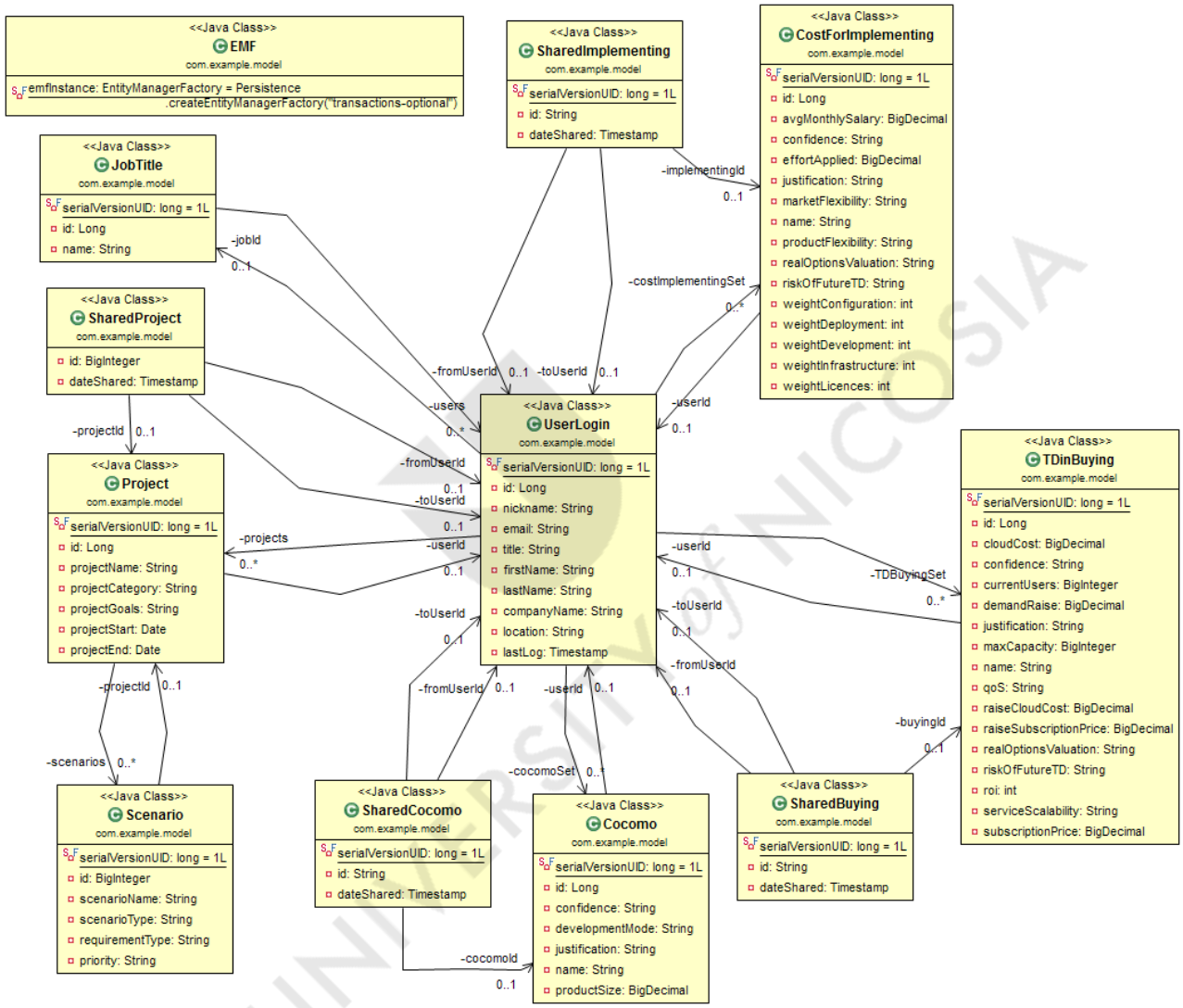
[291] S. P. T. Krishnan and J. L. Ugia Gonzalez, *Building your next big thing with Google Cloud Platform: A guide for developers and enterprise architects*. Springer, 2015.

[292] I. Sommerville, *Software Engineering*. Pearson Education Limited, 2013.

UNIVERSITY of NICOSIA

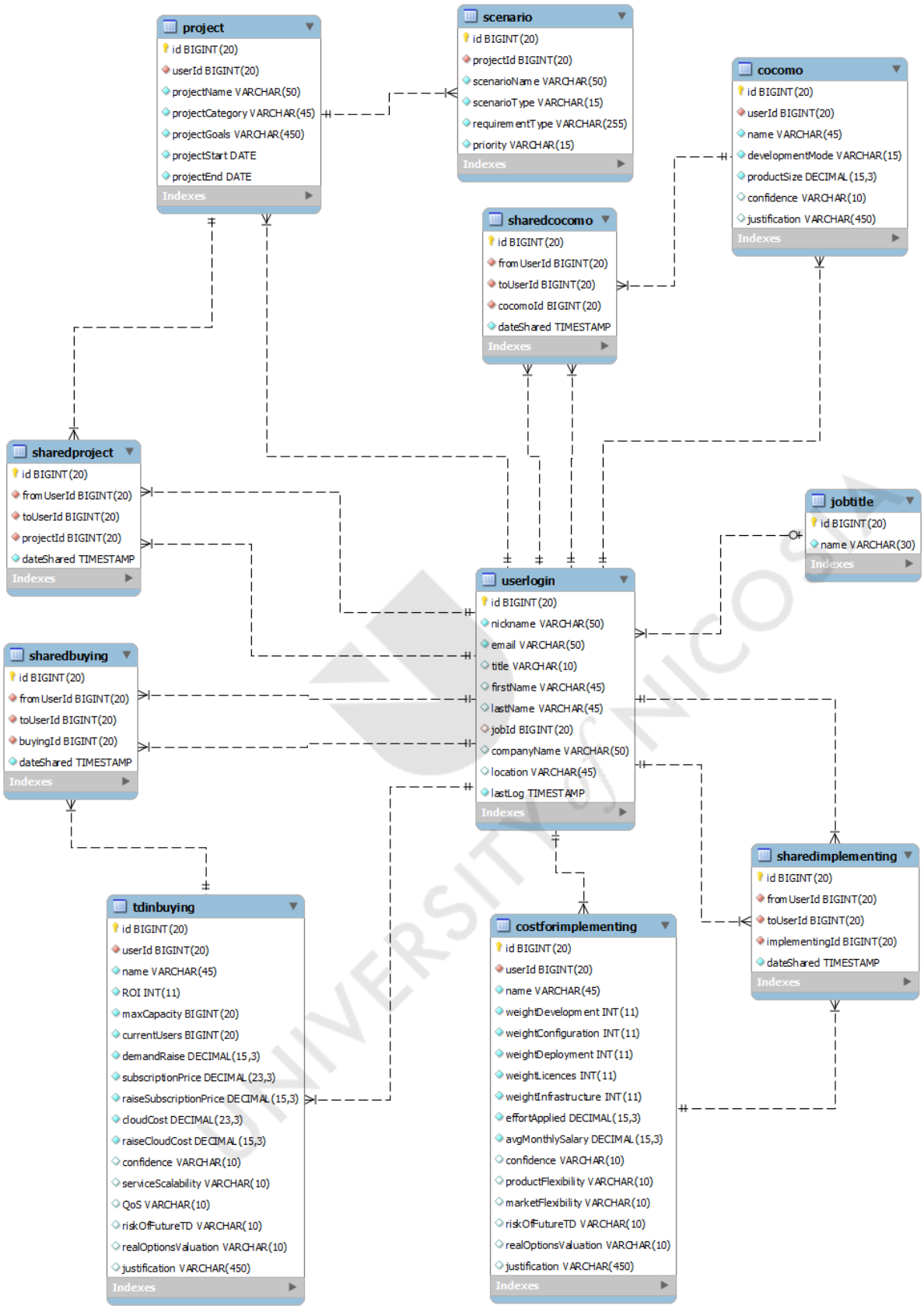
Appendices

Appendix A: UML Class Diagram and Database Design



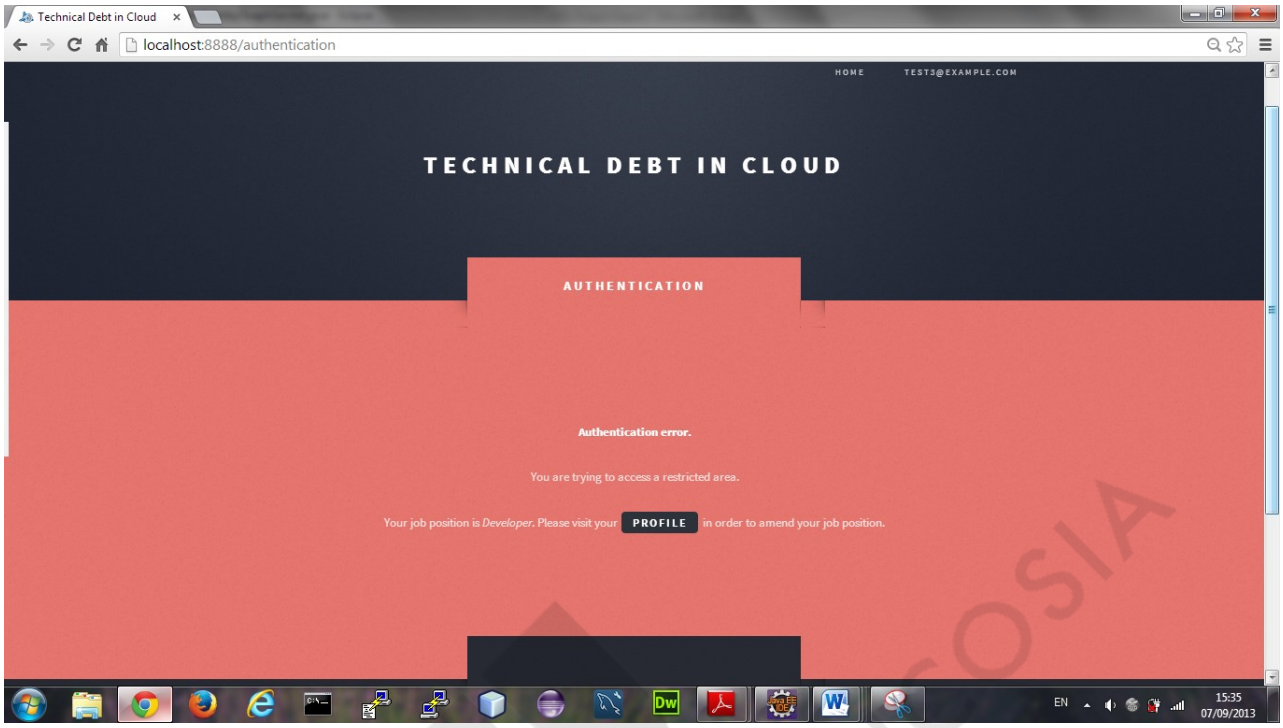
Appendix A.1. UML Class Diagram

UNIVERSITY OF NICOSIA

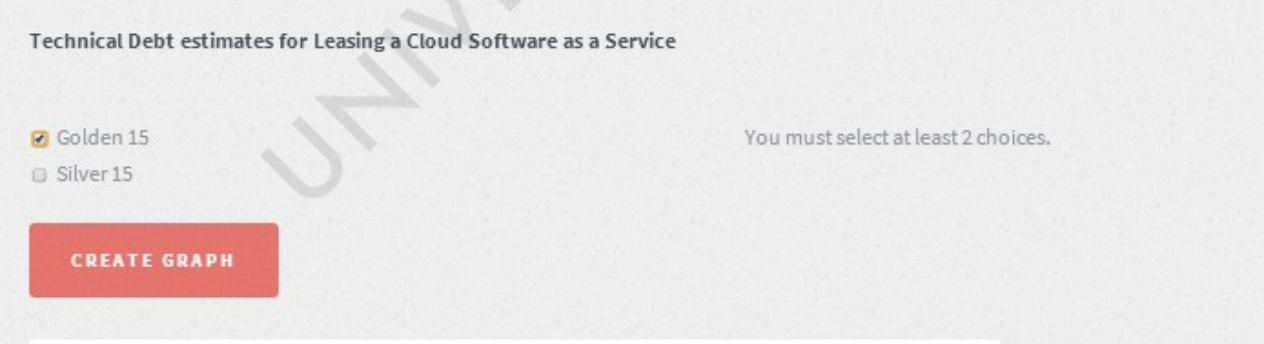


Appendix A.2. Entity-Relationship Diagram (ERD) using Crow's Foot Notation

Appendix B: Client- and Server-Side Validations

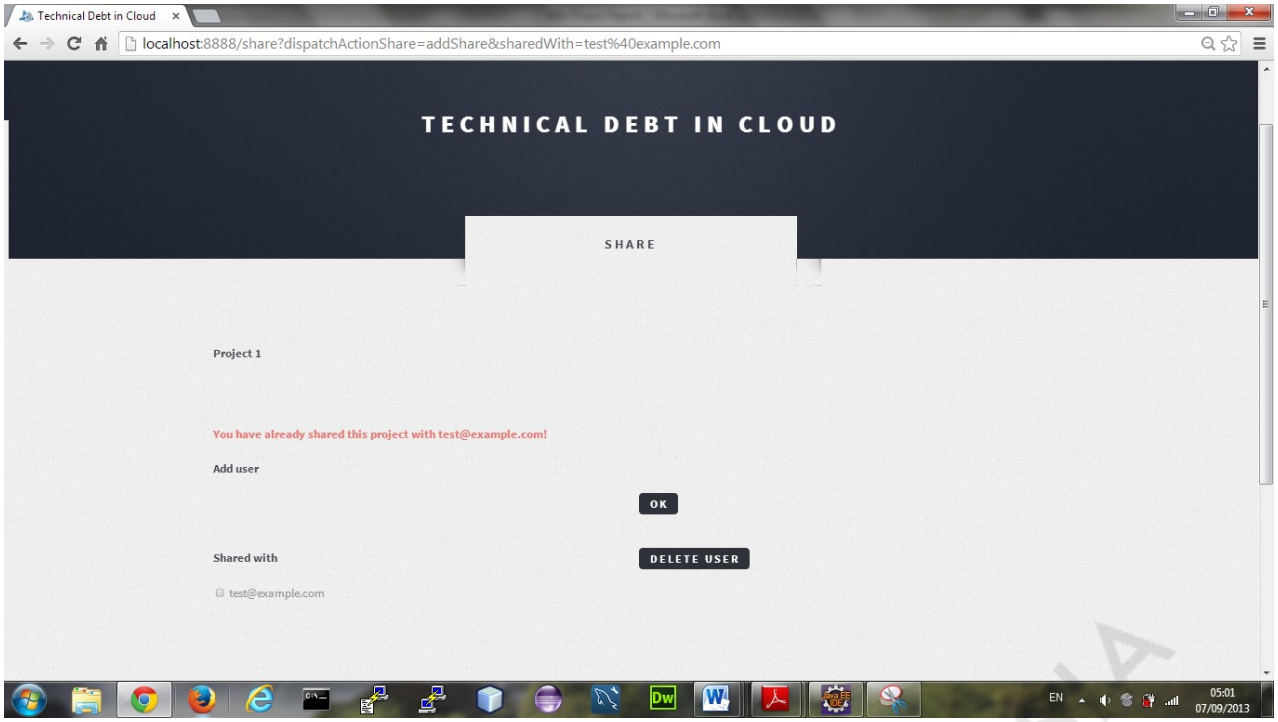


Appendix B.1. Authentication error for entering a restricted profile area



Appendix B.2. Validation for selecting at least two debt estimates to create graph / dashboard

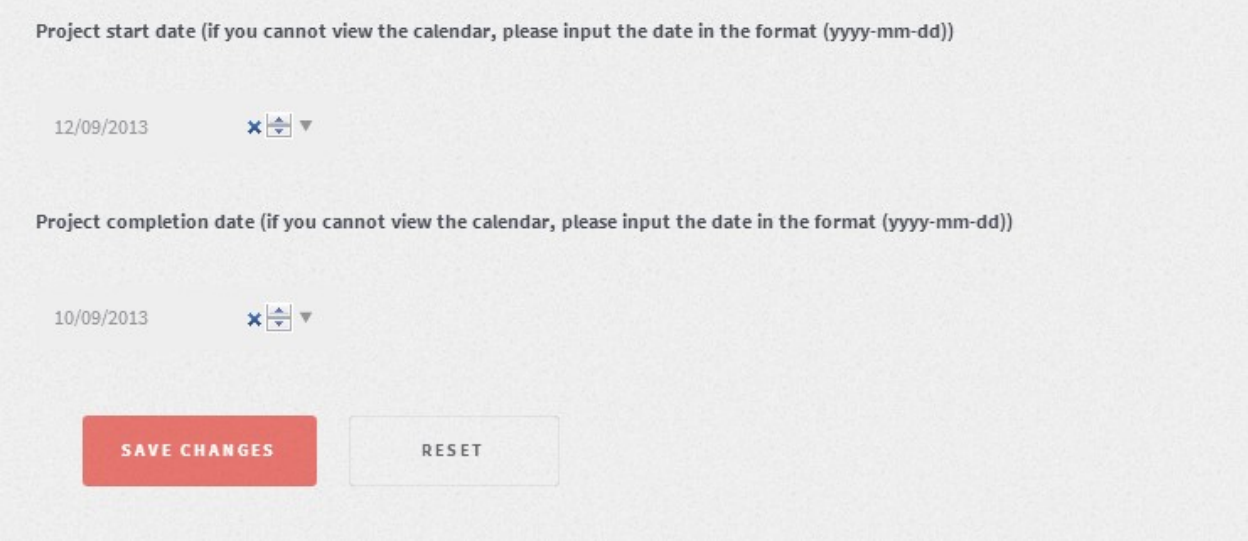
UNIVERSITY of NICOSIA



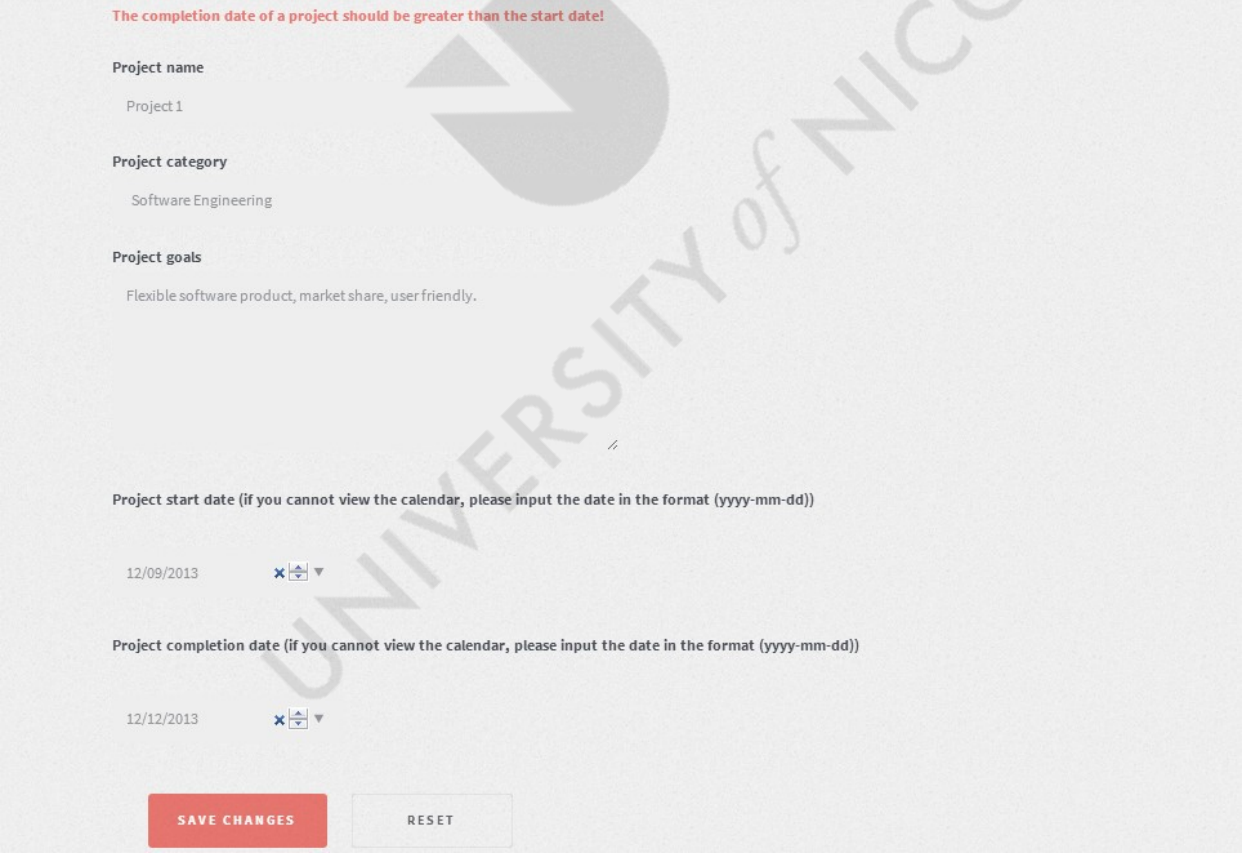
Appendix B.3. Validation for not being able to share same project / estimation twice with a user

Appendix B.4. Validation for existing debt estimation name

UNIVERSITY of NICOSIA

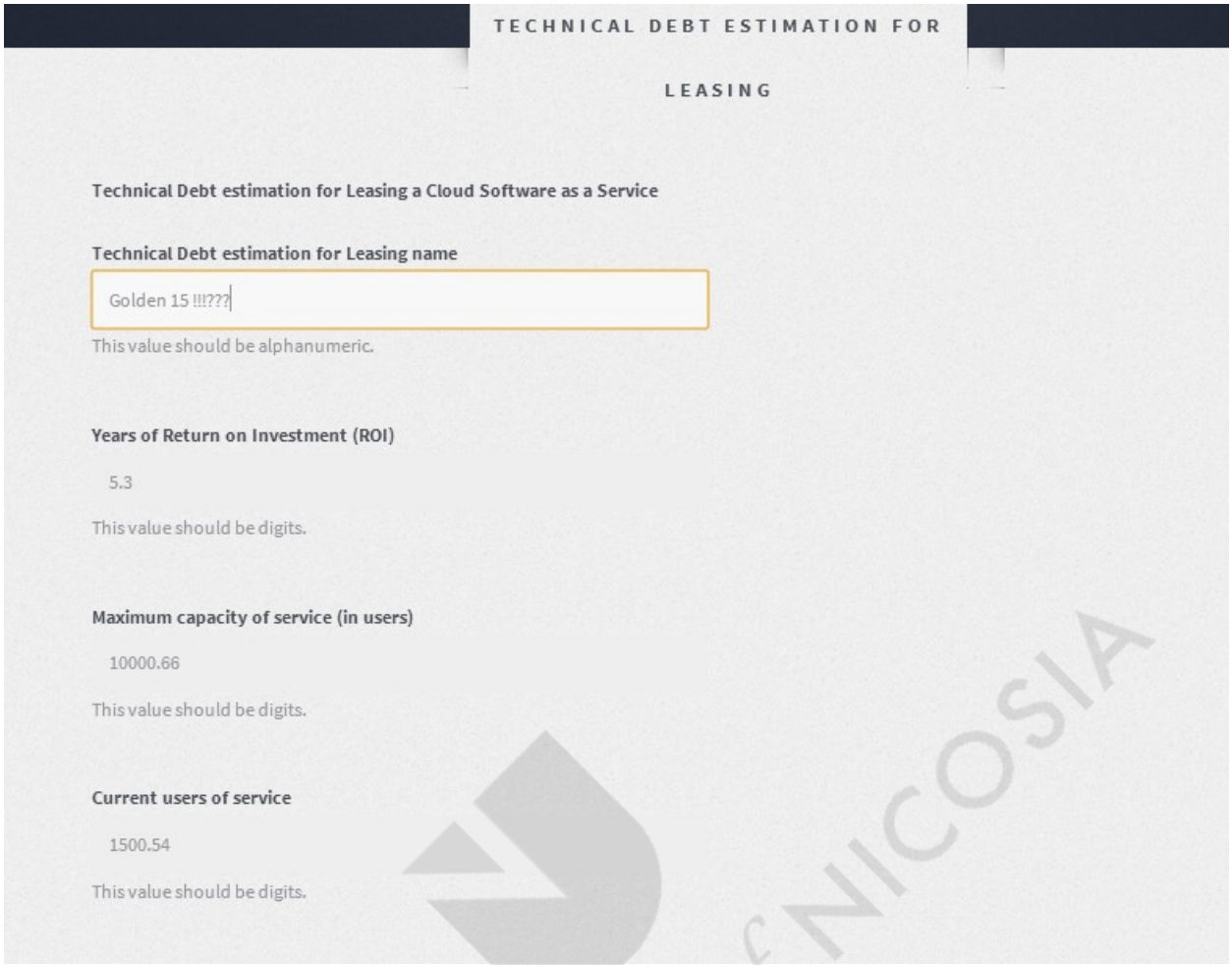


Appendix B.5. Inserting correct completion date in the project form (step 1)



Appendix B.6. Inserting correct completion date in the project form (step 2)

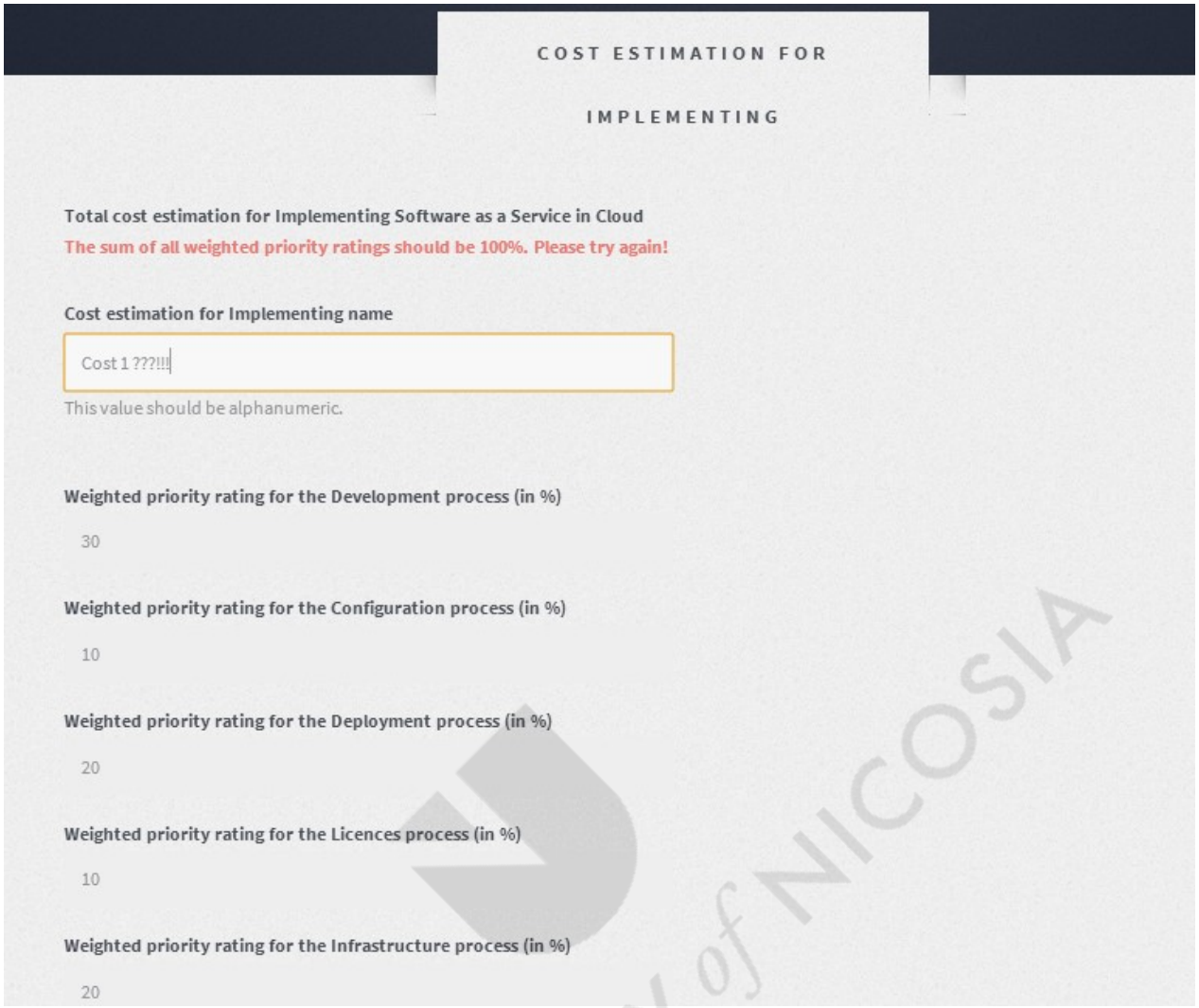
UNIVERSITY of NICOSIA



Appendix B.7. Inserting the correct data input in the debt estimation form (step 1)

Appendix B.8. Inserting the correct data input in the debt estimation form (step 2)

UNIVERSITY of NICOSIA



Appendix B.9. Validation for providing a sum of weighted priority ratings equal to 100%

UNIVERSITY of NICOSIA

Appendix C: Database Management System

Appendix C.1. Database Schema

Database Table	Field	Datatype	Key	Not Null	Constraints
cocomo	id	BIGINT(20)	Primary key	Yes	n/a
	userId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	name	VARCHAR(45)	n/a	Yes	n/a
	developmentMode	VARCHAR(15)	n/a	Yes	n/a
	productSize	DECIMAL(15,3)	n/a	Yes	n/a
	confidence	VARCHAR(10)	n/a	No	n/a
	justification	VARCHAR(450)	n/a	No	n/a
costforimplementing	id	BIGINT(20)	Primary key	Yes	n/a
	userId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	name	VARCHAR(45)	n/a	Yes	n/a
	weightDevelopment	INT(11)	n/a	Yes	n/a
	weightConfiguration	INT(11)	n/a	Yes	n/a
	weightDeployment	INT(11)	n/a	Yes	n/a
	weightLicences	INT(11)	n/a	Yes	n/a
	weightInfrastructure	INT(11)	n/a	Yes	n/a
	effortApplied	DECIMAL(15,3)	n/a	Yes	n/a
	avgMonthlySalary	DECIMAL(15,3)	n/a	Yes	n/a
	confidence	VARCHAR(10)	n/a	No	n/a
	productFlexibility	VARCHAR(10)	n/a	No	n/a
	marketFlexibility	VARCHAR(10)	n/a	No	n/a
	riskOfFutureTD	VARCHAR(10)	n/a	No	n/a
	realOptionsValuation	VARCHAR(10)	n/a	No	n/a
	justification	VARCHAR(450)	n/a	No	n/a

UNIVERSITY OF NICOSIA

Database Table	Field	Datatype	Key	Not Null	Constraints
jobtitle	id	BIGINT(20)	Primary key	Yes	n/a
	name	VARCHAR(30)	n/a	Yes	n/a
project	id	BIGINT(20)	Primary key	Yes	n/a
	userId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	projectName	VARCHAR(50)	n/a	Yes	n/a
	projectCategory	VARCHAR(45)	n/a	Yes	n/a
	projectGoals	VARCHAR(450)	n/a	Yes	n/a
	projectStart	DATE	n/a	Yes	n/a
	projectEnd	DATE	n/a	Yes	n/a
scenario	id	BIGINT(20)	Primary key	Yes	n/a
	projectId	BIGINT(20)	Foreign key (project)	Yes	On update/ delete cascade
	scenarioName	VARCHAR(50)	n/a	Yes	n/a
	scenarioType	VARCHAR(15)	n/a	Yes	n/a
	requirementType	VARCHAR(255)	n/a	Yes	n/a
	priority	VARCHAR(15)	n/a	Yes	n/a
sharedbuying	id	BIGINT(20)	Primary key	Yes	n/a
	fromUserId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	toUserId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	buyingId	BIGINT(20)	Foreign key (tdinbuying)	Yes	On update/ delete cascade
	dateShared	TIMESTAMP	n/a	Yes	n/a
sharedcocomo	id	BIGINT(20)	Primary key	Yes	n/a

UNIVERSITY OF NICOSIA

Database Table	Field	Datatype	Key	Not Null	Constraints
	fromUserId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	toUserId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	cocomoId	BIGINT(20)	Foreign key (cocomo)	Yes	On update/ delete cascade
	dateShared	TIMESTAMP	n/a	Yes	n/a
sharedimplementing	id	BIGINT(20)	Primary key	Yes	n/a
	fromUserId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	toUserId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	implementingId	BIGINT(20)	Foreign key (costforimplementing)	Yes	On update/ delete cascade
	dateShared	TIMESTAMP	n/a	Yes	n/a
sharedproject	id	BIGINT(20)	Primary key	Yes	n/a
	fromUserId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	toUserId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	projectId	BIGINT(20)	Foreign key (project)	Yes	On update/ delete cascade
	dateShared	TIMESTAMP	n/a	Yes	n/a
tdinbuying	id	BIGINT(20)	Primary key	Yes	n/a
	userId	BIGINT(20)	Foreign key (userlogin)	Yes	On update/ delete cascade
	name	VARCHAR(45)	n/a	Yes	n/a

UNIVERSITY OF NICOSIA

Database Table	Field	Datatype	Key	Not Null	Constraints
	ROI	INT(11)	n/a	Yes	n/a
	maxCapacity	BIGINT(20)	n/a	Yes	n/a
	currentUsers	BIGINT(20)	n/a	Yes	n/a
	demandRaise	DECIMAL(15,3)	n/a	Yes	n/a
	subscriptionPrice	DECIMAL(23,3)	n/a	Yes	n/a
	raiseSubscriptionPrice	DECIMAL(15,3)	n/a	Yes	n/a
	cloudCost	DECIMAL(23,3)	n/a	Yes	n/a
	raiseCloudCost	DECIMAL(15,3)	n/a	Yes	n/a
	confidence	VARCHAR(10)	n/a	No	n/a
	serviceScalability	VARCHAR(10)	n/a	No	n/a
	QoS	VARCHAR(10)	n/a	No	n/a
	riskOfFutureTD	VARCHAR(10)	n/a	No	n/a
	realOptionsValuation	VARCHAR(10)	n/a	No	n/a
	justification	VARCHAR(450)	n/a	No	n/a
`userlogin`	id	BIGINT(20)	Primary key	Yes	n/a
	nickname	VARCHAR(50)	n/a	Yes	n/a
	email	VARCHAR(50)	n/a	Yes	n/a
	title	VARCHAR(10)	n/a	No	n/a
	firstName	VARCHAR(45)	n/a	No	n/a
	lastName	VARCHAR(45)	n/a	No	n/a
	jobId	BIGINT(20)	Foreign key (jobtitle)	No	On update/ delete cascade
	companyName	VARCHAR(50)	n/a	No	n/a
	location	VARCHAR(45)	n/a	No	n/a
	lastLog	TIMESTAMP	n/a	Yes	n/a

UNIVERSITY OF NICOSIA

Appendix D: Testing and Product Evaluation

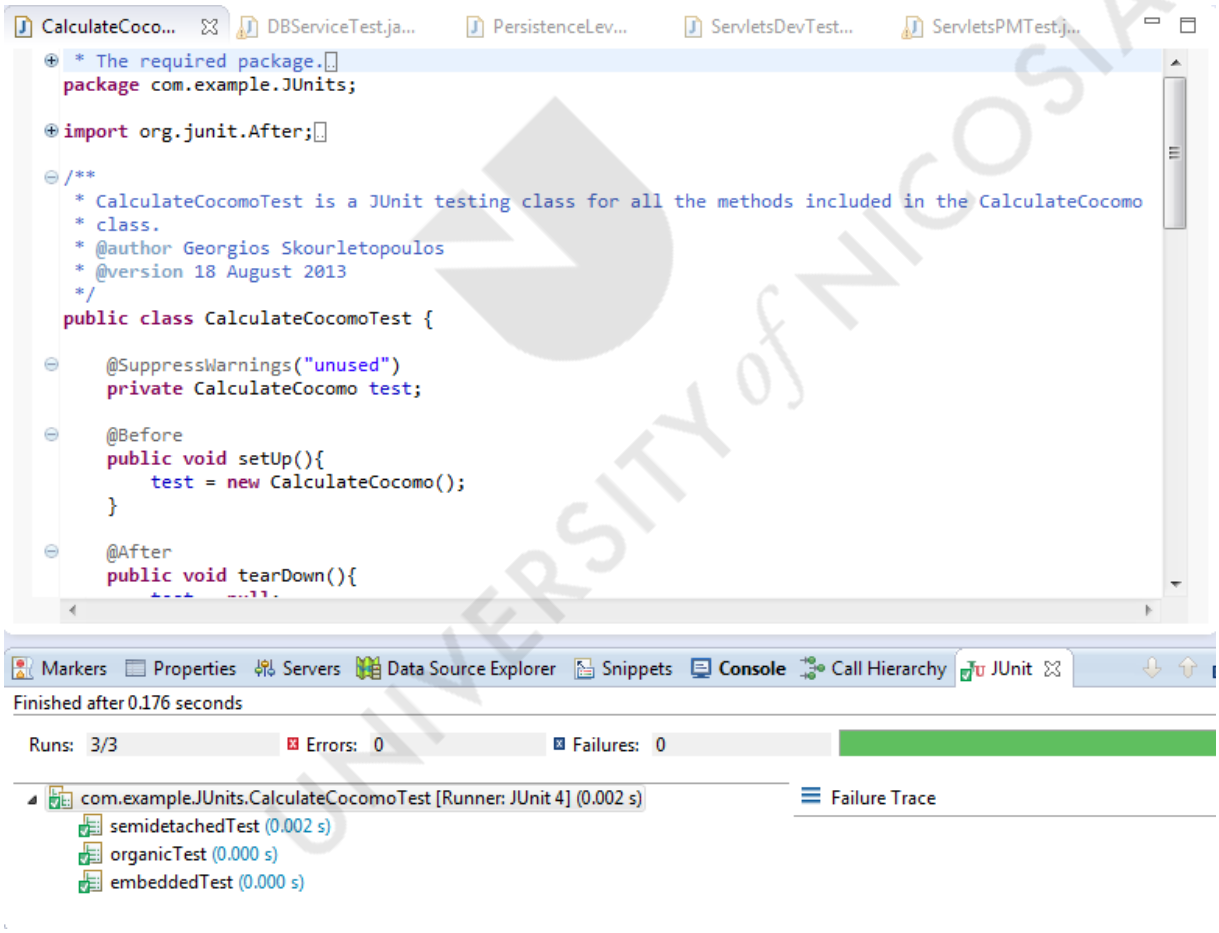
Appendix D.1. Integration testing for different use cases

Use Case	Procedure	Expected Output	Pass / Fail
Create project	Leave a field blank	Submission unsuccessful with the corresponding warning message under the field	Pass
	Provide a name that already exists in the database	Submission unsuccessful with the corresponding warning message	Pass
	Insert a wrong data input type in the project name field	A warning message should appear under the field	Pass
	Insert appropriate data input types and fill in all fields	Submission successful	Pass
	Delete a created project	Deletion successful	Pass
Add scenario	Leave a field blank or insert a wrong data input type	Submission unsuccessful with the corresponding warning message under the field(s)	Pass
	Provide a name that already exists in the database for a created project	Submission unsuccessful with the corresponding warning message	Pass
	Insert appropriate data input types and fill in all fields	Submission successful	Pass
	Delete a created scenario	Deletion successful	Pass
Create COCOMO estimation	Leave a field blank or insert a wrong data input type	Calculation unsuccessful with the corresponding notification message under the affected field	Pass
	Provide a name that already exists in the database	Submission unsuccessful with the corresponding warning message	Pass
	Insert appropriate data input types and fill in all fields	Calculation and submission successful	Pass
	Delete a created COCOMO estimation	Deletion successful	Pass

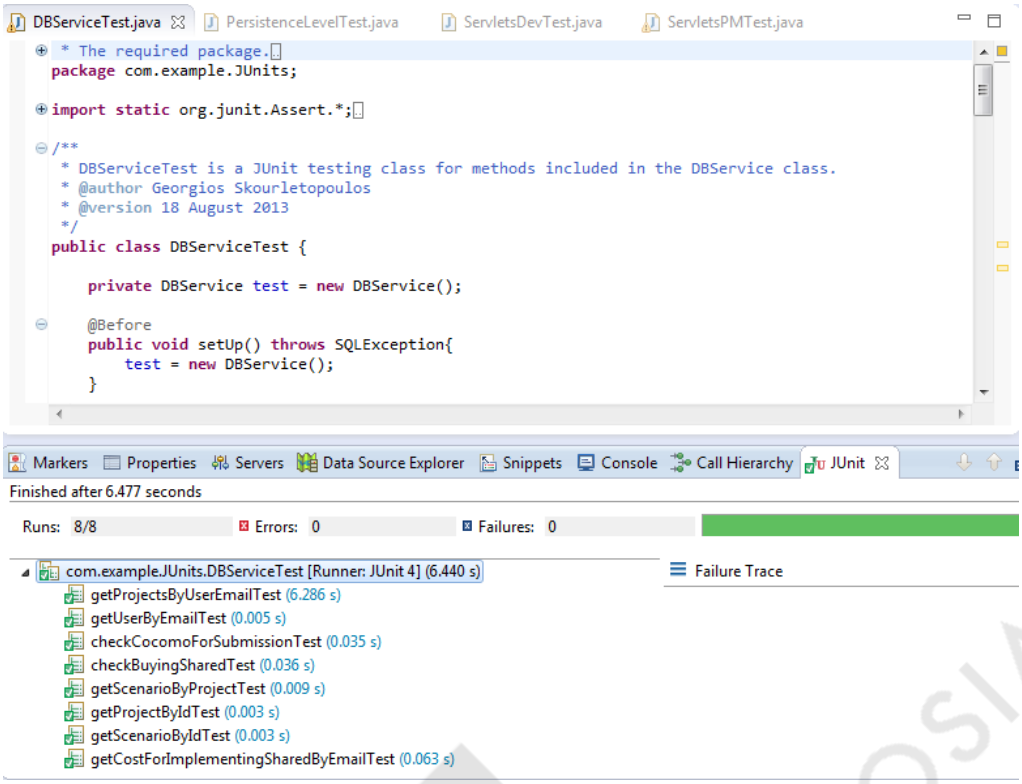
Use Case	Procedure	Expected Output	Pass / Fail
Create cost estimation for SaaS implementation	Leave a field blank or insert a wrong data input type	Calculation unsuccessful with the corresponding notification message under the affected field(s)	Pass
	Provide a name that already exists in the database	Submission unsuccessful with the corresponding warning message	Pass
	Provide weighted priority ratings that do not sum up to 100%	Calculation unsuccessful with the corresponding warning message	Pass
	Insert appropriate data input types and fill in all fields	Calculation and submission successful	Pass
	Delete a created cost estimation	Deletion successful	Pass
Create debt estimation at cloud service utility level	Leave a field blank or insert a wrong data input type	Calculation unsuccessful with the corresponding notification message under the affected field(s)	Pass
	Provide a name that already exists in the database	Submission unsuccessful with the corresponding warning message	Pass
	Insert appropriate data input types and fill in all fields	Calculation and submission successful	Pass
	Delete a created debt estimation	Deletion successful	Pass
Create graph / dashboard	Select no (or one) debt estimation	A notification message should appear	Pass
	Select two or more debt estimations	Generation of graph with the corresponding tables	Pass
Share estimation	User provides a user email that is invalid (not registered user)	Sharing unsuccessful with the corresponding notification message	Pass
	User provides a user email with which the estimation / project has been shared before	Sharing unsuccessful with the corresponding notification message	Pass
	User provides the same email with himself / herself	Sharing unsuccessful with the corresponding notification message	Pass

UNIVERSITY of NICOSIA

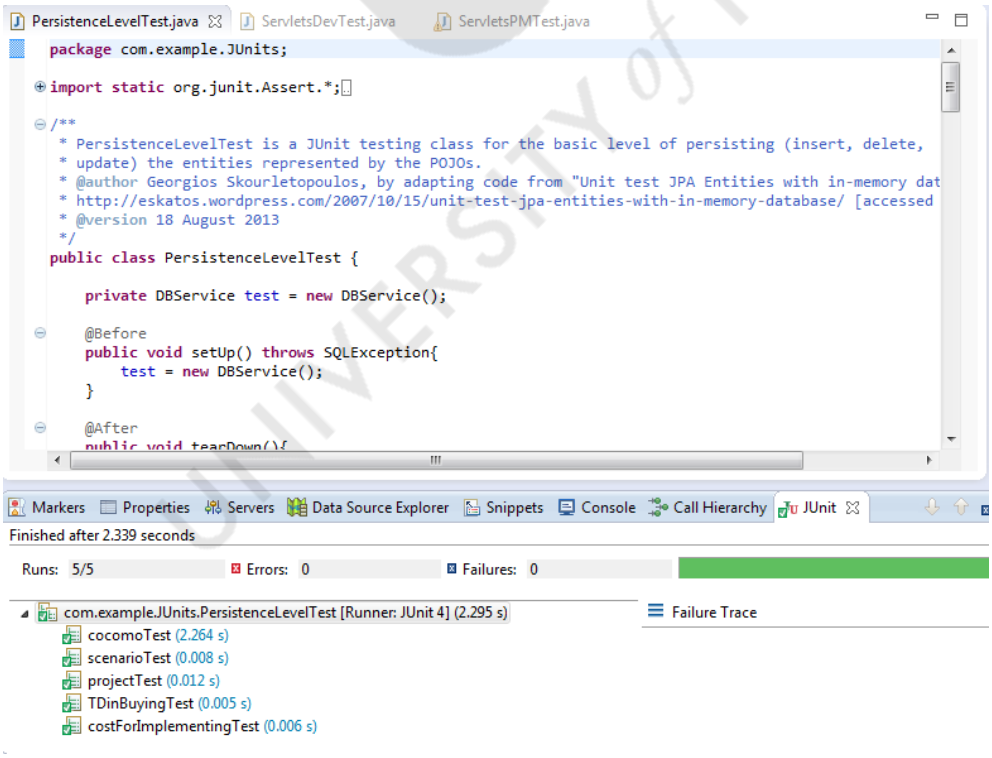
Use Case	Procedure	Expected Output	Pass / Fail
	User provides a valid user email with which the estimation / project has not been shared before	Sharing successful	Pass
Log out	User clicks on logout hyperlink	User session is destroyed and user is directed to dummy login page	Pass



Appendix D.2. Testing for CalculateCocomo methods

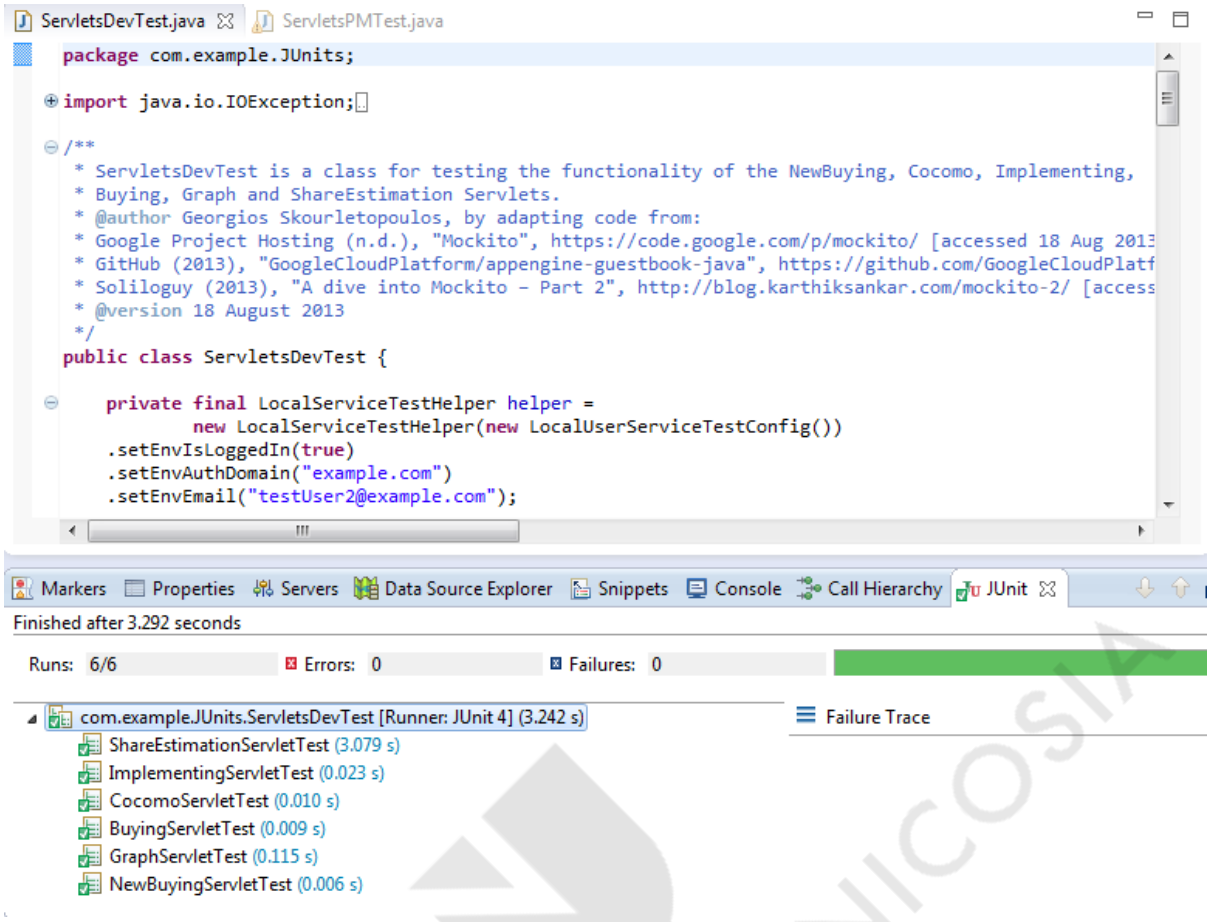


Appendix D.3. Testing for DBService methods



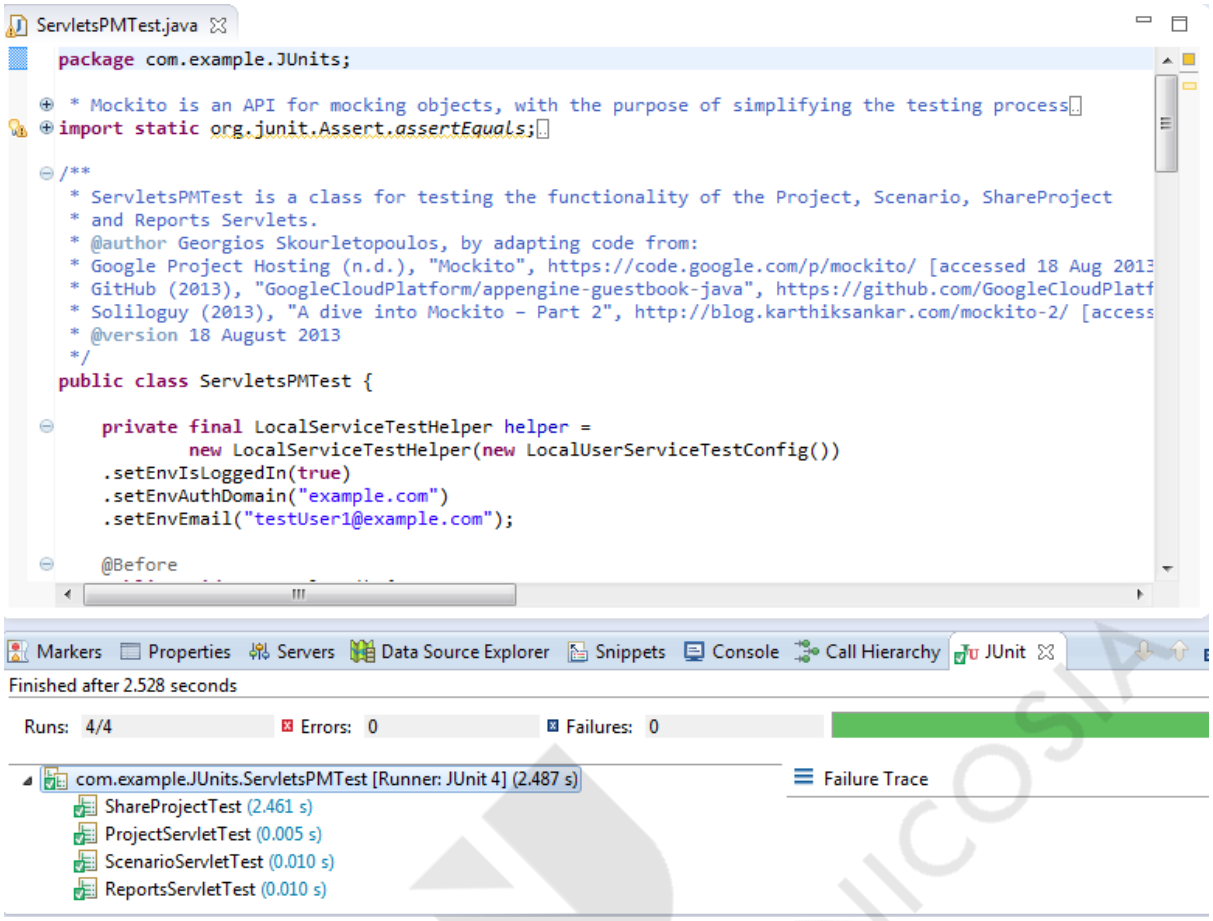
Appendix D.4. Testing for the level of persisting the entities represented by the POJOs

UNIVERSITY of NICOSIA



Appendix D.5. Testing the functionality of `NewBuying`, `Cocomo`, `Implementing`, `Buying`, `Graph` and `ShareEstimation` Servlets

UNIVERSITY of NICOSIA



Appendix D.6. Testing the functionality of `Project`, `Scenario`, `ShareProject` and `Reports` Servlets

UNIVERSITY of NICOSIA

Appendix D.7. Product evaluation vs. Requirements elicitation and analysis

Requirement	Implementation	Rationale
6.2.2.1.1	2	To ensure the tool will operate quickly.
6.2.2.1.6	2	To ensure system's customizability between different operations.
6.2.2.1.8	2	To ensure the tool is appealing and easy to use.
6.2.2.1.10	2	To ensure the interaction between the system and user informing about the outcome of the performed operations.
6.2.2.1.13	2	To avoid possible deliberate or accidental attacks.
6.2.2.1.14	2	To prevent SQL injection attacks and ensure that the user will fill in the correct typed data inputs.
6.2.1.1.1	2	To enable the COCOMO estimation creation.
6.2.1.2.1	2	To enable the cost estimation creation.
6.2.1.3.1	2	To enable the debt estimation creation.
6.2.1.4.1	2	To enable comparisons between different debt estimations at cloud service utility level.

Key	Requirement Satisfaction
0	Not met
1	Partially met
2	Fully met

UNIVERSITY of NICOSIA