EVALUATING THE TRADEOFFS OF MOBILE CODE DESIGN PARADIGMS IN ELECTRONIC COMMERCE APPLICATIONS

THESIS

Presented to the Graduate Council
of Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Naveen Koneru, B.E.

San Marcos, Texas

August 2005

To

My beloved Gurudev,

Sri Sri Ravi Shankar

ACKNOWLEDGEMENTS

It takes patience and willingness to guide a student. Dr. Hall, my advisor and the chair of my thesis committee, has given both to me in abundance. I would like to thank him for his insight, guidance and constant encouragement. My thankfulness and gratitude extend to Professor Davis and Dr. Drissi, members of my thesis committee. I thank Dr. Christine Julien, for her genuine interest and encouragement in this project.

I would also like to thank Anil Enumulapally, Narasimhan Kaliyamoorthy, Sriram Rajan, Dr. David Byrum and Dr. Don Shafer for their technical help; Marisol Mendez and Roberto Renaud for proof reading the manuscript. This research project would not have been complete without their help and kindness.

My deepest gratitude to my parents, K.B.S. Saibabu and K. Krishna Kumari, and my brothers Shravanth, Rajeev, Sreenivas, Vishnu and Vikram, for their constant support and encouragement. They have stood by me through thick and thin, always encouraging me. My thanks to numerous friends and well wishers, all of whose names cannot be mentioned here. They have given me unwavering support and encouragement.

Above all, my deepest gratitude to my Gurudev, His Holiness Sri Sri Ravi Shankar. He has taught me to trust, given me hope and rekindled my inspiration to work. I thank Rajshree, Rikka, Chandu, Sudheer, Tej, Sathyan and Preethi for their care.

This manuscript was submitted on July 29th 2005.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ..... **iv**
LIST OF TABLES ..... **viii**
LIST OF FIGURES ..... **ix**
ABSTRACT ..... **xi**

CHAPTER 1
INTRODUCTION **1**
1.0 Explanation of the Problem 1
1.1 Mobile Code 1
1.2 Electronic Commerce 3
1.3 Mobile Code in Electronic Commerce 4
1.4 Research Objectives and Scope 5
1.5 Methodology 5
1.6 Organization of the Thesis 6

CHAPTER 2
MOBILE CODE DESIGN PARADIGMS **8**
2.0 Overview 8
2.1 Motivations for using Mobile Code 8
2.2 Concepts of Mobile Code 9
2.3 Design Paradigms in Mobile Code 13
2.4 Tradeoffs of Mobile Code Implementations 17
2.5 Domains of Implementation 18

CHAPTER 3

ELECTRONIC COMMERCE APPLICATIONS 20

3.0 Overview 20

3.1 Survey of Electronic Commerce 20

3.2 Business Scenarios in Electronic Commerce 22

3.3 Activities in Electronic Commerce 30

3.4 Issues in Electronic Commerce 33

3.5 Implemented Technologies in Electronic Commerce 34

3.6 Using Mobile Code in Electronic Commerce 35

CHAPTER 4

METHODOLOGY OF APPLYING MOBILE CODE DESIGN PARADIGMS TO ELECTRONIC COMMERCE APPLICATIONS 37

4.0 Overview 37

4.1 Literature Review on Simulation Frameworks 37

4.1.1 Simulation Framework of Electronic Commerce Applications 38

4.1.2 Simulation Framework for Implementing Various Mobile Code Design Paradigms 38

4.1.3 Simulation Framework for Implementing a Recommender E-commerce Application Using Various Mobile Code Paradigms 39

4.2 Simulation Framework Implemented 40

4.2.1 Vision of the Ideal Simulation Framework 40

4.2.2 Features of the Ideal Simulation Framework 41

4.2.3 Description of the Ideal Simulation Framework 42

4.2.4 Scope and Implementation of Simulation Framework 44

4.3 Chosen Scenarios and Activities in Electronic Commerce for Implementation 47

4.4 Application of Mobile Code Paradigms 49

4.4.1 Remote Evaluation Design Paradigm 51

4.4.2 Code on Demand Design Paradigm 52

4.4.3 Mobile Agent Design Paradigm 54

4.5 Limitations of the Implemented Framework 56
4.6 Tradeoffs Selected for Evaluation of Mobile Code Design Paradigms 57
4.6.1 Latency of Response to the Query 57

CHAPTER 5
RESULTS AND ANALYSIS 64
5.0 Overview 64
5.1 Quantitative Metrics 64
5.1.1 Data Sets of Input Parameters and Output Results 64
5.1.2 Latency of Response to the Query 67
5.1.3 Code Metrics 80
5.2 Qualitative Analysis 81
5.3 Evaluation of Trade-offs of Mobile Code Design Paradigms for Recommend Activity in the Experiment 85
5.4 Inference About Other E-commerce Scenarios from Information and Statistics on the Current Scenario 88
5.5 Abstraction of all Mobile Code Design Paradigms 89

CHAPTER 6
CONCLUSIONS AND FUTURE WORK 91
6.0 Overview 91
6.1 Conclusions 91
6.2 Limitations 94
6.3 Directions for Future Research 95

APPENDIX A 98
APPENDIX B 100
REFERENCES 123

LIST OF TABLES

Table 2.1: Relocation strategies of Mobile Code design paradigms.....15

Table 3.1: Industries in manufacturing and wholesale trade sectors.....32

Table 3.2: Industries in selected services and retail sectors.....32

Table 3.3: Miscellaneous areas of business where e-commerce is used.....32

Table 5.1: Data sets of all implemented input parameters 66

Table 5.2 Data sets of implemented input parameters for B2C paradigm.....67

Table 5.3: Data set 1 of implemented input parameters to measure latency of query response.....68

Table 5.4: Data sets of input parameters for measuring latency of query response with increasing number of Businesses and 20 customers 70

Table 5.5: Author and design parameters for of computational components for CS paradigm 82

Table 5.6 Author and design parameters for of computational components for REV paradigm.....83

Table 5.7 Author and design parameters for of computational components for COD paradigm.....84

Table 5.8 Author and design parameters for of computational components for MA paradigm.....85

Table 5.9: Comparison of tradeoffs of REV, COD and MA paradigm 87

LIST OF FIGURES

Fig 2.1 Traditional Distributed System.....10

Fig 2.2 Mobile Code System.....11

Fig 2.3 The internal structure of an executing unit.....12

Fig 3.1 B2B e-commerce.....23

Fig 3.2: B2C e-commerce.....25

Fig 3.3: P2P or C2C e-commerce.....27

Fig 3.4: m-commerce.....29

Fig 3.5: Layered technology architecture of e-commerce applications.....34

Figure 4.1: Ideal Simulation Framework.....43

Figure 4.2: Software structure of the computer host 44

Figure 4.3: System architecture of the Framework used for simulation 46

Figure 4.4: Remote Evaluation Design Paradigm.....51

Figure 4.5: Code on Demand Design Paradigm.....53

Figure 4.6: Mobile Agent Design Paradigm.....55

Fig 4.7: Timestamp noted in the simulation framework for performing a query using Mobile Code.....59

Figure 5.1: Latency of query response with increasing number of Customers and 1 Business per host 69

Figure 5.2: Figure 5.2: Latency of query response with increasing number of Businesses and 20 customers per host.....72

Figure 5.3: Travel time of Mobile Code to the Business with increasing number of Customers and 1 Business per host.....74

Figure 5.4 Travel time of Mobile Code to the Business with increasing number of Businesses and 20 Customers per host.....75

Figure 5.5: Changes in average times by increase in number of hosts in REV paradigm.....78

Figure 5.6: Changes in average times by increase in number of hosts in COD paradigm.....78

Figure 5.7: Changes in average times by increase in number of hosts in MA paradigm.....79

Figure 5.8: Comparison of executable lines of code.....81

Figure A1: Interaction of classes in the simulation framework.....99

ABSTRACT

EVALUATING THE TRADEOFFS OF MOBILE CODE DESIGN PARADIGMS IN ELECTRONIC COMMERCE APPLICATIONS

by

Naveen Koneru, B.E.

Texas State University – San Marcos

August 2005

SUPERVISING PROFESSOR: GREGORY HALL

Mobile Code has the ability to relocate dynamically and change bindings between computational environments and code fragments. The flexibility of Mobile Code to relocate and execute code in different patterns has introduced Mobile Code design paradigms. Mobile Code design paradigms can be applied to a specific domain of application, independent of the implementation technology. The results of applying Mobile Code design paradigms in e-commerce applications yield tradeoffs depending on the conditions of the application. The tradeoffs have been evaluated to study the behavior of the paradigms under varying conditions of their implementation.

CHAPTER 1

INTRODUCTION

1.0 Explanation of the Problem

The ability of code to dynamically relocate and change bindings between code fragments and the location is known as Mobile Code [Fuggetta 1998]. The concepts of Mobile Code introduce new possibilities for design paradigms [Carzaniga1997]. Mobile Code has features and capabilities that can be beneficial in implementing it in e-commerce applications. Research has indicated that the Mobile Code design paradigms implemented in other domains of applications have had tradeoffs depending on the conditions of the application of the Mobile Code design paradigms [Baldi 1998]. There has not been considerable investigation in the tradeoffs of the Mobile Code design paradigms when applied to e-commerce applications.

1.1 Mobile Code

In conventional systems, code executed by a program at any location is bound to the resources and the location where the code resides. Mobile Code

introduces the possibility for a programmer to control the dynamic relocation of the code with its data from one computational environment to another.

Though the implementations of Mobile Code have existed for several years, they have only recently been classified into a category known as Mobile Code. There is a general misconception that Mobile Code means mobile agents. While it is true that mobile agents are one kind of Mobile Code, there are other forms of Mobile Code that are different from mobile agents.

The difference between Mobile Code and conventional distributed computing is that the various hosts participating in the distributed computing remain anonymous to the programmer. The programmer does not have to worry about the implementation details concerning the identity of the host where the program is executed. The distributed system makes a conglomeration of hosts appear as only one layer and provides support for the layer. In the case of Mobile Code, the programmer is aware of the identity of the host where the Mobile Code program is executed. The programmer has control to relocate the Mobile Code to a specific host along with the data state, and in some types of Mobile Code, the execution state also.

Mobile Code can relocate in 2 different ways depending on the degree of binding that is retained after relocation. This results in 2 types of Mobile Code; strong mobility and weak mobility. Strong mobility is the ability of the code to relocate its execution state as well as the data state to another computational environment and retain the execution and data states. Weak mobility is the ability of the Mobile Code to relocate only the data state of the Mobile Code from one

computational environment to another. An example of strong mobility is Tcl Agents while Java is an example of weak mobility.

The ability to relocate code is the motivation for the use of Mobile Code. It introduces possibilities of new patterns of interaction with resources. The emerging patterns of interaction are known as Mobile Code design paradigms. These include paradigms like remote evaluation, code on demand and mobile agents. As aforementioned, some of these paradigms have existed prior to their classification as Mobile Code.

In every design paradigm, the pattern of interaction among the computational components has tradeoffs both quantitatively and qualitatively.

1.2 Electronic Commerce

Trade of goods and services conducted through the digital transaction of information, money or any monetary worth is known as electronic commerce (e-commerce). E-commerce has produced huge revenues in previous years during its short span of existence.

At its advent, e-commerce was used only by business organizations. However, e-commerce is now being widely used by business organizational entities to become an implicitly integral part of the organization. The drop in the price of computing power, and usability of the Internet has widely brought accessibility of computers to the end-consumer. Hence, in the last decade, there has been a huge rise in the e-commerce sectors involving the end-customers also.

There are several scenarios of trade in e-commerce depending on the participants. They are Business-to-Business (B2B), Business-to-Customer (B2C), Peer-to-Peer (P2P) and Mobile commerce (m-commerce). The highest revenues are in the B2B scenario followed by the B2C scenario. E-commerce is applied in several fields of goods and services. The technologies introduced by e-commerce have introduced new possibilities of trade.

There are several technologies that are utilized and implemented by e-commerce. Among these are Hyper Text Markup Language (HTML), Hyper Text Transfer Protocol (HTTP), Extensible Markup Language (XML), Java, etc. E-commerce has developed so rapidly that large scale technologies are emerging specifically for e-commerce. Mobile Code has also been deployed in the domain of e-commerce applications to a minor extent.

1.3 Mobile Code in Electronic Commerce

The most implemented paradigms and technologies of Mobile Code in e-commerce have been mobile agents and Java applets respectively. The Java applets have been introduced by Sun Microsystems since the early days of Java. They have gained popularity and usage with the advent of the Internet. Java, which features weak mobility, has been used in the implementation of e-commerce applications. Some of the other e-commerce implementation technologies that use Mobile Code are Common Object Request Broker Architecture (CORBA) and Java Remote Method Invocation (RMI).

Mobile Code has great potential for implementations in e-commerce applications because of the flexibility in its ability to interact. More avenues for the application of Mobile Code in e-commerce applications are yet to be explored.

1.4 Research Objectives and Scope

The objective of the research is to evaluate the tradeoffs of various Mobile Code design paradigms when they are applied to e-commerce applications. The tradeoffs are measured quantitatively based on a metric which is the latency of response to query.

The scope of the research is restricted to the B2C scenario in e-commerce where the business and customer entities will be simulated in a framework in a lab environment on a limited number of participating hosts in a network. The Mobile Code paradigms that are implemented are Remote Evaluation (REV), Code on Demand (COD) and Mobile Agents (MA).

1.5 Methodology

A framework is developed where the various Mobile Code design paradigms can be implemented. The framework is built on another lightweight mobile code system known are μCode. The proposed metric for measurement is latency of response to query.

Data sets of combinations of a number of businesses and customers are chosen that coincide with the characteristics of a B2C scenario. The data set indicates the number of businesses, customers and number of hosts running at a time using the simulation framework. The simulation framework allows the choice of any one of REV, COD or MA paradigms to be implemented.

Upon execution, the metric gives the value of latency in the response to the query. Other quantitative metrics can also be calculated for each design paradigm. Upon analysis of the procured data on quantitative metrics and qualitative analysis, it is concluded that there are tradeoffs for every Mobile Code design paradigm under the given conditions of application.

1.6 Organization of the Thesis

A description of the problem is presented in the Introduction, Chapter 1. A brief introduction to the concepts of Mobile Code and e-commerce are presented in the Introduction Chapter followed by a description of the methodology of the experiment.

Mobile Code and its design paradigms are dealt with in detail in Chapter 2. The finer concepts of Mobile Code with a detailed explanation of the functionality of Mobile Code are then presented.

Chapter 3 begins by introducing the basics of electronic commerce. Then, the various activities and scenarios of the functioning are discussed. The issues and implemented technologies in e-commerce are discussed followed by the use of Mobile Code in e-commerce applications.

The Methodology, or Chapter 4, discusses the literature survey on the different simulation frameworks. Chapter 4 then describes the chosen activities and scenarios in e-commerce. The way the metric is measured is explained along with the methodology of the experiment.

The results of the experiment are discussed and analyzed in chapter 5. The tradeoffs are discussed followed by conclusions in Chapter 6. Chapter 6 concludes with directions for future research work in Mobile Code. Appendix A contains the program classes with descriptions of their data members and member functions.

CHAPTER 2

MOBILE CODE DESIGN PARADIGMS

2.0 Overview

This chapter gives an introduction to Mobile Code. Motivations for using Mobile Code are discussed followed by fundamental concepts of Mobile Code such as definitions, functional details and the types of Mobile Code. Concepts of Mobile Code relevant for this research, such as design paradigms, are explained. The tradeoffs of using Mobile Code are presented and followed by a few examples in the domains of implementation.

2.1 Motivations for using Mobile Code

Distributed computing has grown such over the years that is has become pervasive. The large extent of connectivity through the Internet has posed problems such as wide distribution of resources. Therefore, such problems lead to the inefficient utilization of resources. There is a constant search for distributed solutions to scale up to the size of a large distribution of computing resources

[Carzaniga 1997]. Mobile Code is one of the approaches to a solution. Though Mobile Code is not an entirely new concept, the different approaches that are practiced have been brought together under this concept to lead to distributed solutions that make more efficient utilization of these resources even though they are widely distributed [Cugola 1997].

2.2 Concepts of Mobile Code

> 'Code Mobility' can be defined informally as the capability to dynamically change the bindings between code fragments and the location where they are executed [Fuggetta 1998.].

Code mobility can be used interchangeably with terms such as Mobile Code and Program Mobility. In order to maintain consistency, we will use the term 'Mobile Code' in this document.

In the distributed systems that use Mobile Code, as represented in Fig 2.1 [Fuggetta 1998], the host is identified as the hardware of a single unit of the distributed system. The lowest layer in the software is the Core Operating System (COS).

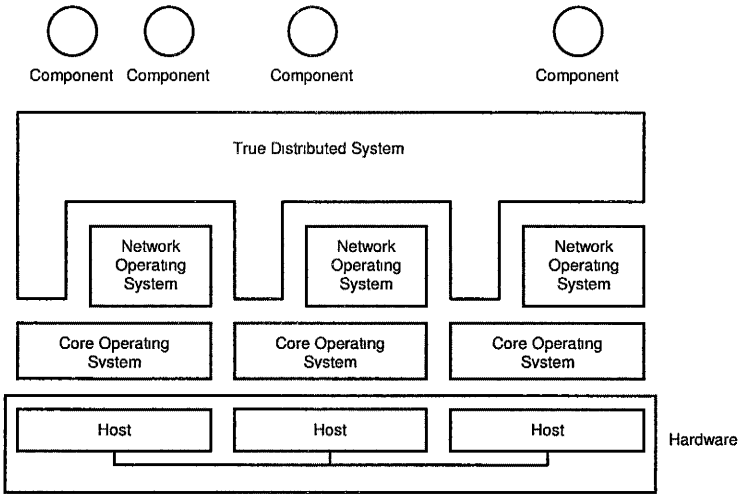

Figure 2.1 Traditional Distributed System [Fuggetta 1998]

The COS is responsible for basic activities such as memory management, file operations, input/output device management, etc. The COS is not responsible for any network activities. The layer above the COS is the Network Operating System (NOS). The NOS provides services to applications that have communication addressed to specific hosts. An example of such a service would be socket services.

In traditional distributed systems as represented in Fig 2.1 [Fuggetta 1998], there is a True Distributed System (TDS) layer above the NOS. The TDS layer conceals the identity of the host and the lower details that are handled by the NOS and the COS. The purpose of the TDS is to make the entire system look like one system to the programmer.

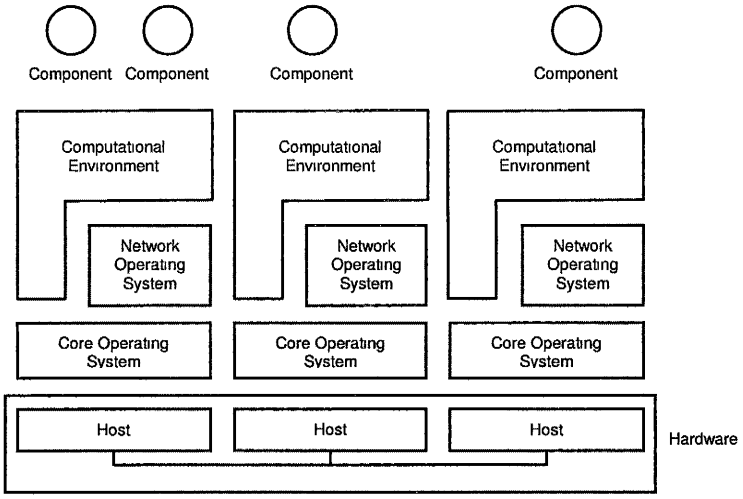

Fig 2.2 Mobile Code System [Fuggetta 1998]

In Mobile Code Systems (MCS) as represented in Fig 2.2 [Fuggetta 1998], the Computational Environment (CE) replaces the TDS layer in traditional distributed systems. In contrast to TDS in traditional distributed systems, CE in Mobile Code Systems retains the identity of the host where it is located in the network. The CE has access to the lower level services of the NOS and the COS. By doing so, the CE provides the ability for the resident applications to dynamically relocate components to a specific host. The components can be classified into two categories, namely Computational Components (or Executing Units) and resources. Computational components or Executing Units (EU) are a combination of elements that represent a flow of computation. The resources are the components that are used by the EUs like files, etc. EUs contain references to the resources in their data spaces.

Typically, an EU is comprised of a code segment to be executed utilizing the information about the execution state of the EU and the data space as

illustrated in Fig 2.3. The code segment is a static description of the EU. The execution state consists of information related to the EU state such as the call stack and the instruction pointer. This is data that is private to the EU and cannot be shared. The data space is a set of references to resources that can be accessed by the EU.

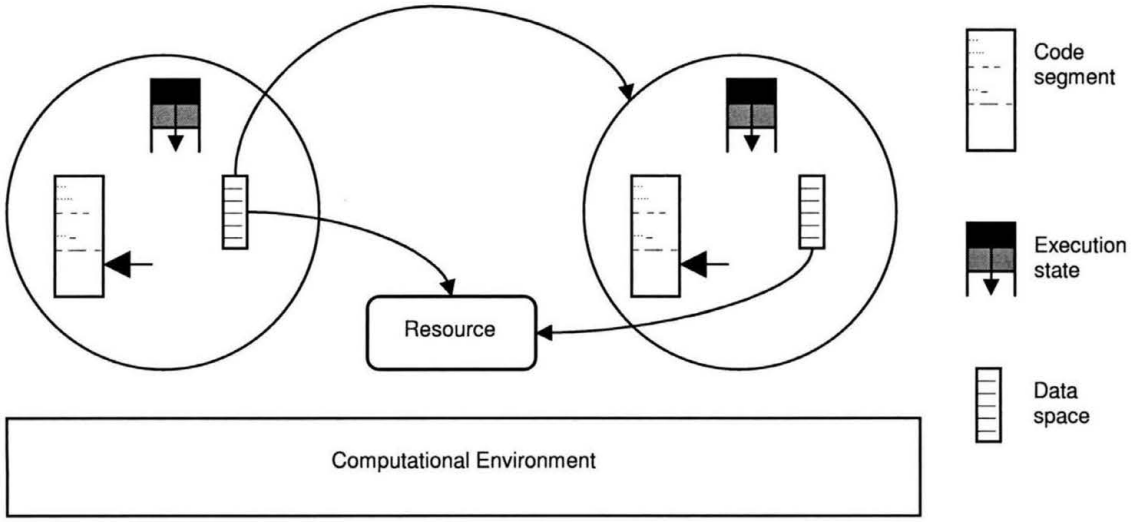

Fig 2.3 Internal structure of an executing unit [Fuggetta 1998]

In conventional systems, the binding between the EU and the CE is static. In Mobile Code Systems, the components of the EU are not tightly bound to the CE. The code segment, execution state and data space of an EU can be relocated to another EU in another CE. The degree of mobility of the components of EU to the CE is known as the degree of mobility in the Mobile Code systems. Based on the binding between the Execution Units (EUs) and the Computational Environments (CEs), there are two types of Mobile Code [Carzaniga 1997], namely: Strong Mobility and Weak Mobility.

Strong Mobility is the ability of a Mobile Code Language to allow EUs to move their code to a different CE by moving both the code and the execution state completely. The executing units are suspended, transmitted to the destination site, and resumed there.

Weak Mobility is the ability of a Mobile Code Language to allow an EU in a site to be bound dynamically to code coming from a different site. This encompasses two cases: either the EU dynamically links the code downloaded from the network, or the EU receives its code from another EU [Carzaniga 1997].

2.3 Design Paradigms in Mobile Code

The complexity that is introduced in Mobile Code due to the relocation of computational components and resources has led to the development of Mobile Code Languages (MCL) in recent years. [Cugola 1997]. A few examples of MCLs are D'Agents [Gray 2002], Telescript [Thomas], IBMAglets [Lange 1999] and Java [Sandakly]. Some MCLs are developed based on a specific design paradigm. However, there exist design paradigms in Mobile Code which are architectural abstractions that can be implemented irrespective of the MCLs [Carzaniga 1997].

Architectural abstraction of the components, resources, and the interactions between them is known as Software design. Design principles in software engineering indicate that the design of software should be independent of the implementation technologies. While it is true that the availability of good

languages and tools will be a positive aid in the development of software, it is not imperative that the languages and tools enforce the implementation of good design principles. For example, an object oriented language is ideal for implementation when the software is designed using the object oriented methodology. However, an object oriented language can also be used to implement programs with sequential design. Similarly, though structured programming languages are more suitable to implement sequential design in programs, they can also be used to implement object oriented design.

Hence, developing design paradigms in Mobile Code that are independent of the implementing programming language is significant. The design paradigms will largely depend on the nature of Mobile Code, which arises out of a need to relocate distributed resources efficiently.

In distributed systems, it is possible that the resources and the parts of the computational components may be located at different computational environments (CE). A computational process or an operation often involves the presence of all these components in one CE. The process of relocation of the components to a CE is known as the design paradigm. The objective in the abstraction of the design paradigms in Mobile Code is the methodology of relocation of computational components and resources to perform a computing operation.

In Table 2.1, A and B are computational components that are originally in the computational sites S_A and S_B. The computation is to be performed by A.

The code needed to complete the process or the resources that are required to perform the computation are distributed in the sites S_A and S_B.

Paradigm	Before		After	
	S_A	S_B	S_A	S_B
Client - Server	A	Know-how Resource B	A	Know-how Resource B
Remote Evaluation	Know-how A	Resource B	A	Know-how Resource B
Code on Demand	Resource A	Know-how B	Know-how Resource A	B
Mobile Agent	Know-how A	Resource	---	Know-how Resource A

Table 2.1: Relocation strategies of Mobile Code design paradigms Ref[2.3]

Common design paradigms in Mobile Code are Remote Evaluation (REV), Code on Demand (COD) and Mobile Agent (MA). A comparison of the functional aspect of these design paradigms along with the Client-Server paradigm is presented below:

1) Client Server

In this paradigm, A is the client and B is the server. Both the code and the resources are resident at S_B along with B. When A, which is the client, has to perform a computation, it sends a request to B along with the details of the computation. B, which is the server, performs the computation since both the

code and resources are available locally at site S_B and sends the results back to A at S_A.

2) Remote Evaluation

In this paradigm, A has the required code that needs to perform the computation in S_A and S_B, but it lacks the availability of the resources. The resources are available to B in S_B. A sends the code component to be computed to S_B. Now, B has both the code and the resources available. Thus, it performs the computation and returns the results to A in S_A.

3) Code On Demand

In this paradigm, A has the required resources at S_A but does not have the required code to perform the computation. The code is at S_B where both the code and B reside. A sends a request to B and the code is transferred to site S_A. Since the code and resources are available to A in S_A, A can then complete the computation.

4) Mobile Agent

i) In this paradigm, A is in S_A and has the required code to perform the computation but the resources are present in S_B. A migrates from S_A to S_B transferring the code along with the code component. Now, A is in S_B along with the code and the resources. Therefore, it completes the computation.

ii) A common phenomenon in this paradigm is to now migrate to other sites like S_B where more computations are to be performed and then return to the original site S_A.

2.4 Tradeoffs of Mobile Code Implementations

Advantages of implementing Mobile Code are:

1) The code components and the resources are distributed over the network. Due to the availability of the resources, the efficiency of usage of resources is higher.
2) There is control for the programmer over the choice of a host to where the code is migrated. In contrast, the distributed applications present a non-transparency and spare the programmer with the details of the host from where the information is being received or sent.
3) The resources and code are loosely coupled since they are distributed in different locations. This gives more flexibility for the programmer to modify the code and the resources without affecting each other while doing so.
4) The pattern of interaction of the computational components dictates the architecture of Mobile Code paradigms. The computational components, being very loosely coupled, present possibilities for more architectures than in conventional software systems.
5) On complying with standards, Mobile Code provides a good possibility for computational components that are developed independently to interact and

perform computations. This introduces the possibility for users to access code and resources outside their site of computation.

6) Development time for the software system can also be affected with the use of Mobile Code. The unique architectures of Mobile Code that are based on the loose coupling amongst the computational components also affect the maintenance phase in the software life cycle.

2.5 Domains of Implementation

Some of the domains where Mobile Code can be implemented is in network management [Baldi 1998], electronic commerce [Merz 1996], defense applications [Gray 2002], software maintenance [Binder 2002], maintenance of power systems [Kezunovic 2001], adhoc networks[Qi 2001], distributed information retrieval [Carzaniga 1997].

1) Network Management: Various Mobile Code paradigms have been implemented to perform network management tasks like optimizing the network traffic by collecting load level of network interfaces. Different Mobile Code paradigms have been proven to be better than the others depending on the nature of the management functionality and quantitative characterization of the management protocol and the Mobile Code paradigm used [Baldi 1998].
2) Electronic Commerce: Electronic Market (EM) systems allow both buyers and sellers of supplies and goods in the worldwide communication networks to

communicate based on a set of rules. The embedded agents, along with the Petri Nets, provide value added services and make it extensible to an open distributed platform [Merz 1996].

3) Defense Applications: D'Agents, which are an instance of the Mobile Agent paradigm, have been used as operational support for military personnel in providing efficient, application specific access to remote information resources [Gray 2002].

4) Software Maintenance: An autonomous station which executes dynamically uploaded applications has been developed. Mobile Agent technology is used for distribution of applications to the station, remote maintenance and configuration [Binder 2002].

5) Maintenance of power systems: Mobile Agents are applied in facility maintenance scheduling in deregulation power systems. They are used as the means of communication and coordination between two problem solving entities [Kezunovic 2001].

CHAPTER 3

ELECTRONIC COMMERCE APPLICATIONS

3.0 Overview

Electronic commerce (e-commerce) is an extensive subject in itself. Though the field of e-commerce is currently not of age, there have been exponential developments because of its potential benefits to business and society. The purpose of this chapter is to give an introduction to e-commerce and its current practices. Therefore, the chapter begins with a brief survey of e-commerce where definitions and the evolution of e-commerce are summarized. The different scenarios of e-commerce to conduct business are explained followed by the activities in e-commerce. The various issues in e-commerce are discussed followed by the implemented technologies in e-commerce and the role of Mobile Code in e-commerce.

3.1 Survey of Electronic Commerce

Business is an integral component of society that emerged in the process of civilization. We have gradually grown from trading in barter systems to the use of currency, which is the common denomination paid for the exchange of goods.

Because currency is developed to the local standards, depending on the economic situation of the country, an equality of various currencies had to be found to facilitate trade when it occurred between two countries with different currencies. In the evolution of business, the nature of business has extended from selling goods or objects to selling services.

Technology has brought a consistent rise in the breakdown of barriers like communication and transportation. This breakdown has gradually led to the large shrinkage of geographical and communication limitations in business. This has changed the conduct of business within organizations and with other business organizations. The role of information technology has become inevitable in the field of business. The use of digital technology to digitally enable transactions within or amongst various organizations and individuals is known as e-commerce [Laudon 2002]. It is estimated that trade in e-commerce (in B2B and B2C) will be $12,837.3 billion by 2006 [United Nations 2002].

There is varied opinion on the definition and scope of e-commerce and e-business [Davis 2003]. Some refer to e-business as a broader term that refers to selling, buying, servicing customers and interacting with business partners over the Internet. However, the term 'e-commerce' will be used throughout this document to refer to any digitally enabled transactions conducted by customers or between businesses purchasing goods and services.

Researchers have made an attempt to classify the applications in e-commerce. The applications in e-commerce are driven by the requirements of business, which change dynamically. It is also to be observed that breakthroughs

in technology redefine business processes thus introducing new requirements and, in turn, producing new applications. The classification in e-commerce is therefore currently based on the business scenarios and the activities that occur in the field.

3.2 Business Scenarios in Electronic Commerce

The purpose of e-commerce is to facilitate and aid business and commerce in general [Davis 2003]. Therefore, it is imperative to understand the structure of the businesses, functioning within the organization, and their association with other business organizations. Breakthroughs in technology and widespread networking have also enabled the introduction of new business models. An example of a new business model is the Peer to Peer (P2P) business scenario [White 2001].

The activities in business pertaining to an organization can be classified under 2 major categories:

1) Intra-business activities, i.e. business activities within an organization
2) External business activities, i.e. business activities in collaboration with other business organizations.

The distinction between these activities is gradually blurring due to the changes brought forth by the use of e-commerce technologies. The following is an explanation of the various business models that are currently available in e-commerce:

1) Business to Business (B2B) E-commerce

This is the business model that accounts for the maximum revenue in ecommerce. B2B has accounted for $8296 billion in 2003 [U.S. Dept. of Commerce 2005]. B2B is the type of e-commerce that is used for digital transaction of information, money or anything of value between two individual organizational entities.

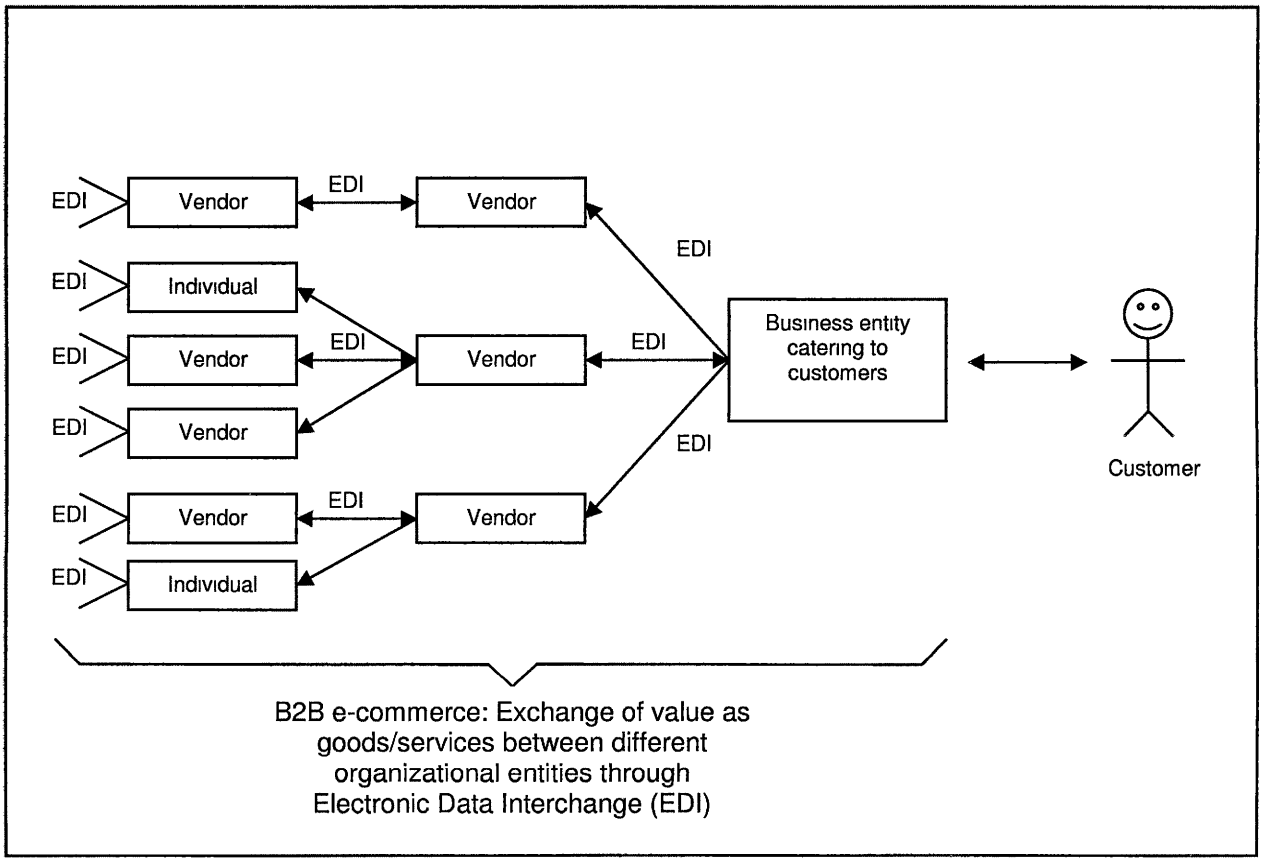

Fig 3.1 B2B e-commerce

A product that reaches the consumers goes through numerous organizational entities and individual processes. Numerous organizational entities or individuals provide the basic raw materials required to begin the

manufacturing or production of a product. However, there is a need to acquire several raw materials to begin the manufacturing process from organizations or individuals known as vendors. Another possible scenario for vendors is to manufacture or produce parts that are requisites for the intermediate stage of manufacturing of a product. These materials or manufactured parts are then purchased by the entity that manufactures or produces the final product. These products are disbursed to distributors or organizational distribution centers who supply it to the retail stores where it is eventually purchased by the-end customer.

This entire process is known as the supply chain process. The objective of e-commerce is to optimize the functioning of the supply chain by transacting information during the exchange of value and money amongst entities at the appropriate time and location.

As an example, Hewlett-Packard Company (HP) has 200,000 product offerings, customers, and global supply chain points involving more than 100,000 suppliers [HP 2004]. The basic components that are needed to manufacture a complete product are supplied by several vendors. The vendors that supply the components to HP may in turn have other vendors supply components to them. After manufacturing a product, HP has retailers and resellers that sell their products to customers. This is an example of a supply chain. E-commerce is deployed for transaction of information to co-ordinate the business activities amongst various business entities like vendors, retailers and HP.

2) Business to Consumer (B2C) E-commerce

Business to Consumer (B2C) e-commerce is a scenario where the business organization sells its products directly to the customers through the use of a website over the Internet.

Examples of similar business organizations are Amazon.com [Amazon 2005] and Half.com [Half 2005]. It is to be observed that companies like Dell also conduct business online in the B2C scenario while the manufacturing process involves a B2B scenario. It is very common for almost all large scale businesses to sell their products online, if possible thus following the B2C scenario.

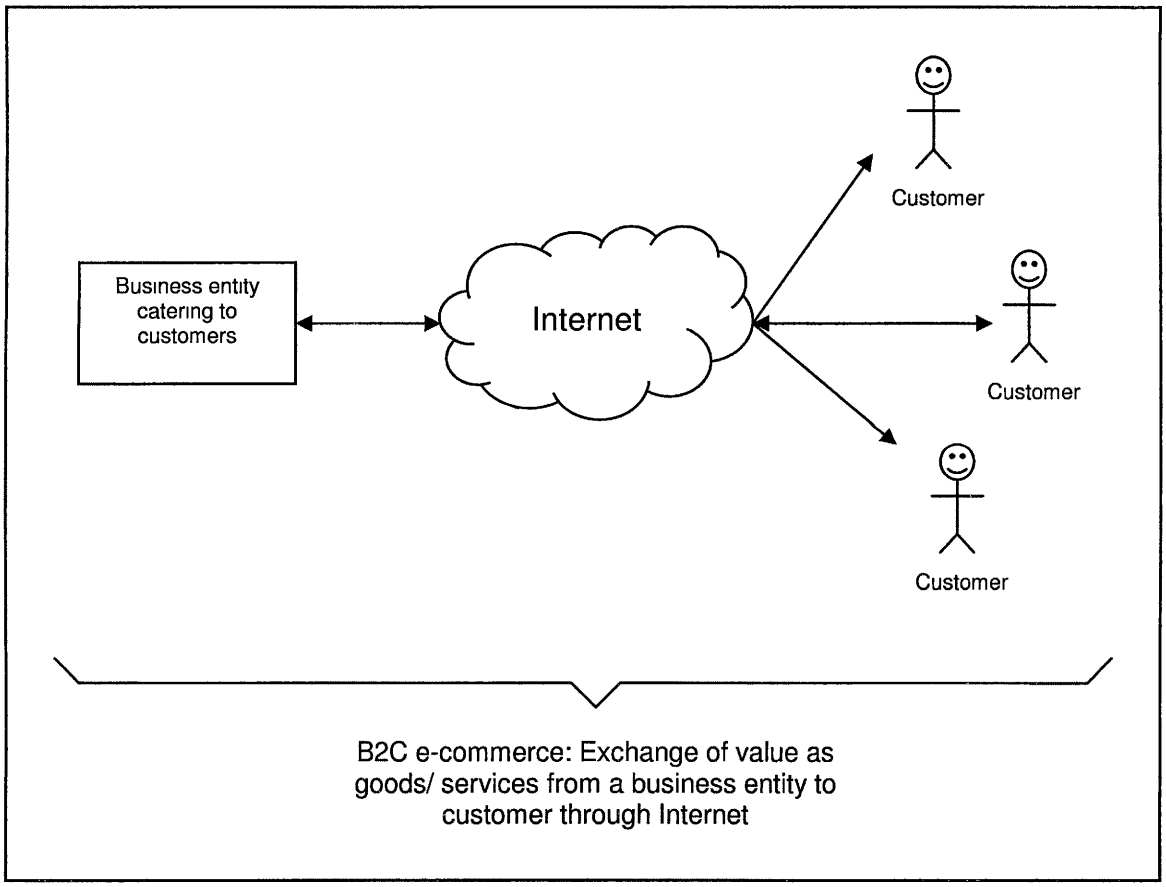

Fig 3.2: B2C e-commerce

As noted earlier, it is quite possible that organizations implement B2C technology for the customers and implement B2B technology in the process of manufacturing the product. And in recent times, both the scenarios have become integrated to optimize the business processes leading to newer business paradigms like just-in-time inventory [Laudon 2002].

B2C was a hyped investment sector in the late 20th century. The revenue that was generated through B2C was $8,352 billion in 2003 [U.S. Dept. of Commerce 2005]. The highest revenue generating scenario of e-commerce is B2B.

3) Customer to Customer (C2C) or Peer to Peer (P2P) E-commerce

Reduction in the cost for computers and accessibility of the Internet to the common man has created possibilities for new business paradigms. One such paradigm is Customer to Customer (C2C) which is inspired by the Peer to Peer (P2P) technology that was originally deployed for information sharing. Often C2C and P2P are used interchangeably.

The Peer to Peer (P2P) concept was initially started by websites such as Napster [Napster 2005] to share information amongst users. This was achieved by sharing information by all the Napster users [Datta 2003]. When a user participated in retrieving information, the user also gave information to other interested users.

This paradigm P2P is being introduced in business scenarios [Datta 2003]. It is especially useful in specific instances such as car part dealers, where

the dealers can share information about each other and trade their goods. C2C or P2P is a business paradigm where both the buyers and sellers are the customers. The organization involved in the business process is neither selling products that it has manufactured nor acting as a selling agent for another organization. The intervention of the organization is only to facilitate the trade of items between the buyers and sellers. All the customers who want to sell or purchase a product participate in the conduct of business and sell to each other. The involvement of an organization is to facilitate this process rather than participate in directly selling goods or services.

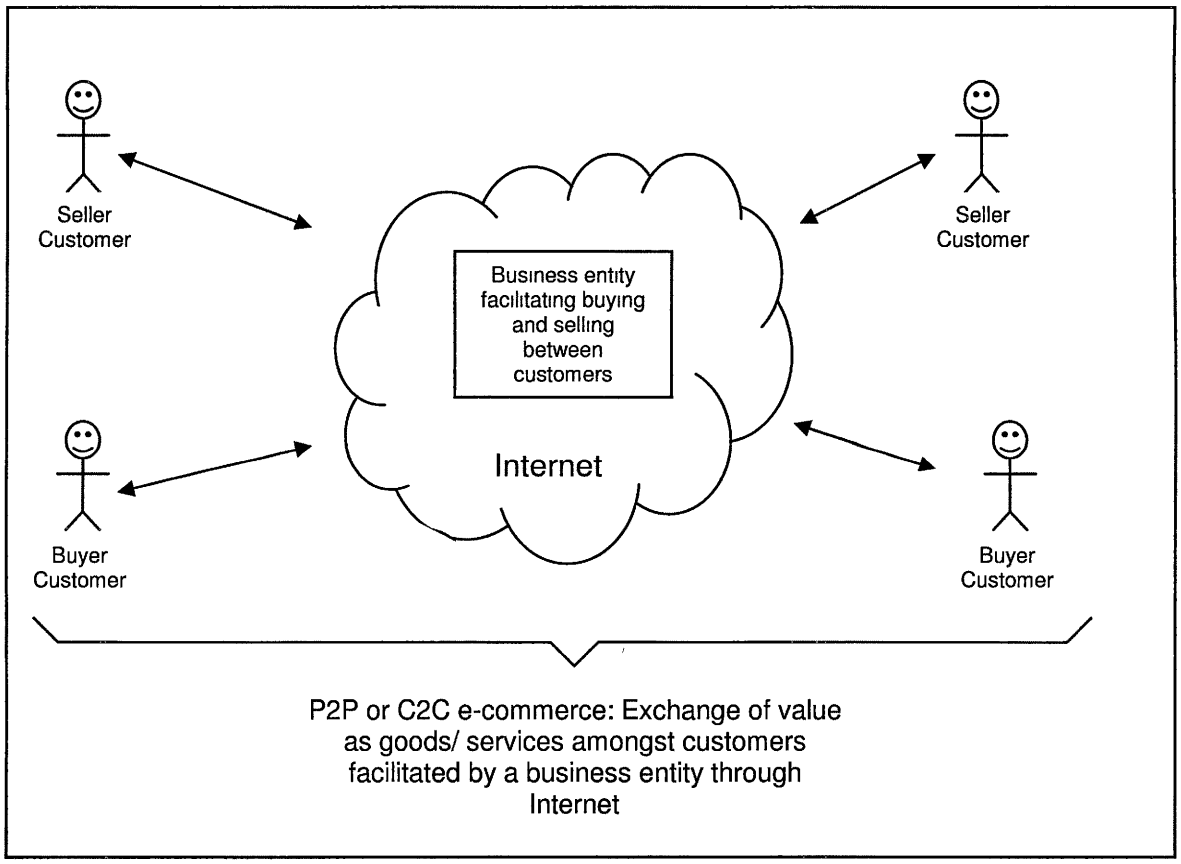

Fig 3.3: P2P or C2C e-commerce

A classic example of this paradigm is eBay [eBay 2005]. eBay has been very successful in implementing this paradigm. It can be observed that there are a range of products that are traded on eBay.

4) Mobile Commerce (m-commerce)

Distributed computing encompasses the underlying concepts that deal with the issues of location and location management. There are some products that are dependent upon location and time. Examples of these are the location of the nearest restaurant of a particular choice, the nearest drug store, etc., at a particular instance of time and place of a customer. These types of services, products or information have a potential to be sold to customers through mobile devices like cell phones, PDA's, etc. [Samaras 2002] This type of commerce is known as m-commerce.

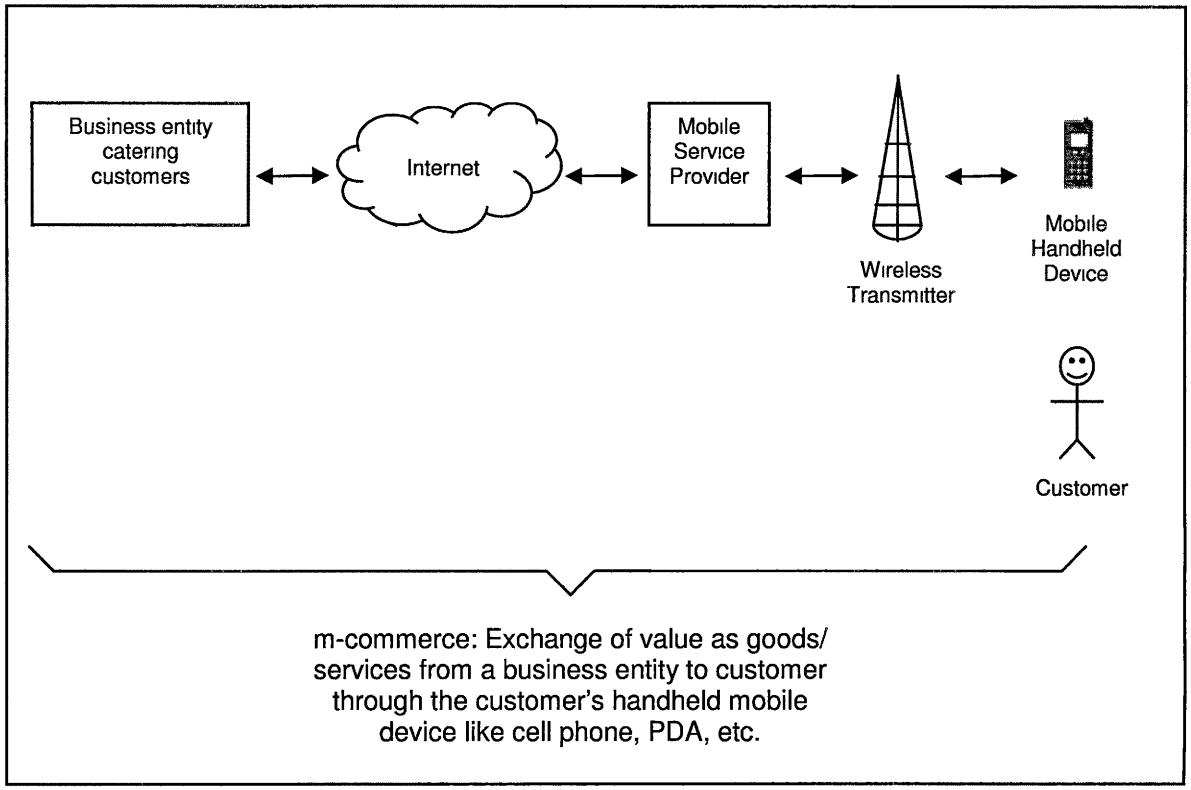

Fig 3.4: m-commerce

5) Enterprise Resource Planning (ERP)

An organization deploys several departments within itself to perform one or more business processes. Examples of these departments are payroll, sales, human resources, etc. The co-ordination amongst all of these departments is essential for the business organization to function smoothly to achieve the objectives of the business. ERP is a system of software that integrates the functioning of internal business processes for the total functioning of the organization [Vogt 2002].

SAP, PeopleSoft and Oracle are the major vendors of ERP software. Initially, the ERP systems were developed to integrate activities within the

business organization. But due to the growing demands and the possibilities to exchange information, a second generation of ERP software known as ERP 2, integrates the processes within the organizations and also the ERP systems of distinct organizations that are associated with the organization in conducting business.

3.3 Activities in Electronic Commerce

The typical commercial activities of any value exchange between the customer/client and the provider are as follows: the customer receiving an offer from the provider, requesting more information about the product, re-evaluating the offer, negotiating terms, ordering the product/service, receiving the bill, making payment, receiving the product and receiving after-sales service. E-commerce can facilitate any or all of these activities for an organization. The digitization of these activities is performed in conjunction with the business strategies of the organization since the objective of digitization is to enhance the operation of business.

A list of the activities of large business organizations is [COMERGENT 2004]

- product catalog
- order management
- portal
- customer service support

- invoicing
- reporting and analysis
- pricing
- parts catalog
- guided selling
- configuration
- replenishment/ Vendor Managed Inventory(VMI)
- quoting
- Product Information Management(PIM)
- distributing orders to channel partners
- returns
- product data synchronization with trading partners
- Partner Relationship Management (PRM)
- services and scheduling
- warranty management

The commercial sectors in which e-commerce is used are manufacturing, merchant wholesale trade, selected service industries and retail trade. B2B e-commerce depends critically on Electronic Data Interchange (EDI) [U.S. Dept. of Commerce 2005]. Examples of the aforementioned commercial sectors that implement e-commerce are:

- Manufacturing and wholesale trade sectors include the following industries[U.S. Census Bureau 2005] as shown in Table 3.1 :

Food	Beverage	Tobacco Products
Textile Products	Leather	Apparel
Wood	Paper	Printing
Petroleum	Chemicals	Plastics And Rubber
Drugs	Non-Metallic Minerals	Metal
Computer Products	Electronic Products	Electrical Equipment
Transportation Equipment	Furniture	Groceries

Table 3.1: Industries in manufacturing and wholesale trade sectors

- Selected services and retails sectors include the following industries[U.S. Census Bureau 2005] as shown in Table 3.2 :

Transportation	Information	Financial Services
Rental And Leasing	Computer System Design	Travel
Accommodation	Food	Health Care
Arts	Entertainment	Repair And Maintenance
Religious	Grant Making	Miscellaneous

Table 3.2: Industries in selected services and retail sectors

- Some more areas of business where e-commerce is used widely [Adam 1999] are shown in Table 3.3.

Banking	Government	Finance
Accounting	Marketing	Customer Service
Human Resources	Research	

Table 3.3: Miscellaneous areas of business where e-commerce is used

Some of the industries overlap due to the presence of the industry in more than one sector.

3.4 Issues in Electronic Commerce

E-commerce has not only introduced new methods of organizing existing businesses but also introduced new paradigms of conducting business with customers. Consequently, there are several issues that have risen to the surface in implementing e-commerce. Some of them are trust, technology, workforce issues, public policy, taxation, business processes, costs and consumer attitudes. Many implementers of e-commerce have expressed confidence about developments to overcome the technical issues. They have come to a strong consensus that user issues will play a major part in the development of e-commerce [COMERGENT 2004].

Some of the technical issues in e-commerce are:

- Decision support systems for e-commerce, which include issues like data warehousing, knowledge discovery in data mining and data warehouses
- Interoperability between applications and connecting multiple object systems that reside in different machines in heterogeneous distributed environments
- Data storage and retrieval
- Workflow technology, mark up languages and their standardization for global implementation
- Security, an issue of the highest concern

- Recent social changes that have raised the concern of trust and security on the Internet

3.5 Implemented Technologies in Electronic Commerce

E-commerce is implemented in a multiple level layered architecture. Though there are several technologies and platforms that are widely used in the market, there is no standardized structure. As a matter of fact, the architecture of an e-commerce application depends on the platform chosen to implement it and the features of the application. The various platforms can be broadly classified as Java with J2EE (Java 2 enterprise Edition and Web Services), ERP, which includes SAP, PeopleSoft, Oracle, etc. and Microsoft with its .Net platform.

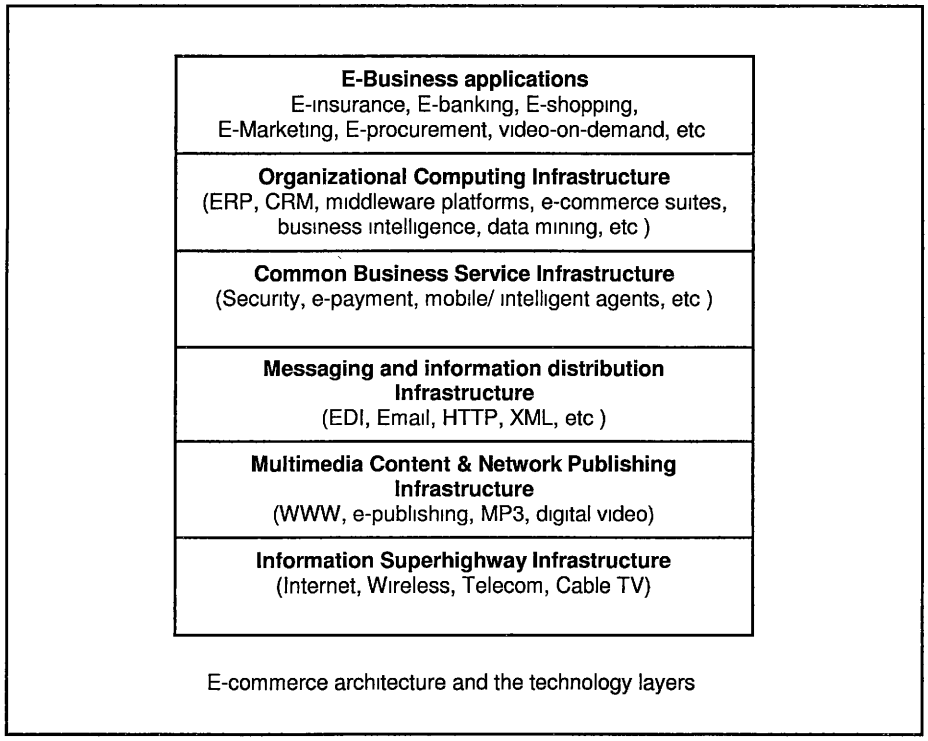

Fig 3.5: Layered technology architecture of e-commerce applications

The bottom most layer in e-commerce is the Internet or the information super highway illustrated below in Fig. 3.5. The multimedia content is published in the content and network infrastructure layer above the Internet layer. The communication is conducted in the messaging and information distribution infrastructure layer, which is comprised of various infrastructure methods to transact information. These include email, Electronic Data Interchange, HTTP, XML, WSDL, etc. Common business services such as security, e-payment, intelligent agents, etc., are present in the common business services infrastructure layer. The organizational computing infrastructure hosts the e-commerce frameworks and applications that are resident in the organization such as ERP, CRM, EAI, middleware platforms, e-commerce suites, business intelligence and data mining. The top most layer consists of the e-business applications such as e-insurance, e-marketing, e-shopping, etc.

3.6 Using Mobile Code in Electronic Commerce

The most common ways to utilize the concept of Mobile Code is in its application as applets, mobile agents and in Common Object Request Broker Architecture (CORBA).

Applets were amongst the early applications during the advent of Java and the Internet. An applet is a program designed to be executed from within another application. While they can be invoked from different applications, they cannot be directly controlled from the operating system. Their small size and

cross platform compatibility makes them ideal for small Internet applications accessible from a browser.

Mobile agents are used for information gathering, facilitating service when acting as a third party entity and for information gathering [Merz 1996]. Mobile Code design paradigms such as remote evaluation and code on demand are also used in the frameworks such as CORBA, Java Remote Method Invocation (RMI), etc.

CHAPTER 4

METHODOLOGY OF APPLYING MOBILE CODE DESIGN PARADIGMS TO ELECTRONIC COMMERCE APPLICATIONS

4.0 Overview

This chapter explains the process and the methodology that has been implemented during the research in applying the concepts of Mobile Code design paradigms in electronic commerce applications. The process began with developing a framework to simulate the e-commerce applications. A specific activity in e-commerce applications was chosen to implement for a specific scenario. The chosen activity was implemented in various Mobile Code paradigms to measure specific tradeoffs.

4.1 Literature Review on Simulation Frameworks

An extensive literature search has been done to discover frameworks that have been developed so that they could be aptly used for this research. The search also considered the methodologies that were used in developing the

researched frameworks. The most relevant results in their respective categories of search are discussed below:

4.1.1 Simulation Framework of Electronic Commerce Applications

Some of the frameworks that were brought to light by the literature review were an e-commerce business simulation called Web-TRECS[Parker 2000], simulation of behavioral models in an event-based modeling language called Rapide [Luckham 2003], a dynamic e-services interaction framework specification called e-speak [Balakrishnan 2000]. According to the researched literature that discussed the frameworks, either the features of the framework did not satisfy the requirements of this research or they exceeded the limitations of the available resources.

4.1.2 Simulation Framework for Implementing Various Mobile Code Design Paradigms

Though there have been several frameworks that were developed for implementing various paradigms of code mobility [Agent List], most of them have focused on implementing the mobile agent paradigm [Picco 1998]. The literature review of several frameworks has led to two viable frameworks for implementation in this research within the limitation of available resources. They are D'Agents [Gray 2002] and μCode [Picco 1998]. Of the two frameworks

considered, μCode was found to be the most compatible framework because of the following reasons [Picco 1998]:

1) μCode provides a uniform platform to implement various Mobile Code design paradigms
2) μCode is lightweight to transfer the code with respect to the overhead introduced due to code transfer
3) μCode satisfies the requirements of the methodology of this research and fits within the limitations of the available resources
4) It is possible to conveniently develop the specific e-commerce application and measure tradeoffs

4.1.3 Simulation Framework for Implementing a Recommender E-commerce Application Using Various Mobile Code Paradigms

Researchers have tried to implement Mobile Code in e-commerce. Examples include the use of open mobile agent systems in electronic markets as buying and selling agents [Bredin 1999] and integration of mobile agents in electronic markets [Merz 1996]. As aforementioned, the literature has been mostly confined to the implementation of the Mobile Code paradigm using mobile agents. Therefore, it was imperative to develop a framework that could simulate e-commerce applications using various Mobile Code paradigms to measure their tradeoffs.

4.2 Simulation Framework Implemented

The vision of an ideal framework for simulation of the Mobile Code design paradigms is presented. The features and the description of the ideal framework follow the vision. The scope and implementation of the ideal framework in the research is discussed.

4.2.1 Vision of the Ideal Simulation Framework

An electronic market includes several activities in various scenarios, as explained in Chapter 3, and mobile code design paradigms can be applied to all of them. The objective of doing so will be to understand the best suited design paradigm for a specific activity by evaluating a set of tradeoffs. The vision for an ideal framework would be to simulate several activities in a business cycle like product advertisement, recommend systems, purchase, feedback, etc. in any of the various e-commerce business scenarios like B2C, B2B, P2P, etc. Since an electronic market usually involves more than 2 scenarios simultaneously, it would be ideal to incorporate the simulation of all activities in all the scenarios simultaneously. By having control points to measure the desired metrics, they can be optimized by using the suitable paradigms. Such a framework can help recognize the most suitable paradigm under the given circumstances yielding an overall optimization of the electronic market.

4.2.2 Features of the Ideal Simulation Framework

Some of the features of a framework mentioned in the vision for an ideal framework in section 4.2.1 would be:

1. Independence from the implementation technology. This could also be called lower level independence.
2. Support for various activities in different scenarios in parallel.
3. Control to choose the paradigms and metric points for each activity in each scenario. This feature could help in trying different combinations of paradigms for activities in different scenarios, and thus help to optimize the required metrics for any or all phases of the business cycle.
4. Controlling the metric points within the activity, between the activities, and in the activities as a whole. It should be observed that there could be considerable overhead introduced by the nature and the number of metric points.
5. A comparative basis of cost to define activities. An example is the difference between a 'payment' activity in a B2C scenario and a B2B scenario. Though the purpose of the transaction is similar in both scenarios, the nature of the activity is different because of the difference in the procedures. A B2B transaction typically involves a more intricate procedure ensuring the appropriateness of the transaction because larger amounts are involved
6. Independence of the framework from the business activity that it simulates
7. Independence and openness to incorporate a Graphical User Interface. This can be called high level independence.

4.2.3 Description of the Ideal Simulation Framework

A simple version of the electronic market in the B2C, B2B and the P2P scenarios involving the customers and the businesses is presented in Figure 4.1. Each of the participants (customer or business) represents a computer host on the network. The framework presented here is an instance of the P2P paradigm being present within the B2B scenario.

The software structure of the simulation framework on each computer host is described in Fig 4.2. The ideal framework for simulation on each host can be described as being made up of several tiers, as shown in Fig 4.2. As explained in the features of the framework in section 4.2.2, the simulation framework tier has upper level and lower level independence. The framework is also independent from any or all of the business activities and the scenarios that are being simulated.

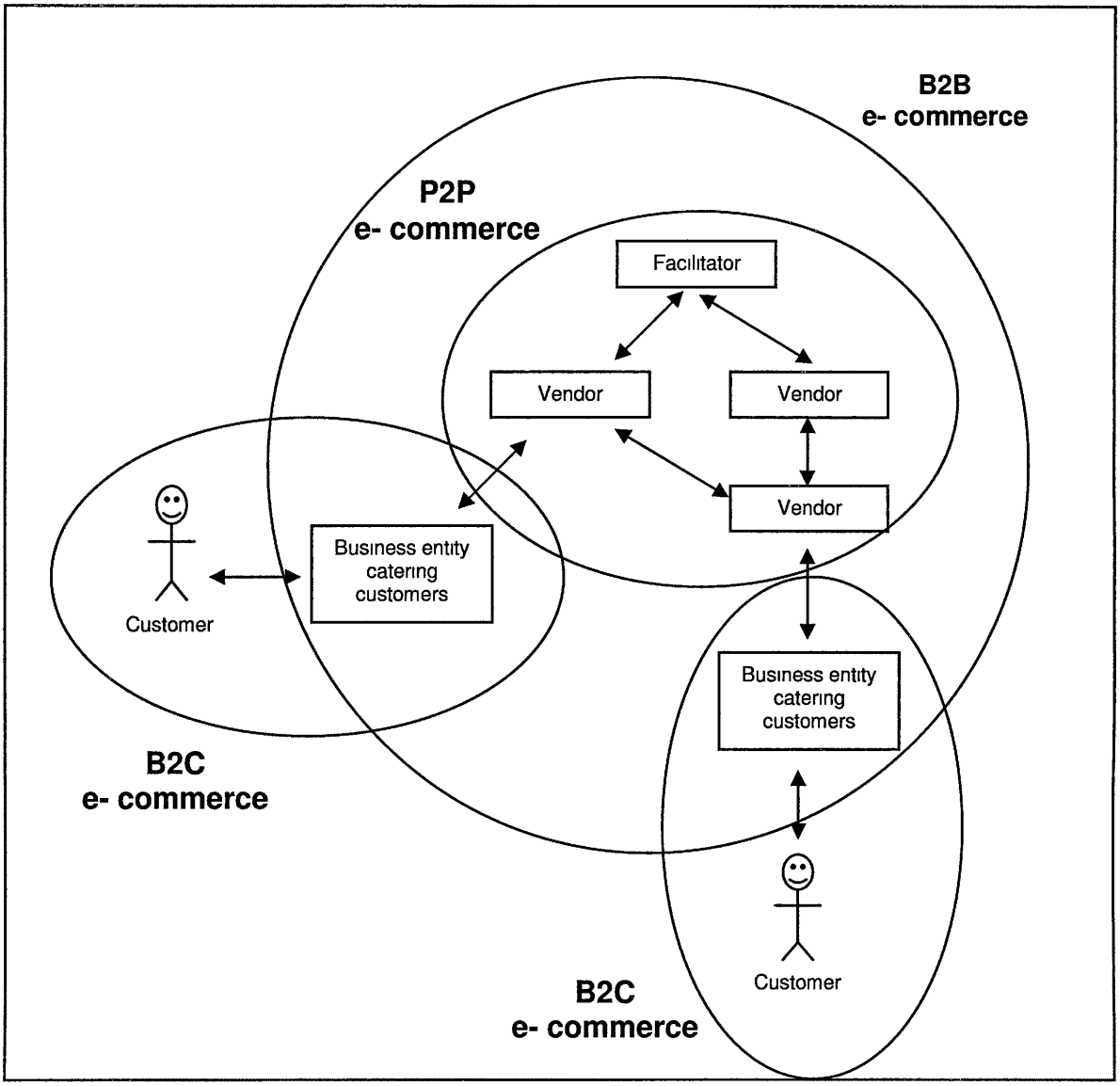

Figure 4.1: Ideal Simulation Framework

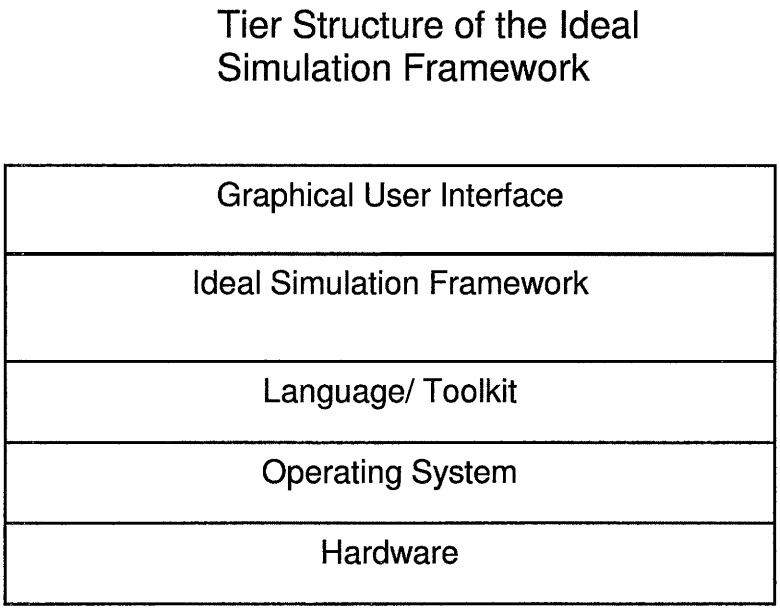

Figure 4.2: Software structure of the computer host

4.2.4 Scope and Implementation of Simulation Framework

Since the vision of a framework that is described in the previous section exceeded the resources available for implementation during research, there were compromises in the incorporation of some features. The scope of the framework that was implemented was restricted to implementation of one activity, namely 'Recommend,' in one e-commerce scenario, namely Business to Customer (B2C).

The concept of the simulation framework that is implemented during this research is an abstract type. More specifically, there is no software or a set of packages that are explicitly written for this framework. Rather, packages from existing toolkits that satisfy the requirements of the research have been used. The framework that is implemented is an idea that is manifested through the programs that have been written and implemented to procure the required

metrics. However, it is a part of the vision to develop packages that could be utilized specifically for this framework.

The resources that were utilized during the experiment are a total of 4 computers with the following configurations:

3 Computers used as Hosts in the Simulation Framework:

Processor information			
Processor vendor	:	Intel	
Model	:	Pentium III (Coppermine)	
CPU M Hz	:	598.077	
Cache size	:	256 KB	
Model	:	8	
Memory Information			
Memory	:	253,996 bytes	
Swap Memory	:	522,104 bytes	
Storage disk information			
Hard disk 1	:	101089 KB	(98.72MB)
Hard drive 2	:	14088992 KB	(13.43 GB)
Operating System	:	Linux Kernel 2.4.20-8	

1 Computer used as a Database Server for Storing the Result Data Sets:

Processor information		
Processor vendor	:	Intel
Model	:	Celeron (Mobile)
CPU M Hz	:	2.40 G Hz

Memory Information

Memory : 496 MB

Storage disk information

Hard disk 1 : 24 GB

Operating System : Microsoft Windows XP, Home Edition, Version 2002, Service Pack 2

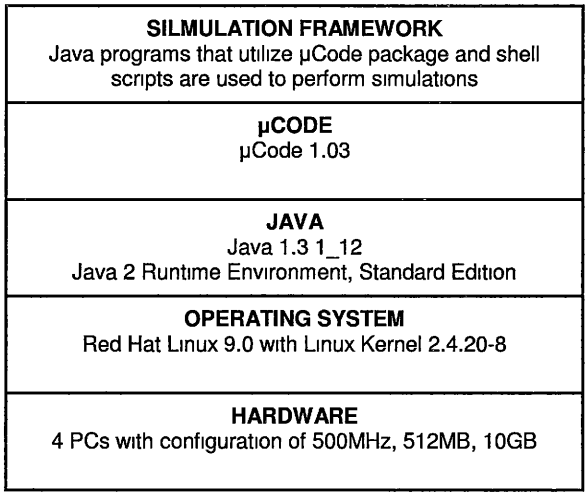

Fig 4.3: System architecture of the Framework used for simulation

Fig 4.3 is a representation of the structure of the computer host that was used for the purpose of the research. The operating system that is installed upon all 3 computer hosts is Red Hat Linux 9.0 with Linux kernel 2.4.20-8. Java 1.3.1_12 is installed on Red Hat Linux 9.0. µCode 1.03, which is essentially a set of packages in Java, is installed and configured on Red Hat Linux 9.0. The framework was implemented on the mentioned software platform through programs that were written in Java 1.3.1_12 utilizing the µCode 1.03 package.

μCode is a set of 3 packages, namely mucode, mucode.util and mucode.abstractions [Picco 2000]. The package mucode contains a minimum set of primitives that are essential to build higher levels of code mobile operations. The package mucode.util contains utilities such as MuServer, which are used on every computer host involved in the experiment. These utilities are run time support systems for code mobility that support the transfer of code. The mucode.abstractions package contains abstractions for Mobile Code that are built on top of the runtime support systems. Mobile Agent is an abstraction to relocate code and state [Picco 2000].

The metrics measured by the framework which are the results for the experiment are stored in a MySQL database server. The configuration of the MySQL database server is MySQL Server version 4.1.12a-nt via TCP/IP. The operating system on which MySQL Server is installed is Microsoft Windows XP 2002 Home Edition.

4.3 Chosen Scenarios and Activities in Electronic Commerce for Implementation

The scenario that is chosen for implementation is the Business-to-Consumer (B2C) scenario in an electronic market. Some of the reasons for considering it among the other scenarios such as B2B, P2P, M2C are as follows:

1. It is the most common scenario in the electronic market. Hence, choosing this scenario for the initial stages of implementation would be reasonable.

2. There is less complexity involved in B2C when compared to other scenarios like B2B.
3. It involves both the business and the customer who are the fundamental actors in most businesses. Therefore, there is a possibility to study the qualitative tradeoffs of both the parties in such a setting.

The chosen activity for implementation during the research is the 'Recommend' activity. In the electronic market, several products of the same genre are sold by numerous businesses to customers. The amount of information that is available to the customer is enormous. When a variety of products are available, it is important for the customer to identify a product that satisfies their criteria. One of the common ways of recommending certain products to a customer is by using recommender systems on websites [Schafer 1999]. Another way for customers to find recommendations is through third party websites like http://www.epinions.com.

The 'Recommend' activity itself, when performed by businesses, could be of 3 types: rule based filtering, content based filtering and collaborative filtering [Tuzhilin 1999 and Schafer 1999]. Since the focus of the research is not on the recommender systems per se, a basic system is presented based on the rule based approach. It is to be observed that the algorithm and the methodology used in the Recommend system is not of significance in the methodology of this research. The same Recommend system is deployed in all the Mobile Code design paradigms as a standard for comparison. The focus of the research is on evaluating the tradeoffs of different paradigms in utilizing this Recommend system.

4.4 Application of Mobile Code Paradigms

The objective in the implementation of the 'Recommend' activity in different Mobile Code design paradigms in the B2C scenario is to measure the proposed metrics. An analysis of the metrics will be performed to evaluate the tradeoffs of the various design paradigms.

In all paradigms, typically, there are two sites of computation – the Customer and the Business. The Recommend activity involves a customer receiving the Recommendation from a business about a specific product that satisfies the customer's requirements. The customer's requirements are mentioned in 'Customer Requirements' (CR). The CR includes the details of the product in search, their priority, the desired value of each detail and the range of the desired value if the exact value is not available. The product information of all the products that are promoted by the business is stored in the 'Business Information' (BI). The Recommend activity is essentially a 'process' that takes the CR and the BI as inputs and generates the recommendations, the output, for the customer. The CR, BI and the Recommend (R) process put together are known as the 'resources.' It is to be noted that the subject of 'design' and 'author' will be mentioned in the forthcoming sections. 'Design of CR' refers to the participant (customer or business) who designed the CR document while 'Author of CR' refers to the participant (customer or business) who filled in the requirements in the CR. A typical example is a Client-Server paradigm where the business designed the CR, but the customer authored it. A few of the variables that play a role in the implementation of the mobile code design paradigms are the initial

location of the resources, the site of computation, the transfer of resources and the interactions involved. The design paradigms will also be extended from the participation of one customer and one business to several customers and several businesses running on more than one machine. The reason for including this in the experiment is that, from a customer's perspective, there are several businesses from whom a customer seeks recommendations and it is possible to include several businesses for a customer in this experiment. From the business perspective, a given business site may have several thousands of customers, which is not possible to simulate in this experiment with the available resources.

The following sections explain in detail the computational components, the site of computation and the nature of interaction between the components for the remote evaluation, code on demand and mobile agent paradigms when applied to the Recommend activity in the B2C scenario in the research with the mentioned available resources. The notations used in the following sections for application of Mobile Code only are:

- business: An entity acting as a supplier in trade
- Business: Site of location of the business entity and its host
- customer: An entity acting as a purchaser in trade
- Customer: Site of location of the customer entity and its host
- CR: Customer Requirements
- BI: Business Information
- Recommend: The recommend process

4.4.1 Remote Evaluation Design Paradigm

In Remote Evaluation, the CR and the Recommend process are both designed and authored by the customer and initially reside on the Customer while the BI is designed and authored by the business and initially resides on the Business as shown in Fig 4.4.

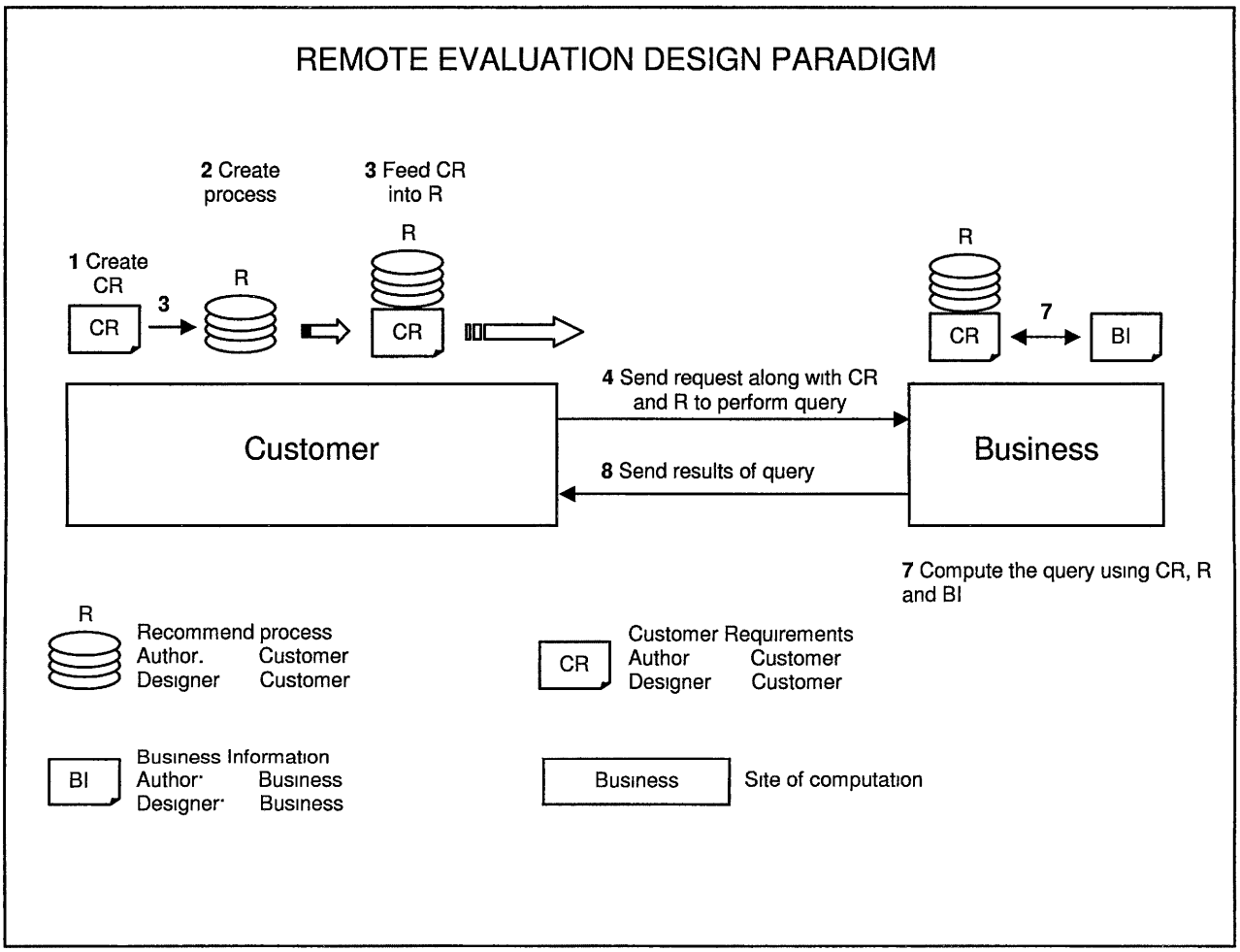

Fig 4.4: Remote Evaluation Design Paradigm

To initiate the process, the customer will input the authored CR into the Recommend process. A request is initiated by the Customer to the Business to begin the Recommend activity and thus sends the Recommend process loaded

with the CR from the Customer site to the Business site. The Recommend process searches the BI and computes the recommendations which are then sent back from the Business site to the Customer site. One of the advantages of using this paradigm is that the customer can use the same Recommend process loaded with CR when he/she desires to perform the activity with several business entities in search of the same product.

4.4.2 Code on Demand Design Paradigm

The CR are designed, authored by the customer and initially reside on the Customer while the BI and the Recommend process are both designed, authored by the business and initially reside on the Business.

To initiate a Recommend activity, the customer authors the CR and sends an 'initiate' request to the Business as shown in Fig 4.5. The Business then inputs the BI into the Recommend process and sends the BI loaded Recommend process to the Customer site from the Business site. The Recommend process then inputs the CR and computes the recommendations.

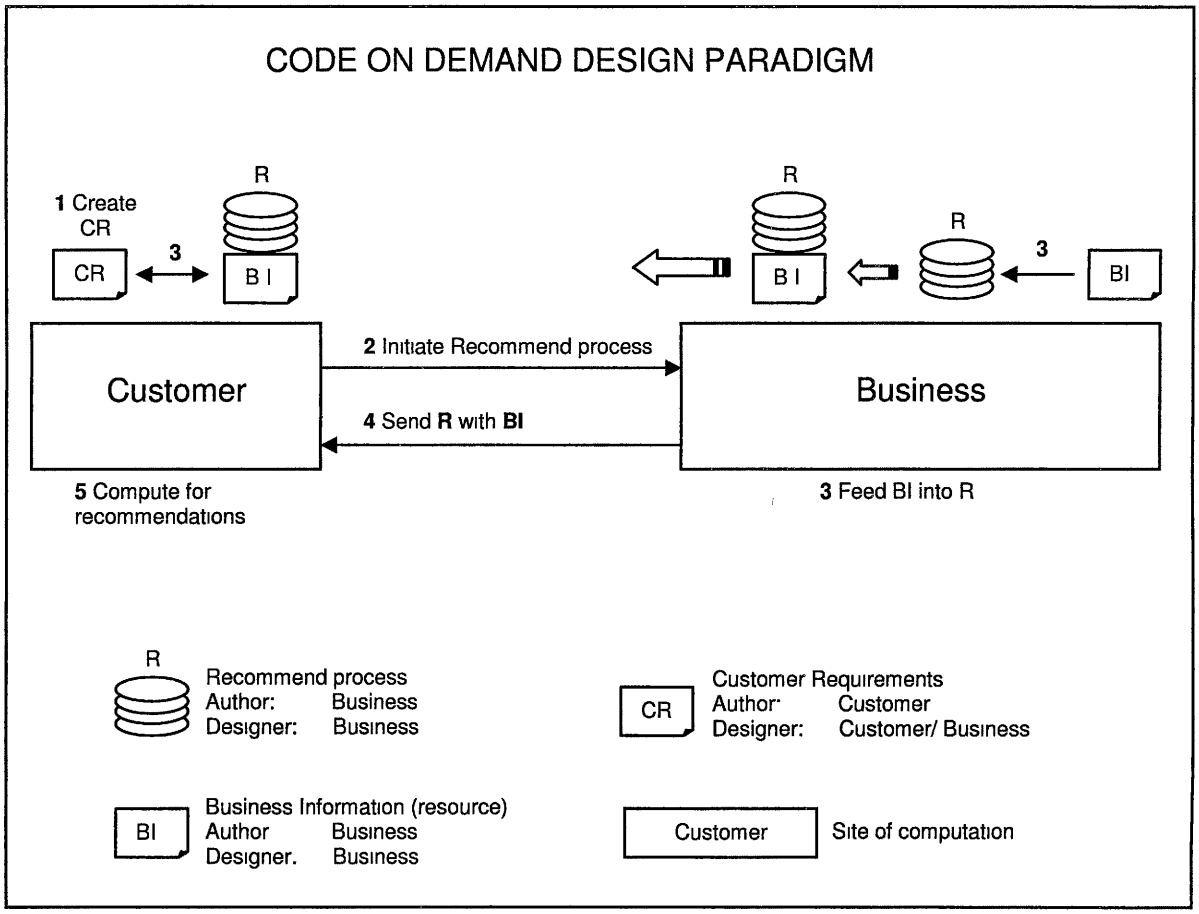

Fig 4.5: Code on Demand Design Paradigm

In a more realistic situation, since the BI will be too large to be loaded into the Recommend process, only the Recommend process is migrated to the Customer site. Then, upon input of the CR to the Recommend process and processing it, a request is sent to the BI database to send the required BI to the Customer site for further computations of the Recommend process. An advantage of this paradigm is that the customer can reuse the authored CR for other businesses when searching for the same product.

4.4.3 Mobile Agent Design Paradigm

In the Mobile Agent design paradigm, the CR and Recommend process are designed, authored and initially reside on the Customer site. The BI is authored, designed by the business and initially resides on the Business site.

When the customer desires to initiate a Recommend activity, they author the CR and input it into the Recommend process along with the address(es) of the Business(es) that the customer intends to receive recommendations from. The Customer sends a request to initiate the Recommend process along with the loaded Recommend process (CR and R) to one of the Businesses. The Recommend process then searches the BI, processes the recommendations and stores the results in the agent. The agent transfers to the other Businesses and continues the process until all the Businesses enlisted are completed. During the process, the agent either stores all the results or only a few of the best results based on the design. Then, the agent moves to the customer and presents the results that it has stored. An advantage of this paradigm is that the customer does not have to perform the initiation of the Recommend process repeatedly on every business that they intend to investigate.

In the implementation of this paradigm in the research experiment, the results from every Business have been appended to the existing results in the Mobile Agent. The motivation for this was to measure the latency of the query response when the Mobile Agent is carrying increasing amounts of data as it travels from one business to the other.

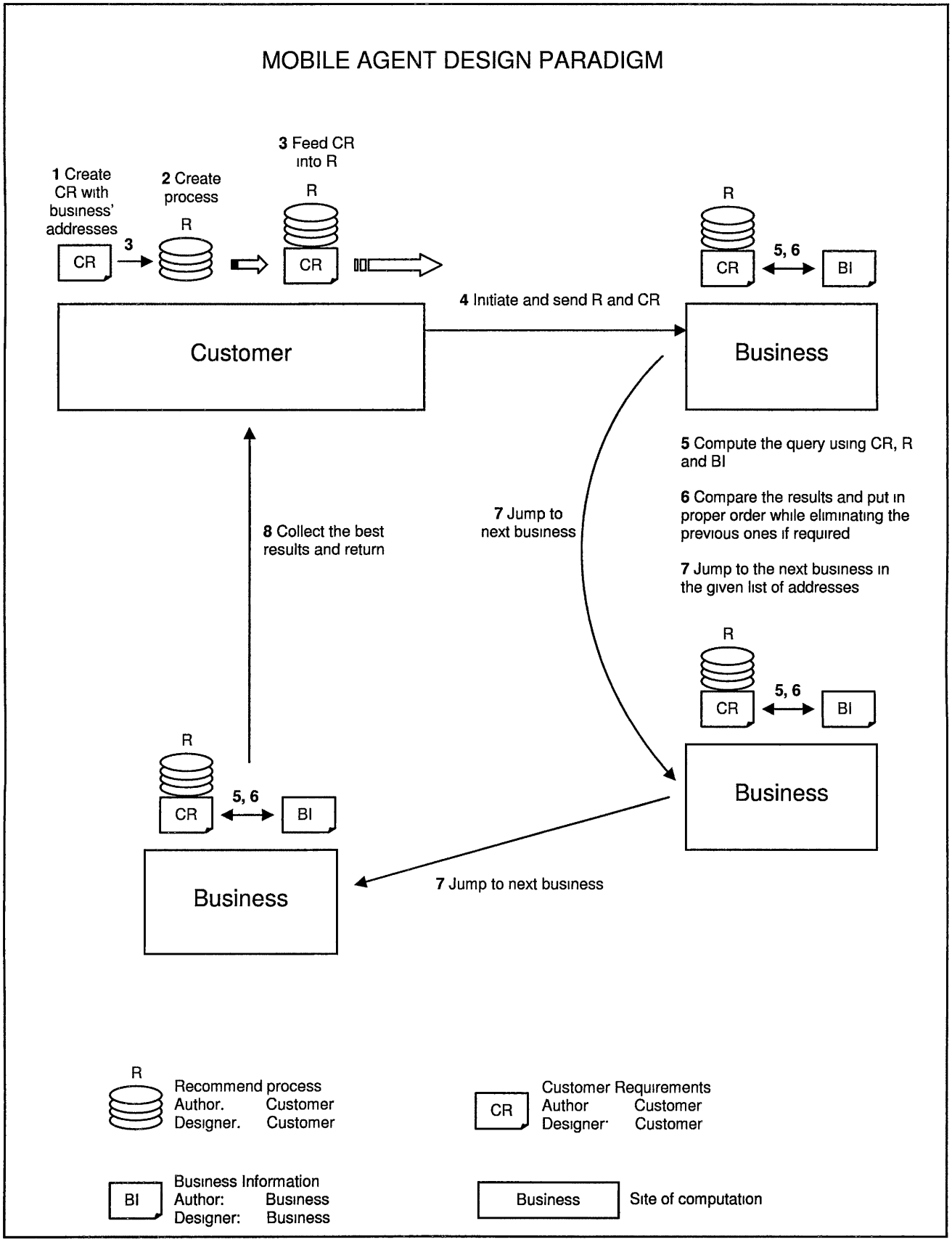

Fig 4.6: Mobile Agent Design Paradigm

4.5 Limitations of the Implemented Framework

The limitations of the simulation framework that is developed and implemented in this research experiment in comparison to the features of the Ideal Framework for Simulation are as follows:

1. The framework is restricted to only one activity (Recommend)
2. The framework implements only B2C e-commerce trading scenario. It can be modified to simulate other specific scenario in e-commerce, namely B2B, P2P, etc., by incorporating their relative parameters like ratio of customers to businesses, network load, etc.
3. Currently, there is only one metric that can be measured, namely latency of query response.
4. There is no Graphical User Interface (GUI) developed for this framework in this experiment.
5. In most e-commerce scenarios, typically, the computational capacity of the business hosts (Business) is higher than the computational capacity of the customer host (Customer). This is due to the high volume of requests that Businesses receive from the Customers. This factor is not incorporated in the current simulation with the available resources.

4.6 Tradeoffs Selected for Evaluation of Mobile Code Design Paradigms

4.6.1 Latency of Response to the Query

Definition of the Metric

The time delay between the instance when a customer sends a request to the business and the instance when the results are received by the customer is known as latency of query response. After the customer sends a query request to the business, there is an appropriate transfer of Mobile Code as per the design paradigm. The calculation of the query cannot be executed unless all the 3 entities of code, namely Customer Request (CR), Recommend process (R) and Business Information (BI), are available at one host. Therefore, it is postulated that there is a definite delay in the query response time which becomes an important factor to be measurable for this research.

Methodology of Measurement

The MobileCodeGroup class carrying the object and the class objects from a Customer to the Business in the experiment. During the execution of the program, timestamps have been noted using the currentTimeMillis() function. The desired values for the metrics can be calculated with the difference in the values of the corresponding timestamps. Description of the java.lang System.currentTimeMillis() function from Sun Microsystems is as follows:

java.lang
Class System

currentTimeMillis

`public static long` **`currentTimeMillis`**`()`
Returns the current time in milliseconds. Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class `Date` for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

Returns:
the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC. [Sun 2004]

Figure 4.7 shows the location and transfer of Mobile Code and the instances of timestamps in the REV and COD paradigms. In the case of the Mobile Agent (MA) paradigm, the agent visits all the Businesses before it comes to the Customer.

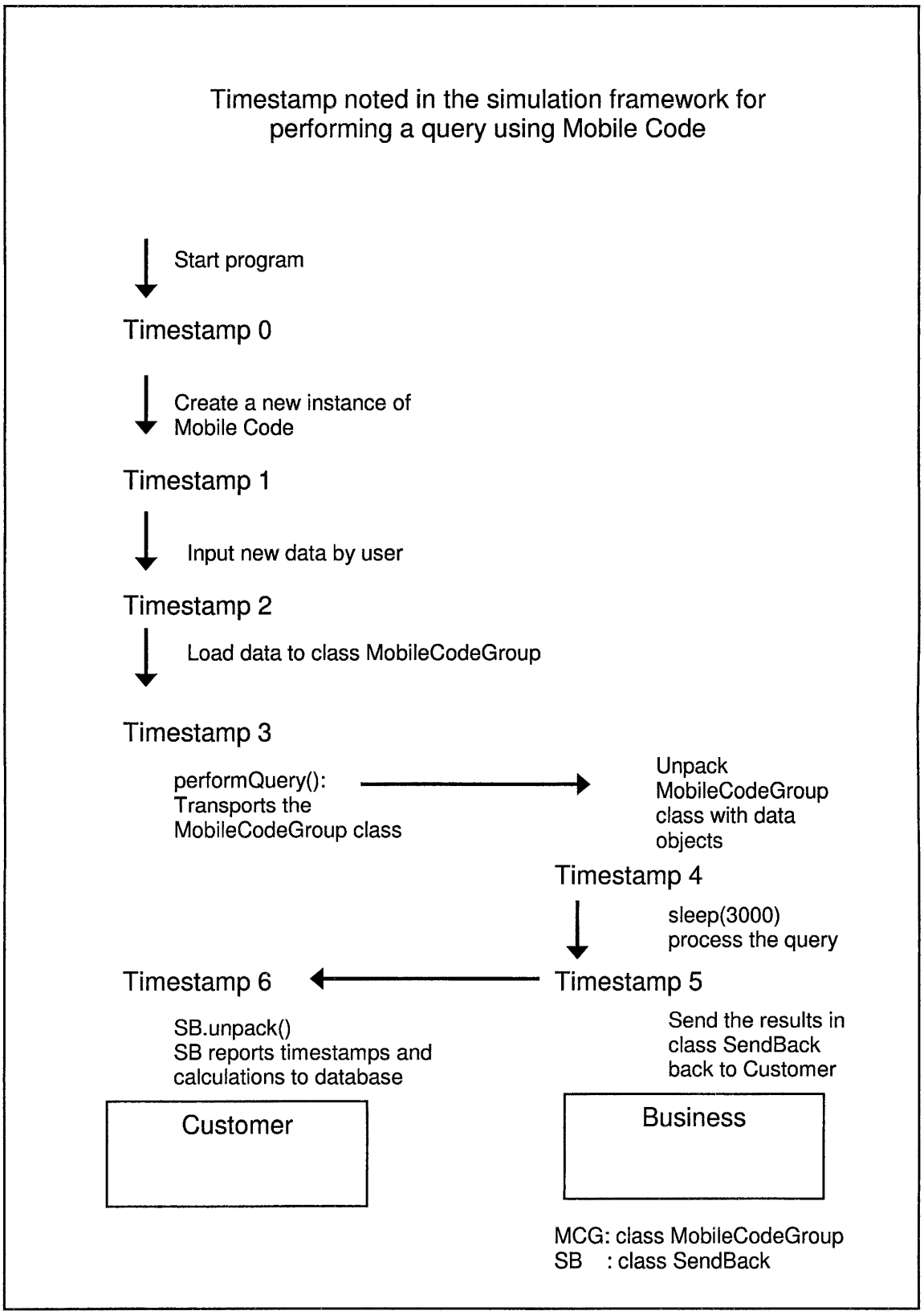

Fig 4.7: Timestamp noted in the simulation framework for performing a query using Mobile Code

The MA paradigm is incorporated into MCG by programming MCG to travel to all the Businesses before returning to the Customer. In the Figure 4.7, it is to be noted that the MobileCodeGroup class, after unpacking, is put to sleep for 3000 milliseconds. Hence 3000 is subtracted from the timing calculations of TimeCalcs[0], TimeCalcs[1] and TimeCalcs[4]. The calculations that are performed and stored in the database are as follows:

TimeCalcs [0]: Time taken to perform query. This is also the latency in response to the query when a request is sent by the Customer to the Business and the results are received by Customer. This is the primary metric that is measured during this research experiment. It is to be noted that 3000 is the time that the Mobile Code Thread sleeps at the Business before traveling back to Customer.

TimeCalcs[0] = Timestamp6 – Timestamp3 – 3000

TimeCalcs[1] : Overall time taken by the experiment from creating a new instance of the Mobile Code until the Customer receives the results of the query. It is to be noted that 3000 is the time that the Mobile Code Thread sleeps at the Business before traveling back to Customer.

TimeCalcs[1] = Timestamp6 – Timestamp0 - 3000

TimeCalcs[2] : Input new data by the user into class MobileCodeGroup (MCG) in some paradigms like REV and MA. The data objects and class objects are stored in MCG and transported from Customer to Business.

TimeCalcs[2] = Timestamp2 – Timestamp1

TimeCalcs[3] : Load data objects and classes required to perform the query in Mobile Code.

TimeCalcs[3] = Timestamp3 – Timestamp2

TimeCalcs[4] :

REV and COD paradigms: Time taken by MCG to travel from Business to Customer and unpack the data and class objects in MCG.

TimeCalcs[4] = Timestamp4 – Timestamp3

MA paradigm: Time taken by the MCG to travel from the Customer to the current Business after visiting other Businesses. It is to be noted that 3000 is the time that the Mobile Code Thread sleeps at the Business before traveling back to Customer.

TimeCalcs[4] = Timestamp4 – Timestamp3 – ((no. of hops – 1) * 3000)

no. of hops: The number of Businesses visited prior to visiting the current business.

TimeCalcs[5] :

REV and COD paradigm: Time taken by MCG to travel from Business to Customer, unpack the data and class objects in Send Back (SB), unpack the data and class objects. The functionality of SB is similar to MCG. SB transports the results in a data object from Business to Customer.

TimeCalcs[5] = Timestamp6 – Timestamp5

MA paradigm: Time taken by MCG to travel to the current business from the previous business.

If current business destination is the first business destination

TimeCalcs[5] = Timestamp4 – Timestamp3

Else

TimeCalcs[5] = Timestamp4 – Timestamp5

The metric will be measured based on the 3 classes of measurements during implementation. There are 3 hosts that are used in the experiment

1) 1 Customer – 1 Business per host
2) n Customer – 1 Business per 3 hosts
3) n Customer – m Business per 3 hosts

Range of values:

n ϵ { 3, 5, 10, 15, 20}

m ϵ { 3, 5, 10, 15, 20}

For each of these classes, a query response is performed between the Client(s) and the Business(es) for every Mobile Code design paradigm. The results obtained from this experiment are presented in Chapter 5: Results and Analysis.

CHAPTER 5

RESULTS AND ANALYSIS

5.0 Overview

The results of the experiment conducted during the research and the relevant analysis are presented in this chapter. The quantitative metrics are discussed followed by the qualitative analysis of the experiment. The trade-offs of the various Mobile Code design paradigms implemented in the experiment are discussed. Inferences about e-commerce scenarios that are not implemented in this experiment are also presented. An abstraction of all the design paradigms concludes the chapter.

5.1 Quantitative Metrics

5.1.1 Data Sets of Input Parameters and Output Results

The framework implemented in the experiment is explained in section 4.2.4. The variable parameters that could potentially produce a variation in the values of the metrics in the implemented framework have been recognized and they are:

1) Mobile Code design paradigm

2) Number of Businesses

3) Number of Customers

4) No of hosts participating in the simulation framework

Various combinations of these parameters produced data sets of values for input variables for every Mobile Code design paradigm in the experiment. The relevant data sets of the parameters that are most suitable for the simulation of B2C e-commerce scenario have been isolated. This motivated choosing data sets so that the number of customers is always less than the number of businesses. The implemented data sets for each Remote Evaluation (REV), Code on Demand (COD) and Mobile Agent (MA) are presented in Table 5.1. Data sets with the number of hosts equal to 3 have the number of business or customers equal to 3 when the corresponding value in the data sets with number of hosts equal to 1 is equal to 1. Data sets have been designed to accommodate at least 1 Business or Customer per host. A total of 96 data sets have been implemented in the experiment.

data sets with hosts = 1			data sets with hosts = 3		
no. of business	no. of customers	no. of hosts	no. of business	no. of customers	no. of hosts
1	1	1	3	3	3
5	5	1	5	5	3
1	5	1	3	5	3
5	1	1	5	3	3
10	10	1	10	10	3
1	10	1	3	10	3
10	1	1	10	3	3
15	15	1	15	15	3
1	15	1	3	15	3
15	1	1	15	3	3
20	20	1	20	20	3
1	20	1	3	20	3
20	1	1	20	3	3
5	20	1	5	20	3
10	20	1	10	20	3
15	20	1	15	20	3

Table 5.1: Data sets of all implemented input parameters

The data that is collected in the process of the experiment lead to inferences about

1) Latency in the query response
2) Time taken to load data into Mobile Code
3) Overall time taken for the experiment

4) Reliability or the percentage of the response for the query

5) Travel time To or From the Customer to the Business

5.1.2 Latency of Response to the Query

Latency of response to the query has been the primary metric of measurement for this research experiment as explained in Section 4.6.1.2. The Recommend process that is sent or received by the Customer to or from the Business is considered to be the query with the Business Information (BI) of 25 records residing in the Business. The results obtained for the B2C paradigm for this metric are from the data sets presented in Table 5.2.

data sets with hosts = 1			data sets with hosts = 3		
no. of business	no. of customers	hosts	no. of business	no. of customers	hosts
1	1	1	3	3	3
1	5	1	3	5	3
1	10	1	3	10	3
1	15	1	3	15	3
1	20	1	3	20	3
5	20	1	5	20	3
10	20	1	10	20	3
15	20	1	15	20	3
20	20	1	20	20	3

Table 5.2 Data sets of implemented input parameters for B2C paradigm

In most cases, the Mobile Agent takes the highest amount of latent time in query response. The reason for this phenomenon is that the Mobile Agent travels

to all the Businesses to gather the data before it returns to the Customer with the query results.

The latency of query response metric has been calculated for 4 data sets. They are presented in the following sections:

1. Latency of Query Response with Increasing Number of Customers and 1 Business per Host

The results produced from this data set are with 1 Business and an increase in the number of Customers from 1 to 20 as indicated in Table 5.3.

data sets with hosts = 1			data sets with hosts = 3		
no. of business	no. of customers	hosts	no. of business	no. of customers	hosts
1	1	1	3	3	3
1	5	1	3	5	3
1	10	1	3	10	3
1	15	1	3	15	3
1	20	1	3	20	3

Table 5.3: Data set 1 of implemented input parameters to measure latency of query response

The graph in Figure 5.1 consists of the averages of the delay in the query response in milliseconds on the y-axis to the number of Customers within the range of {1, 5, 10, 15, 20} on the x-axis with 2 sets involving 1 host and 3 hosts.

It can be observed from the graph in Figure 5.1 that the most time taken is by the Mobile Agent paradigm for 3 hosts, 150216.83 milliseconds. Though there is an increase in the computational capacity when using 3 hosts in the network of the simulation framework, the time taken by the Mobile Agent paradigm performed on 1 host is 96486.1milliseconds, 35.76% less than the time taken by the same paradigm for I host.

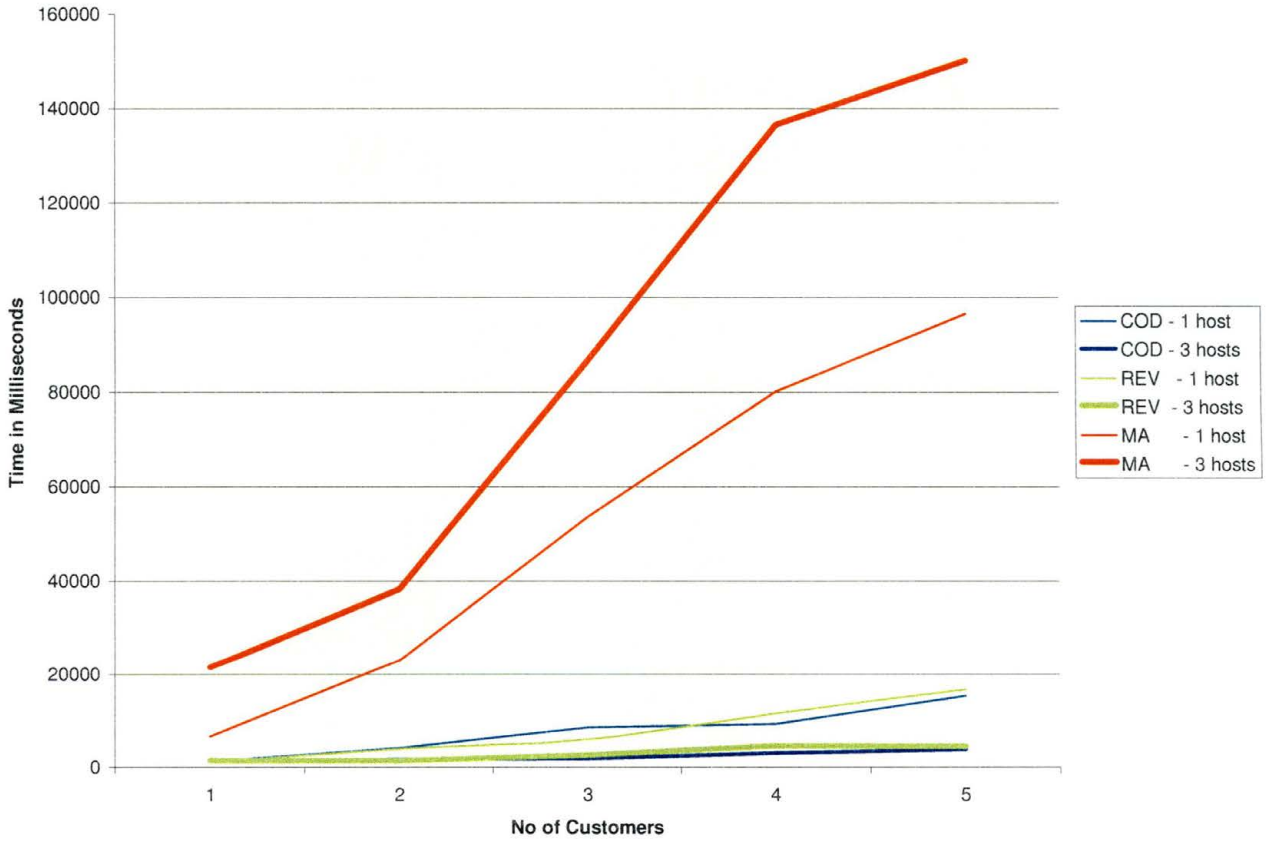

Figure 5.1: Latency of query response with increasing number of Customers and 1 Business per host

It can be inferred that the increase in the amount of time taken by the Mobile Agent for 3 hosts is due to the delay caused by the transportation of data

through the network by the Mobile Agent. However, in the COD paradigm, the increase in the number of hosts in the network causes a significant drop of 75.40% in the delay in the response time from 15340.80 milliseconds for 1 host to 3773.26 milliseconds for 3 hosts. This is due to an increase in the net computational power of the framework. The REV does have the same effect to reduce by 73.17% when there is an increase in the computational capacity by an increase in the number of hosts in the network. The effect of network load is accompanied by the increase in the computational capacity of the simulation framework.

2. Latency of Query Response with Increasing Number of Businesses and 20 Customers per Host

The data set of experiment parameters that was used in this set of experiments is presented in Table 5.4. The number of Customers in this scenario is kept constant at 20 while the number of Businesses is ascending in number.

data sets with host = 1			data sets with hosts = 3		
no. of business	no. of customers	hosts	no. of business	no. of customers	hosts
1	20	1	3	20	3
5	20	1	5	20	3
10	20	1	10	20	3
15	20	1	15	20	3
20	20	1	20	20	3

Table 5.4: Data sets of input parameters for measuring latency of query response with increasing number of Businesses and 20 customers per host

A comparison of the results obtained is presented in Fig 5.2 with the averages of the delay in the query response in milliseconds on the y-axis to the number of businesses within the range of {1, 5, 10, 15, 20} with 1 host and 3 hosts on the x-axis.

There is not a significant difference in the delay in response time for the Mobile Agent paradigm between 1 host and 3 hosts, though it is higher for 3 hosts for lower number of Customers and equal at the point for 5 Businesses. There is a marginal decrease in the delay in response to the query from 1 host to 3 hosts beyond the point of 5 Businesses. A noticeable change in the REV paradigm is that the delay time drops for 20 Businesses from 217424.77 milliseconds for 1 host to 20263.07 milliseconds for 3 hosts by 90.68%. The delay time also surges from the point of 15 Businesses to 20 Businesses by 227%. A higher decrease is observed in the COD paradigm with an increase in the number of Businesses from 5 Businesses to 15 Businesses.

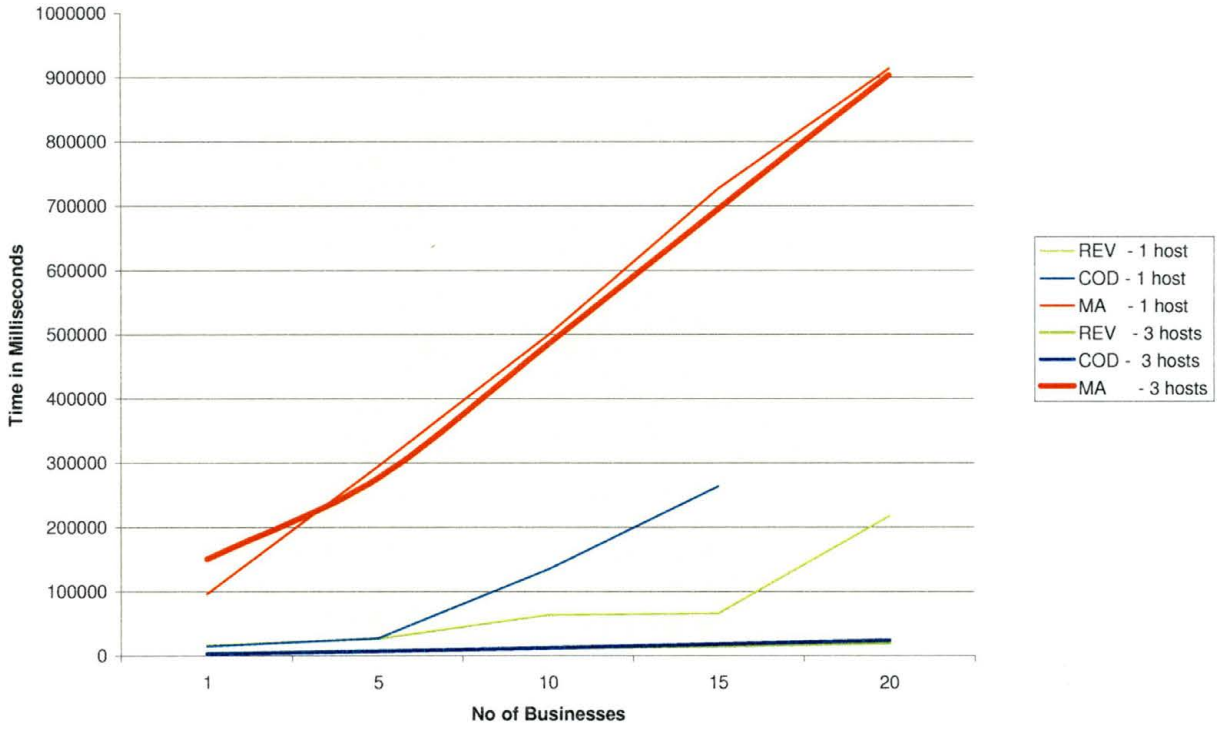

Figure 5.2: Latency of query response with increasing number of Businesses and 20 customers per host

This indicates that an increase in the net computational capacity of the REV and COD paradigms decreases the delay in the query response time and remains little affected by the increase in the number of the participating entities within the range of the experiment.

3. Travel time of Mobile Code to the Business with Increasing Number of Customers and 1 Business per Host

The data for this experiment is located in Table. 5.3. Since one instance of Mobile Code in MA has to visit all the businesses while an instance of Mobile Code in REV and COD paradigms travels only to one business, the travel time for the Mobile Code in MA has been averaged with the number of Businesses that it visits.

The graph in Figure 5.3 indicates that on average the MA paradigm takes more time to travel to the Businesses. However, it varies only by 27.7% for 3 hosts and 38.06% for 1 host due to the increase in the no. of customers. On average, it increases approximately by 48.34% when there is an increase in the net computational power while the other paradigms show a significant decrease. This is possible because of an increase in the network traffic that is introduced when the mobile agents in MA paradigm have to travel to different hosts due to their distribution with an increase in the number of hosts. The travel time is also high in MA paradigm for 1 host when there are less number of Businesses and Customers. This is because the time taken by the Mobile Code across the network with the accompanying data is less.

The REV paradigm increases by an average of 1050 milliseconds for every new Customer after 10 Customers. However, it reduces by 80% from 1 host to 3 hosts for 20 Customers. This is the observed highest decrease among all the paradigms in this case. The COD paradigm appears to be the lowest when

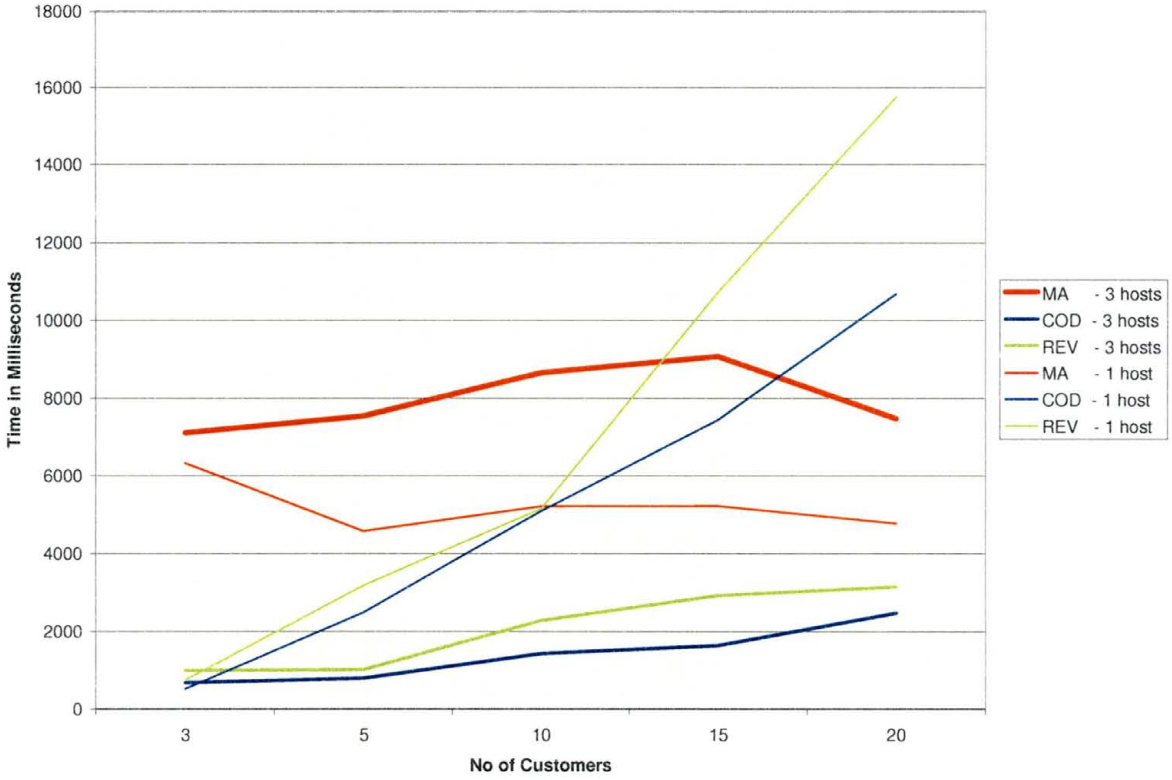

Figure 5.3: Travel time of Mobile Code to the Business with increasing number of Customers and 1 Business per host

there is an increase in the number of hosts in the network thereby increasing the computational potential. COD also has the lowest travel time up to 10 Customers after which it increases constantly with increase in the number of Customer hosts. It is inferred that the travel time is the least for COD in most cases unless the computation power is less and there are more participating entities in the network.

4. Travel time of Mobile Code to the Business with Increasing Number of Businesses and 20 Customers per Host

The data set used for this set of experiments is in Table 5.4. Since the Mobile Code in MA has to visit all the businesses while the Mobile Code in REV and COD paradigms travel only to one business, the travel time for the Mobile Code in MA has been averaged with the number of Businesses that it visits.

An analysis of the graph in Figure 5.4 indicates that with the least computational power and fewest number of customers, the MA paradigm

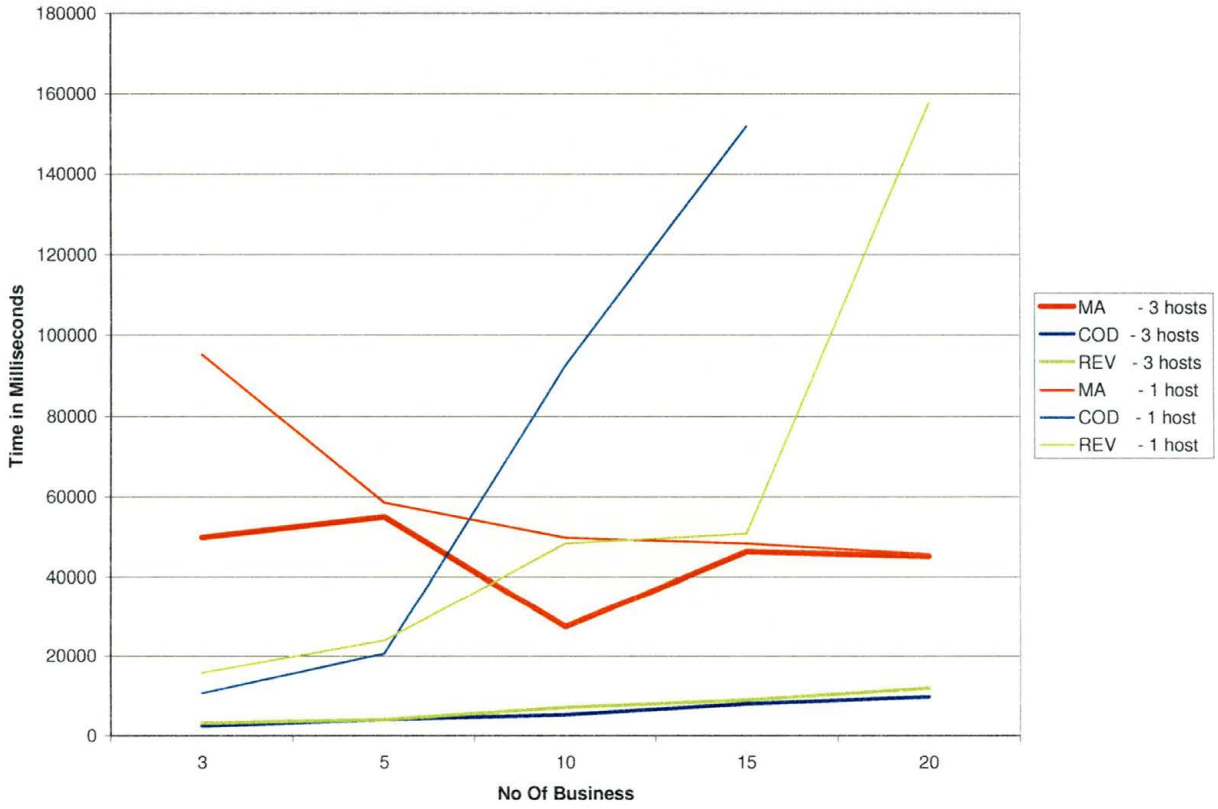

Figure 5.4: Travel time of Mobile Code to the Business with increasing number of Businesses and 20 Customers per host

takes the most time takes the most time to travel. But with the same computational power, i.e. 1 host, the travel time decreases from 95407.15 milliseconds for 3 customers to 45621.98 milliseconds for 20 customers by 52.18%. However, the REV and the COD paradigms continue to rise as expected under the same circumstances they exhibited in previous cases. The REV and COD paradigms are consistent and low in the time they take to travel when the source of computation is 3 hosts. In comparison to their travel time to 20 Businesses with only 1 host as their source of computation, there is a huge decrease when the number of Businesses to be visited is 20 with 1 host. It has decreased by 92% in the case of REV and 93% in COD.

5. Effect of Changes in Computational Power on Total Average Times in the Mobile Code Paradigms

The data about the values of timestamps collected during the experiments is used to perform various timing calculations. The definitions of the timing calculations are presented below:

(1) Latency in query response: Delay in response to the query

(2) Total time for Customer: Overall time taken for the Mobile Code to travel from the Customer to the Business and back

(3) Input New Data: Time taken to input new data by the user at Customer host.

(4) Load Data to Mobile Code: Time taken to load data to Mobile Code

(5) Travel Time To Business: Time taken to travel to Business from Customer. In the case of MA paradigm, it is the sum total of the time taken to travel by the agent from the Customer to all the Businesses and back to the Customer.

(6) Travel Time From Business: Time taken to travel back from Business to Customer. In the case of MA paradigm, it is the time taken to travel by the mobile agent from one Business to the other.

The average of all the timing calculations for 1 host and 3 hosts for REV, COD and MA paradigms using the data sets presented in Table 5.1 is calculated and presented in the following graphs. Upon comparison of the graphs in Figure 5.4, Figure 5.5 and Figure 5.6, it is observed that the COD paradigm has the lowest values for all average time measurements when there are 3 hosts. And MA paradigm has the lowest time delay values when there is 1 host.

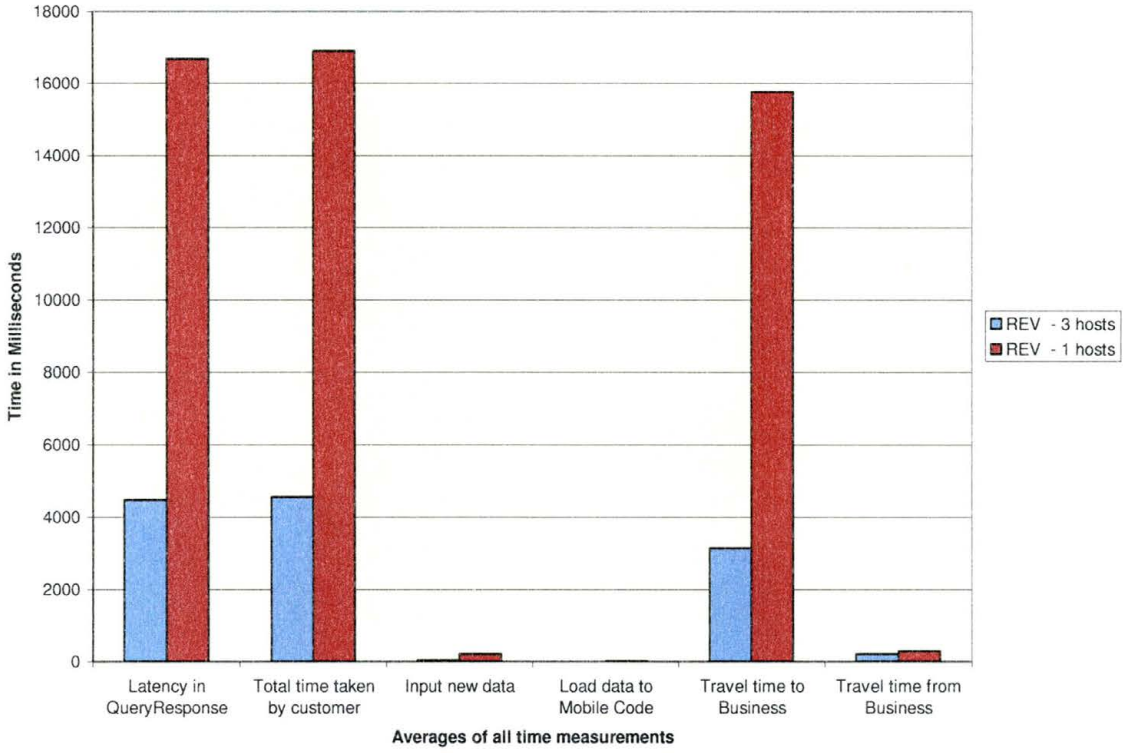

Figure 5.5: Changes in average times by increase in number of hosts in the REV paradigm

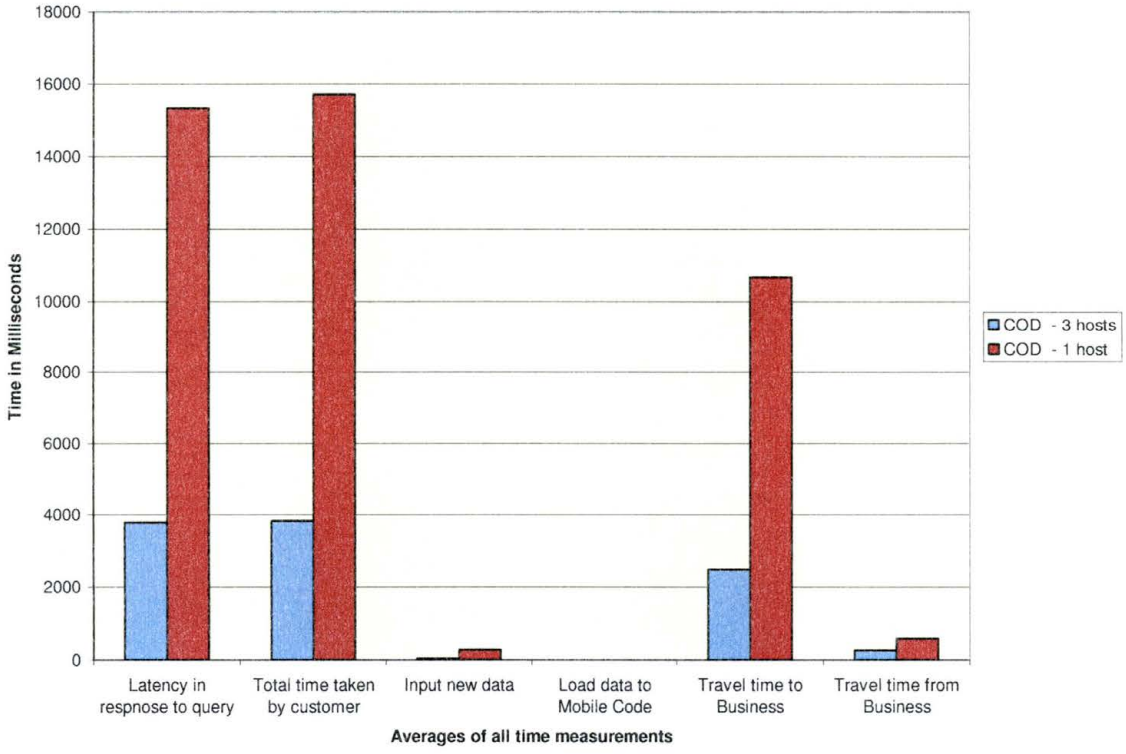

Figure 5.6: Changes in average times by increase in number of hosts in the COD paradigm

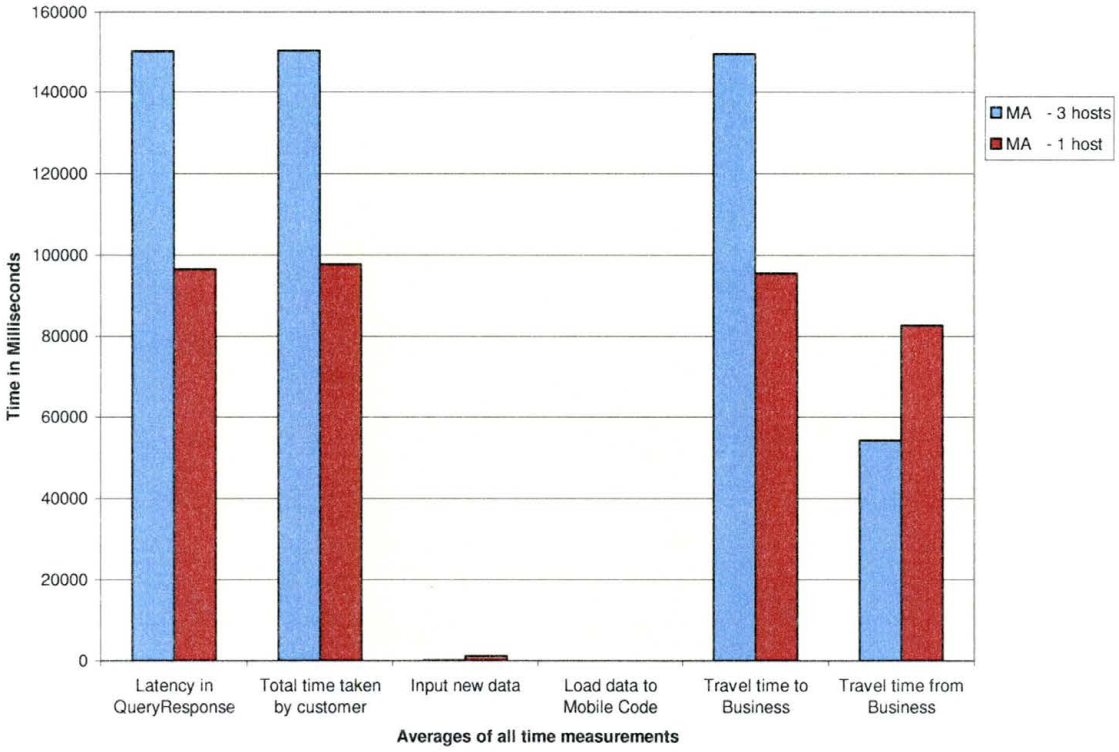

Figure 5.7: Changes in average times by increase in number of hosts in the MA paradigm

There is only a slight difference in the average values between the REV and COD paradigm in the graphs presented in Figure 5.5 and Figure 5.6. However, when there are 3 hosts, there is a large increase in the delay time. This is primarily because of the nature of the Mobile Agent which visits all the Businesses before returning to the Customer.

5.1.3 Code Metrics

5.1.3.1 Number of Executable Lines of code

Definition of the Metric

The number of executable lines of code measured in this experiment is considered to be lines of code in the program that that do not include empty lines in between lines of code, code to output data for debugging and comments used in code documentation.

Results

Amongst the classes that were used in the implementation of the Mobile Code in simulation framework, MobileCodeGroup.java and SendBack.java contain class MobileCodeGroup and class SendBack. The latter two classes are the core classes in which the design paradigms of Mobile Code are implemented. A comparison of the executable number of lines of code per paradigm is shown in the Figure 5.8. It can be inferred from the graph in Figure 5.8 that the COD paradigm has the least number of executable lines of code in contrast to REV paradigm, which has the highest. The reason for MA paradigm to contain fewer lines of code than REV is because it implements only one class, namely class MobileCodeGroup, unlike REV and COD paradigms that implement 2 classes.

It is to be noted that this metric gives information only about the classes that implement the Mobile Code paradigms but not about the other classes that are used in the framework.

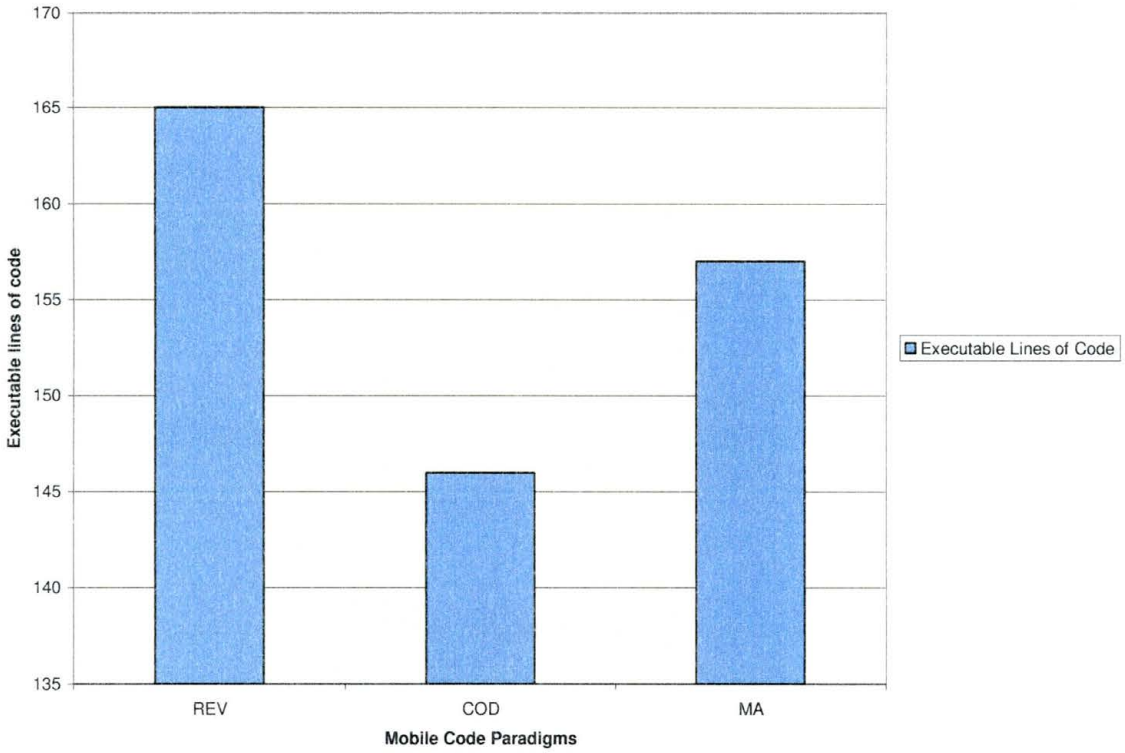

Figure 5.8: Comparison of executable lines of code

5.2 Qualitative Analysis

The qualitative metrics that can be reflected upon are the effects of Mobile Code design paradigms on the business processes and paradigms.

A considerable change brought by the Mobile Code design paradigms concerning the business paradigms of Mobile Code is the shift in ownership and control of the application. Such a shift in control and ownership by the users will also affect the developing technologies relevant to the Mobile Code paradigms. In conventional Client-Server systems, since the business entity is the sponsor of the e-commerce activity, a major part of the control over the processes, information and application features are with the business entity. With the use of Mobile Code, the users or consumers have the potential and freedom to design

and implement a few of the computational components by themselves. Still, the business entity will need to embrace openness and standards to honor the implementation of the customer-developed applications. Lack of standards would foster an environment with potential confusion while adapting to the openness of executing customer-developed Mobile Code on the business sites.

A review of the 'author' and 'design' parameters of entities that are subjects of computation, namely Customer Requirements (CR), Recommend process (R) and Business Information (BI), in the research experiment, reports the hypothesis. The Client – Server paradigm is also presented to draw comparisons to the Mobile Code paradigms:

Client – Server (CS):

Recommend process R	**Author**: Business
	Design: Business
Customer Requirements CR	**Author**: Customer
	Design: Business
Business Information BI	**Author**: Business
	Design: Business
Site of computation	Business

Table 5.5: Author and design parameters for of computational components for CS paradigm

Since the Business, which is the site of computation of the business entity, is the major sponsor of the e-commerce activity in this paradigm, most of the control is vested in the Business.

Remote Evaluation (REV):

Recommend process R	**Author**: Customer
	Design: Customer
Customer Requirements CR	**Author**: Customer
	Design: Customer or Business
Business Information BI	**Author**: Business
	Design: Business
Site of computation	Business

Table 5.6 Author and design parameters for of computational components for REV paradigm

The advantages of this paradigm are the reusability of the subjects of computation by the customers. A customer can use the same code and requirements for a series of Businesses. Also the customer can utilize the resources of the Business.

Code on Demand (COD):

Recommend process R	**Author**: Business
	Design: Business
Customer Requirements CR	**Author**: Customer
	Design: Customer or Business
Business Information BI	**Author**: Business
	Design: Business
Site of computation	Business

Table 5.7 Author and design parameters for of computational components for COD paradigm

One advantage of this paradigm for customers is that the BI is brought to the site of computation of the Customer. Hence, the customer can utilize the information several times without the need to retrieve it.

An advantage to the business in this paradigm is that the business could send information to the customer. This provides the business a good opportunity to promote their products and services better.

Mobile Agent (MA):

Recommend process R	**Author**: Customer
	Design: Customer
Customer Requirements CR	**Author**: Customer
	Design: Customer
Business Information BI	**Author**: Business
	Design: Business
Site of computation	Business

Table 5.8 Author and design parameters for of computational components for MA paradigm

The most beneficial feature of the mobile agent system is that a list of businesses to be searched can be given to the agent at once unlike initiating the process several times in other paradigms. The agent completes the searches by itself thus avoiding requiring the user having to initiate the process repeatedly.

5.3 Evaluation of Trade-offs of Mobile Code Design Paradigms for Recommend Activity in the Experiment

Each Mobile Code paradigm has tradeoffs depending on the conditions of its implementation. From the results and their analysis presented in the earlier part of this chapter, the summary about the Mobile Code design paradigms is presented in Table 5.9.

It can be inferred from Table 5.9 that there are conditions under which each Mobile Code paradigm is the most favourable paradigm. Or, if a paradigm is used under any given conditions, there are trade-offs that affect the outcome of the use of the paradigm. The most suitable paradigms of a given set of parameters are shaded in grey.

Tradeoffs	Remote Evaluation (REV)	Code on Demand (COD)	Mobile Agent (MA)
Latency in response to query Businesses: 1 per host Customers: {1,5,10, 15, 20} per 1 and 3 hosts	- least for 1 host until no. of customers< 3 - less for multiple hosts	- least for 1 host when no. of customer>3 - least for multiple hosts	- consistently high for rising for all values of no. of customers
Latency in response to query Businesses: {1,5,10, 15, 20} Customers: 20 per host per 1 and 3 hosts	- the least for 1 host -least for multiple hosts	- not favorable for 1 host - close to least for multiple hosts	- nearly the same for 1 and multiple hosts
Travel time To Business Businesses: 1 per host Customers: {1,5, 10, 15, 20} per 1 and 3 hosts	- favorable for 1 host until no. of customers < 10 - favorable for multiple hosts	- least for 1 hosts - least for multiple hosts	- remains similar for several sets of customers
Travel time for Mobile Code Businesses: {1,5, 10, 15, 20} Customers: 20 per host per 1 and 3 hosts	- favorable for 1 host when 5<no. of Businesses<10 - close to least for multiple hosts	- favorable for 1 host until no of Businesses < 5 - least for multiple hosts	- decreases as the no of Businesses increase - not favorable for multiple hosts
Net average effect of changes on 1 host v/s 3 hosts	- close to least for most parameters in multiple hosts	- least for most parameters in multiple hosts	- has lesser values for single host that than multiple hosts
Degree of design and authority by Customer	- very high - R and CR	- good - CR	- very high - R and CR

Table 5.9: Comparison of tradeoffs of REV, COD and MA paradigm

5.4 Inference About Other E-commerce Scenarios from Information and Statistics on the Current Scenario

Inferences about other paradigms based on the analyses on B2C paradigm using this simulation framework are as follows:

1. B2B scenario

With the knowledge of delay in query response from the experiment performed, since the COD paradigm has the least average values for most parameters, it is perhaps the paradigm that should be considered for incorporating in the current e-commerce applications or in the design of forthcoming applications. Qualitatively, the REV and MA paradigms change the paradigm of business by tending to distribute control more equally, thus creating a better opportunity for e-commerce. These are definite reasons why REV and MA should be incorporated more in e-commerce.

2. P2P scenario:

The wide ranges of users in the P2P scenario provide several opportunities to use the different Mobile Code paradigms. Examples include:

- Mobile agents can be used for extracting information about the location of required data
- REV can be used when there is a lack of available resources for computation
- COD is useful in utilizing the available resources rather than taxing an entity that does not have the available resources.

3. M-commerce:

Light weight mobile agents are useful for this e-commerce scenario. Mobile devices typically do not have adequate computing power compared to desktop computers. Therefore, agents that can relocate and execute code and manipulate information would be ideal for this scenario. In addition, the MA paradigm yields good results when being executed on a single machine.

5.5 Abstraction of all Mobile Code Design Paradigms

The abstraction of all the Mobile Code design paradigms is presented here with a series of steps to construct a new Mobile Code paradigms.

Requirements of a Mobile Code Design Paradigm

The requirements are resources and a site for computation. The resources are Customer Requirements (CR), Recommend or any executable process (R) and Business Information (BI). The site for computation is typically the Business or the Customer site.

Procedure for Designing a Mobile Code Paradigm

Step 1: Identify the location of all the resources. Identify a site for computation.

Step 2: Transfer the CR and R to the computation site

Step 3: Procure the required BI from the BI database to the computation site.

Step 4: Compute for recommendations and send the results to the client.

Step 5: Move the R and CR to the next computation site and

repeat step 1.

The benefits of realizing an abstraction are that it gives an insight and a possibility to create or discover new paradigms. One of the changes is the interactions between components.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.0 Overview

The conclusions derived from the research are discussed in this chapter. The limitations in the research are presented. The directions for future research are discussed at the end of this chapter.

6.1 Conclusions

The objective of the research is to evaluate the trade offs of Mobile Code design paradigms in e-commerce applications. Delay in response to the query is the primary metric that was measured. The conclusions that can be drawn from the implementation of this experiment are as follows:

- **Most Applicable Paradigm**

The most applicable Mobile Code design paradigm to a problem in a given condition depends on the implicit factors of the condition. The factor that has been measured in the research experiment was the delay in query response time.

The above mentioned conclusion can be supported from the results obtained in the research experiment. All the Mobile Code design paradigms, namely Remote Evaluation (REV), Code on Demand (COD) and Mobile Agent (MA), have been found to be the most suitable under varying parameters of the conditions for the number of Businesses and number of Customers involved in the experiment.

- **Increase in the Number of Hosts in the Simulation Network**

An increase in the number of hosts in the simulation network has distributed the load and raised an increase in the availability of the computational power. In the research experiment, upon increasing the number of hosts in the network and maintaining the remaining parameters constant, the REV and the COD paradigm exhibit similar performance. The MA paradigm implemented in the research is not affected beneficially. This is due to the increase in the network traffic with the introduction of a greater number of hosts in the experiment.

- **Time Taken to Travel by Mobile Code**

Mobile Code provides enormous flexibility in the design of architectures for systems demanding code mobility. The time taken to travel by the Mobile Code depends on the architectural abstraction of the paradigm.

The time taken to travel by the MA paradigm is the highest in this experiment. But this is because the mobile agent designed appends the entire set of results acquired from every business it visits in contrast to updating the existing results with the business it is visiting, and thereby carrying only a limited set of results. In the REV and COD paradigms, the Mobile Code takes more time to travel from the customers to reach the businesses than the time it takes to travel from businesses to reach the customer.

- **Qualitative Analysis**

The various possibilities of interactions in the Mobile Code paradigms create new possibilities for business paradigms in e-commerce. The advent of such business paradigms can significantly contribute to the growth of commerce and society.

- **Simulation Framework for Mobile Code Paradigms**

The framework that is developed in the research experiment is an attempt to develop a simulation framework that is independent of the Mobile Code paradigm that it implements. Though it is not completely independent from the Mobile Code paradigms, a great degree of flexibility and independence from them has been achieved.

6.2 Limitations

Limitations in resources decrease the scope of implementation. And limitations in implementation decrease the scope of research. Identifying the limitations will help in understanding of the scope of the research better. Also, research in future should aim in exceeding the limitations in the current research. The limitations in resources and implementation that have been encountered during the research are:

- **Number of Hosts in the Simulation Framework**

  One of the factors affecting the proposed metric, delay in the response to query, is the number of hosts in the simulation framework. The number of hosts participating in the simulation network has been limited to only 3. With the availability of more hosts, it is possible to simulate the experiment with more businesses and customers.

- **Metrics Incorporated in the Simulation Framework**

  Delay of response to the query is the only metric that has been measured in the research experiment. It is desirable to have more metrics incorporated in the framework that can be measured for every Mobile Code paradigm.

- **Simulated E-commerce Scenarios**

  The simulated e-commerce scenario was suitable mostly only for the B2C e-commerce scenario. A modification in the framework to incorporate multiple business scenarios that simulate their conditions is desirable.

- **Strong Coupling of Metric Points to the Framework**

  The metric that has been measured is strongly coupled to the simulated framework. Such hard binding is not desirable when there is need to simulate the frameworks without the measurement of the metrics. It could also be possible that the very metric that is measured also affects the performance of the system.

6.3 Directions for Future Research

Mobile Code design paradigms present new opportunities to technology and business. There is an ample scope for beneficial research in this field. Some directions for future research are:

- **Ideal Framework for Simulation**

  µCode has been the only Mobile Code implementation technology that is independent of the Mobile Code design paradigm. However, there is no framework that has been developed thus far that implements the Mobile Code paradigms and is independent of them. An attempt has been made to develop a simulation framework that is independent of the Mobile Code paradigms. The

Ideal Simulation Framework mentioned in the section 4.2.1 can be pursued. Such a framework could decide the best possible Mobile Code design paradigm in the given conditions by quantitatively evaluating the tradeoffs.

The Ideal Simulation Framework should be developed to incorporate the abstraction of Mobile Code design paradigms discussed in Results Chapter 5. It will facilitate the discovery of new Mobile Code paradigms and enhance the development of the existing ones.

- **New Design Paradigms in Mobile Code**

This research has involved only 3 Mobile Code paradigms, which are REV, COD and MA. The flexibility of the interactions in Mobile Code presents more opportunities to develop new design paradigms. The new paradigms will have to be tested and compared to the existing paradigms. However, the highest motivation for using these paradigms will depend on their application.

- **Quantitative Comparison of Different E-commerce Scenarios**

There exist various e-commerce scenarios namely: B2B, B2C, P2P, etc. Investigation of the various scenarios quantitatively on factors such as the ratio of customers to businesses, the number or customers and businesses, the application level maturity of the users can be very useful for not only research in Mobile Code but also for several other fields in e-commerce.

- **Flexibility of Change in the Implemented Paradigm in an Application**

Since the Mobile Code paradigms have tradeoffs and are best suited under different conditions, the possibility of dynamically changing the Mobile Code paradigms implemented in an application under the given conditions can be explored.

- **Survey of the Current Applications and their Technologies used in E-commerce**

An attempt to survey and classify the current applications and their technologies has been made in the initial stage of the research, but it was not concluded due to the large size of the problem and its deviation from the research. Knowledge about the current applications and their implementation technologies can forecast the benefits that can be achieved by incorporating Mobile Code in the applications.

Mobile Code has possibilities for several applications and implementations. This research project has only brought to surface the potential and richness of Mobile Code.

APPENDIX A

In this section, an overview of the program used to implement the simulation framework for Mobile Code paradigms is presented. The entire code is too long to be included in this document. A major part of it has been included in Appendix B. The code can be obtained by request from the author by emailing him to naveenkoneru@yahoo.com.

The classes that are used in the implementation of the simulation framework are:

- Class ParadigmCustomerCntrllr
- Class ParadigmCustomer
- MobileCodeGroup
- SendBack
- noObjClass
- CalculateHostPortId
- ExpDataHandler
- BIDataHandler
- CRDataHandler
- RDataHandler
- Product

A script was developed to start the execution of the simulation framework with the required input parameters. Figure A1 shows the interaction of classes in the developed simulation framework.

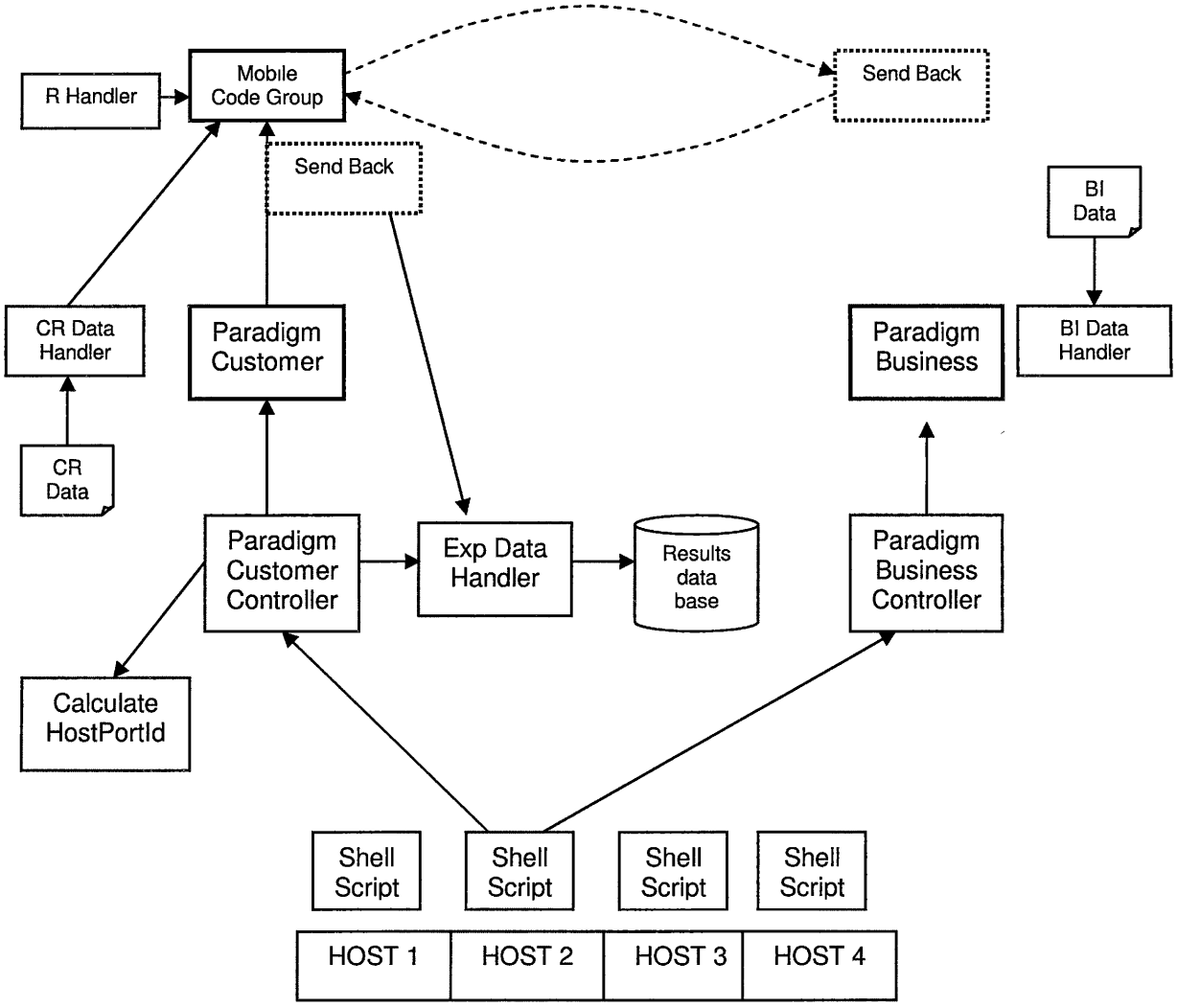

Figure A1: Interaction of classes in the simulation framework

APPENDIX B

```
/*****************************************
 *
 *     Class BIDataHandler
 *
 *****************************************
 *
 * BIDataHandler is the class that handles the operations of BIData
 * which is stored in BIData.txt
 *
 *****************************************/


package data;

import java.io.*;
import java.lang.*;

public class BIDataHandler {
  static final int MAX_BIData_SIZE = 6;
  private Product[] BIData = new Product[MAX_BIData_SIZE];

  public void loadBIDataFromFile() {
  // Load the BIData[] with the input data from BIData.txt  //change
    try {
      BufferedReader in = new BufferedReader(new
FileReader("data/BIData.txt"));
      for ( int i = 0; i < MAX_BIData_SIZE; i++) {
        String line =  in.readLine();
        int     productId        = Integer.parseInt ( quotedString(line, 0,
'\t', '\t') );
        String  productName      =                    quotedString(line, 1,
'\t', '\t')  ;
        String  manufacturerName =                   quotedString(line, 2, '\t',
'\t')  ;
        int     itemId           = Integer.parseInt ( quotedString(line, 3,
'\t', '\t') );
        String  processor        =                   quotedString(line, 4, '\t',
'\t')  ;
        int     memory           = Integer.parseInt ( quotedString(line, 5,
'\t', '\t') );
        double  price            = Double.parseDouble(quotedString(line, 6,
'\t', '\t') );
        int     qualityRanking   = Integer.parseInt ( quotedString(line, 7,
'\t', '\t') );
        int     customerRating   = Integer.parseInt ( quotedString(line, 8,
'\t', '\t') );
        BIData[i] = new Product( productId, productName,manufacturerName,
itemId,
              processor, memory, price, qualityRanking, customerRating );
      }
    } catch (IOException e){
```

```
      System.out.println("BIDataHandler Exception: Could not open the file
BIData.txt");
    }
  }


  private static String quotedString( String from, int startCharPos,
      //change sig
                                    char start, char end) {
  // This function is used in loadBIDataFromFile() to extract the
  // required BIData from a line of data. Used only within the class
  // and is hence "private".
    try {
        int startPos = 0;
        for( int i=0; i<startCharPos; i++ )
          startPos = from.indexOf(start, startPos) + 1;
        int endPos = from.indexOf(end, startPos + 1);

        if ( endPos == -1 )          // end is the length of the string
          endPos = from.length();

        if (startPos > endPos)    // start after end
          return null;
        else if (startPos == -1)  // no start found
          return null;
        else
          return from.substring(startPos, endPos );
    } catch (NullPointerException e){
        return "EOF for BIData.txt";
    }
  }

  // toString
  public String toString() {
  // does not really return the BIData but prints them
    for ( int i = 0; i < MAX_BIData_SIZE; i++) {
        System.out.println(BIData[i]);
    }
    return "The above is BIData.txt ";
  }

  // accessor
  public Product[] getBIData() {
  // Returns the BIData pointer and the objects
  // can be accessed from that pointer
    return BIData;
  }

  // Copying the BIData
  public void  makeCopy(BIDataHandler CopyBIDataHandler) {
    System.arraycopy(this.BIData, 0, CopyBIDataHandler.getBIData(), 0,
MAX_BIData_SIZE);
  }

  // getters for records :  these return the index of all the
  // records in the BIData[] which satisfy the given conditions eg: price

  public void getRecordsFromBIData(int[] indexPrice, double price) {
    try {
      for ( int i = 0, j = 0; i < MAX_BIData_SIZE; i++  ) {
        if ( BIData[i].getPrice() <= price  ) {
          indexPrice[j] = i;
          j++;
```

```
        }
      }
    } catch (ArrayIndexOutOfBoundsExceptıon e) {
      System.out.println("EXCEPTION ArrayIndexOutOfBounds: No of results
exceeded "
              + "array size in
BIDataHandler.getRecordsFromBIData(indexPrice,price)");
    }
  }

  public void getRecordsFromBIData(int[] indexMemory, int memory) {
   try {
      for ( int i = 0, j = 0; i < MAX_BIData_SIZE; i++  ) {
        if ( BIData[i].getMemory() >=  memory) {
          indexMemory[j] = i;
          j++;
        }
      }
    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("EXCEPTION ArrayIndexOutOfBounds: No of results
exceeded "
             + "array size in
BIDataHandler.getRecordsFromBIData(indexPrice,price)");
    }
  }

}//EOF


/****************************************
 *
 *     Class CalculateHostPortId
 *
 ****************************************/
 *
 * This class calculates the Business and the Customer addresses and
 * informs the Customers about all the Business addresses that the
 * Mobile Code needs to travel. An address is a combination of the
 * Host Id and the Port Number.
 *
 *
 ****************************************/

publıc class CalculateHostPortId{
  public void CalculateBusinessHostPortId(String[] businessHostId,
             int[] busınessPortId, int noOfB, int noOfMachines){



    final int MAX_noOfB = 40;           // Maximum no. of Busınesses on all
machines
    final int MAX_noOfMachınes = 4;     // Maximum no. of Machines
    final String[] MACHINE = new String[5];    // Business Machine IPaddresses
    MACHINE[0] = null;     // dummny
    MACHINE[1] = "147.26.101.222";
    MACHINE[2] = "147.26.101.223";
    MACHINE[3] = "147.26.101.224";
    MACHINE[4] = "147.26.101.221";

    if( noOfB > MAX_noOfB){
      System.out.println("noOfB > MAX_noOfB Hence quitting
CalculateBusinessHostPortId");
    }
```

```
    if( noOfMachines > MAX_noOfMachines){
      System.out.println("noOfMachines > MAX_noOfMachines " +
                         "Hence quitting CalculateBusinessHostPortId");
    }

    try {
      // This method is used by ParadigmCustomerControllr
      int[] noOfBMachine = new int[5]; // 5 is chosen so that we can
                                //go from 1 - 4 instead of 0 - 3.

      // Min Hosts per every machine
      int perMachine = noOfB/noOfMachines;
      int reminder = noOfB % noOfMachines;

      // No of Hosts for every machine
      // Machine 1
      for(int i = 1; i <= noOfMachines; i++){
        noOfBMachine[i] = perMachine;
        if(reminder >= i)
          noOfBMachine[i]++;
      }

      // Checking for the noOfBMachine per each machine
      System.out.println("The No of Businesses per each machine are:");   //
test
      for(int i = 1; i < 5; i++)                                          //
test
        System.out.println("noOfBMachine[" +i+ "] = " + noOfBMachine[i]); //test

      int k = 0, port = 1970;
      for(int l = 1; l <= 4; i++){
        for(int j = 0; j < noOfBMachine[i]; j++){
         businessHostId[k] = MACHINE[l];
         businessPortId[k] = port;
          k++;
          port = port + 1;
        }
      }

      for(k = 0; k < businessHostId.length ; k++){                        //
test
        System.out.println("businessHostId[" + k + "] = " + businessHostId[k]);
       // test
        System.out.println("businessPortId[" + k + "] = " + businessPortId[k]);
       // test
      }                                                                   //
test

    } catch (ArrayIndexOutOfBoundsException e) {
      System.err.println("Array out of bounds in
CalculateBusinessHostPortId()");

      if( noOfB > MAX_noOfB)
        System.out.println("noOfB > MAX_noOfB Hence quitting
CalculateBusinessHostPortId()");

      if( noOfMachines > MAX_noOfMachines)
        System.out.println("noOfMachines > MAX_noOfMachines " +
                           "Hence quitting CalculateBusinessHostPortId");

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

```
      System.err.println("Error in CalculateBusinessHostPortId()");
      e.printStackTrace();
    }
  }

}//EoClass


/*****************************************
 *
 *     Class CRDataHandler
 *
 *****************************************
 *
 * CRDataHandler is the class that handles the operations of
 * CRData which is stored in CRData.txt
 *
 *****************************************/

package data;

import java.io.*;
import java.lang.*;

public class CRDataHandler {
  private String processor;
  private int memory;
  private double price;
  private int qualityRanking;
  private int customerRating;

  // Accessors

  public String toString() {
    String CRlist = " CR are processor: " +  processor +
                " Memory : " + memory +
                " Price : " + price +
                " Quality Ranking : " + qualityRanking +
                " CustomerRating : " + customerRating;
    return CRlist;
  }

  public String getProcessor(){
    return processor;
  }

  public int getMemory(){
    return memory;
  }

  public double getPrice(){
    return price;
  }

  public int getQualityRanking(){
    return qualityRanking;
  }

  public int getCustomerRating(){
    return customerRating;
  }

  // Setters
```

```
  public void loadCRDataToFile(String processor, int memory, double price,
                      int qualityRanking, int customerRating){

    // Load the CRData.txt file with the given input CRData
    try{
       BufferedWriter out = new BufferedWriter(new
       FileWriter("data/CRData.txt"));
       out.write(processor + "\n" + memory + "\n" + price + "\n" +
              qualityRanking + "\n" + customerRating + " -1");
       out.close();
    } catch (IOException e){ }
  }

  public void loadCRDataToFile(String filename, String processor, int memory,
               double price, int qualityRanking, int customerRating){

    // Load the <filename> file with the input CRData
    try{
       BufferedWriter out = new BufferedWriter(new FileWriter(filename));
       out.write(processor + "\n" + memory + "\n" + price + "\n" +
              qualityRanking + "\n" + customerRating + " -1");
       out.close();
    } catch (IOException e){ }
  }


  public void loadCRDataFromFile(){
    // Load the CRData with the input data from CRData.txt
    try{
       System.out.println("loadCRDataFromFile() is called."); //test
       BufferedReader in = new BufferedReader(new
FileReader("data/CRData.txt"));
       setProcessor( in.readLine() );
       setMemory( Integer.parseInt(in.readLine()) );
       setPrice( Double.parseDouble(in.readLine()) );
       setQualityRanking( Integer.parseInt(in.readLine()) );
       setCustomerRating( Integer.parseInt(in.readLine()) );
       in.close();
    } catch (IOException e){ }
  }


  public void loadCRDataFromFile(String filename){
    // Load the CRData with the input data from CRData.txt
    try{
       System.out.println("loadCRDataFromFile(filename) is called."); //test
       filename = "data/" + filename;                              // ???
       BufferedReader in = new BufferedReader(new FileReader(filename));
       setProcessor( in.readLine() );
       setMemory( Integer.parseInt(in.readLine()) );
       setPrice( Double.parseDouble(in.readLine()) );
       setQualityRanking( Integer.parseInt(in.readLine()) );
       setCustomerRating( Integer.parseInt(in.readLine()) );
       in.close();
    } catch (IOException e){ }
  }

  public void inputNewData(String processor, int memory, double price,
                      int qualityRanking, int customerRating){
    this.processor = processor;
    this.memory = memory;
    this.price  = price;
```

```
    this.qualityRanking = qualityRanking;
    this.customerRating = customerRating;
  }

  public void setProcessor(String processor){
    this.processor = processor;
  }

  public void setMemory( int memory){
    this.memory = memory;
  }

  public void setPrice(double price){
    this.price = price;
  }

  public void setQualityRanking(int qualityRanking){
    this.qualityRanking = qualityRanking;
  }

  public void setCustomerRating(int customerRating){
    this.customerRating = customerRating;
  }
}


/*****************************************
 *
 *     Class ExpDataHandler
 *
 *****************************************
 *
 * ExpDataHandler is the class that handles the operations
 * to store the Timestamps that are recorded in the experiement
 *
 *****************************************/

import java.io.*;
import java.sql.*;

public class ExpDataHandler{

  final String databaseHost   = "147.26.101.225";
  final String database = "data";
  final String userid   = "naveen";
  final String password = "results1";

  ExpDataHandler() {
    System.out.println("edh: In the contructor of ExpDataHandler()");     //
test
  }

  // Report the values to the database
  public void reportCustomerDataToDB( int expId, String destination,
                                long[] TimeCalcs, long[] TimeCounters){
    String url = "";
    String urlMaskPassword = "";
    try {
      // Test the Driver
      Class.forName ( "com.mysql.jdbc.Driver" );
      System.out.println ( "MySQL Driver Found" );                 // test

      // Connection string
```

```
      url = "jdbc:mysql://" + databaseHost + "/" + database +
                "?user=" + userid + "&password=" + password;
      urlMaskPassword = "jdbc:mysql://" + databaseHost + "/" + database +
                "?user=" + userid + "&password=**********";
      Connection con = DriverManager.getConnection(url);
      System.out.println("Connection established to " + urlMaskPassword +
"...");  // test

      // Write the insert query
      String insertStmt = "insert into timerdata values (" + expId + ",' "
       + destination + "', " + TimeCalcs[0] + ", "
       + TimeCalcs[1] + ", " + TimeCalcs[2] + ", " + TimeCalcs[3] + ", " +
TimeCalcs[4] + ", "
       + TimeCalcs[5]+ ", " + TimeCounters[0] + ", " + TimeCounters[1]  + ", " +
TimeCounters[2]  + ", "
       + TimeCounters[3]  + ", " +TimeCounters[4]  + ", " +TimeCounters[5]  + ",
"
       + TimeCounters[6] + ", 0)"; // the 0(zero) is for the 'index' field,
                                   // which is actually an auto-incremented
      System.out.println("The insertStmt is : " + insertStmt);           //
test

      // Create a statement
      Statement s = con.createStatement();
      s.execute(insertStmt);
      s.close();
      con.close();
    } catch ( java.lang.ClassNotFoundException e ) {
      System.out.println("MySQL JDBC Driver not found ... ");
      e.printStackTrace();
    } catch ( java.sql.SQLException e) {
      System.out.println("SQLException" );
      e.printStackTrace();
    }
  }

  // Load the TimerData.xls file with the given input Experiment Data
  public void reportCustomerData( int expId, String destination,
                                  long[] TimeCalcs, long[] TimeCounters){
    System.out.println("edh: In the reportCustomerData()....");          //
test
    System.out.println("edh: NOTE: Should we change the path of the file" +
                       " 'TimerData.xls' ");              // test

    FileWriter fout;
    try{
       // change - Should we change the path of the file 'ExpData.xls'
//change
       // change - should we convert 'paradigm' into int

       fout = new FileWriter("TimerData.xls", true); // true - for appending at
EOF

       fout.write( expId + "\t" + destination );

       // TimeCalcs values
       for(int i = 0; i< 6; i++)
         fout.write("\t" + TimeCalcs[i]);

       // TimeCounter values
       for(int i = 0; i< 7; i++)
         fout.write("\t" + TimeCounters[i]);
```

```
        fout.write("\n");
        fout.close();

    } catch (FileNotFoundException fex){
      System.out.println("edh: fex: File 'ExpData.xls' not found exception"
                         + " in reportExpData()");
    } catch (Exception e){
      e.printStackTrace();
    }
  }

  // Load the expdata file in the 'data' database with the given input
Experiment Data
  public void reportExpDataToDB( int expId, boolean ifCreateReqs, boolean
ifExecute,
        String paradigm, int noOfB, int noOfC, int noOfMachines ) {

    String url = "";
    String urlMaskPassword = "";
    try {
      // Test the Driver
      Class.forName ( "com.mysql.jdbc.Driver" );
      System.out.println ( "MySQL Driver Found" );                    // test

      // Connection string
      url = "jdbc:mysql://" + databaseHost + "/" + database +
                  "?user=" + userid + "&password=" + password;
      urlMaskPassword = "jdbc:mysql://" + databaseHost + "/" + database +
                  "?user=" + userid + "&password=***********";
      Connection con = DriverManager.getConnection(url);
      System.out.println("Connection established to " + urlMaskPassword +
"...");   // test

      // Write the insert query
      String insertStmt = "insert into expdata values (" + expId + ", '"
        + ifCreateReqs + "', '" + ifExecute + "', '" + paradigm + "', " + noOfB
+ ", " + noOfC + ", "
       + noOfMachines + ",  NOW(), 0 ) "; // NOW() is a function for the 'time'
field  which is of
       // the type TIMESTAMP and 0 for 'index' which is autoincremented
      System.out.println("The insertStmt is : " + insertStmt);          // test

      // Create a statement
      Statement s = con.createStatement();
      s.execute(insertStmt);
      s.close();
      con.close();
    } catch ( java.lang.ClassNotFoundException e ) {
      System.out.println("MySQL JDBC Driver not found ... ");
      e.printStackTrace();
    } catch ( java.sql.SQLException e) {
      System.out.println("SQL Exception" );
      e.printStackTrace();
    }
  }

  // Load the ExpData.xls file with the given input Experiment Data
  public void reportExpData( int expId, boolean ifCreateReqs, boolean
ifExecute,
        String paradigm, int noOfB, int noOfC, int noOfMachines ) {

    System.out.println("edh: In the reportExpData()....");         // test
    System.out.println("edh: NOTE: Should we change the path of the file" +
```

```
                        " 'ExpData.xls' ");           // test

    FileWriter fout;
    try{
       // change - Should we change the path of the file 'ExpData.xls'
//change
       // change - should we convert 'paradigm' into int

       fout = new FileWriter("ExpData.xls", true); // true - for appending at
EOF
       fout.write(expId+ "\t" + ifCreateReqs + "\t" + ifExecute + "\t" +
paradigm +
                      "\t" + noOfB + "\t"+ noOfC + "\t"+ noOfMachines + "\n");
       fout.close();

    } catch (FileNotFoundException fex){
      System.out.println("edh: fex: File 'ExpData.xls' not found exception"
                       + " in reportExpData()");
    } catch (Exception e){
      e.printStackTrace();
    }
  }

}


/*****************************************
 *
 *     Class MobileCodeGroup
 *
 *****************************************
 *
 * MobileCodeGroup is the Mobile Code Group class. It loads the
 * The data objects and class objects in a "Group" and transfers
 * Them to the destination. It "unpacks" the contents there and
 * Starts a new thread of execution.
 *
 *****************************************/

package muCodeExtensions;

import mucode.*;
import java.lang.reflect.*;
import java.io.*;

public class MobileCodeGroup extends Thread
       implements GroupHandler, Serializable {
  private transient MuServer server = null;
  private noObjClass noObj  = new noObjClass();
  private String source = null;

  public MobileCodeGroup() {
    System.out.println("mcg: In the MobileCodeGroup parameterless
constructor"); //test
  }

  public MobileCodeGroup(MuServer server){
    this.server = server;
    System.out.println("mcg: In the 'MobileCodeGroup parameter' constructor");
      //test
    System.out.println("mcg: The MuServer is " + server);                //test
  }
```

```
  public final void ship(String destination, String[] classNames,
                         Object[] objectNames)
       throws MuCodeException, ClassNotFoundException, IOException {
    if(server == null)
      server = MuServer.getServer(this);

    // Checking if control is in ship()                                    //test
    System.out.println("mcg: In ship() of MobileCodeGroup" );              //test
    System.out.println("mcg: MobileCodeGroup's MuServer is " + server );  //test

    // Creating a 'group' with 'Handler' and 'Root' as  "shipClassName"
    // shipClassName can be invoked in unpack if this(i.e. MobileCodeGroup)
    // can be a ubiquitous class
    String shipClassName = this.getClass().getName();
    System.out.println("mcg: MobileCodeGroup's Handler and Root are "+
      shipClassName);                                                     //test

    // Setting 'noObj.no' to the reqd value before creating the group
    noObj.no = objectNames.length;                                        //test
    System.out.println("mcg: Before packing, noObj.no: " + noObj.no);     //test

    // Creating a group
    Group group = server.createGroup(shipClassName, shipClassName);
    group.addObject("ShipHandlerLabel", this);
    group.addObject("noObj", noObj);                                      //test
    group.setDynamicLinkSource(group.getSource());
    group.setSynchronousTransfer(false);

    // Adding Classes in the group
    group.addClasses(classNames);

    // Adding Objects in the group
    // NOTE - The classes of the objects will also be added here individually,
if
    // not added before
    for (int i = 0; i < objectNames.length; i++) {
      String objectKey = "Obj" + i;
      System.out.println("mcg: The name of " + objectNames[i] + " is " +
objectKey); //test
      group.addObject(objectKey, objectNames[i]);
      group.addClass(objectNames[i].getClass());
    }
    System.out.println("mcg: The noObj.no are " + noObj.no );             //test

    // List the classes in the group
    String[] groupClasses = group.getClasses();                   //test
    System.out.println("mcg: Group Classes to be SHIPPED are");          //test
    for (int i =0; i < groupClasses.length; i++)                  //test
      System.out.println("   " + groupClasses[i]);                    //test

    // REAL DEAL - finally
    try {
      group.ship(destination);
    } catch (Exception e) { e.printStackTrace(); }

  }

  public synchronized Thread unpack(Group group)
       throws MuCodeException {

    System.out.println("mcg: In unpack() of MobileCodeGroup");           //test
    MobileCodeGroup destnHandler
=(MobileCodeGroup)group.getObject("ShipHandlerLabel");
```

```
    server = group.getServer();
    System.out.println("mcg: The Thread after unpacking is " +
Thread.currentThread()); //test
    System.out.println("mcg: And the  MuServer is " + server);          //test
    noObjClass noObj = (noObjClass)group.getObject("noObj");            //test

    // List the classes in the group
    String[] groupClasses = group.getClasses();                    //test
    System.out.println("mcg: Group Classes that are SHIPPED are");      //test
    for (int i =0; i < groupClasses.length; i++)                   //test
      System.out.println("   " + groupClasses[i]);                 //test

    // List the objects in the group                               //test
    System.out.println("mcg: The noObj.no are " +noObj.no );            //test
    System.out.println("mcg: The objects are ");                    //test

    for (int i = 0; i < noObj.no; i++) {                           //change
      String objectKey = "Obj" + i;                                //test
      Object objectListed = group.getObject(objectKey);                //test
      System.out.println("mcg:   " + objectListed);                //test
    }

  // For the source of the Group
  this.source = group.getSource();

  // To run() the thread
  this.start();

  return null;
  }

  public void run(){
    // Checking if control is in run()
    System.out.println("mcg: In the run of MobileCodeGroup");           //test
    System.out.println("mcg: Do the required processing and send back "
             + "the required classes and objects with SendBack");  //test

    try {
      // Sample return information
      String destination = this.source;                            // change
      //String[] classNames = {ToolkitNext.class.getName()};
       //change
      String[] classNames = {};
      //ToolkitNext tkn1 = new ToolkitNext();                      //change
      Object[] objectNames = new Object[]{};

      // Creating a 'SendBack' Thread to send back classes and objects
      Thread.sleep(5000);
      SendBack shipHandlerReturn = new SendBack(server);
      System.out.println("mcg: While sending back ");              //test
      System.out.println("mcg: destination : " + destination);          //test
      System.out.println("mcg: classNames : " + classNames[0]);         //test
      System.out.println("mcg: objectNames : " + objectNames[0]); //test

      // Make change to 'destination'                              //change
      //shipHandlerReturn.shipBack(destination, classNames,
objectNames);//change
      shipHandlerReturn.shipBack("127.0.0.1:1988", classNames,
objectNames);//change
    } catch (Exception e) { e.printStackTrace(); }
  }

}
```

```
/*****************************************
 *
 *     Class ParadigmBusinessCntrllr
 *
 *****************************************
 *
 * ParadigmBusinessCntrllr is the class that is started at the
 * Business to start a MuServer ready to receive Mobile Code sent
 * by the Customers.
 *****************************************/

class ParadigmBusinessCntrllr{

void main(ExpId, IfCreateReqs, IfExecute, paradigm,
CustomerHostId, CustomerPortId, BusinessHostId, BusinessPortId)
{
  TimeStamp;
  BusinessTimeCounter1;
According to the paradigms -
  new ParadigmBusiness( NoOfB, NoOfC, NoOfMachines,
                        BusinessHostId, BusinessPortId)
  // Will NOT load any muServers. Will ONLY STORE CHostId,etc..
  // This will create a ParadigmBusiness Object
  if ( IfCreateReqs == yes )
    ParadigmBusiness.InputNewData()
    // Asks what do you want to input eg: CR, R, BI
    // Each paradigm knows what it needs
    // Calls CRHandlers & inputs new CR Data and R Data
  BusinessTimeCounter2;
  // These times may have to be counted by the MobileCodeGroup itself
  // and their values will have to be passed to SendBack to be taken back
  // to ParadigmCustomerCntrllr which can store them in the database
  if ( IfExecute == no)
    if( IfCreateReqs == yes)
      ExpDataHandler.ReportBusinessMeasurements()
      // At this point measurements 0 1 2  can be sent
      // with the rest as 0 or NULL
    quit()
  ParadigmBusiness.PerformQuery()
  // 0. fork() the listener and continue PerformQuery() with the thread
  // Since the CustomerHostId, CustomerPortId, BusinessHostId, BusinessPortId
  // are already there,
  // 1. ParadigmBusiness launches to recieve Mobile Code. It
  // has the required code (eg.BI Handler,R Handler, etc) in it's
  // shared class space. And hence the MobileCode that comes with the
  // ParadigmBusiness can access the classes
  // 2. MobileCode.PerformQuery() - Mobile Code performs Query  by interacting
  //  with BI Data through BI Data Handlers
  // 3. MobileCode stores the results in it
  // 4. LoadDataToMobileCode() - In the case of sending it back to Customer,
  // add/delete the classes that are recieved with the MobileCode when
  // sending it back to Customer
  // 5. MobileCode tells ParadigmBusiness where it needs to send back
MobileCode
  // Refer to Picco's examples
  TimeCoutner3;
  ParadigmBusiness.end()
  // End of ParadigmBusiness object - perhaps cleanup, etc.
  ExpDataHandler.ReportBusinessMeasurements()
  // Sends all the measurements to ExperimentDataHandler
  DisplayMeasurements()
  // Can display measurements that have been sent to the  database
```

```
void DisplayMeasurements()
  // Display all measurement taken


package paradigm;

import muCodeExtensions.*;
import data.*;

public interface ParadigmBusinessInterface {
  ParadigmBusiness( String BHostIP, int BHostPort );
  void loadBDataToMobileCode();
  void performQuery();
  void stopParadigmBusinessServer();
}
import mucode.*;
import mucode.abstractions.*;
import mucode.util.*;


/*****************************************
 *
 *     Class ParadigmCustomer
 *
 *****************************************
 *
 * ParadigmCustomer is the class that is starts an instance of the
 * Customer. It then, loads the Mobile Code with the objects and classes
 * and ships them to the destination. It then starts a MuServer to receive
 * Customer Mobile Code that are returning from the Business.
 * which is stored in BIData.txt
 *
 *****************************************/

public class ParadigmCustomer{

  private int CHostPort              = 0;
  private String customerSource      = null;
  private String[] businessDestination = new String[40];
  private MuServer s                 = null;
  private CRDataHandler CR           = null;
  private RHandler R                 = null;
  private String[] classesToBeShipped= null;
  private Object[] objectsToBeShipped= null;

  ParadigmCustomer(String CHostIP, int CHostPort,
                           String[] BHostIP, int[] BHostPort ) {
    // Concatenating strings to resolve source and destination MuServers
    this.CHostPort       = CHostPort;
    this.customerSource      = CHostIP + ":" + CHostPort;
    //for(int i = 0; i < 40; i++)
    for(int i = 0; (i < 40) && ((BHostIP[i] != null)); i++)        // test
      this.businessDestination[i] = BHostIP[i] + ":" + BHostPort[i];


    System.out.println("In ParadigmCustomer constructor\n " +
                    "CustomerSource is " + customerSource);        // test
    // Checking on values
    for(int i = 0; (i < 40) && ((BHostIP[i] != null)); i++)        // test
      System.out.println("businessDestination[" +i+ "] = " +
businessDestination[i]);//test
  }
```

```
  public void inputNewCData() {
    // This function is used to input values of CR, etc by creating
    // objects of CRDataHandler and inputting CRData.txt, etc.
    // NOT working right now - This function can be used to input values in CR,
etc. //???
    CR = new CRDataHandler();
    CR.loadCRDataFromFile();
    System.out.println("CR is " + CR);                                    //test
  }

  public void loadCDataToMobileCode() {
    // In the REV Paradigm, we load CRDataHandler and RHandler Objects
    // This is NOT really loading data to MobileCode but preparing the
    // list of classes and objects to add to MobileCodeGroup

    // Objects to be shipped
    objectsToBeShipped = new Object[]{};

    System.out.println("Objects to be shipped are  " );
       //test
    for (int i = 0; i <  objectsToBeShipped.length; i++ )
       //test
      System.out.println(" " + objectsToBeShipped[i]);
       //test

    // Classes to be shipped
    // None until now, since all classes are there on both the business &
customer side
    if(classesToBeShipped == null)                                       //test
      System.out.println(" classesToBeShipped are none");
       //test
    else {                                                        //test
      System.out.println(" classesToBeShipped are ");                    //test
      for (int i = 0; i <  classesToBeShipped.length; i++ )              //test
        System.out.println(" " + classesToBeShipped[i]);                 //test
    }                                                             //test
  }

  public void performQuery(int expId, long[] TimeCounter) {
    // Starting the MuServer
    s = new MuServer();

    // Having ubiquitous classes for recieval
    s.addUbiquitousClass(SendBack.class.getName());
    s.addUbiquitousClass(noObjClass.class.getName());
    s.addUbiquitousClass(BIDataHandler.class.getName());
    s.addUbiquitousClass(RHandler.class.getName());

    try {
      // Booting a server to listen for incoming groups
      s.boot();

      // Setting the port number for incoming groups
      String CHostPortValue = "" + CHostPort;
      s.setProperty("port", CHostPortValue);                             //???
      System.out.println("MuServer s is listening to " + s.getPort());
       //test
      System.out.println("CHostPortValue is " + CHostPortValue);         //test
      System.out.println("Current MuServer Port Number : " + s.getPort() );
//test

      // Creating a MobileCodeGroup called 'shipHandler' and
```

```
      // shipping it to all businesses
      MobileCodeGroup shipHandler = new MobileCodeGroup(s);
      for(int i = 0; (i < 40) && ((businessDestination[ı] != null)); i++) {
        System.out.println("businessDestination = "  + businessDestınation[i] +
//test
                    "\ncustomerSource = "     + customerSource);        //test
        shipHandler.ship(businessDestinatıon[ı], customerSource,
                    classesToBeShıpped, objectsToBeShipped, expId,
TımeCounter);
      }

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

  public Product[] displayQueryResults() {
    return null;
  }

  public void stopParadigmCustomerServer() {
  }
}
class ParadigmCustomerCntrllr{

void main(expId, ifCreateReqs, ifExecute, paradigm,
customerHostId, customerPortId, noOfB, noOfC, noOfMachınes)
{
  2DArrayOfHostId_PortId  CalculateBusinessHostPortId(
           customerHostId, customerPortId, noOfB, noOfC, noOfMachines)
  // Using class CalculateHostPortId, calculate the IP Address and Port of all
  // the ParadigmBusiness that the ParadgimCustomer will send MobileCode.
  // This functıon will return address pairs in a 2D array

  TimeStamp;
  CustomerTimeCounter1;
According to the paradigms -
  new ParadıgmCustomer(customerHostId, customerPortId,
                       busınessHostId, busınessPortId)
  // Will NOT load any muServers. Will ONLY STORE CHostId,etc..
  // This will create a ParadigmCustomerObject
  if ( IfCreateReqs == yes )
    ParadigmCustomer.InputNewData()
    // Asks what do you want to input eg: CR, R, BI
    // Each paradigm knows what it needs
    // Calls CRHandlers & inputs new CR Data and R Data
  CustomerTımeCounter2;
  if ( IfExecute == no)
    ıf( IfCreateReqs == yes)
      ExpDataHandler.ReportCustomerMeasurements()
      // At thıs point measurements 1 2 3 4 5 can be sent
      // wıth the rest as 0 or NULL
    quit()
  ParadıgmCustomer.LoadDataToMobileCode()
  // Knows what to load into  MobileCode if CR, R, etc
  // At this point we have to start the MuServer (since we
  // need to have a MuServer started to create a 'group' i.e.
  // to be transported)
  CustomerTimeCounter3;
  ParadigmCustomer.PerformQuery()
  // Since the CustomerHostId, CustomerPortId,
2D[BusinessHostId,BusinessPortId]
  // are already there, 1. MobileCode is sent to ParadigmBusiness
  // 2. Performs query there and stores the results in Mobile Code
  // 3. Recıeves the results
```

```
  // Primarily what is in Picco's examples
  CustomerTimeCoutner4;
  ParadigmCustomer.DisplayQueryResults()
  // Mobile Code has query results. They are NOT stored with ParadigmCustomer.
  // Therefore acces results from Mobile Code
  ParadigmCustomer.end()
  // End of ParadigmCustomer object - perhaps cleanup, etc.
  ExpDataHandler.ReportCustomerMeasurements()
  // Sends all the measurements to ExperimentDataHandler
  DisplayMeasurements()
  // Can display measurements that have been sent to the  database

void DisplayMeasurements()
  // Display all measurement taken


package paradigm;

import muCodeExtensions.*;
import data.*;

public interface ParadigmCustomerInterface {

  public void ParadigmCustomer( String CHostIP, int CHostPort,
                          String BHostIP, int BHostPort );
  public void inputNewCData();
  public void loadCDataToMobileCode();
  public void performQuery();
  public Product[] displayQueryResults();
  public void stopParadigmCustomerServer();
}


/*****************************************
 *
 *     Class Product
 *
 *****************************************
 *
 * Product is the class that handles the operations of Product
 *
 *****************************************/

package data;

import java.io.*;

public class Product {
  private int             productId;
  private String    productName;
  private String    manufacturerName;
  private int             itemId;
  private String    processor;
  private int             memory;
  private double    price;
  private int             qualityRanking;
  private int             customerRating;

  // Constructors
  public Product ( int productId, String productName,
        String      manufacturerName, int itemId, String processor,
        int memory, double price, int qualityRanking, int customerRating ){
```

```
    this.productId = productId;
    this.productName = productName;
    this.manufacturerName = manufacturerName;
    this.itemId = itemId;
    this.processor = processor;
    this.memory = memory;
    this.price = price;
    this.qualityRanking = qualityRanking;
    this.customerRating= customerRating;
  }

  //toString()
  public String toString() {
    String ProductList = productId + "\t" + productName + "\t" +
        manufacturerName + "\t" + itemId + "\t" + processor + "\t"+
        memory + "\t" + price + "\t" + qualityRanking + "\t"+
        customerRating;
    return ProductList;
  }

/*
  //toString()
  public String toString() {
    String ProductList = " Product Id : " + productId +
                         " Product Name : " + productName +
                    " Manufacturer : " + manufacturerName +
                    "\n Item Id : " + itemId +
                    " processor: " +  processor +
                    " Memory : " + memory +
                    "\n Price : " + price +
                    " Quality Ranking : " + qualityRanking +
                    " CustomerRating : " + customerRating + "\n\n";
            return ProductList;
  }
*/

  //Accessors
  public int getProductId(){
    return productId;
  }

  public String getProductName(){
    return productName;
  }

  public String getManufacturerName(){
    return manufacturerName;
  }

  public int getItemId(){
    return itemId;
  }

  public String getProcessor(){
    return processor;
  }

  public int getMemory(){
    return memory;
  }

  public double getPrice(){
    return price;
```

```
  }

  public int getQualityRanking(){
    return qualityRanking;
  }

  public int getCustomerRating(){
    return customerRating;
  }

  // Writing BIData to File
  public void addProductToBIDataFile ( int productId, String productName,
        String      manufacturerName, int itemId, String processor,
        int memory, double price, int qualityRanking, int customerRating ){
    // Load the BIData.txt file with the given input BIData

    try{
       BufferedWriter out = new BufferedWriter(new
FileWriter("data/BIData.txt",true));
       out.write(productId + "\t" + productName + "\t" +
        manufacturerName + "\t" + itemId + "\t" + processor + "\t"+
        memory + "\t" + price + "\t" + qualityRanking + "\t"+
        customerRating + "\n");
       out.close();
    } catch (IOException e) { }
  }

  // Setters
  public void setProductId(int productId){
    this.productId = productId;
  }

  public void setProductName(String productName){
   this.productName =  productName;
  }

  public void setManufacturerName(String manufacturerName){
    this.manufacturerName = manufacturerName;
  }

  public void setItemId(int itemId){
    this.itemId = itemId;
  }

  public void setProcessor(String processor){
    this.processor = processor;
  }

  public void setMemory(int memory){
    this.memory = memory;
  }

  public void setPrice(double price){
    this.price = price;
  }

  public void setQualityRanking(int qualityRanking){
    this.qualityRanking = qualityRanking;
  }

  public void setCustomerRating(int customerRating){
    this.customerRating = customerRating;
  }
```

```
}//EOF


/****************************************
 *
 *     Class RHandler
 *
 ****************************************
 *
 * RHandler is the class that handles the operations of the
 * Recommend (R)  Process
 *
 ****************************************/

package data;

public class RHandler{

  // queryAlgorithm() takes the CRDataHandler Object, BIDataHandler Object
  // and 'indexResults', which contains the index number of  records in
BIData.txt.
  // Returns indexResults[maxNoOfResults]
  public int[] queryAlgorithm(CRDataHandler CRDataOp, BIDataHandler BIDataOp,
                              int[] indexResults ){

    int MAX_BIData_SIZE = 6;

    int[] indexPrice = new int[MAX_BIData_SIZE ];
    BIDataOp.getRecordsFromBIData( indexPrice,CRDataOp.getPrice() );
    int[] indexMemory = new int[MAX_BIData_SIZE];
    BIDataOp.getRecordsFromBIData( indexMemory, CRDataOp.getMemory());
/*
    System.out.println("The indexPrice[] values are:"); //test
    for (int i = 0; i < MAX_BIData_SIZE; i++)        //test
      System.out.println(indexPrice[i]);             //test

    System.out.println("The indexMemory[] values are:"); //test
    for (int i = 0; i < MAX_BIData_SIZE; i++)         //test
      System.out.println(indexMemory[i] + " " );      //test
*/
    int k =0;
    for (int i = 0; i < MAX_BIData_SIZE; i++) {
      for (int j = 0; j < MAX_BIData_SIZE; j++) {
        if ( (indexMemory[i] == indexPrice[j]) && (k < indexResults.length) ) {
          indexResults[k] = i;
          k++;
          break;
        }
      }
    }

/*
    System.out.println("The indexResults[] values are:"); //test
    for (int i = 0; i < indexResults.length; i++)      //test
      System.out.println( indexResults[i] );           //test
*/
    return indexResults; //change when moved
  }
}
```

```
/*****************************************
 *
 *     Class SendBack
 *
 *****************************************
 *
 * Send Back is the Mobile Code that is sent back from the
 * Business to the Customer after the processing at the Business
 * is complete.
 *
 *****************************************/

package muCodeExtensions;

import mucode.*;
import mucode.abstractions.*;
import java.io.*;
import java.lang.reflect.*;

public class SendBack extends Thread
       implements GroupHandler, Serializable {
  private String results;
  private transient MuServer server = null;
  private noObjClass noObj  = new noObjClass();

  public SendBack() {
    System.out.println("sb: SendBack parameterless constructor ");
    System.out.println("sb: SendBack MuServer is " + server );//test
  }

  public SendBack(MuServer server) {
    this.server = server;
  }

  public SendBack(String results) {                          // ???
    this.results = results;
  }

  public void shipBack(String destination, String[] classNames,
                       Object[] objectNames)
         throws MuCodeException, ClassNotFoundException, IOException {
    if(server == null)
      this.server = MuServer.getServer();

    // Checking if control is in ship()                                  //test
    System.out.println("sb: In shipBack() of SendBack" );                 //test
    System.out.println("sb: SendBack's MuServer is " + server );          //test

    // Creating a 'group' with 'Handler' and 'Root' as  "shipBackClassName"
    // shipBackClassName can be invoked in unpack if this(i.e. SendBack)
    // can be a ubiquitous class
    String shipBackClassName = this.getClass().getName();
    System.out.println("sb: SendBack's Handler and Root are "+
      shipBackClassName);
//test

    // Setting 'noObj.no' to the reqd value before creating the group
    noObj.no = objectNames.length;
    System.out.println("sb: Before packing, noObj.no : " + noObj.no );
//test

    // Group group = server.createGroup("SendBack", shipBackClassName);
    Group group = server.createGroup(shipBackClassName, shipBackClassName);
```

```
    group.addObject("SendBackHandlerLabel", this);
    group.addObject("noObj", noObj);                                    //test
    group.setDynamicLinkSource(group.getSource());
    group.setSynchronousTransfer(false);

    // Adding Classes in the group
    group.addClasses(classNames);

    // Adding Objects in the group
    // NOTE - The classes of the objects will also be added here individually
???
    for (int i = 0; i < objectNames.length; i++) {
      String objectKey = "Obj" + i;
      System.out.println("sb: The name of " + objectNames[i] + " is " +
objectKey); //test
      group.addObject(objectKey, objectNames[i]);
      group.addClass(objectNames[i].getClass());
    }
    System.out.println("sb: The noObj.no are " + noObj.no );   //test

    // List the classes in the group
    String[] groupClasses = group.getClasses();                  //test
    System.out.println("sb: Group Classes to be SHIPPED are");  //test
    for (int i =0; i < groupClasses.length; i++)                 //test
      System.out.println("  " + groupClasses[i]);                //test

    // REAL DEAL - finally
    group.ship(destination);
  }

  // REAL DEAL - continuation... after arriving at destination
  public Thread unpack(Group group)
        throws MuCodeException {

    System.out.println("sb: In unpack() of MobileCodeGroup");   //test
    SendBack destnHandler =(SendBack)group.getObject("SendBackHandlerLabel");
    server = group.getServer();
    System.out.println("sb: The thread after unpacking is " +
Thread.currentThread()); //test
    noObjClass noObj = (noObjClass)group.getObject("noObj");            //test


    // List the classes in the group
    String[] groupClasses = group.getClasses();                  //test
    System.out.println("sb: Group Classes that are SHIPPED are");//test
    for (int i =0; i < groupClasses.length; i++)                 //test
      System.out.println("sb:  " + groupClasses[i]);             //test

    // List the objects in the group                             //test
    System.out.println("sb: The noObj.no are " +noObj.no ); //test
    System.out.println("sb: The objects are ");                  //test

    for (int i = 0; i < noObj.no; i++) {                        //change
      String objectKey = "Obj" + i;                              //test
      Object objectListed = group.getObject(objectKey);          //test
      System.out.println("sb:  " + objectListed);                //test
    }
  this.start();
  return null;
  }


  public void run(){
```

```
    System.out.println("sb: In the run of SendBack");       //test
    System.out.println("sb: Reaching the SendBack and bringing back"
                       + " required objects and classes");       //test
    System.out.println("sb: Do the required processing if necessary");//test
  }
}
```

REFERENCES

N. R.Adam, O. Dogramaci, A. Gangopadhyay, Y. Yesha. *Electronic Commerce Technical, Business, and Legal Issues*, Prentice Hall 1999.

Amazon.com, Inc. www.amazon.com, 2005.

R. Balakrishnan. A Service Framework Specification for dynamic e-services interaction, In *Proceedings of Fourth International Enterprise Distributed Object Computing Conference*, 2000.

M. Baldi, G. P. Picco. Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications, In *Proceedings of the 20th International Conference on Software Engineering*, Kyoto, pages 146 – 155, Japan, 1998.

W. Binder. Using Mobile Agents for Software Distribution and Maintenance: Autonomous Stations Capable of Securely Executing Dynamically Uploaded Applications, In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, vol. 00, pages 352, 2002.

J. Bredin, D. Kotz, D. Rus. Economic Markets as a Means of Open Mobile-Agent Systems, In *Workshop on Mobile Agents in the Context of Competition and Cooperation*, Seattle, USA, 1999.

T Campbell. Insider Secrets to Successful eBay Trading, 2000.

A. Carzaniga, G. P. Picco, G. Vigna. Designing Distributed Applications with Mobile Code Paradigms, In *Proceedings of the 19th International Conference on Software Engineering*, pages 22-32, ACM Press, 1997.

COMERGENT. 2004 E-Commerce Survey Benchmarking of Best Practices, In *A survey of leading manufacturers, distributors and retailers,* 2004.

G. Cugola, C. Ghezzi, G. P. Picco, G. Vigna. Analyzing Mobile Code Languages, In *Mobile Object Systems: Towards the Programmable Internet*, pages 94-109, February 1997.

A. Datta, M. Hauswirth, K. Aberer. Beyond "Web of Trust": Enabling P2P E-commerce, In *CEC 03, IEEE Conference on E-Commerce*, pages 24-27, Newport Beach, June 2003.

W. S. Davis and J. Benamati. *E-Commerce Basics*, Pearson Education, Addison Wesley, 2003.

eBay Inc. www.ebay.com, 2005.

A. Fuggetta, G. P. Picco, G. Vigna. Understanding Code Mobility, In *IEEE Transactions on Software Engineering*, Vol. 24, No. 5, pages 342-361, MAY 1998.

R. S. Gray, G. Cybenko, D. Kotz, R. A. Peterson and D. Rus. D'Agents: Applications and Performance of a Mobile-Agent System, *Software--Practice and Experience*, 32(6):543-573, May, 2002.

Half.com Inc. www.half.com, 2005.

HP. Supply Chain Management for the adaptive enterprise. The innovation, success, and vision of HP's Global Supply Chain, 2004.

M. Merz, K. Müller-Jones, W. Lamersdorf. Agents, Services, and Electronic Markets: How do they Integrate? In *IFIP/IEEE International Conference on Distributed Platforms*, Dresden, 1996.

H. Qi, F. Wang. Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks, In *The 13th International Conference on Wireless Communications*, vol. 1, pages 147-153. Calgary, Canada, July, 2001.

F. Sandakly, P. Poyet. Java and Code Mobility in B&C Applications: A Case Study.

C. G. Thomas, R. Oppermann. Supporting Information Consumers by Search Agents in the World-Wide Web.

M. Kezunovic, X. Xu. Mobile Agent Software Applied in Maintenance Scheduling, IEEE, 2001.

D. B. Lange, M. Oshima. Mobile Agents with Java: The Aglet API, In *Mobility: processes, computers, and agents,* pages 494 – 512, ACM Press, New York, 1999.

K. C. Laudon, C. G. Traver. *E-commerce business, technology society*, page 7, Addison Wesley, 2002.

J. Lee, K. Siau and S. Hong. Enterprise Integration with ERP and EAI, In *Communications of the ACM,* volume 46, issue 2, pages 54-60, 2003.

D. Luckham, A. Manens, S. Bhansali, W. Park, S. Daswani. Modeling and Causal Event Simulation of Electronic Business Processes, In *ACM Conference on Electronic Commerce (EC'03)*, San Diego, U.S.A., 2003.

M. Merz, W. Lamersdorf. Agents, Services, and Electronic Markets: How do they Integrate?, In *IFIP/IEEE International Conference on Distributed Platforms*, 1996.

Mobile Agent List. http://draco.cis.uoguelph.ca/link.html.

Napster, LLC. www.napster.com, 2005.

C.M. Parker and Swatman. Web-TRECS: The Design and Use of an E-commerce Business, In *Quarterly Journal of Electronic Commerce*, Vol 1, No 1, pp. 77-88, Greenwich, 2000.

G. P. Picco. Mobile Agents, In *Proceedings of the 2nd International Workshop on Mobile Agents 98 (MA'98)*, Springer, Lecture Notes on Computer Science vol. 1477, pp. 160-171, Stuttgart (Germany), K. Rothermel and F. Hohl eds., 1998.

G. P. Picco. A Mobile Code toolkit, http://mucode.sourceforge.net/, 2000.

G. Samaras. Mobile Commerce: Vision and Challenges (Location and its Management), In *2002 Symposium on Applications and the Internet*, pages 43-45, 2002.

J. Schafer, J. Konstan, and J. Riedl. Recommender Systems in E-Commerce, In *Proceedings of the First ACM Conference on Electronic Commerce,* pages 158—166, ACM Press, 1999.

Sun MicroSystems. Java™ 2 Platform Standard Ed.5.0, http://java.sun.com/j2se/1.5.0/docs/api/java/lang/System.html#currentTimeMillis(), 2004.

A. Tuzhilin, G. Adomavicius. Integrating User Behavior and Collaborative Methods in Recommender Systems, In *CHI' 99 Workshop Interacting with Recommender Systems*, Pittsburgh, U.S.A., 1999.

United Nations. E-COMMERCE AND DEVELOPMENT REPORT 2002, In United Nations Conference on Trade and Development, New York and Geneva, 2002.

UNITED STATES DEPARTMENT OF COMMERCE. E-Stats, In *www.census.gov/estats*, May 11, 2005.

U.S. Census Bureau, 2003. 2003 Annual Trade Survey, www.census.gov/eos/www/whestats.html , 2003.

Vogt, C. Intractable ERP: A Comprehensive Analysis of Failed Enterprise-Resource-Planning Projects, In *ACM SIGSOFT Software Engineering Notes*, Vol. 27, No. 2, March 2002.

A. G. White. Convergence of Peer-2-Peer Computing (P2P) and Buisness-2-Business (B2B) 'New Economy' Business Models, In *ITtoolbox Supply Chain*, 2001.

WITSA. INTERNATIONAL SURVEY OF E- COMMERCE 2000, The World Information Technology and Services Alliance, 2002.

VITA

Naveen Koneru was born in Hyderabad, India, on November 2nd 1978, the first son of K. B. S. Saibabu and Dr. K. Krishna Kumari. He developed an interest in computers while attending St. Patrick's High School, Hyderabad. He pursued his interest and graduated with a Bachelor of Engineering in Computer Science from Osmania University, Hyderabad, India, in 2001. He entered the Graduate College of Texas State University - San Marcos in September 2001 to pursue a Master of Science degree in Computer Science. While attending Graduate School, he has worked as a Team Lead of Technical Support Specialists in the College of Education, Texas State University – San Marcos. He is interested in Software Engineering and Business Management.

Permanent Address:

House no: N 2/4,

Kakateeyanagar,

Habsiguda,

Hyderabad – 500 007.

INDIA

Email : naveenkoneru@yahoo.com

This thesis was typed by Naveen Koneru in Microsoft Office Word 2003.