COORDINATION-THEORETIC FRAMEWORK
FOR
SHARING E-BUSINESS RESOURCES

AREMU, DAYO REUBEN, B.Sc., M.Sc.

December 2008.

COORDINATION-THEORETIC FRAMEWORK FOR SHARING E-BUSINESS RESOURCES

AREMU, DAYO REUBEN, B.Sc., M.Sc.

A thesis submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

Department of Computer Science, Faculty of Science and Agriculture,

University of Zululand, South Africa

December 2008.

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material which has been accepted for the award of any other degree or diploma at any University or other Institution of higher learning, except where due acknowledgement has been made in the text.

Signature:............................ 08/01/2009.........

Aremu Dayo Reuben Date:

Approval

Signature:...Date: ....

Supervisor

Prof. M. O. Adigun,

Department of Computer Science University of Zululand

DEDICATION

This dissertation is first and foremost dedicated to God, and to the members of my family: my darling wife, Victoria Iyabode Adenike; my wonderful children: Mercy Adenike Adejoke; Blessing Folashade Eyiwumi; John Victor Adedayo; and Grace (Ore–Ofe). You have all been so wonderful to me.

ACKNOWLEDGMENTS

Yours, O Lord, is the greatness, and the power, and the glory, and the victory, and the majesty, for all that is in the heaven and in the earth are yours; yours is the kingdom, O Lord, and you are exalted as head above all [I Chronicle 29: 11].

My profound gratitude goes to the following bodies and individuals who have made this study possible:

(i) National Research Foundation (NRF), South Africa, for making money available for the study;

(ii) Telkom, for the financial assistance to pursue the study.

(iii) University of Ilorin, Ilorin, Nigeria, for granting me study leave to pursue the study.

(iv) My supervisor and his wife, Professor and Mrs M. O. Adigun, for guidance and assistance, both in cash and in kind during the course of the study.

(v) Mr and Mrs Joseph Odetokun, of His Grace Saloon International, South Africa, for being a wonderful family friend in the time of need.

(vi) Mr and Mrs Joseph Okharedia, for being part of my struggles during the course of the study in South Africa.

(vii) My faithful prayer partners: Elder and Mrs Elegba Emmanuel and the ECWA Chapel family

(viii) Prof. J. A. Sadiku, Head of Computer Science Department, University of Ilorin, Ilorin, Nigeria, for being an understanding head of department.

(ix) Prof. J. A. Gbadeyan, for being a worthy lecturer and supervisor at both Honors' and master's degrees levels, and for being a worthy referee that one can always count on.

(x) Staff and students of the Department of Computer Science. University of Zululand, South Africa, for providing a challenging environment that made the study possible.

(xi) Dr Alice Wafula Kituyi-Kwake, for the useful criticism and suggestions.

(xii) My colleagues, Post Graduate students, at both master's and Doctoral levels, for their contributions to the success of the study.

Last but not the least, my special appreciation goes to my darling wife, Victoria, for being a wife in deed, a faithful prayer partner, and for holding forth in the family during the long period of absence from home, and to my wonderful children: Mercy, Blessing, John, and Grace, my niece, Yemisi, I am proud of you all.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTS OF GRAPHS

ABSTRACT

Market-based models are frequently used in the resource allocation on the computational grid. However, as the size of the grid grows, it becomes more difficult for customers to negotiate directly with all the grid resource providers. Middle agents are introduced to mediate between the providers and the customers so as to facilitate the resource allocation process. The most frequently deployed middle agents are the matchmakers, the brokers, and the market-maker. The matchmaking agent finds possible candidate providers who can satisfy the requirements of the customer, after which the customer directly negotiates with the candidates. The broker agents are mediating the negotiation with the providers in real time. The market-maker acquires resources and resource reservations in large quantities, and resells it to the customers.

The purpose of this study was to establish a negotiation framework that enables optimal allocation of resources in a grid computing environment, such that: clients are allowed to have an on – demand access to pool of resources; collaboration is enhanced among resource providers; cost saving and efficiency are ensured in resource allocation. The objectives to realize this purpose were: first, to design an appropriate negotiation model that could be adopted to achieve optimal resource allocation; second, to determine an effective search strategy that could be employed in order to reach a Pareto efficient negotiation solution; third, to adopt a negotiation strategy or tactics that negotiators could use to arrive at optimal resource allocation.

In order to achieve the goals and objectives set for the study, the following methodologies were used: (i) a critical survey of the existing economic approach and models for negotiating grid resources was conducted; (ii) The knowledge gained from the literature surveyed was used to construct a novel model called Co-operative Modeler for mediating grid resource sharing negotiation. We used Mathematical notations: first, to construct a theoretical model for allocation of resources to clients' task; second, to present a novel **C**ombinatorial **M**ulti-**Ob**jective **O**ptimization model (CoMbO), by modeling negotiation offers of agents as multi-objective optimization problem; and third, to present Genetic and Bayesian Learning Algorithms for implementing the model presented; (iii) an implementation prototype of the Co-

operative Modeler was developed by: first, implementing the Co-operative Modeler to mimic the real world negotiation situation; second, using time–dependent negotiation tactics to evaluate the negotiation behavior of the Co-operative agents.

The Co-operative Modeler has been shown to guarantee: (i) Scalability of number of users. i.e. multiple users can access a virtualized pool of resources in order to obtain the best possible response time overall by maximizing utilization of the computing resources;(ii) Enhanced collaboration.– that is promoting collaboration so that grid resources can be shared and utilized collectively, efficiently, and effectively to solve computer–intensive problems; (iii) Improved business agi!ity - that is decreasing time to process data and deliver quicker results; and (iv) Cost saving. – i.e. leveraging and exploiting unutilized or under utilized power of all computing resources within a grid environment.

CHAPTER ONE

INTRODUCTION

1.1 Background

In a grid environment, resources must be shared. The nodes in the grid, being possibly independent domains would, therefore, require a coordination framework such that resource sharing can be based on the most appropriate strategies. Recently, analogy has been drawn between resources in a Utility Computing environment and the way businesses would like to share their jointly owned resources, especially when operating in groups. In this study, therefore, we have carried out an extensive survey of economic models that exhibit useful negotiation strategies for sharing resources in a grid environment. As a result of the survey, we have come up with a Co-operative Modeler concept designed to mediate resource negotiation in an on-demand context. The concept is based on the following assumptions: resources of members of a Co-operative are pooled into a common pool, and administered by the Co-operative; individual members of the Co-operative are only charged for the resources they use; resource sharing takes place in trusted party environment such as in a grid; resource allocation is coordinated as an economic activity using negotiation.

The idea of a Co-operative is central to this study. A Co-operative is defined as an autonomous association of agents united voluntarily to meet their common economic, social, and cultural needs and aspirations through a jointly owned and democratically controlled enterprise. Co-operative business is a business owned and controlled by the people who use its services. By pooling resources together, the Co-operative members can combine both sales returns and operating expenses, while distributing sales among members in proportion to volume each provides through the Co-operative over a specified period. A Co-operative may be operated as a single resource pool or a multiple resource pool. In a pool operation, members bear the risk and gains of changes in market prices. So, the advantages of pooling are that: (i) it spreads market risks and,

(ii) it reduces the costs of making service available to service consumers. (iii) Pooling also permits the merchandising of resources according to a program management deems most desirable and which can be planned with considerable precision in advance. (iv) it also permits the use of utmost caution in placing and timing shipments to market demands especially in new markets i.e., orderly marketing; Finally, it can help finance the Co-operative.

The allocation of pooled resources is modeled as a new theoretical framework for negotiating grid computing resources. In other words, participants make negotiation offers which are treated as input into a typical resource allocation problem, which in turn is reduced to an optimization problem. The solution's strategy selected the adoption of genetic algorithm that searches for a set of negotiation offers that would maximize or minimize the conflicting objectives of the users. In addition to the above, the study discussed Bayesian learning algorithm for equipping negotiating agents with learning capability.

1.2 Statement of the problem

Negotiation in real world situations are usually characterized by combinatorial complex negotiation spaces which involve multiple parties and many issues such as price, quality of service, delivery time etc. Not only that negotiators are constrained by limited computational resources, time, and limited information about the opponents. While game theoretical negotiation model provides excellent theoretical analysis of the optimal outcomes (e.g. Nash equilibrium) given the scenarios of bilateral negotiations. game theories fail when it comes to advising the course of actions that a negotiator can follow to reach the optimum outcome in real world negotiations. One main concern for practical application of game theories is that the search space of all the possible strategies and interactions required to identify the equilibrium solutions grows exponentially. It, therefore, means that the task of finding an optimum negotiation strategy is in general computationally complex. Another challenge is that classical game theories assume that complete information about every negotiation agent is available to a centralized decision making mechanism. It turns out that such an assumption does not hold in most real world negotiation environment. Other models of games which are based on incomplete information about the opponent (e.g. in the form of priori probability distributions of other's strategies) have been proposed. Nevertheless, it is still problematic to find an automated means of acquiring or estimating accurate probability distribution given a negotiation situation (von Neuman, 2004 & Robinstein, 1982).

In order to resolve the foregoing dilemma with respect to game theory, we turned to economic models that exhibit useful negotiation strategies for our particular purpose of sharing e-business resources. There are three research questions we intend to answer. First, can we find an appropriate negotiation model as a mechanism for achieving optimal resource allocation? Second, is there an effective search strategy that can be employed in order to reach an optimal negotiation solution? Third, what negotiation strategy or tactics would lead to an optimal resource allocation?

1.3 Contribution to Knowledge

Market-based models are frequently used in resource allocation on the computational grid. However, as the size of the grid grows, it becomes more difficult for the customer to negotiate directly with all the grid resource providers. Middle agents are introduced to mediate between the providers and customers so as to facilitate the resource allocation process. The most frequently deployed middle agents are the match-makers, the brokers, and the market-maker. The matchmaking agent finds possible candidate providers who can satisfy the requirements of the customer, after which the customer directly negotiates with the candidates. The broker agents are mediating the negotiation with the providers in real time. The market-maker acquires resources and resource reservations in large quantities, and resells it to the customers.

It has been argued in the literature (Bai, et al., 2006) that the market-maker model favored the grid marketplace most, in contrast to the matchmaker and the Broker agent models. However, the market-maker agent's offer of resources is usually decoupled from the price at which it acquires resources. The implication of this to both the resource producers and the resource consumers are not far fetched: the market-maker agent is favored more than resource producers by making a higher profit than the owner of the resource; the market-maker agent also exploits the resource consumers by offering them resources at far higher prices than the marginal costs of the resources. Consequently, the market-maker negotiation model distorts the trade-off in the grid economy, and moves it away from Pareto efficiency. It follows from these arguments that the market-maker model cannot be used to achieve optimal resource allocation.

In this study we proposed a new Middleware agent concept called Co-operative Modeler for mediating resource negotiation. The Co-operative Modeler bridges the gap between the competitive market models (e.g. auction models) and the Monopoly/Oligopoly models by enabling the resource providers to come together in a Co-operative relationship to make their resources accessible to the end users at reduced costs. Another contribution of the work is a new theoretical model called **Com**binatorial **M**ulti-O**b**jective **O**ptimization model (CoMbO) for coordinating resource sharing negotiation, which modeled agent negotiation offers as a multi-objective optimization problem, and used genetic algorithm for searching for the negotiation offer that would optimize the conflicting objectives of the negotiators. Finally, the work culminates in an implementation prototype of the Co-operative Modeler that enables us to study, analyze and understand real world negotiation between agents.

1.4 Goal of the study

The goal of this research is to propose a theoretical framework that allows electronic business resources to be negotiated among collaborating resource providers and consumers; such that on – demand access to pool of resources. cost saving and efficiency are ensured in resource allocation.

1.5 Objectives of the study

To achieve the above set goal, the following objectives are addressed:

(i) To formulate an appropriate negotiation model that could be adopted to achieve optimal resource allocation;.

(ii) To determine an effective search strategy that should be employed in order to reach a Pareto-efficient negotiation solution and

(iii) To experiment with a negotiation strategy or tactics showing how negotiators can use the strategy to arrive at optimal resource allocation.

1.6 Research Design and Methodology

The study focuses on Coordination of grid resource sharing using conventional economic negotiation models. We categorized the economic negotiation models into five categories namely: Auction Models; Pricing Models; Bartering/Exchange Model; Bilateral Negotiation Models; and Monopoly/Oligopoly Models [Table 2.2]. The study investigated the various negotiation models and tactics based on our categorization and determine which of them to improve upon and use in order to analyze and evaluate the negotiation behavior of the Co-operative Modeler agents. We used the following methodology in order to achieve the goals and objectives set for the study:

(i) Literature Survey

A critical survey of the existing economic approach and models for negotiating grid resources was conducted by examining in detail two broad categories of Resource Sharing Economic Models. namely competitive and non competitive paradigms. The Auction and the bilateral models are presented as examples of negotiation models in a competitive market setting. while Monopoly/Oligopoly model represents non competitive alternative.

(ii) Construction of Models

A formalism has been formulated for the proposed Co-operative Modeler. There were three parts to the formalism. (a) A theoretical model for allocating resources to a client's task was constructed based on the following three assumptions. First, each resource type consists of variants such that a variant is identified by quality of the resource, the capacity (size) of the resource, the response time or the cost as the case may be. Second, a client is allowed to indicate the importance attached to a resource variant. Third, resources are allocated in such a way that the number allocated does not exceed the number available. (b) Then, the modus operandi of making negotiation offers by agents is formulated as a multi-objective optimization problem. (c) Finally, the Bayesian learning algorithm is also modeled formally using mathematical notations.

(iii) Prototype **Implementation of the Co-operative Modeler**

The implementation prototype of the Co-operative Modeler was constructed to mimic the various aspect of the Co-operative Modeler including agent: selecting item to buy; negotiating for the price of the item; when agreement is reached, indicating quantity of item to buy; giving bank or credit card detail to pay for the item bought; and receiving receipt for the item(s) bought. The implementation prototype was constructed using NetBeans Interface Development Environment coupled with Document Object Model (DOM) –Based parser for passing xml messages between the Co-operative agents.

(iv) Evaluation of the Co-operative Modeler

We used time – dependent negotiation tactics to evaluate the negotiation behavior of the Co-operative Modeler agents.

1.7 Synopsis of the rest of the thesis

The thesis is organized into five chapters in all. The following is an overview of the content of each chapter. The second chapter presents a survey of the existing economic approach and models for negotiating grid resources by examining in detail, negotiation in two broad categories of market settings, namely competitive and non competitive market settings. The Auction and the bilateral models are presented as examples of negotiation models in a competitive market setting while Monopoly/Oligopoly model represents a negotiation model in a non competitive market setting. Chapter Three presents the architecture of the Co-operative Modeler by using formal notations: First, to construct a theoretical model for allocation of resources to tasks with the assumptions that: (i) each resource type is made of different variants; (ii) the importance attached to each resource variants differ from one client to another; (iii) the quantity of resources allocated to a client's task can not be more than the available resources; Second to construct a model of the negotiation offers of agents as multi-objective optimization problem; and third to present Bayesian Learning Algorithm. The chapter also presents designs by using sequence and class diagrams for implementing the Co-operative Modeler. Chapter four covers the implementation prototype of the Co-operative Modeler. The chapter discussed an experimental design to evaluate the negotiation behavior of the Co-operative Modeler agents, by adopting time -dependent negotiation tactics to simulate: (i) the effect of time on the importance an agent attached to a negotiation issue, and (ii) the effect of time on the negotiation offers of an agent. Chapter five concludes the study by providing a summary of the achievements of the study, makes recommendations and suggestions for future direction.

CHAPTER TWO

CO-ORDINATION IN THE GRID AS A RESOURCE MARKET

2.1 Introduction

This study looks forward to a future of grid computing, in which all Internet users will participate in a global grid where computational resources will be acquired on demand and provided with an agreed quality of service. Such a global grid will allow the emergence of new marketplaces for trading application services, computation, bandwidth and storage. This will require a number of economic strategies to co-ordinate the sharing activities between providers and consumers in such a market.

In a grid computing environment, providers and consumers of resources must co-ordinate their activities or pursuit and when conflicts arise, they must be able to negotiate. Conflicts may arise from simple limited resource contention to more complex issues where both the providers and the consumers of resources disagree because of discrepancies between their domain expertise. Negotiation is a resource sharing co-ordination technique, that refers to the process by which group of providers and consumers of resources communicate with one another in order to reach a mutually acceptable agreement on resource allocation (Comuscio & Jennings, 2003). In a well known economic model, negotiation can either be based on auctions, or with bilateral negotiation. Negotiation models discussed in this study are classified into two types of market settings: competitive and non competitive market settings. Negotiation in a competitive grid market setting is where several grid resource providers and brokers or consumers determine the market price, while negotiation in non competitive market setting refers to how a single grid resource provider dominates the market and is, therefore, the single provider of a particular resource.

Table 2. 1: Framework for Dependencies between Activities and Alternative Coordination Processes for Managing Them (Malone & Crowston, 1994).

Dependency	Examples of Coordination Processes for Managing Dependency
Shared Resources	"First come/first serve" Priority Order, Budgets, Managerial Decision, Market-like Bidding
Task Assignments	Same as for Shared Resources
Producer / Consumer Relationships	Same as for Shared Resources
Prerequisite Constraints	Notification, Sequencing, Tracking
Transfer	Inventory Management (e.g. "Just In Time, Economic Order Quantity")
Usability	Standardization, Ask Users, Participatory Design
Design for Manufacturability	Concurrent Engineering
Simultaneity Constraints	Scheduling, Synchronization
Task / Subtask	Goal Selection, Task Decomposition

(Malone & Crowston, 1994) defined the term Co-ordination as managing dependencies between activities. They argued that Co-ordination can occur in many kinds of systems: human; computational; biological; and others. According to them, questions about how people manage dependencies among activities are central to parts of organization theory, economics, management science, sociology, social psychology, anthropology, linguistics, law, and political science. In computer science, in their opinion, dependencies between different computational processes must certainly be managed. They pointed out that certain kinds of interactions among computational processes resemble interactions among people. They further characterized different kinds of dependencies and identified the coordination processes that can be used to manage them [Table 2.1].

In the next section, we discuss the concept of grid computing, and the context provided by the grid computing that gives us the basis for co-ordinating its resources.

2.2 The Context Provided By Grid Computing

The advent of the Internet is rapidly changing how firms conduct their business. Among various e-business services offered by the Internet, is the potential to enhance co-operative planning, resource sharing, quality of service provisioning and decision making via cost effective, web-based mechanisms. These technological developments, not only facilitate intra-firm coordination across divisional or functional boundaries, but also inter-firm decisions between consumers of resources and suppliers of resources.

The Internet is a massive network of networks, a networking infrastructure connecting millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet. Information that travels over the Internet does so via a variety of languages known as protocols.

According to (Tian, 2003), the Internet connects dispersed resources but does not support co-ordination of these resources to resolve a common problem. New protocols such as resource discovery, negotiation or coordination (protocols) are necessary for trading resources on the Internet. (Tian, 2003) argued further that "as more resources are connected to the Internet, a framework is required to approach all these resources on the net, no matter what protocols each of them uses and where they are located, and to collect them together in order to help people collaborate and work on a common and complicated problem."

(Buyya, 2001) claimed that the ubiquity of the Internet as well as the availability of powerful computers and high – speed network technologies as well as low cost computers is increasingly changing the computing landscape and the society all together. These technological opportunities have led to the possibility of using wide area distributed computers for solving large – scale problems, leading to what is popularly known as Grid Computing.

The Grid promises the revolution of the Internet by a novel and advanced support for creation of Virtual Enterprises (VE) for sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices. These are geographically distributed and owned by different organizations for solving large – scale computational and data intensive problems in science, engineering, and commerce, with the ultimate goal of dealing with any virtual resources in the same way as with real commodities. However, they are distinct in that a virtual resource has an independent location or consists of different components.

The term Grid comes from the analogy to the electric power grid. Inside the power grid, are different types of resource producers such as hydroelectric power plant, solar power plants, atomic power plants, etc. The transmission and handling also differ, but the consumer gets the resource in the expected quality, unlimited in the consumer's point of view, without considering the background. In the same way the user of virtual resources should not be concerned with the production or provision of it, but the user may access it dynamically and get it with an expected quality.

The Grid promises to enable the homogeneous access to virtual resources without revealing the heterogeneous and dynamic manner of the underlying real world. The fulfillment of this promise highly depends on the user's requirements and their expectations about quality of service and other non functional requirements. An ideal virtual resource is independent of its real properties and corresponds perfectly to the user requirements.

A Grid provides an unlimited resource to customers without revealing the internal structure. Like in real markets, the resource limitation should not be noticed by customers. The ideal combination of all demands and all available resources can be seen as a perfect market. This is the overall goal of Business in the Grid.

2.2.1 The nature of Grid Computing

Even though there have been controversies in the past as to what grid computing is, we provide as follows, definitions from few researchers as to what grid computing actually is.

According to (Foster 2002), grid computing has the following primary attributes: (i) computing resources are not administered centrally; (ii) open standards are used; and (iii) non-trivial quality of service is achieved.

(IBM, 2005) also defines grid computing as the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and a vast array of other computing resources over the Internet.

In their own definition, (Plaszczak & Wellner, 2005) define grid technology as the technology that enables resource virtualization, on-demand provisioning, and resource sharing between organizations.

(Buyya, 2005), also defined a grid as a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.

The concept of Grid Computing actually started as a project to link geographically dispersed supercomputers, but it has since grown from beyond its original intent. The grid Infrastructure can benefit many applications, including collaborative engineering, data exploration, high throughput computing, distributed supercomputing, and service – oriented computing. The Grid allows users to solve larger – scale problems by pooling together resources that could not be coupled easily before. The Grid is not only a computing Infrastructure for large applications, it is a technology that can bond and unify remote and diverse distributed resources ranging from meteorological sensors to data vaults, and from parallel supercomputers to personal digital organizers. As such it will provide pervasive services to all users that need them. In conclusion, we take the Grid as a potential market in which providers and consumers trade scarce resources; and there is a strong need for the coordination theoretical framework such as is being provided in this study.

2.2.2 Towards an Emerging Grid Market

As the Grid matures, a vision of a true global grid computing infrastructure has started to emerge. In the global grid, computational resources are acquired on demand and provided with an agreed quality of service. Participation is open to all, and resources may be used or potentially provided by the general public, institutions or companies. Such a global grid will allow the emergence of new marketplaces for trading application services, computation, bandwidth and storage. In order to motivate everyone on the Internet to contribute their machines (idle) resources in order to achieve the vision of a true global grid, therefore, calls for a mechanism that allows resource owners to earn money by letting others use their (idle) computational resources for solving their problems. One of the best mechanisms for achieving this, in line with our earlier position to adopt market - based model from [Table 2.1], and supported by (Buyya, 2000), is to support the concept of computational economy in building and coordination of grid resources. Several Grid economy models drawn from conventional markets have been proposed for co-ordinating the emerging Grid market.

2.2.3 Grid Market as a Seamless Computing Environment

The general design features that are required by a Grid market to provide users with a seamless computing environment are:

(i) Multiple Administrative Domains and Autonomy

Grid resources are geographically distributed across multiple administrative domains and owned by different organizations. Grid resources, Owners, and usage policies are autonomous. This means that distributed resources being traded in the grid marketplaces are provided and consumed by autonomous providers and consumers respectively.

(ii) Heterogeneity

A Grid market involves a multiplicity of resources that are heterogeneous in nature and will encompass a vast range of technologies. That is to say that the resources in the grid marketplaces are of different types, and each type has different variants.

(iii) Scalability

Admission into a grid market is open to all Internet users. The implication of this is that a grid might grow from a few integrated resources to millions. This raises the problem of potential performance degradation. Consequently, applications that require a large number of geographically located resources must be designed to be latency and bandwidth tolerant.

(iv) Dynamicity or Adaptability

In a Grid, resource failure is the rule rather than exception. With so many resources in a Grid, the probability of some resource failing is very high. Resource managers or applications must tailor their behavior dynamically and use the available resources and services efficiently and effectively. The dynamicity of the grid marketplace allows participants in the grid to join and leave the market at will.

The next section examines in detail two broad categories of Resource Sharing Economic models, namely competitive and non competitive paradigms. The Auction model is presented as an example of a competitive approach to co-ordination while Monopoly/Oligopoly represents the non competitive alternative.

2.3 Economic Models for Coordinating the Grid Resource Market

The purpose of this section is to address the first research question raised in this study by investigating the existing economic-based approaches for coordinating resource sharing in the grid computing environment. The economic-based approaches provide a fair basis in successfully coordinating decentralization and heterogeneity that is present in the grid economies. The economic models can be based on auctions, prices, bartering/exchange, or with bilateral negotiation. The economic models investigated are categorized and structured into the framework provided in [Table 2.2] below.

Table 2. 2: Categorization of Economics Models for Discussion (Aremu, & Adigun, (2008).

Type of Market Setting	**Resource Sharing Negotiation Model**
Competitive Market	**2.3.1 Auction Models** **2.3.1.1 Auction Protocols** 2.3.1.1.1 English auction 2.3.1.1.2 First price sealed-bid Auction 2.3.1.1.3 Vickrey (second-price sealed-bid) auctions 2.3.1.1.4 Dutch auction 2.3.1.2 A tendering/Contract-Net Model 2.3.1.3 A Bid-Based Proportional Resource Sharing Model **2.3.2 Pricing Model** 2.3.2.1 A Commodity Market(Flat or Demand and Supply driven) Model 2.3.2.2 Posted Prices 2.3.2.3 A Bargaining Model **2.3.3 Bartering/Exchange Model** 2.3.3.1 A Community/Coalition/Bartering Model **2.3.4 Bilateral Negotiation Model** 2.3.4.1 Service – Oriented Negotiation Model
Non Competitive Market	**2.3.5 Monopoly/Oligopoly Model**

2.3.1 Auction Models

Auctions are highly structured forms of negotiation between several agents. An auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants (McAfee & McMillian, 1987). Auctions enable the sales of products without a fixed price or a standard value. The typical purpose of an auction is for the seller to obtain a price, which lies as close as possible to the highest valuation among potential buyers. The auction model supports one to many negotiations, between a grid service provider (seller) and many grid service consumers (buyers), and reduces negotiation to a single value (i.e. price). The auctioneer sets the rules of auction, acceptable for the consumers and the providers.

The three key players involved in auction are: the grid resource owners, the auctioneer (mediator) and the buyers. Many e-commerce portals such as Amazon.com and eBay.com are serving as mediators (auctioneers). In a Grid environment, providers can use an auction protocol for deciding service value/price. The steps involved in the auction process are:

(i) a Grid Service Provider announces its services and invites bids;

(ii) brokers offer their bids (and they can see what other consumers offer if they like – depending on how open/closed and

(iii) the broker and Grid Service provider communicate privately and use the resource.

The contents of the deal template used for work announcements include the addresses of the users, the eligibility requirements specifications, the task/service abstraction, an optional price that user is willing to invest, the bid specification (what should the offer contain) and the expiration time (the deadline for receiving bids).

From a contractor's/Grid Service Providers' perspective, the process is:

(i) receive tender announcements/advertisements (in Grid market Directory);

(ii) evaluate the service capability;

(iii) respond with a bid;

(iv) deliver service if a bid is accepted and

(v) report result and bills the broker/user as per the usage and agreed bid.

An auction consists of an auctioneer and potential bidders. Auctions are usually discussed in situations where the auctioneer wants to sell an item and get the highest possible payment for it while the bidders want to acquire the item at the lowest possible price.

2.3.1.1 Auction Protocols

Auctions can be conducted as open or closed depending on whether back and forth offers and counter offers are allowed. The consumer may update the bid and the provider may update the offer sale price. Depending on these parameters, auctions can be classified into five types; (i) English auction (first-price open cry); (ii) First price sealed-bid auction; (iii) Vickrey (second-price sealed-bid) auctions (Rosenschein & Zlotkin, 1994); (iv) Dutch auction; (v) Double auction (continuous).

2.3.1.1.1 English Auction (first – price open cry)

In the English (first – price open- cry) auction, each bidder is free to raise his bid. When no bidder is willing to raise any more, the auction ends, and the highest bidder wins the item at the price of his bid. In this model, the key issue is how Grid Resource Brokers decide how much to bid. A Grid Resource Broker has a private value (as defined by the user) and can have a strategy for a series of bids as function of its private value and prior estimation of other bidder's valuations, and the past bids of other. The Grid Resource Broker decides the private value depending on the user-defined requirements (mainly deadline and budget that he is willing to invest in the solution of the problem). In the case of private-value, English auctions, a Grid Resource Brokers' dominant strategy is to always bid a small amount 'higher' than the current highest bid, and stop when its private-value price is reached. In correlated value auctions, the policies are different and they allow the auctioneer to increase the price at a constant rate or at the rate he or she wishes. Those not interested in bidding anymore can openly declare so (open exit) without the possibility of re-entry. This information helps other bidders and gives them a chance to adjust their valuation.

2.3.1.1.2 First price sealed – bid Auction

In the first – price sealed – bid auction, each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of his bid. In this case, a broker strategy is his bid as a function of his private value and the prior beliefs of other bidders' valuations. In general there is no dominant strategy for bidding in this auction. The best strategy is to bid less than his true valuation but how much less depends on what the others bid. The agent would want to bid the lowest amount that still wins the auction – given that this amount does not exceed his valuation. With common knowledge assumptions regarding the probability distributions of the agents' values, it is possible to determine Nash equilibrium strategies for the agents.

Definition 2.3.1: Agent Bids

We defined that in a private value auction where the valuation v_i for each agent i is drawn independently from a uniform distribution between 0 and $\bar{v}$, there is a Nash equilibrium where every agent i bids is $\frac{|A|-1}{|A|}v_i$.

2.3.1.1.3 Vickrey (second – price sealed - bid) Auction

In the Vickrey (second-price sealed bid) auction, each bidder submits one bid without knowing the other's bids. The highest bidder wins the item at the price of the second highest bidder (Vickrey, 1961). An agent's strategy is his bid as a function of his private value and prior beliefs of others' valuations.

2.3.1.1.4 Dutch Auction

In the Dutch (descending) auction, the auctioneer starts with a high bid/price and continuously lowers the price until one of the bidders takes the item at the current price. The Dutch auction is similar to the first – price sealed – bid auction, because in both cases, the agent's bid matters only if it is the highest, and no relevant information is revealed during the auction process. From the broker's bidding strategic point of view, a Dutch auction is similar to an English auction (first-price sealed-bid auction.). The key difference between them is that in an English auction the bids starts with a low opening and increase progressively until demand falls, whereas in a Dutch auction the bids start with a high opening price and decrease progressively until demand rises to match supply. Dutch auctions are efficient in terms of real time because the auctioneer can decrease the price at a brisk pace.

2.3.1.2 A Tendering or Contract-Net Model

Tender or Contract-net model is one of the most widely used models for service negotiation in a distributed problem solving environments (Smith & David, 1980). The model is based on the contracting mechanism used by businesses to govern the exchange of goods and services. It helps in finding an appropriate service provider to work on a given task. In the Tender/Contract-Net model, the consumer (GRB) invites sealed bids from several GSPs and selects those bids that offer lowest service cost within their deadline and budget. In the Auction model, producers invite bids from many consumers and each bidder is free to raise their bid accordingly. The auction ends when no new bids are received. The auction can be performed through open or closed bidding protocols.

A user or service broker asking for a task to be solved is called the manager and the service that might be able to solve the task is called the potential contractor. From a manager's perspective, the process is:

(i) the consumer (broker) announces its requirement (using a deal template) and invites bids from Grid Service Providers;

(ii) interested Grid Service Providers evaluate the announcement and respond by submitting their bids;

(iii) the brokers evaluates and awards the contract to the most appropriate Grid Service Provider(s);

(iv) step (ii) goes on until one is willing to bid a higher price or the auctioneer stops if the mini price line is not met;

(v) the Grid Service Provider offers the service to the one who wins and

(vi) the consumer uses the service.

2.3.1.3 A Bid-Based Proportional Resource Sharing Model

Market-based proportional sharing systems are quite popular in co-operative problem-solving environments such as clusters (in a single administrative domain). In this model, the amount of resources allocated to consumers is proportional to the value of their bids. The consumers are allocated credits or tokens, which they can use to have access to resources. The value of each credit depends on the resource demand and the value that other users place on the resource at the time of usage. Consider two users wishing to access a resource with similar requirements, but the first user is willing to spend 2 tokens and the second user is willing to spend 4 tokens. In this case the first user gets one third of the resource share whereas the second user gets two third of the resource share, which is proportional to the value that both users place on the resource for executing their applications.

2.3.2 Pricing Models

The price of a commodity is the price at which it can be bought or sold for immediate delivery to the buyer. This section discusses existing models for pricing resources in the grid computing environment.

2.3.2.1 A commodity Market (Flat or Demand and Supply driven Pricing) Model

In the Commodity market model, service providers specify their service price and charge users according to the amount of resources they consume. The pricing policy can be based on a flat model or the resource supply and demand. In the supply and demand model, services are priced in such a way that supply and demand equilibrium is maintained. In flat price model, once pricing is fixed for a certain period, it remains the same irrespective of service quality. It is not significantly influenced by the demand, whereas in a supply and demand model prices change very often based on supply and demand changes. In principle, when the demand increases or supply decreases, prices are increased until there exists equilibrium between supply and demand.

2.3.2.2 A Posted Pricing Model

The posted price model is similar to commodity market except that it posts offers long before scheduling. In the Posted Price Model, resource consumers share resources at the posted price. It would not spend time negotiating. The resource consumers share the resource at the posted price according to their plans and pocketbooks.

2.3.2.3 A Bargaining Model

In the bargaining model, providers and consumers negotiate for resource access cost and time that maximizes their objectives. The negotiation happens privately between a consumer and a provider and there is no way for a consumer to know how much others value the resource services. Accordingly, the consumers need to decide whether to accept/reject an offer depending on its private objective function.

2.3.3 Bartering or Exchange Model

A barter is a mechanism in which a participant offers a valuable resource(s) to peers in order to receive a valuable resource(s) from them. It is an economic model in which goods or services are directly exchanged for other goods and/or services. A barter model for resource sharing can enable people or computers to directly get something in return for letting their resources to be used by others.

2.3.3.1 A Community, Coalition or Bartering Model

In the community, coalition or bartering model, a group of individuals shares each other's resources to create a Co-operative computing environment. Those who are contributing their resources to a common pool can get access to the pool. A model can be employed for deciding how much share of resources, each contributor can get and can allow a user to accumulate credit for future needs. A system like Mojonation.net employs this model for storage sharing across the community network. This model works when those participating in the Grid have to be both service providers and consumers.

2.3.4 Bilateral Negotiation Model

Unlike the auction models which supports one to many negotiations, between a grid service provider (seller) and many grid service consumers (buyers), and reduces negotiation to a single issue value (i.e. price), the bilateral negotiation model involves two parties, multiple issues value scoring model as discussed by (Haifa, 1982). The model is for bilateral negotiations about a set of quantitative variables. In a two party negotiation sequence called negotiation thread, offers and counter-offers are generated by linear combinations of simple functions called tactics. Tactics generate an offer and counter-offer considering multi criterions such as price, quantity, delivery time, resources, etc. To achieve flexibility in negotiation, agents may wish to change their ratings of the importance of the different criteria, and their tactics may vary over time. Strategy is the term used to denote the way in which an agent changes the weights of tactics over time. Through strategy an agent changes the weight of tactics over time. Strategy combines tactics depending on the negotiation history.

Definition 2.3.2: (Negotiation issues value scoring)

We defined $i \in \{a,b\}$ as the negotiation agents and $j \in \{1, \quad . \quad . \quad . \quad n\}$ the issues under negotiation. Let $x_j \in [\min_j, \max_j]$ be a value for issue j. Each agent has a scoring function $V_j^i : [\min_j, \max_j] \rightarrow [0, 1]$ that gives the score agent i assign to a value of issue j in the range of acceptable values. For convenience, scores are kept in the interval $[0, 1]$. w_j^i is the importance of issue j for agent i. The weight for all agents are normalized, i.e. $\sum_{j=1}^{n} w_j^i \ = \ 1$, for all i in $\{a,b\}$. An agent's scoring function for a contract – that is for a value $x \ = \ (x_1, \ . \ . \ . \ x_n)$, is defined as:

$$V^i(x) \ = \ \sum_{j=1}^{n} w_j^i V_j^i(x) \qquad (1)$$

Definition 2.3.3 (Negotiation thread)

The moment the negotiating agents have agreed on the set of variables over which they will negotiate, the negotiation process between two agents is made of an alternate exchange of offers and counter offers of values for those variables. This continues until an offer or counter offer is accepted or an agent terminates negotiation.

We defined $x_{a \rightarrow b}^t$ as the value of the offer proposed by agent **a** to agent **b**, and $x_{a \rightarrow b}^t[j]$ as the value of negotiation issue j proposed from **a** to **b** at time t.
A negotiation thread between agents *a and b* at time t_n is a finite sequence of proposals from one agent to the other ordered over time in the form:

$$x_{a \leftrightarrow b}^{t_n} \ = \ (x_{a \rightarrow a}^{t_1}, x_{b \rightarrow a}^{t_2}, \ . \ . \ . \ ) \qquad (2)$$

In a negotiation thread, the negotiators are out to maximize their payoffs while ensuring that an agreement is reached. This can be computed according to a negotiator's private utility function developed based on multi-attribute theory. As it is indicative in the Multi-attribute Theory, negotiation can range over a number of quantitative and qualitative issues. Quantitative issues in negotiation are defined over a real domain i.e.

$$x[j] \in D_j \quad = \quad < q_1, \quad q_2, \quad . \quad . \quad . \quad q_n > \tag{3}$$

When an agent receives an offer

$$x \quad = \quad x[1], \quad x[2], \quad . \quad . \quad x[n] \ , \tag{4}$$

where n is the total number of issues, it rates it by using a function that combines the scores of the different issues by a linear combination:

$$V^a(x^t) \quad = \quad \sum_{j=1}^{n} w_j^a V_j^a(x^t[j]) \tag{5}$$

Where $w_j^a(t)$ is the importance of issue j for agent a at time t. The weights for all agents are normalized, i.e.

$$\sum_{j=1}^{n} w_j \quad = \quad 1. \quad \forall\, i \in \{a,b\}. \tag{6}$$

Each agent has a scoring function $V_j^a : D_j^a \rightarrow [0, 1]$ that gives the score agent *a* assigns to a value of issue j in the set of its acceptable value D_j^a. If the score of the received offer is greater than the score of the counter offer the agent would send at this point, then the offer is accepted. If the present constant deadline $(t_{\max}^a)$ at which the negotiation must have been completed by agent *a* is reached, the offer is rejected by *a* otherwise, a counter offer is sent.

Definition 2.3.4: (Offer)

We define a negotiation offer sent from agent **b** to agent **a** at time t as $x_{b \to a}^t$.The interpretation an agent *a* gives to the offer $x_{b \to a}^t$ sent at time $t \le t'$ can be formalized as follows:

$$I^a(t'.x_{b\to a}^t) \quad = \quad \begin{cases} reject & if \quad t' \ \ge \ t_{\max}^a \\ accept & , if \quad V^a(x_{b \to a}^t) \ \ge \ V^a(x_{a \to b}^{t'}) \\ 0 & \\ & \end{cases} \tag{7}$$

2.3.4.1 Service–Oriented Negotiation Model

In a service – oriented negotiations, the agents undertake two possible roles that are in conflict, the client and the server. Roman letters $c,\ c_1,\ c_2,\ .\ .\ .$ are used to represent client and $s,\ s_1,\ s_2,\ .\ .\ .$ are used for server agents. The objective of the Clients are to maximize utility, to minimize service (resource) delivery time, minimize the cost of service (resource) consumed in terms of low price, and maximize quantity and quality of service (resource) purchased. On the other hand, the objective of the Server agent is to maximize profit (i.e. higher price) for larger quantity of services (resources) and later delivery date. The objectives of the negotiating agents are clearly conflicting.

Definition 2.3.5

In terms of negotiation values, the scoring functions of the Client and the Server agent show opposite tendencies for negotiation issue j,

$$\textit{if}\quad x_j, y_j \in [\min_j, \max_j] \quad \textit{and} \quad x_j \le y_j, \text{then}$$

$$V_j^{servert}(x_j) \le V_j^{server}(y_j) \Leftrightarrow \quad V_j^{client}(y_j) \le V_j^{client}(x_j).$$

2.3.5 Monopoly and Oligopoly

The negotiation models discussed so far assumed a competitive market setting where several Grid Service Providers and brokers/consumers determine the market price. However, there exist cases, where a single Grid Service Provider dominates the market and is therefore the single provider of a particular service. In economic theory this model is known as monopoly. Users cannot influence the prices of services and have to choose the service at the price given by the single Grid Service Provider who monopolizes the Grid marketplace. For example, a single site puts the prices into the Grid Market Directory or information services and brokers consult it without any possibility of negotiating prices.

The competitive markets are one extreme and monopolies are the other extreme. In most of the cases, the market situation is oligopoly, which is in between these two extreme cases: a small number of Grid Service Providers dominate the market and set the prices.

The competitive approach to resource sharing co-ordination is not discussed further. However, the next section covers in detail how we think co-ordination of resource sharing should be optimized via negotiation. First, various approaches to negotiation using a mediator are described. Second, three categories of negotiation tactics or strategies are discussed exhaustively.

2.4 Towards Appropriate Negotiation Model for Sharing Resources

The grid as a resource market is characterized by self-interested participants with conflicting objectives. For example, a grid resource provider would want to maximize profits as much as possible by selling its resources at highest possible price, and minimize costs to the barest minimum. On the other hand, a consumer of grid resources would want to pay the barest minimum price for a given resource. and maximize its utility as much as possible. The conflict of interest between the resource provider and the resource consumer has to be managed in order to realize an optimal resource allocation. This section covers potential negotiation models and strategies that can be used to achieve optimal resource allocation.

2.4.1 Mediator-Based Negotiation Model

In a grid market place. a consumer of resources usually contacts a resource provider directly. to negotiate for the needed resources. However. as the size of the grid grows. it becomes more difficult and more time consuming for the customer to negotiate directly with all the grid resource providers. Middle agents are introduced to mediate between the providers and customers so as to facilitate the resource allocation process. The most frequently deployed middle agents for mediating resource negotiation are the matchmakers. the brokers, and the market-maker(Bai et al.. 2006).

2.4.1.1 The Match-maker Model

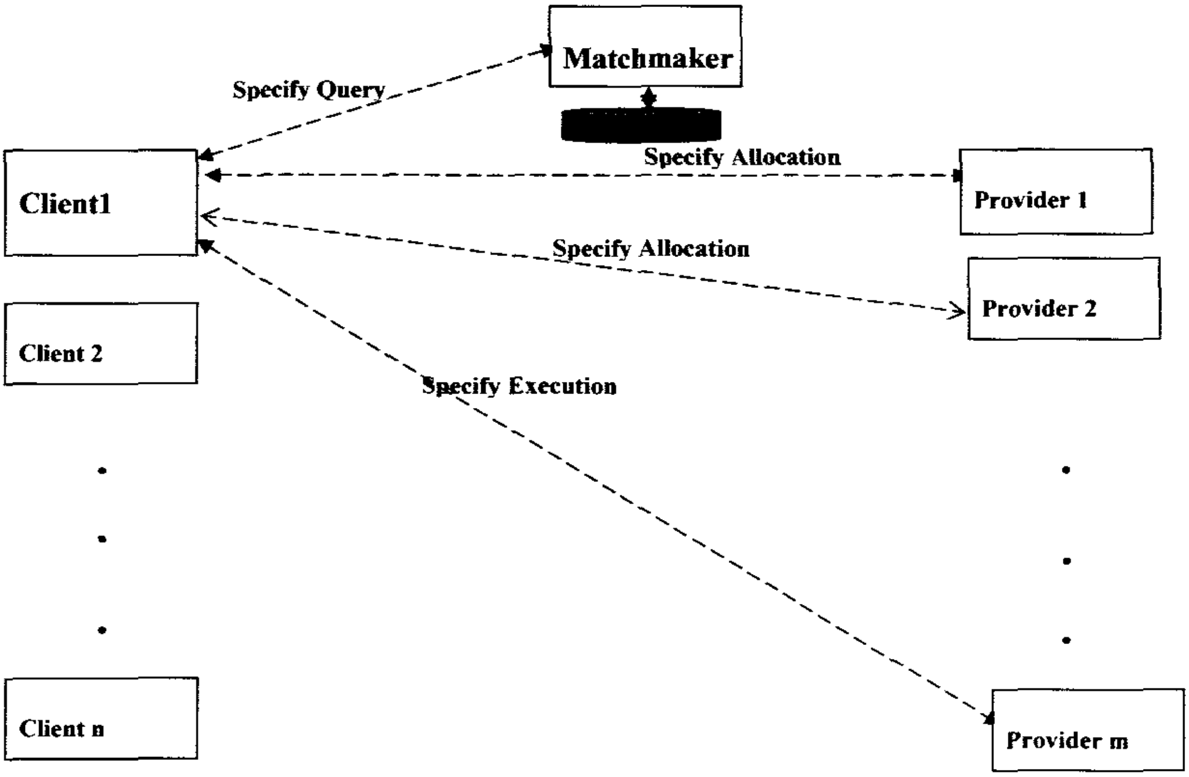


Figure 2. 1: The match-maker mediating negotiation between consumers of resources and Providers of resources

During a negotiation process, a matchmaker stands in between the resource consumers and the resources producers to mediate the negotiation process. The model (Figure 2.1) shows clients (consumers of resources) contacting the matchmaker agent (who maintains a knowledge base of the grid resources) for a collection of providers, which are a good match for the task of the client. The clients then start negotiations with all the returned matches. From all the returned offers, the client selects the one, which provides the highest satisfaction (considering its utility and price), and pays for the allocation of resources. The other negotiations are terminated. After allocation, the client immediately sends the task for execution. If no offers were returned, the client immediately restarts the process of searching for providers from the matchmaker. After the termination of a task, the client immediately moves to the next task. The shortcoming of this model results from the fact that the consumers (clients) have extensive responsibilities of directly contacting and negotiating with the providers.

2.4.1.2 The Broker Agent Model

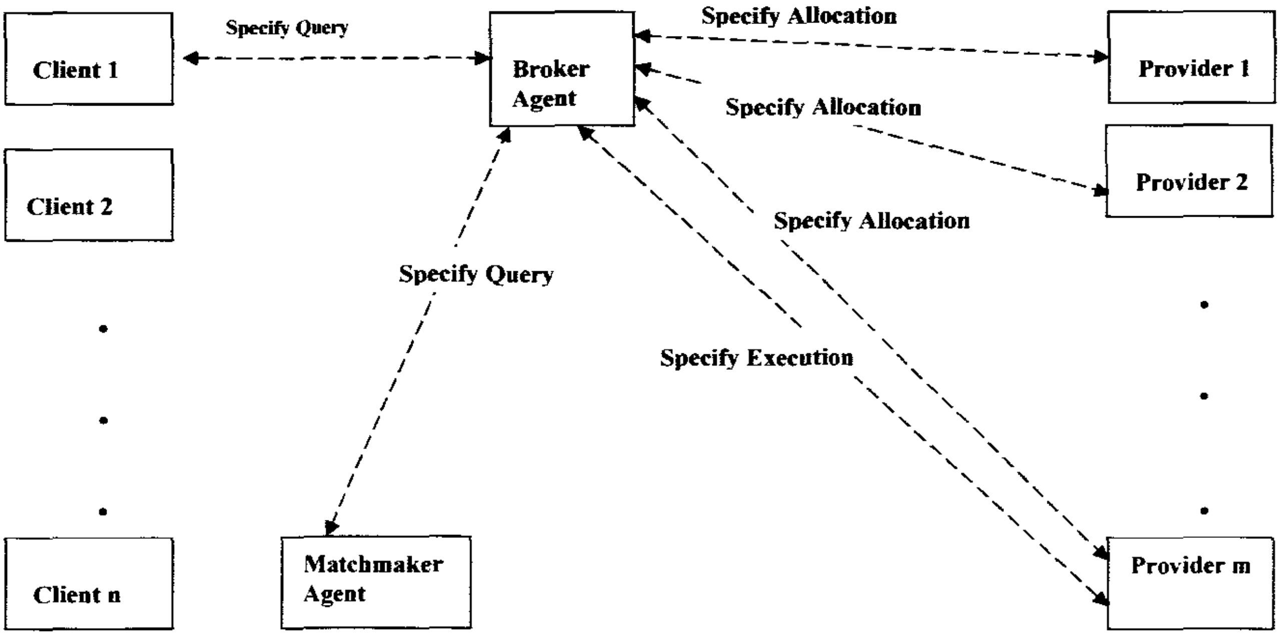


Figure 2. 2: The broker agent mediating resource negotiation between consumers of resources and Providers of resources

The Broker Agent Model (figure 2.2) shows the clients querying a broker agent for resources. The broker agent in turn contacts the matchmaker agent for matches. and starts a set of negotiations with the returned providers. The shortcoming of this model results from poor bandwidth utilization due to brokers messaging overhead.

2.4.1.3 The Market-maker Model

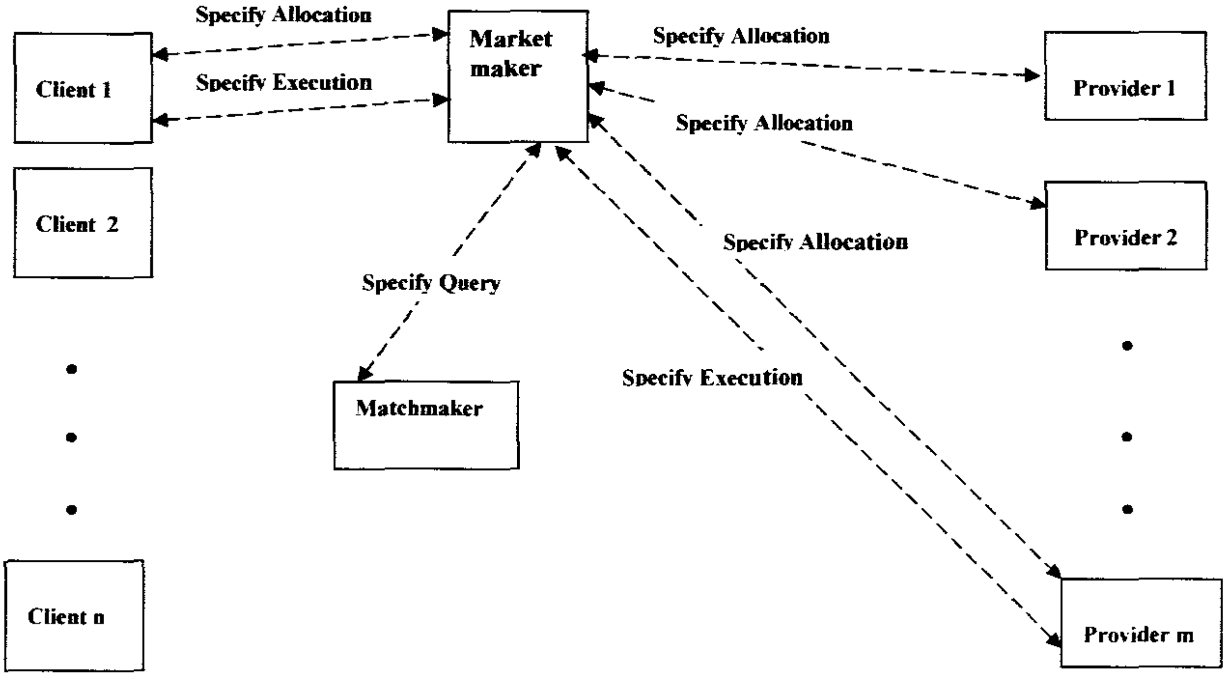


Figure 2. 3: The market-maker agent mediating resource negotiation between consumers and providers of resources (**Bai, et al. , 2006**).

The market-maker negotiates and buys resources and resource reservation in large quantities from the providers of resources and re sells them to the clients. Thus a market-maker acts as a "Super-provider" which aggregates resource reservations. When the client consults the market-maker, the market-maker checks if it can satisfy the request, and generates an offer for the resource allocation. While the broker agent only passes a filtered set of offers to the client. The market-maker's offer is decoupled from the price at which it acquired the resources.

2.4.2 Negotiation Tactics

Presented in this section, is a comparative study of families of negotiation tactics that negotiation agents can adopt to generate offers and counter-offers during a negotiation process. These tactics have been categorized as time-dependent, resource-dependent, and imitative (Axelrod, 1984; Deveaux et al., 2001; Faratin et al., 1997; Matos et al., 1998; Pruit ,1981; Raiffa, 1982;).

2.4.2.1 The Time - dependent Tactic

In a Time-dependent negotiation tactic, an agent has a time deadline by which an agreement must be reached. This tactic model the fact that the agent is likely to concede more rapidly as the deadline approaches. Using this tactic, negotiating agents vary their proposals and counter proposals as the deadline time approaches. The time-dependent tactic uses a function depending on time that can be parameterized.

Researchers on human negotiation have determined that time is an important factor in negotiation. The amount of time elapsed from the beginning of a negotiation creates a time pressure, which forces negotiator to reach agreements quickly. Two effects of time pressure are lower demands and faster concessions (Pruitt, 1981). When negotiators want to end the negotiations quickly, they have to sacrifice by accepting deals that are more favorable to their opponents, (Raiffa, 1982) states that as negotiators are willing to wait longer and to appear less eager for reaching agreements, they achieve higher profits. According to (Pruit, 1981; Raiffa, 1982), there are two types of time dependent negotiation behavior namely: Boulware and Conceder. A negotiation with Boulware behavior stays firm at the beginning of a negotiation, while their concession rates become flattened as the time limits are approached, while a Conceder behavior urges to make a deal and reaches its reservation value quickly.

Time-depended behavior can be realized in several different ways by choosing the function $\alpha_j^a(t)$.

The function must satisfy the following constraints:

$$\left.\begin{aligned} &0 \le \alpha_j^a(t) \le 1 \\ &\alpha_j^a(0) = k_j^a \\ &\alpha_j^a(t_{\max}^a) = 1 \end{aligned}\right\} \qquad (8)$$

That means that every negotiation offer will satisfy the range of values for a negotiation issue. As the initial negotiation offer has to be in the range of acceptable values for a negotiation issue, a constant k_j^a is used to adjust the initial value. At the initial time this constant is used to generate the initial offer. The reservation value is offered after the deadline was reached. The reservation value is the smallest result of the score function V_j^a for an issue *j* of agent *a*. In order to satisfy these constraints two classes of functions are presented:

2.4.2.1.1 Polynomial Function

$$\alpha_j^a(t) = k_j^a + (1 - k_j^a)\left(\frac{\min(t, t_{\max})}{t_{\max}}\right)^{\frac{1}{\beta}} \qquad (9)$$

2.4.2.1.2 Exponential Function

$$\alpha_j^a(t) = e^{\left(1 - \frac{\min(t,\ t_{\max})}{t_{\max}}\right)^{\beta} \ln k_j^a} \qquad (10)$$

The parameter $\beta \in \Re^{+}$ is used in equations (10) and (11) to adjust the convexity degree of the curve. As there is an infinite number of possible β the same applies to the amount of possible tactics. For values of β < *1* the behaviour of both exponential and polynomial functions can be described as Boulware, i.e., concessions are only made when the deadline is almost reached otherwise the proposals are only slightly changed. With β > *1* the behaviour of both classes of functions can be described as *conceder*. An agent prepared like this urges to make a deal and reaches its reservation value quickly. The difference of both function classes in *conceder* behaviour is that polynomial functions generally concede faster at the beginning. In *Boulware* behaviour polynomial functions also concede earlier.

2.4.2.2 Resource - dependent Tactics

Resource-dependent tactics model the pressure in reaching an agreement that limited resources (e.g. money, labor, raw material, or any other) and the environment (e.g. number of clients, number of servers, or economic parameters) impose upon the agent's behavior. These tactics allow agents to vary their proposals based on the quantity of available resources. Resource-dependent tactics are similar to the time-dependent ones, except that the domain of the function used is the quantity of a resource other than time. This is done either by making the deadline time dynamic or by making the function depend on an estimation of the amount of the resource.

Resource-dependent negotiation tactics vary the agents' proposals based on the quantity of available resources. These tactics are similar to the time-dependent ones, except that the domain of the function used is the quantity of a resource other than time. This is done either by making the time deadline dynamic or by making the function depend on an estimation of the amount of the resource.

2.4.2.2.1 Dynamic – deadline Tactics

This tactic sub-family considers a dynamic deadline by varying the agent's deadline according to the availability of a particular resource. The resource modeled here is the number of agents that are negotiating and for how long, that is, the average length of the active negotiation threads. If a selling agent a notices many interested parties for its good then there is no need to urge for an agreement. The deadline has influence on agent *a*'s decisions because it probably does better to hurry up when time is almost over. The set of agents negotiating with agent a at time *t* is

$$N^{a}(t) \;=\; x^{t}_{i \leftrightarrow a} \text{ is active} \qquad (11)$$

A dynamic time deadline can be defined as:

$$t^{a}_{\max} \;=\; \mu^{a}_{\max} \;=\; \frac{\mu^{a}\left|N^{a}(t_c)\right|^{2}}{\sum_{i}\left|x_{i \leftrightarrow a}\right|} \qquad (12)$$

Where:

μ^a is the time agent *a* assumes to be needed to negotiate with another agent

$|x^{t_c}_{i \leftrightarrow a}|$ is the length of negotiation thread between agent *a* and agent *i*.

2.4.2.2.2 Resource – estimation Tactics

Resource estimation tactics are based on measuring the quantity and availability of a resource at a time *t*. Yet an agent has to be prepared for a compromise when competition is hard, the resources are very limited or time is running out. Boulware behavior, in the presence of a large amount of resources, should change to *conceder* behavior when resources run short. Function α can be used to model this behavior changing process:

$$\alpha_j^a(t) = k_j^a + (1 - k_j^a)\, e^{-resource(t)} \qquad (13)$$

The function *resource* is here used to evaluate the amount of available resources at time *t*. For instance:

$$\mathrm{Re}\,source(t) = |N^a(t)| \qquad (14)$$

$\mathrm{Re}\,source(t) = |N^a(t)|$ models interested parties and reduced pressure for agreement for a larger amount of interested agents

$$\mathrm{Re}\,source\ (t) = \mu^a \frac{|N^a(t)|^2}{\sum_i |x_{i\leftrightarrow a}^t|} \qquad (15)$$

models interested parties and length of negotiation threads, i.e., more agents reduce the pressure but long negotiation threads increase the pressure

$$\mathrm{Re}\,source(t) = \max(0, t_{\max} - t) \qquad (16)$$

models time as resource in a linearly decreasing way.

2.4.2.3 Behavior-dependent Tactics

In situations where the negotiation agent is not under great pressure to reach an agreement, the choice may be to use imitative tactics that protect the agent from being exploited by other agents. In this case the counter offer depends on the behavior of the negotiation opponent. Agents try to imitate the behavior of their opponents in some degree. Different types of imitation can be performed, based on the opponent's negotiation policy over a sequence of its proposals: proportional imitation, absolute imitation and averaged proportional imitation.

Behavior-dependent tactics try to imitate the behavior of the agent's opponents up to a certain extent. This can be useful once opponents will not be able to exploit the agent's strategy. These tactics imitate an agent opponent's behavior thus making counter-proposals being influenced by the opponent's former actions. There are three different ways of using imitating behaviors, assuming the negotiation thread:

$$\{\ldots, x_{b\to a}^{t_{n-2\delta}}, x_{b\to a}^{t_{n-2\delta-1}}, x_{b\to a}^{t_{n-2\delta-2}}, \ldots x_{b\to a}^{t_{n-2}}, x_{b\to a}^{t_n}\}$$

Where $\delta \geq 1$

2.4.2.3.1 Relative Tit-For-Tat

Tactics of this family imitate proportionally an opponent's behavior during the last n-1 steps. Of course there must be a history of at least $n > 2\delta$. The generated counter-proposal would be:

$$x_{a\to b}^{t_{n+1}}[j] = \min(\max(\frac{x_{b\to a}^{t_{n-2\delta}}[j]}{x_{b\to a}^{t_{n-2\delta-2}}[j]} x_{a\to b}^{t_{n-1}}[j], \min{}_j^a), \max{}_j^a) \qquad (17)$$

2.4.2.3.2 Random Absolute Tit-For-Tat

Tactics of this family imitate in absolute terms the opponent's behavior. This tactic requires the existence of a history of at least $n > 2\delta$. The resulting counter-proposal is:

$$x_{a\to b}^{t_{n-1}}[j] = \min(\max(x_{a\to b}^{t_{n-1}}[j] + x_{b\to a}^{t_{n-2\delta}}[j] - x_{b\to a}^{t_{n-2\delta-2}}[j] + (-1)^s R(M), \min{}_j^a), \max{}_j^a) \qquad (18)$$

Where:

$$s = \begin{cases} 0, & V_j^a \; decreasing \\ 1, & V_j^a \; increasing \end{cases} \qquad (19)$$

Parameter *R(M)* provides a way to overcome function's local minima. Let *M* be the threshold at which an agent cancels its imitative behavior. The function *R(M)* returns a random integer in the space *[0, M]*.

2.4.2.3.3 Averaged Tit-For-Tat

Respectively to a certain number of an opponent's proposals the proportional average is calculated. The parameter γ describes the number of the past actions that are considered. The counter-proposal obtained is:

$$x'^{t_{n+1}}_{a \to b}[j] = \min\left(\max\left(\frac{x^{n-2\gamma}_{b \to a}[j]}{x_{b \to a}[j]} x'^{t_{n-1}}_{a \to b}[j], \min^{a}_{j} \right), \max^{a}_{j} \right) \qquad (20)$$

Where $n > 2\gamma$

The behavior of averaged Tit-For-Tat is similar to relative Tit-For-Tat with $\delta = 1$. when choosing $\gamma = 1$.

2.2.5 The Argument against Market-Maker Model

Even though the market-maker was considered to be best, and has a smaller number of messaging overhead as compared to the broker and the matchmaker models, the following however, outlined the shortcomings of a market-maker model:

i. a market-maker's offer of resources is usually decoupled from the price at which it acquired the resource. That is to say that a market – maker model does not permit the users of grid services to influence the prices of services. The users have to choose the service at the price given by the single Grid Service Provider who monopolizes the Grid marketplace.

ii That, in a way implies that a market-maker has monopolistic tendencies.

iii the classical problem of a monopoly is that it sets higher price than marginal cost and this distorts the trade-off in the grid economy, and moves it away from Pareto efficiency.

iv. the fact that a monopoly does not face the discipline of competition means that a monopoly may operate inefficiently without being corrected by the grid marketplace.

We conclude from the above reasoning that the market-maker agent is not efficient for resource allocation.

In order to overcome the weaknesses of the market-maker model, we introduce a new middle agent called the Co-operative Modeler.

CHAPTER THREE

MODELS OF THE GRID AS A RESOURCE MARKET

3.1 Introduction

This chapter discusses the architecture of the proposed Co-operative Modeler for mediating grid resource sharing negotiation. Using formal notations, first, we constructed a theoretical model for allocation of resources to client's task with the assumptions that: (a) each grid resource type is made of different variants. The resource variants are differentiated in terms of the quality of the resource, the capacity (size) of the resource, the response time of the resource, and the cost of the resource, to name a few.; (b) the importance attached to each resource variants differ from one client to another, (c) the quantity of resource allocated to a client's task can not be more than the available resource. Second, we crafted a model of negotiation offers of agents as multi-objective optimization problem. Third, the formal aspect is concluded with the Bayesian Learning Algorithm. The chapter also discussed the software specification and design process as a prelude to implementing a prototype of the Co-operative Modeler.

3.2 Co-operative Modeler for Resource Negotiation

The purpose of creating a novel middleware agent, called Co-operative Modeler is for mediating resource negotiation so as to lower cost and maximize resource utilization. It also serves the need for an implementation prototype that can be studied and analyzed to understand resource negotiation.

In order to nullify the monopolistic influence of the market-maker (Figure 2.3) in the grid economy, providers of resources collaborate to form a Co-operative group for the purpose of trading their resources. A Co-operative is a virtual organization, made of distributed, autonomous, service providers, united voluntarily to meet their common economic, social, and cultural needs and aspirations through a jointly owned and democratically controlled virtual enterprise. Co-operative businesses are business-owned and controlled by the people who use their services. By pooling resources together in a Co-operative manner, providers of resources combine sales returns, and operating expenses, and distributing sales among members in proportion to volume each provides to the Co-operative.

The Co-operative Modeler depicts practically the concept of Utility Grid which derives from utility computing where clients can request for resources whenever needed (on-demand) and only be charged for the amount used. The Utility Grid is a typical infrastructure for utility computing to take place. Clients need not own expensive resources for specific project but can instead choose to "rent" or share (especially among trusted parties with sharing agreements in place), the required resources for a limited time and this is expected to be cheaper than buying the actual resource. Grid resources, however, are not free. Users of the resources have to negotiate with the resource providers, the terms of accessing their resources.

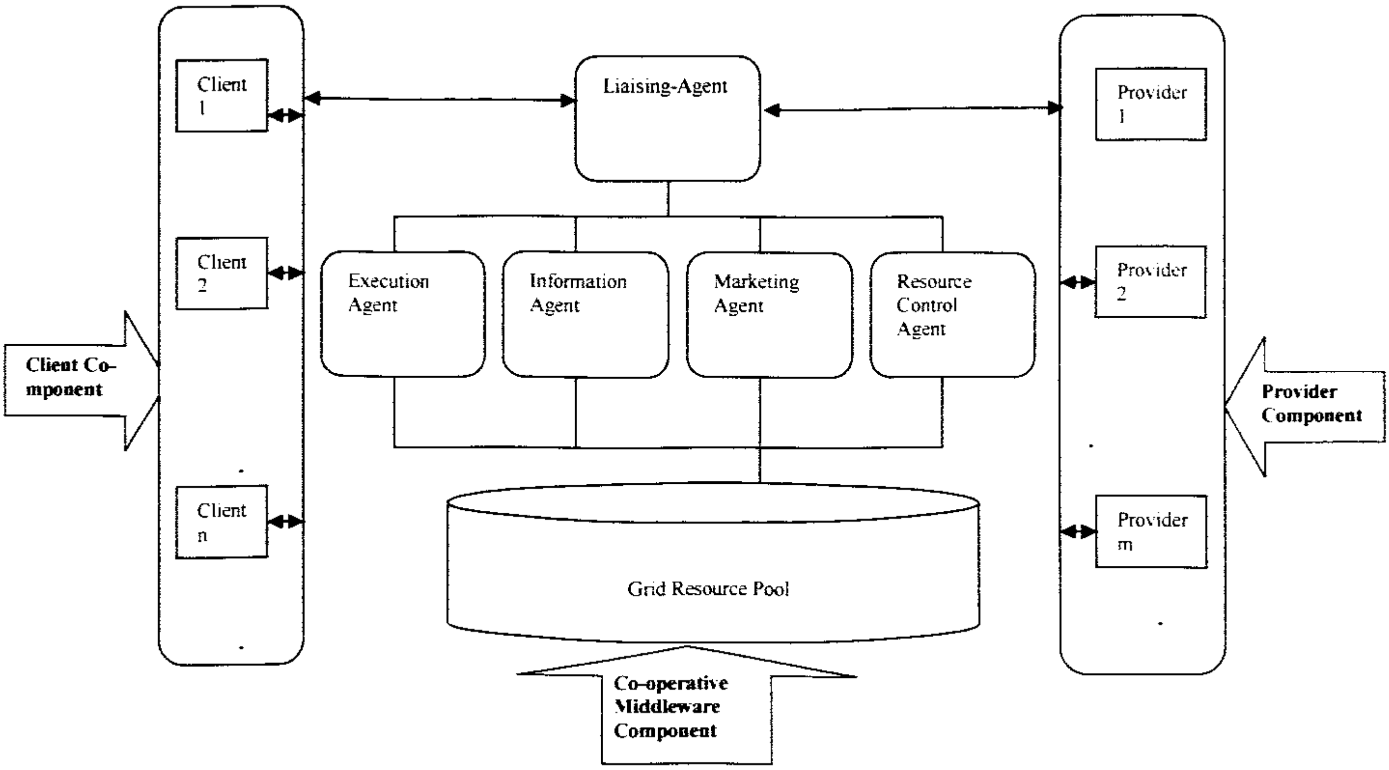


Figure 3. 1: The Architecture of the Co-operative Modeler (Aremu, & Adigun, 2007).

The Co-operative Modeler consists of autonomous resource providers that are distributed across multiple organizations and administrative domains, enabled by Utility Grid Computing Infrastructure, and linked together in a Co-operative relationship to exchange their collective resources in form of shared services. Therefore it models a dynamic alliance of virtual organizations bringing their complementary competencies and resources that are collectively available to each other, with the aim to deliver products or services to the market as a collective effort. The end user need not circle round each service provider for services, but consult the grid marketplace for the needed resources. The architecture of the Co-operative Modeler is made up of three components [Figure 3.1] namely: the Client component, the Co-operative Middleware component, and the Providers component. The architecture promotes a business situation involving three stakeholders with three major business roles: (i) The end-user role which is played by a stakeholder (client) who consumes services, (ii) The Mediator role, which is the key player and this role is played by the Co-operative Middleware agents, (iii) The service Provider role is played by a service owner who offers his resources as services to the end user (the client).

Table 3. 1: Role Definition of the Co-operative Middleware agents

Name of Agent	Responsibility of the agent
Client-Agent	This agent is acting on behalf of the Clients (the end user) to negotiate for resources
Liaising-Agent	This agent is acting as the controller, manager of managers to the other agents at the resource pool. Request for resources is always directed to it. It liaises between the clients, the providers and the agents at the resource pool.
Information-Agent	This agent is the manager in charge of the resource pool. It maintains knowledge base of the resources at the pool. It interacts closely with the pool and the Marketing-Agent.
Marketing-Agent	The Marketing-Agent is the expert (manager) in charge of resource negotiation. It is equipped with all kinds of negotiation tactics so as to optimize the objectives of the resource providers.
Resource-Control-Agent	This agent is responsible for sales recording/documentation. and issuance of resource id. It is the manager (accountant) in charge of record keeping at the resource pool.
Execution-Agent	This agent is responsible for the execution of the clients' tasks.

The Client Component is made up of a set of n clients. Each client has at least one task of various length and resource requirement to execute. The Provider Component is composed of a set of m Providers, who formed the Co-operative group. The dynamic nature of this model makes it possible for a provider of resources to be a client requesting for resources at the same time. The Co-operative Modeler is the co-ordination center with responsibilities distributed among five interacting agents each contributing to the negotiating of resource sharing of grid resources. These agents are: the Liaising-Agent, the Information-Agent, the Marketing-Agent, the Resource-Control-Agent, and the Execution-Agent, whose roles are defined in Table 3.1.

The interaction of agents during the process of negotiating resource sharing can be very complex. However, the flow of information among the agents is better illustrated via an algorithm as follows:

1. Client-Agent queries the Liaising-Agent for resource(s),
2. The Liaising-Agent passes the query to Information-Agent,
3. The Information-Agent searches the pool for resource that match the query.
4. If there is resource(s) that match the query, then,

    4.1 the Information-Agent fetches the list of resource(s) together with the reservation values of their providers, and then passes same to the Marketing-Agent.

    4.2 the Marketing-Agent adjusts the reservation value of each of the resources to arrive at their negotiation values, and then passes on the resource list to the Client-Agent for negotiation.

    4.3 the Client-Agent selects a resource to buy and then start to negotiate with the Marketing-Agent.

    4.4 IF Reservation_value_Of_Negotiation_Issue_MarketingAgent ≤ Reservation_value_Of_Negotiation_Issue_MarketingAgent

        4.4.1 a message specifying acceptance of sales will be passed from the Client-Agent to the Marketing-Agent.

        4.4.2 the Client-Agent is charged for the resource.

        4.4.3 payment is made to the Marketing-Agent.

        4.4.4 the control is then transferred to the Resource-Control-Agent to document sales.

```
                4.4.5  the Resource-Control-Agent confirms/documents the
                       sales made, issues a Resource-ID and sends the resource
                       id through the Marketing-Agent to the Client-Agent.
                4.4.6  the Client-Agent receives the Resource-Id and passes
                       same to the Client for subsequent task execution.
            Else
                4.4.8  The zone of agreement does not exist
                4.4.9  No deal
                4.4.10 Negotiation ends in conflict.
            End IF
 Else
      4.5 Report there is no resource that matches the query request.
End if
```

3.3 The Resource Allocation Model

In a grid marketplace, it is assumed that a client requesting for resources has one or more task(s) to achieve. Each client's task requires one or more resources to achieve them. The theoretical model being formulated in this section addresses the problem of assigning the right resource to each client's task so that the utility of the client can be maximized. In formulating the theoretical model, we made the following basic assumptions: (i) each resource type consumed by a client's tasks is made of different variants. The resource variants (extra-functional properties) may be in terms of the quality of the resource, the capacity (size) of the resource, the response time of the resource, and the cost of the resource; (ii) the importance attached to each resource variant differs from one client to another, (iii) the quantity of resource allocated to a client's task can not be more than the available resource.

Table 3. 2: The Description of the Co-operative Modeler notations for resource allocation

Symbol	Meaning	Limit	Constraint
CA_i	Client-Agents	I	
T_n	Tasks	N	$N \geq I$
$N(i)$	# Tasks for Client-agent i	I	$N = \sum_{i=1}^{I} N(i)$
b_{in}	Set to 1 if task n is used for client-agent i	IN	$\forall_i \forall_n x_{in} \in \{0,1\}$
R_m	Amount available of resource dimension m	M	
r_{nm}	For each task n, the resource usage for resource m	NM	
P_k	Extra – functional properties	K	
P_{nk}	For each task n, the amount of extra – functional properties k	NK	
f_i	Weighted client preferences for extra – functional properties	K	$\sum_{k=1}^{K} f_k = 1$
u_n	Utility for n tasks	N	$\forall_n u_n \in [0,1]$
T_s	Total utility for the combination of one task per client-agent	$S = \prod_{i=1}^{I} N(i)$	$\forall_s T_s \in [0,1]$
x_n	Set to 1 if this client-agent task is used, otherwise 0	N	$\forall_n x_n \in \{0,1\}$

Definition 3.3.1: Client-Agent Total Task

Suppose CA_i (i = 1, 2, . . . I), is a set of Client-Agents. Let Client-Agent CA_i have $N(i)$ tasks to execute. The total number of Client-Agent tasks T_n, is

$$T_n = \sum_{i=1}^{I} N(i) \tag{21}$$

That means the total number of Client-Agents' tasks is the sum of the individual Client-Agent's tasks. Each Client-agent task consumes resources from some of the M resource dimension R_m Examples of resource dimensions are processing requirements such as average instructions per second, disk and memory storage requirements and network bandwidth requirements.

Task T_n consumes the resources r_{nm} for each of the M resource dimensions. For each resource dimension R_m the total resources consumed must be less than the available resources for all resource dimensions:

$$\sum r_{nm} x_n \quad \leq \quad R_m, \quad m \; = \; 1, \; . \; . \; . \; M \qquad (22)$$

In this equation, R_m is the amount of available resources of dimension m. x_n is a decision variable which is set to 1 if this client-agent task is used for this client-agent. As an additional constraint only one client-agent task shall be selected for each client-agent:

$$\sum_{n=1}^{N} b_{in} x_n \quad = \quad 1, \quad i \; = \; 1, \; . \; . \; ., \; N \qquad (23)$$

Each of the Client-Agent tasks satisfies one or more extra-function properties (non-functional requirements). By this we mean that each resource type consumed by client's tasks is made of different variants (extra-functional properties).

In this model we assume that there are K extra-functional requirements P_k. The Client-Agent weighs the importance of the extra-functional requirements according to the preference function f_k.

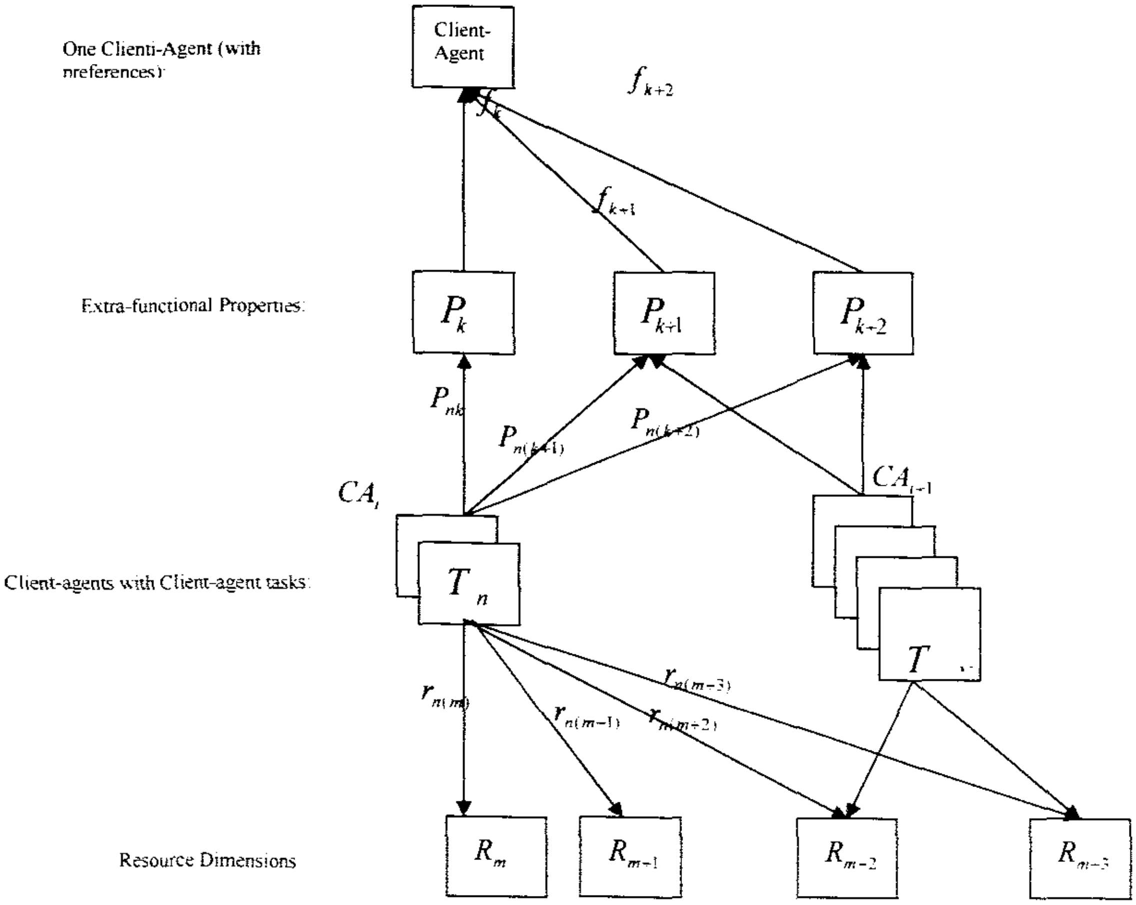


Figure 3. 2: Client task consumes resources and results in extra functional properties

Based on the amount of the extra – functional properties and the client preferences, a utility u_n is calculated for each client-agent task as shown in figure 3.2. This utility is between 0 and 1. This model addressed the question: "what resource, considering all the different resource variants, should be assigned to client's task so that the utility of the client be maximized?"

Putting this formally, we have:

$$\text{Maximize } \sum_{n=1}^{N} u_n x_n \qquad (24)$$

Subject to

$$\sum_{n=1}^{N} r_{nm} x_n \;\leq\; R_m, \quad m = 1, \;.\;.\;.\; M \qquad (25)$$

$$\sum_{n=1}^{N} b_{in} x_n \;=\; 1, \quad i = 1, \;.\;.\;.\; N \qquad (26)$$

This formulation (24) expresses that the total utility shall be maximized. (25) is a set of M resource constraints one for each resource dimension. (26) is a set of N partition constraints, expressing that one and only one client task shall be selected per client-agent.

3.4 Resource Usage in a Typical Grid Market

The purpose of this section is to discuss the transient nature of the states of grid resources and to specify how these resources dynamically move from one state to another before, during and after a negotiation process up to execution stage.

Resources in the market can assume the same states as Grid Resources as shown in Table 3.3.

Table 3. 3: The States of Grid Resources (Bai, et aL, 2006).

State	Explanation
Free	This implies the resources are not used and available for allocation
Reserved	This means that the resources are reserved for future allocation for the time of the negotiation. This state is strictly tied to a conversation and the user. The role of this state is to prevent multiple allocations of the same resources.
Committed	This implies that the resource is allocated to the user for future use, but they are not in current use.
In Use	This is an indication that the resources are currently in use by a task.
Expired	That is to say that the resource usage chunk occurs in the past ($t_{end} \leq t$), these allocation units might be kept around for accounting purposes, but they will be eventually garbage collected.

These states and the transition possibilities are represented in the state machine state machine diagram (figure 3.3) illustrating how the states transit from one state to another state during a negotiation process. The diagram shows that a grid resource can assume any of the following five states: Free, Reserved, Committed, In-Use states, and Expired states.

Before negotiation commences, a resource assumes a free state. The interpretation of free state is that, the resources are not being used presently, and are available for allocation.

Just at the commencement of a negotiation, the resource state changes to reserved state. This means that the resources are reserved for future allocation at the time of the negotiation. The role of this state is to prevent multiple allocations of the same resources.

At a point during a negotiation process, when the offer of the buyer agent is acceptable to the selling agent, the buyer agent would pay for the transaction, and the resource is allocated to the buyer agent for future use. The state of the resource changes at this point from Reserved to Committed state. In other words, the committed state implies that the resource is allocated to the user for future use, but they are not in current use and cannot be allocated to other agents because it has been paid for.

In-Use state indicates that the resources are currently in use by a task. That means that the resource is being used currently to execute the task of the agent.

The expired state indicates that the resource was in use in the immediate past ($t_{end} \leq t$), but garbage collection has not taken place. However, expired resources will soon be released and made available for other agents to use. When eventually released, the resource state goes back to a free- state.

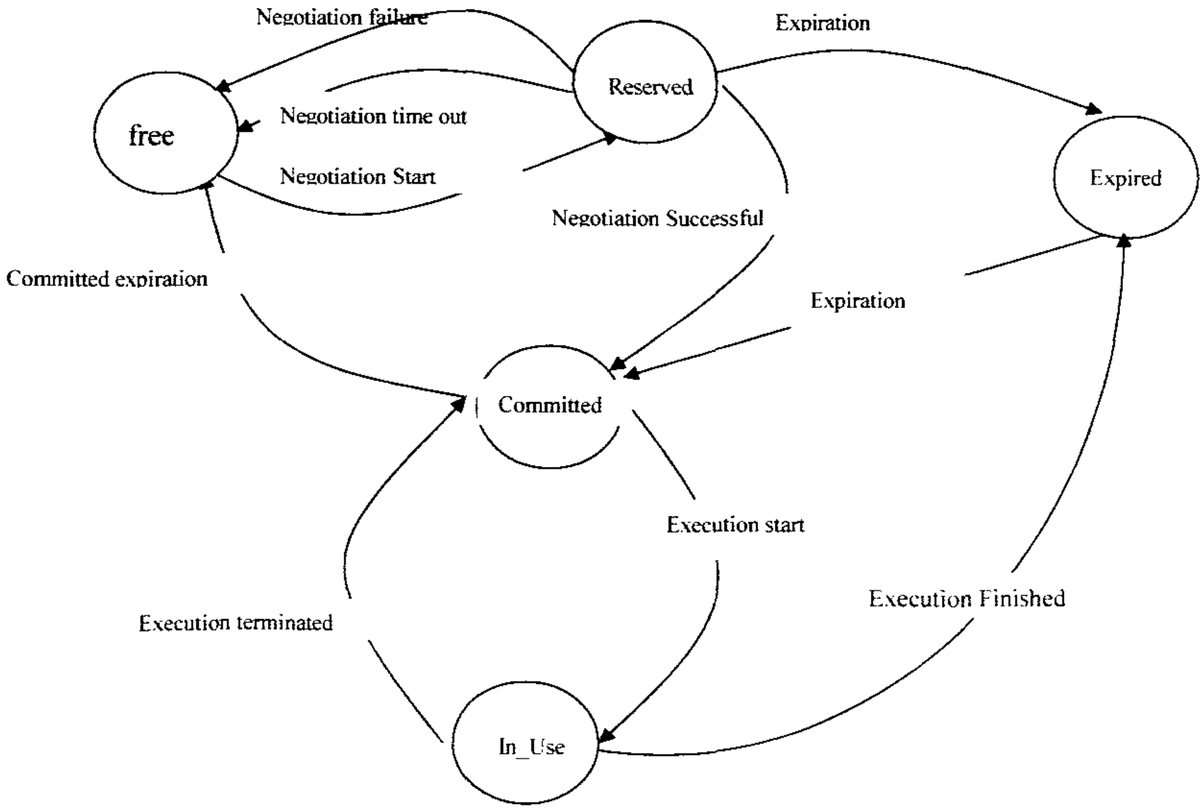


Figure 3. 3: The State machine describing the transitions between the various states of resource usage.

3.5 Optimizing the Conflicting Objectives of Negotiating Agents

The goal in a negotiation thread as agents make negotiation proposals and counter proposals is a search towards a negotiation offer that maximizes or minimizes the conflicting objectives of the negotiators. Therefore, we present the model of the conflicting objectives of the negotiating agents using Multi-attribute Utility Theory Approach. Then we introduce a novel **Combinatorial Multi-Objective Optimization** model (CoMbO) for capturing agents' negotiation offers. Finally, two learning algorithms, namely: Genetic and Bayesian learning algorithms for implementing the model are discussed and applied as appropriate.

3.5.1 Modeling Negotiation Offers Using Multi-attribute Utility Theory Approach

When a negotiation involves multiple issues, users might be interested in different issues. To capture user's preferences on different negotiating issues, agents require utility functions to determine how much they like each received offer (Maes, et al., 1999) and then to make counter offers by choosing the strategies that maximize the utilities. For example, services that clients buy usually differ considerably along dimensions of price, quality, brand appeal, warranty, and delivery time, but service consumers (clients) attach different utilities determined along specific dimensions of price, quality, brand appeal, warranty and delivery time with possible trade-offs. Multi-attribute utility theory helps to derive a single number that is indicative of joint utility based on different service characteristics. Given this single utility number, services can be ranked and the user advised accordingly. Multi-attribute Utility Theory can be used to rank different choices. (e.g. items to buy, movies to watch, and sports event to attend etc.).

There are various functions to represent utility as a real number; some of the popular utility functions being:

$$\left.\begin{array}{lcl} U(x) & = & \log(x+b) \\ U(x) & = & a+bx+cx^2 \\ U(x) & = & \frac{1}{k}(1-e^{-kx}) \end{array}\right\} \qquad (27)$$

Where U is the utility, x is the measure of the attribute, a, b, c, and k are constants. For a client wishing to buy a resource for example, x could be price, quality, delivery time etc. In case of multiple attributes, various additive and non-additive functions are used to obtain the joint utility. In the case of the client wishing to buy a service as earlier mentioned, such a function might appear as follows:

$$U(resource) = k_1U_1(delivery\ time) + k_2U_2(price) + k_3U_3(quality) \qquad (28)$$

Where the k's are weights for each negotiating issue and the U_s are utility functions. The k's and U_S are estimated from data sets, and the function obtained can be used to get a single utility number for a particular resource. The multi attribute utility theory consists of the user-specific weight of each negotiating issue and the utility of each negotiating issue. The weights represent the important levels of the issue to the users, and the utilities stand for the scores of the issues for the values in question. If a user has a higher concern on price than quality, a higher weight is placed on price than on quality.

3.5.2 Combinatorial Multi-Objective Optimization Model (CoMbO) for Capturing Negotiation Offers

The purpose of this section is to model agents' negotiation offers as an optimization problem, and to discuss a Genetic algorithm for searching for a set of negotiation offers that would maximize or minimize the conflicting objectives of the negotiators.

Definition 3.5.1: (Negotiation Offers)

A negotiation thread has been defined as a sequence of offers and counter – offers in a two party negotiation.

A negotiation offer $o = < d_{a_1}, d_{a_2}, \ . \ . \ . \ d_{a_n} >$ is a tuple of attribute values (interval) pertaining to a finite set of attributes $A = \{a_1, a_2, \ . \ . \ . \ a_n\}$. An offer can also be viewed as a vector of attribute values in a geometric negotiation space with each dimension representing a negotiation issue. Each attribute a_i takes its value from the corresponding domain D_{a_i}.

Generally speaking, a finite set of candidate negotiation offers o acceptable to an agent P (i.e satisfying its hard constraints) is constructed via the cartesian product $D_{a_1} \, x \, D_{a_2} \, x \ . \ . \ . \ x \, D_{a_n}$.

As human agents tend to specify their preference in terms of a range of values, a more general representation of an offer is a tuple of attribute value interval such as

$$o = \ < 10 - 50\,(Rand), 1 - 5\,(years), 10 - 30(days), 100 - 600(units) > \qquad (29)$$

One common way to quantify an agent's preference (i.e. the utility function U_p^o) for an offer o is by linear aggregation of the valuations:

$$U_p^o(x) = \sum\nolimits_{a \in A} U_p^A(a) \, x \, U_p^{D_a}(d_a) \qquad (30)$$

A, being a finite set of attributes, equation (30) now becomes

$$U_p^o(x) = \sum_{i=1}^{n} U_p^A(a_i) \, x \, U_p^{D_{a_i}}(d_{a_i}) \qquad (31a)$$

If we let $\mu_i = U_p^A(a_i). \, and \ u_i = U_p^{D_{a_i}}(d_{a_i})$, equation (31) reduces to:

$$U_p^o(x) = \sum_{i=1}^{n} \mu_i u_i \qquad (31b)$$

Where:

solution $x = \{x_1, x_2, \ . \ . \ . \ x_m\}$ is a vector of discrete decision variables, D is the set of solutions, U_i is vector of negotiation issue decision variables, μ_i is weight vector such that x is the unique global optimum.

Suppose we get every negotiation issues' utility functions from (31) as $\{u_1, u_2, \ . \ . \ . \ u_n \}$, the problem (31) is transformed into a general multi-objectives combination optimization problem described as a vector function u that maps a tuple of m parameters (decision variables) to a tuple of n objectives, formally described as:

$$\begin{cases} \max imize \quad y = U(x) = (u_1(x), u_2(x), \ . \ . \ . \ u_n(x)) \\ \\ \text{Subject to } x = (x_1, x_2, \ . \ . \ . \ x_m) \in D. \end{cases} \tag{32}$$

where $x = \{x_1, x_2, \ . \ . \ . \ x_m\} \in D$ is a vector of discrete decision variables,

D is the set of solutions $y = (u_1, \ . \ . \ . \ u_n) \in U$

and x is called decision (parameter) vector, D is parameter space, y objective vector and U objective space.

3.5.3 Optimization Problems

The formulated theoretical model (equation 32) represents a combinatorial multi-objective optimization problem. Optimization deals with the problem of seeking solutions over a set of possible choices to optimize certain criteria. Combinatorial optimization is a branch of optimization. Its domain is optimization problems where the set of feasible solutions is discrete or can be reduced to a discrete one. If there is one criterion to consider, it becomes a single objective optimization problem. The single objective optimization problem has been well studied. The problem becomes extremely difficult when a consideration of several conflicting objectives is required. In many cases, it is not likely that different objectives would be optimized by the same choices of decision variables.

In principle, multi-objective optimization problems are very different from single-objective optimization problems. In the single-objective case, one attempts to obtain the best solution, which is absolutely superior to all other alternatives. In the case of multiple objectives, there does not necessarily exist a solution that is best with respect to all objectives because of conflict among objectives. A solution may be best in one objective but worst in other objectives. Therefore, there usually exist a set of solutions for the multiple objective case which cannot simply be compared with each other. For such solutions, called non-dominated solutions or Pareto optimal solutions, no improvement in any objective function is possible without sacrificing at least one of the other objective functions. For a given non-dominated point in the criterion space D , its image point in the decision space D is called efficient or non-inferior. A point in D is efficient if and only if its image in D is non-dominated.

Multi-objective optimization (also known as multi-criterion, vector, or Pareto optimization) extends optimization theory by permitting several design objectives to be optimized simultaneously. Although the basic methods can be traced back to the work of Leibniz and Euler, the principle of multi-objective optimization was first formalized by Vilfredo Pareto, an Italian economist, whose theories (Pareto, 1906) are now considered the basis of modern welfare economics. He introduced the concept of the *Pareto optimum*, a standard of judgment in which the optimum allocation of the resources of a society is not attained as long as it is possible to make at least one individual better off in his own estimation while keeping others as well off in their own estimation.

Multiple objective optimization problems arise in the design. modeling and planning of many complex real systems in the area of industry production, urban transportation, capital budgeting, forest management, reservoir management layout and landscaping of new cities and energy distribution. Almost every important real-world decision problem involves multiple and conflicting objectives that need to be tackled while respecting various constraints. leading to overwhelming problem complexity (Gen & Chang, 2000).

Most multi–objective optimization methods use the concept called domination. In these methods, two solutions are compared on the basis of whether or not one solution dominates the other solution.

Definition 3.5.2 (Dominance and Pareto-optimality)

A decision vector $a \in D$ is said to dominate a decision vector $b \in D$ also written $a \succ b$ if and only if

$\forall i \in \{1,\ 2\ \ .\ \ .\ \ .\ \ n\}:\ f_i(a) \geq f_i(b)$

$\wedge \exists j \in \{1,\ 2,\ \ .\ \ .\ \ .n\}: f_j(a) > f_j(b)$.

Additionally, we say a covers b $(a \succ= b)$ if and only if $a > b$ *or* $f(a) = f(b)$. Based on this convention, we can define non dominated, Pareto-optimal solutions as follows:

Let $a \in D$ be an arbitrary decision (parameter) vector:

(a) The decision vector a is said to be non dominated regarding a set $D' \subseteq D$ if and only if there is no vector in D' which dominates a; Formally: $\exists a' \in D' : a' \succ a$

(a) The decision (parameter) vector a is called Pareto – optimal if and only if a is non-dominated regarding the whole parameter space D. Pareto-optimal parameter vectors cannot be improved in any objective without causing a degradation in at least one of the other objectives. They represent in that sense globally optimal solutions. We must note, however, that the optimal set does not necessarily contain all Pareto-optimal solutions in D. The set of objective vectors $f(a')$, $a' \in D'$ corresponding to a set of Pareto-optimal parameter vectors $a' \in D$ is called "Pareto – optimal front" or "Pareto front".

3.5.4 Solution to Multi-Objective Optimization Problems

Genetic algorithms have received considerable attention as a novel approach to solving multi-objective optimization problems, resulting in a fresh body of research and applications known as genetic multi-objective optimizations. Genetic algorithms find application in many NP-Hard and NP-complete optimization problems. Essentially those problems that do not solve nicely using strictly analytic methods and therefore search strategies are typically applied (Gen & Chang, 2000)..

Genetic algorithms (GA) mimic the biological processes underlying classical Darwinian evolution in order to find solutions to optimization or classification problems. Genetic Algorithm implementations utilize a population of candidate solutions (or chromosomes). Each candidate solution in the current generation is evaluated using a fitness function and ranked. From the ranking, candidates are selected from which the next generation is created. The process repeats until either the number of iterations is exceeded or an acceptable solution is found.

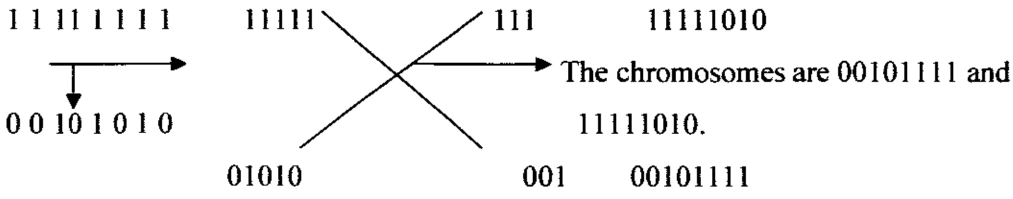
Definition 3.5.3 (Basic Concepts)
The fitness function is a measure of how well a candidate solves the problem at hand. For the Standard Genetic Algorithm, the fitness f is a function from the set of possible candidates to the positive real.

Selection is the process of deciding which candidate (chromosome) in the current population will pass their solution information to the next generation. There are several generic selection methods. Examples of selection methods include random. elitist. roulette wheel. tournament. etc.

Genetic operators provide mixing of candidate (chromosome) portions from the parent or parents to form the offspring of the next generation. Examples of genetic operators include crossover, mutation, inversion, etc.

Crossover is genetic operator used to vary the programming of a chromosome or chromosomes (candidate solution(s)) from one generation to the next. It is analogy to reproduction and biological crossover, a process of exchanging the genes between the two individuals that are reproducing. There are several such processes. but we will consider only two: (i) one-point crossover and (ii) two-point crossover.

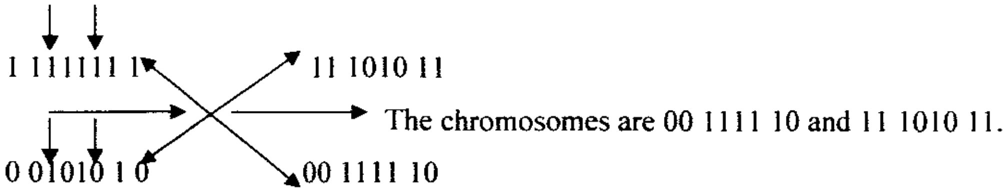
One-Point crossover is a process that is both standard and simple. A random integer i is selected uniformly between 1 and n. This is the place in the chromosome at which, with probability P_c, crossover will occur. If crossover does occur, then the chunks up to i of the two chromosomes are swapped. For example, the chromosome 11111111 when crossed with 00101010 at i = 4 gives



The chromosomes are 00101111 and 11111010.

Two – point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms rendering two child organisms.

For example, the chromosome 11111111 when crossed with 00101010 at $3 \leq i \leq 6$ gives



The chromosomes are 00 1111 10 and 11 1010 11.

Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible. Mutation as an important part of the search has helped to prevent the population from stagnating at any local optimum. Mutation occurs during evolution according to a user-definable mutation probability. This probability should usually be set fairly low (0.01 is a good first choice). If it is set too high, the search will turn into a primitive random search.

Mutation is the process of randomly altering the chromosomes. Say that P_m is the probability that bit i will be flipped. Let i vary from i to n. For each i a random number is selected uniformly between 0 and 1. If the number is less than $P_{m,}$ then the bit is flipped.

There are many types of mutation:

A Flip Bit Mutation: a mutation operator that simply inverts the value of the chosen gene, (0 goes to 1 and 1 goes to 0). This mutation operator can only be used for binary genes.

Boundary Mutation: a mutation operator that replaces the value of the chosen gene with either the upper or lower bound for that gene (chosen randomly).This mutation operator can only be used for integer and float genes.

Non–Uniform Mutation: a mutation operator that increases the probability that the amount of the mutation will be close to 0 as the generation number increases. This mutation operator keeps the population from stagnating in the early stages of the evolution then allows the genetic algorithm to fine tune the solution in the latter stages of evolution. This mutation operator can only be used for integer and float genes.

Uniform Mutation: a mutation operator that replaces the value of the chosen gene with a uniform random value selected between the user-specific upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

Gaussian Mutation: a mutation operator that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped it falls outside of the user–specific lower or upper bounds for that gene. This mutation operator can only be used for integer and float genes (Yonet & Eracar, 2003).

3.5.4.1 Challenges of the Multi-Objective Evolutionary Algorithm

The two major problems that have to be addressed when considering extending the idea of single objective evolutionary algorithms to multi-objectives cases are:

(i) How to accomplish fitness assignment and selection so as to guide the search towards the Pareto-optimal set and

(ii) How to maintain a diverse population in order to prevent premature convergence

The first problem stated above was addressed by reviewing the general fitness assignment strategies as follows:

3.5.4.1.1 Fitness Assignment Strategies

In multi–objectives optimization problems, the objective function no longer qualifies as fitness function since it is vector valued and we know that fitness function has to be scalar valued. Different strategies have been used to relate the fitness function to the objective function: We give an overview of these strategies as follows:

(i) Aggregation methods

Any multi – objective optimization problem may be converted to a single – objective optimization problem by aggregating the objectives into a scalar function $u : x \rightarrow \Re$; that is how classical optimization techniques handle multi – objectives. This method combines the objective into a higher scalar function that is used for fitness calculation.

Scalarization with changing parameters

One scalarization method, known as the weighted-sum approach, associates a real weight w_t with each objective u_t accumulating the weighted objective values yields the combined objective value:

$$u(x) = \sum_{t=1}^{n} w_t u_t(x) \qquad (33)$$

(ii) Random-Weight Approach

Murata, Ishibuchi, and Tanaka proposed a random-weight approach to obtaining a variable search direction toward the Pareto frontier (Gen & Chang, 2000). Typically, there are two types of search behavior in the objective space: fixed-direction search and multiple – direction search.

The fixed – weight approach gives the genetic algorithms a tendency to sample the area toward a fixed point in the criterion space, while the random-weight approach gives the genetic algorithms a tendency to demonstrate a variable search direction, therefore, the ability to sample the area uniformly over the entire frontier.

Suppose that we are giving to maximize q objective functions, the weighted – sum objective is given as follows:

$$U(x) \;=\; \sum_{k=1}^{q} w_k u_k(x) \tag{34}$$

The random weight w_k are calculated by the equation

$$w_k \;=\; \frac{r_k}{\sum_{j=1}^{q} r_j}, \qquad k = 1, 2, \quad . \quad . \quad . \quad q \tag{35}$$

Where r_j are nonnegative random numbers.

Before selecting a pair of parents for crossover operation, a new set of random weights is specified by equation (35), and fitness values for each individual are calculated by equation (34). The selection probability P_j for individual i is then defined by the following linear scaling function:

$$P_i \;=\; \frac{u_i \;-\; u_{\min}}{\sum_{j=1}^{pop\_size}(u_j - u_{\min})} \tag{36}$$

Where $u_{\min}$ is the worst fitness value in the current population. A tentative set of Pareto solutions is stored and updated at each generation.

(iii) Target vector optimization method

This method tries to minimize the distance to a given goal $y_g \in U$ in the objective space:

$$s(x) \;=\; \left\| f(x) - y_g \right\| \tag{37}$$

Where the metric $\| \,.\, \|$, is defined as the Euclidean metric. Several such scalarization methods exists, however, they require profound knowledge and preference information provided by the decision maker; for many practical problems this knowledge is not available.

(iv) Fitness sharing

Fitness sharing is a niching technique which is based on the idea that individuals in a particular niche have to share the resources available, similar to nature. Thus, the fitness value of a certain individual is measured in terms of the individuals located in its neighborhood – neighborhood is defined in terms of a distance measured $d(i, j)$ and specified by the so - called niche radius σ_{share}. Mathematically, the shared fitness s_i of an individual i is equal to its old fitness f_i divided by its niche count:

$$s_i = \frac{f_i}{\sum_{j=1}^{n} sh(d(i,j))} \tag{38}$$

An individual's niche count is the sum of sharing function (sh) values between itself and the individuals in the population. The commonly used functions are of the form:

$$sh(d(i,j)) = \begin{cases} 1 - \left(\dfrac{d(i,j)}{\sigma_{share}}\right)^{\alpha} & \mathit{if}\ d(i,j) < \sigma_{share} \\ 0 & \mathit{otherwise} \end{cases} \tag{39}$$

3.5.4.1.2 Maintaining Diverse Population to prevent Premature Convergence

The second problem of how to maintain a diverse population in order to prevent premature convergence when considering extending the idea of single objective evolutionary algorithms to multi-objectives cases, is usually solved by introducing elitism and intermediate recombination. Elitism is a way to ensure that good individuals do not get lost (by mutation or set reduction), simply by storing them away in an external set, which only participates in selection. Intermediate recombination, on the other hand, averages the parameter vectors of two parents in order to generate one offspring according to (JeffNve, 2004).

$$\begin{aligned} x_j' &= \alpha x_{j1}^{g} + (1-\alpha)x_{j2}^{g}, \quad j, j_1, j_2 \in \{1, \ . \ . \ . \ \mu\} \\ x_i^{g+1} &= x_j' + \mathrm{N}(0, \textstyle\sum), \quad i = 1, \ . \ . \ . \ \lambda, \quad j \in \{1, 2, \ . \ . \ . \ \mu\} \end{aligned} \tag{40}$$

3.5.5 Implementation of the Formulated Model by Applying Genetic Algorithm

The problem formulated as equation (32) has m objectives, plus m extreme points in the Pareto solutions, each of which maximizes one objective. An elite preserving strategy is suggested for putting the n extreme points plus some randomly selected Pareto solutions into the next population.

Let N_{pop} denote the population size and N_{elite} denote the number of elite solution to be preserved. The overall structure of the implementation of genetic algorithms is given as follows:

Step1. Initialization:

Randomly generate an initial population containing N_{pop} individuals

Step2. Evaluation:

Calculate the values of m objective functions for each individual.

Update a tentative set of Pareto solutions.

Step3. Selection:

Repeat the following steps to select $(N_{pop} - N_{elite})$ pairs of parents.

Specify random weights by equation (38)

Calculate fitness value by equation (37)

Calculate selection probability by equation (39)

Select a pair of parent individuals for a crossover operation.

Step 4 Crossover:

For each pair selected, apply a crossover operation to generate offspring.

Step5. Mutation:

Apply a mutation operation to each offspring generated by the crossover operation.

Step 6 Elitist Strategy:

Randomly select N_{elite} individuals from the tentative set of Pareto solutions

Add the selected solutions N_{elite} to $(N_{pop} - N_{elite})$ individuals generated in the foregoing steps to construct a population of N_{pop} of individual.

Step 7 Termination test

If a pre-specified stopping condition is satisfied, stop the run, otherwise Return to step 1.

The next section presents Bayesian learning algorithm which negotiating agents can use to learn other's negotiation behavior so as to optimize their conflicting objectives.

3.5.6 Bayesian learning algorithm

The goal of this section is to discuss Bayesian algorithm for equipping a negotiation agent with learning capability, so as to enhance agent's autonomy in dynamic environment such as an electronic market. The section contains the statement of Bayesian theorem, derivation of Bayesian theorem from conditional probabilities, and the Bayesian network. Usually Bayesian behavior is considered as the only rational agent's behavior that maximizes the utility of the user. Bayesian learning is built on Bayesian reasoning which provides a probabilistic approach to inference. The Bayesian learning algorithm manipulates probabilities together with observed data. During negotiation, the agents use the Bayesian framework to update knowledge and belief that they have about the other agents and the environment. For example, an agent (buyer/seller) could update his belief about the reservation price of the other agent (buyer/seller) based on his interactions with the seller/buyer and on his domain knowledge. The agent's belief is expressed as a set of hypotheses. Each agent tries to model the others in a recursive way during the negotiation process, and any change in the environment, if relevant and perceived by an agent, will have an impact on the agent's subsequent decision making.

3.5.6.1 Statement of Bayesian's theorem

Bayesian's theorem relates the conditional and marginal probabilities of stochastic events H_i and O:

$$\left.\begin{aligned} \Pr(H_i \setminus O) &= \frac{\Pr(O \setminus H_i)\Pr(H_i)}{\Pr(O)} \\ &\propto L(H_i \setminus O)\Pr(H_i) \end{aligned}\right\} \tag{41}$$

Where $L(H_i \setminus O)$ is the likelihood of H_i given fixed O.

Each term in Bayesian's theorem has a conventional name:

$\Pr(H_i)$ is the prior probability or marginal probability of H_i. It is "prior" in the sense that it does not take into account any information about O.

$\Pr(H_i \setminus O)$ is the conditional probability of H_i, given O. It is also called the posterior probability because it is derived from or depends upon the specified value of O.

$\Pr(O \setminus H_i)$ is the conditional probability of O given H_i.

$\Pr(O)$ is the prior or marginal probability of O, and acts as a normalizing constant.

With this terminology, the theorem may be paraphrased as

$$Posterior \quad = \quad \frac{likelihood\ x\ prior}{normalizing\ cons\tan t} \tag{42}$$

In other words: the posterior probability is proportional to the prior probability times the likelihood. In addition, the ratio $\Pr(O \setminus H_i)\Pr(O)$is sometimes called the standardized likelihood, so the theorem may also be paraphrased as

$$Posterior \quad = \quad s\tan dardised\ likelihood\ X\ prior. \tag{43}$$

3.5.6.2 Derivation of Bayesian theorem from Conditional Probabilities

To derive the Bayesian theorem, we let H_i represent an opponent reservation prices, and O to represent a negotiation offer. We start from the definition of conditional probability. The probability of event H_i given an event (offer) $O$is defined as:

$$\Pr(H_i \setminus O) \quad = \quad \frac{\Pr(H_i \cap O)}{\Pr(O)} \tag{44}$$

Likewise, the probability of offer O given opponent reservation prices H_i is

$$\Pr(O \setminus H_i) \quad = \quad \frac{\Pr(H_i \cap O)}{\Pr(H_i)} \tag{45}$$

Rearranging and combining equations (44) and (45), we have:

$$\Pr(H_i \setminus O)\Pr(O) \quad = \quad \Pr(H_i \cap O) \quad = \quad \Pr(O \setminus H_i)\Pr(H_i) \tag{46}$$

This lemma is sometimes called the product rule for probabilities. Dividing both sides of equation (46) by $\Pr(H_i)$, $\Pr(O) \quad \neq \quad 0$, we obtain Bayesian's theorem as:

$$\Pr(H_i \setminus O) = \frac{\Pr(O \setminus H_i)\Pr(H_i).}{\Pr(O)} \tag{47}$$

$$\Pr(H_i \setminus O) = \frac{\Pr(O \setminus H_i)\Pr(H_i).}{\sum_{k=1}^{n}\Pr(O \setminus H_k)\Pr(H_k)} \tag{48}$$

where:

$\Pr(H_i \setminus O)$ is the probability that opponent reservation price H_i is true given an offer O

$\Pr(H_i)$ is the probability that the opponent reservation price H_i is true overall.

$\Pr(O \setminus H_i)$ is the probability of observing offer O when the opponent reservation price H_i is true. n is the number of possible hypotheses.

3.5.6.3 Probability Negotiation Decision Making

The preference relation $\precsim p$ of a negotiation agent is a total ordering induced by the product of the agent's private utility function U_p^o and the probability function $\Pr(accept \setminus o)$ which characterizes the probability of acceptance of an offer o by the opponent. In other words, the feasible offers of an agent p are ranked in descending order according to:

$$[U_p^o(o)]^\alpha x[\Pr(accept \setminus o)]^{(1-\alpha)}$$

Where $\alpha \in [0,1]$ is a trade-off factor for maximizing one's own payoff or maximizing the chance of the offer being accepted by the opponent. If $\alpha = 0$ is set, a negotiation agent only considers maximizing the opponent's payoff. The trade-off factor α is provided by the human user of the negotiation agent. and its value will remain the same during the negotiation process.

A counter-offer with the least amount of concession (in terms of the least decrement of own payoff and the chance of acceptability) is picked up from the top of the list $(\precsim p, \{O_p - O_p'\})$ ranked by an agent p in each negotiation round. Once the counter-offer is determined, it will be added to the set O_p' the revised O_p' forms the basis to evaluate the incoming offers. The probability of acceptance of an offer o can be computed according to the Bayesian theorem:

$$\Pr(c_j \backslash o) \quad = \quad \frac{\Pr(o \backslash c_j) x \Pr(cj)}{\Pr(o)} \tag{49}$$

where $c_j \in \{accpt, reject\}$. If the naïve assumption of feature (i.e. negotiation issue) independency is made, the prior probability

$$\Pr\ (o \backslash c_j) \quad = \quad \prod_{i=1}^{|A|} \Pr\ (d_{a_i} \backslash c_j) \tag{50}$$

By the addition rule of probability theory,

$\Pr(o) \ = \ \sum \Pr(o \backslash c_j)\ x\ \Pr(c_j)$ is held. Therefore, the probability of acceptance of an offer o by the opponent can be estimated according to:

$$\Pr(accept \backslash o) = \frac{(\Pr(accept)\ x \prod_{i=1}^{|A|} \Pr(d_{a_i} \backslash accept))}{[\Pr(accept)\ x \prod_{i=1}^{|A|} \Pr(d_{a_1} \backslash accept) + \Pr(reject)\ x \prod_{i=1}^{|A|} \Pr(d_{a_i} \backslash reject)]} \tag{51}$$

In the section following this, we present the designs for the implementation of the Co-operative Modeler.

3.6 Software Specification and Design for the Co-operative Modeler Prototype

Table 3.4 gives the specification of the negotiation protocol between the Marketing-Agent and the Client-Agent of the proposed Co-operative modeler. The role players in this protocol are the Client-Agent who plays the role of a buyer of resources, while the Marketing-Agent plays the role of a seller of resources. The communication between the two agents consist of exchange of messages in form of buyers querying sellers for available resource to meet their task requirements, query for prices of the available resources, exchanges of proposal followed by proposal-accepted or counter-proposals. If the two agents agree over a deal, sales accepted must be reported followed by sales confirmed.

The execution protocol (Table 3.5) consists of exchange of requests/responses between the Client and the Execution-Agent over task execution. The Execution-Agent is acting the role of task executor. The protocol allows the Client to request for the status of the task execution, during the execution period. End of task execution is reported immediately after execution is finished.

Table 3. 4: Specification of the Negotiation protocol between the Client-Agent and the Marketing-Agent.

Protocol Content	Protocol Specification
Roles	Client-Agent as Buyer, Marketing-Agent as Seller
Messages	ResourceQuery ($X_{ClientAgent\rightarrow Marketing-Agent}$) PriceQuery ($X_{ClientAgent\rightarrow Marketing-Agent}$) PriceOffer ($X_{ClientAgent\leftarrow Marketing-Agent}$) NoOffer ($X_{ClientAgent\leftarrow Marketing-Agent}$) SaleAccept ($X_{ClientAgent\rightarrow Marketing-Agent}$) CounterOffer ($X_{ClientAgent\rightarrow Marketing-Agent}$) TerminateNegotiation ($X_{Client\rightarrow Marketing-Agent}$) SaleConfirm ($X_{ClientAgent\rightarrow Marketing-Agent}$)
Contract	SaleAccept Must be followed by SaleConfirmed

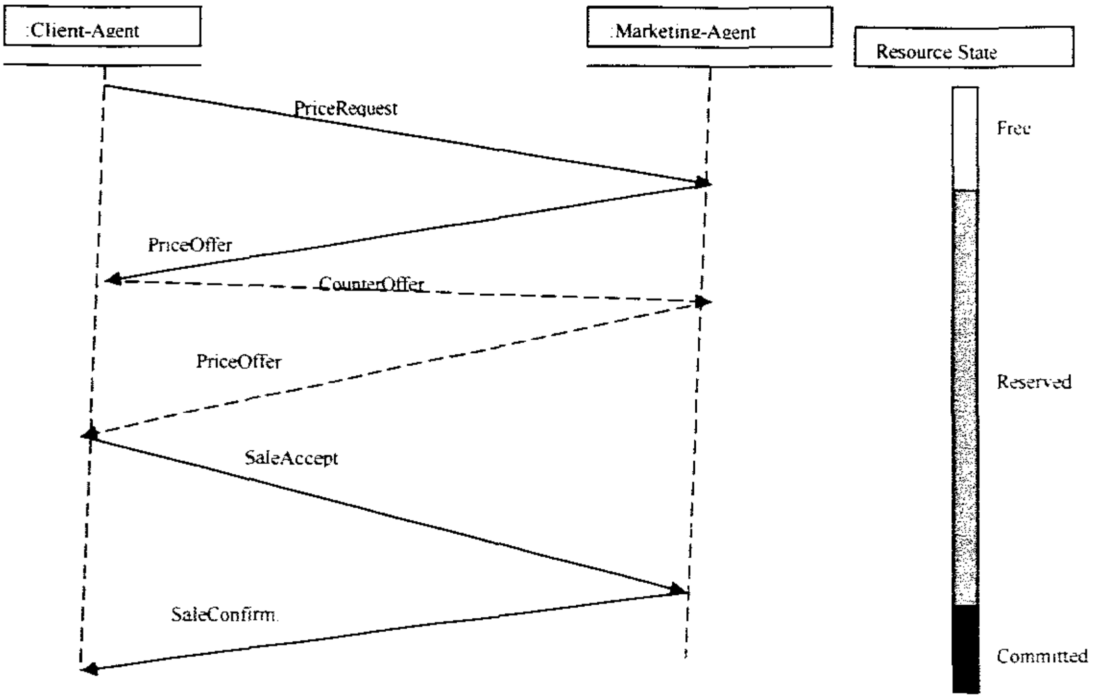


Figure 3. 4: Sequence Diagram Describing Negotiation Thread in relation to the states of a negotiated Grid Resource.

Table 3. 5 : Specification of the Execution Protocol

Protocol Content	Task Execution Specification
Roles	Client as Client, Execution –Agent as Executor
Messages	ExecutionRequest ($X_{Client\rightarrow Execution\text{-}Agent}$) ExecutionAccept ($X_{Client\leftarrow Execution\text{-}Agent}$) ExecutionReject ($X_{Client\leftarrow Execution\text{-}Agent}$) ExecutionFinished ($X_{Client\leftarrow Execution\text{-}Agent}$) ExecutionQuery ($X_{Client\rightarrow Execution\text{-}Agent}$) ExecutionStatus ($X_{Client\leftarrow Execution\text{-}Agent}$)
Contracts	Execution Must be started immediately if resource id is valid Execution finished Must be reported immediately

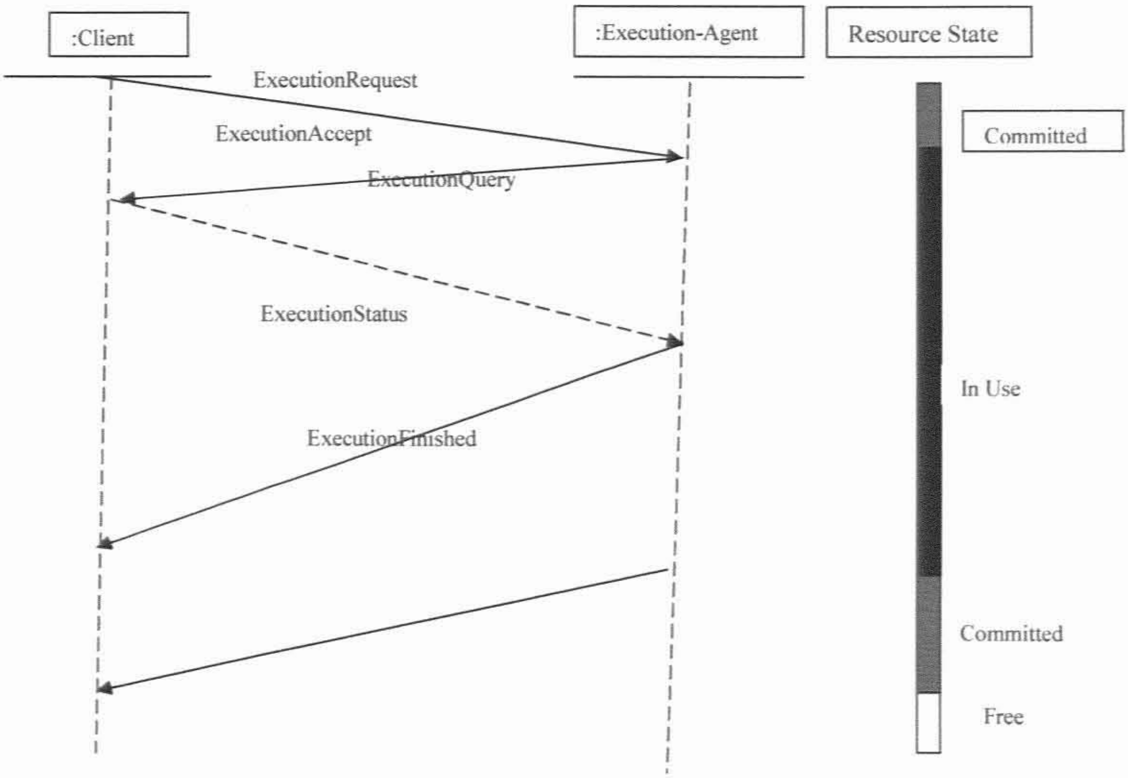


Figure 3. 5: Sequence Diagram Relating Task Execution Stages with the States of Grid Resources.

The negotiation protocol (Table 3.4) and the Execution Protocol (Table 3.5) represented above using Sequence diagrams in Figure 3.4 and Figure 3.5 respectively, are summarized using the sequence diagram in Figure 3.6. Figure 3.7 is the Class Design pattern for the implementation of the presented Co-operative Modeler.

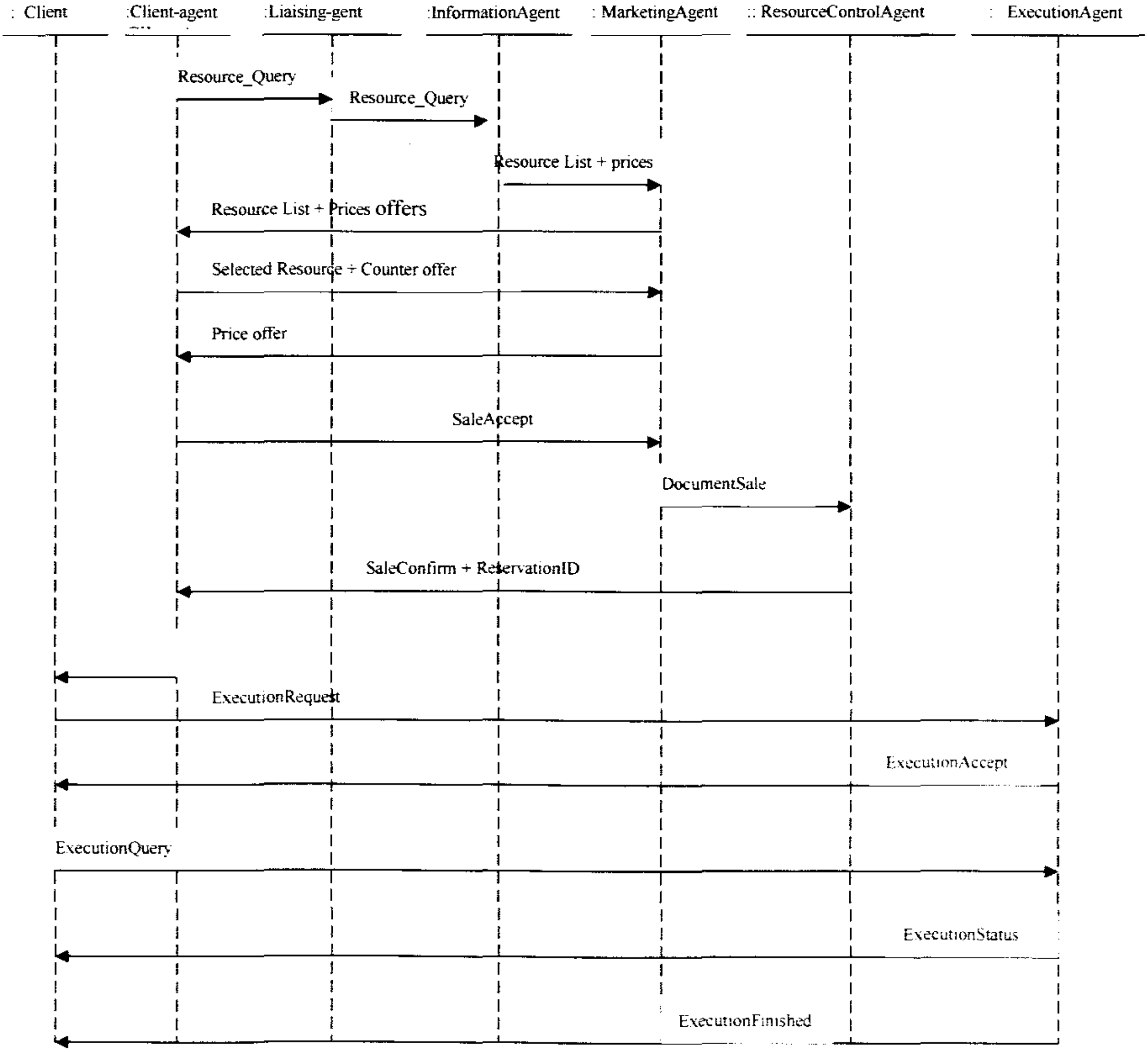


Figure 3. 6: Sequence diagram showing Negotiation Communication in a Co-operative Market

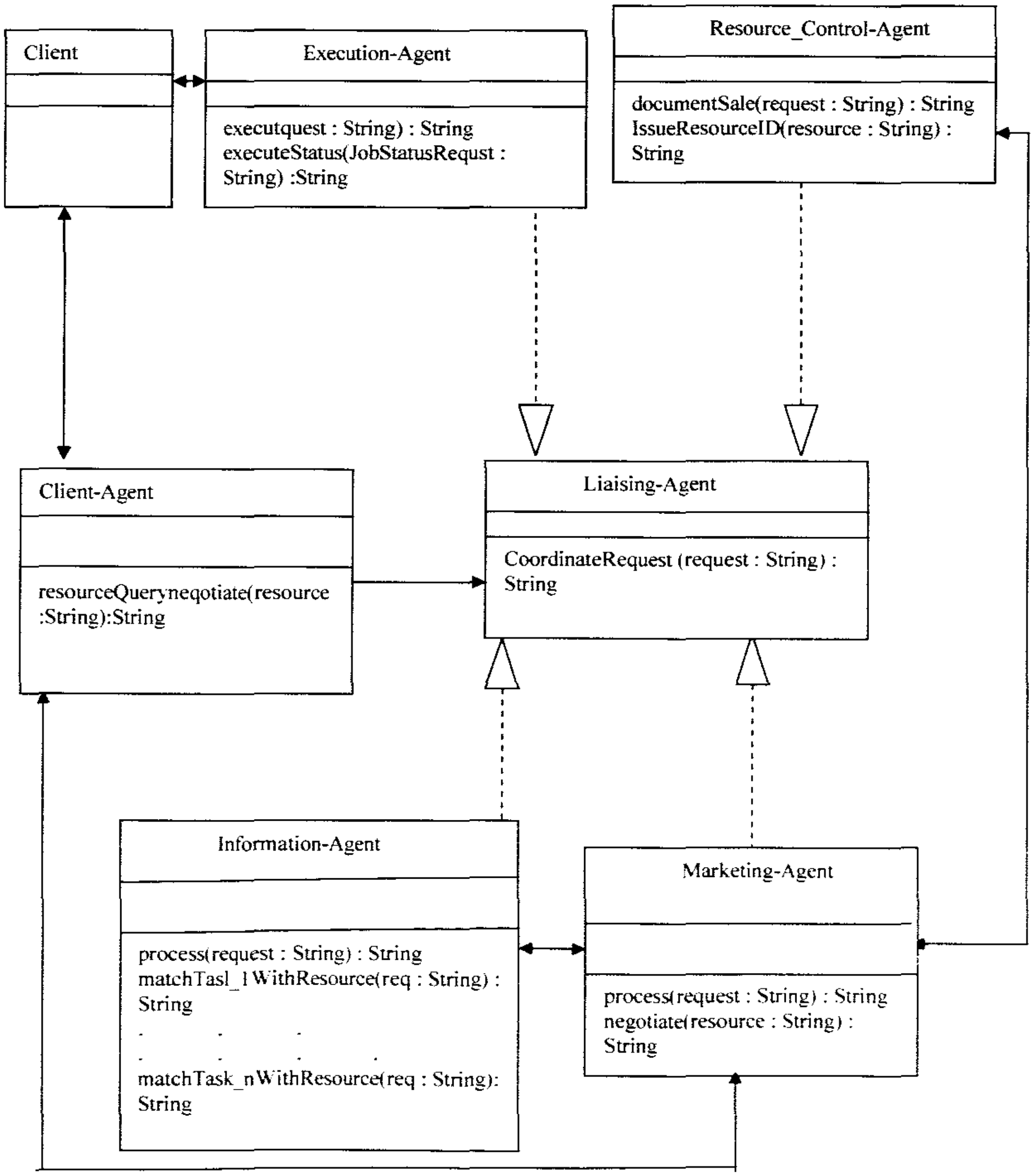

Client
Execution-Agent
executquest : String) : String
executeStatus(JobStatusRequst : String) :String
Resource_Control-Agent
documentSale(request : String) : String
IssueResourceID(resource : String) : String
Client-Agent
resourceQueryneqotiate(resource :String):String
Liaising-Agent
CoordinateRequest (request : String) : String
Information-Agent
process(request : String) : String
matchTasl_1WithResource(req : String) : String
. . .
. . . .
matchTask_nWithResource(req : String): String
Marketing-Agent
process(request : String) : String
negotiate(resource : String) : String

Figure 3. 7: The Class Diagram for implementing the proposed Co-operative Modeler

3.7 Conclusion

We conclude this chapter by presenting in the following sub sections, the bilateral negotiation model, and the flowchart used for the implementation of the Co-operative Modeler Prototype.

3.7.1 Bilateral Negotiation Model

This sub section describes the bilateral negotiation between providers and consumers of resources. The Co-operative agents are mediating the negotiation interaction between the providers and the consumers of resources. A client wishing to buy a resource reveals her price valuation (v) to the Co-operative. The resource provider also reveals her price valuation (s) to the Co-operative. Based on the players' reports, the Co-operative agent specifies a mechanism Γ(β, p, w) as follows:

i. The Co-operative agent determines $\beta(s,v)$ which specifies if the current trade is to take place:

$$\beta(s,v) = \begin{cases} 1 & \text{if} \quad v \geq p \quad \text{and} \quad w \geq s \\ 0 & \quad \text{otherwise} \end{cases}$$

ii. If the trade is to take place $\beta(s,v) = 1$,

the Co-operative agent collects ask price (p) from the buyer and pay the bid price (w) to the resource provider. The Co-operative agent determines the bid – ask spread (p,w) to maximize its utility function.

If the trade is not to take place $\beta(s,v) = 0$, the players take their outside options.

We implemented the bilateral negotiation model based on the class diagram in [Figure 3.7], by using the algorithm described by the flow chart provided in [Figure 3.8].

3.7.2 Flowchart for Implementing the Co-operative Modeler Negotiation Model

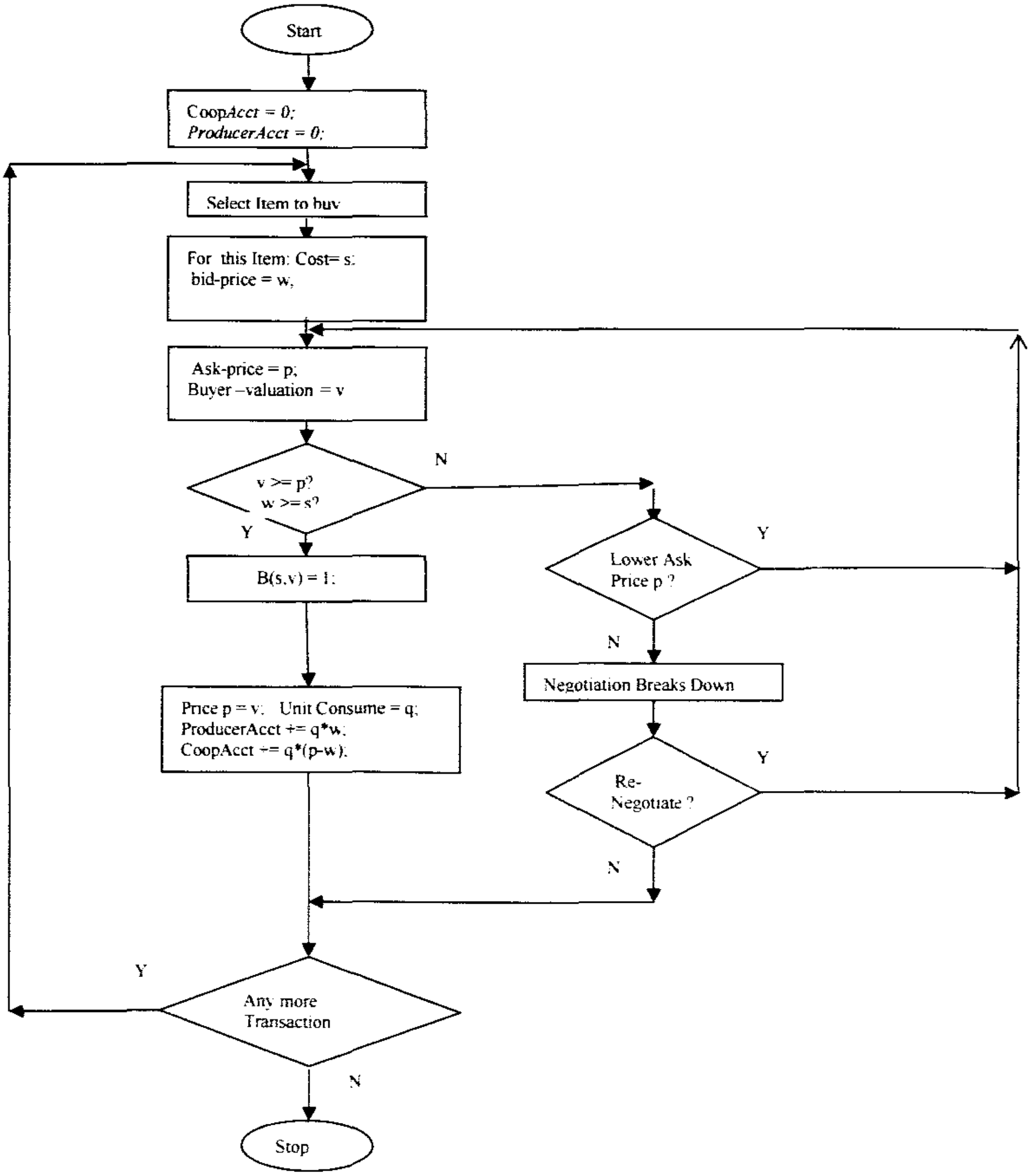


Figure 3. 8: The Flow Chart for implementing the prototype of the Co-operative Modeler (Aremu & Adigun, 2007).

CHAPTER FOUR

SIMULATION AND DISCUSSION OF RESULTS

4.1 Introduction

This chapter presents an implementation prototype of the Co-operative Modeler as designed, and an experimental system to simulate the behavior of the Co-operative Modeler agents. The chapter is divided into two parts. The first part discusses the implementation prototype of the Co-operative Modeler, while the second part discusses the experimental design to simulate the negotiating behavior of the Co-operative Modeler agents using time-dependent negotiation tactics.

4.2 The Co-operative Modeler Prototype

The Co-operative Modeler Prototype mimics the way negotiation is being carried out in a real world negotiation. The purpose of the implementation is to demonstrate a possible extension to the current systems of buying and selling online that do not allow buyers to have opportunity to negotiate the prices of the items they buy. The implementation prototype is based on the class diagram presented in chapter three (figure 3.9). The issue of negotiation considered for the implementation is price.

4.2.1 Implementation Environment

We used the Document Object Model (DOM)-based parser for communicating XML messages between the agents. We chose the DOM – based parser because the XML Document Object Model (DOM) is the standard recommendation by the W3C for building a tree structure in memory for XML documents. The NetBeasns IDE was used to implement the model.

4.2.2 Implementation Result

When a Client wants to buy a product online using the implemented prototype of the Co-operative Modeler, the client is required to click on the Co-operative agent icon, the screen snap shot in Figure 4.1 will be displayed for the client to select the product he wants to buy. On clicking the product menu bar, the screen snap shot in Figure 4.2 is displayed for the client to select the product category he is interested in. When the client clicks on a product category, the list of the product items under that category is displayed in Figure 4.3, for the client to select an item of his choice. On selecting an item, the screen snap shot in Figure 4.4 is displayed for the client to commence negotiation. On the negotiation screen, the Marketing-agent displays: the picture of the item the client selects for purchase, and the proposed price the Marketing-agent is willing to sell the item. The Client has two choices: either to accept the price offer by the Marketing-agent by clicking on the purchase icon, or to counter - propose a more suitable price to him, by clicking on the negotiate icon. If the clients chooses to negotiate by clicking on the negotiate icon, the screen snap short in Figure 4.5 is displayed to allow the client to counter – propose a more suitable price to him, by adjusting the price proposal of the Marketing-agent.

After adjusting the price proposed by the Marketing-agent, the client clicks on the negotiate icon to submits his price to the Marketing-agent. The Marketing-agent evaluate the price proposal from the Client, and takes a decision to either accept this proposal or refuse the proposal by either reduce his own earlier price proposed or stayed firm by maintaining the price proposed. The system at this point [Figure 4.6] would remind the negotiators of the remaining time left for the current negotiation thread. If the client chooses to re-negotiate, the negotiation process will re-start from Figure 4.4. But if the client accepts the price proposal from the Marketing-agent by clicking on the purchase icon, the screen snap shot in Figure 4.7 is displayed for the client to select the quantity of the item he wants to buyer, enter his bank account details or credit card details, and click on purchase to submit this transaction. A screen will be displayed for the client to confirm this transaction by clicking on the Ok icon, or adjust the transaction by clicking on the Cancel icon. If the Ok icon is clicked, a receipt is generated for the client showing the detail of the transaction and the amount to be paid for the transaction. The client may choose to buy another item by going through the whole process from the beginning, or end the current transaction. The system generates, and prints a typical receipt as seen in Figure 4.9 for the transaction.

4.2.3 Prototype Snap Shots

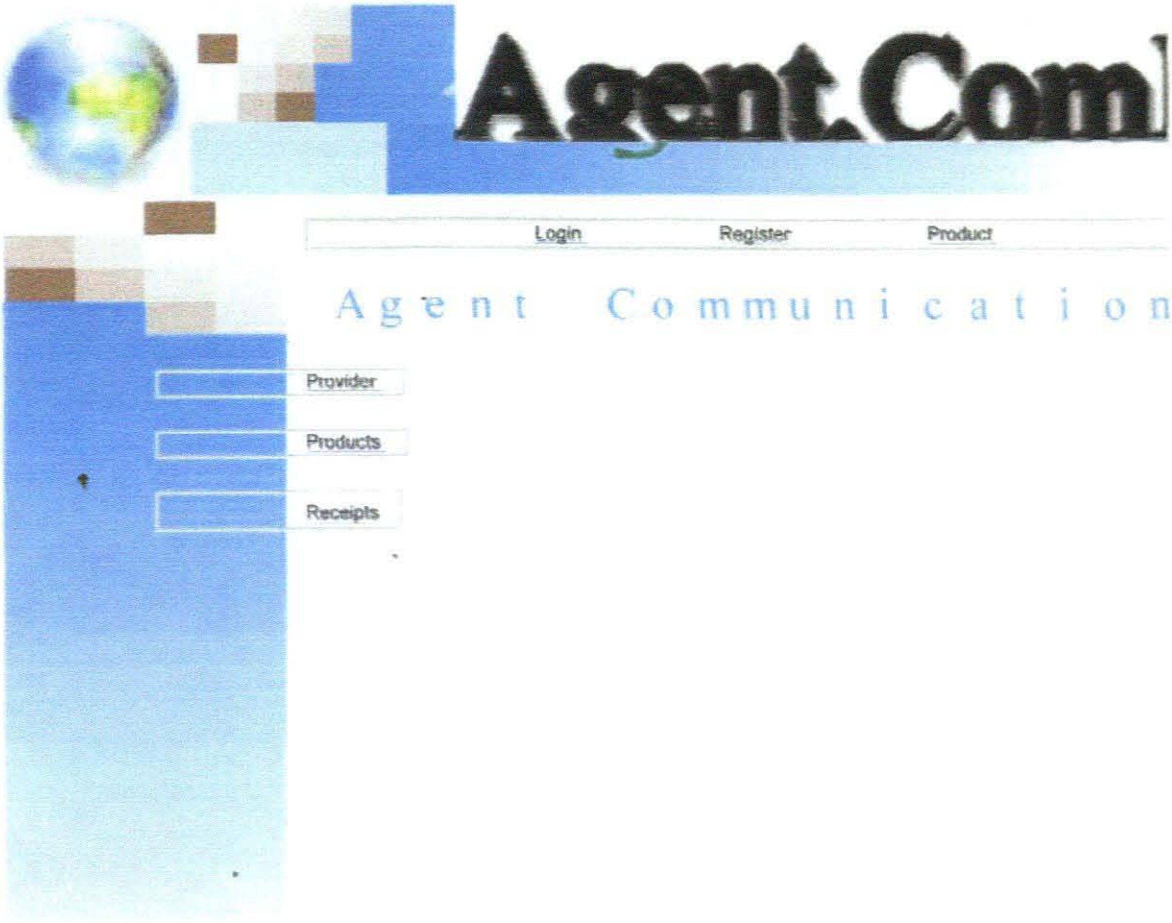
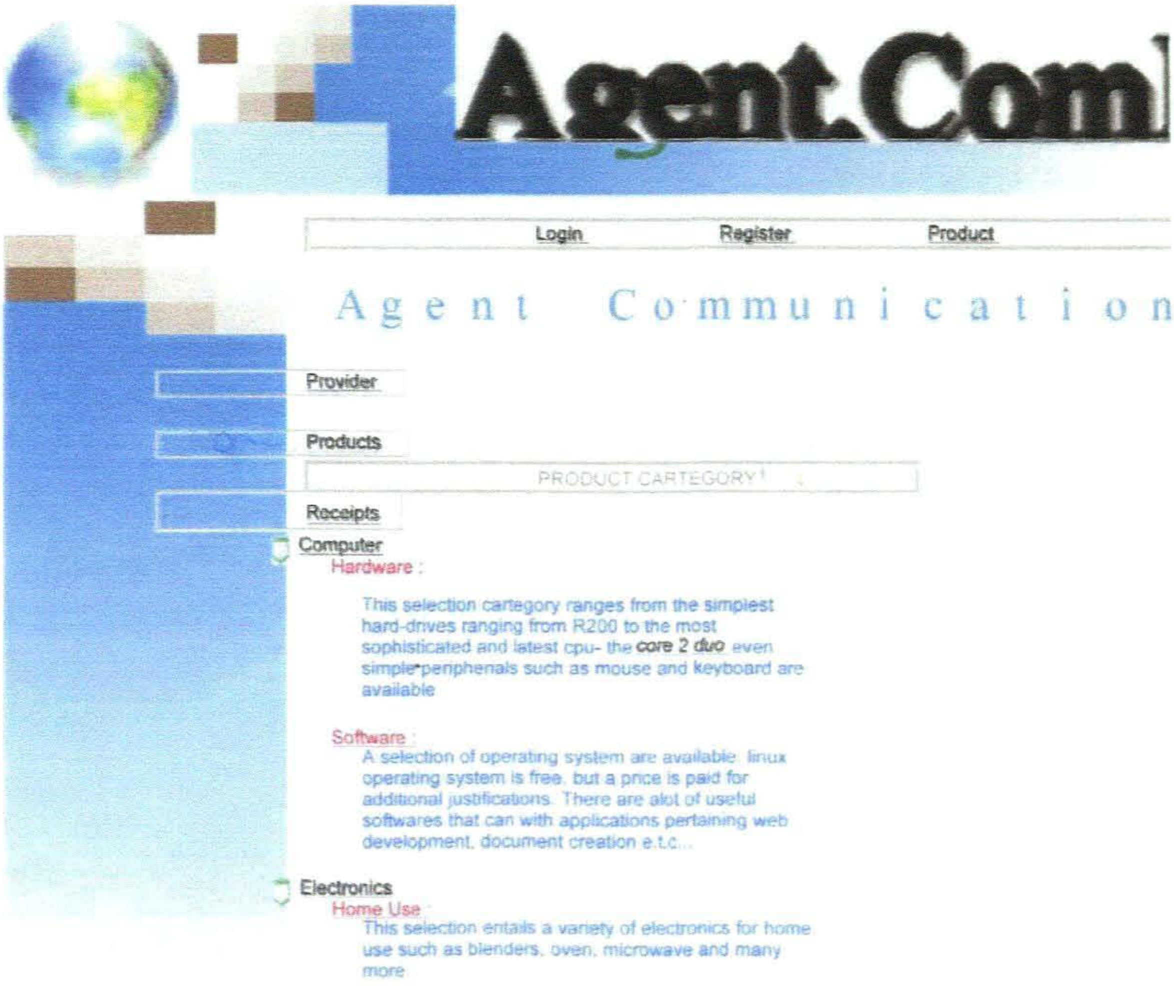


Figure 4. 1: Main Menu of the Co-operative Middleware.

Clients are required to click on the Product menu bar to start the process of buying an item using the Co-operative Middleware agent.



Computer

Hardware :

This selection cartegory ranges from the simplest hard-drives ranging from R200 to the most sophisticated and latest cpu- the *core 2 duo* even simple peripherals such as mouse and keyboard are available

Software :

A selection of operating system are available. linux operating system is free. but a price is paid for additional justifications. There are alot of useful softwares that can with applications pertaining web development, document creation e.t.c...

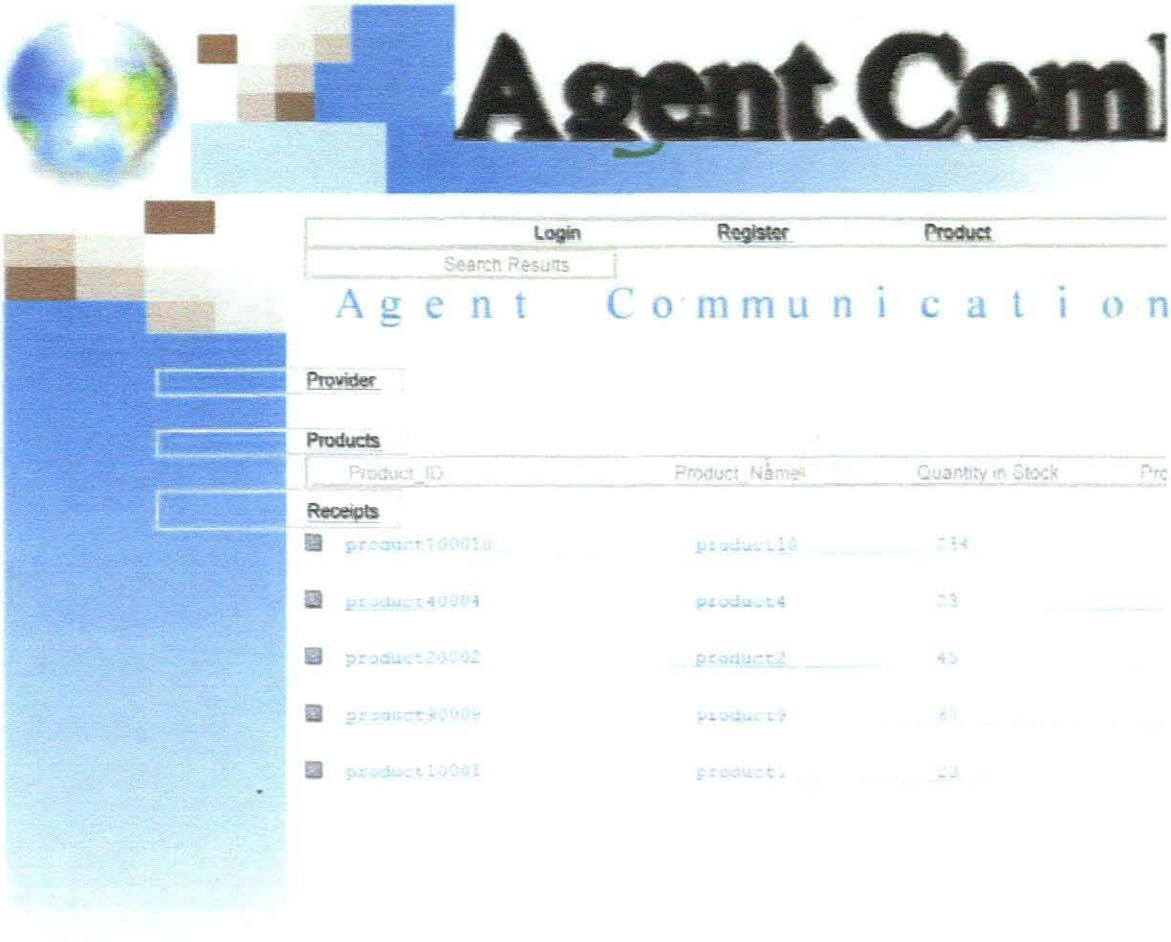
Electronics

Home Use :

This selection entails a variety of electronics for home use such as blenders, oven, microwave and many more

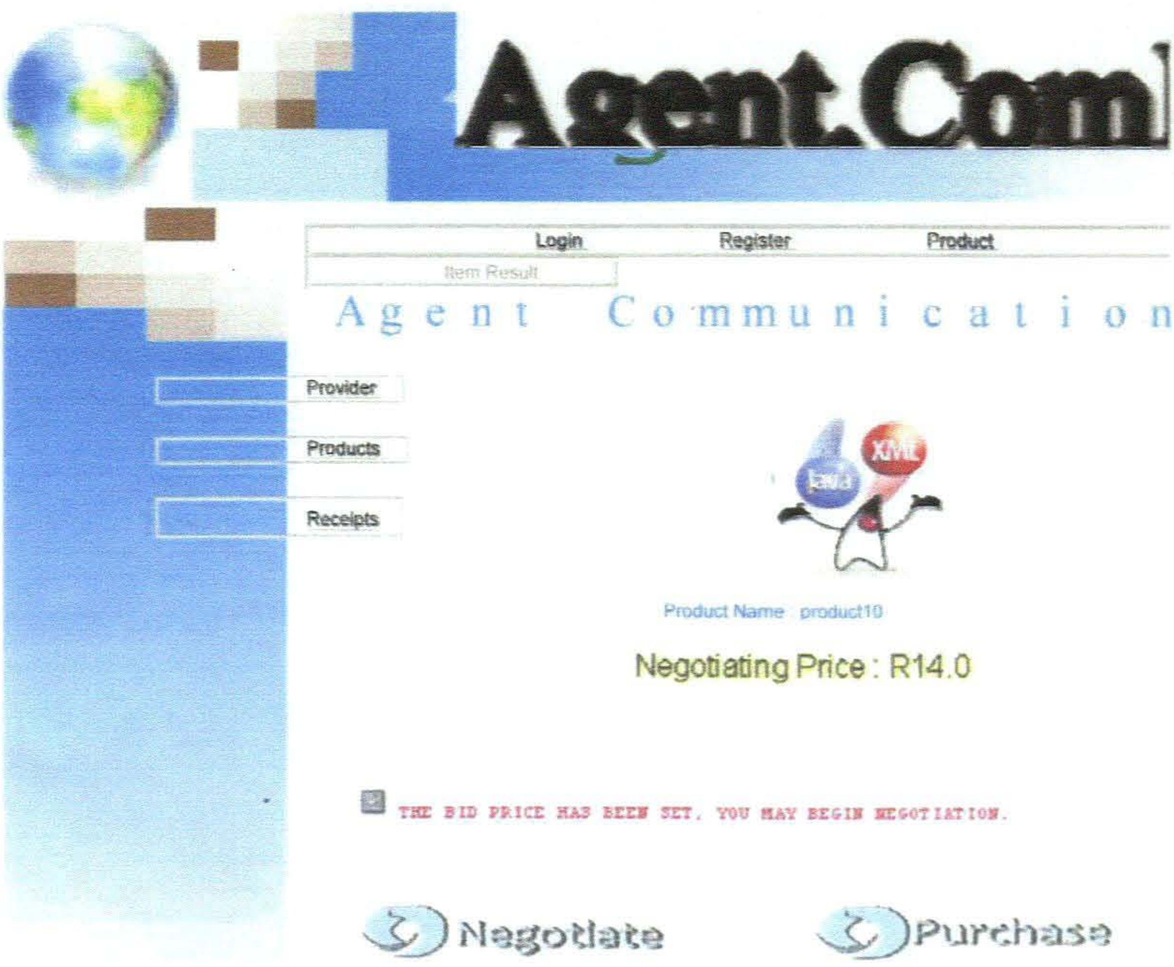
Figure 4. 2: Product Category Menu

Clients are to click on the Product category of interest from list above.



Product_ID	Product_Name	Quantity in Stock	Pro
product100010	product10	234	
product40004	product4	23	
product20002	product2	45	
product90009	product9	[illegible]	
product10001	product1	23	

Figure 4. 3: Item List Menu.
A Client is required to click on the item he desires to buy from the listed items.



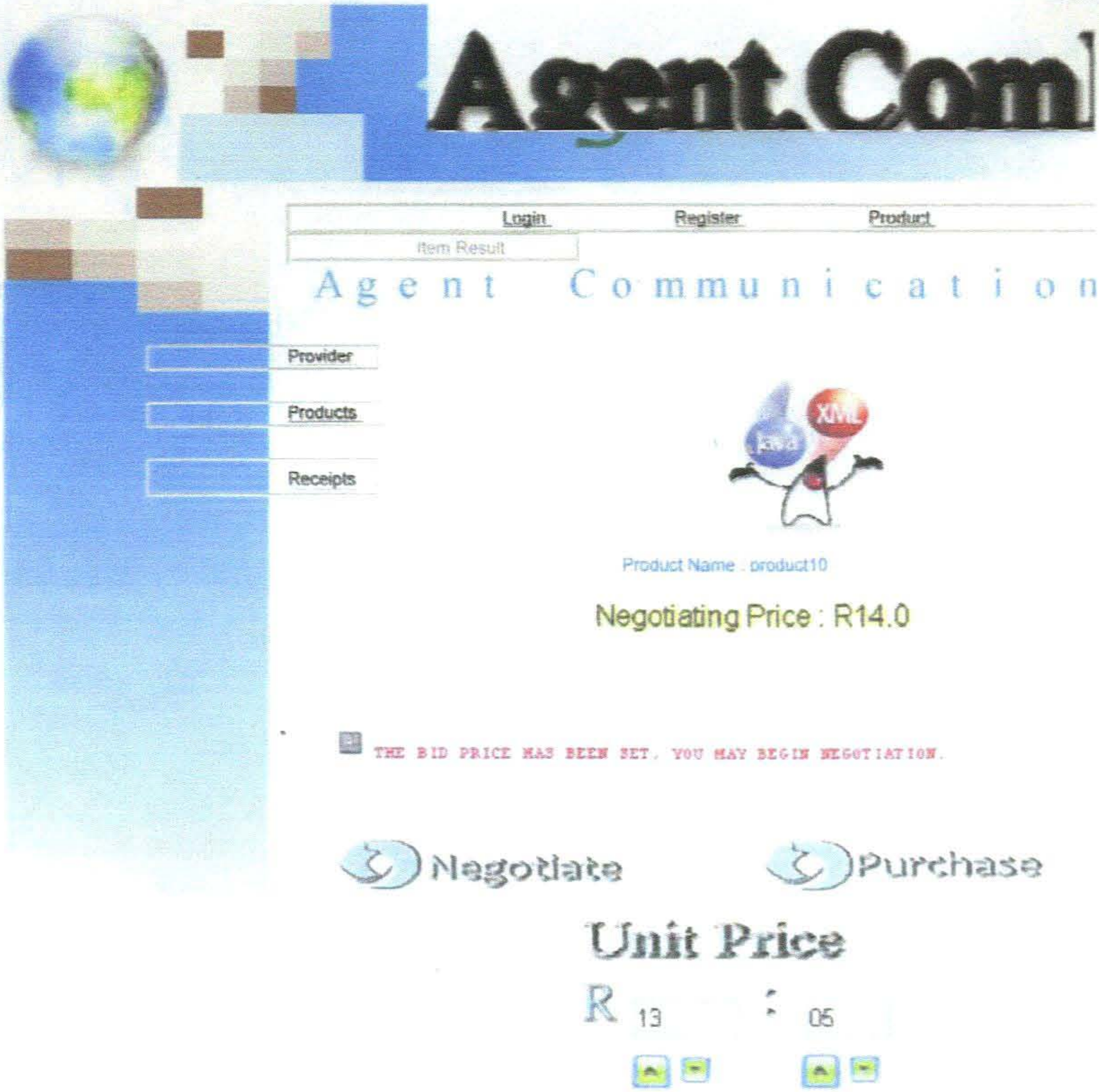
Product Name product10

Negotiating Price : R14.0

THE BID PRICE HAS BEEN SET, YOU MAY BEGIN NEGOTIATION.

Figure 4. 4: Negotiation Start Menu

Clients used their bargaining power here to beat down cost.



Product Name : product10

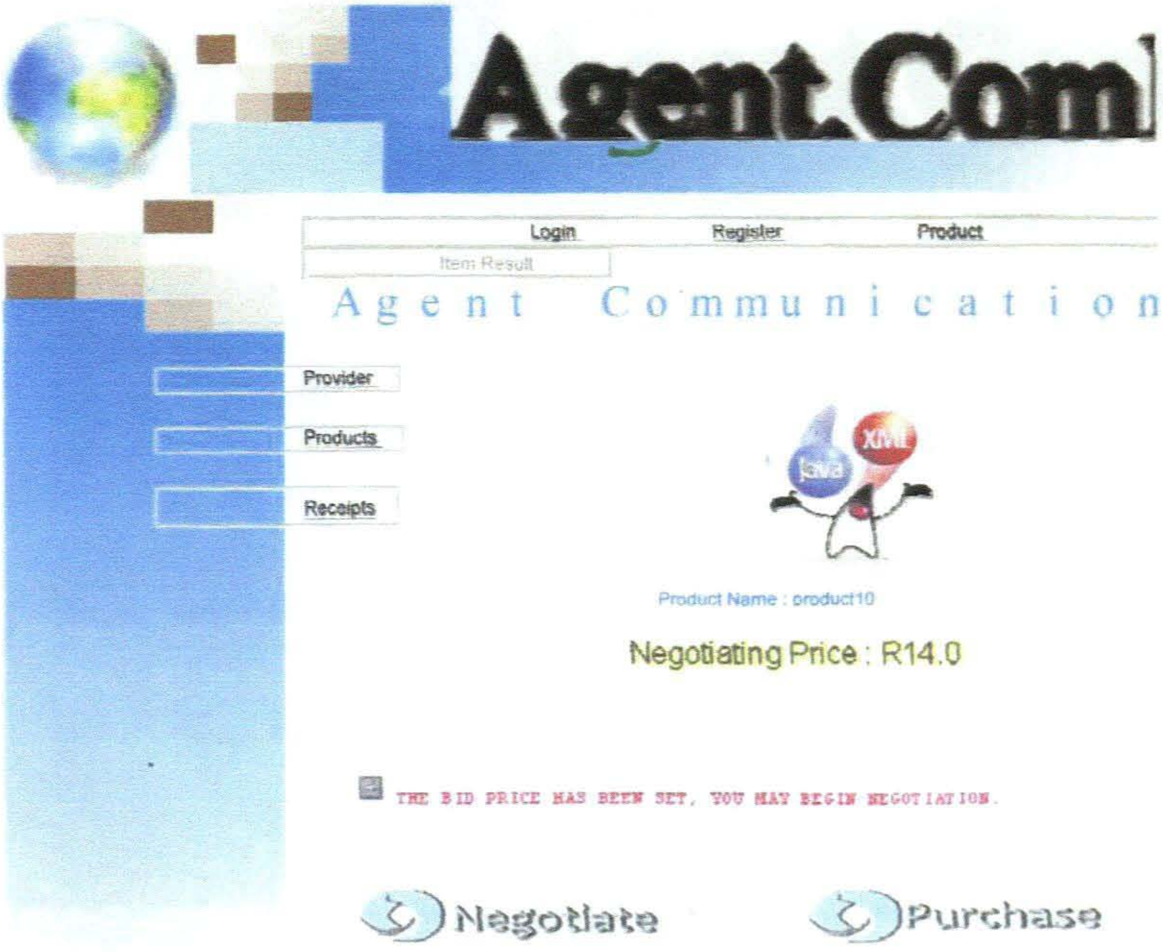
Negotiating Price : R14.0

THE BID PRICE HAS BEEN SET. YOU MAY BEGIN NEGOTIATION.

Unit Price

R 13 : 05

Figure 4. 5: Negotiation/Purchase Menu.
Clients negotiates ad re-negotiates so as to beat down cost.

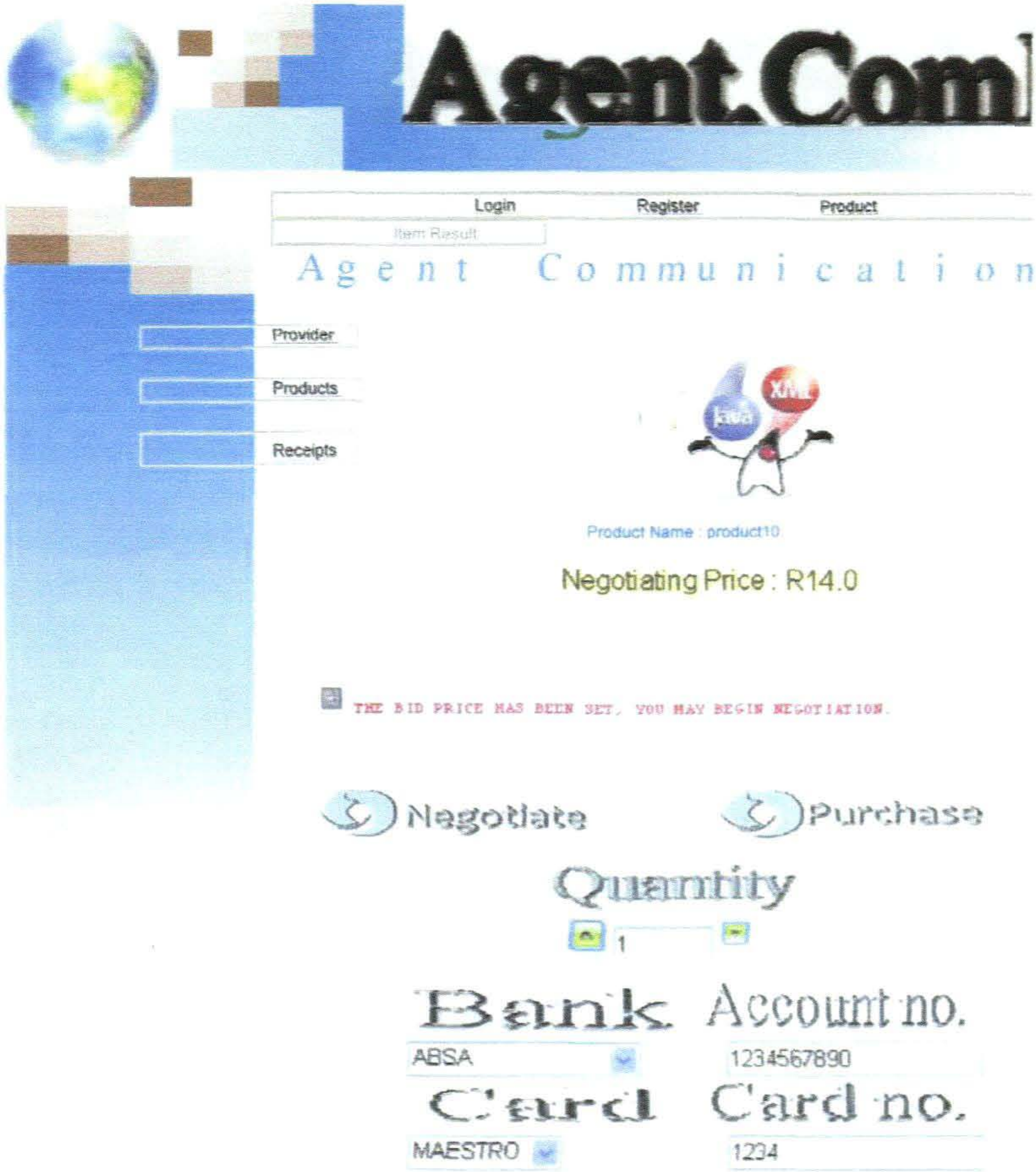


Product Name : product10

Negotiating Price : R14.0

THE BID PRICE HAS BEEN SET, YOU MAY BEGIN NEGOTIATION.

Figure 4. 6: Negotiation/Purchase Menu.
Client is at a dilemma here, to either purchase the negotiated item at the price proposed by the Marketing-agent or to negotiate further.



Product Name : product10

Negotiating Price : R14.0

THE BID PRICE HAS BEEN SET, YOU MAY BEGIN NEGOTIATION.

Quantity: 1

Bank: ABSA

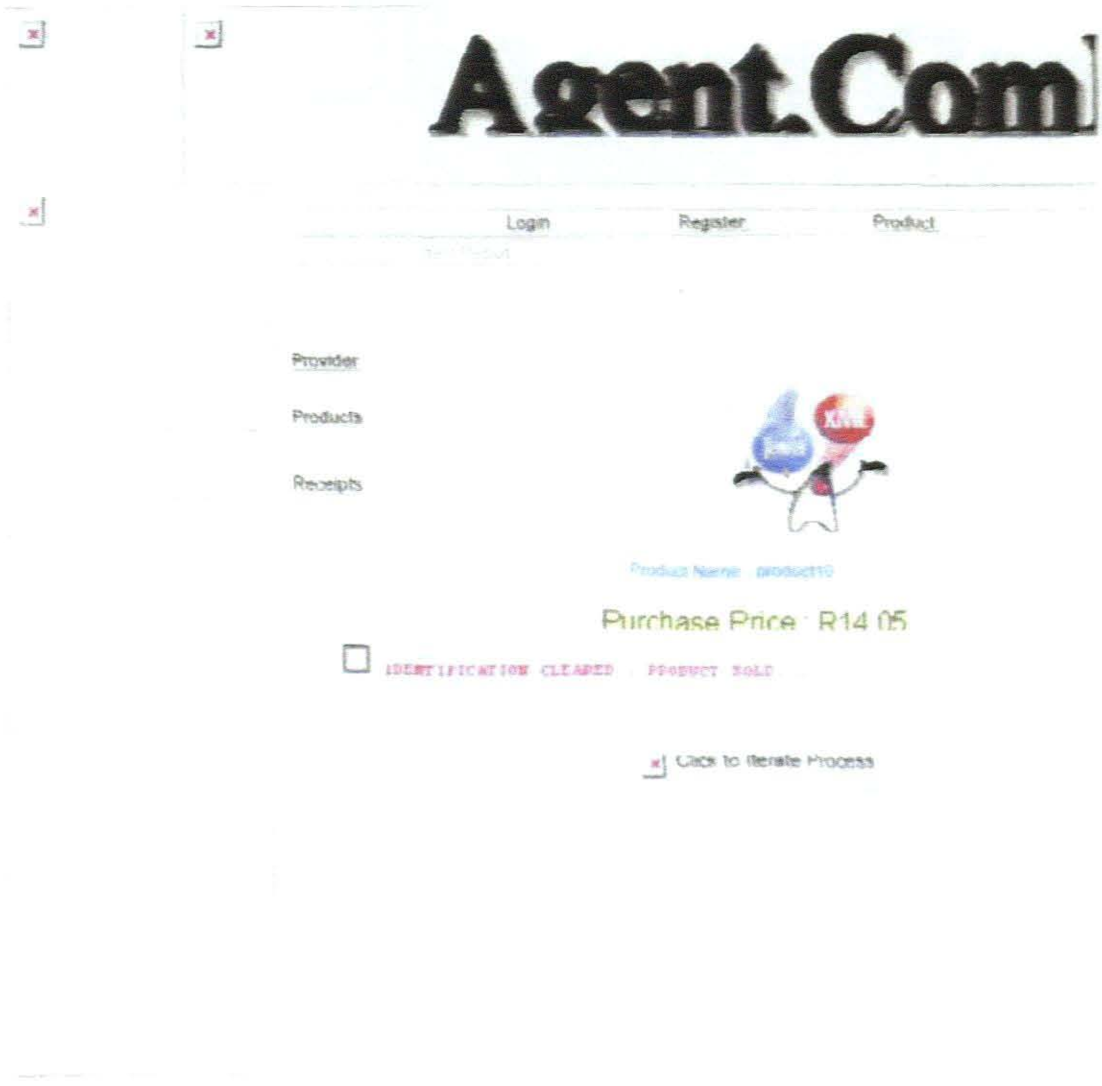
Account no.: 1234567890

Card: MAESTRO

Card no.: 1234

Figure 4. 7: Purchase Menu.

If a client at this point chooses to accept the price proposal of the Marketing-agent, he or she ought to indicate the quantity of item he or she desires, and provided his or her bank or credit card details for payment purpose.



Product Name : product10

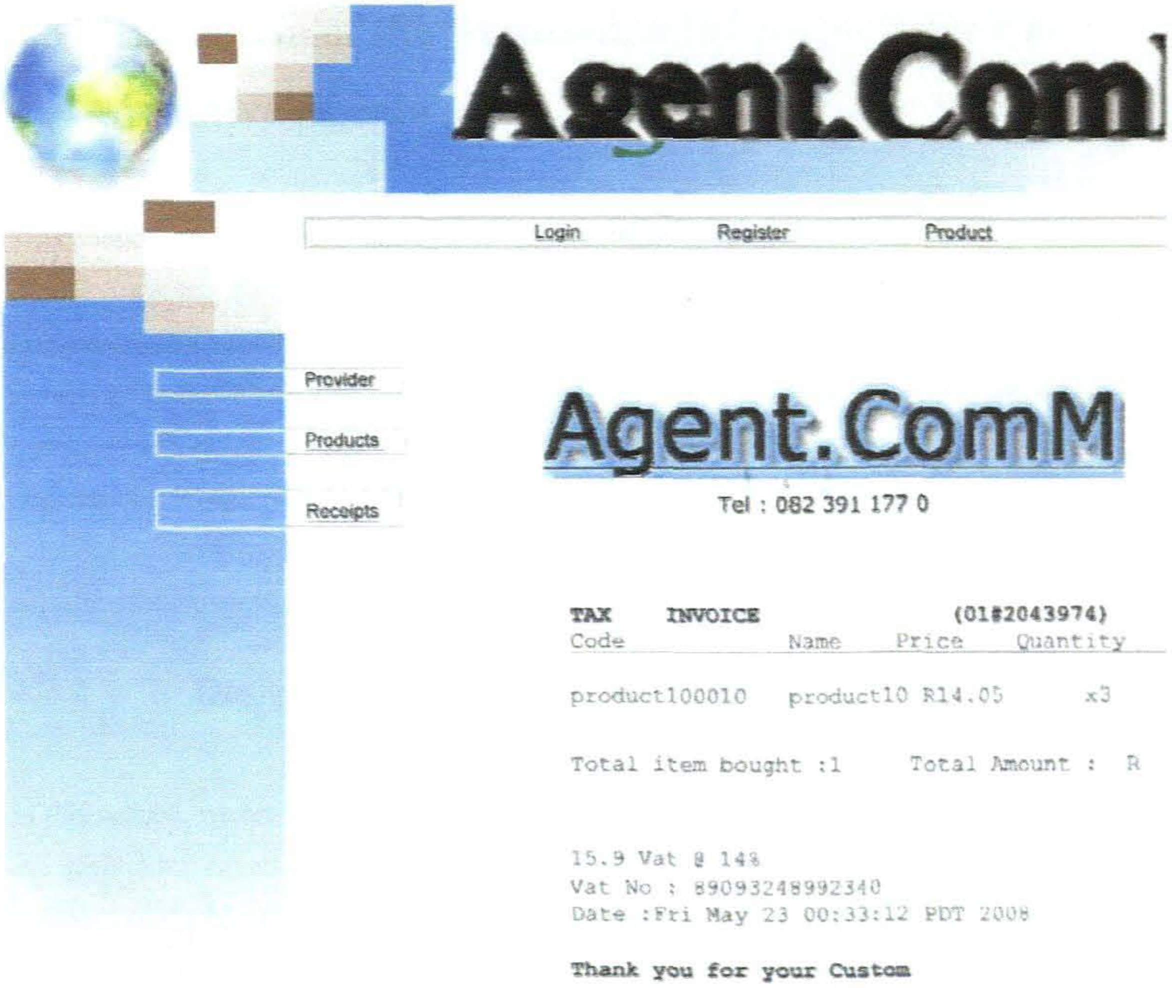
Purchase Price : R14.05

□ IDENTIFICATION CLEARED : PRODUCT SOLD

Click to Iterate Process

Figure 4. 8: Purchase Confirmation Menu.

The Marketing-agent confirms transaction at this point, and uses the transaction information to generate receipt for the client.



```
Tel : 082 391 177 0


TAX     INVOICE                 (01#2043974)
Code               Name    Price    Quantity

product100010   product10 R14.05       x3

Total item bought :1     Total Amount :  R


15.9 Vat @ 14%
Vat No : 89093248992340
Date :Fri May 23 00:33:12 PDT 2008

Thank you for your Custom
```

Figure 4. 9: Receipt Generation Menu.

The receipt generated as a result of the transaction is printed and handed over to the client.

4.3 Evaluation of the Negotiation Behavior of the Co-operative Agents

This section presents the evaluation of the negotiation behavior of the Co-operative Modeler agents, by using time–dependent negotiation tactics to simulate: (i) the importance of a negotiation issue to a negotiating agent, and (ii) the negotiating offer of an agent, during a negotiation process. The time dependent negotiation tactics help agents determine the values of negotiating issues based on the time elapsed in the negotiation. If agents can learn and understand the time – dependent negotiation behavior of their opponent, they might improve their negotiation performance by adjusting their own negotiation models.

4.3.1 Design of Negotiation Experiment

In this section, we set up the experiment to study the behavior of the scoring function of the negotiating agents. The goal of the experiment was to test the effect of time on the behavior of the Marketing-agent and the Client-agent of the Co-operative Modeler. and to compare these behaviors of the negotiators relative to each other over the time of negotiation.

The experiments involve multiple agents. When multiple clients come to the Co-operative Marketplace bargaining over resources, the Marketing-agent starts a separate negotiation thread with each client. Each negotiation thread progresses as it is defined in the bilateral negotiation model. Separate deadlines $T_{\max}$ are assigned to each thread. The decision variables (issues) under negotiation are price (p). quantity (q). and delivery date of the resource. The scoring function of the Marketing-agent is defined as:

$$V^{mktAgent}(x^t) = \sum_j w_j^{mktAgent} V_j^{mktAgent}. \quad j \in \{p. \quad q. \quad dd\} \qquad (53)$$

The assumption for the scoring functions is that the Marketing-agent desires higher price. larger quantity. and latter delivery date. Therefore. for all issues

$j\ (j \in \{p, q, dd\})$, the model's weights for all issues j for the Marketing-Agent is a function of time given as

$$\left.\begin{aligned} w^{mktAgent}(t) &= k + (1 - k)\frac{\min(t', T_{\max})^{\beta}}{T_{\max}}, \quad 0 \le k \le 1 \\ V_j^{mktAgent}(x_j^t) &= \begin{cases} \dfrac{\max_j - x_j^t}{\max_j - \min_j}, & for\ x_j^t \in [\min_j, \max_j] \\ 0 & for\ other\ values \end{cases} \end{aligned}\right\} \tag{54}$$

For the Client-Agent, the overall scoring function is

$$V^{client}(x^t) = \sum_j w_j^{client} V_j^{client}, \quad j \in \{p,\ q,\ dd\} \tag{55}$$

While the marketing-agent desires a higher price, a larger quantity, and later delivery date, the Client-agent desires low price for larger quantity, and early delivery date. Therefore, for all issues $j\ (j \in \{p, dd\})$,

$$V_j^{client}(x_j^t) = \begin{cases} \dfrac{x_j^t - \min_j}{\max_j - \min_j}, & for\ x_j^t \in [\min_j, \max_j] \\ 0 & for\ other\ values \end{cases} \tag{56a}$$

Therefore, for other $j\ (j \in \{q\})$.

$$V_j^{client}(x_j^t) = \begin{cases} \dfrac{\max_j - x_j^t}{\max_j - \min_j}, & for\ x_j^t \in [\min_j, \max_j] \\ 0 & for\ other\ values \end{cases} \tag{56b}$$

The weights for all issues j for the Client-Agent is a function of time given as

$$w_j^{client}(t) = 1 - w_j^{mkt}(t) \tag{57}$$

The negotiators used Time-dependent tactics to decide which value to offer (x_j^t) at time t will be proposed. The tactics vary the value of the issue depending on the remaining negotiation time. The following formula is used to calculate the value to be offered by an agent for issue j at time t' :

$$x_j^t = \begin{cases} \min_j + e^{\left(1-\frac{t}{T_{\max}}\right)^{\beta}\ln(K)}(\max_j - \min_j) & (58) \\ \textit{if} \quad V_j \text{ decreases as the value of issue j increases} & \\ \min_j + (1 - e^{\left(1-\frac{t}{T_{\max}}\right)^{\beta}\ln(K)})(\max_j - \min_j) & (59) \\ \text{Otherwise} & \end{cases}$$

$0 \le t \le T_{\max}$

Where

t is time (number of turns) in the interval [0, $T_{m,x}$],

x_j^t is the value of the negotiating issue proposed by an agent at time t,

$\min_j$ is the minimum value of the negotiating issue,

$\max_j$ is the maximum value of the negotiating issue,

K is a constant that determines the value of the negotiating issue in the first offer, $K \in (0,1)$,

$T_{\max}$ is the time limit (maximum number of turns) proposed by an agent

V is the utility function of the negotiating issue and,

β is a constant that determines the degree of convexity.

We assume that an increase in the value of an issue decreases the Client-agent's utility, *but increases the Marketing-agent's utility. Under this assumption, Equation (58) and* (59) represent the time dependent tactic (TDT) expressions for a buying (Client-agent) and a selling (Marketing-agent) agent, respectively.

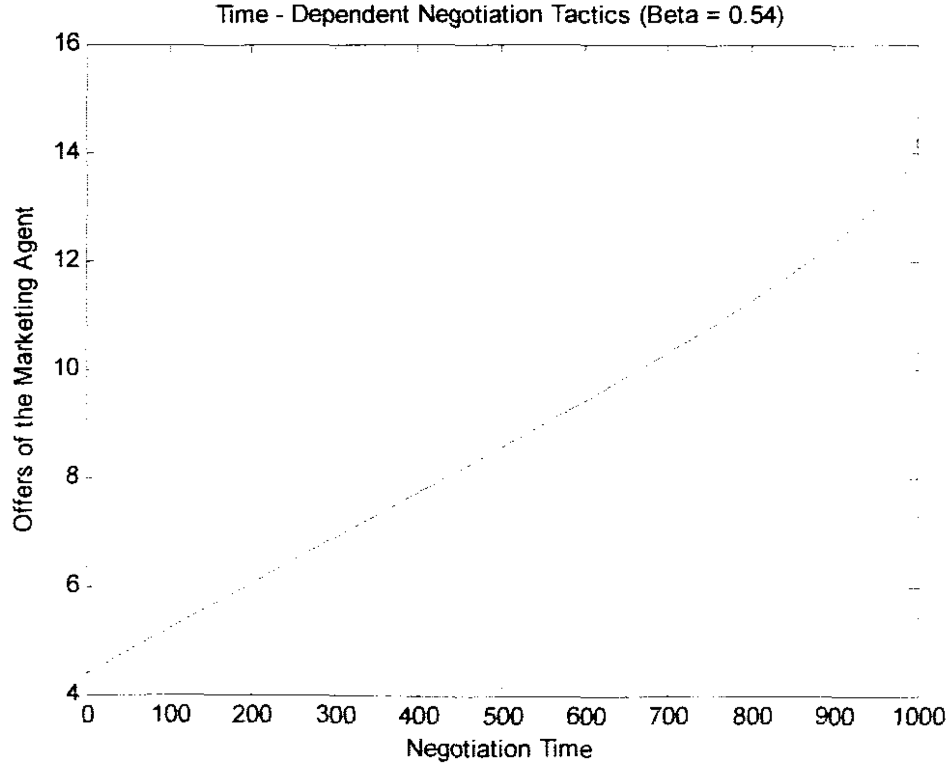
In the experiment conducted, we set the initial values of each of the parameters in the negotiation model as follows:

1. the negotiation time (number of negotiation turns) in the interval [0, 100] seconds;
2. the value of k = 0.14;
3. the value of beta = 0.60;
4. minimum value of negotiation price = 20;
5. maximum value of negotiation price = 40;
6. minimum value of negotiation quantity = 10;
7. maximum value of negotiation quantity = 100;
8. maximum value of negotiation DeliveryDate = 3;
9. maximum value of negotiation DeliveryDate = 14;

By putting these initial values into equations (53) – (59), we used MatLab to generate, and simulate responses of the negotiators as presented in [Appendix A] to [Appendix F] and the results discussed using graphs 4.1 to 4.11.

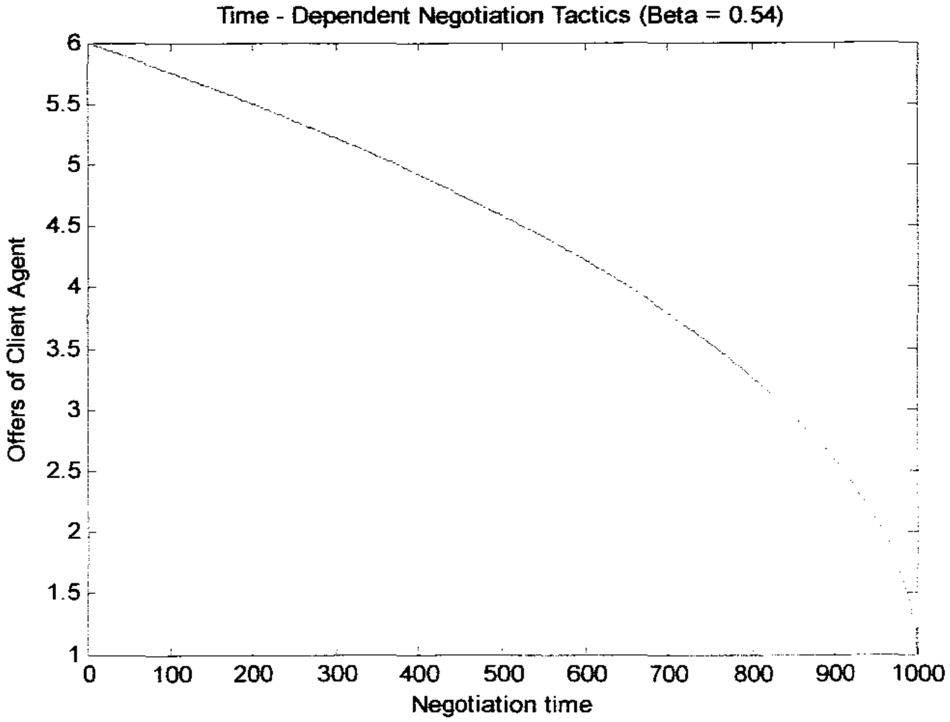
4.3.2 Discussion of Results

This section discusses the result of the experiment to test the effect of time on the behavior of a negotiator. By varying the value of β, in equations (58) and (59), the negotiating agents demonstrated two types of behaviors: (i) when the value of $\beta < 1$, we observed a Boulware Behavior; (ii) when the value of $\beta > 1$, the negotiator reach concession earlier than when $\beta < 1$. This behavior has been described as conceder behavior. A negotiator with conceder behavior urges to make a deal and reaches its reservation value quickly.



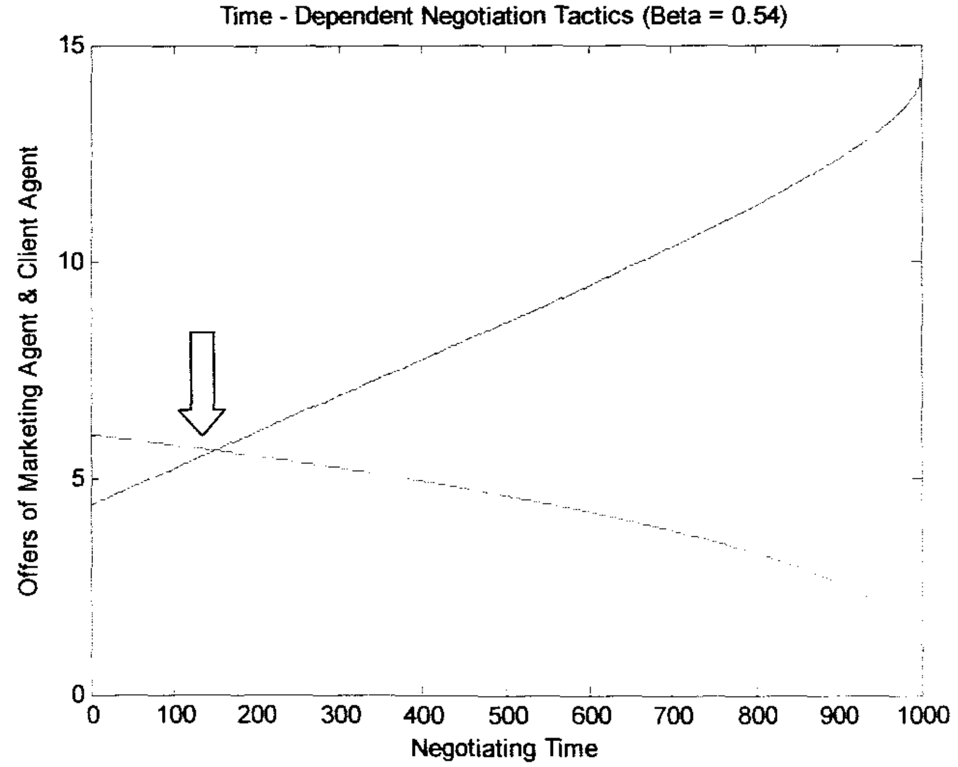
Graph 4. 1: The Negotiation Behavior of the Seller agent (Marketing agent) when β<1.

The graph 4.1 shows the negotiation price offers of the Marketing agent (Seller agent). The result shows that the negotiation offers of the seller agent increases with time.



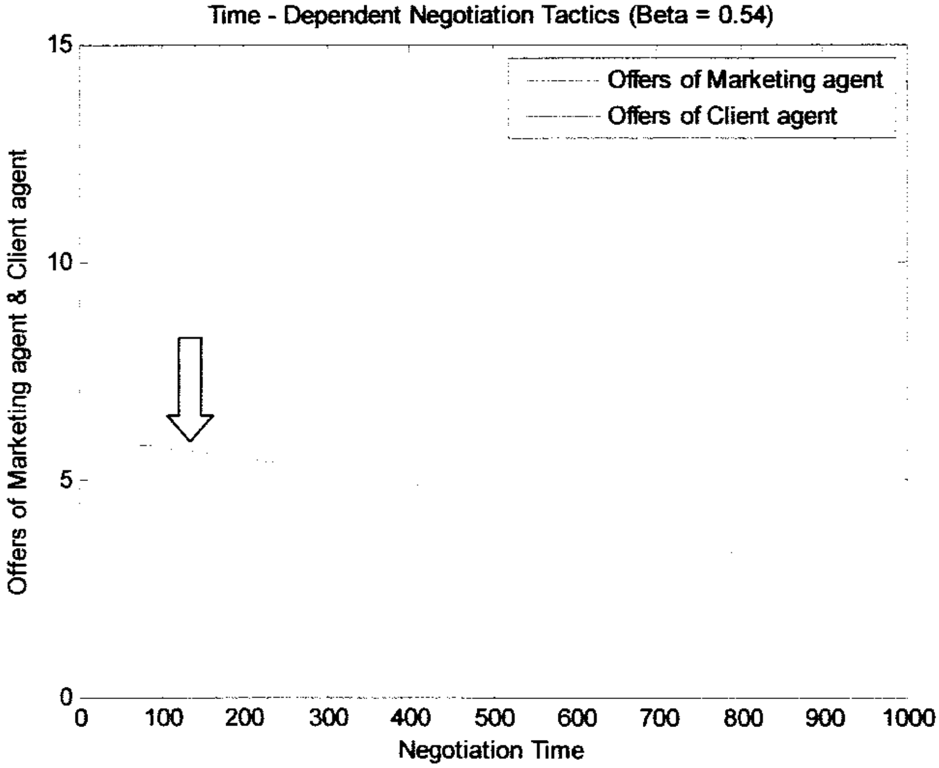
Graph 4. 2: The Negotiation Behavior of the Buyer agent (Client agent) when β<1.

The graph 4.2 shows that the negotiation price offers by the Client agent (Buyer agent) decreases with time.



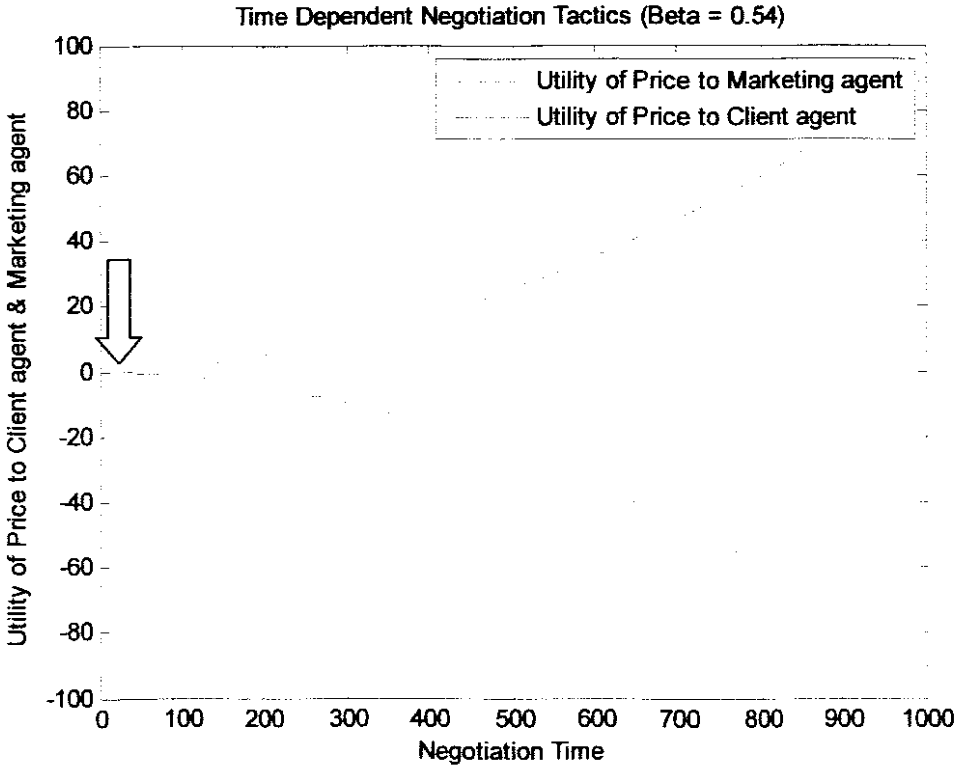
Graph 4. 3: The Negotiation Behavior of the Marketing agent and the Client agent when the value of $\beta < 1$.

The graph 4.3 shows the negotiation offers of both the Marketing agent and the Client agent. The result shows that the seller agent and the buyer agent demonstrate opposite tendencies towards the negotiation decision variable (i.e. price). While the Marketing agent offers of price increases with time, the Client agent's offers of price decreases with time.



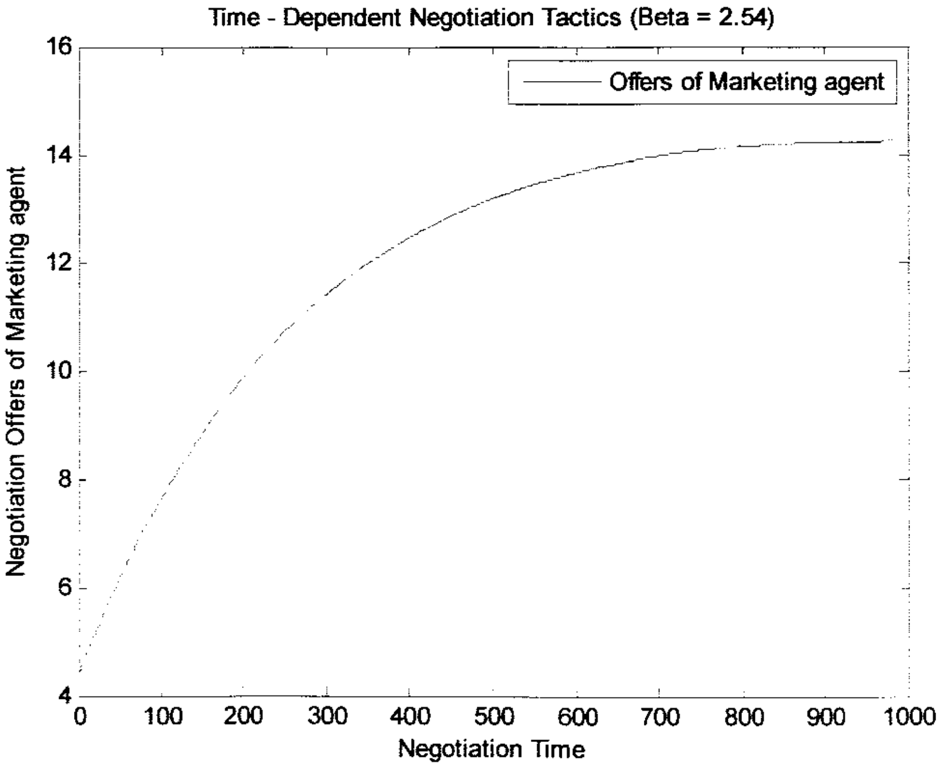
Graph 4. 4: The values attached to price by the Client-agent and the Marketing-agent

The arrow in graph 4.4 points to where both the seller agent and the buyer agent reach concession during negotiation. The result shows that there is likely for sale take place between time 0 – 150 micro seconds, since the value of price attached to the item under negotiation by the Client-agent is more than the value of price attached to the item by the Market-agent. Beyond time 150 micro seconds, the two agents no longer agree on resource allocation. The implication of this is that, the Marketing agent will readily want to sell its resources at time between 0 – 150ms as indicated on the graph.



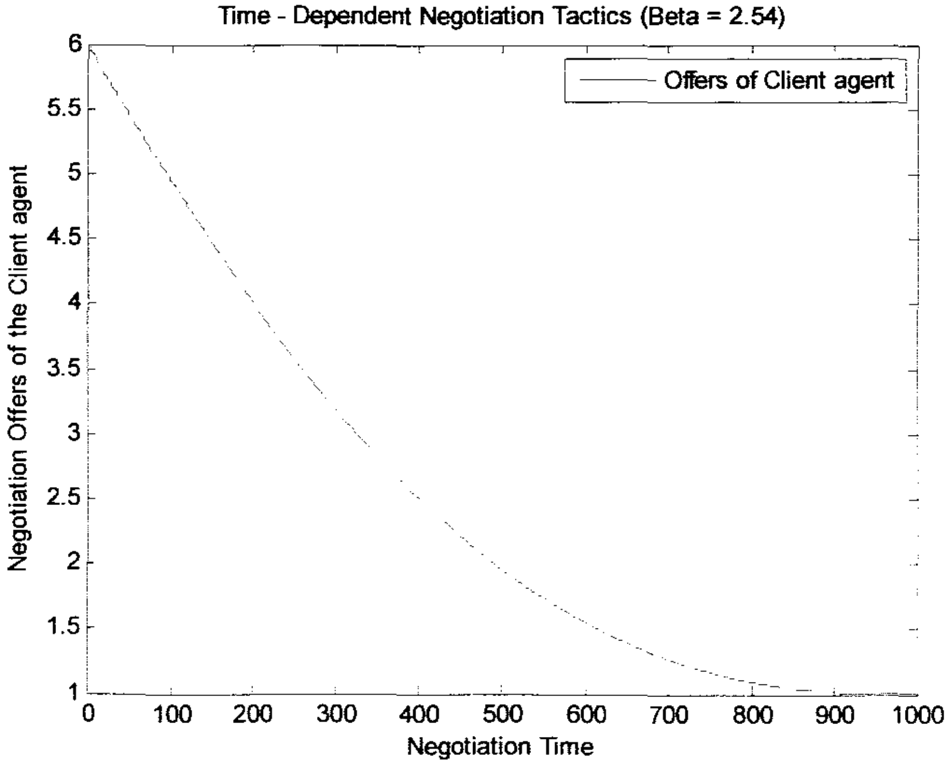
Graph 4. 5:The Graph relating the Utilities of the Marketing-agent and the Client-agent

The graph 4.5 compares the utility of the Marketing agent with the utility of the Client agent. The result shows that the utilities of the seller and the buyer agents show opposite tendencies. While the utility of the Marketing agent rises with time, the utility of the Client agent falls with time.



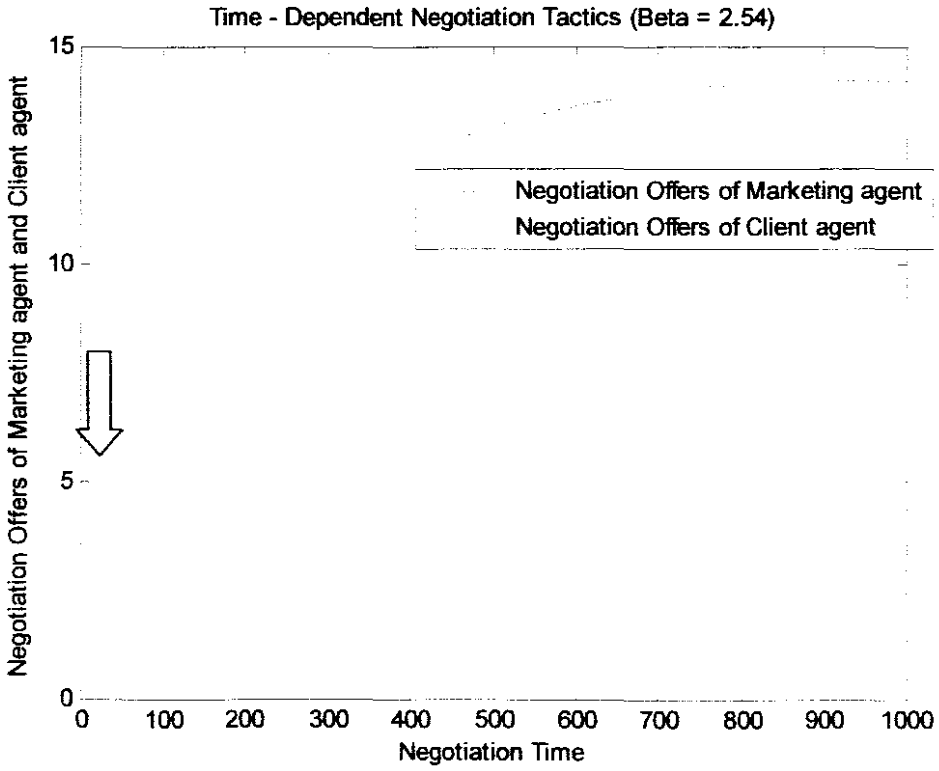
Graph 4. 6: The Negotiation Behavior of the Seller agent (Marketing agent) when β>1.

The Graph 4.6 shows the negotiation offers of the Marketing agent when the value of β = 2.54.



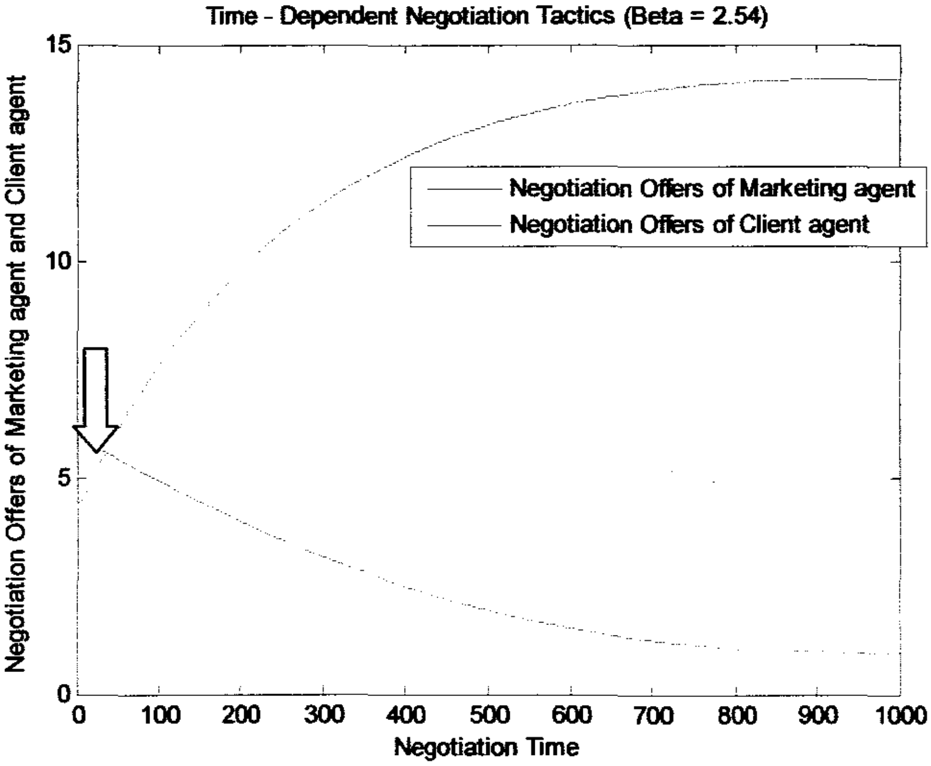
Graph 4. 7: The Negotiation Behavior of the Buyer agent (Client agent) when β>1.

Graph 4.7 shows the offers of the Client agent during a negotiation thread.



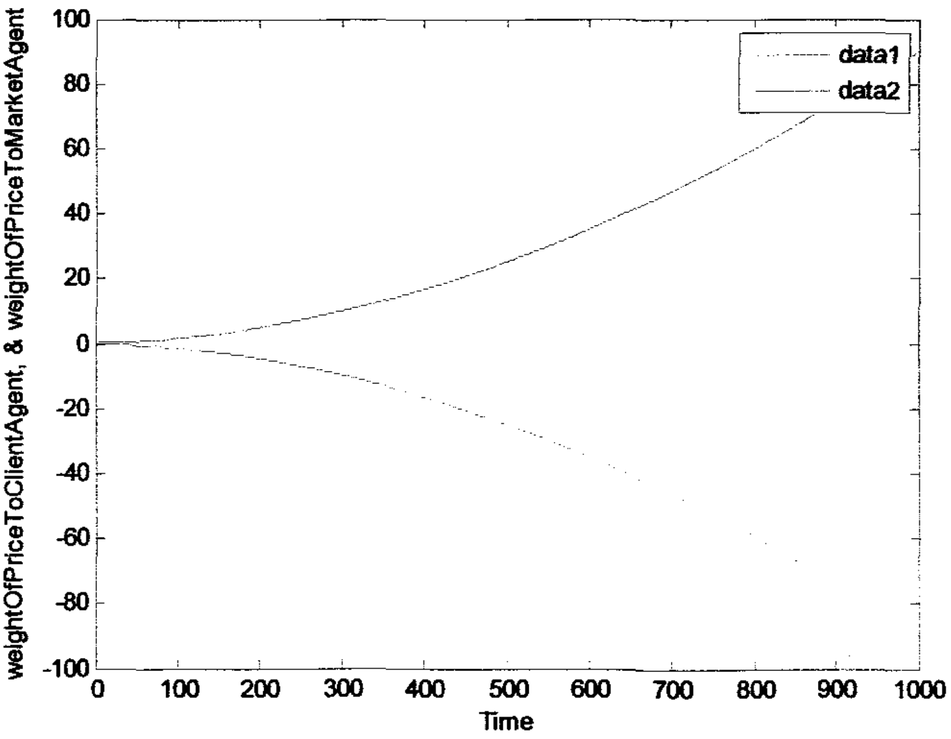
Graph 4. 8: The Negotiation Behavior of the Marketing agent and the Client agent when the value of β > 1.

Graph 4.8 shows the results of the price negotiation offers of both the Marketing agent and the Client agent during a negotiation thread, and when the value of β = 2.54. The result shows that the seller agent and the buyer agent demonstrate opposite tendencies towards the issue of negotiation (i.e. price).While the Marketing agent offers of price increases with time, the Client agent's offers of price decreases with time. Agreement is only reached at the early stage of the negotiation where the price offered by the Client agent is equal to the price offered by the Marketing agent. The arrow on the graph points to the agreement point. That is to say that the two agents agree on resource allocation at this point. Beyond the point of intersection of the two curves no agreement is found.

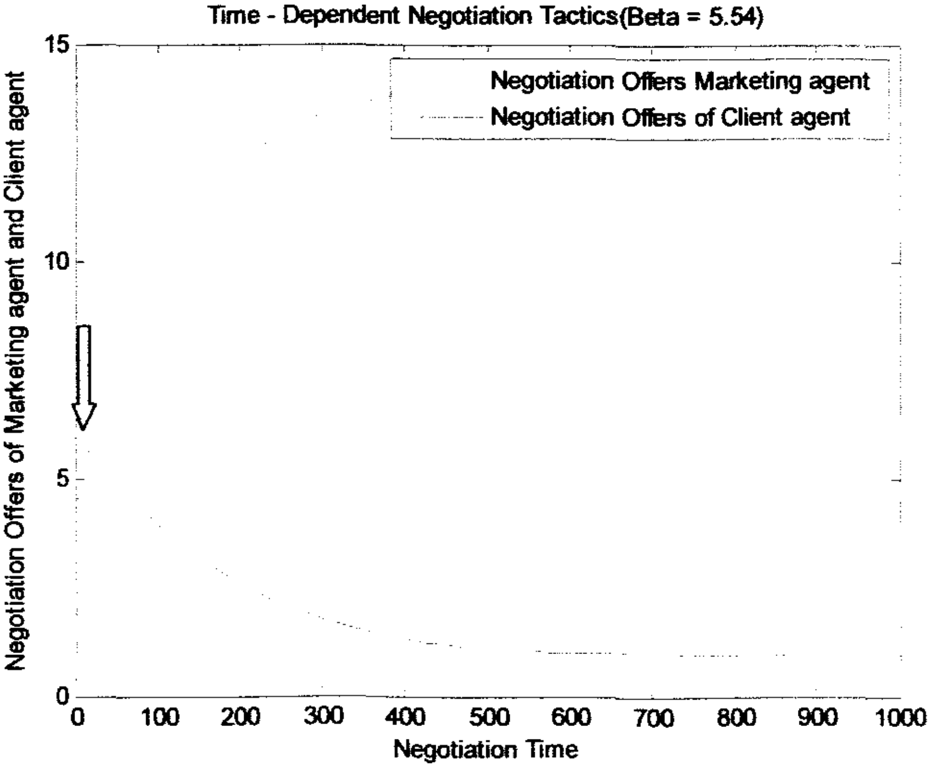


Graph 4. 8: The Negotiation Behavior of the Marketing agent and the Client agent when the value of β > 1.

Graph 4.8 shows the results of the price negotiation offers of both the Marketing agent and the Client agent during a negotiation thread, and when the value of β = 2.54. The result shows that the seller agent and the buyer agent demonstrate opposite tendencies towards the issue of negotiation (i.e. price).While the Marketing agent offers of price increases with time, the Client agent's offers of price decreases with time. Agreement is only reached at the early stage of the negotiation where the price offered by the Client agent is equal to the price offered by the Marketing agent. The arrow on the graph points to the agreement point. That is to say that the two agents agree on resource allocation at this point. Beyond the point of intersection of the two curves no agreement is found.

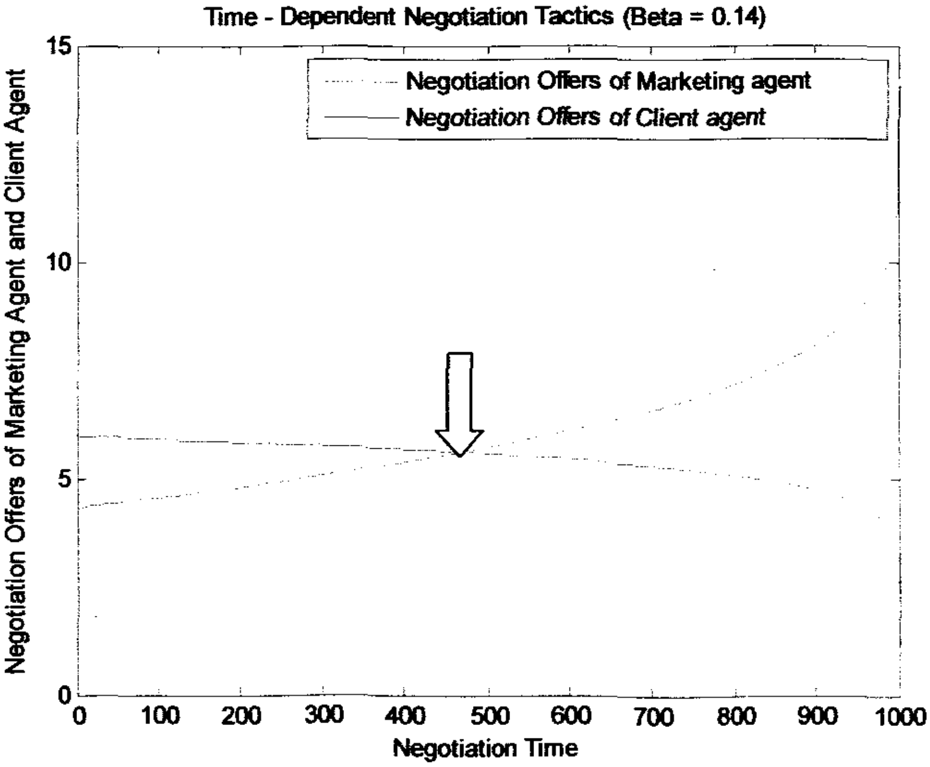


Graph 4. 9: The weight the two agents attached to price shows opposite tendencies.



Graph 4. 10: The Negotiation Offers of the Negotiators when β >1.

The Graph 4.10 show the results of the negotiation offers of the Marketing agent and the Client agent, when the value β = 5.54.



Graph 4. 11: The Negotiation Offers of the Negotiators when β < 1.

The Graph 4.11 show the results of the negotiation offers of the Marketing agent and the Client agent, when the value β = 0.14.

Comparing the negotiation offers of the Marketing agent in [Graph 4.1, when the value of β = 0.54] with the negotiation offers of the Marketing agent in [Graph 4.6, when the value of β = 2.54], shows that the greater the value of β, the greater the convexity of the curve of the graph of the negotiation offers.

The results in [Graph 4.4], [Graph 4.8], [Graph 4.10] and [Graph 4.11] demonstrated that the greater the value of β, the shorter, the time it takes for negotiators to reach concession, and the smaller the value of β, the longer, the time it takes for negotiators reach concession during negotiation.

The [Table 4.1] to [Table 4.3] show the negotiation offers of the Marketing agent and the Client agent as we vary the value of β in the negotiation model of equations (58) and (59). These responses of the negotiators were plotted and result discussed as in graphs 4.1 to 4.11.

Table 4 1: Price Negotiation Offers of the Marketing agent & the Client agent

	β = 0.14		β = 5.54	
Time 1.0*e+003	*OfferOfMarketingAgent*	*OfferOfClientAgent*	*OfferOfMarketingAgent*	*OfferOfClientAgent*
0	4.3600	6.0000	4.3600	6.0000
0.0002	4.3604	5.9999	4.3773	5.9952
0.0004	4.3609	5.9998	4.3946	5.9904
0.0006	4.3613	5.9996	4.4118	5.9857
0.0008	4.3617	5.9995	4.4290	5.9809
0.0010	4.3622	5.9994	4.4462	5.9761
0.0012	4.3626	5.9993	4.4633	5.9714
0.0014	4.3630	5.9992	4.4804	5.9666
0.0016	4.3635	5.9990	4.4975	5.9618
0.0018	4.3639	5.9989	4.5145	5.9571
0.0020	4.3644	5.9988	4.5315	5.9523
0.0022	4.3648	5.9987	4.5485	5.9476
0.0024	4.3652	5.9985	4.5654	5.9428
0.0026	4.3657	5.9984	4.5823	5.9381
0.0028	4.3661	5.9983	4.5992	5.9333
0.0030	4.3666	5.9982	4.6160	5.9286
0.0032	4.3670	5.9981	4.6328	5.9239
0.0034	4.3674	5.9979	4.6496	5.9191
0.0036	4.3679	5.9978	4.6663	5.9144
0.0038	4.3683	5.9977	4.6830	5.9097
0.0040	4.3688	5.9976	4.6997	5.9049
....	....	....	....	....
....	....	....	....	....
....	....	....	....	....
0.9974	11.1143	3.3506	14.2464	1.0000
0.9976	11.1573	3.3260	14.2464	1.0000
0.9978	11.2032	3.2995	14.2464	1.0000
0.9980	11.2525	3.2708	14.2464	1.0000
0.9982	11.3060	3.2395	14.2464	1.0000
0.9984	11.3645	3.2050	14.2464	1.0000
0.9986	11.4292	3.1664	14.2464	1.0000
0.9988	11.5018	3.1227	14.2464	1.0000
0.9990	11.5849	3.0721	14.2464	1.0000
0.9992	11.6828	3.0117	14.2464	1.0000
0.9994	11.8029	2.9362	14.2464	1.0000
0.9996	11.9614	2.8345	14.2464	1.0000
0.9998	12.2062	2.6721	14.2464	1.0000
1.0000	14.2464	1.0000	14.2464	1.0000

Table 4 2: Price Negotiation Offers & Utilities of the Negotiators

β = 0.54				
Time (*1.0*e+003*)	Price Offers Of Marketing Agent	Price Offer Of Client Agent	Utility Of Price To Marketing gent	Utility Of Price To ClientAgent
0	4.3600	6.0000	1.3365	-0.1750
0.0002	4.3617	5.9995	1.3365	-0.1750
0.0004	4.3634	5.9991	1.3365	-0.1750
0.0006	4.3650	5.9986	1.3365	-0.1749
0.0008	4.3667	5.9981	1.3365	-0.1749
0.0010	4.3684	5.9977	1.3366	-0.1749
0.0012	4.3701	5.9972	1.3367	-0.1748
0.0014	4.3718	5.9967	1.3369	-0.1747
0.0016	4.3735	5.9963	1.3371	-0.1746
0.0018	4.3752	5.9958	1.3373	-0.1745
0.0020	4.3769	5.9953	1.3375	-0.1744
0.0022	4.3786	5.9949	1.3377	-0.1743
0.0024	4.3802	5.9944	1.3380	-0.1742
0.0026	4.3819	5.9939	1.3383	-0.1740
0.0028	4.3836	5.9935	1.3386	-0.1739
0.0030	4.3853	5.9930	1.3390	-0.1737
0.0032	4.3870	5.9925	1.3393	-0.1736
0.0034	4.3887	5.9921	1.3397	-0.1734
0.0036	4.3904	5.9916	1.3401	-0.1732
0.0038	4.3921	5.9911	1.3406	-0.1730
0.0040	4.3938	5.9907	1.3410	-0.1728
....	....	....	....	....
....	....	....	....	....
....	....	....	....	....
0.9974	14.0104	1.2299	117.1652	83.6802
0.9976	14.0206	1.2202	117.1625	83.7545
0.9978	14.0312	1.2102	117.1580	83.8305
0.9980	14.0421	1.1997	117.1515	83.9085
0.9982	14.0536	1.1887	117.1428	83.9888
0.9984	14.0657	1.1771	117.1315	84.0718
0.9986	14.0784	1.1648	117.1169	84.1579
0.9988	14.0920	1.1517	117.0986	84.2478
0.9990	14.1067	1.1375	117.0754	84.3426
0.9992	14.1227	1.1220	117.0460	84.4435
0.9994	14.1407	1.1045	117.0078	84.5533
0.9996	14.1616	1.0840	116.9560	84.6765
0.9998	14.1882	1.0578	116.8785	84.8252
1.0000	14.2464	1.0000	116.6580	85.1156

Table 4 3: Price Negotiation Offers of the Marketing agent & Client agent

Beta = 2.54		
Time 1.0*e+003	OfferOfMarketingAgent	OfferOfClientAgent
0	4.3838	55.6400
0.0002	4.3917	55.6321
0.0004	4.3996	55.6241
0.0006	4.4075	55.6162
0.0008	4.4154	55.6083
0.0010	4.4233	55.6004
0.0012	4.4312	55.5925
0.0014	4.4391	55.5846
0.0016	4.4470	55.5767
0.0018	4.4549	55.5688
0.0020	4.4627	55.5609
0.0022	4.4706	55.5530
0.0024	4.4784	55.5451
0.0026	4.4863	55.5373
0.0028	4.4941	55.5294
0.0030	4.5019	55.5216
0.0032	4.5098	55.5137
0.0034	4.5176	55.5059
0.0036	4.5254	55.4981
0.0038	4.5332	55.4902
0.0040	4.5410	55.4824
....	....	....
....	....	....
....	....	....
0.9974	14.2464	45.7536
0.9976	14.2464	45.7536
0.9978	14.2464	45.7536
0.9980	14.2464	45.7536
0.9982	14.2464	45.7536
0.9984	14.2464	45.7536
0.9986	14.2464	45.7536
0.9988	14.2464	45.7536
0.9990	14.2464	45.7536
0.9992	14.2464	45.7536
0.9994	14.2464	45.7536
0.9996	14.2464	45.7536
0.9998	14.2464	45.7536
1.0000	14.2464	45.7536

4.4 Performance Evaluation of the Co-operative Modeler

In the previous chapters and sections, we discussed extensively about the Co-operative Modeler designed for mediating resource negotiation between resource providers and consumers. The design and implementation prototype of the modeler has also been presented. The Co-operative Modeler might grow from few integrated resources/resource providers to millions. This of course raises the problem of scalability. The problem might lead to potential performance degradation. Consequently, such an application as the Co-operative Modeler that required a large number of geographically located resources or resource providers must be designed to be latency and bandwidth tolerant. For this reason, we consider in this section the issue of scalability.

4.4.1 Scalability

According to (Neuman, 1994), scalability of a system such as the Co-operative Modeler can be measured along at least three different dimensions. First, a system can be scalable with respect to its size, meaning that we can easily add more users and resources to the system. Second, a geographically scalable system is one in which the users and resources may lie far apart. Third, a system can be administratively scalable, meaning that it can still be easy to manage even if it spans many independent administrative organizations.

We know from the knowledge gathered from literature that when a system needs to scale, very many different types of problems need to be solved. First, let us consider scalability with respect to size. If more users or resources need to be supported, we are often confronted with the problems associated with centralized services, data, and algorithms. For example, many services are centralized in the sense that they are implemented by means of only a single server running on a specific machine in the geographically distributed system. The problem with this scheme is obvious: the server can simply become a bottleneck as the numbers of users grow. Even if we have virtually unlimited processing and storage capacity, communication with that server will eventually prohibit further growth.

Geographical scalability has its own problems. One of he major reasons why it is currently very difficult to scale existing geographically distributed systems that were designed for local-area networks is that they are based on synchronous communication. In this form of communication, a party requesting for service, generally referred to as a client, blocks communication until a reply is sent back. This approach generally works well in Local Area Networks (LANs) where communication between two machines is generally at worst a few hundred microseconds. However, in a wide-area system, we need to take into consideration that inter- process communication may be hundreds of milliseconds, three orders of magnitude slower.

Another problem that hinders geographical scalability is that communication in wide-area networks is inherently unreliable, and virtually always point-to-point. In contrast, local-area networks generally provide highly reliable communication facilities based on broadcasting, making it much easier to develop distributed systems. For example, consider the problem of locating a service. In a local-area system, a process can simply broadcast a message to every machine, asking if it is running the service it needs. Only those machines that have that service respond, each providing its network address in the reply message. Such a location scheme will not work well in a wide-area system. Instead, special location services need to be designed, which may need to scale worldwide and be capable of servicing several billion of users.

Geographical scalability is strongly related to the problems of centralized solutions that hinder size scalability. If we have a system with many centralized components, it is clear that geographical scalability will be limited due to the performance and reliability problems resulting from wide-area communication. In addition, central components now lead to a waste of network resources.

Finally, a difficulty arises, and in many cases this raises the open question of how to scale a geographically distributed systems across multiple, independent administrative domains. A major problem that needs to be solved is that of conflicting policies with respect to resource usage (and payment), management, and security.

This study does not pretend to have solved all the scalability problems that may arise in the Co-operative Modeler. However, we make some basic assumptions. The Grid Infrastructure upon which the Co-operative Modeler is built is a scalable computing environment. The assumption is that most of the scalability problems have been taken care of by the Grid Infrastructure on top of which the Co-operative Modeler is built. Not only that, one plausible approach that the Co-operative Modeler adopt in order to address the concern of scalability is decentralization. The Co-operative Modeler is designed to comprise of geographically distributed autonomous resource providers and consumers. As against the traditional clients – server model which can be a performance bottleneck and a single point of failure, the Co-operative Modeler agents co-ordinate computing resources without using centralized servers. This permits the Co-operative Modeler computing to scale more effectively than the traditional client – server. Decentralization is attractive with respect to scalability and fault tolerance.

4.4.2 Experiment to Measure the Performance of the Co-operative Modeler

Queue length, delays and losses are more meaningful performance metrics in the network contexts. Queues provide the most intuitive language for describing traffic and its dependence structure. Since both delay and loss are functions of queuing dynamics, synthetic network traffic must result in a realistic queuing process in order to obtain relevant results from experiments. Realistic queues are also needed in experiments that study for instance, routine capacity planning, quality of service guarantees, and denial of service attacks (Henrique Lopes et al.,). This subsection presents an experimental set up to capture the network traffic in the Co-operative Modeler resource market by using Fractional Brownian Motion to model the Network traffic. The model was simulated and the result of the simulation is presented.

4.4.2.1 Design of the Experiment

This study used **Fractional Brownian Motion** (Rolls, et al., 2004) to capture the characteristics of the network traffic at the Co-operative Modeler resource market. Let us consider execution time at the Co-operative resource pool [Figure 3.1] as being divided into "slots" of fixed length (e.g. 10ms) Denote the amount of task that arrives at the resource pool in slot k by

$$X_k = Y_k + m \tag{60}$$

where $E[X_k] = 0$ and $m > 0$, It is assumed that

Y_k is a stationary, ergodic process (Ergodic process is a random process in which the time series produced are the same in statistical properties).. The quantity Y_k captures the tasks of the clients and m the mean amount of task that arrives in each time slot. The total cumulative input to the Co-operative pool for execution in time {1, 2, . . . n} is

$$A_k = \sum_{k=1}^{n} X_k . \tag{61}$$

Suppose that the Execution Agent at Co-operative resource pool [Figure 3.1] can process C units of task in each time slot. Then the net input in slot k is $X_k - C$ and a net input process can be defined as

$$I_k = A_k - C_n = \sum_{k=1}^{n} X_k - C_n . \tag{62}$$

The queue length process can be defined through the **Lindley recurrence formula** qiven by

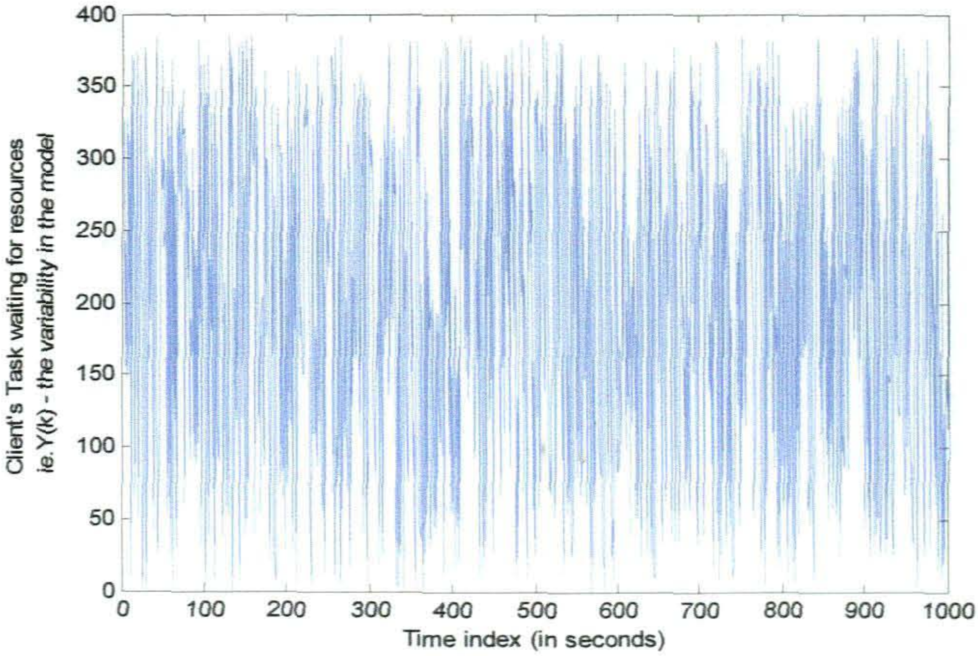
$$Q_0 = 0 \text{ and } Q_n = (Q_{n-1} + X_n - C)_+, \tag{63}$$
$$n = 1, 2, \ . \ . \ .$$

4.4.2.2 Simulation Result of the Experiment

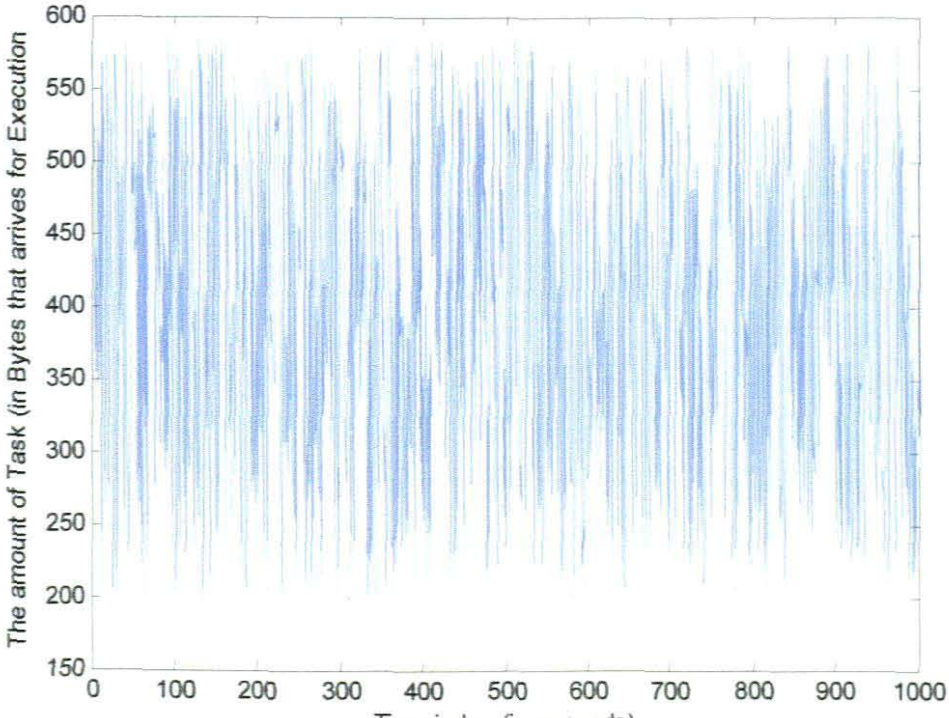
This sub section presents the results of the analysis of the Network *Traffic at the Co-*operative resource pool. The results, as shown in [Table 4.4] and [Graphs 4.12] to [Graphs 4.15], were obtained by using MatLab to simulate the models presented in equations (60) – (63) to capture the characteristics of the Network Traffic at the Co-operative Modeler Pool.

Table 4 4: Queue Analysis As a result of the Network Traffic at the Co-operative Resource Pool

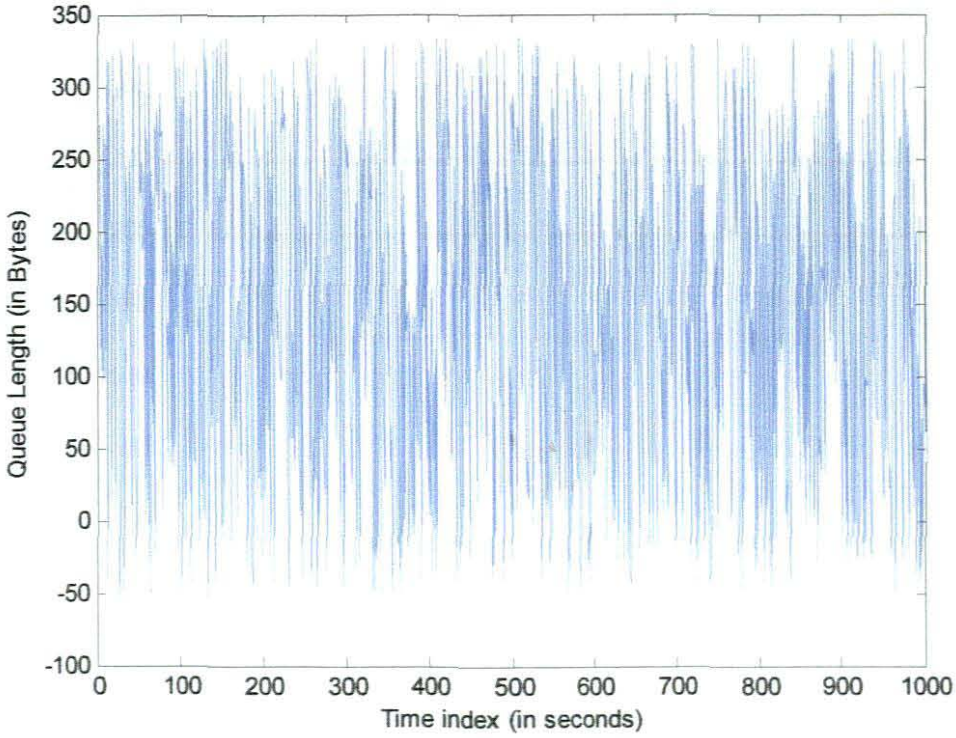
K	Q	Y	X	M	C
1	-26.5255	31.3308	223.4745	194.1823	250
2	58.8206	116.6768	308.8206	194.1823	250
3	11.8131	69.6694	261.8131	194.1823	250
4	219.2999	277.1562	469.2999	194.1823	250
5	247.9762	305.8325	497.9762	194.1823	250
6	70.3675	128.2238	320.3675	194.1823	250
7	244.6290	302.4853	494.6290	194.1823	250
8	242.7108	300.5671	492.7108	194.1823	250
9	-17.3399	40.5164	232.6601	194.1823	250
10	69.4588	127.3151	319.4588	194.1823	250
20	43.1917	101.0480	293.1917	194.1823	250
11	92.4543	150.3105	342.4543	194.1823	250
12	289.4273	347.2836	539.4273	194.1823	250
13	238.8866	296.7429	488.8866	194.1823	250
14	36.6842	94.5405	286.6842	194.1823	250
15	118.0923	175.9486	368.0923	194.1823	250
16	47.5461	105.4024	297.5461	194.1823	250
17	105.9149	163.7712	355.9149	194.1823	250
18	212.6493	270.5056	462.6493	194.1823	250
19	258.4900	316.3462	508.4900	194.1823	250
20	261.6023	319.4586	511.6023	194.1823	250
21	98.2072	156.0635	348.2072	194.1823	250
22	194.3125	252.1687	444.3125	194.1823	250
23	1.1346	58.9909	251.1346	194.1823	250
24	73.6873	131.5436	323.6873	194.1823	250
………	…………	…………	………	…………	…..
………	…………	…………	………	…………	……
………	…………	…………	………	…………	……
987	92.2424	50.0987	342.2424	194.1823	250
988	57.8771	115.7333	307.8771	194.1823	250
989	37.9459	95.8022	287.9459	194.1823	250
990	-38.3632	19.4931	211.6368	194.1823	250
991	156.6254	214.4817	406.6254	194.1823	250
992	189.4811	247.3374	439.4811	194.1823	250
993	24.5220	82.3782	274.5220	194.1823	250
994	29.7755	87.6318	279.7755	194.1823	250
995	-32.1277	25.7286	217.8723	194.1823	250
996	153.8113	211.6675	403.8113	194.1823	250
997	77.7710	135.6273	327.7710	194.1823	250
998	-34.8857	22.9705	215.1143	194.1823	250
999	192.5330	250.3893	442.5330	194.1823	250
1000	318.7311	376.5874	568.7311	194.1823	250



Graph 4. 12: Graph showing the result of clients' tasks waiting to be passed into queue to be process.

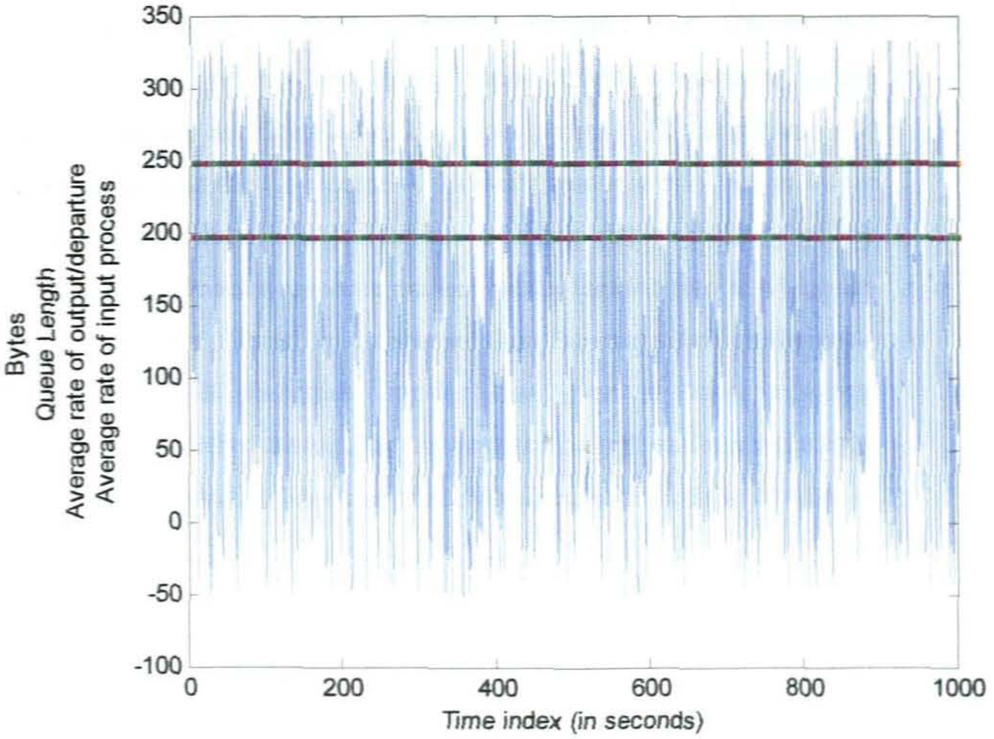


Graph 4. 13: Tasks Arrival Rate for Execution at the Co-operative Pool



Graph 4. 14: The Graph relating the Length of Queue against Time

The result shows the rate of Queue Length growth at the Co-operative Modeler resource pool.



Graph 4. 15: Performance Evaluation Result of the Co-operative Modeler.

The result [Graph 4.15], compares the queue length, average rate of output/departure, and the average rate of input process at the Co-operative Modeler resource pool.

4.5 Conclusion

This chapter presented an implementation prototype of the Co-operative Modeler. An evaluation of the negotiation behavior of the Co-operative agents was carried out using the Time – Dependent Negotiation Tactics. The results of the evaluation demonstrated two types of behaviors: (i) Boulware behavior was observed when $\beta < 1$. In Boulware behavior, the negotiators stay firm at the beginning of a negotiation, while concessions are only made when the deadline is almost reached. (ii) Conceder behavior was observed when $\beta > 1$. A negotiator with conceder behavior urges to make a deal and reaches its reservation value quickly.

In the chapter, a performance evaluation of the Co-operative Modeler was also carried out by using Fractional Brownian Motion to capture the characteristics of the network traffic at the

Co-operative Modeler resource market. The results, [Graph 4.12] to Graph [4.15] described the queue analysis of the Network Traffic at the Co-operative Resource Pool. The analysis showed that ($C > m$, where C = 250, and m = 200 on the Graph 4.15) the average rate of the Output/departure process exceeds the average rate of the input process. The implication of this is that the queue length process at the Co-operative Modeler resource pool has a proper steady-state distribution Q and the queue is characterized as being stable. Stability of the network traffic at the Co-operative Modeler resource market means that the rate at which the Execution-agent at the pool processes clients' tasks is more than the rate at which the tasks arrive the pool for execution. It therefore implies that even if more resource providers/consumers join the Co-operative market, the network traffic will still remain stable. We therefore conclude that the Co-operative Modeler guarantees Scalability of the number of users.

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

5.1 Introduction

The purpose of the study was to establish a negotiation framework that enables optimal allocation of resources in a grid computing environment, such that· clients are allowed to have an on – demand access to pool of resources; collaboration is enhanced among resource providers; and cost saving and efficiency are ensured in resource allocation. The objectives to realize this purpose were: first, to design an appropriate negotiation model that could be adopted in order to achieve optimal resource allocation. Another objective is to determine an effective search strategy that could be employed in order to reach a Pareto efficient negotiation solution. Finally, the third objective is to adopt a negotiation strategy or tactics that negotiators could use to arrive at an optimal resource allocation. In order to achieve the goals and objectives set for the study, therefore, we used the following methodologies: (i) a critical survey of the existing economic approach and models for negotiating grid resources was conducted, by first categorizing the economic models into competitive and non - competitive market settings, and then structure the models into auction models. Pricing Models, Bartering or Exchange Models, Bilateral Negotiation Models, and Monopoly/Oligopoly Models. (ii) The knowledge gathered from the literature survey was used to construct a novel model called Co-operative Modeler for mediating grid resources sharing negotiation. We used formal notations: to construct a theoretical model for allocation of resources to clients' task, and to present a novel Combinatorial Multi-Objective Optimization Model (CoMbO), by modeling negotiation offers of agents as a multi-objective optimization problem. Finally, to present Genetic and Bayesian Learning Algorithms for implementing the model presented; (iii) an implementation prototype of the Co-operative Modeler was carried out by: implementing the Co-operative Modeler to mimic the real world negotiation; and used time–dependent negotiation tactics to evaluate and simulate the negotiation behavior of the Co-operative agents.

5.2 Conclusion

The study presented Co-operative Modeler for mediating resource negotiation between providers and consumers of resources. The Co-operative Modeler guarantees: (i) Scalability of number of users. i.e. multiple users can access a virtualized pool of resources in order to obtain the best possible response time overall by maximizing utilization of the computing resources.(ii) Enhanced collaboration.– that is promoting collaboration so that grid resources can be shared and utilized collectively, efficiently, and effectively to solve computer–intensive problems. (iii) Improved business agility - that is decreasing time to process data and deliver quicker results. (iv) Cost saving. – i.e. leveraging and exploiting unutilized or under utilized power of all computing resources within a grid environment.

The result of our simulation revealed that the Co-operative Modeler agents manifested two types of behavior: Boulware Behavior, and Conceder Behavior. We observed from the results of the simulation that the value of beta (β) from the simulation model used, determines how long it takes for negotiators to reach agreement during a negotiation process: the smaller the value of beta (β), the longer the time it takes for negotiators to agree over resource allocation, and the larger the value of beta (β), the shorter the time it takes to agree over resource allocation.

5.3 Recommendation

The global trend in the business world today is collaboration among business partners. The competence of a virtual business enterprise is dependent on how well these partners collaborate. The virtual business enterprise is a network of several resource providers which contribute their core competences and share resources such as information, knowledge, and market access in order to exploit fast – changing market opportunities. We will not be exaggerating, therefore, if we maintain the fact that for any business to stay relevant in the next generation global grid business venture, that the business must collaborate with other business partners. It is even more required that the Small Medium and Micro Enterprise (SMME) must come together in a Co-operative relationship to do business. The benefits from such relationship can not be over stretched.

Our recommendation for the next generation utility service providers, therefore, is collaboration and yet more collaboration. The utility grid service providers must come together in a co-operative relationship to make services available in a more affordable and timely manner to the end users.

5.4 Future Work

Grids and agent communities offer two approaches to open distributed systems. Grids focus on robust infrastructure and services, and agents on the operation of autonomous intelligent problem solvers. The interests of grids and agents converge when efficient management of the services provided by grids comes to the forefront. Cooperation, coordination and negotiation are issues that Grid users and Grid resource providers need to address in order to successfully interoperate. There is the need to develop an agent-based Grid Resource Management (GRM) system compliant with adopted standards for both Grid and Agent computing. The Grid Resource Management problems include the management of user requests, their matching with suitable resources through service discovery, and the scheduling of tasks on the matched resources. A high level of complexity arises from the various issues Grid Resource Management must address: resource heterogeneity, site autonomy and security mechanisms, the need to negotiate between resource users and resource providers, the handling of the addition of new policies, scalability, fault tolerance and QoS, just to cite the most relevant.

This study presented Co-operative Modeler for sharing e – business resources. The objective of the Co-operative Modeler is to provide the basic mechanisms for forming and operating dynamic distributed virtual organizations. While the grid infrastructure focused on the means for discovering and monitoring dynamic services, managing faults and failures, creating and managing service level agreements, creating and enforcing dynamic policy, etc. There is the need for basic theoretical model that will enable autonomous co-operative members to interact with one another with partial knowledge and have a robust desirable behavior. It seems most likely therefore, that agent technology will play a vital role in the development of the Co-operative Modeler as a pervasive infrastructure.

The future work will explore the relationship of intelligent agents to the Co-operative Modeler, how agent technology can be applied to enhance effective interaction among co-operative members, and address challenges faced by Grid infrastructure and applications.

REFERENCES

Abramson, D., Giddy, J. & Kotler, L., (2000). High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?, IPDPS'2000, Mexico, IEEE CS Press, USA .

AccessGrid, http://www.accessgrid.org (accessed November 2006).

Aremu, D. R. & Adigun, M. O., (2006). Peer to Peer Architecture for Sharing e-Business Resources, Work-in Progress, Abstract Book & Technical Programme Pp. 17 Southern African Telecommunication Networks and Application Conference (SATNAC) 2006. Cape Town, South Africa.

Aremu, D. R. & Adigun, M. O., (2007). A Grid-Enabled e-Business Resource Sharing With Cooperative Agent, Abstract Book & Technical Programme Pp. 13 Southern African Telecommunication Networks and Application Conference (SATNAC) 2007, Sugar Beach Resort, Mauritius.

Aremu, D. R. & Adigun, M. O., (2007). A Grid-Enabled e-Business Resource Sharing in a Cooperative Market Setting, Proceeding of The 2007 International Conference On Grid Computing & Applications. Pages. 25 – 31 WORLDCOMP' 07, Las Vegas Nevada, USA.

Aremu, D. R. & Adigun, M. O., (2008). A Coordination Framework for Sharing Grid Resources, Proceeding of The 2007 International Conference On Grid Computing & Applications, WORLDCOMP' 08, Las Vegas Nevada, USA.

Bai, X. Kresimir, H., Sivoncik, Turgut, D., & Boloni, L., (2006). "Grid Coordination with Marketmaker Agents", *International Journal of Computational Intelligence*, Vol. 3, No. 2.

Bai, X., (2006). "Grid Coordination with Marketmaker Agents", *International Journal of Computational Intelligence*, Vol. 3, No. 2 .

Bui, H. H., Kieronska, D. & Venkatesh, S. (1996). Learning other agents' preferences in multiagent negotiation. In H. Shrobe and T. Senator, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence Conference, Vol. 114-119. America Association for Artificial Intelligence*, AAAI Press.

Bui, H. H., Venkatesh, S. & Kieronska, D., (1999). Learning other agents' preferences in multiagent negotiation using the Bayesian classifier. *International Journal of Co-operative Information Systems*, 8(4):275-293.

Buyya, R. & Venugopal, S., (2004). The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report, *Proceedings of the First IEEE International Workshop on Grid Economics and Business Models* (GECON 2004, April 23, 2004, Seoul, Korea), 19-36pp, ISBN 0-7803-8525-X, IEEE Press, New Jersey, USA.

Buyya, R., & Venugopal, S., (2005, July). A Gentle Introduction to Grid Computing and Technologies.

Buyya, R., (2002, April). "Economic-based Distributed Resource Management and Scheduling for Grid Computing", Unpublished Ph. D. Thesis, School of Computer Science and Software Engineering, Monash University. Melbourne, Australia.

Buyya, R., Abramson, D. & Giddy, J., (2001, April). An Economy Grid Architecture for Service-riented Grid Computing, 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), with IPDPS 2001, SF, California, USA.

Buyya, R., Stockinger, H., Giddy, J. & Abramson, D., (2001). Economic Models for Management of Resources in Peer-to-Peer and Grid Computing, Technical Track on Commercial Application for High-Performance Computing, SPIE International Symposium on The Convergence of Information Technologies and Communication (ITCom 2001), Denver, Colorado, USA.

Cannataro, M., & Talia, D., (2003). The Knowledge Grid, *Communications of the ACM*, vol. 46, no. 1, pp. 89-93.

Chaudhury, A., & Kuilboer, J., (2001). e-Business and e-Commerce Infrastructure: Technologies Supporting the E-Business Initiative, McGraw – Hill Higher Education 1st Edition, pp.341-342.

Chiu, C. F. et al., (2002). "The Design of Interactive Negotiation Agent On the Web", Proceedings of the 22nd *International Conference On Distributed Computing Systems Workshops* (ICDSW'02).

Comuscio, A. R. & Jennings, N. R. (2003). A classification Scheme for negotiation in electronic Commerce, *Journal of Group Decision and Negotiation* 12(1): 31-56.

Comuscio, A. R., & Jennings, N. R., (2002). A classification Scheme for negotiation in electronic Commerce, *Journal of Group Decision and Negotiation* 12(1): 31-56.

Comuscio, A. R., & Jennings, N. R., (2003). A classification Scheme for negotiation in electronic Commerce, *Journal of Group Decision and Negotiation* 12(1): 31-56.

Cramton, P., Shoham, Y. & Steinberg, R., (2006). Combinatorial Auctions, MIT Press.

DevMaster.net Jeff Nve, 19/052004.

Endriss, U., (2006). "Monotonic Concession Protocols for Multilateral Negotiation", *International Conference on Autonomous Agents*, Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems Argumentation and negotiation,pages: 392 – 399 .

Fang, L., Song, Y., & Wang, Y., (2006, Oct.). "A Bilateral Negotiation Model of Technology Pricing", *Systems,Man, and Cybernetics, 1996., IEEE International Conference on Volume 3*. Issue , Page(s):2006 - 2010 vol.3, 14-17.

Foster, I., (2002). What is Grid? A Three Point Checklist", Argonne National Laboratory & University of Chicago foster@mcs.an.gov, July 20, 2002.

Gen, M. & Chang, R., (2000, January). Genetic Algorithm and Engineering Optimization , ISBN: 978-0-471-315311.

He, M., Jennings, N. R. & Leung, H., (2003). On agent-mediated electronic Commerce. *IEEE Trans. On Knowledge and Data Engineering*, 15(4): 985-1003.

He, M., Jennings, N. R., & Leung, H. (2003). On agent-mediated electronic Commerce. IEEE Trans. *On Knowledge and Data Engineering*, 15(4): 985-1003.

Henrique Lopes et al., "A Multi-Agent System for Electronic Commerce including Adaptive Strategic Behaviours", Henrique Lopes Cardoso, Max Schaefer, Eugénio Oliveira Faculdade de Engenharia, Universidade do Porto, NIAD&R-LIACC Rua dos Bragas 4099 Porto Codex, Portugal cardoso@oat.ncc.up.pt max@brooks.fe.up.pt eco@fe.up.pt

http: //agents.umbc.edu.Introduction/ao/4.stml (accessed April 2008).

IBM Redbook, (2003, September). International Support Organization, Introduction to Grid Computing with Globus.

IBM, (2005) Solutions Grid for Business Partners: Helping IBM Business Partners to Grid-enable applications for the next phase of e-business on demand, National Library Board, Singapore.

Jennings, N. R., Faratin, P., Lomuscio. A., Parsons, S., Sierra, C. & Wooldridge, M., (2001). Automated negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation*, 10(2): 199-215.

Jonathan L., Byrnes, S. & Roy D. Shapiro, Inter-company Operating Ties: Unlocking the Value In *Channel Restructuring*, Harvard Business School Working paper No. 92-058

Kraus, S., (2005, January). Strategic Negotiation in Multi-agent Environments MIT Press, 2001, *Autonomous Agents and Multi Agent Systems*, Volume 10 Issue 1 pages 91-93, Kluwer Academic Publisher, Hingham, M.A.

Krauter, K., Buyya, R. & Maheswaran, M., (2002, February). A taxonomy and survey of grid resource management systems for distributed comput'ng, *Software: Practice and Experience*, vol. 32, no. 2, pp. 135-164.

Kuchana, P. "Software Architecture Design Patterns in Java", *AUERBACH Publications* AU2142 ISBN 0 – 8493 – 2142 – 5

Maes, P., Guttman, R. & Moukas, A., (1999, March). Agents that buy and sell. *Communications of the ACM*, 42(3):81-91.

Malone, T. W., & Crowston, K., (1994, March). "The Interdisciplinary Study of Coordination ", *ACM Computing Surveys*, Vol. 26, No. 1.

McAfee & McMillian, (1987, June). Auctions and Bidding, *Journal of Economic Literature*, Vol. 25(2), 1987 pp. 699-738.

Meliksetian, D.S., Girard, J. Y., & Prost, J. P., (2004). Design and Implementation of an enterprise grid , IBM *Systems Journal*, Vol. 43, No 4.

Ong, H. S., (2003, September). "Grid Computing: Business and Policy Implications". Unpublished Master Thesis in Technology and Policy, Massachusetts Institute of Technology.

Oprea, M., (2002, September). "An Adaptive Negotiation Model for Agent-Based Electronic Commerce". *Studies in Information and Control*, Vol. 11, No. 3.

Parashar, M., & Lee, C.A. "Grid Computing: Introduction and Overview". http://www.caip.rutgers.edu/TASSL/Papers/proc-ieee-intro-04.pdf

Ping L., & Yongtong, H., (1997). "Using Counter – Proposal to Optimize Genetic Algorithm Based Co-operative Negotiation", *IEEE International Conference On Intelligence Processing Systems*, October 28 – 31, Beijing, China.

Plaszczak, P., & Wellner, R., (2005). *Grid computing*, Elsevier/Morgan Kaufmann, San Francisco.

Pruitt, D. G., (1982). "The Art and Science of Negotiation", Harvard University Press, Cambridge. USA.

Raiffa, H., (1982). The Art and Science of Negotiation, *Cambridge: Harvard University Press.*

Robinstein, A., (1982). Perfect equilibrium in a bargaining model. *Econometrica*, 50(1): 97-109.

Rolls, D. R., Michailidis, G. Hernandez-Campos, F., (2004). Queueing Analysis of Network Traffic: Theoretical Framework and Visualization Tools, Preprint submitted to Elsevier Science.

Rosenschein, J. S. & Zlotkin, G., (1994). Rules of Encounter MIT Press.

Sandholm, T., (2002). "Algorithm for Optimal winner determination in Combinatorial auctions", *Artificial Intelligence* 135 1 – 54.

Sbalzarini, I. F., Muller, S., & Koumoutsakos, P., (2000). Multi-objective optimization using evolutionary algorithms, Proceedings of the Summer Program, 63.

Schikuta, E., (2005). "Business and the Grid: Economic and Transparent Utilization of Virtual Resources". Unpublished Dissertation. Ao Univ. Prof. Dipl. –Ing. Dr Erich Schikuta, wien, im Oktober 2005.

Simchi-Levi, D., & Wu, S. D., (2004). Handbook of quantitative supply chain analysis: Modelling in the e-business era, New York, Springer.

Smith, R. & David, R., (1980). The Contract Net protocol: High level communication and control in a distributed problem solver, *IEEE Transaction on Computers* ; 29(12): 1104 – 1113.

Sycara, K., (1989). Multi-agent Compromise via negotiation. In L. Gasser and M. Hahms, editors, *Distributed Artificial Intelligence II*, pages 119-139. Morgan Kaufmann.

Sycara, K., (1989). Multi-agent Compromise via negotiation. In L. Gasser and M. Hahms, editors, *Distributed Artificial Intelligence II*, pages 119-139. Morgan Kaufmann.

Sycara, K.P., Widoff, S., Klusch, M., & Larks, J. L., (2002). "Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5(2): 173-203.

Tia, H., (2003, June). Grid Computing as an Integrated Force in Virtual Enterprises. Unpublished Thesis of Engineering in Civil and Environmental Engineering at the Massachusetts Institute of Technology.

Venkatraman, N. and Henderson, J.C., (1998). Real Strategies for Virtual Organizing, *Sloan Management Review*, Fall 1998, Volume 40, Number 1, P33(16).

Vickrey, W., (1961). Counter-speculation, auctions, and competitive sealed tenders, *Journal of Finance*; 16(1): 9-37.

von Neumann, J. & Margenstern, O., (2004). The Theory of Games and Economic Behaviour. Princeton University Press.

Wilkenfeld, J. Kraus, S., Holley, K. & Harr, M. (1995). Genie: A decision support System for crisis negotiations. *A decision Support Systems*. 14(4): 369-391.

Wilkenfeld, J., Kraus, S., Holley, K., & Genie, M. H., (1995). A decision support System for crisis negotiations. *A decision Support Systems*, 14(4): 369-391.

Yeo, C. S., de Assuncao, M. D. & Yu, J. "Utility Computing and Global Grids", GRIDS Laboratory Department of Computer Science and Software Engineering, The University of Melbourne, Vic 3010<Australia..

Yeo, C.S., de Assuncao, V. & Yu, J., Utility Computing on Global Grids Grid Computing and Distributed Systems (GRIDS) Laboratory Department of Computer Science and Software Engineering The University of Melbourne, Melbourne, VIC 3010, Australia Email: {csyeo, raj, marcosd, jiayu, anthony, srikumar, mplac}@csse.unimelb.edu.au

Yonet A. Eracar, A., (2003), November. Negotiation-Based Approach to Finding Satisficing Solutions of Declaratively Specified Multi-Objective Optimization Problems, November 6.

Yu, J., Venugopal, S. & Buyya, R. "Grid Market Directory: A Web Services based Grid Service Publication Directory".

Zeng, D. & Sycara, K., (1998). Bayesian learning in negotiation. *International Journal of Human-Computer Studies*, 48(1):125-141.

Zeng, D., & Sycara, K., (1997, July). Benefits of learning in negotiation. In Proceedings of the 14th National Conference on Artificial Intelligence and 9th *Innovative Applications of Artificial Intelligence Conference* (AAAI-97), pages 36-42, Menlo Park, AAAI Press.

Zitzler, E., & Thiele, L., (1999, November). "MultiObjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach" *IEEE Transaction On Evolutionary Computation*, Vol. 3, No. 4.

Pareto, V. (1906). *Manual of Political Economy*, 1906, p. 106.

APPENDICES

APPENDIX A: Negotiation Behavior of Client-Agent(Price – Negotiation Issue)

Time	PriceOffered	UtilityOfPriceTo	ValueOfPriceTo	weightOfPriceTo
$1.0*e^{+003}$	ByClientAgent	ClientAgent	ClientAgent	MarketAgent
0	37.2000	0.1204	0.1400	0.8600
0.0002	37.1993	0.1204	0.1400	0.8599
0.0004	37.1987	0.1204	0.1401	0.8598
0.0006	37.1980	0.1204	0.1401	0.8596
0.0008	37.1974	0.1204	0.1401	0.8594
0.0010	37.1967	0.1204	0.1402	0.8591
0.0012	37.1960	0.1204	0.1402	0.8588
0.0014	37.1954	0.1204	0.1402	0.8585
0.0016	37.1947	0.1204	0.1403	0.8581
0.0018	37.1940	0.1203	0.1403	0.8577
0.0020	37.1934	0.1203	0.1403	0.8573
0.0022	37.1927	0.1203	0.1404	0.8568
0.0024	37.1921	0.1202	0.1404	0.8563
0.0026	37.1914	0.1202	0.1404	0.8558
0.0028	37.1907	0.1201	0.1405	0.8552
0.0030	37.1901	0.1201	0.1405	0.8546
0.0032	37.1894	0.1200	0.1405	0.8540
0.0034	37.1887	0.1200	0.1406	0.8534
0.0036	37.1881	0.1199	0.1406	0.8527
0.0038	37.1874	0.1198	0.1406	0.8520
0.0040	37.1867	0.1198	0.1407	0.8513
0.0042	37.1861	0.1197	0.1407	0.8506
0.0044	37.1854	0.1196	0.1407	0.8498
0.0046	37.1848	0.1195	0.1408	0.8491
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
0.9960	21.3817	-78.7250	0.9309	-84.5674
0.9962	21.3413	-78.9226	0.9329	-84.5960
0.9964	21.2999	-79.1245	0.9350	-84.6246
0.9966	21.2575	-79.3307	0.9371	-84.6532
0.9968	21.2140	-79.5418	0.9393	-84.6818
0.9970	21.1692	-79.7582	0.9415	-84.7104
0.9972	21.1232	-79.9802	0.9438	-84.7390
0.9974	21.0757	-80.2086	0.9462	-84.7677
0.9976	21.0265	-80.4440	0.9487	-84.7963
0.9978	20.9756	-80.6871	0.9512	-84.8249
0.9980	20.9227	-80.9390	0.9539	-84.8535
0.9982	20.8674	-81.2009	0.9566	-84.8822
0.9984	20.8094	-81.4744	0.9595	-84.9108
0.9986	20.7483	-81.7615	0.9626	-84.9394
0.9988	20.6833	-82.0651	0.9658	-84.9681
0.9990	20.6136	-82.3890	0.9693	-84.9967
0.9992	20.5378	-82.7392	0.9731	-85.0254
0.9994	20.4535	-83.1255	0.9773	-85.0540
0.9996	20.3564	-83.5664	0.9822	-85.0827
0.9998	20.2359	-84.1076	0.9882	-85.1113
1.0000	20.0000	-85.1400	1.0000	-85.1400

APPENDIX B: Negotiation Behavior of Marketing-Agent(Price – Negotiation Issue)

Time $1.0*e^{+003}$	PriceOfferedBy MarketAgent	UtilityOfPriceTo MarketAgent	valueOfPriceTo MarketAgent	weightOfPriceToMarketAgent
0	37.2000	0.0196	0.1400	0.1400
0.0002	37.1993	0.0196	0.1400	0.1401
0.0004	37.1987	0.0196	0.1401	0.1402
0.0006	37.1980	0.0197	0.1401	0.1404
0.0008	37.1974	0.0197	0.1401	0.1406
0.0010	37.1967	0.0197	0.1402	0.1409
0.0012	37.1960	0.0198	0.1402	0.1412
0.0014	37.1954	0.0198	0.1402	0.1415
0.0016	37.1947	0.0199	0.1403	0.1419
0.0018	37.1940	0.0200	0.1403	0.1423
0.0020	37.1934	0.0200	0.1403	0.1427
0.0022	37.1927	0.0201	0.1404	0.1432
0.0024	37.1921	0.0202	0.1404	0.1437
0.0026	37.1914	0.0203	0.1404	0.1442
0.0028	37.1907	0.0203	0.1405	0.1448
0.0030	37.1901	0.0204	0.1405	0.1454
0.0032	37.1894	0.0205	0.1405	0.1460
0.0034	37.1887	0.0206	0.1406	0.1466
0.0036	37.1881	0.0207	0.1406	0.1473
0.0038	37.1874	0.0208	0.1406	0.1480
0.0040	37.1867	0.0209	0.1407	0.1487
0.0042	37.1861	0.0210	0.1407	0.1494
0.0044	37.1854	0.0211	0.1407	0.1502
0.0046	37.1848	0.0212	0.1408	0.1509
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
0.9960	21.3817	79.6559	0.9309	85.5674
0.9962	21.3413	79.8556	0.9329	85.5960
0.9964	21.2999	80.0595	0.9350	85.6246
0.9966	21.2575	80.2679	0.9371	85.6532
0.9968	21.2140	80.4811	0.9393	85.6818
0.9970	21.1692	80.6997	0.9415	85.7104
0.9972	21.1232	80.9241	0.9438	85.7390
0.9974	21.0757	81.1548	0.9462	85.7677
0.9976	21.0265	81.3926	0.9487	85.7963
0.9978	20.9756	81.6383	0.9512	85.8249
0.9980	20.9227	81.8929	0.9539	85.8535
0.9982	20.8674	82.1575	0.9566	85.8822
0.9984	20.8094	82.4339	0.9595	85.9108
0.9986	20.7483	82.7241	0.9626	85.9394
0.9988	20.6833	83.0309	0.9658	85.9681
0.9990	20.6136	83.3583	0.9693	85.9967
0.9992	20.5378	83.7123	0.9731	86.0254
0.9994	20.4535	84.1028	0.9773	86.0540
0.9996	20.3564	84.5485	0.9822	86.0827
0.9998	20.2359	85.0958	0.9882	86.1113
1.0000	20.0000	86.1400	1.0000	86.1400

APPENDIX C: Negotiation Behavior of Marketing-Agent(Quantity – Negotiation Issue)

Time 1.0*e^{+003}	QuantityOfferedBy MarketAgent	UtilityOfQuantityTo MarketAgent	valueOfQuantityTo MarketAgent	weightOfQuantityTo MarketAgent
0	87.4000	0.0196	0.1400	0.1400
0.0002	87.3970	0.0196	0.1400	0.1401
0.0004	87.3941	0.0196	0.1401	0.1402
0.0006	87.3911	0.0197	0.1401	0.1404
0.0008	87.3881	0.0197	0.1401	0.1406
0.0010	87.3851	0.0197	0.1402	0.1409
0.0012	87.3821	0.0198	0.1402	0.1412
0.0014	87.3792	0.0198	0.1402	0.1415
0.0016	87.3762	0.0199	0.1403	0.1419
0.0018	87.3732	0.0200	0.1403	0.1423
0.0020	87.3702	0.0200	0.1403	0.1427
0.0022	87.3672	0.0201	0.1404	0.1432
0.0024	87.3643	0.0202	0.1404	0.1437
0.0026	87.3613	0.0203	0.1404	0.1442
0.0028	87.3583	0.0203	0.1405	0.1448
0.0030	87.3553	0.0204	0.1405	0.1454
0.0032	87.3523	0.0205	0.1405	0.1460
0.0034	87.3493	0.0206	0.1406	0.1466
0.0036	87.3463	0.0207	0.1406	0.1473
0.0038	87.3433	0.0208	0.1406	0.1480
0.0040	87.3404	0.0209	0.1407	0.1487
0.0042	87.3374	0.0210	0.1407	0.1494
0.0044	87.3344	0.0211	0.1407	0.1502
0.0046	87.3314	0.0212	0.1408	0.1509
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
0.9960	16.2178	79.6559	0.9309	85.5674
0.9962	16.0358	79.8556	0.9329	85.5960
0.9964	15.8495	80.0595	0.9350	85.6246
0.9966	15.6587	80.2679	0.9371	85.6532
0.9968	15.4628	80.4811	0.9393	85.6818
0.9970	15.2615	80.6997	0.9415	85.7104
0.9972	15.0542	80.9241	0.9438	85.7390
0.9974	14.8404	81.1548	0.9462	85.7677
0.9976	14.6194	81.3926	0.9487	85.7963
0.9978	14.3903	81.6383	0.9512	85.8249
0.9980	14.1520	81.8929	0.9539	85.8535
0.9982	13.9032	82.1575	0.9566	85.8822
0.9984	13.6424	82.4339	0.9595	85.9108
0.9986	13.3672	82.7241	0.9626	85.9394
0.9988	13.0749	83.0309	0.9658	85.9681
0.9990	12.7612	83.3583	0.9693	85.9967
0.9992	12.4199	83.7123	0.9731	86.0254
0.9994	12.0407	84.1028	0.9773	86.0540
0.9996	11.6039	84.5485	0.9822	86.0827
0.9998	11.0614	85.0958	0.9882	86.1113
1.0000	10.0000	86.1400	1.0000	86.1400

APPENDIX D: Negotiation Behavior of Client-Agent(Quantity – Negotiation Issue)

Time	QuantityOfferedBy	UtilityOfQuantityTo	ValueOfQuantityTo	weightOfQuantityTo
1.0*e^{+003}	ClientAgent	ClientAgent	ClientAgent	ClientAgent
0	87.4000	0.1204	0.1400	0.8600
0.0002	87.3970	0.1204	0.1400	0.8599
0.0004	87.3941	0.1204	0.1401	0.8598
0.0006	87.3911	0.1204	0.1401	0.8596
0.0008	87.3881	0.1204	0.1401	0.8594
0.0010	87.3851	0.1204	0.1402	0.8591
0.0012	87.3821	0.1204	0.1402	0.8588
0.0014	87.3792	0.1204	0.1402	0.8585
0.0016	87.3762	0.1204	0.1403	0.8581
0.0018	87.3732	0.1203	0.1403	0.8577
0.0020	87.3702	0.1203	0.1403	0.8573
0.0022	87.3672	0.1203	0.1404	0.8568
0.0024	87.3643	0.1202	0.1404	0.8563
0.0026	87.3613	0.1202	0.1404	0.8558
0.0028	87.3583	0.1201	0.1405	0.8552
0.0030	87.3553	0.1201	0.1405	0.8546
0.0032	87.3523	0.1200	0.1405	0.8540
0.0034	87.3493	0.1200	0.1406	0.8534
0.0036	87.3463	0.1199	0.1406	0.8527
0.0038	87.3433	0.1198	0.1406	0.8520
0.0040	87.3404	0.1198	0.1407	0.8513
0.0042	87.3374	0.1197	0.1407	0.8506
0.0044	87.3344	0.1196	0.1407	0.8498
0.0046	87.3314	0.1195	0.1408	0.8491
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
0.9960	16.2178	-78.7250	0.9309	-84.5674
0.9962	16.0358	-78.9226	0.9329	-84.5960
0.9964	15.8495	-79.1245	0.9350	-84.6246
0.9966	15.6587	-79.3307	0.9371	-84.6532
0.9968	15.4628	-79.5418	0.9393	-84.6818
0.9970	15.2615	-79.7582	0.9415	-84.7104
0.9972	15.0542	-79.9802	0.9438	-84.7390
0.9974	14.8404	-80.2086	0.9462	-84.7677
0.9976	14.6194	-80.4440	0.9487	-84.7963
0.9978	14.3903	-80.6871	0.9512	-84.8249
0.9980	14.1520	-80.9390	0.9539	-84.8535
0.9982	13.9032	-81.2009	0.9566	-84.8822
0.9984	13.6424	-81.4744	0.9595	-84.9108
0.9986	13.3672	-81.7615	0.9626	-84.9394
0.9988	13.0749	-82.0651	0.9658	-84.9681
0.9990	12.7612	-82.3890	0.9693	-84.9967
0.9992	12.4199	-82.7392	0.9731	-85.0254
0.9994	12.0407	-83.1255	0.9773	-85.0540
0.9996	11.6039	-83.5664	0.9822	-85.0827
0.9998	11.0614	-84.1076	0.9882	-85.1113
1.0000	10.0000	-85.1400	1.0000	-85.1400

APPENDIX E: Negotiation Behavior of Marketing-Agent(Delivery-Date – Negotiation Issue)

Time $1.0*e^{+003}$	DeliveryDateOffered ByMarketAgent	UtilityOfDelivery DateToMarketAgent	valueOfDeliveryDateTo MarketAgent	weightOfDeliveryDateTo MarketAgent
0	12.4600	0.1204	0.1400	0.8600
0.0002	12.4596	0.1204	0.1400	0.8599
0.0004	12.4593	0.1204	0.1401	0.8598
0.0006	12.4589	0.1204	0.1401	0.8596
0.0008	12.4585	0.1204	0.1401	0.8594
0.0010	12.4582	0.1204	0.1402	0.8591
0.0012	12.4578	0.1204	0.1402	0.8588
0.0014	12.4575	0.1204	0.1402	0.8585
0.0016	12.4571	0.1204	0.1403	0.8581
0.0018	12.4567	0.1203	0.1403	0.8577
0.0020	12.4564	0.1203	0.1403	0.8573
0.0022	12.4560	0.1203	0.1404	0.8568
0.0024	12.4556	0.1202	0.1404	0.8563
0.0026	12.4553	0.1202	0.1404	0.8558
0.0028	12.4549	0.1201	0.1405	0.8552
0.0030	12.4545	0.1201	0.1405	0.8546
0.0032	12.4542	0.1200	0.1405	0.8540
0.0034	12.4538	0.1200	0.1406	0.8534
0.0036	12.4534	0.1199	0.1406	0.8527
0.0038	12.4531	0.1198	0.1406	0.8520
0.0040	12.4527	0.1198	0.1407	0.8513
0.0042	12.4523	0.1197	0.1407	0.8506
0.0044	12.4520	0.1196	0.1407	0.8498
0.0046	12.4516	0.1195	0.1408	0.8491
. . .	. . .	. . .	. .	. .
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
0.9960	3.7599	-78.7250	0.9309	-84.5674
0.9962	3.7377	-78.9226	0.9329	-84.5960
0.9964	3.7149	-79.1245	0.9350	-84.6246
0.9966	3.6916	-79.3307	0.9371	-84.6532
0.9968	3.6677	-79.5418	0.9393	-84.6818
0.9970	3.6431	-79.7582	0.9415	-84.7104
0.9972	3.6177	-79.9802	0.9438	-84.7390
0.9974	3.5916	-80.2086	0.9462	-84.7677
0.9976	3.5646	-80.4440	0.9487	-84.7963
0.9978	3.5366	-80.6871	0.9512	-84.8249
0.9980	3.5075	-80.9390	0.9539	-84.8535
0.9982	3.4771	-81.2009	0.9566	-84.8822
0.9984	3.4452	-81.4744	0.9595	-84.9108
0.9986	3.4116	-81.7615	0.9626	-84.9394
0.9988	3.3758	-82.0651	0.9658	-84.9681
0.9990	3.3375	-82.3890	0.9693	-84.9967
0.9992	3.2958	-82.7392	0.9731	-85.0254
0.9994	3.2494	-83.1255	0.9773	-85.0540
0.9996	3.1960	-83.5664	0.9822	-85.0827
0.9998	3.1297	-84.1076	0.9882	-85.1113
1.0000	3.0000	-85.1400	1.0000	-85.1400

APPENDIX F: Negotiation Behavior of Client-Agent(Delivery-Date – Negotiation Issue)

Time $1.0*e^{+003}$	DeliveryDateOffered ByClientAgent	UtilityOfDeliveryDate ToClientAgent	ValueOfDeliveryDate ToClientAgent	weightOfDeliveryDate ToClientAgent
0	4.5400	0.0196	0.1400	0.1400
0.0002	4.5404	0.0196	0.1400	0.1401
0.0004	4.5407	0.0196	0.1401	0.1402
0.0006	4.5411	0.0197	0.1401	0.1404
0.0008	4.5415	0.0197	0.1401	0.1406
0.0010	4.5418	0.0197	0.1402	0.1409
0.0012	4.5422	0.0198	0.1402	0.1412
0.0014	4.5425	0.0198	0.1402	0.1415
0.0016	4.5429	0.0199	0.1403	0.1419
0.0018	4.5433	0.0200	0.1403	0.1423
0.0020	4.5436	0.0200	0.1403	0.1427
0.0022	4.5440	0.0201	0.1404	0.1432
0.0024	4.5444	0.0202	0.1404	0.1437
0.0026	4.5447	0.0203	0.1404	0.1442
0.0028	4.5451	0.0203	0.1405	0.1448
0.0030	4.5455	0.0204	0.1405	0.1454
0.0032	4.5458	0.0205	0.1405	0.1460
0.0034	4.5462	0.0206	0.1406	0.1466
0.0036	4.5466	0.0207	0.1406	0.1473
0.0038	4.5469	0.0208	0.1406	0.1480
0.0040	4.5473	0.0209	0.1407	0.1487
0.0042	4.5477	0.0210	0.1407	0.1494
0.0044	4.5480	0.0211	0.1407	0.1502
0.0046	4.5484	0.0212	0.1408	0.1509
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
0.9960	13.2401	79.6559	0.9309	85.5674
0.9962	13.2623	79.8556	0.9329	85.5960
0.9964	13.2851	80.0595	0.9350	85.6246
0.9966	13.3084	80.2679	0.9371	85.6532
0.9968	13.3323	80.4811	0.9393	85.6818
0.9970	13.3569	80.6997	0.9415	85.7104
0.9972	13.3823	80.9241	0.9438	85.7390
0.9974	13.4084	81.1548	0.9462	85.7677
0.9976	13.4354	81.3926	0.9487	85.7963
0.9978	13.4634	81.6383	0.9512	85.8249
0.9980	13.4925	81.8929	0.9539	85.8535
0.9982	13.5229	82.1575	0.9566	85.8822
0.9984	13.5548	82.4339	0.9595	85.9108
0.9986	13.5884	82.7241	0.9626	85.9394
0.9988	13.6242	83.0309	0.9658	85.9681
0.9990	13.6625	83.3583	0.9693	85.9967
0.9992	13.7042	83.7123	0.9731	86.0254
0.9994	13.7506	84.1028	0.9773	86.0540
0.9996	13.8040	84.5485	0.9822	86.0827
0.9998	13.8703	85.0958	0.9882	86.1113
1.0000	14.0000	86.1400	1.0000	86.1400

APPENDIX G: Program Documentation

All Classes

Agent

AgentFactory

ChairMan_Agent

ChairManProxy_Agent

Client_Agent

ClientProxy_Agent

Information_Agent

Marketing_Agent

Test

Package Class Use Tree Deprecated Index Help

PREV PACKAGE   NEXT PACKAGE   FRAMES   NO FRAMES   All Classes

---

Package AgentCommunicationMechanism

Interface Summary

Agent	

Class Summary

AgentFactory	
ChairMan_Agent	
ChairManProxy_Agent	
Client_Agent	
ClientProxy_Agent	
Information_Agent	
Marketing_Agent	
Test	

Package Class Use Tree Deprecated Index Help

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES All Classes

Package [redacted] Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

AgentCommunicationMechanism
Interface Agent
All Known Implementing Classes:

AgentFactory, ChairMan_Agent, ChairManProxy_Agent, Client_Agent, ClientProxy_Agent, Information_Agent, Marketing_Agent

---

```
public interface Agent
```

Package [redacted] Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package [redacted] Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

AgentCommunicationMechanism

Class AgentFactory

```
java.lang.Object
  └─AgentCommunicationMechanism.AgentFactory
```

All Implemented Interfaces:

Agent

Direct Known Subclasses:

ChairMan_Agent, Client_Agent

```
public abstract class AgentFactory
extends java.lang.Object
implements Agent
```

Constructor Summary

`AgentFactory()`

Method Summary

`void`	`transform2XML(org.w3c.dom.Document doc)`
`void`	`transform2XML(org.w3c.dom.Element element)`
`void`	`transform2XML(java.sql.ResultSet resultSet,   java.io.File file)`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

AgentFactory

`public AgentFactory()`

Method Detail

transform2XML

`public void transform2XML(org.w3c.dom.Document doc)`

transform2XML

`public void transform2XML(org.w3c.dom.Element element)`

transform2XML

```
public void transform2XML(java.sql.ResultSet resultSet,
                          java.io.File file)
```

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS   NEXT CLASS   FRAMES   NO FRAMES   All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD   DETAIL: FIELD | CONSTR | METHOD

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS   NEXT CLASS   FRAMES   NO FRAMES   All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD   DETAIL: FIELD | CONSTR | METHOD

AgentCommunicationMechanism

Class ChairMan Agent

```
java.lang.Object
  └ AgentCommunicationMechanism.AgentFactory
      └ AgentCommunicationMechanism.ChairMan_Agent
```

All Implemented Interfaces:

Agent

Direct Known Subclasses:

ChairManProxy_Agent, Information_Agent, Marketing_Agent

```
public abstract class ChairMan_Agent
extends AgentFactory
```

Constructor Summary

```
ChairMan Agent()
```

Method Summary

`protected static ChairMan_Agent`	`newInstance()`
`protected abstract void`	`parse(org.w3c.dom.Document doc)`
`protected abstract void`	`parse(java.io.File file)`

Methods inherited from class AgentCommunicationMechanism.AgentFactory

`transform2XML, transform2XML, transform2XML`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ChairMan_Agent

`public ChairMan_Agent()`

Method Detail

newInstance

`protected static ChairMan_Agent newInstance()`

parse

`protected abstract void parse(java.io.File file)`

parse

`protected abstract void parse(org.w3c.dom.Document doc)`

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES *All Classes*

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

AgentCommunicationMechanism
Class Client_Agent

```
java.lang.Object
  └AgentCommunicationMechanism.AgentFactory
      └AgentCommunicationMechanism.Client_Agent
```

All Implemented Interfaces:

Agent

Direct Known Subclasses:

ClientProxy_Agent

```
public abstract class Client_Agent
extends AgentFactory
```

Field Summary

```
static java.lang.String test
```

Constructor Summary

Client Agent()

Method Summary

abstract boolean	authenticate(java.io.File fil java.lang.String username, java.lang.String password)
abstract java.util.Hashtable<java.lang.String,java.lang.String>	find(java.io.File file, java.lang.String productid)
abstract java.util.Hashtable<java.lang.String,java.lang.String>	negotiate(java.io.File file, int *quantity*, double *pric*
static Client Agent	newInstance()
abstract java.util.Hashtable<java.lang.String,java.lang.String>	purchase(java.io.File item, java.lang.String quantity, java.lang.String bankname, java.lang.String accountno, java.lang.String card, java.lang.String cardno)
abstract java.util.Hashtable<java.lang.String,java.lang.String>	search(java.io.File file, java.lang.String category)

Methods inherited from class AgentCommunicationMechanism.AgentFactory

transform2XML, transform2XML, transform2XML

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

test

```
public static java.lang.String test
```

Constructor Detail

Client_Agent

```
public Client_Agent()
```

Method Detail

newInstance

```
public static Client Agent newInstance()
```

---

authenticate

```
public abstract boolean authenticate(java.io.File file,
                                     java.lang.String username,
                                     java.lang.String password)
```

---

search

```
public abstract java.util.Hashtable<java.lang.String,java.lang.String>
search(java.io.File file,

java.lang.String category)
```

---

find

```
public abstract java.util.Hashtable<java.lang.String,java.lang.String>
find(java.io.File file,

java.lang.String productid)
```

---

negotiate

```
public abstract java.util.Hashtable<java.lang.String,java.lang.String>
negotiate(java.io.File file,

int quantity,

double price)
```

---

purchase

```
public abstract java.util.Hashtable<java.lang.String,java.lang.String>
purchase(java.io.File item,

java.lang.String quantity,

java.lang.String bankname,

java.lang.String accountno,

java.lang.String card,

java.lang.String cardno)
```

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS *NEXT CLASS* *FRAMES* *NO FRAMES* *All Classes*

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS *NEXT CLASS* *FRAMES* *NO FRAMES* *All Classes*

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

AgentCommunicationMechanism

Class ClientProxy_Agent

```
java.lang.Object
  └AgentCommunicationMechanism.AgentFactory
      └AgentCommunicationMechanism.Client Agent
  └AgentCommunicationMechanism.ClientProxy_Agent
```

All Implemented Interfaces:

Agent

```
public class ClientProxy_Agent
extends Client_Agent
```

Field Summary

Fields inherited from class AgentCommunicationMechanism.Client_Agent
test

Constructor Summary

ClientProxy_Agent()

Method Summary

boolean	authenticate(java.io.F java.lang.String usern java.lang.String passw
java.util.Hashtable<java.lang.String,java.lang.String>	find(java.io.File file java.lang.String produ
java.util.Hashtable<java.lang.String,java.lang.String>	negotiate(java.io.File int Quantity, douk

void	parse(java.io.File file
java.util.Hashtable<java.lang.String,java.lang.String>	purchase(java.io.File i java.lang.String Quanti java.lang.String bankna java.lang.String accoun java.lang.String card, java.lang.String cardno
java.util.Hashtable<java.lang.String,java.lang.String>	search(java.io.File fil java.lang.String carteg

Methods inherited from class AgentCommunicationMechanism.Client_Agent
newInstance

Methods inherited from class AgentCommunicationMechanism.AgentFactory
transform2XML, transform2XML, transform2XML

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ClientProxy_Agent

```
public ClientProxy_Agent()
```

Method Detail

parse

```
public void parse(java.io.File file)
```

---

authenticate

```
public boolean authenticate(java.io.File file,
                            java.lang.String username,
                            java.lang.String password)
```

Specified by:

`authenticate` in class `Client Agent`

---

search

```
public          java.util.Hashtable<java.lang.String,java.lang.String>
search(java.io.File file,

java.lang.String cartegory)
```

Specified by:

`search` in class `Client Agent`

---

find

```
public          java.util.Hashtable<java.lang.String,java.lang.String>
find(java.io.File file,

java.lang.String productid)
```

Specified by:

`find` in class `Client Agent`

---

negotiate

```
public          java.util.Hashtable<java.lang.String,java.lang.String>
negotiate(java.io.File file,

int Quantity,

double Price)
```

Specified by:

`negotiate` in class `Client Agent`

---

purchase

```
public          java.util.Hashtable<java.lang.String,java.lang.String>
purchase(java.io.File item,

java.lang.String Quantity,

java.lang.String bankname,

java.lang.String accountno,

java.lang.String card,

java.lang.String cardno)
```

Specified by:

`purchase` in class `Client_Agent`

Package **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

AgentCommunicationMechanism

Class Information_Agent

```
java.lang.Object
  └─AgentCommunicationMechanism.AgentFactory
        └─AgentCommunicationMechanism.ChairMan_Agent
              └─AgentCommunicationMechanism.Information_Agent
```

All Implemented Interfaces:

Agent

```
public class Information_Agent
```

extends ChairMan Agent

Constructor Summary
Information Agent()

Method Summary	
void	parse(org.w3c.dom.Document doc)
void	parse(java.io.File file)

Methods inherited from class AgentCommunicationMechanism.ChairMan_Agent
newInstance

Methods inherited from class AgentCommunicationMechanism.AgentFactory
transform2XML, transform2XML, transform2XML

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Information_Agent

```
public Information_Agent()
```

Method Detail

parse

```
public void parse(java.io.File file)
```

Specified by:

parse in class ChairMan_Agent

---

parse

```
public void parse(org.w3c.dom.Document doc)
```

Specified by:

parse in class ChairMan_Agent

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

AgentCommunicationMechanism

Class Marketing_Agent

```
java.lang.Object
  └─AgentCommunicationMechanism.AgentFactory
      └─AgentCommunicationMechanism.ChairMan_Agent
          └─AgentCommunicationMechanism.Marketing_Agent
```

All Implemented Interfaces:

Agent

```
public class Marketing_Agent
extends ChairMan_Agent
```

Constructor Summary
Marketing_Agent()

Method Summary	
void	parse(org.w3c.dom.Document doc)
void	parse(java.io.File file)

Methods inherited from class AgentCommunicationMechanism.ChairMan_Agent
newInstance

Methods inherited from class AgentCommunicationMechanism.AgentFactory
transform2XML, transform2XML, transform2XML

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Marketing_Agent

```
public Marketing_Agent()
```

Method Detail

parse

```
public void parse(java.io.File file)
```

Specified by:

`parse` in class `ChairMan_Agent`

---

parse

```
public void parse(org.w3c.dom.Document doc)
```

Specified by:

`parse` in class `ChairMan_Agent`

Package Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

AgentCommunicationMechanism

Class Test

```
java.lang.Object
  └AgentCommunicationMechanism.Test
```

```
public class Test
extends java.lang.Object
```

Constructor Summary

`Test()` Creates a new instance of Test

Method Summary

`static void`	`main(java.lang.String[] args)`
`java.util.Hashtable<java.lang.String,java.lang.String>`	`purchase(java.io.File item, java.lang.String bankname, java.lang.String accountno, java.lang.String card, java.lang.String cardno)`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Test

```
public Test()
```

Creates a new instance of Test

Method Detail

purchase

```
public           java.util.Hashtable<java.lang.String,java.lang.String>
purchase(java.io.File item,


java.lang.String bankname,


java.lang.String accountno,


java.lang.String card,


java.lang.String cardno)
```

---

main

```
public static void main(java.lang.String[] args)
```

Package [redacted] **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS   NEXT CLASS   FRAMES   NO FRAMES   All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD   DETAIL: FIELD | CONSTR | METHOD

```
package AgentCommunicationMechanism;


public interface Agent
{}
package AgentCommunicationMechanism;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintStream;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.Vector;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;

public abstract class AgentFactory implements Agent
{

  private  Vector<String>tag=new Vector<String>();

public AgentFactory(){}

  public void transform2XML(Document doc)
  {
    tag.removeAllElements();
    try
    {
      File file=new File(doc.getDocumentURI().substring(6));
      BufferedWriter bfr = new BufferedWriter(new FileWriter(file));
      Element
agentcom=((Element)doc.getElementsByTagName("agentcom").item(0));
      transform2XML(agentcom);
      bfr.flush();
      bfr.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
      bfr.write("<agentcom>\n");
      for(int i=0;i<tag.size();i++)
      {
        bfr.flush();
        bfr.write(tag.elementAt(i)+"\n");
      }
      bfr.write("</agentcom>\n");
    bfr.close();
  }
  catch (IOException ex)
  {
    ex.printStackTrace();
  }

}

public void transform2XML(Element element)
{
  String nodename="",nodeattr="",nodetext="";
  Node node=null;
  boolean endtag=false;
  for(int i=0;i<element.getChildNodes().getLength();i++)
  {
    node=element.getChildNodes().item(i);

    if(node.getNodeName().equals("#text"))
    {
      node=null;
    }
    else
    {
      if(node.hasAttributes())
      {
        nodeattr="";
        for(int j=0;j<node.getAttributes().getLength();j++)
          nodeattr+=node.getAttributes().item(j).toString()+" ";
      }
      nodename=node.getNodeName();
      nodetext=node.getTextContent();
    }
    if(node!=null)
    {
      String out="<"+nodename+" "+nodeattr;
      Element element2=(Element)node;
      if(element2.getChildNodes().getLength()==0)
        out+="/>";
      else if(element2.getChildNodes().getLength()==1)
        out+=">"+nodetext+"</"+nodename+">";
      else
      {
        out+=">";
        endtag=true;
      }
      tag.addElement(out);
      transform2XML(element2);
      if(endtag)
      { tag.addElement("</"+nodename+">");}
    }
  }
}
public void transform2XML(ResultSet resultSet,File file)
{
  try
  {

    FileOutputStream out=new FileOutputStream(file);
    PrintStream p=new PrintStream(out);
    p.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
    ResultSetMetaData   metaData = resultSet.getMetaData();
    p.println("<agentcom>\n");
    int a=1;
    int b=1;
    while(!resultSet.isLast())
    {
      p.println("<product>");
```

```
          for(int i=1;i<=metaData.getColumnCount();i++)
          {
            p.print("<" + metaData.getColumnName(i) + ">");
            resultSet.absolute(a);
            p.print(resultSet.getObject(b));
            p.println("</" + metaData.getColumnName(i) + ">");
            b++;
          }
          p.println("</product>");
          a++;
          b=1;
        }
        p.println("</agentcom>");

        p.close();
        out.close();


      }

      catch (SQLException ex)
      {System.out.println("error !");ex.printStackTrace();}
      catch (FileNotFoundException ex)
      {System.out.println("error !");ex.printStackTrace();}
      catch (IOException ex)
      {ex.printStackTrace();}
   }
}
package AgentCommunicationMechanism;

import java.io.File;
import org.w3c.dom.Document;
public abstract class ChairMan_Agent extends AgentFactory
```

```
{

public ChairMan_Agent(){}

   protected static ChairMan_Agent newInstance()
   {
      return new ChairManProxy_Agent();
   }
   protected abstract void parse(File file);
   protected abstract void parse(Document doc);
}

package AgentCommunicationMechanism;
import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.Document;
import org.w3c.dom.DocumentType;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class ChairManProxy_Agent extends ChairMan_Agent
{

   public ChairManProxy_Agent(){}
   public void parse(File xmlfile)
   {
      try
      {
         DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory
               .newInstance();
         DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
         Document doc = docBuilder.parse(xmlfile);
      doc.getDocumentElement().normalize();

      if (doc.getElementsByTagName("request").item(0).getAttributes()
      .getNamedItem("type").getNodeValue().equals("transaction"))
      {
        new Marketing_Agent().parse(doc);
      }
      else
         new Information_Agent().parse(doc);
    }
    catch (SAXParseException err)
    {
      System.out.println("** Parsing error" + ", line "
           + err.getLineNumber() + ", uri " + err.getSystemId());
      System.out.println(" " + err.getMessage());
    }
    catch (SAXException e)
    {
      Exception x = e.getException();
      ((x == null) ? e : x).printStackTrace();
    }
    catch (Throwable t)
    {
      t.printStackTrace();
    }

  }

  public void parse(Document doc)
  {}

}
package AgentCommunicationMechanism;
import java.io.File;
import java.util.Hashtable;
public abstract  class Client_Agent extends AgentFactory
{
   public Client_Agent(){}
public static String test="Testing...";
   public static Client_Agent newInstance()
   {
      return new ClientProxy_Agent();
   }
   public abstract boolean authenticate(File file,String username,String password);
   public abstract Hashtable<String,String>search(File file,String category);
   public abstract Hashtable<String,String>find(File file,String productid);
   public abstract Hashtable<String,String>negotiate(File file,int quantity,double price);
}
package AgentCommunicationMechanism;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Hashtable;
import java.util.StringTokenizer;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.DOMException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
public class ClientProxy_Agent extends Client_Agent
{
   private ChairMan_Agent agent;
   public void parse(File file)
   {}
```

```
public ClientProxy_Agent()
{
  agent= ChairMan_Agent.newInstance();
}

public boolean authenticate(File file,String username,String password)
{

  try
  {

    BufferedWriter bfr = new BufferedWriter(new FileWriter(file));
    bfr.write("<agentcom>");
    bfr.write("<request type=\"authenticate\">");
    bfr.write("<username>");
    bfr.write(username);
    bfr.write("</username>");
    bfr.write("<password>");
    bfr.write(password);
    bfr.write("</password>");
    bfr.write("<accepted>");
    bfr.write("false");
    bfr.write("</accepted>");
    bfr.write("<clientname>");
    bfr.write("null");
    bfr.write("</clientname>");
    bfr.write("</request>");
    bfr.write("</agentcom>");
    bfr.close();

    agent.parse(file);

    DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory
        .newInstance();
        DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(file);
        doc.getDocumentElement().normalize();
        if (doc.getElementsByTagName("accepted").item(0).getFirstChild()
        .getNodeValue().equals("true"))
            return true;
        else
            return false;
    }
    catch (DOMException e)
    {
        e.printStackTrace();
        return false;
    }
    catch (IOException e)
    {
        e.printStackTrace();
        return false;
    }
    catch (ParserConfigurationException e)
    {
        e.printStackTrace();
        return false;
    }
    catch (SAXException e)
    {
        e.printStackTrace();
        return false;
    }
}

public Hashtable<String,String> search(File file,String cartegory)
{
    try
```

```
  {
    BufferedWriter bfr = new BufferedWriter(new FileWriter(file));
    bfr.write("<agentcom>\n");
    bfr.write("<request type=\"search\">\n");
    bfr.write("<productcartegory name=\""+cartegory+"\"/>\n");
    bfr.write("</request>\n");
    bfr.write("</agentcom>");
    bfr.close();
    agent.parse(file);

    Hashtable<String,String> h=new Hashtable<String,String>();
    DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory
        .newInstance();
    DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
    Document doc = docBuilder.parse(file);

    doc.getDocumentElement().normalize();

    NodeList products = doc.getElementsByTagName("product");
    String seperator="\t\t";
    for(int i=0;i<products.getLength();i++)
    {
      Element product=(Element)products.item(i);
      String
product_id=product.getElementsByTagName("product_id").item(0).getFirstChild().get
NodeValue();
      String
product_name=product.getElementsByTagName("product_name").item(0).getFirstChil
d().getNodeValue();
      String
product_quantity_in_stock=product.getElementsByTagName("product_quantity_in_sto
ck").item(0).getFirstChild().getNodeValue();
```

```
        String                                          product_price="R
"+product.getElementsByTagName("product_price").item(0).getFirstChild().getNodeV
alue();
        System.out.println(product_id+"\t"+product_name);

h.put(product_id,seperator+product_name+seperator+product_quantity_in_stock+seper
ator+product_price);
      }
      return h;

    }
    catch (DOMException e)
    {
      e.printStackTrace();
      return null;
    }
    catch (IOException e)
    {
      e.printStackTrace();
      return null;
    }
    catch (ParserConfigurationException e)
    {
      e.printStackTrace();
      return null;
    }
    catch (SAXException e)
    {
      e.printStackTrace();
      return null;
    }
  }
  public Hashtable<String,String>find(File file,String productid)
  {
    try
    {
      Hashtable <String,String> product=new Hashtable();
      BufferedWriter bfr = new BufferedWriter(new FileWriter(file));
      bfr.write("<agentcom>\n");
      bfr.write("\t<request type=\"find\">\n");
      bfr.write("\t\t<product>\n");
      bfr.write("\t\t\t<product_id>"+productid+"</product_id>\n");
      bfr.write("\t\t\t<product_name> </product_name>\n");
      bfr.write("\t\t\t<product_price> </product_price>\n");
      bfr.write("\t\t\t<product_quantity_on_stock>
</product_quantity_on_stock>\n");
      bfr.write("\t\t\t<catalog_id> </catalog_id>\n");
      bfr.write("\t\t\t<provider_id> </provider_id>\n");
      bfr.write("\t\t\t<image> </image>\n");
      bfr.write("\t\t\t<description> </description>\n");
      bfr.write("\t\t\t<message> </message>\n");
      bfr.write("\t\t\t<negotiation     reservation_price=\"\"     negotiation_price=\"\"
buyer_price=\"\" quantity=\"1\" status=\"\" count=\"\" />\n");
      bfr.write("\t\t</product>\n");
      bfr.write("\t</request>\n");
      bfr.write("</agentcom>");
      bfr.close();
      agent.parse(file);
      DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory
          .newInstance();
      DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
      Document doc = docBuilder.parse(file);

      doc.getDocumentElement().normalize();
      String                                                   product_name=
doc.getElementsByTagName("product_name").item(0).getFirstChild().getNodeValue()
;
```

```
        String
product_price=doc.getElementsByTagName("product_price").item(0).getFirstChild().g
etNodeValue();
        String
provider=doc.getElementsByTagName("provider_id").item(0).getFirstChild().getNode
Value();
        String
image=doc.getElementsByTagName("image").item(0).getFirstChild().getNodeValue();
        String
description=doc.getElementsByTagName("description").item(0).getFirstChild().getNo
deValue();
        String
negotiation_price=doc.getElementsByTagName("negotiation").item(0).getAttributes().
getNamedItem("negotiation_price").getNodeValue();
        String
buyer_price=doc.getElementsByTagName("negotiation").item(0).getAttributes().getNa
medItem("buyer_price").getNodeValue();
        String
quantity=doc.getElementsByTagName("negotiation").item(0).getAttributes().getName
dItem("quantity").getNodeValue();
        String
message=doc.getElementsByTagName("message").item(0).getFirstChild().getNodeVal
ue();
        product.put("product_name",product_name);
        product.put("product_price",product_price);
        product.put("provider_id",provider);
        product.put("image",image);
        product.put("description",description);
        product.put("negotiation_price",negotiation_price);
        product.put("buyer_price",buyer_price);
        product.put("quantity",quantity);
        product.put("message",message);
        return product;
```

```
    }
    catch (DOMException e)
    {

        e.printStackTrace();
        return null;
    }
    catch (IOException e)
    {
        e.printStackTrace();
        return null;
    }
    catch (ParserConfigurationException e)
    {
        e.printStackTrace();
        return null;
    }
    catch (SAXException e)
    {
        e.printStackTrace();
        return null;
    }
}

public Hashtable<String,String> negotiate(File file,int Quantity, double Price)
{
    try
    {
        Hashtable <String,String> product=new Hashtable();

        DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory
                .newInstance();
        DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(file);
        doc.getDocumentElement().normalize();
        String
count=doc.getElementsByTagName("negotiation").item(0).getAttributes().getNamedIt
em("count").getNodeValue();
        String
status=doc.getElementsByTagName("negotiation").item(0).getAttributes().getNamedIt
em("status").getNodeValue();
        if(count.equals(""))

doc.getElementsByTagName("negotiation").item(0).getAttributes().getNamedItem("co
unt").setNodeValue("0");
        if(status.equals(""))

doc.getElementsByTagName("negotiation").item(0).getAttributes().getNamedItem("sta
tus").setNodeValue("NEGOTIATING");
        int Count=Integer.parseInt(count);

doc.getElementsByTagName("negotiation").item(0).getAttributes().getNamedItem("co
unt").setNodeValue(((Integer)(Count+1)).toString());

doc.getElementsByTagName("negotiation").item(0).getAttributes().getNamedItem("qu
antity").setNodeValue(((Integer)Quantity).toString());

doc.getElementsByTagName("negotiation").item(0).getAttributes().getNamedItem("bu
yer_price").setNodeValue(((Double)Price).toString());

doc.getElementsByTagName("request").item(0).getAttributes().getNamedItem("type").
setNodeValue("transaction");
        transform2XML(doc);

        agent.parse(file);

        DocumentBuilderFactory docBuilderFactory2 = DocumentBuilderFactory
            .newInstance();
        DocumentBuilder docBuilder2 = docBuilderFactory2.newDocumentBuilder();
        Document doc2= docBuilder2.parse(file);
        doc2.getDocumentElement().normalize();

        String                                          product_name=
doc2.getElementsByTagName("product_name").item(0).getFirstChild().getNodeValue(
);
        String
product_price=doc2.getElementsByTagName("product_price").item(0).getFirstChild().
getNodeValue();
        String
provider=doc2.getElementsByTagName("provider_id").item(0).getFirstChild().getNod
eValue();
        String
image=doc2.getElementsByTagName("image").item(0).getFirstChild().getNodeValue(
);
        String
description=doc2.getElementsByTagName("description").item(0).getFirstChild().getN
odeValue();
        String
negotiation_price=doc2.getElementsByTagName("negotiation").item(0).getAttributes()
.getNamedItem("negotiation_price").getNodeValue();
        StringTokenizer st=new StringTokenizer(negotiation_price,".");
        String delimeter=".";
        String absolute=st.nextToken();
        String decimal=st.nextToken();
        if(decimal.length()==1)
          negotiation_price=absolute+delimeter+decimal+"0";
        else if(decimal.length()>2)
          negotiation_price=absolute+delimeter+decimal.substring(0,1);
        String
buyer_price=doc2.getElementsByTagName("negotiation").item(0).getAttributes().getN
amedItem("buyer_price").getNodeValue();
```

```
      String
quantity=doc.getElementsByTagName("negotiation").item(0).getAttributes().getName
dItem("quantity").getNodeValue();
      String
message=doc2.getElementsByTagName("message").item(0).getFirstChild().getNodeV
alue();

      product.put("product_name",product_name);
      product.put("product_price",product_price);
      product.put("provider_id",provider);
      product.put("image",image);
      product.put("description",description);
      product.put("negotiation_price",negotiation_price);
      product.put("buyer_price",buyer_price);
      product.put("quantity",quantity);
      product.put("message",message);

      return product;

    }
    catch (DOMException e)
    {

      e.printStackTrace();
      return null;
    }
    catch (IOException e)
    {
      e.printStackTrace();
      return null;
    }
    catch (ParserConfigurationException e)
    {
      e.printStackTrace();
```

```
            return null;
        }
        catch (SAXException e)
        {
            e.printStackTrace();
            return null;
        }
    }
}

package AgentCommunicationMechanism;

import java.sql.*;
import org.w3c.dom.Document;
import java.io.*;
import java.util.Vector;
import org.w3c.dom.NamedNodeMap;

public class Information_Agent extends ChairMan_Agent

{

    private ResultSet resultSet;

    private ResultSetMetaData metaData;

    private Connection connection;

    private Statement statement;

    private final String JDBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";

    private final String DB_URL = "jdbc:odbc:AgentCommunicationMechanism";
```

```
public Information_Agent()
{
  openConnection();
}

private final void openConnection()
{
  try
  {

    Class.forName(JDBC_DRIVER);

    connection = DriverManager.getConnection(DB_URL);
    statement = connection.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_UPDATABLE);

  }
  catch (SQLException sql)
  {
    sql.printStackTrace();
    System.exit(1);
  }
  catch (ClassNotFoundException cnf)
  {
    cnf.printStackTrace();
    System.exit(1);
  }
}

public void parse(File file)
{}
```

```
  public void parse(Document doc)

  {



    if (doc.getElementsByTagName("request").item(0).getAttributes()
    .getNamedItem("type").getNodeValue().equals("find"))
    {
      NamedNodeMap
negotiation=doc.getElementsByTagName("negotiation").item(0).getAttributes();
      negotiation.getNamedItem("reservation_price").setNodeValue("0.0");
      negotiation.getNamedItem("negotiation_price").setNodeValue("0.0");
      negotiation.getNamedItem("buyer_price").setNodeValue("0.0");
      negotiation.getNamedItem("status").setNodeValue("");
      negotiation.getNamedItem("count").setNodeValue("0");
      String
productid=doc.getElementsByTagName("product_id").item(0).getFirstChild().getNode
Value();
      find(productid);
      String
name=null,price=null,catalogid=null,image=null,description=null,provider=null;
      Vector result=new Vector();
      try
      {
        while (resultSet.next())
        {
          for (int i = 1; i <= metaData.getColumnCount(); i++)
          {
            result.add(resultSet.getObject(i).toString());
          }

        }
        name=result.elementAt(0).toString();
        price=result.elementAt(1).toString();
```

```
            provider=result.elementAt(2).toString();
            catalogid=result.elementAt(3).toString();
            description=result.elementAt(4).toString();
            image=result.elementAt(5).toString();
            closeConnection();
            if(price!=null)
            {
doc.getElementsByTagName("product_name").item(0).getFirstChild().setNodeValue(name);
doc.getElementsByTagName("product_price").item(0).getFirstChild().setNodeValue(price);
doc.getElementsByTagName("provider_id").item(0).getFirstChild().setNodeValue(provider);
doc.getElementsByTagName("catalog_id").item(0).getFirstChild().setNodeValue(catalogid);
doc.getElementsByTagName("description").item(0).getFirstChild().setNodeValue(description);
doc.getElementsByTagName("image").item(0).getFirstChild().setNodeValue(image);
                new Marketing_Agent().parse(doc);
            }
            else
            {
doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("Product NOT Available !");
                transform2XML(doc);
            }
        }
```

```
 catch (SQLException e)
 {
   e.printStackTrace();
 }
}


else if (doc.getElementsByTagName("request").item(0).getAttributes()
.getNamedItem("type").getNodeValue().equals("search"))
{
 String cartegory=doc.getElementsByTagName("productcartegory").item(0)
 .getAttributes().item(0).getTextContent();
 File file=new File( doc.getDocumentURI().substring(6));
 search(cartegory);
 transform2XML(resultSet,file);
 closeConnection();
}

else if (doc.getElementsByTagName("request").item(0).getAttributes()
.getNamedItem("type").getNodeValue().equals("authenticate"))
{
 String username = doc.getElementsByTagName("username").item(0)
 .getFirstChild().getNodeValue();
 String password = doc.getElementsByTagName("password").item(0)
 .getFirstChild().getNodeValue();

 authorize(username, password);

 if (resultSet == null)
   transform2XML(doc);
 else
 {
   try
   {
        while (resultSet.next())
        {
          for (int i = 1; i <= metaData.getColumnCount(); i++)
          {
            doc.getElementsByTagName("accepted").item(0)
            .getFirstChild().setNodeValue("true");
            doc.getElementsByTagName("clientname").item(0)
            .getFirstChild().setNodeValue(
                resultSet.getObject(i).toString());
          }

        }
        closeConnection();
        transform2XML(doc);
      }

      catch (SQLException e)
      {
        e.printStackTrace();
      }

    }

  }

}

private final void search(String cartegory)
{
  openConnection();
  String query ="";
  try
  {
    if(!cartegory.equals(""))
```

```
          query=  "SELECT  PRODUCT.*  FROM  PRODUCT  INNER  JOIN
  "+cartegory+" ON PRODUCT.product_id = "+cartegory+".product_id";
        else
          query="select * from product";
        resultSet = statement.executeQuery(query);
        metaData = resultSet.getMetaData();
      }
      catch (SQLException e)
      {
        System.out
            .println("process not successful\ncheck duplicacy or invalid input !");
        e.printStackTrace();
      }
    }

    private final void authorize(String username, String password)
    {
      openConnection();
      try
      {

        String query = "select agent_name  from agent where agent_id = '"
            + username + password + "'";
        resultSet = statement.executeQuery(query);
        metaData = resultSet.getMetaData();
      }
      catch (SQLException e)
      {
        System.out
            .println("process not successful\ncheck duplicacy or invalid input !");
        e.printStackTrace();
      }
    }
  private final void find(String productid)
  {
    openConnection();
    try
    {

      String query= "SELECT PRODUCT.product_name, PRODUCT.product_price,
PRODUCT.agent_id, CATALOG.* FROM [CATALOG] " +
          "INNER JOIN PRODUCT ON CATALOG.catalog_id =
PRODUCT.catalog_id WHERE (((PRODUCT.product_id)= '"+productid+"'))";

      resultSet = statement.executeQuery(query);
      metaData = resultSet.getMetaData();
    }
    catch (SQLException e)
    {
      System.out
          .println("process not successful\ncheck duplicacy or invalid input !");
      e.printStackTrace();
    }
  }


  private final void closeConnection()
  {
    try
    {
      statement.close();
      connection.close();
    }
    catch (SQLException e)
    {
      e.printStackTrace();
    }
  }

}
package AgentCommunicationMechanism;
import java.io.File;
import java.util.Random;
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
public class Marketing_Agent extends ChairMan_Agent
{
  private  final static double RESERVATION_INCREMENT=5.0;
  private                    final              static              double
NEGOTIATION_INCREMENT=RESERVATION_INCREMENT*2;
  private final static double  DECREMENT=0.5;
  private final static double PROFIT=5.0;
  private final static int COUNT=5;

  public Marketing_Agent()
  {}

  public void parse(File file)
  {}

  public void parse(Document doc)
  {
    NamedNodeMap
negotiation=doc.getElementsByTagName("negotiation").item(0).getAttributes();
    String status=negotiation.getNamedItem("status").getNodeValue();
    double
reservation_price=Double.parseDouble(negotiation.getNamedItem("reservation_price")
.getNodeValue());
```

```
    double
negotiation_price=Double.parseDouble(negotiation.getNamedItem("negotiation_price"
).getNodeValue());
    double
buyer_price=Double.parseDouble(negotiation.getNamedItem("buyer_price").getNodeV
alue());
    double
product_price=Double.parseDouble(doc.getElementsByTagName("product_price").ite
m(0).getFirstChild().getNodeValue());
    int count=Integer.parseInt(negotiation.getNamedItem("count").getNodeValue());

    if(status.equals(""))
    {
      String
price1=((Double)(product_price+RESERVATION_INCREMENT)).toString();
      String
price2=((Double)(product_price+NEGOTIATION_INCREMENT)).toString();
      negotiation.getNamedItem("reservation_price").setNodeValue(price1);
      negotiation.getNamedItem("negotiation_price").setNodeValue(price2);
      negotiation.getNamedItem("buyer_price").setNodeValue(price2);

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("THE
BID PRICE HAS BEEN SET, YOU MAY BEGIN NEGOTIATION.");
    }
    else if(status.equals("NEGOTIATING") || status.equals("AGREEMENT"))
    {
      if(buyer_price>=negotiation_price)
      {
       negotiation.getNamedItem("status").setNodeValue("AGREEMENT");

negotiation.getNamedItem("negotiation_price").setNodeValue(Double.valueOf(buyer_
price).toString());

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("YOU
R NEGOTIATION HAS COME TO AN AGREEMENT, PURCHASE THE
PRODUCT.");
        }
        else
if(processNegotiation(reservation_price,negotiation_price,buyer_price,count))
        {

negotiation.getNamedItem("negotiation_price").setNodeValue(Double.valueOf(buyer_
price).toString());
          negotiation.getNamedItem("status").setNodeValue("AGREEMENT");

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("YOU
R NEGOTIATION HAS COME TO AN AGREEMENT, PURCHASE THE
PRODUCT.");
        }
        else
        {

          if(count<=COUNT)
          {
            double                                    newnegotiation_price=
lowerPrice(reservation_price,negotiation_price);
            System.out.println("NEW      NEGOTIATION      PRICE              :
"+newnegotiation_price);
             System.out.println(((Double)newnegotiation_price).doubleValue());
            if( newnegotiation_price==negotiation_price)

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("PRIC
E COULD NOT BE AMENDED : INCREASE PRICE AND RENEGOTIATE, YOU
HAVE  "+(COUNT-count)+" GRACE NEGOTIATION LEFT.");
            else

negotiation.getNamedItem("negotiation_price").setNodeValue(Double.valueOf(newne
gotiation_price).toString());
```

```
doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("PRIC
E AMENDED, RENEGOTIATE, YOU HAVE "+(COUNT-count)+" GRACE
NEGOTIATION LEFT.");
        }
        else
        {
          negotiation.getNamedItem("status").setNodeValue("BREAKDOWN");

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("NEG
OTIATION BREAKDOWN , YOU HAVE EXCEEDED YOUR GARACE LIMIT");
        }
      }


    }
    else if(status.equals("BREAKDOWN"))
    {

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("PRO
DUCT CANNOT BE PURCHASED, NO FURTHER NEGOTIATION IS
ALLOWED.");
    }

    transform2XML(doc);
  }
  private boolean processNegotiation(double reservation_price,double
negotiation_price,double buyer_price,int count)
  {
    if(buyer_price > reservation_price)
    {
      if((buyer_price - reservation_price) >= PROFIT)
        return true;
      else
      {
        if((COUNT-count)<=0)
          return true;
        else if((buyer_price - reservation_price) >= PROFIT-1 && (COUNT-
count)<=1)
          return true;
        else
          return false;
      }
    }
    else
      return false;
  }

  private double lowerPrice(double reservation_price,double negotiation_price)
  {
    double newnegotiation_price=0.0;
    String random=((Integer)new Random().nextInt(6)).toString();
    Double randomvalue=new Double(random);
    newnegotiation_price=negotiation_price -((Double)(randomvalue/10));
    if (negotiation_price == newnegotiation_price || reservation_price >=
negotiation_price )
      lowerPrice(reservation_price,negotiation_price);
    if(reservation_price >= negotiation_price )
      return negotiation_price;
    else
      return newnegotiation_price;
  }
}
package AgentCommunicationMechanism;
import java.io.File;
import java.util.Random;
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
public class Marketing_Agent extends ChairMan_Agent
```

```
{
  private  final static double RESERVATION_INCREMENT=5.0;
  private final static double
NEGOTIATION_INCREMENT=RESERVATION_INCREMENT*2;
  private final static double  DECREMENT=0.5;
  private final static double PROFIT=5.0;
  private final static int COUNT=5;

  public Marketing_Agent()
  {}

  public void parse(File file)
  {}

  public void parse(Document doc)
  {
    NamedNodeMap
negotiation=doc.getElementsByTagName("negotiation").item(0).getAttributes();
    String status=negotiation.getNamedItem("status").getNodeValue();
    double
reservation_price=Double.parseDouble(negotiation.getNamedItem("reservation_price")
.getNodeValue());
    double
negotiation_price=Double.parseDouble(negotiation.getNamedItem("negotiation_price"
).getNodeValue());
    double
buyer_price=Double.parseDouble(negotiation.getNamedItem("buyer_price").getNodeV
alue());
    double
product_price=Double.parseDouble(doc.getElementsByTagName("product_price").ite
m(0).getFirstChild().getNodeValue());
    int count=Integer.parseInt(negotiation.getNamedItem("count").getNodeValue());

    if(status.equals(""))
    {
       String
price1=((Double)(product_price+RESERVATION_INCREMENT)).toString();
       String
price2=((Double)(product_price+NEGOTIATION_INCREMENT)).toString();
       negotiation.getNamedItem("reservation_price").setNodeValue(price1);
       negotiation.getNamedItem("negotiation_price").setNodeValue(price2);
       negotiation.getNamedItem("buyer_price").setNodeValue(price2);

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("THE
BID PRICE HAS BEEN SET, YOU MAY BEGIN NEGOTIATION.");
    }
    else if(status.equals("NEGOTIATING") || status.equals("AGREEMENT"))
    {
       if(buyer_price>=negotiation_price)
       {
        negotiation.getNamedItem("status").setNodeValue("AGREEMENT");

negotiation.getNamedItem("negotiation_price").setNodeValue(Double.valueOf(buyer_
price).toString());

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("YOU
R NEGOTIATION HAS COME TO AN AGREEMENT, PURCHASE THE
PRODUCT.");
       }
       else
if(processNegotiation(reservation_price,negotiation_price,buyer_price,count))
       {

negotiation.getNamedItem("negotiation_price").setNodeValue(Double.valueOf(buyer_
price).toString());
          negotiation.getNamedItem("status").setNodeValue("AGREEMENT");

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("YOU
R NEGOTIATION HAS COME TO AN AGREEMENT, PURCHASE THE
PRODUCT.");
        }
        else
        {

           if(count<=COUNT)
           {
             double                                     newnegotiation_price=
lowerPrice(reservation_price,negotiation_price);
             System.out.println("NEW      NEGOTIATION      PRICE            :
"+newnegotiation_price);
              System.out.println(((Double)newnegotiation_price).doubleValue());
             if( newnegotiation_price==negotiation_price)

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("PRIC
E COULD NOT BE AMENDED : INCREASE PRICE AND RENEGOTIATE, YOU
HAVE "+(COUNT-count)+" GRACE NEGOTIATION LEFT.");
             else

negotiation.getNamedItem("negotiation_price").setNodeValue(Double.valueOf(newne
gotiation_price).toString());

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("PRIC
E AMENDED, RENEGOTIATE, YOU HAVE "+(COUNT-count)+" GRACE
NEGOTIATION LEFT.");
           }
           else
           {
             negotiation.getNamedItem("status").setNodeValue("BREAKDOWN");

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("NEG
OTIATION BREAKDOWN , YOU HAVE EXCEEDED YOUR GARACE LIMIT");
           }
```

```
      }
    }
    else if(status.equals("BREAKDOWN"))
    {

doc.getElementsByTagName("message").item(0).getFirstChild().setNodeValue("PRO
DUCT CANNOT BE PURCHASED, NO FURTHER NEGOTIATION IS
ALLOWED.");
    }

    transform2XML(doc);
  }
  private boolean processNegotiation(double reservation_price,double
negotiation_price,double buyer_price,int count)
  {
    if(buyer_price > reservation_price)
    {
      if((buyer_price - reservation_price) >= PROFIT)
        return true;
      else
      {
        if((COUNT-count)<=0)
          return true;
        else if((buyer_price - reservation_price) >= PROFIT-1 && (COUNT-
count)<=1)
          return true;
        else
          return false;
      }
    }
    else
      return false;
  }
  private double lowerPrice(double reservation_price,double negotiation_price)
  {
    double newnegotiation_price=0.0;
    String random=((Integer)new Random().nextInt(6)).toString();
    Double randomvalue=new Double(random);
    newnegotiation_price=negotiation_price -((Double)(randomvalue/10));
    if (negotiation_price == newnegotiation_price || reservation_price >=
negotiation_price )
      lowerPrice(reservation_price,negotiation_price);
    if(reservation_price >= negotiation_price )
      return negotiation_price;
    else
      return newnegotiation_price;
  }
}
```

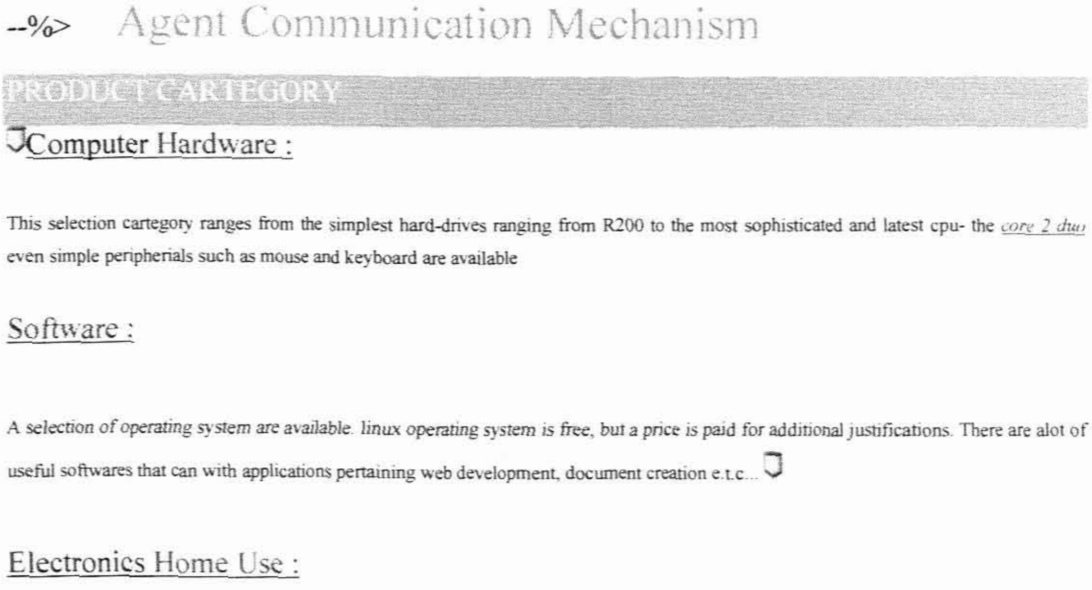
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>Agent.ComM</display-name>
  <context-param>
    <param-name>agentcom</param-name>
    <param-value>C:/webapps/agentcom/</param-value>
  </context-param>
  <servlet>
    <servlet-name>Login</servlet-name>
    <jsp-file>/Login.jsp</jsp-file>
  </servlet>
  <servlet>
    <servlet-name>Authenticate</servlet-name>
    <jsp-file>/Authenticate.jsp</jsp-file>
    </servlet>
  <servlet>
  <servlet-name>Catalog</servlet-name>
  <jsp-file>/Product.jsp</jsp-file>
</servlet>
<servlet>
  <servlet-name>Search</servlet-name>
  <jsp-file>/Result.jsp</jsp-file>
</servlet>
<servlet>
  <servlet-name>Negotiate</servlet-name>
  <jsp-file>/Negotiate.jsp</jsp-file>
</servlet>
<servlet>
  <servlet-name>Purchase</servlet-name>
  <jsp-file>/Purchase.jsp</jsp-file>
</servlet>
<servlet>
  <servlet-name>Home</servlet-name>
  <jsp-file>/Home.jsp</jsp-file>
</servlet>
<servlet>
  <servlet-name>Advertisement</servlet-name>
  <jsp-file>/Advertisement.jsp</jsp-file>
</servlet>
<servlet-mapping>
  <servlet-name>Login</servlet-name>
  <url-pattern>/login</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>Authenticate</servlet-name>
  <url-pattern>/authenticate</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>Catalog</servlet-name>
  <url-pattern>/product</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Search</servlet-name>
    <url-pattern>/search</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Negotiate</servlet-name>
    <url-pattern>/transaction/negotiate</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Purchase</servlet-name>
    <url-pattern>/purchase</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Home</servlet-name>
    <url-pattern>/home</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Advertisement</servlet-name>
    <url-pattern>/advertisement</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>Home.jsp</welcome-file>
  </welcome-file-list>
  </web-app>
```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
```

```
<%--
The taglib directive below imports the JSTL library. If you uncomment it,
you must also add the JSTL library to the project. The Add Library... action
on Libraries node in Projects view can be used to add the JSTL 1.1 library.
--%>
<%--
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<%@ page  import="java.io.File" %>
<%@ page  import="java.io.FileNotFoundException" %>
<%@ page  import=" java.util.Enumeration" %>
<%@ page  import="java.util.Hashtable" %>

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

      <script src="all.js" type="text/javascript">
      </script>

      <style  type="text/css">
         <jsp:include page="Style.jsp" flush="true"/>




      </style>

      <title>Search</title>
   </head>
   <body vlink="black" alink="black" link="black" >
```

```
<jsp:include page="Background.jsp" flush="true"/>


<div class="menu" style="top:15em;background-color:pink;width:20em">
  <span class="menuitem" style="padding-left:12em;color:white;text-decoration:none"> PRODUCT CARTEGORY </span>
</div>


<span class="searchtext"style="top:27em;margin-left:2em">
  <img style="menuitem" src="pics/thumb.bmp" width="10" height="15" alt="expand"/>
</span>
<span class="searchtext"style="margin-left:3em;top:27em"><a href="/agentcom/search?cartegory=computer"> Computer </a></span>
<span class="searchtext"style="margin-left:4em;top:28em"><a href="/agentcom/search?cartegory=hardware" style="color:red">Hardware :</a></span>
<span class="searchtext"style="margin-left:5em;top:30em;font-size:7.5pt;font-family:veranda;width:24em;color:blue">
    This selection cartegory ranges from the simplest hard-drives ranging from R200 to the most sophisticated
    and latest cpu- the <em><a href="/agentcom/product?product=core2duo"> core 2 duo </a></em> even simple peripherials such as mouse and keyboard are available
</span>
<span class="searchtext"style="margin-left:4em;top:37em"><a href="/agentcom/search?cartegory=software" style="color:red">Software :</a></span>
<span class="searchtext"style="margin-left:5em;top:38em;font-size:7.5pt;font-family:veranda;width:24em;color:blue">
    A selection of operating system are available. linux operating system is free, but a price is paid for additional justifications.
    There are alot of useful softwares that can with applications pertaining web development, document creation e.t.c...
</span>
```

```
        <span class="searchtext"style="top:45em;margin-left:2em">
           <img style="menuitem" src="pics/thumb.bmp" width="10" height="15"
alt="expand"/>
        </span>
        <span class="searchtext"style="margin-left:3em;top:45em"><a
href="/agentcom/search?cartegory=electronic"> Electronics </a></span>
        <span class="searchtext"style="margin-left:4em;top:46em"><a
href="/agentcom/search?cartegory=home_electronic" style="color:red">Home Use
:</a></span>
        <span class="searchtext"style="margin-left:5em;top:47em;font-size:7.5pt;font-
family:veranda;width:24em;color:blue">
           This selection entails a variety of electronics for home use such as blenders,
oven, microwave and many more...
        </span>



    </body>
</html>
--%>
```



Agent Communication Mechanism

PRODUCT CARTEGORY

Computer Hardware :

This selection cartegory ranges from the simplest hard-drives ranging from R200 to the most sophisticated and latest cpu- the *core 2 duo* even simple peripherials such as mouse and keyboard are available

Software :

A selection of operating system are available. linux operating system is free, but a price is paid for additional justifications. There are alot of useful softwares that can with applications pertaining web development, document creation e.t.c...

Electronics Home Use :

This selection entails a variety of electronics for home use such as blenders, oven, microwave and many more...