ACCOUNTING, PRICING AND CHARGING SERVICES FOR GRID-BASED SERVICE PROVISIONING ENVIRONMENT

Buthelezi Mcebo Elijah

(20022424)
(B.Sc. Hons. Computer Science)

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Department of Computer Science,
Faculty of Science and Agriculture,
University of Zululand

2008

DECLARATION

I declare that this dissertation on, Accounting, Pricing and Charging Services for Grid-based Service Provisioning Environment is my work, and has never been presented or submitted in any form for any degree or diploma in any university. All the sources of information used have been duly acknowledged both in text and in the bibliography.

*Signature*_____

BUTHELEZI M.E.

DEDICATION

To my Mom

ACKNOWLEDGEMENTS

I am thankful to a number of people. Without their assistance and support, the outcome of this research work would not have been nearly as good as it turned out. First and foremost, I thank my supervisor, Prof M.O Adigun. Besides being a coauthor on the publications that came from this work, he was always eager to discuss problems and ideas, and his constant enthusiasm and positive spirit was a great source of inspiration.

I also thank O.O. Ekabua and J.S. Iyilade, my mentors for constructive comments, useful criticisms and support.

My gratitude and special thanks also go to all my research colleagues. Over the years, we have come to share a lot in common: research field, office, hostel rooms, thoughts, and laughter. It has been a pleasure working together.

I also thank Sabathile Mkhwanazi for her words of encouragement and useful contribution.

I also take this opportunity to thank the staff members at the Department of Computer for their friendliness and support.

Last but not least, I thank the ones I hold dear: mom, my brothers and sisters, relatives and friends for cheering me up and for making my spare time a great pleasure.

Thank you all.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Acronyms

GUAPCA	GUISET Usage Accounting, Pricing and Charging System Architecture
ABC	Activity-Based-Costing
ASP	Application Service Provider
CA	Charging Agent
CPU	Central Processing Unit
CSUB	Consumer-Service usage Bill
CSUR	Consumer-Service usage Record
DGAS	Distributed Grid Accounting System
e-Business	electronic Business
e-Commerce	electronic Commerce
e-Service	electronic Service
GQoSM	Grid Quality of Service Management
GRACE	Grid Architecture for Computational Economy
GRM	Grid Resources Meter
GRS	Grid Resource Supermarket
GSAX	Grid Service Accounting Extension
GTM	Grid Trade Manager
GTS	Grid Trade Server
GUISET	Grid-Based Utility Infrastructure for SMMEs - enabling Technologies
HLRs	Home Location Registers
IT	Information Technology
MACS	Metering and Accounting Composite e-Services
MOGAS	Multi-Organization Grid Accounting System
OGSA	Open Grid Services Architecture
PAM	Price Adjusting Mechanism
PCM	Price Controlling Mechanism
PREC	Price Recommender
PREG	Price Regulator

QoS	Quality of Service
RPFM	Resources Pricing Fluctuation Manager
RUR	Resource Usage Record
SaaS	Software-as-a-Service
SLA	Service Level Agreement
SMMEs	Small Medium and Micro Enterprises
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SUR	Service Usage Record
UDDI	Universal
UM	Utilization Monitor
URA	User Rating Agent
WSDL	Web Service Definition Language
MPRC	Market Price Rate of Change
QCPA	QoS Based Competitive Pricing Algorithm
UCA	User Charging Algorithm
SUR	Service Usage Record
RMP	Regulated Market Price
SEM	Service Evaluation Module
MRC	

ABSTRACT

The Grid computing idea has recently received widespread interest within the commercial environment. However, in spite of progress made to introduce Grids into the commercial environment, *Accounting, Charging and Pricing of Service* usage are still challenging issues. Current *Accounting, Pricing and Charging systems* are inadequate because most of them are based on a rigid and inflexible pricing and charging mechanisms.

In this study we address these challenges by proposing *GUISET Usage Accounting, Pricing, and Charging System Architecture* (GUAPCA). GUAPCA is based on a competitive market approach, where prices are determined by the forces of demand and supply. We also assume a *Grid Service* market environment that is dynamic and, therefore, service providers and consumers can join and leave the system at anytime. GUAPCA is comprised of three main components namely - accounting, pricing and charging service components. The accounting service component aggregates the service usage by specific users and consumers while the pricing service component determines the price of services based on the market forces of demand and supply. It further controls the market unit prices using market price limits set by the delegated pricing authorities. Moreover, charging is achieved through an incentive-compatible model where consumers are charged based on a combination of their reputation and actual service usage.

We carried out simulation study of the GUAPCA system and evaluated its performance experimentally using market efficiency and fairness of service price or charge as metrics. Our simulation results showed that GUAPCA price adjusting mechanism conforms to the micro-economic principle of determining the market unit price based on demand and supply. For instance, when quantity supplied was 10 units and quantity demanded was 2 units, the market unit price was decreased from $10.00 to $2.00 thereby encouraging more consumers for the service. Also, our approach shows fairness in pricing and charging users. In a case where actual cost of service usage was $260.00, a rebate of 10% was given based on the consumer's rating for service usage hence, the actual charge reduces to $222.30.

We concluded that GUAPCA, as proposed in this research, is an efficient and fair mechanism for pricing and charging service usage in a perfectly competitive Grid-based service provisioning environment.

CHAPTER ONE

INTRODUCION

As the Internet becomes ubiquitous and broadband access becomes commonplace, the drive towards services within the Information Technology (IT) industry is gradually taking shape. The Grid computing technology has been evolving beyond the borders of research and academia, to becoming a key infrastructure for business collaboration and enterprise application integration (Afgan and Bangalore, 2007; Huhns and Singh, 2005). Similarly, the utility computing business model is becoming a dominant business model for providing on-demand access to Grid services within a commercial environment. In the software industry, this has led to the notion of Software-as-a-Service (SaaS).

In view of these trends, utilizing Web Services and Grid Services have emerged as two complementary technologies facilitating the realization of service oriented commercial Grid. As it should be expected, the move from product-orientation to service-orientation within the software industry is already having significant impact on applications (Vassiliadis, et al,2006), such as e-commerce, e-health, e-business, thereby allowing more collaboration and coordinated resource sharing. Within the service environment, service consumers can remotely access resources offered by independent service providers and only pay for what is used. Usage accounting, pricing and charging methodologies are, therefore, vital to the successful operation of such service environments. Currently, most pricing and charging methodologies employed are too rigid and inflexible to cater for the dynamic nature and high-level customization required of service oriented commercial Grid. In this study, we present usage accounting, pricing and charging mechanisms for service oriented commercial

Grid. The pricing mechanism employs a competitive pricing approach with Quality of Service (QoS) constraint, while charging is done with consideration of various incentives for loyal and long-term consumers.

The rest of this chapter is organized as follows: In Section 1.1, the background of the study is described. The problem that this work addresses is described in Section 1.2. The research questions are presented in Section 1.3. The research goal and objectives are presented in Section 1.4. The research methodologies and publication outputs are listed in Section 1.5 and 1.6, respectively. The overall structure of the dissertation is presented in Section 1.7.

1.1 Background

The rapid adoption of the Internet and broadband technologies worldwide has tremendously increased the delivery of different online services (Reichl and Stiller, 2003; Narahari, et al, 2005; Jagamathan and Almeroth, 2004; Kannan, et al, 2008). These developments have opened new opportunities for service provisioning through the global information superhighway. We are now witnessing the emergence of new infrastructure for collaboration and resource sharing through the Grid. In addition, new service delivery models (Utility Computing and Software-as-a-Service) have equally emerged to facilitate the delivery of IT services on a "pay-per-use" basis.

Although, the Grid idea was originally motivated by problems within the science and academic community, its adoption of a service-oriented architecture through the Open Grid Service Architecture (OGSA) is making it possible to apply the Grid ideas in some other application domains requiring stateful and reliable services (Foster, et al, 2002b). It is, therefore, expected that Grid would find more application in commercial environment by enabling a more stable infrastructure for service provisioning. In

addition, the utility computing business model is evolving as the paradigm for fulfilling computing needs and IT services (Rappa, 2004). It is a key to the adoption of Grid in a commercial environment. In the utility business model, service providers make resources and infrastructure management services available on-demand to consumers and charge them per-usage rather than flat rates. It is a revolutionary financial and technological model for delivering IT services, which is enabled by a virtualized, optimized, scalable, fully automated and shared IT infrastructure like Grid (Rappa, 2004). Yeo, et al,(2007) discussed the benefits of utility computing and how it has changed IT in the past few years. Moreover, in the software industry, the idea of on-demand delivery of services and management infrastructure has led to the concept of SaaS. The goal of SaaS is the delivery of application software remotely through a subscription-based fee model rather than an acquisition model.

Grid is an infrastructure that has the potential to exploit these service delivery models. Grid infrastructure is described as a collaboration of different dispersed organizations with the aim of sharing and coordinating physically distributed resource virtualized as a single resource to the users (Yeo, et al, 2007). To realize virtualization in Grid, Grid middleware systems like Globus, Condor, and Sun Grid Engine have been developed in different Grid projects. The middleware systems are intermediaries between the Grid service providers and consumers. Therefore, Grid middleware contain secondary services such as usage accounting, pricing, charging, billing, monitoring, metering, etc, that are required to manage Grid network and primary Grid services (Software applications, CPU, storage space, etc).

In view of the on-going IT trends and emerging technologies, our research center has proposed the Grid-Based Utility Infrastructure for Small Medium and Micro

Enterprises (SMMEs)-enabling Technologies (GUISET) as an e-infrastructure that will provide access to IT services as utilities to SMMEs. To enable variable costing for Grid services, usage accounting, pricing and charging becomes essential aspects of the GUISET Grid-based services. The lack of applicable, distributed and efficient accounting schemes for commercial resource and service consumption has equally been identified as an important open problem in most commercial grid environments (Chowdhury, 2006). An accounting service aggregates the service usage by specific users, while the charging service applies the service provider's pricing schemes to the accounting data, and generates bills for the users (Agarwal, 2003).

Pricing can be considered as an effective means to recover Grid service production costs (input costs). In order to attract consumers, some service providers are offering lowest prices for services, and this result in different prices for similar services. The primary pricing scheme currently offered by service providers is a flat rate pricing scheme that allows users to access a service for a monthly flat fee. Although, the usage based pricing scheme exists, service providers mainly offer flat rate pricing schemes. Pricing schemes should be such that consumers are allowed to select among different set of services in a controlled manner (GÖhner, et al, 2007). The charging model of a service defines its usage metrics and the basis on which users are charged. Some services may use resource consumption measures; others may have pre-defined costs per request, or a cost that varies predictably with certain parameters of the request (Agarwal, 2003).

In the service oriented commercial Grid, service providers enter or depart the environment any time and this has negative impact on QoS levels. Therefore, in the service oriented commercial Grid, service consumers are more concerned about the

level of QoS that a given Grid service provider offers in fulfilling the task/job submitted. Thus, Service Level Agreements (SLAs) have been proposed (Czajkowski, et al, 2004; Kounev, et al, 2007; Xiaorong, et al, 2008) as a mechanism to enforce the QoS level that the service consumers and providers have agreed upon. The Grid Quality of Service Management (GQoSM) proposed in Al-Ali, et al, (2003) have three distinct features:

i. Support for resource and service discovery based on QoS properties.
ii. Support for providing QoS guarantees at the middleware and network level, and establishing SLAs to enforce these.
iii. Providing QoS management on allocated resource based on a pre-negotiated SLA.

In GQoSM, three QoS levels for Grid Services are identified. These are guaranteed, controlled load, and best effort. This classification categorizes the Grid Services according to their capabilities. In this study, we exploit this classification of services in order to price service unit based on the QoS levels that service possesses.

1.2 Statement of the problem

The acquisition of IT resources such as storage space, Computer Processing Unit (CPU), and software by SMMEs especially in developing countries is a daunting problem. The high costs imposed on IT resources, infrastructure, and professionals needed to maintain the IT resources make it difficult for SMMEs to survive in the business environment (Vickery, et al, 2006; Guarise, et al, 2005). Thus, the GUISET architecture is proposed partly to reduce these problems by enabling the provision of the IT resources as services. Existing usage accounting, pricing and charging schemes may probably not be suitable for GUISET environment. This is because GUISET

services are to be delivered based on consumer's specific QoS requirements, on an on-demand basis, in a dynamic Grid environment. Therefore, usage accounting, pricing and charging of services may be uncertain. This research addresses these uncertainties of usage accounting, pricing and charging with regard to our GUISET Grid research environment.

1.3 Research Questions

The research questions that were addressed by this dissertation are:

i. What pricing scheme is suitable for GUISET such that SMMEs can affordably have access to IT services provided?
ii. How are the accounting-records supplied then mapped to the charging service in GUISET?
iii. How can incentives be awarded to service providers and consumers in our GUISET Grid environment?

1.4 Goal and Objectives

1.4.1 Goal

The goal of this research work was to develop a custom-made usage accounting, pricing and charging services architectural framework and computational model for GUISET.

1.4.2 Objectives

In order to realize the goal above, the following objectives were identified, which were to:

i. investigation existing approaches to manage usage accounting, pricing and in a Grid environment.

ii. develop integrated system architectural model for usage accounting, pricing and charging in GUISET.

iii. implement and evaluate the model as a proof of concepts.

1.5 Research Methodology

This research work used both theoretical and formulative approaches. To this end, a number of research methodologies were pursued in the effort to realize the set objectives of the research. The activities involved were:

a. Literature Review

Literature search on usage accounting, pricing and charging was conducted to evaluate the work that other scholars had done. The search focused on how existing scholarship formulated service usage accounting, pricing and charging, how the usage data were collected from different service sites, how that data were stored, how reports were generated, and how charges were calculated?

b. Formulation of integrated system model architecture

An architecture integrating, Usage Accounting, Pricing and Charging System Architecture for GUISET (GUAPCA) was developed. First a number of design criteria were identified then Algorithms and techniques for usage accounting, Pricing and Charging services suitable for fulfilling the design criteria were developed.

c. Simulation Experiment

A simulator was developed in order to test the performance of GUAPCA as proof of concept. The metrics used in order to verify the performance of GUAPCA were efficiency and fairness. Several experiments were conducted in the simulator.

1.6 Research Contributions

Given the research questions delineated in Section 1.3, our main goal was to develop a custom-made usage accounting, pricing and charging services architecture for GUISET. This was to particularly advance the state-of-the-art in usage accounting, pricing and charging both to our GUISET research focus and other Grid-based service provisioning environment.

Therefore, in meeting this goal, our main contribution to knowledge in this research is the development of GUAPCA as a usage accounting, pricing and charging services in GUISET. In the context of this architecture, we describe a number of strategies in order to achieve a custom-made usage accounting, pricing and charging for GUISET. The strategies are as follows:

1. We presented the design objectives for usage accounting service in GUISET. Thereafter, we proposed two subcomponents that have complementary task to handle service usage accounting in the service provisioning environment.
2. We further presented the pricing service that comprising of two main algorithms: (i) recommend the unit price of the service based on the market demand and supply with the consideration of the QoS constraints, (ii) evaluate the recommended unit price of the service against the predefined market price limits. The market price limits are preset to prevent the over-pricing and under-pricing of the service unit.
3. Finally, we described the incentive-compatible charging approach that can be expanded in order to enforce other policies such as SLAs.

This study has resulted in the following peer-reviewed publications:

- Buthelezi,M.E. Adigun,M.O. Ekabua,O.O. and Iyilade,J.S. (2008), *"Accounting, Pricing and Charging Service Models for a GUISET Grid-Based*

Service Provisioning Environment", In Proceeding of the 2008 International conference on E-learning, E-business, Enterprise Information system, and E-government, USA, Page(s) 350 - 355. In this paper, we describe an SMME-based enabling infrastructure technology called GUISET and present an adaptive usage accounting, pricing and charging models to facilitate efficient and effective service provisioning in our Grid-based service provisioning environment (GUISET).

- Buthelezi, M.E. Iyilade, J.S. Adigun, M.O. (2008). *"Dynamic Pricing and Charging Models for Next Generation e-Services"*, In proceeding of the 10th Annual Conference on World Wide Web Applications, South Africa, 3-5 Sept 2008. The paper presents the requirements of pricing and charging in e-services. Thereafter it proposes the pricing model that employs a competitive pricing approach with QoS constraint, while the charging is done with consideration of various incentives for loyal and long-term consumers.

The research publications presented above are a summarized usage accounting, pricing and charging models descriptions and expanded upon by this dissertation with the simulation results presented in chapter five. In order to guide the reader through the remaining chapters, the following section contains a brief outline of the dissertation structure.

1.7 Dissertation Outline

The remainder of the dissertation is organized as follows: In Chapter Two, we explain foundation concepts of distributed systems that this research work builds upon. We are particularly interested in the Grid paradigm.

Following this, Chapter Three reviews the existing literature related to the key issues addressed in this study: usage accounting, QoS constrained competitive pricing approach, and incentive-compatible charging approach. In Chapter Four, we describe the GUACPA system architecture and techniques that were used to accomplish the objectives of this research work.

Next in Chapter Five, we present the simulation and results analysis of the system architecture. We conclude in Chapter Six with a summary of our research and give suggestions for future work.

CHAPTER TWO

BACKGROUND CONCEPTS

This chapter presents the foundation technologies that this research work builds upon. The background is divided into six main Sections. Section 2.1 presented an overview of distributed computing. In Sections 2.2 – 2.5, we specifically discussed in detail the Service Oriented Computing (Section 2.2), the Grid computing (Section 2.3), the Utility computing (Section 2.4), and SaaS (Section 2.5). Section 2.6 presented GUISET architecture, which is an architecture that intends to exploit the distributed computing concepts that were described in Section 2.2 - 2.5. In Section 2.7, the summary of the chapter is presented.

2.1 Overview of Distributed Computing

Historically, computer systems had undergone two major revolutions. The first was the development of powerful microprocessors and the second one was the invention of high-speed computer networks. As a result of these developments, distributed systems have become a powerful tool for sharing resources to achieve high performance in accomplishing different tasks. The paradigm of Distributed systems has led to the following three types of distributed systems, namely *distributed computing systems (Cluster and Grid computing), distributed information systems* and *distributed pervasive systems*. Tanenbaum and Steen, (2007) discuss the four important goals that should be met to make building a distributed system worth the effort. A distributed system should make resources easily accessible; it should reasonably hide the fact that resources are distributed across a network; it should be open; and also be scalable. This study focuses more on Grid paradigm of

Distributed computing. Mattern, (2000) defined distributed computing as several computers that communicate over a network to coordinate the actions and processes of common application.

Distributed computing techniques have gained much interest in recent years due to the proliferation of the Web and other Internet-based systems and services. The success of Web services has influenced the way in which Grid applications are written (Patel and Darlington, 2006). Web services (W3C, 2002) and Grid computing (Foster, 2002a) are emerging complementary technologies towards realizing the service-oriented promise. Whereas Web services technology and standards focus more on discovery and invocation of services, Grid Computing addresses the issue of virtualization of resources and their state management (Jacob, et al, 2005; Ramaswamy and Malarvannan, 2006).

In the next Sections, Service Oriented Computing, Grid Computing, Utility Computing and SaaS are briefly introduced as background concepts to this research.

2.2 Service Oriented Computing

Previously, the sharing of resources between distributed computers has been considered thus it is not a new concept in itself (Tanenbaum and Steen, 2007; Galli, 2000). However, most early systems were built for special purposes and so they usually employed ad hoc mechanisms in order to interoperate. This meant that systems were inflexible, relied on static links between components and used application specific protocols and data models (Huhns and Singh, 2005). Service Oriented Computing (SOC) addresses these shortcomings by allowing services to be discovered and invoked automatically at run-time (Papzoglou, et al, 2007) rather than through manually specified and fixed application interfaces. Therefore, it has been

suggested as a suitable paradigm for distributed systems where many diverse software components need to interact seamlessly. In this approach software functionalities and other behaviors are offered by their providers as services over the network (Huhns and Singh, 2005).

SOC is based upon an underlying Service Oriented Architecture (SOA). SOA is an architectural approach whereby an application is composed of independent, distributed and co-operating components called services. The services can be distributed within or outside of the organizations physical boundaries and security domains. The SOA has been successfully implemented using Web Services. Web services are loosely-coupled, platform-independent, self-describing software components that can be published, located and invoked via the web infrastructure using a stack of standards such as Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL), and Universal Description, Discovery and Integration (UDDI) (Zhang, et al, 2007). Figure 2.2 provides a description of the relationship amongst the major elements of SOA.

SOC is based on the elements of SOA, which are: loose coupling, implementation neutrality, flexible configurability, persistence, granularity and teams (Huhns and Singh, 2005). It supports the development of applications as if they were a connected network of functionalities (services) available in a network-enabled environment, within and across different organizations. Through the adoption of SOC, traditional electronic-Commerce (e-Commerce) is giving way to the new service paradigm referred to as electronic-Services (e-Services) (Vassiliadis, et al, 2006). Although, the Grid idea was originally motivated by problems within the science and academic community, its adoption of a SOA through the OGSA is making

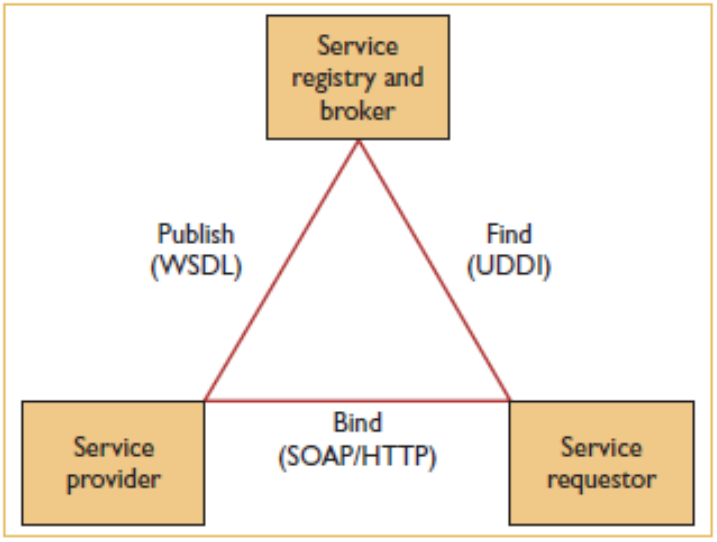

Figure 2. 1: Service Oriented Architecture

it possible to apply the Grid ideas in some other application domains requiring stateful and reliable services (Foster, et al, 2002b).Therefore, Grid computing has become a key infrastructure for business collaboration and enterprise application integration. Utility computing has become its business model for providing services on-demand (Rappa, 2004; Huhns and Singh, 2005). In the software industry, this has led to the notion of SaaS. In the following Sections we discuss Grid Computing.

2.3 Grid Computing

Grid computing (Foster, et al, 2002a), is concerned with "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations." Srinivasan and Treadwell (2005) also defined Grid computing as a form of distributed computing in which the use of disparate services such as compute nodes, storage, applications and data, spread across different physical locations and administrative domains, are optimized through virtualization and collective management.

Therefore, in a Grid Computing network, services and resources are made readily available to consumers, analogous to electrical power and other public utilities. The services are consumed efficiently and securely with minimal human intervention (Parashar and Lee, 2005; Papzoglou, et al, 2007) during service composition, discovering, selection, and so forth. The SOC is the paradigm which enables the provision of services.

The move of Grid computing from academia and scientific research to the mainstream of enterprise applications (Nadiminti and Buyya, 2005) has resulted in challenges such as; regulation of service usage, transaction management, usage accounting, pricing of services and charging for consumers for service usage. However, deciding the appropriate pricing of services in the service market is a non-trivial issue. For example, in a Grid environment, a major issue is how to obtain the information on demand and supply of resources, which invariably have a great impact on service prices. In addition, Grid systems are dynamic, in the sense that both service providers and consumers can join and leave at their own desires. So market demand and supply of services are dynamic and stochastic (Zhao, et al, 2007).

Grid service providers are decentralized and heterogeneous, belonging to different organizations. Consumers from different regions harness these services. As shown in Figure 2.1, assuming *organizations A, B, C, and D* formed a Grid, the organizations are not physically connected or located in the same geographical environment, but they are able to share the services they own with one another. For example, *Organization A* may access resources owned by *organization B* for a particular period of time. It is, therefore, crucial to ensure that necessary coordination schemes are in

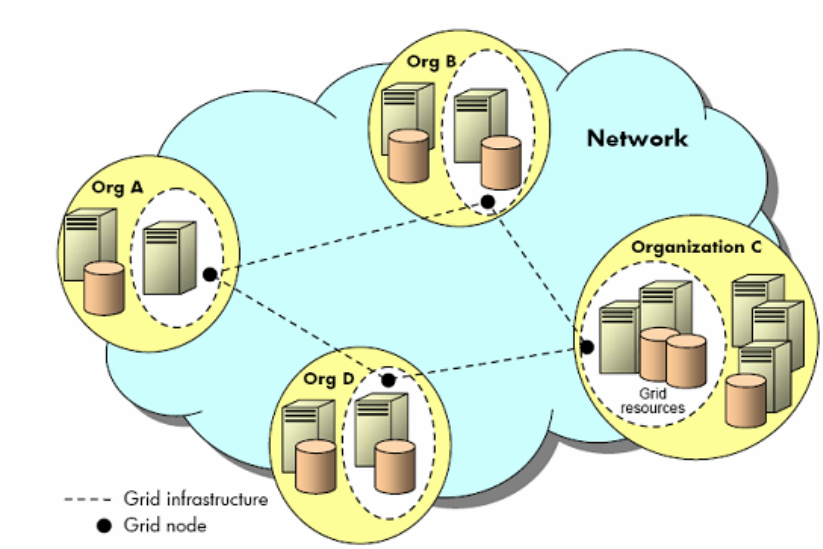

Figure 2.2: Overview of Grid Infrastructure: Grids allow resources to be shared across organizational boundaries (Srinivasan and Treadwell, 2005)

place such that the Grid functions well and meet its primary purpose of sharing geographically dispersed resources.

Utility computing and SaaS are models that are exploited in Grid computing. In the next Sections we discuss these models.

2.4 Utility Computing

Utility computing is a service provisioning model where service providers make computing resources and infrastructure management available to the service consumers on-demand and charge them for usage. It is a revolutionary financial and technological model for delivering IT services, which is enabled by virtualized, optimized, scalable, fully automated and shared Grid. According to Rappa (2004), this service provisioning model is envisioned to be the next generation in IT evolution that

depicts how computing needs can be fulfilled in the future IT industry. As a service provisioning model, it advances the capabilities of distributed system to both service providers and consumers.

2.5 Software-as-a-Service

The concept of SaaS supported by SOC is revolutionary and appeared first with the Applications Service Provider (ASP) software model (Vassiliadis, et al, 2006). Therefore, in the software industry, the idea of on-demand delivery of services and management infrastructure has led to the trend towards SaaS. The goal of SaaS is the delivery of application software remotely through a subscription-based fee rather than being sold for perpetual use. The users do not buy the license of the software, but only a right to use it. Therefore, SaaS is the model in which application software is delivered remotely through a subscription-based fee rather than being sold for perpetual use (Goldi, 2007; Li, et al, 2008). A user may subscribe to all the features or functionalities or just some of them for use. The applications are hosted in a data centre and maintained by the service provider (Anerousis and Mohindra, 2006). The characteristics of the SaaS model are as follows:

i. A multi-tenant design where an instance of the application accommodates multiple users or even multiple resellers of the service with each reseller serving its own pool of users.
ii. A charging model where customers pay for the services on a metered basis.
iii. Support for all the functions necessary to provide the application as a service.
iv. High level of application customization to avoid major implementation and integration costs.

SaaS enables the software industry to deliver customized software applications to different consumers over the network.

2.6 Grid-Based Utility Infrastructure for Small, Medium and Micro Enterprises (SMMEs)-enabling Technologies (GUISET) Architecture

In view of the capabilities that distributed computing possesses, our research centre (Adigun, et al 2006) proposed GUISET architecture shown in Figure 2.2. GUISET is an architecture that aims at providing distributed services such as application software, storage, and CPU capabilities as utilities.

This is envisioned as an approach that can provide an affordable access to services deployed in Grid environment. The SMMEs, who lack their own internal IT infrastructure, are prominent beneficiaries of such a service provisioning environment as they will have access to the IT services available on-demand. There is no longer the need for them to invest heavily or encounter difficulties in building and maintaining internal IT infrastructure as these will be available on-demand.

The fundamental component of this GUISET architecture is a pool of Grid Services. The Grid Services are high-level services composed by the multiple lower level services provided by distinct independent distributed organizations. All the Grid Services are integrated seamlessly into the GUISET based environment to form and support the running of applications tasks, jobs which are submitted to the middleware layer (utility broker). The service management strategies are the key to GUISET. For an efficient and effective service management, the following service management strategies need to be investigated for GUISET:

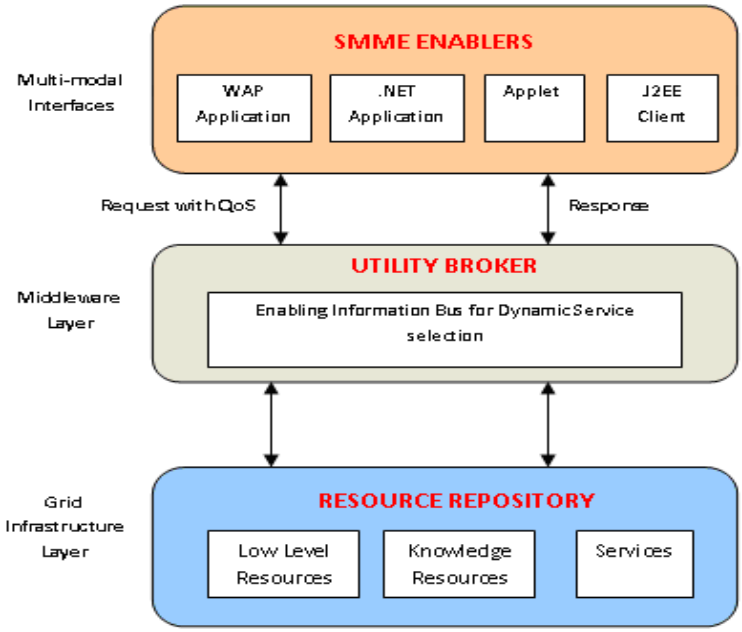

Figure 2. 3: An overview of GUISET architecture (Adigun, et al, 2006)

i. *Dynamic Service Composition and Selection, SLAs and Workflow Management* – to automate and dynamically manage the entire end-to-end application lifecycle of interlinked stages with policy enforced across organizations.

ii. *Monitoring, Metering Services* - the monitoring services keep track of how services and resources are performing, while the metering services accommodate end-to-end resource consumption measurements.

iii. *Accounting, Pricing and Charging Services* – accounting service provide the mechanism for service providers to be paid for authorized use of their services. It supports the recording of usage data, analysis of that data for the purposes of charging. Pricing service provides the mechanism to fix prices for the services based on their market demand, supply and QoS. Charging

service computes the total bill for service consumer based on service(s) consumption, and award incentive to those who may qualify based on determined policies.

From the perspective of this work; these are some of the strategies that are needed in GUISET. This research work provides solution to how the accounting, pricing and charging services can be formulated and implemented in GUISET service provisioning environment.

2.7 Summary of the Chapter

This chapter presented the background distributed computing concepts that are related to this research work. GUISET architecture has also been described as the background of this research.

CHAPTER THREE

LITERATURE REVIEW

In this chapter we review the existing literature related to the key issues addressed in this dissertation. In Section 3.1, we review the literature on existing approaches and models for usage accounting, pricing and incentive based charging model and some research efforts towards addressing the challenges of usage accounting, pricing and charging model for the distributed computing environment is presented. In Section 3.2 we present the summary of the chapter.

3.1 Usage Accounting, Pricing and Charging Approaches and Models

The management of services in distributed computing environment, specifically Grid computing has become a critical issue as it becomes the medium for electronic-Business (e-Business). The current solutions do not address the challenges that are emanating from the commercialization of Grid. Thus, issues such as service usage accounting, pricing and incentive based charging of services need to be addressed to enhance the resource management techniques for commercial Grids. In this Section, the works that have been done towards addressing these issues are presented.

3.1.1 Usage Accounting Approaches and Models in Grid Environment

Service usage accounting in a commercial Grid environment enforces proper resource management. Thus, Gardfjäll (2004) described usage accounting system as a system that should provide data to do the following:

i. form the basis for economic compensation.
ii. used to enforce Grid resource allocation.
iii. allow the tracking of resource usage and jobs submitted, and

iv. enable the dynamic allocation of resources based on the priority and reputation of the user.

These make usage accounting to be an important component of commercial Grid usage. Therefore, usage accounting forms a crucial part in seeking compensation for service usage by consumers.

3.1.1.1 Existing Usage Accounting Approaches

Agarwal, et al (2003) proposed an architecture for metering and accounting for composite e-Services (MACS) that provided an accounting, metering, billing, and monitoring components. MACS supports metering at request-level granularity thereby formulating a distributed accounting architecture that is scalable and supports service independence. Within this architecture, a particular service usage is expressed using application-level parameters rather than server-side resource usage metrics. This provides some trustworthiness between a service provider and consumers because the metering records are real-time managed. Furthermore, the service providers easily classify service consumption records. The architecture also supports service independence. Figure 3.1 shows an overview of MACS, with S_1-S_5 services fulfilling the request of the user, metering records from each service are mapped to the classifier which aggregates them to the database.

GÖhner, et al (2007) also, proposed an accounting model for dynamic virtual organizations in a Grid environment. This model adopts an Activity-Based-Costing (ABC) accounting approach. Within their model, the authors identify the costs of the service following an hierarchical approach, that is, from a parent service to the child services, the services are composed to accomplish any given task or job. The final costs in this model include the administration costs. Although, the accounting model

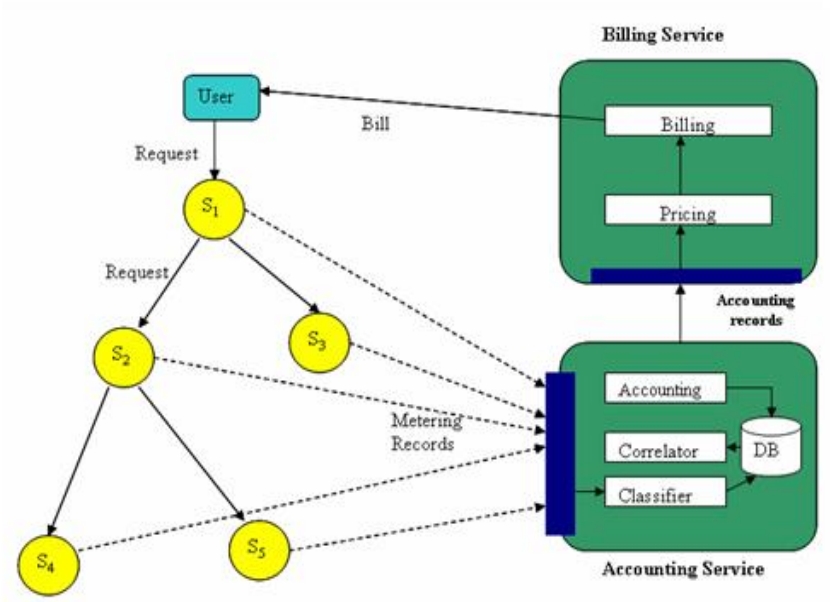

Figure 3. 1: Architecture for Metering and Accounting for Composite e-Services (Agarwal, et al, 2003)

proposed by these authors provided the possibility to bridge concepts of the Traditional Cost Accounting system and ABC accounting, it does not elucidate how the service's usage data is going to be gauged from the distributed, heterogeneous service provisioning environment and how the usage data is mapped to respective service providers. Therefore, there is a high possibility that service providers and consumers take advantage of each other.

Lim, et al (2005) proposed a Multi-Organization Grid Accounting System (MOGAS) that supports a multi-organization environment like the commercial Grid. The authors acknowledged the importance of proper management and usage accounting on resource spanning across dispersed organization. The lack of standardization of usage accounting in a multi-organization environment like Grid motivated the authors.

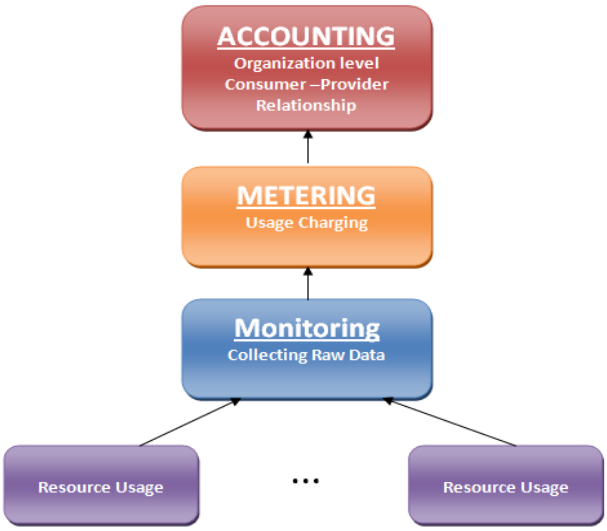

Figure 3. 2: Fundamental Building Blocks of Grid Accounting System (Lim, et al, 2005)

Thus, their accounting system architecture focuses more on the building blocks of any usage-based accounting system. Figure 3.2 shows how the underlying components of a usage-based accounting system interact with each other such that a usage accounting record is created with minimal difficulties.

The Grid Service Accounting Extension (GSAX) (Beardsmore, et al, 2002) was proposed to provide a modular approach accounting framework. The GSAX framework can be expanded by adding or changing the core components to suit the environment where it is applied. Furthermore, the underlying accounting system allows for accounting to be carried out at various application levels and they provide information at different degree of granularity. Another aspect of this theoretical framework is the possibility to integrate QoS parameters and SLAs at different levels

of the accounting framework. It makes provision for economy based QoS and SLAs to be implemented at the accounting level.

The underlying accounting subcomponent comprises two basic services: the account management service, which provides accounting-related information, and accounting records to higher-level components via adequate interfaces and accounting service which handles metering events. Thereby establishing interfaces with the lower level components of the framework. The accounting management and accounting service hold an instance of an account, which contains information about the current balance and the list of users authorized to use the account.

Barmouta and Buyya (2003), driven by the challenges of Grid accounting, proposed an infrastructure called the GridBank that provides accounting services for a grid environment. Gridbank is a secure accounting and payment handling system which maintains the user's accounts and resource usage records in the database. The work of these authors emphasizes the importance of proper resource management systems by including some payment mechanism.

Another relevant work was the Distributed Grid Accounting System (DGAS) (Guarise, et al, 2005), designed to support an economy-based approach usage accounting in order to regulate the distribution of Grid resources amongst authorized Grid users and implement resource usage metering, accounting and account balancing in a distributed Grid environment. The consumption of Grid resources by Grid users is registered in Home Location Registers (HLRs), which are responsible for managing both user and resource usage accounts. Furthermore, the HLRs have the capability to facilitate communication between different HLRs. They are also able to credit or debit different users or resource owners' accounts for the respective amount of

resource usage. These usage accounting systems show how the different accounts need to be credited or debited. Additionally, they only allow authorized users to access Grid resources, which make them even more trustworthy to service consumers and service providers. Pettipher, et al (2007), discussed the different usage accounting systems which have been developed and applied in various computational Grid project worldwide. The authors' discussion shows how important usage accounting is for accountability in resource usage in the Grid environment.

3.1.1.2 State of the Art in Usage Accounting in Grids

Usage accounting forms the basis for economical compensation and management of service usage in a commercial Grid environment. Currently proposed mechanisms are the foundation for usage accounting services that need to be customized and extended to meet the usage accounting for commercial setting of Grid such that services usage are traceable and easily evaluated. In this study we proposed a customizable usage accounting mechanisms that focuses more on tracing service usage by service consumers in order to enable the usage based incentive compatible charging of services.

3.1.2 Pricing Grid Services

Pricing a service is one of the important processes in a market. The price of the service influences its market supply and demand and vice versa. Therefore, price plays an important role in influencing user's preference for services. In order to determine the price in a Grid environment, Yeo and Buyya (2007) outlined the following four essential requirements for defining a pricing function for Grid:

i. The need for flexibility in the pricing function to help resource owners for easy configuration.

ii. Fair pricing function to enable resources to be priced based on actual units consumed by users.

iii. The need for dynamic pricing functions such that the price of each resource is not static and can change depending on operating condition.

iv. The pricing function should be adaptive to changing supply and demand of resources so as to compute the relevant price accordingly. As an example, if demand for resource is high, the price of the resource should be increased so as to discourage users from overloading this resource and to maintain equilibrium of supply and demand of resources (Yeo and Buyya, 2007).

In the following sub-Section 3.1.2.1, we discuss flat rate and usage pricing schemes. These pricing schemes have been used as the mechanism to overcome the pricing challenge associated with trading Grid resources or services.

3.1.2.1 Pricing Schemes

(i) Flat rate Pricing

Commented [M1]: Definr Flat rate pricing

Blefari-Melazzi, et al (2003) discuss flat rate and usage based pricing in the case of the Internet. For example, the flat rate pricing scheme:

i. is very simple and does not need any additional accounting architecture;

ii. allows consumers and service providers to have an accurate idea of costs and revenues, respectively; and

iii. promotes unrestricted use of services by consumers.

At the same time, it has some serious disadvantages and drawbacks namely:

i. Service charges do not depend on the usage of service, so it penalizes light consumers as compared to heavy ones (Blefari-Melazzi, et al, 2002).

ii. It is not an efficient pricing scheme from economic point of view; as it does not provide the possibility to pay on basis of the perceived QoS.

iii. It does not favor service efficiency, but it encourages resource waste and therefore, it does not guarantee high resource utilization.

(ii) Usage-Based Pricing

The usage-based pricing is the pricing scheme where consumers are charged based on the actual usage of a service. The advantages of usage-based pricing scheme are:

i. Consumers are charged based on their actual resource usage; therefore, it is fair to all consumers, light and heavy.

ii. Consumers are able to monitor the service usage cost without limitations to service usage.

iii. It promotes high resource utilization amongst consumers. However, it has the disadvantage that it is not easy to implement in the service provisioning environment.

In addition, Chowshury (2006) emphasized that the usage-based pricing should be fair so that it allows consumers to monitor their resource usage and to pay according to their QoS specifications. Therefore, we adopt usage based pricing scheme for GUISET as it is the pricing scheme that allows consumers to specify their QoS requirements.

3.1.2.2 Economic Models in Service Provisioning Environment

Various economic models have been applied in different service provisioning environment. Buyya, et al, (2002) discussed the different economic models which include: commodity market, posted price, bargaining, the tendering/contract-net, auctions, the bid-based proportional resources sharing, the

community/coalition/bartering, the monopoly and the oligopoly. We present a brief description of these models below:

a) The Commodity model

In this model, pricing strategies are the major concern as it employs pricing methods such as flat rate, usage-based, subscription (fixed rate for a period of time), and demand-supply. The resource owners specify their service price and charge users according to the amount of resource they consume. In the flat rate, once the price is fixed for a certain period, it remains the same irrespective of service quality or service demand. On the other hand, in the supply and demand approach, the price changes very often based on the quantity supplied or demanded. In principle, when the quantity demanded increases or quantity supplied decreases, price increases until a point of equilibrium between quantity supplied and demanded is reached.

b) The Posted price model

This model is based on advertisement for service discount or promotion offers in order to attract consumers to establish market share or motivate consumers to consider cheaper slots. Sales advertisement can occur, in a Grid computing environment, when a Grid opens new services and wants to attract users, or when a Grid wants to maximize resource utilization during off-peak time.

c) The Bargaining model

A prospective consumer can negotiate with a producer for a reasonable price. In a market, this often occurs when the consumer finds a more competitive price from other producers (price match). In a Grid environment, bargaining is based on different objective functions of resource owner and resource user. For example, resource owner

may reduce price for the resources with lower utilization or poor performance. Resource user may bargain for a lower price with promise to use more resources from this owner in the future.

d) The tendering or contract-net model

In this model, a bidding process is initiated by consumer. Each eligible producer responds with their available commodities and intended prices. Consumer compares each producer's bid and chooses the winner. The final result is a contract.

e) The Auction model

This model is quite popular for consumers to bid on a commodity advertised by a producer. The process is initiated by producer. There are many traditional auction methods, such as English auction, first-price sealed-bid auction, Vickrey auction, Dutch auction, and double auction.

f) The bid-based proportional resource sharing model

This model deals with shared resources, while most economic models deal with competitive resources. Each resource user gives a bid for a resource based on its demand function. Resource owner collects all bids and allocates resource to some or all of users based on the proportion of each user's bid.

g) Community/Coalition or Bartering/Share Holders Model

A community of individuals shares each other's resources to create a cooperative computing environment. Those who are contributing their resources to a common pool can get access to that pool. A sophisticated model can also be employed here for deciding how much resources share contributors can get. It can involve credits that one can earn by sharing resource, which can then be used when needed. A system like Mojonation.net employs this model for storage sharing. This model works when those participating in the Grid have to be both service providers and consumers.

h) Monopoly and oligopoly

Monopoly means a single resource owner dominates the market and set the price. Oligopoly means a small number of resource owners dominate the market and set the price.

Buyya, et al (2005) discuss the economic models that have been implemented in computational Grid environment. Authors give brief remarks on how the systems apply the economic models. The models discussed apply one or more of the following pricing schemes: the flat rate, usage based, smart market, Paris-Metro, per-time, Culumus, priority and the expected capacity pricing scheme. The combination of the pricing schemes and the economic models seeks to determine the fair access price for any given service. Therefore, price management is significant in commercial Grid, and the economic model has influence on the time that is spent in negotiating the price. Some economic models like the bargaining model, the posted price model dwell much on price negotiation.

3.1.2.3 Existing Pricing Models for Service Provisioning Environment

The Posted Price Model is amongst the models that deals more on price negotiations. With regards to minimizing the time spent on price negotiation, Mingbiao, et al (2007), proposed a Posted Price Based Grid Resource Supermarket (GRS) model. The Posted Price Based GRS model consists of the manager of GRS, which gains the profits by serving the Grid resource providers and consumers. The resource consumers share the resources at posted price according to his plan and pocketbook. The resource provider gains income for his resource being shared. The pricing strategy of GRS is a key problem for the GRS manager. Therefore, the profits of GRS depends on the policy of the resource that how much to buy in and how much to sell out. The resource value in GRS is defined as a function of many parameters as follows: Resource Value = Function (Resource Strength, Cost of physical resources, Service overhead, Demand, Value perceived by the user, Preferences). Figure 3.3 is an overview of the GRS based posted price model with its components.

Song, et al, (2007) proposed a competitive pricing model which uses competitive strategy to determine the price thereby, maximizing the utilization rate of the Grid system. Competitive pricing strategy means pricing the service within a competitive market. The authors also considered three variables for determining the price:

i. Quality of product.
ii. Quantity of product sold, and
iii. The products provided by competitors.

In this model, the quality of products becomes its QoS level. The authors explain how the QoS affects the price of the resource and how the price affects the QoS of the

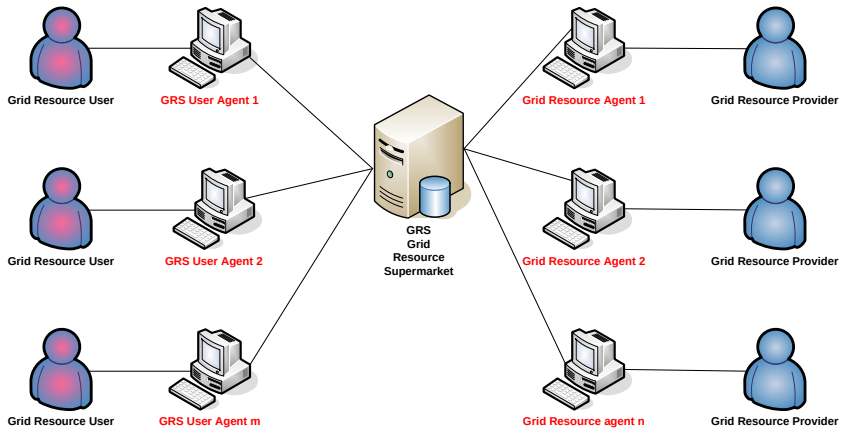

Figure 3. 3: An Extended Posted Price Model with Grid Resource Supermarket (Mingbiao, et al, 2007)

resource. They further argue that price should be determined such that the service providers will be able to recover the cost of producing the service and at the same time maintain reasonable profit level.

Yuan, et al (2005), defined a mechanism that tackled unreasonable resource pricing strategies in market-oriented Grid systems. Therefore, a price-adjusting mechanism that is responsible for adjusting unreasonable access prices of resources is then proposed. This price-adjusting mechanism is based on the supply and demand of a resource, such that, price is adjusted based on the fundamental economic theory of the supply and demand model. It effectively achieves the equilibrium price to promote the supply and demand globally in a more balanced fashion. In Yuan, et al, (2006) the Price Influence Model, the price-adjusting mechanism that was implemented is briefly discussed and furthermore, the dependence of Grid resources are introduced with a suggested potential mechanism that fairly prices and charges those resource.

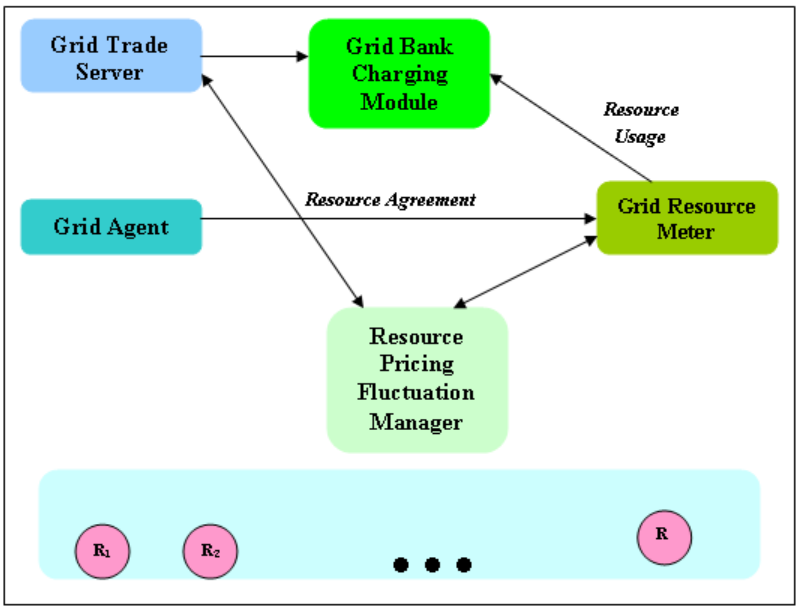

Figure 3. 4: An extended GRACE architecture with RPFM Module (Liu and Xu, 2007)

Liu and Xu (2007) extended Grid Architecture for Computational Economy (GRACE) architecture (Buyya, et al, 2001) by adding the Resources Pricing Fluctuation Manager (RPFM). The RPFM enables both resource consumers and providers to maximize their profits. Figure 3.4 is an overview of the extended GRACE architecture with RPFM module. The extended GRACE architecture is based on the bidding pricing process that is similar to the original GRACE. The RPFM add the more convenient approach of determining the price. It is responsible for the pricing fluctuation of resources and correspondence with Grid Trade Server (GTS) and Grid Trade Manager (GTM). RPFM gets bids information of resources from GTS, gets Resource Usage Records (RURs) information from Grid Resource Meter (GRM), calculates the ratio of resources utilization according to some algorithms, dispatches new pricing to GTS and GRM.

RPFM consists of four components that are integrated to achieve dynamic price fluctuation. Those components are Resource Object, Pricing (PR), Utilization Monitor (UM), and Controllers. These components interact with the GTS and the GRM to handle the price fluctuation based on the external resource usage information.

Zhao, et al (2007) proposed a dynamic price model based on the demand prediction and task classification. The predicted resource demands adjust the resource price according to the results of the demand prediction. In addition, this model can calculate the future price of each task based on demand prediction. Three parts constitute the model and these are: resource demand prediction mechanism, resource price-adjusting mechanism and task pricing mechanism. It applies the Markov chain to predict the future demands of Grid resources and uses a price-adjusting mechanism based on the future demands, which takes into the interdependence of price and demand into consideration. Therefore, the mechanism can balance resource loads and guarantee the profits of resource providers at the same time. In addition, tasks are classified into two categories: exclusive tasks and shared tasks. According to the differences between them, the authors proposed two different task-pricing strategies. Based on these, they introduced a novel task-pricing mechanism, which takes serving time, the workload and the type of the task into consideration at the same time.

3.1.2.3 The State of the Art in Grid Services Pricing

Pricing of Grid services still remains a challenge as Grid became the medium for trading with services on-demand. A current pricing model that provides the mechanism to set a price based on the market demand, supply and QoS level as determinants is the one proposed by Song, et al, (2007). However, it is not fully

suitable for GUISET. The service pricing mechanism for commercial Grid environment must be dynamic so

Table 3. 1: Summary Evaluation of Pricing Models

Pricing Model	Production cost Recovery	Fairness to consumer and provider	Dynamic	Price for QoS
GRS based Posted Price Model (Mangbiao, et al, 2007)	✓	X	X	✓
Competitive Pricing Model(Song, et al, 2007)	✓	✓	✓	✓
Price Influence Model (Yuan, et al, 2006)	X	✓	✓	X
Extended GRACE(Lui and Xu, 2007)	X	X	✓	X
Dynamic Price Model (Zhao, et al, 2007)	X	X	✓	X

✓ Denotes that the pricing model considers the feature.

X Denotes that the pricing model does not consider the feature

as to reflect the current market situation and to be fair to both service providers and consumers. The current pricing approaches do not fully cater for GUISET environment pricing requirement. In Table 3.1 we present the evaluation of the reviewed pricing models against the GUISET pricing model design criteria. Therefore in this study we proposed a customized price service mechanism for GUISET.

3.1.3 Incentives and Charging Models in Distributed Computing

Distributed computing paradigms have become the major network tool for sharing and cooperating of different independent, distributed organization to accomplish their different task (Iyilade, et al, 2007; Hales, 2004). Incentives have been introduced as the mechanism to enforce cooperation amongst the organizations and prevent selfishness (Ip, et al, 2008). Due to the acceptance of different distributed computing

paradigm as the medium for trading resources, charging models have been introduced to manage the economical compensation for service usage to service providers. In this Section, we review the mechanisms that have been used to award incentives and charge service providers and consumers.

3.1.3.1 Incentive Approaches in Distributed Systems

Incentives have been mostly applied in peer-to-peer networks to enforce cooperative file sharing amongst the peers. It is generally agreed that cooperative network performs significantly better than traditional client-server model in supporting large amount of users. Distributed Systems provide an inexpensive platform for applications that require scalability, efficiency and robustness (Ip, et al, 2008). There are two major reasons for incentive in distributed systems; first, it is to prevent the free-riding problem, and second, it motivates the users to cooperate for the success of the systems.

In order to increase the involvement of users to the systems different incentives mechanisms have been proposed. Ip, et. al, (2008) identified two mechanisms, namely the reputation based system and contribution-rewarding mechanism. In the reputation based system, participating users accumulates points to reflect their resource contribution to the distributed system. The authors further identified the following issues that are faced by this mechanism such as how to quantify the user's contribution, how to provide a secure and trusted reputation system to prevent fake reputation. The contribution-rewarding mechanism is based on credit the users for cooperating in the system. The rewards may come from the overall revenue of the cooperative network, by means of service pricing or cost reduction.

Iyilade, et al, (2007), proposed a two fold incentive model. The first part of the model is to credit resource providers for the resource they contribute and for becoming a member of the Grid. Its second part is based on the negotiation of deadline for the execution of job by service consumers. The authors provided the two algorithms for their model, the donor credit allocation and deadline negotiation algorithm. This mechanism is likely to have the same problems that are faced by the reputation based system mechanism and it may cause unnecessary delays on execution of job submitted by service consumers.

Feldman and Chuang, (2005) classify incentive mechanisms into three groups, namely the inherent generosity, monetary payment and reciprocity based schemes. These schemes faces the same issues that we identified by Ip, et al (2008). Obreitar and Nimis (2003) discussed the taxonomy of incentive patterns for peer-to-peer systems, multi-agent systems and ad hoc network. The incentive patterns shown in Figure 3.5 are classified into two groups namely, trust and trade based patterns.

In Grid computing, the issue of incentives is classified into two approaches (Behsaz, et al, 2006): market based and cooperative-based. The market based incentive approach is when service consumers are awarded points for rating each time they consume services deployed in the environment. Therefore, the ratings are used as the foundation of calculating and incentive the consumer may qualify for. Economic models are used to achieve the incentive mechanism. The cooperative based approach is when users' main goal is to achieve organizational virtualization of resources in order to achieve cooperative sharing of resources. This approach is characterized by free riders, users using services without contributing anything (Emmert and Jorns,

2006). In this study, we employed an integrated approach that effectively combines the strength of both the market based and cooperative based incentive approach.

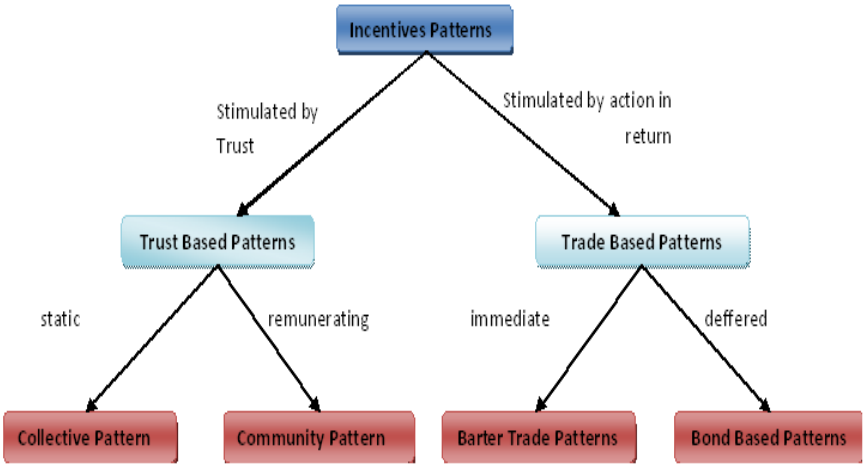

Figure 3. 5: Taxonomy of Incentive Patterns (Obreitar and Nimis, 2003)

3.1.3.2 Charging Models in Grids

Charging of services in a distributed system become more and more important for systems which are utilized commercially (Stiller, et al, 2001). In general a charging model is a quadruple (Q, T, C, U) of the quantity Q, the time T, the quality class C, and the user profile U (Caracas and Altmann, 2007). The quality class C of services allows specifying different quality types. The user profile U, represents history information of user's consumption, the valuation of the user's importance to the business, or special promotions. In this Subsection we will review how charging and incentives have been awarded to service providers and consumers.

3.1.3.3 The State of the Art in Incentives and Charging Models for Grids

The requirements to provide capable and manageable incentive-based charging mechanisms for GUISET infrastructures remain a challenge. Existing incentives mechanisms focus more on preventing selfishness and encouraging cooperation amongst the users of the network. Therefore, they need to be customized in order to meet the requirements of the commercial Grid environment, such as to provide incentives to both service providers and consumers as a tool to encourage them to contribute and utilize services. The charging mechanism should, therefore, be incentive compatible and be customizable to apply different policies (like Service Level Agreements) that may need to be applied in the commercial Grid environment. Current charging mechanisms do not accomplish this requirement. In this study we proposed an incentive-compatible charging mechanism that can be extended to cater for other policies that may need to be applied during the creation of the consumer's bill.

3.2 Summary of the Chapter

This chapter presented a literature survey on how usage accounting, pricing and incentive based charging are conducted in a distributed system environment. It has been discussed that the service demand, supply and QoS level contribute to the pricing of the Grid service, while the usage accounting data forms the basis of the economic compensation of the service consumption as it manages the metered usage data for multi-services. The charging of service remains the exit point of the whole process of achieving an appropriate compensation for Grid service usage. At this stage the incentives or rebates are applied based on user's previous ratings or reputation. This research is about finding the most appropriate usage accounting, pricing and charging model for GUISET with flexibility and non-rigidity as main

design criteria. In view of the current state of the art, our own proposed system architecture for usage accounting, pricing and charging is, therefore, presented in Chapter Four.

CHAPTER FOUR

MODEL DEVELOPMENT

This chapter presents the proposed usage accounting, pricing and charging system architecture for GUISET. Based on the research questions outlined in chapter one (Section 1.3), there is a need to identify the suitable pricing approach for GUISET, map accounting records to charging service such that consumers are charged based on their service usage and incentives awarded to service providers and consumers based on the services contributed to GUISET and consumed from GUISET, respectively. Therefore, our first design criterion is to integrate a pricing approach that uses market demand, supply and QoS as determinants of the market unit price for the services. Thus, Subsection 4.1.1 presents the design criterion for the *Pricing Service*. The second design criterion is to manage robustness in usage accounting such that resource usage records (RUR) from metering systems are mapped accordingly. In Subsection 4.1.2, *Usage Accounting Service* design criteria are outlined. The last design criterion is to make the charging model to be incentive based or compatible. In Subsection 4.1.3 design criteria of *Charging Service* are outlined. Section 4.2 discusses the proposed system architecture.

4.1 Design Criteria for Usage Accounting, Pricing and Charging in GUISET

GUISET environment is envisioned to be the community of a large number of heterogeneous services deployed in different administrative domains. The service providers and consumers may join and leave GUISET dynamically. Service management is, therefore, a challenge as it is hard to manage the capacity and ensure that enough services are available to provide satisfactory QoS to consumers. In this

Section, the design criteria for *service usage accounting, pricing and charging* are outlined to craft the mechanisms that will contribute towards service management.

4.1.1 Custom GUISET Pricing Approach

GUISET as an environment for trading diverse services to different consumers over the Internet, the price of the service is important as it reflects service value. In order to monitor changes in market unit price and to prevent unreasonable profit and under-pricing of services, the following design goals become imperative:

i. *Production cost recovery*: the price should be able to recover the service production costs.
ii. *Fairness to consumer and provider*: the price should be fair for both consumers and providers.
iii. *Dynamic*: the price should be dynamic such that it reflects the market forces.
iv. *Price for QoS*: the price should match the quality level of the service: the higher the QoS level, the high the price.

4.1.2 Managed Robustness in Usage Accounting

Usage accounting forms the basis for economic compensation of the service consumed and provided. We define the following design objectives for usage accounting service in GUISET:

i. *Tracking*: the service usage should be traceable, so that all services consumed are compensated correctly.
ii. *Evaluation*: the service usage should be evaluated through the usage accounting service such that their distribution will be fair to all service providers.

iii. *Fairness*: to enforce fairness on service compensation, the service usage metering records should be arranged according to service providers.

4.1.3 Incentive-Compatible Charging for GUISET

The following design features are crucial to incentive-compatible charging approach for GUISET:

i. *Incentive-compatible*: It must be incentive-compatible in order to encourage service providers and consumers to contribute or consume services respectively.

ii. *Flexibility*: It must be flexible to pricing schemes such that different incentives are applicable.

iii. *Customization*: It must have capability to manage information about the user's profiles and charges data.

4.2 GUISET Usage Accounting, Pricing and Charging System Architecture (GUAPCA)

We now present *GUISET Usage Accounting, Pricing and Charging System Architecture (GUAPCA).* In designing the architecture the following assumptions were taken into consideration:

1. The services are classified into QoS classes: *Guaranteed, Control-load, and Best effort.*

2. The market is competitive and governed by supply and demand. For the supply curve to exist there must be a large number of service providers in the market, and for a demand curve to exist, there must be many consumers.

3. Both service providers and consumers must be price takers and no-one must be a price setter. A price taker cannot influence the price but must take it or leave it.

4. Both service providers and consumers have good information about service qualities and availability.
5. The monitoring system continuously updates the market demand and supply. The market demand (D_x) is basically the total number of units requested (r_i) for service A belonging to class *x*, and market supply (S_x) is the total number of units that providers (s_i) of service A belonging to QoS class *x* (QoS_x) are willing to provide.

$$D_x = \sum_{i=1}^{n} r_i \,, \forall QoS_x \quad x \in \{Quaranteed, Control-load, Best-Effort\}$$

$$S_x = \sum_{i=1}^{m} s_i \,, \forall QoS_x \quad x \in \{Quaranteed, Control-load, Best-Effort\}$$

6. In order to control the change of price, the floor (lower) and ceiling (upper) price limits for a given QoS class are set.

Based on the above mentioned assumptions and design criteria outlined in Section 4.1, we formulated *Usage Accounting, Pricing and Charging System Architecture* for our GUISET research focus. Figure 4.1 shows the *GUISET Usage Accounting, Pricing and Charging System Architecture* (GUAPCA). The GUAPCA is integrated to realize the design criteria outlined in *Subsections 4.1.1 – 4.1.3* through its components. The *Pricing Service* consist of two components namely, *Price Regulator (PREG)* and *Price Recommender (PREC),* has been designed to achieve the design goals outlined in Subsection 4.1.1. The *Accounting Service* has also been designed to target design objectives of Subsection 4.1.2 via components namely *Classifier* and *Correlator* . Finally, the *Charging Service* makes up the *Charging Agent* (CA) and *User Rating Agent* (URA) to meet the design features envisaged in Subsection 4.1.3. Therefore, GUAPCA comprises three main components as services, namely:

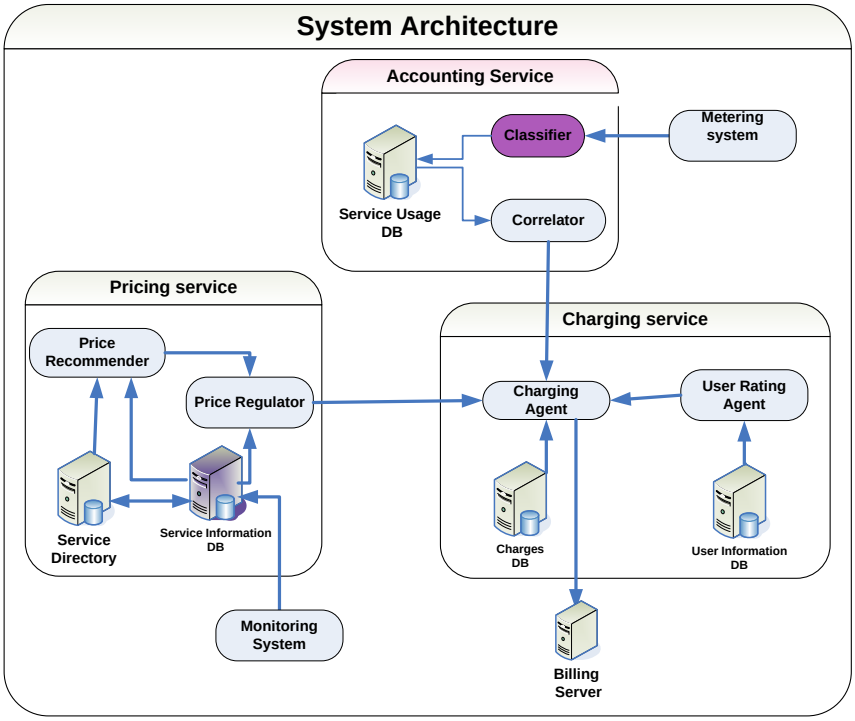

Figure 4. 1: GUISET Usage Accounting, Pricing and Charging System Architecture (GUAPCA) (Buthelezi et .al, 2008)

Usage Accounting Service, *Pricing Service* and *Charging Service*. To enhance a better understanding of GUAPCA, the functionalities of each of the components are explained in details in the next Subsections.

4.2.1 Usage Accounting Service Component

For the purpose of GUISET and in line with earlier stipulated design criteria, the usage accounting service comprises of two sub-components: (i) the *Classifier*, and (ii) the *Correlator*. They have complementary responsibilities to accomplish the design criteria and produce an appropriate accounting record such that the economic-based service usage compensation is achieved. The C*lassifier* is set to arrange the mixed metering data received from different service providers' metering systems to form the

Service Usage Record (SUR) according to the *serviceID* and *customerID*. This is done to enable easy trace of service usage and evaluate service utilization by different service consumers in different time slots. The job or task scheduler, which is not part of this research work, can therefore use that usage analysis data to fairly distribute task to different service providers.

The *SUR* are stored in the *Service Usage Database*. For the purpose of this study, the data that is valuable is the total number of units per service that a particular user has consumed at a certain period, as this will form the economical compensation to the service provider. The *Correlator* sub-component is responsible for the creation of the *consumer-service usage records (CSUR)* for each service consumer. This is done to enable the usage-based charging for the services consumed by the service consumer at the specific period. The *CSUR* is, therefore, forwarded to the *Charging Service* where appropriate policies such as pricing, incentives awarding kick in to produce the *Consumer-Service Usage Bill* (*CSUB*). These components work similar to the one proposed by Agarwal, et al (2003), however in our case the *Correlator* retrieves information from the database and sends it to the *Charging Service* in the form of *CSUB* whereas the *Classifier* arranged the metering data.

4.2.2 Pricing Service Component

Pricing is defined, in this study, as the process of determining the market unit price of any given service base on its market demand, supply and QoS level and to regulate the market unit price. Therefore, our *Pricing Service* component consists of two sub-components: the *Price Recommender (PREC)* and *Price Regulator (PREG)*. The PREC is set to recommend the service market unit price based on the price determinants (market demand, supply and QoS level). Therefore, it holds the *Price*

Adjusting Mechanism (PAM). The PAM employs three strategies to recommend the market unit price for services based on the price determinants. The strategies are: *price decrease*, *price keeping* and *price decrease*. The *PREG* takes the recommended price from the PREC and evaluate it against the price limits to prevent the over-pricing and under-pricing of service unit. In order to achieve price regulation the PREG uses the *Price Controlling Mechanism* (PCM). First, we describe the main strategy used to achieve price recommendation (PAM), then the price regulation strategy (PCM).

4.2.2.1 The Price-adjustment Mechanism (PAM)

The PAM from the PREG adjusts the market unit price of a service based on its market demand, supply and QoS level. The quantity of service units demanded by the consumers in a particular period depends on the market unit price of the service, the market unit price of related service, the capital of the consumer, the QoS preferences of the consumer, and the number of consumers in GUISET.

This relationship is expressed as: $Q_d = f(P_x, P_g, Y, QoS, D_x)$, where,

Q_d = quantity of market demand for service

P_x = market unit price of the service

P_g = market unit prices of the related services

Y = the capital of the consumer

QoS = the QoS preferences of the consumer

D_x = the number of consumers in GUISET

A very important factor that determines the market demand for a service in a particular QoS class is the market unit price of the service. Normally, when the

market unit price of a service increases, quantity of market demand for the service will decrease. However, as the market unit price decreases, the quantity of market demand for the service will increase. The market unit price of the service is influenced by the market unit price of the related services. The related services can be classified into two: substitutes and complements.

Substitute services are services that can be used in the place of another service without lessening a consumer's level of satisfaction. For example, *service B* is substitute service for *service A*, if and only if service B offers same services with service A, but with different QoS level. If the market unit price of substitute service B decreases, the quantity of market demand for the service B usually decreases. The opposite, on the other hand, is also true – as the market unit price of substitute service B increases, the quantity of market demand for the other service increases.

Compliment services are services that are often used jointly. A decrease in the market unit price of a complement service will increase the quantity of market demand. In other words, if the market unit price for a compliment service decreases, the quantity of market demand for the other service will increase. An increase in the capital of a consumer increases the market demand for a service.

A decrease in the capital of a consumer decreases the quantity of market demand for a service. The decline in the QoS and preferences of the consumer for a service will cause a decrease in the quantity of market demand for it. The number of consumers in the market determines number of prospective customers for a particular service. The quantity of service units supplied by the service provider for a particular QoS class in GUISET depends on the market unit price of the service, the input costs (production costs), the market unit price of alternative services and technology.

This relationship is expressed as: $Q_s = f(P_x, P_c, P_a, T_c)$ where,

Q_d = quantity of market supply.

P_x = market unit price of the service.

P_c = input costs (cost of production).

P_a = market unit price of alternative services.

T_c = technology.

The above are factors that influence the quantity of market supply for a service. The relationship between the market unit price of a service and the quantity of market supply thereof is very important in economics. As the market unit price of the service increases, the service provider will be willing to supply a higher quantity of service units. However, if the market unit price decreases, service providers will supply lower quantity. The law of market supply states that given that all other things remain the same, if the market unit price of a service increases, the quantity of market supply thereof will increase; and if the market unit price of a service decreases, the quantity of market supply thereof will decrease. A service provider will only be willing to supply a service to GUISET if it can recover its input costs including the profit it plans to make. Any increase in the factors of production will affect a service provider's input costs.

A technological advance that decreases the input costs is an important factor that can influence the market supply of the service. An improved technology causes an increase in market supply. Any new technology that does not lower the input costs will not be of any value to a service provider thus service provider will not buy or use

such technology. The change of the market unit prices of alternative services supplied by the service provider influence the market supply of the current service.

In order to determine the market the price of a service, *QoS-based Competitive Pricing Algorithm (QCPA)* shown in Figure 4.2 was formulated. The *QCPA* fix the market unit price based on the market demand, supply, and QoS level. Basically, the market demand and supply information is used to decide the market unit price of a given service. The services are classified into three categories namely *Guaranteed, Best-effort, and Control-Load,* according to their QoS level. Therefore, each service class has its own market demand and supply which is used to decide the market unit price for the service. As the algorithm is QoS based, the inputs are the QoS class, input cost price, market demand and supply for services.

In order to determine the *Market Unit Price*, the *Market-unit-Price-Rate-of- Change* (MPRC) is needed. Therefore, the MPRC for each service is calculated based on the market demand and supply for that QoS class to which the service belongs. There are three cases that are analyzed to determine market unit price. The first case is when the market demand is greater than the market supply; therefore *price increase strategy* is applied. In our price increase strategy, MPRC is multiplied by the input price then the product is added to the input price to calculate the market unit price for the service.

The second case is when the market demand is less than the market supply; therefore, the price decrease strategy is applied. In the *price decrease strategy*, MPRC is subtracted from 1, the difference is multiply with the input cost then the product is subtracted from the input price to calculate the market unit price for the service. The last case is when the market unit price is equal to the market supply; in this case the

```
INPUTS    : QoS Class, Input_Price, Market_Demand, Market_Supply

PROCESS:
 For QoS Class

        Market_PricerateofChange = Market_Demand/Market_Supply

         If (Market_Demand > Market_Supply)

             Market_Price =  Input_Price  +  (Input_Price *
                              Market_PricerateofChange)

          else if (Market_Demand <Market_Supply)

             Market_Price = Input_Price  -  (Input_Price * (1 –
                              Market_PricerateofChange))

          else

             Market_Price  = Input_Price

          end if

   End For

 OUTPUT: Market_Price
```

Figure 4. 2: QoS-based Competitive Pricing Algorithm

price keeping strategy is applied. The price keeping strategy makes the input cost price to be the market unit price.

4.2.2.2 Price Controlling Mechanism (PCM)

In order to prevent unreasonable profit and under-pricing of services, for our GUISET, we include the *Price Controlling Mechanism* (PCM) for each QoS class in the PREG component of the *Price Service*. Thus, the price floor and price ceiling (lower and upper price) for each QoS class are set from our GUISET. In order to prevent market unit price intersection, the price limits form borders amongst the QoS classes. Thus, the guaranteed, control-load and best-effort QoS classes price limits will not overlap one another. The guaranteed QoS class holds the highest price limits,

control-load QoS class holds the medium price limits and best-effort QoS class holds the lower price limits.

The *PREG* sub-component in the pricing service component ensures that the market unit price of the service is in between the market price limits of its QoS class. Thus, *PREG* evaluates the market unit price recommended by the *PREC* component against the price limits. Figure 4.3 shows the price evaluation algorithm.

The algorithm has the following variables:

P_{rec} = service recommended market unit price by *PREC*

$\lceil p \rceil$ = QoS class price ceiling

$\lfloor p \rfloor$ = QoS class price floor

$P_{c_{\min}}$ = minimum input costs

$P_{c_{\max}}$ = maximum input costs

These variables are used to evaluate the recommended market unit price for the given service at a particular QoS Class.

The market unit price from the PREG and QoS class are the main inputs for the price evaluation algorithm. For a given QoS class, the predefined price limits for each QoS class are retrieved to regulate the price. There are three cases that are considered to regulate the market unit price. The first case is when the market unit price is less than the price floor. The market unit price is therefore increased by the maximum of the difference of the minimum input cost minus the market unit price, and the price floor minus the market unit price.

Input : P_{rec}, QoS_Class

Process :

For a given QoS_Class

Get $\lceil p \rceil$, $\lfloor p \rfloor$

if ($P_{rec} < \lfloor p \rfloor$)

$\Delta P = \max(P_{c_{\min}} - P_{rec}, \lfloor P \rfloor - P_{rec})$

$P_{rec} = P_{rec} + \Delta P$

else if ($P_{rec} > \lceil p \rceil$)

$\Delta P = \max(P_{rec} - P_{c_{\max}}, P_{rec} - \lceil p \rceil)$

$P_{rec} = P_{rec} - \Delta P$

else

return P_{rec}

End If

End For

Output: Price for Service

Figure 4. 3: Price Evaluation Algorithm

For example, if the market unit price is $2.00, price floor $5.00 price ceiling $25.00, minimum input cost $10.00 and maximum input cost $24.00. The market unit price is less than the price floor, therefore the maximum differences will be $8, and therefore the market will be $8.00 plus $2.00 equal to $10.00.

The second case is when the market unit price is greater than the price ceiling. In this case the market unit price is decreased by the maximum of the difference of the market unit price minus maximum input cost and market unit price minus the price ceiling. For example, For example, if the market unit price is $55.20, price floor $5.00 price ceiling $25.00, minimum input cost $10.00 and maximum input cost $24.00. The market unit price is greater than the price floor, therefore, the maximum differences will $31.20, and therefore, the market will be $55.20 minus $31.20 equal to $24.00.

The last case is when the market unit price is in between the market price limits. It is therefore remain unchanged because it meets the boundaries of the PREG. Therefore, the price evaluation algorithm regulates the market unit price for each service belonging to a certain QoS class based on the recommended market unit price and the price limits.

4.2.3 Charging Service Component

Charging in this study is defined as the process of calculating the final bill that a particular service consumer has accumulated during services consumption at a particular period and applies the relevant incentives if necessary. In order to achieve this in GUISET, we proposed *Charging Service* component in the GUAPCA. The *Charging Service* component is designed according to the design criteria outlined in Subsection 4.1.3. It comprises two major sub-components, that is, (i) The ***User Rating Agent (URA)***, and (ii) ***Charging Agent (CA)***. The *URA* sub-component rates the service consumers based on their usage information from the usage accounting service. The service consumers' profiles are updated each time they consume the services. The points accumulated by the service consumers are used to calculate the

discount they qualify for at that particular time. The discounts are given as incentives to the service consumers.

The *CA* sub-component calculates the consumer's bill based on the *Consumer-Service Usage Record* (CSUR) from the *Usage Accounting Service* Correlator component. Briefly, the total number of service units consumed is multiplied by the market unit price for that particular service QoS class taken from the pricing service. This provides the possibility to apply the usage-based pricing approach, and award incentives and penalties to service consumers and providers respectively for their loyalty and commitments. The following parameters are taken into consideration in calculating the consumer's bill and this relationship is expressed as:

$U_c = f(Q_u, C, U_p, P_c)$, where,

U_c = consumer's bill

Q_u = the quantity of service units consumed at a particular period

C = QoS class

U_p = user profile

P_c = the price of the service in C

In order to calculate the service consumer's bill, *User Charging Algorithm* (UCA) shown in Figure 4.4 is devised. UCA takes the userID and gets all the service usage records (SURs) from the usage accounting service for the given userID at that particular period. It then, gets the market unit prices for all the services listed on the SUR for a given userID. Based on the total number of service units utilized for each service, it is multiplied by the corresponding market unit price. This is done to calculate the user bill before applying incentives.

Input: userID,

Process:

For a given UserID

a) Get all the service usage records for particular period

b) Get all the market unit price for each service in service usage record

c) For each service consumed

c_1) Calculate the total charges

c_2) Calculate the discount based on userID ratings

c_3) Rewards points if applicable for each service consumed based on QoS class

c_4) increment serviceID for current UserID

End For

End for

Output: total service consumer's bill for userID

Figure 4. 4: User Charging Algorithm

The discount to the given userID is calculated for each service utilized using the userID previous ratings. Points are therefore awarded to the userID for each service utilized using appropriate QoS class policies of awarding points. The *CA* component uses it to verify and calculate the consumer-service usage bill (CSUB).

In order to achieve the user rating, management of user profiles and encourage users to contribute and utilize services, the rating of users (consumers and providers) is in two parts. Users are awarded points based on the number of services they contribute in GUISET and the number of service units they have consumed. It is done during the contribution and consumption of services respectively.

URA purpose is to rate the user based on their reputation on the usage of services and the number of services that are contributed. These enable the credibility of incentives to users. In order to achieve this we have proposed the *User Rating Algorithm.*

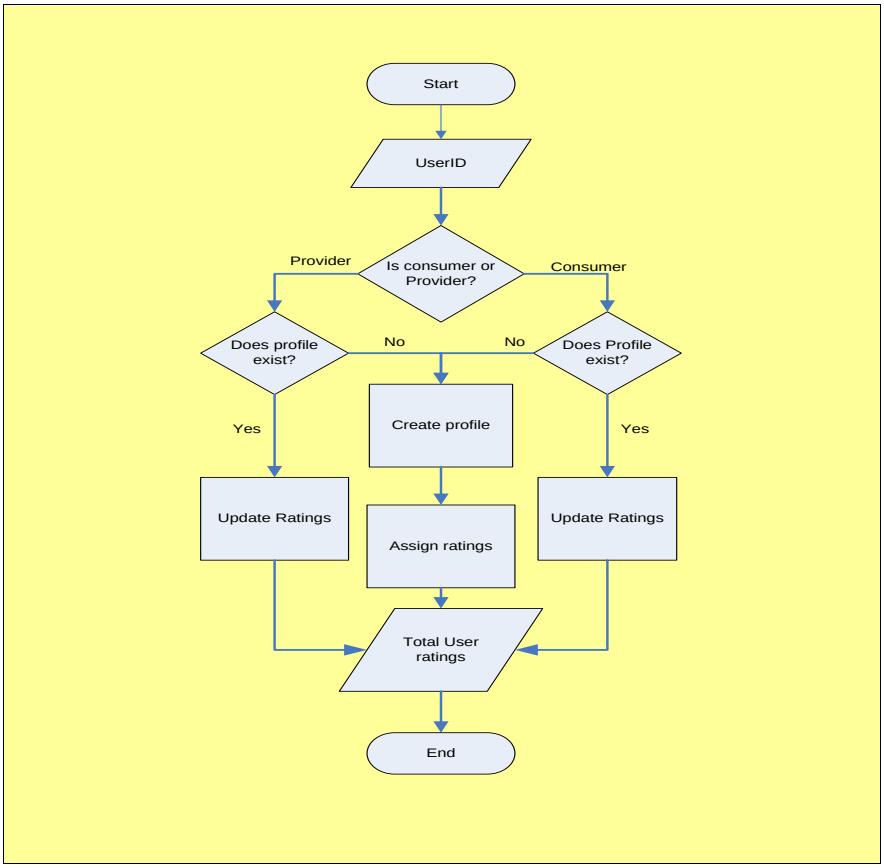

Figure 4. 5: User Rating Algorithm

(UserRA) shown in Figure 4.5. UserRA takes a userID as input and checks if the user at that current time is contributing or utilizing a service. If the user does not have a existing profile, the profile for the user is created and assigned with ratings based on the QoS class policies of ratings. Otherwise, the profile is updated for the given userID.

4.3 Summary of the Chapter

In this chapter, we have presented the design of an integrated system architecture for usage accounting, pricing and charging for GUISET. The design criteria for each

component of the system architecture have been stated and the assumptions that were considered in designing the system architecture are listed. The discussions for each component's functionalities are also discussed.

In chapter five, the simulation environment for the purpose of the evaluation of the system architecture performance is described and the results are analyzed.

CHAPTER FIVE

SIMULATION AND RESULTS ANALYSIS

In chapter four, we presented the GUAPCA model design for GUISET. In this chapter, we focus on the simulation experiment carried out to evaluate GUAPCA. Specifically, Section 5.1 describes the simulation environment. In Section 5.2, the simulation experiments are described together with the results obtained during simulation. Section 5.3 is the summary of the chapter.

5.1 Description of the Simulation Environment

5.1.1 Simulation Setup

Our GUISET-based service provisioning environment reflects the market structure of perfect competition. Therefore, our simulation environment was designed to form a perfect competitive market. It is based on the idea that no single service provider has influence on the price of the service it sells. There are many consumers and service providers. Each provider supplies a number of units for the service in the particular QoS class and consumers request a number of units for the service in the particular QoS class. Therefore, the sum of units requested and supplied in the particular QoS class form our simulation quantity of market demand and supply, respectively. Consumers and providers are also at liberty to enter and leave the environment at any time.

The consumers utilize the service as a utility, so the market demand is expected to change faster compared to the market supply. The pricing service component

recommends the market unit price of the service based on the quantity of market demand and supply at that particular period. It further regulates the market unit price against the preset market price limits. The usage accounting service maintains usage metering data from service providers' sites. This helps to provide the correct data about usage of services by consumers. The service consumers and providers are awarded points for respectively utilizing and contributing services to the market. The points awarded become the ratings of the users. The ratings are then used to calculate the rebates to be given to the consumers when calculating the final consumer's bill.

The simulator was implemented using Netbeans 6.1 IDE (Integrated Development Environment) for Java with Java Development Kit version 1.5 (JDK 1.5). The underlying database was implemented using MySQL 5.0. The default values and range of parameters that were considered for the simulation in this study are presented in *Table 5.1*, with their descriptions.

Parameter Name	Description	QoS Class	Value

Service Demand	Quantity of units demanded per QoS class in particular period for a single consumer.	*Best-Effort*	[1, 20]
		Control-Load	[1, 20]
		Guaranteed	[1, 20]
Service Supply	Quantity of units supplied per QoS class in particular period for a single provider.	*Best-Effort*	[1, 5]
		Control-Load	[1, 5]
		Guaranteed	[1, 5]
Input Costs	Service production costs per QoS class for a single provider.	*Best-Effort*	[$5.00, 25.00]
		Control-Load	[$26.00, 45.00]
		Guaranteed	[$46.00, 65.00]
Points	Total number of points that are awarded per service unit in a particular QoS class.	*Best-Effort*	0
		Control-Load	1
		Guaranteed	2
Market Price limits	The range that the market unit price should be in between.	*Best-Effort*	[$10.00, 20.00]
		Control-Load	[$30.00, 40.00]
		Guarantee	[$50.00, 60.00]
Period		*All*	[1, 9]

Table 5. 1: Parameters and their default values for the simulation

5.1.2 Performance Analysis

In testing the performance of GUAPCA, the following metrics were used:

i. *Efficiency* –This refers to the effectiveness of the pricing service component in reacting to different market situation to recommend and control the market unit price.

ii. *Fairness* – This refers to a state when the pricing service component recommends the market unit price for the service based on the quantities of market demand and supply. In economic terms, for the competitive market approach, the market unit price is fair when it is determined based on the quantities of market demand and supply. It also refers to a state where the charging component awards incentives to users based on their previous profiles.

5.2 Simulation Experiments

This Section presents the simulation experiments and results that were obtained. Each experiment was conducted in order to observe the behavior of the usage accounting, pricing and charging service components of GUAPCA system. The usage accounting service classifies and correlates the data generated as metering data for services usage. This is inline with the design criteria outlined in chapter four (Section 4.1.2).The service usage data is used for the experiments. *Experiment I* in Subsection 5.3.1 was conducted to evaluate the *efficiency* and *fairness* of the pricing service component when *recommending the market unit price* of a service based on the price determinants (market demand, supply and QoS level) and to regulate the market unit price against the market price limits. Furthermore, to meet the design criteria outline in chapter four (Subsection 4.1.1) *Experiment II* in Subsection 5.3.2 was conducted to test the *fairness* of the charging service component in giving rebates to consumers based on their previous ratings and meet the design criteria outlined in Chapter Four (Subsection 4.1.3).

5.2.1 Experiment I: Market Forces and Price Controls

The aim of this experiment was to investigate the performance of *Price Recommender (PREC)* and *Price Regulator (PREG)* in different market situations. The design was to test whether our price adjusting mechanism conforms to the standard micro-economics demand and supply concepts. The law of demand states that if supply is held constant, an increase in demand leads to increased market unit price, while a decrease in demand leads to a decrease in market unit price. Additionally, to test whether the price controlling mechanism was effective in regulating the market unit

price for the service against the market price limits for the particular QoS class, the values of market demand and supply were generated randomly.

(a) The effect of market forces on market unit price

The market forces that we considered in our experiment are the quantity of market demanded and supplied together with the input costs. The input cost becomes the initial market unit price of the service. The values of the market demand and supply affect the market unit price of the service. In order to recommend the market unit price in our price adjusting mechanism, we defined the *market unit price rate of change* (MPRC) in Chapter Four (Subsection 4.2.2).

Definition 1: The market unit price rate of change based on the quantity of market demand and supply:

$$\text{MPRC} = \frac{market\_demand}{\text{market\_supply}} \quad (1)$$

Definition 2: If the quantity of market demand is greater than the quantity of market supply, the market unit price is calculated using the following formula:

$$market\_price = input\_costs + (input\_costs \times MPRC) \quad (2)$$

Definition 3: If the quantity of market demand is less than the market supply, the market unit price is determined by:

$$market\_price = input\_costs - (input\_costs \times (1 - MPRC)) \quad (3)$$

When the quantity of market demanded for the service increases in the particular QoS class and the quantity of market supplied remains constant, the market unit price of service increases. The decrease in quantity of market demanded was compensated for by the decrease in the market unit price for that service. But as the quantity of market demand and supply becomes equal, the market unit price remains unchanged, implying that the market is at its equilibrium point. At equilibrium, the excess quantity of market demand becomes zero, therefore, there is no variation in market unit price.

The experiment was conducted using different QoS classes. The results obtained are shown in *Figures 5.1 – 5.3*. We noticed that as market unit price increases, the quantity of market demanded fell and as market unit price decreases the quantity of market demanded rose which shows that our pricing strategy conforms to the standard law of demand and supply. For instance, in Figure 5.1 at periods 3 and 8, the quantity of market demanded was less than the quantity of market supplied, therefore, the market unit price was decreased to attract more consumers and as a result, the quantity of market demand rose steadily again.

Also, in Figure 5.2 during periods 1 and 6, the quantities of market demanded and supplied were equal; therefore, the market unit price of the service remained unchanged. When the quantity of market demand was above the quantity of market supply, the market unit price was increased by our PREC and the quantity of market demanded fell. In Figure 5.3, during period 1, the over-demand state of the service resulted in a rise in the market unit price to $192.00 and the quantity of market demanded dropped.

Table 5. 2: Simulation parameters for Best-Effort QoS Class in Experiment I (a)

Input Data				Results
Period	**Market Demand**	**Market Supply**	**Inputs Costs ($)**	**Market unit price ($)**
1	11	10	10.00	21.00
2	15	10	10.00	25.00
3	2	10	10.00	2.00
4	20	10	10.00	30.00
5	10	10	24.00	24.00
6	13	10	24.00	55.20
7	15	10	23.00	57.50
8	2	10	23.00	4.60
9	12	10	23.00	50.60

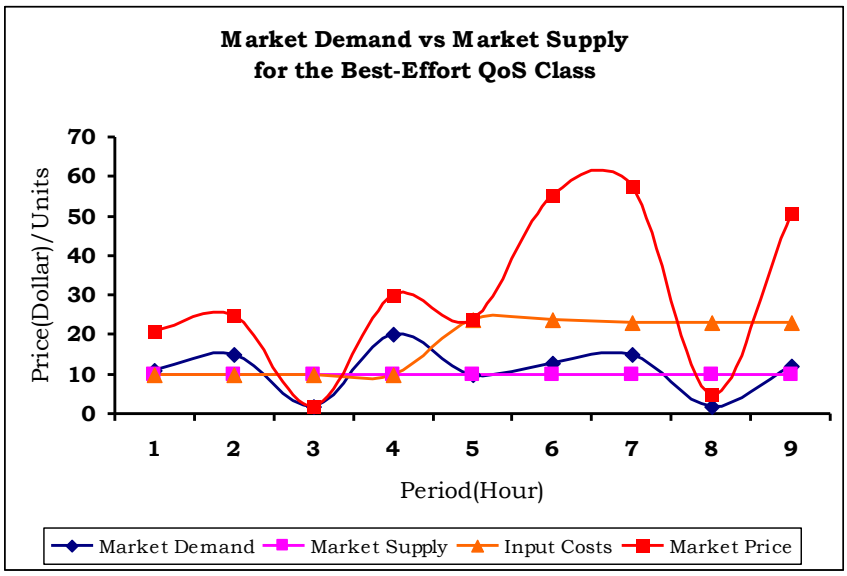

Figure 5. 1: Market Demand versus Market Supply for Best-Effort QoS Class

Table 5. 3: Simulation parameters for Control-Load QoS Class in Experiment I (a)

Input Data				Results
Period	**Market Demand**	**Market Supply**	**Input costs ($)**	**Market unit price ($)**
1	6	6	30.00	30.00
2	12	6	30.00	90.00
3	1	6	30.00	5.00
4	18	6	30.00	120.00
5	16	6	33.00	121.00
6	6	6	33.00	33.00
7	3	6	27.00	17.00
8	8	6	27.00	63.00
9	26	6	27.00	144.00

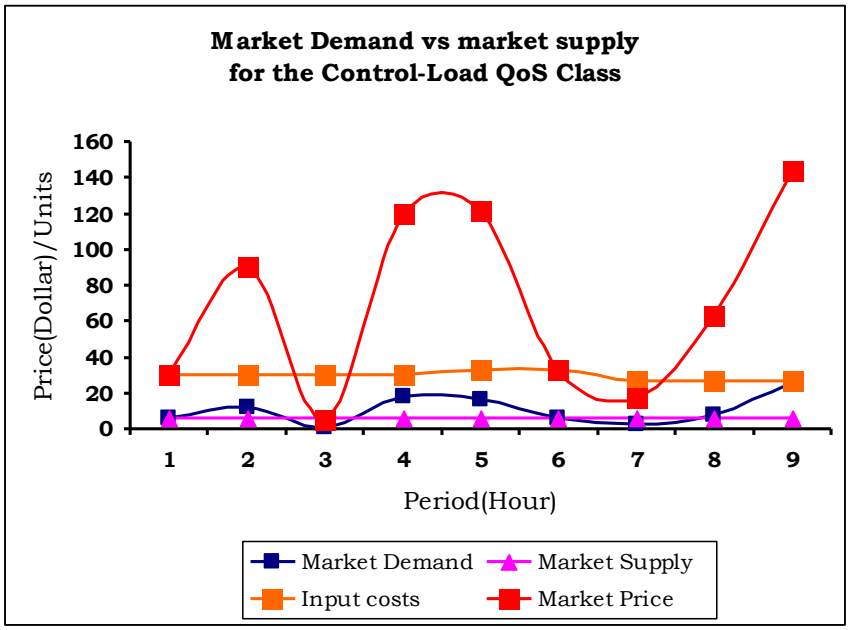

Figure 5. 2: Market Demand versus Market Supply for Control-Load QoS Class

Table 5. 4: Simulation data for Guaranteed QoS Class in Experiment I (a)

Input Data				Results
Period	**Market Demand**	**Market Supply**	**Input Costs ($)**	**Market unit price ($)**
1	36	16	59.00	192.00
2	3	16	59.00	11.10
3	4	16	59.00	148.00
4	12	16	48.00	36.00
5	17	16	48.00	99.00
6	19	16	48.00	105.00
7	18	16	47.00	99.90
8	15	16	47.00	44.10
9	3	16	47.00	8.81

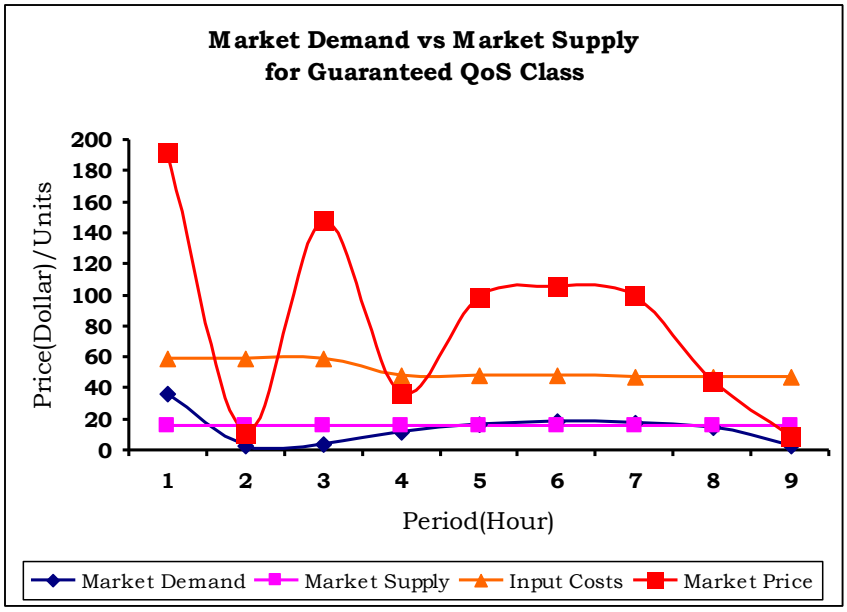

Figure 5. 3: Market Demand versus Market Supply for Guaranteed QoS Class

(b) The effect of price controls on market unit price

In the experiment described in Section 5.3.1(a), the market unit prices were obtained based on the quantity of market demand and supply for each QoS Class. In this experiment, we test the *effectiveness* of our price control mechanism in regulating the recommended market unit price against the market price limits. The price ceiling assumed imposed market price limit on how high a market unit price can be set on a service by GUISET authority. It is set to protect consumers from conditions that could make services inaccessible and prevent providers from over-pricing the services they render. Meanwhile, price floor is an imposed market price limit on how low a market unit price can be charged for a service. It is set to protect the supplier from under-pricing the services they render. In order to obtain the *Regulated Market unit price* (RMP) from our price controlling mechanism, the following formulas were defined.

Definition 4: In situations where the market unit price is greater than the price ceiling, the RMP is calculated using the following formula:

$$RMP = market\_price - MAX\begin{pmatrix} market\_price - input\_costs_{\max}, \\ market\_price - price\_ceiling \end{pmatrix} \quad (4)$$

Definition 5*:* In a situation where, the market unit price is less than the price floor, the RMP is calculated using the following formula:

$$RMP = market\_price + MAX\begin{pmatrix} input\_costs_{\min} - market\_price, \\ price\_floor - market\_price \end{pmatrix} \quad (5)$$

In order to test the *efficiency* of our pricing service component in regulating the market unit price against market price limits, we conducted three tests one each for the three different QoS classes. The market price limits for each QoS class were defined in *Table 5.1*.

As the market price limits are set to control the market unit price of the service, in situations where the market unit prices are greater than the price ceiling they are decreased to the price ceiling or lower depending on the MRC. The decrease of the market unit price result to an over-demand situation, therefore, mechanisms to distribute available services to consumers are needed. On the other hand, the market unit price floor is set to protect the providers from low market unit price. In a situation where the market unit price is lower than the price floor, the market unit price is increased to the price floor or above depending on the MRC. These result into a situation where there is oversupply of services. *Figures 5.4 – 5.6* show the graphical presentation of the results that were obtained from our simulation.

We noticed that as market unit price increases above the price ceiling or decreases below the price floor, the market unit price was reduced or raised to be within the market price limits, respectively. For instance, in Figure 5.4, at periods 3 and 8, the market unit prices for the service were raised because they are below the price floor. In other periods the market unit prices were above the price ceiling; therefore they were reduced.

In Figure 5.5 during periods 1 and 6 the market unit prices were within the price limits for the service; therefore, it was not changed. In periods 3 and 7, the market unit prices were increased, as they were lower than the price floor. Whereas in other

period the market unit prices were reduced by our *PREG* as they were above the price ceiling.

In Figure 5.6, during periods 1, 3, 5, 6, and 7 the market unit prices were higher than the price ceiling; therefore they were reduced to be in between the market price limits. In the other periods, the market unit prices were below the price floor; as a result they were increased to be in between the market price limits.

The observed market unit prices adjust at any time to the market price limits; therefore, we concluded that our price control mechanism was *efficient* in controlling the market unit price in different situations as justified by the results.

Table 5. 5: Simulation Data for Best-Effort QoS Class for Experiment I (b)

Input Data			Results
Period	**Input Costs($)**	**Recommended Market Unit Price($)**	**Regulated Market Unit Price($)**
1	10.00	21.00	20.00
2	10.00	25.00	20.00
3	10.00	2.00	10.00
4	10.00	30.00	20.00
5	24.00	24.00	20.00
6	24.00	55.20	20.00
7	23.00	57.50	20.00
8	23.00	4.60	10.00
9	23.00	50.60	20.00

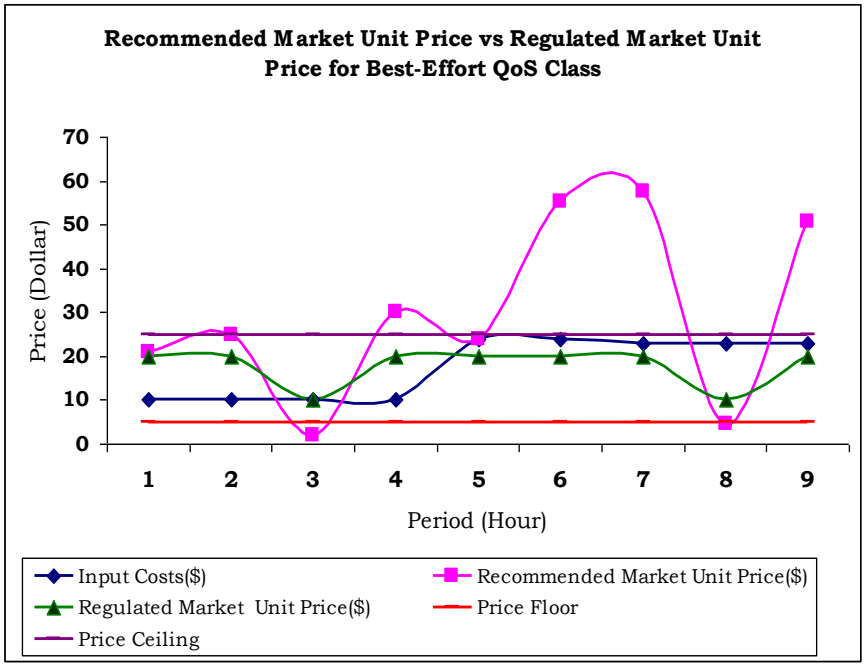

Figure 5. 4: Recommended Market Unit Price versus Regulated Market Unit Price for Best-Effort QoS Class

Table 5. 6: Simulation Data for Control-Load QoS Class for Experiment I (b)

Input Data			Results
Period	**Input Costs($)**	**Recommended Market Unit Price($)**	**Regulated Market unit price($)**
1	30.00	30.00	30.00
2	30.00	90.00	33.00
3	30.00	5.00	30.00
4	30.00	120.00	33.00
5	33.00	121.00	33.00
6	33.00	33.00	33.00
7	27.00	17.00	30.00
8	27.00	63.00	33.00
9	27.00	144.00	33.00

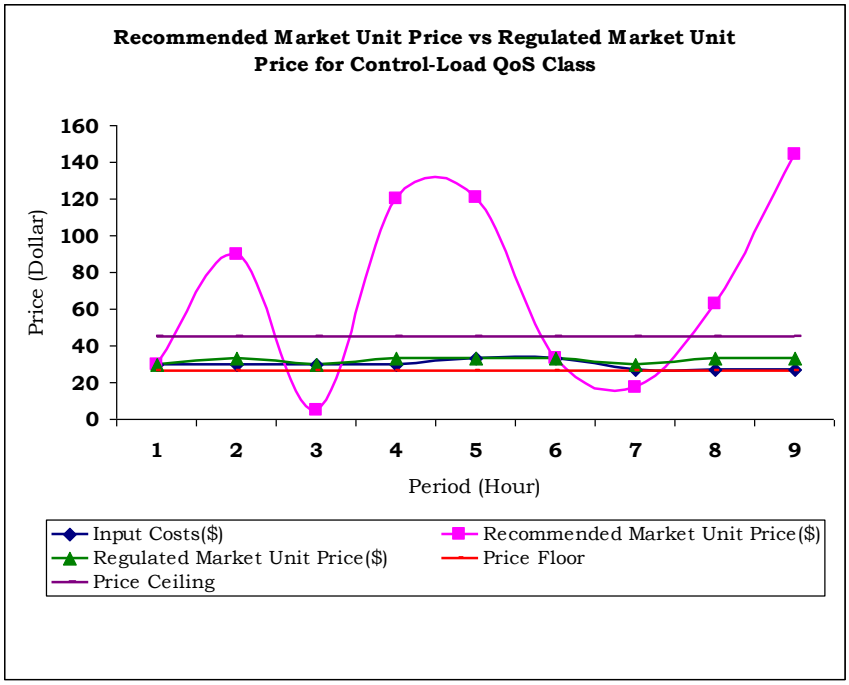

Figure 5. 5: Recommended Market Unit Price versus Regulated Market Unit Price for Control-Load QoS Class

Table 5. 7: Simulation Data for Guaranteed QoS Class for Experiment I (b)

Input Data			Results
Period	**Input Costs($)**	**Recommended Market Unit Price($)**	**Regulated Market unit price($)**
1	59.00	192.00	60.00
2	59.00	11.10	50.00
3	59.00	148.00	50.00
4	48.00	36.00	50.00
5	48.00	99.00	60.00
6	48.00	105.00	60.00
7	47.00	99.90	60.00
8	47.00	44.10	50.00
9	47.00	8.81	50.00

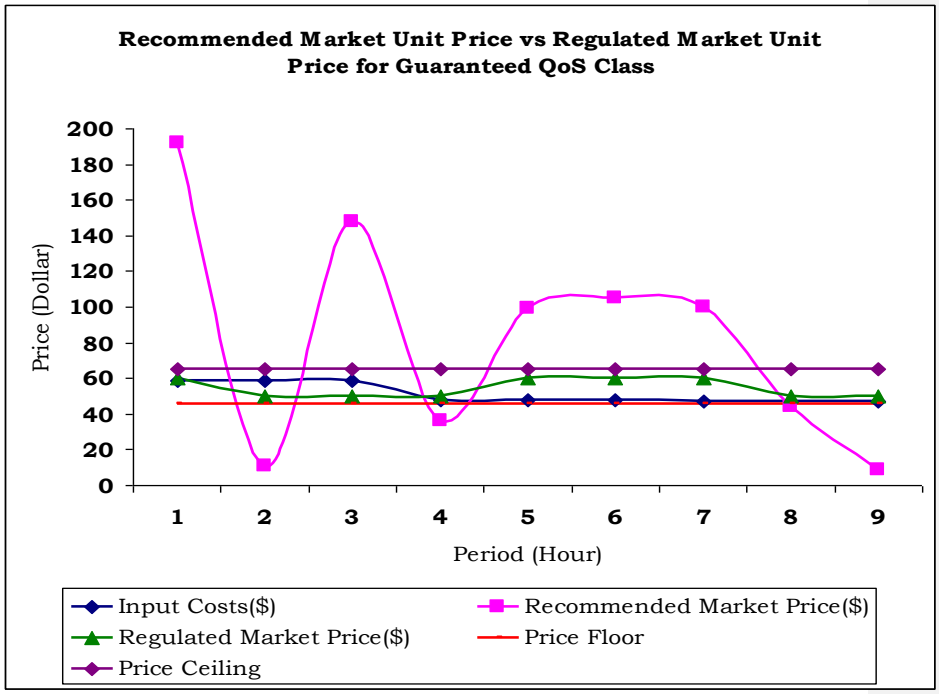

Figure 5. 6: Recommended Market Unit Price versus Regulated Market Unit Price for the Guaranteed QoS Class

5.2.2 Experiment II: Effect of Incentive Based Charging

Incentives were employed to encourage service providers and customers to provide and use the services. In this case, the service providers are awarded points for services they contributed in the environment, with expectation that they would not leave the environment but eventually become consumers of other services too. A consumer is awarded a point for each service he or she consumes in the environment. We assume that the profiles of both service providers and consumers are available based on their usage history. For service providers, we assume that a Service Evaluation Module (SEM) which evaluates the services that the service provider intends to render and is assigned to an appropriate QoS Class, which in turn award the service provider with points based on the number of units that are to be provided.

The consumer is expected to pay lesser amount for using services compared to the actual amount if he has gained enough point to qualify for rebates. This implies that the customers would save some amount to be utilized later for other services. We viewed this as a mechanism to encourage both customers and service providers to utilize and contribute services to the environment. *Figures 5.7 – 5.9*, shows the graphical presentation of the total amount before and after discount that the customer owe. *Table 5.8* present the range of discounts that customer may qualify for based on the point gained before. Therefore, this is used to calculate discount amount for the customer in a particular.

In Figure 5.7, we observed that the total amount and discount that the consumer received depended greatly on the total number of units of service consumed and the points awarded previously. The quantity of units of the service consumed determined

the total amount before discount that the customer owed. The discount amount is calculated based on previous points and is below the total amount before discount.

We further observed that in Figure 5.8; the total amount and discount amount differed when compared to Figure 5.7. The services, respectively rendered in this QoS class carried more points than those in Figure 5.7; therefore, customers were gaining more points for each unit of a service consumed. Thus, the discounts given to them were greater or equal to 10 percent.

In Figure 5.9, the services rendered in this QoS class carried the highest points per service unit consumed. Therefore, the observation is that, the more the consumer utilized the services, the more points were gained and invariable, the discounts became 20 percents. The decrease in service consumption resulted in the decrease in points awarded to the consumers. In all, we concluded that our charging approach is *efficient,* and *fair* in awarding points and discounts to a customer, as this was usage based. The trend that was observed in the results of our experiment below clearly shows that our charging approach was incentive-compatible.

Table 5. 8: Customer Ratings and Discounts

Rating(Points)	Discount(Percentage)
5	5
10	10
50	20

Table 5. 9: Simulation Data for Best-Effort QoS Class for Experiment II

Period (Hour)	User Rating(Points)	Total amount($)	Discount Amount($)	Market Unit Price ($)	Usage Units
1	87	20.00	13.08	20.00	1
2	19	260.00	222.30	20.00	13
3	83	70.00	47.88	10.00	7
4	9	300.00	285.00	20.00	15
5	45	80.00	68.40	20.00	4
6	35	80.00	68.40	20.00	4
7	124	110.00	75.24	20.00	5.5
8	24	240.00	205.20	10.00	24
9	25	400.00	342.00	20.00	20

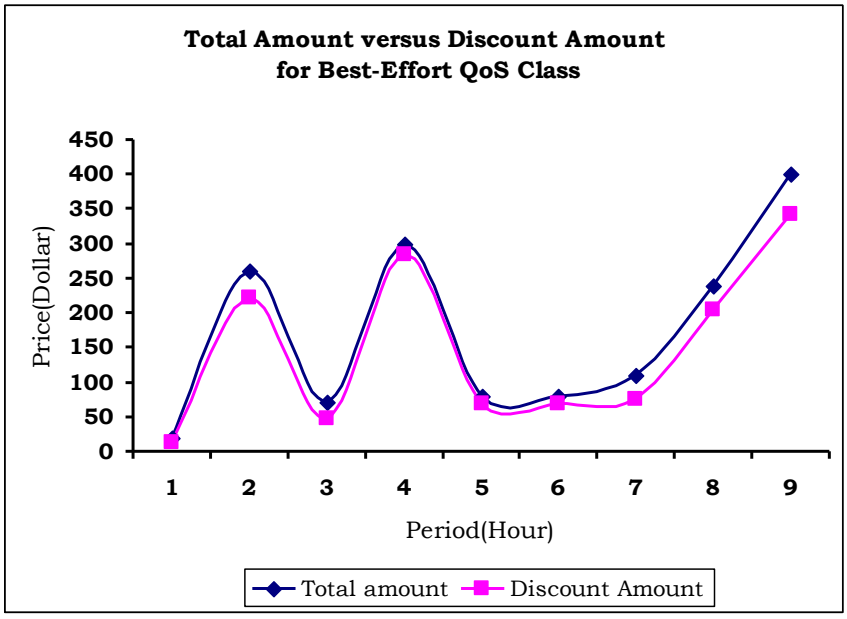

Figure 5. 7: Total amount versus discount amount for Best-Effort QoS Class

Table 5. 10: Simulation Data for Control-Load QoS Class for Experiment II

Period (Hour)	User Rating(Points)	Total amount($)	Discount Amount($)	Market unit price ($)	Usage Units
1	192	528.00	422.40	30.00	176
2	182	660.00	528.00	33.00	20
3	38	180.00	162.00	30.00	7
4	133	90.00	72.00	33.00	2.73
5	29	396.00	356.40	33.00	12
6	193	30.00	24.00	33.00	0.91
7	208	264.00	211.20	30.00	8.8
8	34	198.00	178.20	33.00	6
9	144	180.00	144.00	33.00	45

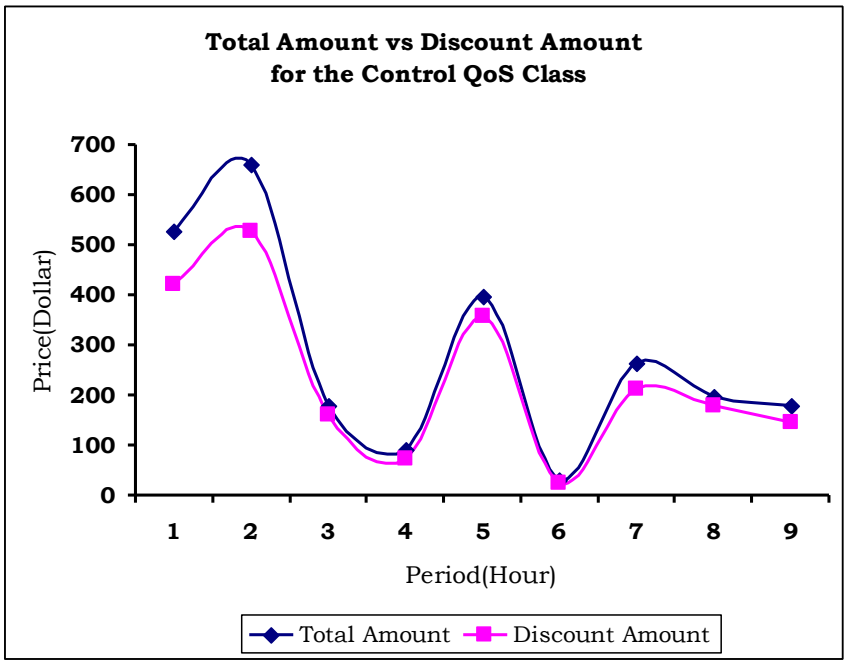

Figure 5. 8: Total amount versus discount amount for the Control QoS Class

Table 5. 11: Simulation Data for Guaranteed QoS Class for Experiment II

Period (Hour)	User Rating(Points)	Total amount($)	Discount Amount($)	Market unit price ($)	Usage Units
1	222	1080.00	864.00	50.00	21.6
2	182	200.00	160.00	50.00	4
3	137	500.00	400.00	50.00	10
4	144	150.00	120.00	60.00	2.5
5	174	1140.00	912.00	60.00	19
6	160	1200.00	960.00	60.00	20
7	112	200.00	160.00	50.00	4
8	12	250.00	225.00	50.00	5
9	9	1140.00	1083.00	50.00	19

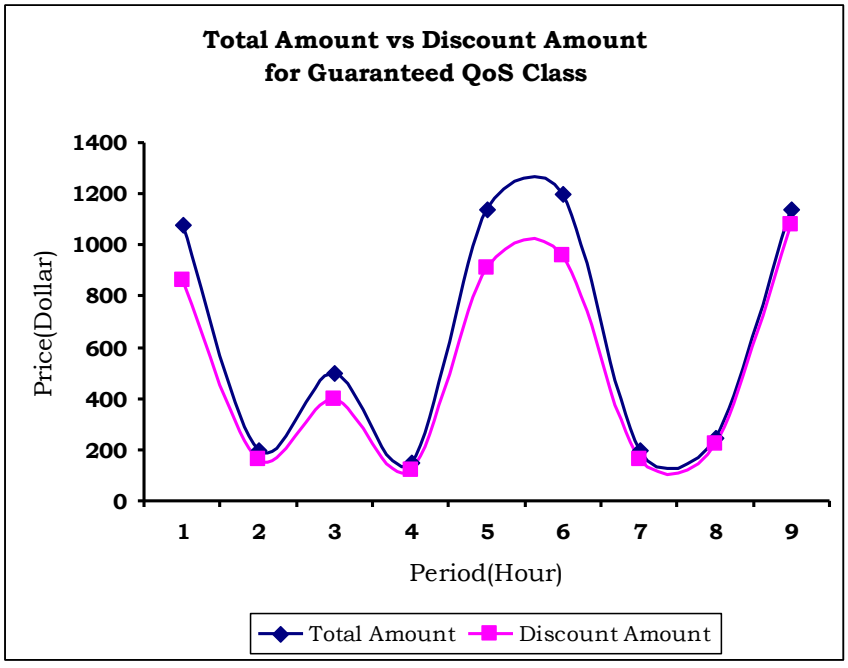

Figure 5. 9: Total amount versus discount amount for Guaranteed QoS Class

5.3 Summary of the Chapter

In this chapter, we have presented the simulation results of the GUAPCA system. The QoS classes were introduced as a mechanism to categorize the service deployed in GUISET. The results obtained show that our price adjusting mechanism conformed to the micro-economics principles of determining the market unit price based on the market demand and supply. For example, when quantity supplied was 6 units and quantity demanded was 1 unit, the market unit price was decreased from $30.00 to $5.00 thereby encouraging more consumers for the service and when the market unit price was $148.00 above the price ceiling of $60.00, it was decreased to $50.00. Therefore, we conclude that our price adjusting and controlling mechanism is effective, and fair in adjusting and controlling the market unit price for the service in a particular QoS class. Thus, our approaches successfully meet the performance metrics defined.

Our charging and usage accounting service is fair in awarding points and supplying usage information for the purpose of charging the customer based on the usage and rewarded credits.

Commented [M2]: Rework these.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

In this chapter, we present the conclusion on the research questions that were presented in this dissertation. Furthermore, the chapter identified the issues that needed to be addressed in a future improvement on this work.

6.1 Conclusions

In this research, an attempt had been made to address the issues of usage accounting, pricing and charging in GUISET Grid environment by developing GUAPCA. Three research questions were identified. The first research question was: which pricing scheme is suitable for GUISET Grid environment such that SMMEs can affordably have access to IT services provided? The competitiveness and dynamism of the Grid-based service provisioning environment required a demand and supply price determining approach, as it is fair to both service provider and consumer. The market unit price limits was introduced to prevent over-pricing and under-pricing of the services.

The second research question was: How are the usage accounting-records mapped and supplied to the charging service component in GUISET? As the usage accounting-records forms the basis of economic compensation for service usage and rebates, we followed a two stage approach of arranging usage data from metering service, the first stage arranged the data from metering service according to service provider's identity, and the second stage arrange the metering data according to consumer's identity.

The third research question was: how can incentives be awarded to users (provider and consumers) in our GUISET? This research question was answered by following a two-mode incentive approach; service providers were awarded points for contributing services and consumers equally awarded points for utilizing service in the environment. Once the points have reached a certain number are changed to be rebates that can be given to users when consuming services

In this research work, three objectives were set in order to realize the goal, the objectives are: (1) conduct an investigation on how accounting, pricing and charging are managed in a Grid environment, (2) emulate existing knowledge and develop integrated system architecture for usage accounting, pricing and charging in GUISET, and, (3) simulate and evaluate the developed system architecture as proof of concepts.

In this dissertation, the literature survey conducted was presented in chapter two. This was done to accomplish the first objective of this research. Therefore, the research efforts presented in chapter two resulted to the integration of the usage accounting, pricing and charging system architecture for GUISET (GUAPCA). The design criteria for usage accounting, pricing and charging services were derived from the work that others have done towards solving the issues of usage accounting, pricing and charging in a multi-service heterogeneous, Grid-based service provisioning environment.

The second objective of this research was achieved by emulating existing knowledge to integrate the GUAPCA. The detailed description of GUAPCA and functionalities of its components were presented in chapter three. The competitive market approach was adopted for determining the market unit price for services. The adoption was motivated by the behavior of service providers and consumers in the Grid-based service provisioning environment, which reflect the competitive market approach.

Based on the description of the GUAPCA, the simulation was implemented to evaluate its performance. The results obtained from the simulation show that, overall the usage accounting, pricing and charging components for GUAPCA are suitable for the GUISET. Thus, third objective of the research was achieved.

6.2 Future Work

The results obtained from the simulations showed suitability of GUAPCA for real-time environment. However, the simulations are only estimation of the reality. Therefore, the viability of GUAPCA still needs to be tested in the real-life environment. In the evaluation and simulation of GUAPCA other factors that may contribute to changing or fixing the market unit price of the service (such as prices of substitute and complementary services) were ignored. In the future, we would like to see how the pricing service component will behave when those factors are taken into consideration. The issues of security have been ignored in this study; therefore, the expansion of the study should look at the security mechanisms that can be used to prevent the usage data from being faked or forged by providers or consumers for self indulgence. Furthermore, issues of service allocation in situation where there is over-demand need to be addressed such that customers are treated with fairness.

BIBLIOGRAPHY

Adigun, M.O. Emuoyibofarhe, O.J. Migiro, S.O. (2006). Challenges to Access and Opportunity to use SMME enabling Technologies in Africa, a presentation at 1st All Africa Technology Diffusion Conference, June 14 – 16, Johannesburg, South Africa.

Afgan, E. Bangalore, P. (2007). Computation Cost in Grid Computing Environments, In Proceedings of the 29th International Conference on Software Engineering Workshops, Page(s):9-12.

Agarwal, V. Karnik, N. Kumar, A. (2003). Metering and Accounting for Composite e-Services, In Proceedings of the IEEE international conference on E-Commerce, Page (s):35-39.

Al-Ali, R. Rana, O. Walker, D. (2003). G-QoSM: A Framework for Quality of Service Management, available online at: http://www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/139.pdf, last accessed on 30 November 2008.

Anerousis, N. and Mohindra, A. (2006). The Software-as-a-Service Model for Model and Ubiquitous Computing Environments, In Proceedings of Third IEEE Annual International Conference on Networking & Services, Page(s):1-6.

Barmouta, A. and Buyya, R.(2003). GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration, In Proceedings of the 17th International Symposium on Parallel and Distributed, Page(s): 245.1

Behsaz, B. Jaferian, P. Meybodi, M.R. (2006). Comparison of Global Computing with Grid Computing, In Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, Page(s): 531-534.

Blefari-Melazzi, N. Di Sorte, D. Reali, G. (2002). Usage-based Pricing Law to Charge IP Network Services with performance Guarantees, In proceedings of IEEE conference on Communications, Page(s):2652 –2656.

Blefari-Melazzi, N. Di Sorte, D. Reali G. (2003). Accounting and Pricing: a forecast of the scenario of the next generation Internet, Computer Communications, Vol. 26, Page(s): 2037-51.

Buthelezi, M.E. Adigun, M.O. Ekabua, O.O. Iyilade, J.S. (2008). Accounting, Pricing and Charging Service Models for a GUISET Grid-Based Service Provisioning Environment, In proceeding of The 2008 International conference on E-learning, E-business, Enterprise Information system, and E-government, Page(s) 350 - 355.

Buyya, R. Abramson, D. Giddy, J. (2001). A Case for Economy Grid Architecture for Service Oriented Grid Computing, available online at: http://www.buyya.com/papers/ecogrid.pdf , last accessed on 30 November 2008.

Buyya, R. Abramson, D. Venugopal, S. (2005). The Grid Economy, In proceedings of *the IEEE*, Page(s): 698-714, ISSN: 0018-9219.

Carcas, A. Altmann, J. (2007). A Pricing Information Service for Grid Computing, In Proceedings of the 5th international workshop on Middleware for grid computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference, ISBN:978-1-59593-944-9.

Chowdhury, H.R. (2006). Internet Pricing, available online at : www. tml. tkk. fi/ Publications/C/21/**Chowdhury**_ready.pdf, last accessed on: 27 November 2008.

Condor Project, http://www.cs.wisc.edu/condor/

Czajkowski, K. Foster,I. Kesselman, C. Tuecke, S. (2004). Grid Service Level Agreements Grid Resource Management with Intermediaries, Grid resource management: state of the art and future trends Page(s): 119 – 134, ISBN: 1-4020-7575-8, available online at: ftp://info.mcs.anl.gov/pub/tech_reports/reports/P1078.pdf, last accessed on 14 November 2008.

Dimitrakos ,T. Mac Dollaral , D. Yuan ,F. Gaeta ,M. Laria ,G. Ritrovato , P. Serhan , B. Wesner , S. Wulf, K. (2003). An Emerging Architecture Enabling Grid Based Application Service Provision, In Proceedings of the 7th International Conference on Enterprise Distributed Object Computing, Page(s):240-251.

Emmert, B. Jorns, O. (2006). Prepaid Peer-to-Peer Services, In Proceedings Sixth IEEE International Conference on Peer-to-Peer Computing, Page(s): 223- 224.

Feldman, M. and Chuang,J. (2005). Overcoming Free-Riding Behavior in Peer-to-Peer Systems, ACM Special Interest Group in Electronic Commerce Exchanges, Vol. 5, Page(s):41-50.

Foster, I. (2002a). What is Grid? A Three Point Checklist, available online at: http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsThe**Grid**.pdf, last accessed on 30 November 2008.

Foster, I. Kesselman, C. Nick, J. M. Tuecke S. (2002b). The Physiology of the Grid An Open Grid Services Architecture for Distributed Systems Integration, available online at: http://www.globus.org/alliance/publications/papers/ogsa.pdf, last accessed on 30 November 2008.

Galli, D.L. (2000). Distributed Operating Systems: Concepts and Practice, Upper Saddle River: Pearson Prentice Hall. 1-27.

Gardfjäll, P. (2004). Accounting in Grid Environments - an Architecture Proposal and a Prototype Implementation, unpublished Masters Thesis, available online at: http://www.cs.umu.se/~peterg/thesis/thesis.pdf, last accessed on 30 November 2008.

Globus Toolkit OGSA, http://www.globus.org/

G**Ö**hner, M. Waldburger, M. Gubler, F. Rodosek, G.D. Stiller, B. (2007). An Accounting Model for Dynamic Virtual Organizations, In Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, Page(s):241 -248.

Goldi, A. (2007. The Emerging Market for Web-based Enterprise Software, unpublished M.Sc Thesis, available online at: http://www.agoeldi.com/media/Thesis_AGoeldi_Final_09MAY07.pdf, last accessed on 30 November 2008.

Guarise, A. Patania, G. Piro, R. (2005). Distributed Grid Accounting System, 4th EGEE Conference, available online at http://personalpages.to.infn.it/~piro/pub/presentations/DGAS_EGEE-4_Pisa_2005-10.pdf, last accessed on 28 November 2008

Haifeng, G. Galligan, P. Mooney, J. Coronado, A. Kehoe, D. (2005). The application of utility computing and Web-services to inventory optimization, In proceeding of IEEE International Conference on Services Computing, Page(s): 185 – 191.

Hales, D. (2004). From Selfish Nodes to Cooperative Networks-Emergent Link-based Incentives in Peer-to-Peer Networks, In Proceedings of the Fourth International Conference on Peer-to-Peer Computing, Page(s):151-158.

Huhns M. and Singh, M.P. (2005). Service-Oriented Computing: Key Concepts and Principles, *IEEE Internet Computing*, Vol. 9, ISBN: 1089-7801, Page(s):75-81.

Ip, A.T.S. Liu, J.C.S. Liu, J. (2008). A Revenue-rewarding Scheme of Providing Incentive for Cooperative Proxy Caching for Media Streaming Systems. ACM Transactions on Multimedia Computing, Communications, and Applications, Vol. 4, No. 1, Article 5.

Iyilade, J. Aderounmu, A. Adigun, M. (2007) .Incentives for Resource Sharing and Cooperation in Grid Computing System, In Proceedings of the International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST), Page(s): 191-198.

Jacob B, Brown M., Fukui K., Trivedi N. (2005). Introduction to Grid Computing, IBM Redbooks, available online at: www.redbooks.ibm.com, last accessed on 30 November 2008.

Jagamathan, S. Almeroth, K.C. (2004). A dynamic pricing scheme for e-content at multiple levels-of-service, International journal on Computer Communications, Vol.27, Page(s): 374-385.

Kannan, P.K. Pope, B.K. Chang, A. (2008). "Pricing Product Lines of Digital Content: a Model Choice Experiment", In Proceedings of 41st Hawaii International Conference on System Sciences, Page(s): 300-308.

Kounev, S. Nou R., Torres, J. (2007). Autonomic QoS-Aware resource management in grid computing using online performance models, In Proceedings of the 2nd international conference on Performance evaluation methodologies and tools ACM International Conference; Vol. 321 Article No. 48 ,ISBN:978-963-9799-00-4.

Li, J. Ziegler, W. Wäldrich, O. Mallmann, D. (2008). Towards SLA Based Software License Management in Grid Computing CoreGRID - Network of Excellence, CoreGRID Technical Report,TR-0136, available online at: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0136.pdf, last accessed on 30 November 2008.

Lim, D. Ho. Q.T. Zhang, J. Lee, B.S. Ong, S.Y. (2005). MOGAS, A Multi-Organizational Grid Accounting System, *International Journal on Information Technology*, Vol. 11, No. 4, Page(s): 84-103.

Liu, G. and Xu, Y. (2007). A New Grid Economy Architecture with Resources Pricing Fluctuation Module, In Proceedings the Sixth IEEE International Conference on Grid and Cooperative Computing, Page(s): 701-704.

Mattern, F. (2000). State of the Art and Future Trends in Distributed Systems and Ubiquitous Computing, available online at: http://www.vs.inf.ethz.ch/publ/papers/DisSysUbiCompReport.html, last accessed on: 01 December 2008.

Mingbiao, L. Jian, L. Shengli, X. (2007). Posted Price Model Based on GRS and Its Optimization Using in Grid Resource Allocation, In Proceeding of International Conference on Wireless Communications, Networking and Mobile Computing, Page(s): 3172-3175.

Nadiminti, K. and Buyya, R. (2005). Enterprise Grid computing: State-of-the-Art, available online at: http://www.gridbus.org/reports/EOSJArticleTR05.pdf, last accessed on: 23 November 2008.

Narahari, Y. Raji, C.V.L. Ravikumar, K. Shah, S. (2005). Dynamic pricing models for electronic Business, *Sardana Journal*, Vol. 30, Page(s): 231-256.

Obreiter, P. and Nimis, J. (2003). A Taxonomy of Incentive Patterns-The Design Space of Incentives for Cooperation, In Proceedings of the Second International Workshop on Agents and Peer-to-Peer Computing, Springer LNCS 2872, Melbourne, Australia.

Papazoglou, M. P. Traverso, P. Dustdar, S. Leymann, F. (2007). Service-Oriented Computing: State of the Art and Research Challenges, *IEEE Computer*, Page(s) 64 - 71.

Parashar,M. and Lee, C. (2005). Grid computing: Introduction and Overview, In Proceedings of the IEEE, Special issue on Grid Computing, available on line at: http://www.caip.rutgers.edu/TASSL/Papers/proc-ieee-intro-04.pdf, last accessed on 27 November 2008.

Patel, Y. and Darlington, J. (2006). Average-Based Workload Allocation Strategy for QoS-Constrained Workflow Based Job in Web Service, In Proceedings of International Conference on Advanced Computing and Communications, Page(s): 664-669.

Pettipher, M.A. Khan, A. Robinson, T.W. Chan, X. (2007). Review of Accounting and Usage Monitoring, available online at: http://www.jisc.ac.uk/media/documents/programmes/einfrastructure/jisc_aum_final_report_wth.pdf, last accessed on 28 November 2008.

Ramaswamy, S. and Malarvannan, M. (2006). Service Oriented Architectures for Grid Computing Environments Opportunities and Challenges, In proceedings of IEEE conference on Granular Computing, Page(s) :325-328 .

Rappa, M.A. (2004). The utility business model and the future of computing services, *IBM Systems Journal*, Vol. 43, no.1, Page(s):32-42.

Reichl, P. Stiller, B. (2003). The Cumulus Pricing Model as an adaptive framework for feasible, efficient and user-friendly tariffing of Internet services, International Journal on Computer Networks, Vol. 43, Page(s): 3-24.

Santos, R. Andrade, A. Cirne, W. (2005). Accurate Autonomous Accounting in Peer-to-Peer Grid, In Proceedings of 3rd International Workshop on Middleware of Grid Computing, Page(s):1-6.

Song, J. Liu, W. Wang, Y. (2007). Competitive Pricing Model for Resource Scheduling in Grid Computing, In Proceedings in Third International Conference on Semantic, Knowledge and Grid, Page(s): 406-409.

Srinivasan, L. and Treadwell, J. (2005). An Overview of Service-Oriented Architecture, Web Services and Grid Computing, HP Software Global Business Unit, available online at: http://h71028.www7.hp.com/ERC/downloads/SOA-Grid-HP-WhitePaper.pdf , last accessed on: 28 November 2008.

Stiller, B. Flury, P. Reichl, P. Hasan, null. (2001). Charging Distributed Services for Computational Grid Architecture, *First IEEE International Symposium on Cluster Computing and the Grid* (CCGrid'01), page(s): 596-601.

Sun Grid Engine project, http://gridengine.sunsource.net/

Tanenbaum, A.S. and Van Steen, M. (2007). Distributed Systems: Principles and Paradigms, Upper Saddle River: Pearson Prentice Hall, 2nd Edition, Page(s): 1-20.

Vassiliadis, S. Tsaknakis, J. Tsakalidis, A. (2006). From Application Service Provision to Service-Oriented Computing: A Study of the IT Outsourcing Evolution, Telemetics and Informatics, Vol. 23, Page(s):271-293.

Vickery, G. Sakai, K. Lee, I. Sim, H. (2006). ICT, E-BUSINESS AND SMEs, available online at: http://www.oecd.org/dataoecd/32/28/34228733.pdf, last accessed on 30 November 2008.

W3C. (2002). Web Services Activity, available online at: http://www.w3.org/2002/ws/, last accessed on 30 November 2008.

Xiaorong, L. Hong, M. C.; Hung, T. Kok, H. T. Turner, S.J. (2008). Design of an SLA-Driven QoS Management Platform for Provisioning Multimedia Personalized Services, In Proceedings of 22nd International Conference on Advanced Information Networking and Applications - Workshops, 2008. Page(s):1405 – 1409.

Yeo, C. S. and Buyya, R. (2007). Pricing for Utility-driven Resource Management and Allocation in Clusters, International Journal of High Performance Computing Applications, Vol. 21, No. 4, Page(s):405-418.

Yeo, C.S. Dias de Assunção, M. Yu, J. Sulistio, A. Venugopal, S. Placek, M. Buyya, R. (2007). Utility Computing and Global Grids, In H. Bidgoli, (Editor), Handbook of Computer Networks, online at: http://www.buyya.com/papers/HandbookCN_Utility_Grids.pdf, last accessed on: 28 November 2008.

Yuan, L. Zeng, G. Mao, X. (2005). A Resource Price-adjusting Mechanism for Supply and Demand Balance in Grid Computing, In Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies. Page(s):97-99.

Yuan, L. Zeng, G. Wang, W. (2006). A Grid Resource Price-adjusting Strategy Based on Price Influence Model, In Proceedings of the Fifth International Conference on Grid and Co-operative Computing. Page(s):311-318.

Zhang, W. Yang, Y. Tang, S. Fang, L. (2007). QoS driven Service Selection Optimization Model and Algorithms for Composite Services, In proceedings of the 31st Annual International Computer Software and Applications Conference, Page(s): 425-431.

Zhao, Z. Xu, L. Wang, B. (2007). A Dynamic Price Model with Demand Prediction and Task Classification in Grid, In Proceeding of the Sixth IEEE International Conferences on Grid and Cooperative Computing, Page(s): 775-782.

APPENDIX A: SIMULATOR SOURCE CODE

```
package sys_arch;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

/**
 *
 * @author Mcebo
 */
@Entity
@Table(name = "meter")
@NamedQueries({@NamedQuery(name = "MeterEntity.findByMeterUID", query =
"SELECT m FROM MeterEntity m WHERE m.meterUID = :meterUID"),
@NamedQuery(name = "MeterEntity.findByResourceUID", query = "SELECT m
FROM MeterEntity m WHERE m.resourceUID = :resourceUID"),
@NamedQuery(name = "MeterEntity.findByConsumerUID", query = "SELECT m
FROM MeterEntity m WHERE m.consumerUID = :consumerUID"),
@NamedQuery(name = "MeterEntity.findByTimeUsage", query = "SELECT m
FROM MeterEntity m WHERE m.timeUsage = :timeUsage"), @NamedQuery(name
= "MeterEntity.findByPeriod", query = "SELECT m FROM MeterEntity m WHERE
m.period = :period")})
public class MeterEntity implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Column(name = "MeterUID", nullable = false)
    private String meterUID;
    @Column(name = "ResourceUID", nullable = false)
    private String resourceUID;
    @Column(name = "ConsumerUID", nullable = false)
    private String consumerUID;
    @Column(name = "TimeUsage", nullable = false)
    private int timeUsage;
    @Column(name = "Period", nullable = false)
    private int period;

    public MeterEntity() {
    }

    public MeterEntity(String meterUID) {
        this.meterUID = meterUID;
    }
```

```
    public MeterEntity(String meterUID, String resourceUID, String consumerUID, int
timeUsage, int period) {
        this.meterUID = meterUID;
        this.resourceUID = resourceUID;
        this.consumerUID = consumerUID;
        this.timeUsage = timeUsage;
        this.period = period;
    }

    public String getMeterUID() {
        return meterUID;
    }

    public void setMeterUID(String meterUID) {
        this.meterUID = meterUID;
    }

    public String getResourceUID() {
        return resourceUID;
    }

    public void setResourceUID(String resourceUID) {
        this.resourceUID = resourceUID;
    }

    public String getConsumerUID() {
        return consumerUID;
    }

    public void setConsumerUID(String consumerUID) {
        this.consumerUID = consumerUID;
    }

    public int getTimeUsage() {
        return timeUsage;
    }

    public void setTimeUsage(int timeUsage) {
        this.timeUsage = timeUsage;
    }

    public int getPeriod() {
        return period;
    }

    public void setPeriod(int period) {
        this.period = period;
    }
```

```
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (meterUID != null ? meterUID.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof MeterEntity)) {
            return false;
        }
        MeterEntity other = (MeterEntity) object;
        if ((this.meterUID == null && other.meterUID != null) || (this.meterUID != null
&& !this.meterUID.equals(other.meterUID))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "sys_arch.MeterEntity[meterUID=" + meterUID + "]";
    }

}
```

Listing 1: Classifier Source Code

```
package sys_arch;

import java.util.List;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.xml.ws.WebServiceRef;

/**
 *
 * @author Joseph
 */
@WebService()
@Stateless()
public class Classifier {

    @WebServiceRef(wsdlLocation =
"http://localhost:8080/ChargingAgentService/ChargingAgent?wsdl")
    private ChargingAgentService service;
    @PersistenceContext
    private EntityManager em;

//    @WebServiceRef(wsdlLocation =
"http://localhost:8080/ChargingAgentService/ChargingAgent?wsdl")
//    private ChargingAgentService service;
    /**
     * Web service operation
     */
    @WebMethod(operationName = "classifier")
    public boolean classifier() {
        List<MeterEntity> meters2bCharged = null;
        //TODO write your implementation code here:
        meters2bCharged = (List<MeterEntity>) em.createQuery("select e from
MeterEntity as e").getResultList();
        for (MeterEntity meterEntity : meters2bCharged) {
            System.out.println(meterEntity.toString());
        }


        try { // Call Web Service Operation

            sys_arch.ChargingAgent port = service.getChargingAgentPort();
            // TODO initialize WS operation arguments here
            java.util.List<sys_arch.MeterEntity> meters2BCharged = null;
            // TODO process result here
            boolean result = port.charge(meters2BCharged);
            System.out.println("Result = " + result);
        } catch (Exception ex) {
```

```
            // TODO handle custom exceptions here
        }


        return true;
    }

    public void persist(Object object) {
        em.persist(object);
    }
}
```

Listing 2 : Correlator Source Code

```
package sys_arch;

import java.util.List;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.xml.ws.WebServiceRef;

/**
 *
 * @author Joseph Okharedia II
 */
@WebService()
@Stateless()
public class ChargingAgent {

    @WebServiceRef(wsdlLocation =
"http://localhost:8080/PriceRegulatorService/PriceRegulator?wsdl")
    private PriceRegulatorService service;
    @PersistenceContext
    private EntityManager em;

    /**
     * Web service operation
     */
    @WebMethod(operationName = "charge")
    public boolean charge(@WebParam(name = "meters2bCharged")
List<MeterEntity> meters2bCharged) {

        System.out.println("ChargingAgentMeter2Charge begins...\n");
        if (meters2bCharged == null) {
            System.out.println("meters2bCharged is null");
```

```
            meters2bCharged = em.createQuery("select m from MeterEntity as
m").getResultList();
        } else if (meters2bCharged.isEmpty()) {
            System.out.println("meters2bCharged is empty");
            meters2bCharged = em.createQuery("select m from MeterEntity as
m").getResultList();
        }
        for (MeterEntity meterEntity : meters2bCharged) {
            System.out.println(meterEntity.toString());
        }
        System.out.println("ChargingAgentMeter2Charge end\n");

        //Check if entity is a consumer or provider

        for (MeterEntity meterEntity : meters2bCharged) {
            if (em.find(ConsumerEntity.class, meterEntity.getConsumerUID()) != null) {
                System.out.println(meterEntity.getConsumerUID() + " is a Consumer");

            }
            if (em.find(ProviderEntity.class, meterEntity.getConsumerUID()) != null) {
                System.out.println(meterEntity.getConsumerUID() + " is a Provider");
            }
        }
        processEntity(meters2bCharged);
        return true;
    }
    //persist into bill table

    public void processEntity(List<MeterEntity> meters2bCharged) {
        for (MeterEntity meterEntity : meters2bCharged) {
            int period = meterEntity.getPeriod();
            String meterUID = meterEntity.getMeterUID();
            String resourceUID = meterEntity.getResourceUID();
            String consumerUID = meterEntity.getConsumerUID();
            int timeUsage = meterEntity.getTimeUsage();
            int units = timeUsage;
            System.out.println("calling Price Recommeder...");
            System.out.println("Resource is : " + resourceUID);
            float price_per_unit = getRegulatedPrice(period, resourceUID, meterUID,
units);
            float totalPrice = price_per_unit * units;
            int rating = updateRating(resourceUID, consumerUID);
            updateDiscount(consumerUID, rating);
            float discountPrice = giveDiscount(totalPrice, rating,consumerUID);
            recordBill(meterUID, resourceUID, units, totalPrice, discountPrice, rating);
        }
    }

    public float getRegulatedPrice(final int period, final String resourceUID, final
String meterUID, final int units) {
```

```
        float regulatedPrice = 0.0f;
        try { // Call Web Service Operation

            sys_arch.PriceRegulator port = service.getPriceRegulatorPort();
            // TODO process result here
            regulatedPrice = port.recommendPrice(period, resourceUID, meterUID,
units);
        } catch (Exception ex) {
            // TODO handle custom exceptions here
        }
        return regulatedPrice;
    }

    public void updateDiscount(String consumerUID, int rating) {
        EntityEntity entity = (EntityEntity) em.find(EntityEntity.class, consumerUID);
        entity.setRating(rating);
    }


    public void recordBill(String meterUID, String resourceUID, int units, float price,
float discountPrice, int rating) {
        String classUID = ((ClassEntity) em.createQuery("SELECT r.classUID FROM
ResourceEntity AS r WHERE r.resourceUID='" + resourceUID +
"'").getSingleResult()).getClassUID();
        BillEntity bill = new BillEntity(meterUID);
        bill.setUnitsUsage(units);
        bill.setResourceUID(resourceUID);
        bill.setClassUID(classUID);
        bill.setCharge(discountPrice);
        bill.setRegulatedPrice(price);
        bill.setRating(rating);
        BillEntity billcopy;
        if ((billcopy = em.find(BillEntity.class, bill.getMeterUID())) != null) {
            billcopy.setUnitsUsage(units);
            bill.setResourceUID(resourceUID);
            bill.setClassUID(classUID);
            billcopy.setCharge(price);
            billcopy.setRating(rating);
            em.merge(billcopy);
        //em.flush();
        //em.clear();
        } else {
            System.out.println("persisting bill : " + bill.toString());
            persist(bill);
        //em.flush();
        //em.clear();
        }
    }

    public float giveDiscount(float price_per_unit, int rating, String customerUID) {
```

```
        int minusValue=0;
        if (rating > 5) {
            price_per_unit *= 0.95;
            minusValue=5;
        }
        if (rating > 10) {
            price_per_unit *= 0.9;
            minusValue=10;
        }
        if (rating > 50) {
            price_per_unit *= 0.8;
            minusValue=50;
        }

        EntityEntity $entity=(EntityEntity)em.find(EntityEntity.class, customerUID);
        $entity.setRating($entity.getRating()-minusValue);
        return price_per_unit;
    }

    public int updateRating(String resourceUID, String consumerUID) {
        ClassEntity classEntity = (ClassEntity) em.createQuery("SELECT r.classUID
FROM ResourceEntity As r WHERE r.resourceUID='" + resourceUID +
"'").getSingleResult();
        int rating = (Integer) em.createQuery("SELECT e.rating FROM EntityEntity AS
e WHERE e.entityUID='" + consumerUID + "'").getSingleResult();
        if (classEntity.getClassUID().equalsIgnoreCase("Guaranteed")) {
            rating += 2;
        } else if (classEntity.getClassUID().equalsIgnoreCase("ControlLoad")) {
            rating += 1;
        }
        return rating;
    }

    public void persist(Object object) {

        em.persist(object);
    }
    // Add business logic below. (Right-click in editor and choose
    // "EJB Methods > Add Business Method" or "Web Service > Add Operation")
}
```

Listing 3: Charging Agent and User Rating Agent Source Code

```
package sys_arch;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.ejb.Stateless;
import javax.management.Query;
import javax.persistence.EntityManager;
import javax.persistence.FlushModeType;
import javax.persistence.PersistenceContext;

/**
 *
 * @author Joseph Okharedia II
 */
@WebService()
@Stateless()
public class PriceRegulator {

    @PersistenceContext
    private EntityManager em;

    /**
     * Web service operation
     */
    @WebMethod(operationName = "recommendPrice")
    public float recommendPrice(@WebParam(name = "Period") int Period,
@WebParam(name = "ResourceUID") String ResourceUID, @WebParam(name =
"MeterUID") String MeterUID, @WebParam(name = "Units") int Units) {
        //TODO write your implementation code here:
        //em.setFlushMode(FlushModeType.AUTO);

        String $QoSClass;
        float $ResourcePrice, $CeilingPrice, $FloorPrice, $MaxCostPrice,
$MinCostPrice;
        int $Demand, $Supply;


        System.out.println("PricingRecommendation begins...");

        $QoSClass = ((ClassEntity) em.createQuery("SELECT r.classUID FROM
ResourceEntity AS r WHERE r.resourceUID ='" + ResourceUID +
"'").getSingleResult()).getClassUID();
        $ResourcePrice = (Float) em.createQuery("SELECT r.resourcePrice FROM
ResourceEntity AS r  WHERE r.resourceUID ='" + ResourceUID +
"'").getSingleResult();
        $MaxCostPrice = (Float) em.createQuery("SELECT MAX(r.resourcePrice)
FROM ResourceEntity  AS r JOIN r.classUID c WHERE c.classUID ='" +
$QoSClass + "'").getSingleResult();
        $MinCostPrice = (Float) em.createQuery("SELECT MIN(r.resourcePrice)
```

```
FROM ResourceEntity AS r JOIN r.classUID c WHERE c.classUID ='" + $QoSClass
+ "'").getSingleResult();

      Object o = ((java.util.Vector) em.createNativeQuery("SELECT SUM(Units)
FROM resource WHERE ClassUID='" + $QoSClass + "'").getSingleResult()).get(0);
      System.out.println("Supply received is : " + o.toString());
      $Supply = Integer.parseInt(o.toString());

      System.out.println("Resource Price : " + $ResourcePrice);
      $CeilingPrice = (Float) em.createQuery("SELECT c.ceilingPrice FROM
ClassEntity  AS c WHERE c.classUID ='" + $QoSClass + "'").getSingleResult();
      $FloorPrice = (Float) em.createQuery("SELECT c.floorPrice FROM ClassEntity
AS c WHERE c.classUID ='" + $QoSClass + "'").getSingleResult();
      SupplyDemandEntityPK pk = new SupplyDemandEntityPK(Period,
ResourceUID);
      SupplyDemandEntity $SupplyDemandEntity = new SupplyDemandEntity(pk);
      $SupplyDemandEntity = findExisting($SupplyDemandEntity);
      $SupplyDemandEntity.setDemand(getResourceDemand($SupplyDemandEntity,
Units));
      $SupplyDemandEntity.setCostPrice($ResourcePrice);
      //$SupplyDemandEntity.setRecommendedPrice($ResourcePrice);
      $SupplyDemandEntity.setCeilingPrice($CeilingPrice);
      $SupplyDemandEntity.setFloorPrice($FloorPrice);
      $SupplyDemandEntity.setQoSClass($QoSClass);
      $SupplyDemandEntity.setMeterUID(MeterUID);
      $SupplyDemandEntity.setSupply($Supply);
      //$SupplyDemandEntity = updateDemand($SupplyDemandEntity);
      $SupplyDemandEntity = recommendPrice($SupplyDemandEntity, Units);
      return regulatePrice($SupplyDemandEntity, $MaxCostPrice, $MinCostPrice);

   }

   private int getResourceDemand(SupplyDemandEntity sndEntity, int units) {
      SupplyDemandEntity $copy = em.find(SupplyDemandEntity.class,
sndEntity.getSupplyDemandEntityPK());
      if ($copy != null) {
         int $demand = $copy.getDemand();
         $demand += units;
         $copy.setDemand($demand);
         em.merge($copy);
         return $demand;
      } else {
         return units;
      }
   }

   private SupplyDemandEntity findExisting(SupplyDemandEntity sndEntity) {
      System.out.println("ResourceUID : " +
sndEntity.getSupplyDemandEntityPK().getResourceUID());
      System.out.println("Period : " +
```

```
sndEntity.getSupplyDemandEntityPK().getPeriod());

      SupplyDemandEntity $XsistinSndEntity = (SupplyDemandEntity)
em.find(SupplyDemandEntity.class, sndEntity.getSupplyDemandEntityPK());
      if ($XsistinSndEntity == null) {
         $XsistinSndEntity = sndEntity;
      } else {
         System.out.println("Existing Entity:" + $XsistinSndEntity.getMeterUID());
      }
      return $XsistinSndEntity;
   }

//   private SupplyDemandEntity updateDemand(SupplyDemandEntity sndEntity) {
//      int $demand = sndEntity.getDemand();
//      sndEntity.setDemand(++$demand);
//      System.out.println("Demand updated");
//      return sndEntity;
//   }
   private int MarketDemand(SupplyDemandEntity sndEntity, int units) {
      int $demand = 0;
      String $QoSClass = sndEntity.getQoSClass();
      System.out.println("got qosClass : " + $QoSClass);
      SupplyDemandEntityPK $sndEntityPK =
sndEntity.getSupplyDemandEntityPK();
      if ($sndEntityPK == null) {
         System.out.println("new market");
         return units;
      }
      int $period = $sndEntityPK.getPeriod();
      System.out.println("Period is : " + $period);
      //$demand=(Integer)em.createQuery("SELECT SUM(s.demand) FROM
SupplyDemandEntity AS s WHERE s.qoSClass='"+$QoSClass+"' AND
s.supplyDemandEntityPK.period="+$period).getSingleResult();
      //System.out.println("Demand : "+$demand);
      Object o = ((java.util.Vector) em.createNativeQuery("SELECT SUM(demand)
FROM supplydemand WHERE QoSClass='" + $QoSClass + "' AND Period=" +
$period).getSingleResult()).get(0);
      if (o == null) {
         return units;
      }
      System.out.println("Supply received is : " + o.toString());
      $demand = Integer.parseInt(o.toString());
      return $demand;
   }

   private SupplyDemandEntity recommendPrice(SupplyDemandEntity sndEntity, int
units) {
      float $MarketSupply = sndEntity.getSupply();
      float $MarketDemand = MarketDemand(sndEntity, units);
      float $RecommendedPrice = sndEntity.getCostPrice();
```

```
        sndEntity.setDemand(((Float)$MarketDemand).intValue());


        System.out.println("start of recommededPrice.....\n\n\n");
        System.out.println("Meter is "+sndEntity.getMeterUID());
        float rateOfChange = ($MarketDemand / $MarketSupply);
        System.out.println("Float Rate of Change : " + rateOfChange);
        if ($MarketDemand == $MarketSupply) {
            sndEntity.setRecommendedPrice($RecommendedPrice);
            return sndEntity;
        }
        if ($MarketDemand > $MarketSupply) {
            System.out.println("MarketDemand > MarketSupply");
            $RecommendedPrice = $RecommendedPrice + ($RecommendedPrice *
rateOfChange);
            System.out.println("Market Demand : " + $MarketDemand);
            System.out.println("Market Supply : " + $MarketSupply);
            System.out.println("Recommended Price : " + $RecommendedPrice);
            System.out.println("\n");
        }
        if ($MarketDemand < $MarketSupply) {
            System.out.println("MarketDemand < MarketSupply");
            $RecommendedPrice = $RecommendedPrice - ($RecommendedPrice * (1 -
rateOfChange));
            System.out.println("Market Demand : " + $MarketDemand);
            System.out.println("Market Supply : " + $MarketSupply);
            System.out.println("Recommended Price : " + $RecommendedPrice);
            System.out.println("\n");
        }
        sndEntity.setRecommendedPrice($RecommendedPrice);
        System.out.println("end of recommededPrice.....\n\n\n");
        return sndEntity;
    }

    private float regulatePrice(SupplyDemandEntity sndEntity, float maxCostPrice,
float minCostPrice) {
        float recommendedPrice = sndEntity.getRecommendedPrice();
        float floorPrice = sndEntity.getFloorPrice();
        float ceilingPrice = sndEntity.getCeilingPrice();
        float regulatedPrice = recommendedPrice;


        if (recommendedPrice < floorPrice) {
            float $maxDiffPrice = Math.max((floorPrice - recommendedPrice),
(minCostPrice - recommendedPrice));
            regulatedPrice = recommendedPrice + $maxDiffPrice;
        } else if (recommendedPrice > ceilingPrice) {
            float $maxDiffPrice = Math.max((recommendedPrice - ceilingPrice),
(recommendedPrice - maxCostPrice));
            regulatedPrice = recommendedPrice - $maxDiffPrice;
```

```
        }
        sndEntity.setRegulatedPrice(regulatedPrice);

        System.out.println("ResourceUID : " +
sndEntity.getSupplyDemandEntityPK().getResourceUID());
        System.out.println("Demand : " + sndEntity.getDemand());
        System.out.println("Ceiling Price : " + sndEntity.getCeilingPrice());
        System.out.println("Floor Price : " + sndEntity.getFloorPrice());
        System.out.println("MeterUID : " + sndEntity.getMeterUID());
        System.out.println("Period : " +
sndEntity.getSupplyDemandEntityPK().getPeriod());
        System.out.println("QOS Class : " + sndEntity.getQoSClass());
        System.out.println("Cost Price : " + sndEntity.getCostPrice());
        System.out.println("Recommended Price : " +
sndEntity.getRecommendedPrice());
        System.out.println("Regulated Price : " + sndEntity.getRegulatedPrice());
        System.out.println("Supply : " + sndEntity.getSupply());
        System.out.println("Max Cost Price : " + maxCostPrice);
        System.out.println("Min Cost Price : " + minCostPrice);


        if (em.find(SupplyDemandEntity.class, sndEntity.getSupplyDemandEntityPK())
== null) {

            em.persist(sndEntity);
            //em.flush();
            //em.refresh(sndEntity);
            System.out.println("persisting from pricing");
        } else {
            em.merge(sndEntity);
            //em.flush();
            //em.refresh(sndEntity);
            System.out.println("merging from pricing");
        }
        //em.flush();
        //em.clear();
        //em.close();
        System.out.println("Pricing Recommendation ends\n\n\n");
        return regulatedPrice;

    }
}
```

Listing 4: Price Regulator and Price Recommender Source code