



**UNIVERSITY OF NAIROBI**

**SCHOOL OF COMPUTING AND ONFORMATICS**

**PROJECT REPORT**

**TITLE: AUTOMATED MULTIAGENT-BASED INTEROPERATOR**

**BILLING AND PAYMENT SYSTEM (AMBIBPSY)**

**BY**

**MICHAEL ONDEJA ADONGO**

**REGISTRATION NUMBER: P58/70472/2008**

**SUPERVISOR: ERIC AYIENGA**

**Submitted in partial fulfillment for the requirements of Master of Science in Computer Science**

**2012**

**DECLARATION**

I **Michael Ondeja Adongo** do hereby declare that this project report is my original work and to the best of my knowledge, it has not been presented to any other examination body.

Signature:.....

Date:.....8/10/2012

P58/70472/2008

This project report is hereby presented for examination with the approval of the project supervisor.

**Name: Eric Ayienga**

Signature:.....

Date:.....12/10/2012

## **Abstract**

Different operators offering telecommunication services cannot operate in isolation. They have to depend on each other in order to provide all inclusive telecommunication services to their customers. A part from offering services to their customers, operators offer interconnect services to each other. These interconnect services are charged. The current billing and payment for interconnect services is semi automatic. This has resulted in delay of bill and payment processing, prolonged period for settlement of billing and payment disputes.

In order to address the above problems, we intended to develop the automated multiagent-based interoperator billing and payment system (AMBIBPSY) which should have the following main functionalities:

- To receive and transform raw CDR from the POI into a formart suitable for bill generation.
- To generate, store and deliver bill to the distant operator.
- To receive, reconcile and store the received bill from the distant operator.
- To generate and store voucher against the received bill from the distant operator.
- To effect the payment against the reconciled voucher.

AMBIBPSY system has been developed using four agents to enable it accomplish the above functionalities. The four agents include: CollectionAgent, BillingAgent, PaymentAgent and BankingAgent. The multiagent system approach to this study is due to the distributed nature of points of interconnection from which the call detail records are captured. The use of agents has reduced the complexity of the system during development and when the system will require expansion afterwards. Each agent has been built successfully including all the roles it should play. CollectionAgent is used to receive and transform raw CDR from the POI into a formart suitable for bill generation. BillingAgent is used to generate, store and deliver bill to the distant operator. PaymentAgent is used to generate and store voucher against the received bill from the distant operator. BankingAgent is used to effect the payment against the reconciled voucher by implementing EFT. The system is able to generate and send bill to the distant operator. Voucher generation and transmission to the distant operator has been accomplished successfully. Electronic fund transfer between the banking agents has been successful.

The success of this study is a relief to telecommunication operators in handling inter-operator billing and payment. Automatic billing and payment processing makes the process faster due to reduction of human errors. Electronic fund transfer reduces the cost of fund transfer processing in terms of time and money.

## **Acknowledgement**

First and foremost, i wish to thank our loving God for giving me life and opportunity to undertake this project. To conduct a successful project requires a wide range of consultation with people. Some people consulted may prove helpful while others unwilling to participate. In this regard, I sincerely thank my supervisor Mr. Eric Ayienga and Mr. Elisha opiyo for accepting to supervise my work and providing me with the constant support, encouragement and the necessary guidance.

I wish to pass my gratitude to Mr.Kibocha and Mr. Matu of Telkom Kenya for enabling me to obtain the relevant information for the proposal. My appreciations also go to Mr. Omolo and Gichengo of Airtel for their time and information provided. I wish finally to congratulate the entire members of my family for their encouragement and financial support.

# Table of contents

|  |     |
|--|-----|
| DECLARATION .....  | ii  |
| Abstract .....   | iii |
| Acknowledgement .....  | iv  |
| Table of contents.....   | v   |
| LIST OF FIGURES .....  | vii |
| LIST OF ABBREVIATIONS.....   | ix  |
| CHAPTER 1: INTRODUCTION .....  | 1   |
| 1.0 Introduction.....  | 1   |
| 1.2 Purpose of the study.....  | 4   |
| 1.3 Objectives .....   | 4   |
| 1.4 Reasons for using multiagent system to provide the solution..... | 5   |
| 1.5 Significance of the study.....                                   | 5   |
| 1.6 Scope of the study .....   | 5   |
| CHAPTER 2: LITERATURE REVIEW .....                                   | 6   |
| 2.0 Literature review.....   | 6   |
| CHAPTER 3: METHODOLOGY .....   | 9   |
| 3.0 Methodology .....  | 9   |
| 3.1 Requirement gathering:.....                                      | 9   |
| 3.2 Analysis and Design of AMBIBPSY:.....                            | 9   |
| 3.3 Coding and debugging: .....                                      | 9   |
| 3.5 Evaluation of the system:.....                                   | 10  |
| 3.6 Documentation (Report writing):.....                             | 10  |
| CHAPTER 4: ANALYSIS .....  | 11  |
| 4.0 Introduction.....  | 11  |
| 4.1 User requirements .....  | 11  |
| 4.2 System requirements.....   | 11  |
| 4.3 Analysis Using MESSAGE concepts .....                            | 14  |
| 4.4 ANALYSIS MODEL AND VIEW .....                                    | 17  |
| 4.5 Level 0 analysis.....  | 18  |
| 4.5.1 Organization view.....   | 18  |
| 4.5.2 Goal/Task view .....   | 19  |
| 4.6 Level 1 analysis.....  | 25  |
| 4.6.1 Organization view.....   | 25  |
| 4.6.2 Agent/Role view .....  | 25  |
| 4.6.3 Textual agents/roles schemas.....                              | 26  |
| 4.6.4 Agent diagrams .....   | 29  |
| 4.7 Interaction view .....   | 32  |
| 4.8 Domain view.....   | 33  |
| CHAPTER 5: DESIGN.....   | 37  |
| 5.0 Introduction.....  | 37  |
| 5.1 Interaction design.....  | 41  |
| 5.1.1 CDRReconciliation interaction.....                             | 41  |
| 5.1.2 ReconciledCDRDelivery interaction .....                        | 43  |
| 5.1.3 BillReconciliation interaction .....                           | 44  |

|  |     |
|--|-----|
| 5.1.4 ReconciledBillDelivery interaction .....       | 45  |
| 5.1.5 VoucherReconciliation interaction .....        | 47  |
| 5.1.6 Payment instruction delivery interaction ..... | 48  |
| 5.1.7 FundTransfer delivery interaction.....         | 49  |
| 5.1.8 PaymentInformationDelivery interaction .....   | 50  |
| 5.2 The design of database tables .....              | 52  |
| 5.2.1 Collection database:.....                      | 52  |
| 5.2.2 Billing database.....                          | 57  |
| 5.2.3 Payment database.....                          | 62  |
| 5.3 Agent class diagrams .....                       | 72  |
| 5.4 Agent instance acquaintance Object diagram ..... | 73  |
| CHAPTER 6: CODING AND DEBUGGING .....                | 74  |
| 6.0 Introduction.....                                | 74  |
| 6.1 Coding and debugging .....                       | 74  |
| 6.2 Coding and debugging requirements .....          | 74  |
| 6.3 Code samples .....                               | 74  |
| CHAPTER 7: TESTING AND SYSTEM EVALUATION .....       | 75  |
| 7.0 Introduction.....                                | 75  |
| 7.1 Component testing .....                          | 75  |
| 7.2 Functional/integration testing .....             | 75  |
| 7.3 System testing .....                             | 76  |
| 7.4 Acceptance testing .....                         | 76  |
| 7.5 Performance testing .....                        | 76  |
| 7.6 Volume testing .....                             | 77  |
| 7.7 Regression testing .....                         | 77  |
| 7.8 Evaluation testing.....                          | 77  |
| 7.9 Tests on Collection System.....                  | 77  |
| 7.10 Tests on Billing system.....                    | 82  |
| 7.11 Tests on Payment system.....                    | 84  |
| 7.12 Tests on Banking system .....                   | 87  |
| 7.13 Evaluation of the system.....                   | 92  |
| CHAPTER 8: CONCLUSION .....                          | 93  |
| 8.1 Summary .....                                    | 93  |
| 8.2 Challenges.....                                  | 94  |
| 8.3 Recommendation for further study .....           | 94  |
| REFERENCES .....                                     | 95  |
| APPENDICES .....                                     | 97  |
| Appendix 1: Collection system test results.....      | 97  |
| Appendix 2: Billing system test results .....        | 100 |
| Appendix 3: Payment system test results.....         | 103 |
| Appendix 4: Banking system test results.....         | 108 |
| Appendix 5: Code samples .....                       | 112 |
| Appendix 6: Installation manual .....                | 118 |

# LIST OF FIGURES

|  |    |
|--|----|
| Figure 1. 1: Telecommunication network.....                                | 1  |
| Figure 1. 2: Schematic diagram for the current System.....                 | 3  |
| Figure 4. 1: Level 0 organization diagram.....                             | 19 |
| Figure 4. 2: Level 0 organization acquaintance diagram.....                | 19 |
| Figure 4. 3: goal implication diagram.....                                 | 21 |
| Figure 4. 4: level 0 workflow diagram.....                                 | 24 |
| Figure 4. 5: Level 1 organization acquaintance relationships.....          | 25 |
| Figure 4. 6: Level 1 Delegation structure diagram.....                     | 26 |
| Figure 4. 7: CollectionAgent.....  | 29 |
| Figure 4. 8: BillingAgent.....   | 30 |
| Figure 4. 9: PaymentAgent.....   | 31 |
| Figure 4. 10: BankingAgent.....  | 32 |
| Figure 4. 11: Domains under the control of collection agent.....           | 33 |
| Figure 4. 12: Domains under control of billing agent.....                  | 34 |
| Figure 4. 13: Domains under control of payment agent.....                  | 35 |
| Figure 4. 14: Domains under control of banking agent.....                  | 36 |
| Figure 5. 1: Schematic Diagram for the system.....                         | 38 |
| Figure 5. 2: DFD Diagram for the Proposed system.....                      | 39 |
| Figure 5. 3: Conceptual model of the system using agents.....              | 40 |
| Figure 5. 4: CDRReconciliation interaction diagram.....                    | 41 |
| Figure 5. 5: CDRReconciliation interaction sequence diagram.....           | 42 |
| Figure 5. 6: ReconciledCDRDelivery interaction diagram.....                | 43 |
| Figure 5. 7: ReconciledCDRDelivery interaction sequence diagram.....       | 44 |
| Figure 5. 8: BillReconciliation interaction diagram.....                   | 44 |
| Figure 5. 9: BillReconciliation interaction sequence diagram.....          | 45 |
| Figure 5. 10: ReconciledBillDelivery interaction diagram.....              | 46 |
| Figure 5. 11: ReconciledBillDelivery interaction sequence diagram.....     | 46 |
| Figure 5. 12: VoucherReconciliation interaction diagram.....               | 47 |
| Figure 5. 13: VoucherReconciliation interaction sequence diagram.....      | 48 |
| Figure 5. 14: PaymentInstructionDelivery interaction diagram.....          | 49 |
| Figure 5. 15: PaymentInstructionDelivery interaction sequence diagram..... | 49 |
| Figure 5. 16: FundTransferDelivery interaction diagram.....                | 50 |
| Figure 5. 17: FundTransferDelivery interaction sequence diagram.....       | 50 |
| Figure 5. 18: PaymentInfoDelivery interaction diagram.....                 | 51 |
| Figure 5. 19: PaymentInfoDelivery interaction sequence.....                | 51 |

## LIST OF TABLES

|  |    |
|--|----|
| Table 4. 1: CollectionAgent/CollectionRole.....  | 26 |
| Table 4. 2: BillingAgent/Billing Role.....       | 27 |
| Table 4. 3: PaymentAgent/PaymentRole.....        | 27 |
| Table 4. 4: BankingAgent/BankingRole.....        | 28 |
|  |    |
| Table 5. 1: Raw CDR table.....                   | 52 |
| Table 5. 2: Operators table.....                 | 52 |
| Table 5. 3: Routing table.....                   | 53 |
| Table 5. 4: POI table.....                       | 53 |
| Table 5. 5: Route definition table.....          | 54 |
| Table 5. 6: ReconciledCDR table.....             | 54 |
| Table 5. 7: CDRDisputes table.....               | 55 |
| Table 5. 8: Dispute description table.....       | 55 |
| Table 5. 9: TransformedCDR table.....            | 56 |
| Table 5. 10: Collection user table.....          | 56 |
| Table 5. 11: Collection Login table.....         | 57 |
| Table 5. 12: ReconciledCDR table.....            | 57 |
| Table 5. 13: Generated bills table.....          | 58 |
| Table 5. 14: Reconciled bills table.....         | 59 |
| Table 5. 15: Received bills table.....           | 59 |
| Table 5. 16: BillDisputes table.....             | 60 |
| Table 5. 17: Contract table.....                 | 60 |
| Table 5. 18: Billing rate table.....             | 61 |
| Table 5. 19: Billing user table.....             | 61 |
| Table 5. 20: Billing Login table.....            | 61 |
| Table 5. 21: Generated bills account table.....  | 62 |
| Table 5. 22: Received bills table.....           | 63 |
| Table 5. 23: Generated vouchers.....             | 63 |
| Table 5. 24: Operator accounts table.....        | 64 |
| Table 5. 25: Received vouchers table.....        | 64 |
| Table 5. 26: Payment instruction table.....      | 65 |
| Table 5. 27: Payment information table.....      | 66 |
| Table 5. 28: Payment user table.....             | 66 |
| Table 5. 29: Payment Login table.....            | 67 |
| Table 5. 30: Payment instruction table.....      | 67 |
| Table 5. 31: Payment information table.....      | 68 |
| Table 5. 32: Customer accounts table.....        | 68 |
| Table 5. 33: Generated fund transfers table..... | 69 |
| Table 5. 34: Operation table.....                | 70 |
| Table 5. 35: Received transfers table.....       | 70 |
| Table 5. 36: Charge table.....                   | 71 |
| Table 5. 37: FeedBack table.....                 | 71 |
| Table 5. 38: Ordinary transfer table.....        | 71 |
| Table 5. 39: Banking user table.....             | 72 |



## LIST OF ABBREVIATIONS

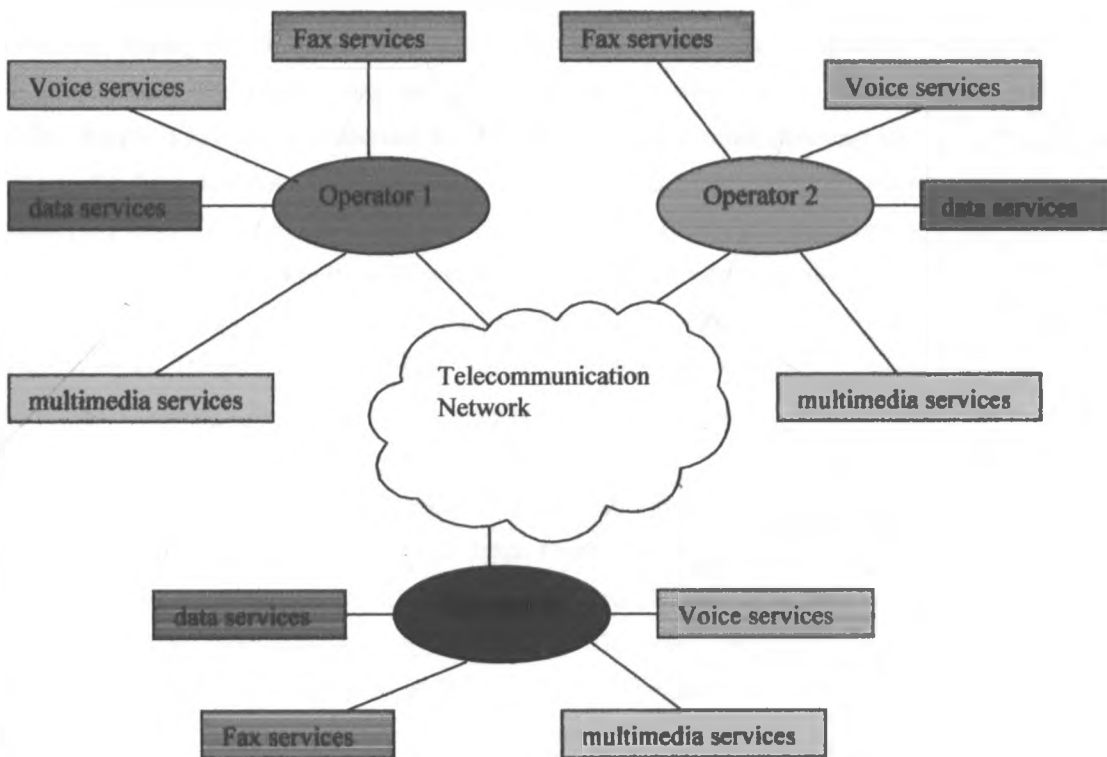
- ACH:** Automated clearing house
- AIR:** Analysis and interpretation of results
- AMBIBPSY:** Automated multiagent-based interoperator billing and payment system
- AV:** Agent/role view
- BA:** Billing agent
- BD:** Bill delivered
- BG:** Bill generated
- BKA:** Banking agent
- BKD:** Banking done
- BLD:** Billing done
- BR:** Bill received
- BRD:** Bill reconciled
- CA:** CDR collection agent
- CEO:** Chief executive officer
- CD:** CDR delivered
- CDR:** Call detail records
- CR:** CDR delivered
- CRR:** CDR reconciled
- CS:** CDR stored
- DFD:** Data flow diagram
- DV:** Domain view
- EFT:** Electronic fund transfer
- FT:** Fund transferred
- FTG:** Fund transfer generated
- GTV:** Goal/task view
- IG:** Invoice generated
- IODRT:** Interoperator dispute resolution team
- IR:** Invoice reconciled
- IT:** Information technology department
- IV:** Interaction view
- JADE:** Java agent development framework
- JDK:** Java development kit
- MAS:** Multiagent system
- MESSAGE:** Methodology for engineering system of software agent

**OP:** Operator  
**OV:** Organization view  
**PA:** Payment agent  
**PID:** Payment instruction delivered  
**PIG:** Payment information generated  
**PGC:** Payment generation completed  
**PING:** Payment instruction generated  
**PINS:** Payment instruction stored  
**PIR:** Payment information reconciled  
**PIRV:** Payment information received  
**PM:** Payment made  
**POI:** Point of interconnection  
**PR:** Payment reconciled  
**RBR:** Reconciled bill received  
**RBS:** Reconciled bill stored  
**RCD:** reconciled CDR delivered  
**RCR:** Reconciled CDR received  
**RCS:** Reconciled CDR stored  
**RFID:** Received fund information delivered  
**RID:** Reconciled invoice delivered  
**RPIS:** Reconciled payment information stored  
**RVPIS:** Received payment information stored  
**RVIS:** Received invoice stored  
**SysAdmin:** System administrator  
**TBSPS:** Telecommunication and banking services provision  
**TFR:** Transferred fund received  
**TFIR:** Transfer fund information received

# CHAPTER 1: INTRODUCTION

## 1.0 Introduction

In telecommunication network, there are various operators which provide variety of telecommunication services to individual customers and other operators. These services range from voice, data, fax, multimedia etc as illustrated in figure 1.1. For service rendered, a bill and payment is expected in return. The process of billing and payment between any two operators is called inter-operator billing and payment (See schematic diagram in figure 1.2 below). In our research, we have concentrate on billing and payment for inter-operator voice services. In this case, the itemized billing data is captured at the point of interconnection (POI) – where different operators are interconnected.



**Figure 1. 1:** Telecommunication network

Itemized billing captures all the details of every call, using a technique called call detail records (CDR). A CDR includes: calling number, called number, type of call (terminating, outgoing or transit), date, start time, end time, call duration, trunk group number, route number, origin, and destination. An operator bills for all calls from other operators terminating into its network.

Each operator has more than one POI for effective and reliable telecommunication networking. In the POI hierarchy, each POI collects CDR then transmits the data to the CDR collection center on some regular

basis. A POI does not retain a copy of the transmitted CDR data. Once the CDR collection center has received CDR data from the POIs, it transforms the data into a format understood by the information technology department (IT). The formatted CDR data is then transmitted online to the IT for analysis and subsequent billing (see data flow diagram in figure 1.3 below).

The IT of each operator (OP) analyses the received CDR data, sums up the call durations for the terminating calls. The sum of the duration is then converted into paid minutes- the unit of billing. Once the units have been converted into paid minutes, representatives from both operators meet to discuss the variations in the paid minutes. If the variation in paid minutes is within the acceptable limit then the operator with more paid minutes is authorized to generate a bill against the difference in paid minutes. The generated bill is then sent to the other operator. The recipient operator verifies the bill then prepares payment against it.

The preparation of the cheque follows the normal procedure of preparing voucher, checking, certification, writing the cheque and eventually signing it by the chief executive officer(CEO) or the financial controller (this process may take some days or even over a week). The ready cheque is then manually delivered to the billing operator for deposit in its bank account (this may take some days). Once the cheque has been deposited, it is manually taken to the clearing house to be cleared to make it ready for payment (this takes about five working days). The process of verifying the paid minutes, preparing the bill, manually sending and receiving the bill; preparing, checking and certifying the voucher; writing, signing and finally delivering the cheque to the billing operator is long, cumbersome, consumes a lot of effort and is prone to errors. The process of taking the cheque to the bank and then to the clearing house in order to make it liquid (although inter-bank money transfer is done electronically using EFT) has some inherent delay in payment.

Incase the variance in the bill is outside the acceptable limits, a dispute is registered. This is done manually by writing and delivering a dispute letter. Once the dispute has been successfully registered, an inter-operator dispute resolution team (IODRT) is convened (this may take some days). The IODRT then follows some formal procedures to resolve the dispute. The task is enormous and ranges from verifying CDRs from a pair of POIs to all pairs of POIs manually. The process is hectic, tiring, time consuming, requires a lot of effort and is also prone to repeated human errors. Once the dispute has been resolved, the normal billing and payment process is followed. Since reporting of disputes for correction is done on monthly basis, similar disputes may arise in the subsequent period. The research is aimed at finding an automated dispute detection, registration, reporting and mitigation. The automated dispute resolution will improve on the required effort and time to mitigate the dispute, reduce human errors, increase efficiency and reliability.

After considering the structure of the current inter-operator billing and payment system, it is observed to be complex, widely distributed and requires a lot of reasoning to achieve its goals effectively and reliably. It is therefore our intention to design and implement a fully automated multiagent-based inter-operator billing and payment system (AMBIBPSY). The system improves on the exiting inter-operator

billing and payment processes. Multiagent system is a system of agents that interact with one another through cooperation, competition, coordination or negotiation [Woldridge, 2002]. An agent can be defined as an object in the environment that perceives and reacts to the states in the environment [Russell and Norvig, 2005].

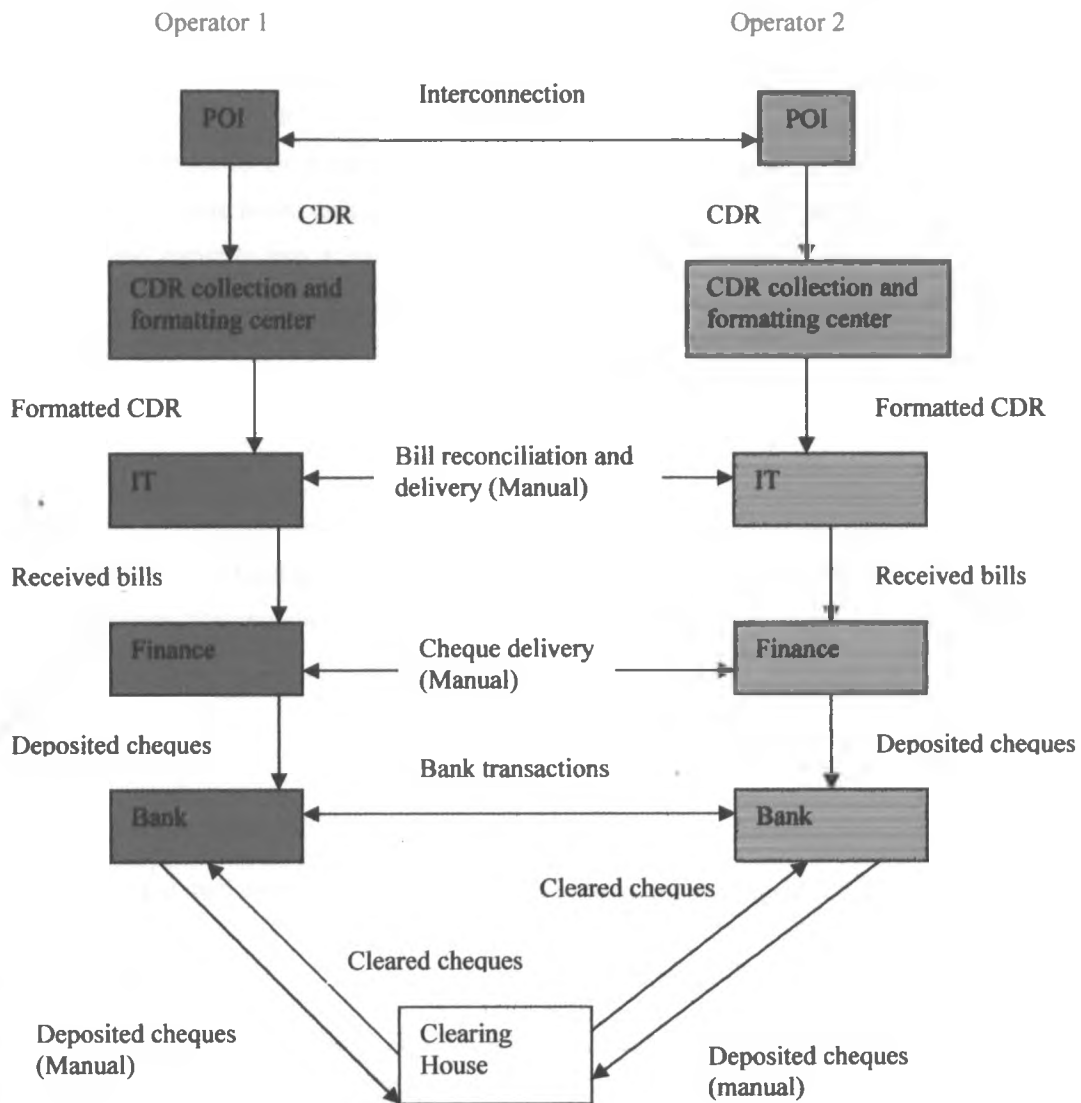


Figure 1. 2: Schematic diagram for the current System

## **1.1 Statement of the research problem**

The current system does not retain a copy of CDR at the POIs after transmission to the billing data collection center. This makes billing data recovery impossible incase data collection system crashes. Semi automatic preparation, verification and delivery of bills require human effort and also results into the delay in bill delivery. The consumer operator manually verifies the received bill to confirm if it is within the acceptable limit according to the signed contract agreement before generating payment. This takes relatively longer time and more effort. The process of preparing voucher, writing, and signing cheque then eventually sending the cheque to the recipient is cumbersome, time consuming and is prone to human errors. Taking the cheque to the bank and finally to the Clearing House to be cleared for payment is expensive in terms of time hence delayed payment. Dispute registration and resolution is manual hence more time required resulting into a number of pending unresolved disputes. Centralization of dispute registration and resolution makes disputes to pile up. AMBIBPSY which has been developed using multiagent system approach has improved on the above stated problems.

## **1.2 Purpose of the study**

We intend to build a fully automated prototype of an Inter-operator billing and payment system using multiagent technology. The main goal of the system is to provide a fully automated billing and payment for voice services between telecommunication operators.

## **1.3 Objectives**

### **1.3.1 Research objectives**

- 1) To produce an overall design of AMBIBPSY
- 2) To design and implement CollectionAgent, BillingAgent, PaymentAgent and BankingAgent.

### **1.3.2 System objectives**

In order to accomplish the proposed study, AMBIBPSY should achieve the following objectives:

- 1) To receive and transform raw CDR from the POI into a formart suitable for bill generation.
- 2) To generate, store and deliver bill to the distant operator.
- 3) To receive, reconcile and store the received bill from the distant operator.
- 4) To generate and store voucher against the received bill from the distant operator.
- 5) To effect the payment against the reconciled voucher electronically.

#### **1.4 Reasons for using multiagent system to provide the solution**

Telecommunication network is widely distributed and complex hence the need for multiagent system (suitability for distributed applications). Data collected at any given point of interconnection (POI) does not depend on other POIs (ability to act autonomously). A single POI is not able to provide the necessary efficiency to collect all the billing data for the entire operator network. The POIs exercise independent control when collecting call detail records (CDR). The use of multiagent system will promote reusability hence reduce cost during expansion and implementation of new POIs. Due to distributed data storage, reliability will be enhanced with the application of multiagent system. The integration of payment and billing processing makes the system to be complicated hence the suitability of multiagent system. Automatic dispute registration and resolution requires reasoning which can best be achieved by agents (ability to work co-operatively in teams). High-level representation of behavior – a level of abstraction above object-oriented constructs is considered most appropriate. There is need to provide flexibility, combining pro-active and reactive behavioral characteristics. The system requires enhancement of real-time performance

#### **1.5 Significance of the study**

Our research outcomes have provided the solutions to the stated research problems. The existing inter-operator billing and payment process is long and is prone to human errors. The errors are due to some parts of the billing and payment process being handled manually by human. The success of the study has produced a fully automated inter-operator billing and payment system for use by telecommunication operators. This has eliminated the manual part inherent in the exiting billing and payment process. The introduction of automatic billing dispute registration and resolution technique has resulted into faster dispute resolving mechanism. Instead of centralized dispute resolution as existing in the current system, the system has a distributed dispute registration and resolution. The provision of distributed data storage has enhanced billing data recovery incase the central billing data collection system crash. Since the entire billing and payment process is now fully automated, clearing house has been eliminated in the payment process in the bill payment processing hence reducing the bill payment period. In overall, the success of the system has resulted in reduced time, effort and cost in inter-operator billing and payment processing.

#### **1.6 Scope of the study**

Our study has been concentrated on analysis, design and implementation of a multiagent system for inter-operator billing and payment. The application has been restricted to billing and payment for voice services.

## CHAPTER 2: LITERATURE REVIEW

### 2.0 Literature review

In order to develop an inter-operator billing and payment system, we have studied some of the existing system for telecommunication billing and payment. The automated-multiagent-based inter-operator billing and payment system (AMBIBPSY) is a fully automated inter-operator billing and payment system. The functionalities offered by the current billing and payment systems have been considered and used to ground the viability of the AMBIBPSY system.

In Kenya, an operator generates a bill which is then sent to the other operator for settlement. The bill can be sent in hard copy or electronically using email. The billed operator writes a cheque which is sent to the billing operator. The cheque is then deposited in the biller's bank. The bank deposits the cheque with the clearing house for clearance. Once the cheque has been cleared, transfer of fund between the banks is done using electronic fund transfer (EFT) [Kenya, 2003]. Payment system in Kenya provides the media of payment which are legally allowed in Kenya. The system has been developed within the legal provisions in Kenya.

According to US Patent 7200551: Automated bill payment system [Senez, 2007], a biller has web site where the bills are posted electronically. A client enters the web site to check the bills and view using a computer connected to the internet. Alternatively, a bill can be sent to the client using email. The bill which is in HTML form is manipulated by the client to facilitate bill payment. The client provides the necessary information on the bill payment form and sends it electronically to the processing web site. Communication is conducted between the client and the processing web site to confirm the information presented in the payment form. Once the information in the payment form has been confirmed, transaction between the processing web site and the bank becomes fully automatic.

Although Automatic payment system has a lot improvement over paper based payment system when applied to general payment, it has some limitations when applied to inter-operator bill payment. Posting of the bill to the biller's web site or sending it to the client using email must be initiated by a human. Extracting the bill from the web site or from the email to effect payment is done by a human. The processing web site has to interact with a human on the client computer to confirm the information on the payment form. The bill payment depends entirely on the decision of the client to make payment, he/she may decide not to act on the form.

In Electronic bill pay system for consumers to generate messages directing financial institutions to pay a biller's bill [Hilt, 2002], a consumer is allowed to pay bills using a network operating according to some preset rules. When the consumer receives either paper or e-mail bill, he/she prepares and transmit electronically, a bill pay order indicating payment details to his/her bank. The bank submits the order to the payment network, which assigns reference number to the biller and forward the payment message to the biller's bank. The consumer's bank debits the consumer's account and the biller's bank credits the biller's bank account.



In this system, the actual payment process is automatic, but it has to be initiated manually by the consumer upon receipt of the bill. This is likely to delay the payment if the consumer becomes reluctant to act on the bill. The biller has no control over the payment, it is the consumer who decide how much to pay and when. A fully automated payment system will ensure that a consumer pays what has been billed since there will be no human intervention.

In Electronic bill presentation and payment system with bill dispute capabilities [Remington et al, 2005], a biller can generate a bill and payment remittance information. The biller submits the bill to the consumer automatically over the internet. The bill is presented in a user interface providing a line-by-line itemized bill with predefined dispute reasons which the consumer can check to challenge particular items on the bill. The consumer can adjust the bill to reflect any disputed amount. The consumer prepares and delivers a direct debit cheque electronically over the internet to the biller. The biller executes the payment instruction by sending it electronically to his/her bank. The consumer entirely controls the payment authorization, specifying the amount to be paid and the date of payment.

One of the limitations of this system is the manual generation of the bill with the aid of a computer. Once the bill has been received by the consumer, he/she has to prepare payment manually using a computer before sending it back electronically to the biller. The biller also has to execute the bill payment manually in order to send it electronically to his/her bank.

Another limitation is that the consumer detects disputes and adjusts payment unilaterally without consulting the biller. This is equivalent to dispute registration as the biller also may not agree with the consumer's decision on the disputed items. The billing and payment process in this aspect is semi automatic in that it cannot continue without human intervention.

Billmax is a billing and provisioning system produced by the ispark Group Inc. [Bilmax, 2008]. This system can be configured to capture billing data, prepare the bill based on usage billing and automatically send the bill to consumer. At the same time, Billmax can be configured to automatically collect funds from the consumers through credit card processor, automated clearing house (ACH) processing, e-cheque processing etc. These tasks are performed with minimal human involvement.

Short coming of Billmax in inter-operator billing and payment is lack of billing dispute registration and reconciliation. Any dispute by the consumer is likely to be registered and mitigated manually. This may result into accumulation of disputed bills and also delay in payment settlement. .

In summary, the reviewed systems have a lot of improvements as far as inter-operator billing and payment is concerned. Payment system in Kenya allows electronic fund transfer (EFT) between banks. Automated bill payment system enables a biller to generate a bill and send it electronically to the consumer. Once the consumer has specified the payment information, inter bank fund transfer becomes fully automatic. The system does not provide for dispute registration and resolution. Electronic bill pay system for consumers to generate messages directing financial institutions to pay a biller's bill allows a consumer to prepare and send payment order electronically to his/her bank. The inter bank fund transfer is done electronically. The system does not have a provision for automatic dispute registration and mitigation.

Electronic bill presentation and payment system with bill dispute capabilities has a provision for a biller to generate a bill and payment remittance information which is then sent to the consumer. There is a provision for a consumer to dispute the bill and unilaterally adjust the payment. The biller may not concur with the disputes and payment adjustment made by the consumer. The inter bank fund transfer is fully automated. In billing and provisioning system (Billmax), there are provisions to electronically collect billing data, generate a bill and collect fund. The only pitfall for Billmax is the lack of automatic dispute registration and resolution.

The main gap left by the reviewed systems is the lack of automatic billing and payment dispute registration and mitigation. This is the major improvement which the proposed automated multiagent-based inter-operator billing and payment system has addressed.

## **CHAPTER 3: METHODOLOGY**

### ***3.0 Methodology***

In order to realize the success of the proposed study, we have followed the following steps: requirement gathering, analysis and design of AMBIBPS, implementation, testing and evaluation of system, documentation and submission of final report. Analysis and design have been done using a multiagent methodology known as MESSAGE.

#### ***3.1 Requirement gathering:***

This has been done by studying the existing manuals together with billing and payment documentation. Some information have been obtained using interviews and questionnaires. We conducted interviews with some stakeholders. Where interviews were not possible, questionnaires were applied.

#### ***3.2 Analysis and Design of AMBIBPSY:***

Analysis and design of automated multiagent-based inter-operator billing and payment system was done using multiagent system (MAS) methodology. In our case, we specifically used Methodology for Engineering System of Software Agent (MESSAGE) methodology. The details of analysis and design of AMBIBPSY are covered in chapter four and five respectively.

#### ***3.3 Coding and debugging:***

In order to implement the automated multiagent-based inter-operator billing and payment system, we used Java Agent Development (JADE) framework. JADE has been used in Java development kit (JDK) version 6.1 environment.

### **3.4 Testing:**

Due to the sensitivity of billing and payment data, it was not possible to use live data for testing purposes. We have therefore used simulated data to test the operation of the developed system.

### **3.5 Evaluation of the system:**

Test results will determine the behavior of the system when subjected to load. The results shall be analyzed and interpreted to establish the correct operation of the system. Based on the interpretation of the results, we are going to recommend further study on the proposed system.

### **3.6 Documentation (Report writing):**

This step started from the beginning of the project up to the end of the project. All pieces of documentation have been refined to come up with a report for submission to the panel of examiners. We have used Microsoft Office XP for documentation and report writing

## **CHAPTER 4: ANALYSIS**

### **4.0 Introduction**

AMBIBPSY system has been developed for automatic billing and payment processing between any two telecommunication operators. It has been developed using multi agent system methodology. Though an operator can provide various services requiring billing and payment, the AMBIBPSY system is based on voice services between the operators. The analysis is categorized into user requirements and system requirements.

### **4.1 User requirements**

User requirements specify what the user wants to achieve from the system. These are the problems which the proposed system should solve. The following are the user requirements

- Automatic collection of call detail record from the point of inter-connection (POI)
- Storage of CDR(raw data from the POIs)
- Formatting of data to suit the billing needs
- Carry out automatic CDR data reconciliation
- Automatically prepare and send the bill to the distant operator
- Reconcile the billing data with the distant operator
- Generate voucher against received bills
- Reconcile the received voucher
- Automatically generate payment instruction against reconciled voucher
- Send the payment instruction to bank and distant operator
- Receive payment information from the distant operator
- Reconciles payment with the distant operators
- Carry out inter-bank money transfer automatically
- Interaction with the administrator and user using GUI interface
- Provision of secure access to the system

### **4.2 System requirements**

System requirements (specifications) include a description of:

- The inputs to the process output
- The operations the system performs for each input
- The output obtained for the corresponding input

Automated multi-agent based interoperator billing and payment system has been developed using four agent programs. These agent programs are: collection agent, billing agent, payment agent and banking agent.

These agents are the processing components of the system. Each agent program requires inputs which are processed to produce outputs. Each agent program has GUI interfaces to enable interaction with human users. The following is the description of each agent program.

### **Collection agent**

The input to this agent program is raw CDR (call detail records) from the point of inter-connection (POI). The format of CDR is a flat text file. Each CDR is composed of date, start time, end time, duration, origination and destination. The collected CDR is translated and stored in a RawCDR table in a collection database. After every one day, the CDR data in the RawCDR table is transformed (formatted) to suite the billing format. The sum of call duration for the whole day for a given operator is computed and stored in a TransformedCDR table in the collection database. When evaluating the sum for daily call duration, there is a different of sum for incoming calls and outgoing calls. Once the sums of incoming calls and outgoing calls have been evaluated and stored, CDR reconciliation process starts. In the CDR reconciliation process the collection agents of any two operators delivers a copy of daily transformedCDR, both incoming calls and outgoing calls. Each collection agent then compares the received copy of the transformedCDR of daily incoming calls and outgoing calls with what it has in its daily TransformedCDR table. If both copies of TransformedCDR are the same or within an acceptable limit, each collection agent from the two operators stores the daily TransformedCDR in a ReconciledCDR table in the collection database.

Incase the copies of the TransformedCDR of the incoming calls and outgoing is outside the acceptable limit, CDR reconciliation is done on the entire daily CDR table. Each collection agent from the two operators extracts CDR for the whole day and sends a copy to the other collection agent. Each collection agent carries on a comparison between its original CDR data and the copy received from the other collection agent. Any abnormal CDR is extracted and stored in a disputable CDR table. The cause for the dispute should be registered for example, date, out of bound if the call date is not the one under evaluation; start time or end time less or more; origin or destination not valid if call is from wrong origin or wrong destination; inroute or outroute invalid if the inroute and outroute are not correct; duration is less or more.

Each case of dispute and its possible solution are stored in a CDR resolution table in the collection database. Each collection agent uses its CDR resolution table to reconcile the CDR record dispute. If all the disputes are resolved, daily CDR sum is computed and daily CDR duration sum reconciliation is done. Incase the whole day CDR reconciliation process cannot solve the problem, a third and a final level dispute resolutions is applied.

Once all the daily disputes are resolved and sum of the daily call duration stored in a ReconciledCDR table, each collection agent transmits daily ReconciledCDR to its own billing agent.

### **Billing agent**

Each operator has one billing agent program, the input data to the billing agent is from all the collection agents belonging to the same operator. The following processes shall be done by the billing agent program.

- Receive reconciled CDR from Collection agent on daily basis.
- The sum of sums of daily call durations of a particular operator is computed to give the total call duration for the entire month. This total for a whole month is converted into paid minutes. Monthly paid minutes is evaluated for both terminating and originating calls
- Reconciliation of the paid minutes is done with the distant billing agent.
- Once paid minutes reconciliation has been done the operator with more terminating paid minutes is allowed to generate a bill for the difference in the two paid minutes
- The generated bill is sent to the distance billing agent for verification and certification. Incase of any discrepancy the bill is rejected and a dispute registered.
- The cause of dispute is recorded and sent back to the bill originating agent.
- Once the bill dispute has been resolved, the reconciled bill is stored in the generated bills table in the billing database of the originating operator. The distant billing agent stores the bill in a received bills table in a billing database.
- Both billing agents from the two operators send the bills to their respective payment agents.

### **Payment agent**

The payment agent program interacts with the billing agent and banking agent programs of the same operator. In addition it also interacts with the payment agent program from the distant operator. Two types of bills shall be received from the billing agent. The first type is the reconciled bill by the local billing agent. The reconciled bill is stored in the ReconciledBill table in the payment database. The reconciled bill is used to verify the received payment from the distant payment agent. The second type of bill is the received bill from the distant billing agent. The received bills are stored in the ReceivedBill table in the payment database. The received bill is used to generate voucher to the distant operator. The generated voucher is stored in the GeneratedVoucher table in the payment database and a copy sent to the distant payment agent for reconciliation. The distant payment agent reconconciles the generated voucher and stores it as received voucher. The reconciled voucher is sent back to the payment agent that initiated the voucher. Once the voucher originating payment agent receives the reconciled voucher, it stores the voucher in its ReceivedVoucher table in the payment database. The payment agent uses the reconciled voucher to generate payment instruction. The generated payment instruction is stored in the GeneratedInstruction table and a copy sent to the local banking agent. The Payment agent also receives and stores payment information from the local banking agent. The payment information contains the payment received from the distant operator.

### **Banking agent**

This is the agent program that performs banking services. It receives input data from the payment agent and the distant banking agent program. It receives the data in the form of payment instructions and received fund transfers. It maintains a banking database which contains various tables according to different services offered to the customers. The payment instructions received from the payment agent is stored in the payment instruction table in the banking database. When the payment instruction has been received from the payment agent, the account details of the originating operator are verified for validity. If the operator's account details are correct, the banking agent generates fund transfer and delivers it to the distant banking agent. The distant banking agent verifies the account details of the originating bank. If the details are correct, the distant banking agent executes the fund transfer transaction. The banking agent stores the received fund transfer and then generates payment information. The generated payment information is sent to the local payment agent as payment received from the distant operator.

### **4.3 Analysis Using MESSAGE concepts**

The analysis of automated multiagent-based interoperator billing and payment system has been done using multiagent system development methodology known as MESSAGE which stands for Methodology for Engineering Systems of Software Agents. The analysis models for the system are represented using the concept and symbols for MESSAGE methodology. The following is a list of various agents which constitute the system, the service /tasks to be performed, goals to be achieved and the collaborators

#### **Collection agent**

##### Tasks/services

1. receive CDR from POI on real time
2. transforms and stores the CDR in a relational database
3. compute sum of call duration of terminating calls per inroute on daily basis
4. compute sum of call duration of outgoing calls per outroute on daily basis
5. reconcile both terminating and outgoing CDRs
6. store the reconciled CDR
7. send the reconciled CDR to the billing agent

#### **Main goal**

Reconciled CDRs sent to the billing agent (RCSBA)

#### **Collaborators**

1. Point of interconnection (POI)
2. Billing agent (BA)
3. Collection database (CDB)
4. Distant collection agent (DCA)



## **Billing agent**

### **Tasks/services**

1. receives reconciled CDR from collection agent
2. store the reconciled CDRs
3. computes the monthly sum of terminating call duration per operator
4. compute the monthly sum of outgoing call duration per operator
5. converts the sum of call duration per month into per minutes
6. generates monthly bill per operator
7. reconcile the generated monthly bill with distant billing agent
8. store the reconciled bill
9. send the reconciled bill to the billing agent
10. receives bill from the distant agent
11. sends both generated bill and received bill to the payment agent

### **Main goal: Billing done**

#### **Sub goals:**

1. bills sent to the payment
2. bill sent to the distant billing agent

### **Collaborators**

1. Collection agent
2. Payment agent
3. Billing database
4. Distant billing agent.

## **Payment agent**

### **Tasks/services**

1. Receives bills from billing agent.
2. Stored the received bills.
3. Generates Payment voucher for bills from the distant billing agent.
4. Reconciles the generated payment voucher with the distant payment agent.
5. Stores the reconciled payment voucher.
6. Receives payment voucher from the distant payment agent.
7. Stores the received payment voucher.
8. Generate and send payment instruction to the banking agent
9. Receives and store payment information from the banking agent.

**Goals**

1. Payment instruction delivered to the banking agent.
2. Payment voucher delivered to the distant payment agent.

**Collaborators**

1. Billing agent.
2. Banking agent.
3. Payment database.
4. Distant payment agent

**Banking agent****Tasks/services**

1. Receives payment instruction from the payment agent.
2. Stores the received payment instruction
3. Generates fund transfer to the distant banking agent.
4. Sends the generated fund transfer to the distant banking agent.
5. Updates the operator's account.
6. Updates the account for the distant bank.
7. Receives fund transfer from the distant banking agent
8. Stores the received fund transfer from the distant banking agent.
9. Sends payment information to the payment agent.
10. Verifies the account details for all account holders.

**Goals:** Has two main goals

1. Payment details sent to the payment agent.
2. Generated fund transfer sent to the distant banking agent.

**Collaborators**

1. Payment agent
2. banking database
3. Distant banking agent.

## **User interfaces**

Each agent in the system should know the identity of the other agents to interact with in pursuit of its goals. For automated multi-agent-based interoperator billing and payment system (AMBIBPSY), the identities of other agents must be configured during system installation or modification. Configuration of the agent identities will ensure that the right information is delivered and received by the right agent. For example, if a new operator joins the existing service, the existing operators will have to include its identity to their databases, at the same time, the new operator will have to configure the identities of the existing operators in the database.

All these configurations and modifications are done by the system administrator. In order to facilitate system configuration and modification, each agent program has at least a configuration interface. This interface can be command-based or graphical user interface (GUI).

During installation, collection agent, billing agent and payment agent can be installed in the same computer as they belong to the same operator. But because of the distributed nature of the telecommunication networks, these agents are more often going to be installed in different computers. The banking agent should be installed in the banking organization computer. It will be configured and maintained by the banking organization computer system administrator.

A part from system configuration and modification done by system administrators, other system users would wish to obtain some information. The interaction between the system users and the system can also be provided by using command-based interface or GUI. For ease of user interaction, user interfaces are provided using GUIs. The feedback from the system due to user's query shall also be in the form of GUIs.

## **4.4 ANALYSIS MODEL AND VIEW**

### **Introduction**

Analysis is carried out to produce system specification. The specification is a model which consists of a collection of views of the system to be developed and its environment. It facilitates communication among the people involved in the system development. AMBIBPSY system has been analyzed accordingly to the following modal views defined in the message using and developing sub-sets of entity and relationships concepts.

### **Organization view**

Represents concrete entities in the system and its environment. These entities include organization, agents, roles and resources. The interaction between the entities is shown by an acquaintance relationship.

### **Goal/Task view**

This represents the goals, tasks, states and relationships between them. UML activity diagram notation is applied.

### **Agent/Role view**

This view describes individual agents and their roles. For each agent/role identified, a scheme diagram shows characteristics such as agent goals, events it needs to sense, resources it controls, tasks it can perform and behavior rules.

### **Interaction view**

The interaction view illustrates interaction between agents/roles. It shows the initiator, the collaborators, the motivator (generally a goal the initiator is responsible for) the relevant information supplied/achieved by each participant the event that triggers the interaction and other effects of the interaction (any other effect becoming responsible for a new goal)

### **Domain view**

This shows the domain specific concepts and relations that are relevant for the system under development e.g. for AMBIBPSY, this view may include bill payment, CDR operator, account payment instruction bank.

The analysis model views have been carried out using level 0 and level 1 work flow diagrams.

## **4.5 Level 0 analysis**

### **4.5.1 Organization view**

Level 0 organization view analysis shows the system to be developed as a black box and its relationships to the entities in the environment (e.g. users stakeholders and resources). The system to be developed is AMBIPSY. Entities in the environment include, users, system administrator, collection database, banking database, information entity (e.g. payment rate, banking rate, rules penalties, ontology and interfaces).

AMBIBPSY is considered as a system in telecommunication and banking services provision system (TBSPS). Two diagrams are used to represent level 0 organization views. The first diagram represents the overall structure and the coarse-relationships existing among the entities of the system. The second diagram shows the acquaintance relationships between the system entities and its environment.

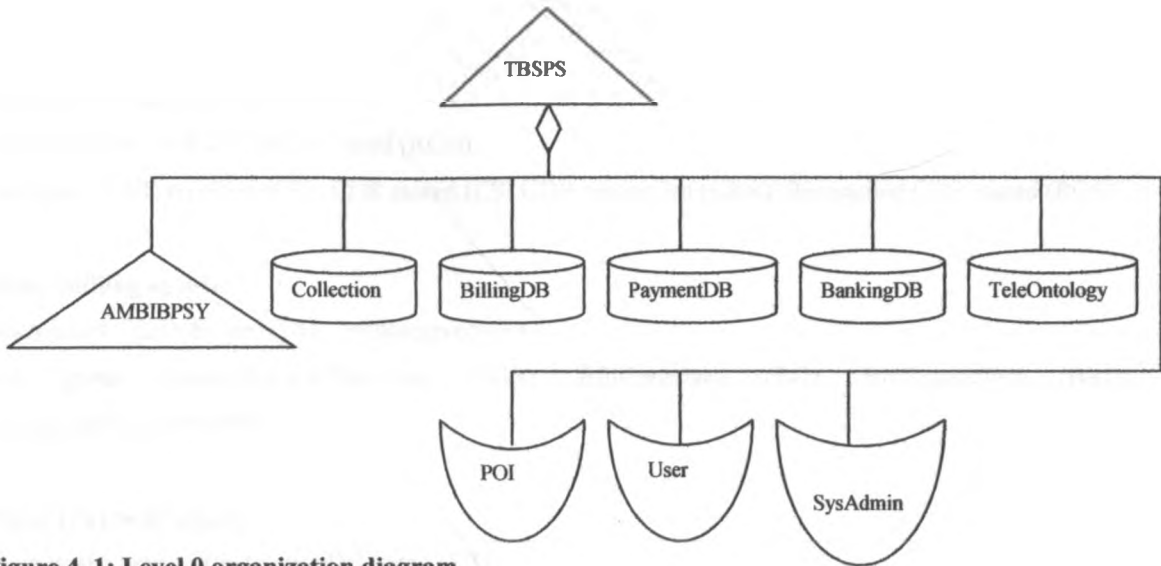


Figure 4. 1: Level 0 organization diagram

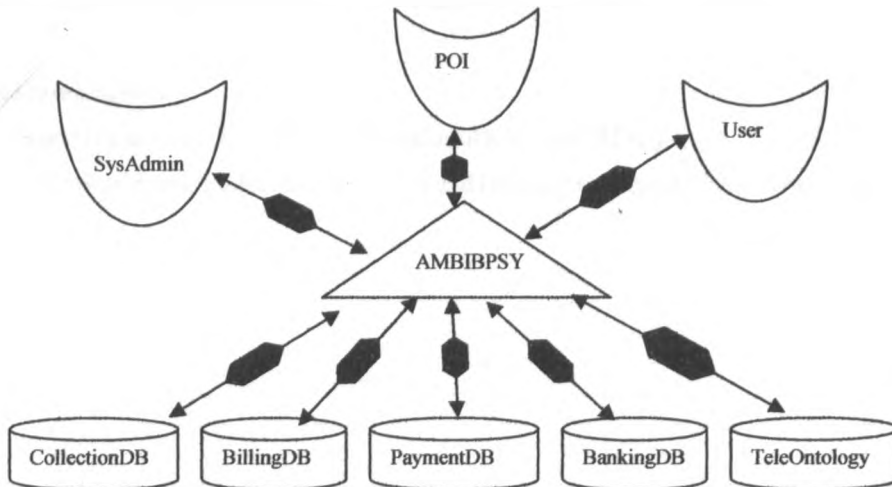


Figure 4. 2: Level 0 organization acquaintance diagram

#### 4.5.2 Goal/Task view

This view represents the main goal to be achieved by the system and the different subgoals which result into the main goal. It also illustrates the various tasks to be performed in order to achieve the stated goals in the form of workflow diagram. The following is an outline of the goals to be achieved by the main system and the roles constituting the system.

**Main system (AMBIBSY)**

Main goal: payment made

**Collector (Collection Agent)**

Main goal: reconciled CDR Delivered (RCD)

Sub goals: CDR received (CR), CDR stored (CS) CDR reconciled (CRR), Reconciled CDR stored (RCS)

**Biller (Billing agent)**

Main goals: BillDelivered (BD), BillReceived (BR)

Sub goals: reconciledCDRReceived (RCR), BillGenerated (BG), BillReconciled (BRD), ReconciledBillStored (RBS)

**Payer (Payment agent)**

Main goals: paymentInstructionDelivered (PID)

PaymentReceived (PR), paymentInfoDelivered (PINFD)

Sub goals: ReconciledBill Received (RBR) PaymentGenerated(PIG), PaymentInfoReconciled (PIR)

ReconciledPaymentInfoStored (RPIS), PaymentInstructionGenerated(PING), PaymentInstruction Stored(PINS), ReceivedPaymentInfoStored (RVPIS), PaymentInfoReceived (PIRV)

**Banker (Banking agent)**

Main goals: FundTransferred (FT), ReceivedFundInfoDelivered (RFID)

Sub goals: PaymentInstructionReceived(PIR), FundTransferGenerated(FTG), TransferredFundReceived (TFR)

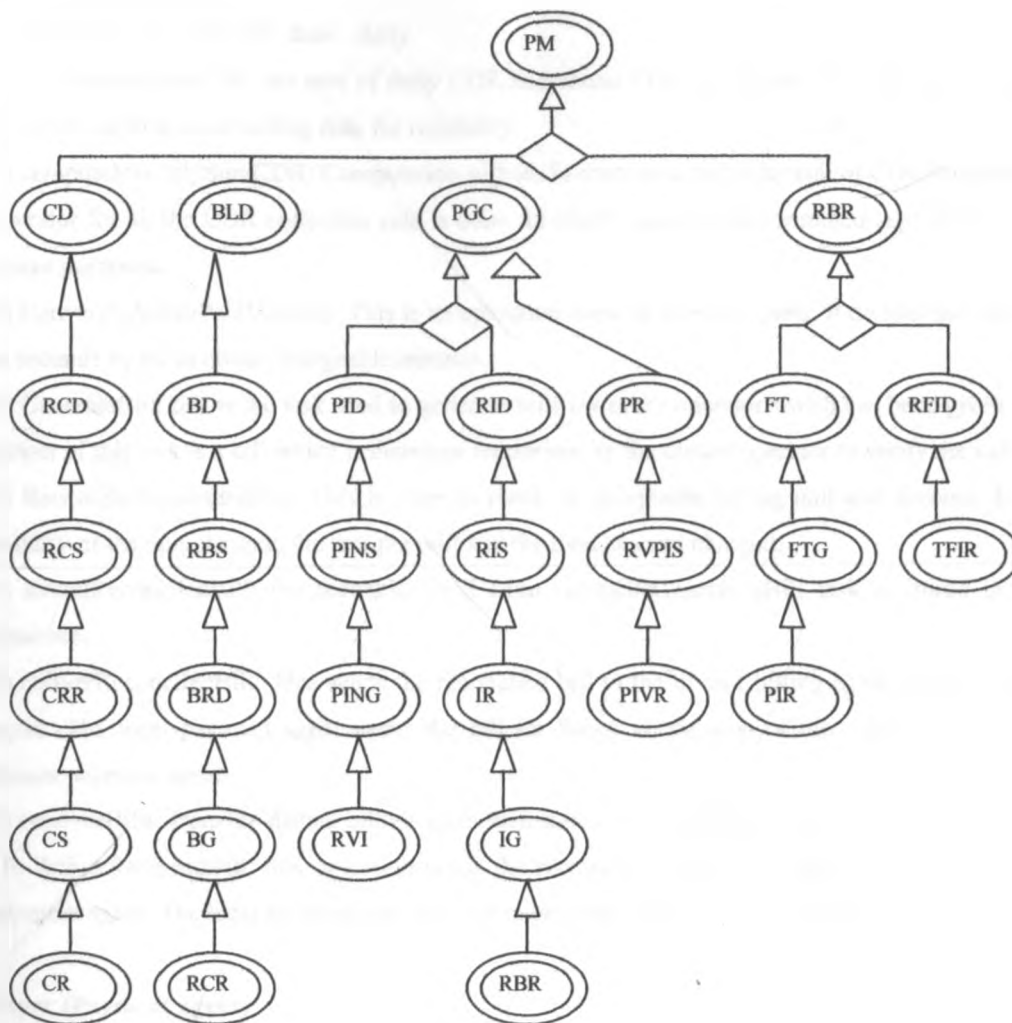


Figure 4. 3: goal implication diagram

### Tasks to be performed by different roles (agents)

#### CDRCollector (Collection agent)

- 1) ReceiveCDR
- 2) transformCDR: Transform and store transformed CDR in the database
- 3) computeSummaryCDR: Done on daily basis for both in route and out route individually
- 4) reconcileCDR: Done once daily on both in route and out route sums individually
- 5) storeReconcileCDR: Both reconciled CDR sum and actual reconciled CDR are stored
- 6) deliverReconcileCDR: Reconciled CDR and the sum of CDR duration are delivered to the billing agent.

### **Biller (Billing agent)**

- 1) receiveReconciledCDR: done daily
- 2) storeReconciledCDR: the sum of daily CDR and actual CDR are stored in a relational database. This is to create duplication of billing data for reliability.
- 3) computeMonthlySumCDR: Computation of both in route and out route sum of CDR duration for a given operator for all the CDR collection role is done. In routes durations are summed separately as well as out routes durations.
- 4) ConvertSumIntoPaidMinutes: This is an operation done on monthly basis. It divides the sum of duration in seconds by 60 to obtain chargeable minutes.
- 5) GenerateBill: this is the task used to generate bills for every operator which has been given service .The output of this task is a bill which is therefore reconciled by the distant operator to verify the validity.
- 6) ReconciledGeneratedBill: This is done to reach an acceptable billing unit and account. It verifies the validity of the unit charged, the rate per unit and the total amount charged.
- 7) storedReconciledBill: the reconciled bill from reconcileGeneratedBill task is stored in a relational database.
- 8) deliverReconciledBill: This sends the reconciled bill to the distant billing agent and the local payment agent .The local payment agent stores the bill for future verification of the payment received from the distant payment agent.
- 9) receivesBills; from the distant billing agent then stores it in a relational database.
- 10) deliverReceivedBill: this operation sends the received bill from the distant billing agent to the local payment agent. The local payment uses this bill to generate payment to the distant payment agent.

### **Payer (Payment agent)**

- 1) receiveBills: This task is used to receive and store bills from the local billing agent .Two types of bills are received; reconciled bill which is used to verify the payment received from the distant payment agent; dreceived bill which is used to generate payment for the distant payment agent,
- 2) generateVoucher: This task gets received bills inputs from the local Biller to generate voucher .The output of the task is the generated voucher which is then sent to the distant payment agent.
- 3) reconciledVoucher: the operation verifies the generated voucher from the distant payment agent. The number of units, rate per unit and the amount for the bill is validated. Any error detected is corrected through negotiation based on the set-up rules in the contract agreement. The output of this operation is a reconciled voucher.
- 4) storedReconciledVoucher: the operation stores the reconciled voucher in a relational database. This reconciled voucher is used to generate payment instruction which is then sent to the local bank agent.
- 5) receiveVoucher: this task is used to receive voucher generated from the distant payment agent. The voucher is stored in a relational database. The information is used to verify the payment information received from the local banker.



6) **GeneratePaymentInstruction**: The payment agent uses this task to generate payment instruction according to the reconciled payment voucher received from the distant payment agent. The payment instruction is sent to the local banking agent to facilitate fund transfer generation.

7) **receivePaymentInfo**: This is an operation used to receive payment information from the local banking agent. This is the payment information as a result of payment voucher received from the distant payment agent. This information is stored in a relational database.

### **Banker (Banking agent)**

1) **receivePaymentInstruction**: This is the operation used to receive payment instructions from the payment agent. The structure of the payment instruction include; instruction I.D, Payer account, Payee account, payment month/payment reason, payer name, payee name, Amount date, This instruction enables the banking agent to generate fund transfer from the payer's account to the payee's account.

2) **storeReceivedPaymentInstruction**: the received payment instruction is stored using this operation. The aim is to keep a record of the received payment instructions for future verification and dispute resolution. The instructions are stored in a relational database.

3) **generateFundTransfer**: After receiving and storing the payment instruction from the payment agent, the banking agent uses this task to generate fund transfer from the payer's account to the payee's account. The generated fund transfer is sent to the distant banking agent. The structure of fund transfer instruction is; Transfer Id, payer Id, payer name, payer account, payee Id, payee name, payee account, invoice number, local banking account, local banking name, assistant bank account, distant bank name, amount

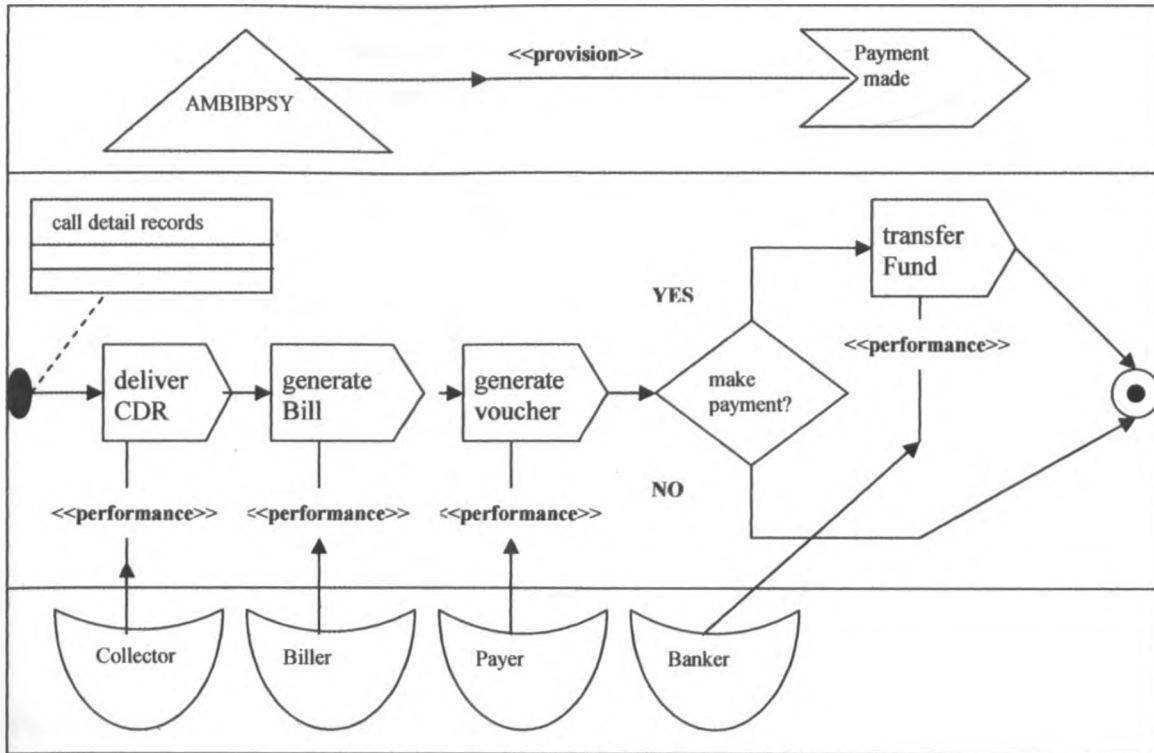
4) **transferFund**: This is the operation used to send the generated fund instruction to the distant banking agent.

7) **receiveTransferredFund**: the task is used to receive the amount transferred from the distant banking agent.

8) **storeReceiveFundTransfer**: The operation is used to store the received fund transfer instruction from the distant banking agent. This is used to maintain records of received fund transfer instruction for future verification incase of any dispute. The structure of received fund transfer instruction is similar to the generated fund transfer instruction.

9) **deliverPaymentInformation**: Once the banking agent has received the fund transfer instruction and completed fund transfer transactions, it uses **deliverPaymentInformation** operation to send the payment received from the distant operator to the local operator. Payment information acts as a proof that the local banking agent has actually credited the operator's account with the amount received from the distant operator's banking agent. The format/ structure include; Information I.D., payer name, payer I.D., payer name, payment month, date, amount, transfer I.D.

**Level 0 workflow diagram**



**Figure 4. 4: level 0 workflow diagram**

This diagram above illustrates how work flows through the main tasks performed by the different roles in order to provide the main service. The origin of the process is the call detail records which are the input to deliverCDR task .The deliverCDR task is composed of sub tasks all of which are performed by the Collector role. generateBill task derives its inputs from the deliverCDR task .The generateBill task is performed by the Biller role .generatePayment Task is performed by payer role .The input to generatePayment task comes from the generateBill task .Once payment has been generated, the banker role performs the transfer fund task in order to accomplish the process.

## 4.6 Level 1 analysis

### 4.6.1 Organization view

The level 1 analysis diagram shows a further refinement of level 0 organization diagram .The diagram shows the different roles participating in the organization, their acquaintance relations and the resources required by the system.

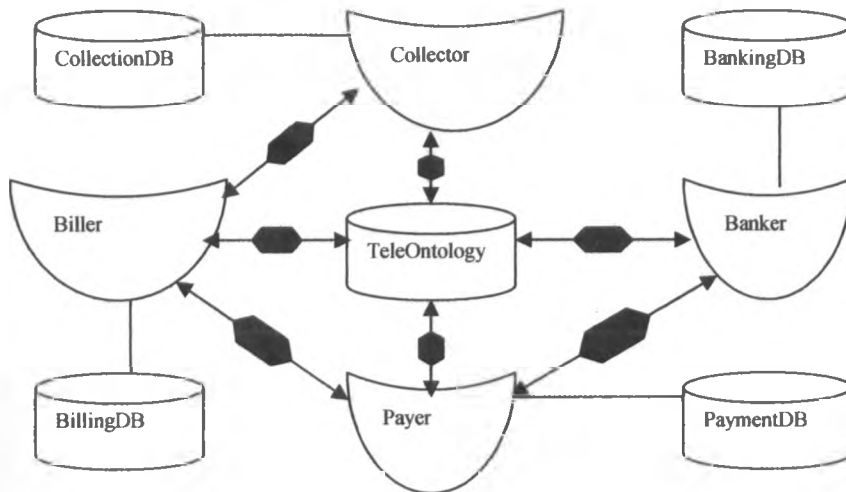


Figure 4. 5: Level 1 organization acquaintance relationships

Figure 4.5 illustrates level 1 of the acquaintance relationships in an organization diagram .The teleOntology can interact directly with all the roles The collector role has a direct interaction with the biller role which also has a direct interaction with the payer role. The payer role has direct acquaintance to the banker role. Each role in the organization is associated with a database from which information can be received or stored .The TeleOntology provides the vocabularies, concepts, predicates and agent actions used by different roles in the system.

### 4.6.2 Agent/Role view

The agent/role view represent the different agents and their roles that the organization to achieve its goals. The agent/role view is described using delegations structure diagrams, workflow diagram and textual/agent role schemas. A delegation structure diagram shows how the sub-goals obtained from decomposing a goal of an organization are assigned to the agents/roles included in the organization.

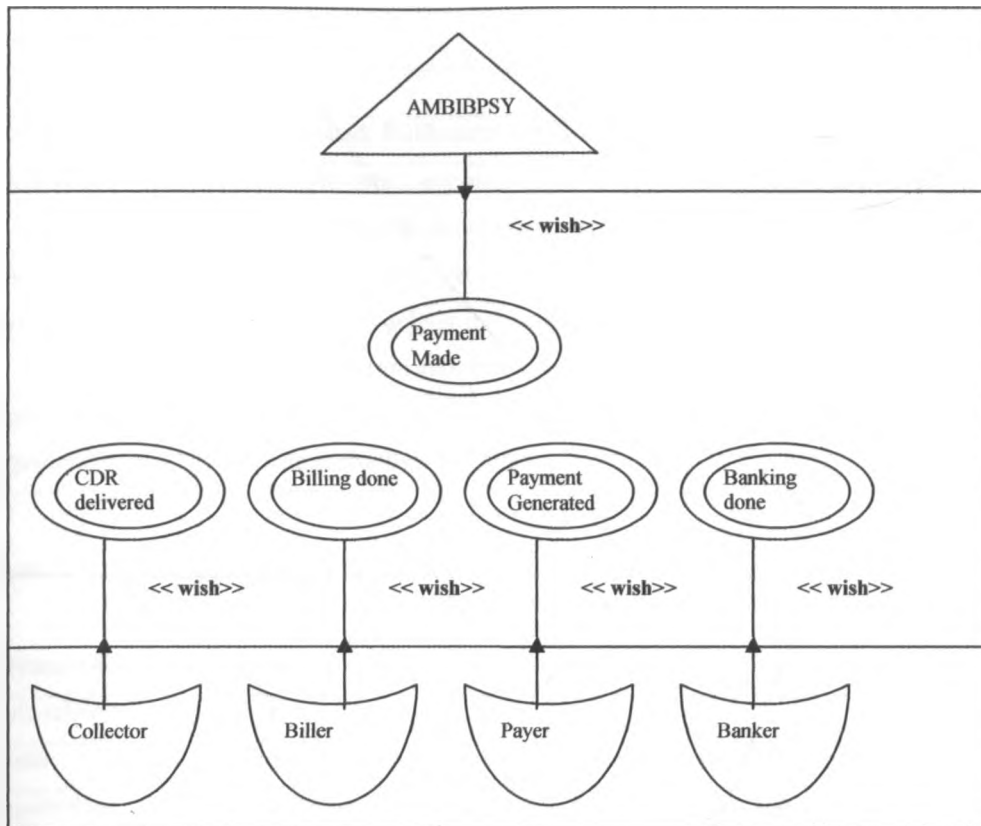


Figure 4. 6: Level 1 Delegation structure diagram

#### 4.6.3 Textual agents/roles schemas

An agent/role schema describes the characteristics of each agent/role identified in the organization. The agents/roles identified in the organization include; CollectionAgent/CollectorRole, BillingAgent/BillerRole, PaymentAgent/PayerRole and BankingAgent/BankerRole. Each agent/role schema is represented using various tables as illustrated below:

##### CollectionAgent/CollectionRole

|                                 |   |
|---------------------------------|---|
| Role schema                     | Collector   |
| Goals                           | CDRReceived, CDRStored, CDRReconciled, ReconciledCDRStored, ReconciledCDRDelivered  |
| Capabilities                    | Learning to detect discrepancies in the CDR, negotiation to help in decision making during CDR reconciliation                                 |
| Knowledge, belief and intension | InRoutes and outRoutes for the CDRAcquantant agent identification, process for CDR reconciliation, end result for each reconciliation process |
| Agent requirements              | The role is played by CollectionAgent which collect, reconcile and deliver CDR  |

Table 4. 1: CollectionAgent/CollectionRole

### BillingAgent/Billing Role

|                                 |  |
|---------------------------------|--|
| Role schema                     | Biller   |
| Goals                           | BillGenerated, BillReconciled, ReconciledBillStored, BillDelivered, BillinDone   |
| Capabilities                    | Learning to detect any error in the reconciled bill, negotiation to help indecision making during bill reconciliation process  |
| Knowledge, belief and intension | A profile of tasks to: receive reconciled CDR, generate bill, reconcile bill, store reconciled bill, deliver reconciled bill. Billing data – units and rate per unit. The order of tasks execution and the expected output of each task. |
| Agent requirements              | The role is played by the BillingAgent which receives and process the billing data to generate and deliver the bill  |

**Table 4. 2:** BillingAgent/Billing Role

### PaymentAgent/PaymentRole

|                                 |  |
|---------------------------------|--|
| Role schema                     | Payer  |
| Goals                           | PaymentInstructionDelivered, ReconciledInvoiceDelivered, PaymentReceived   |
| Capabilities                    | Ability to store and retrieve data from a database, learning to detect any discrepancy in the received voucher, negation during voucher reconciliation.  |
| Knowledge, belief and intension | A profile of tasks to: store received invoice, generate payment instruction, store payment instruction, deliver payment instruction, receive payment information, store payment information, receive reconciled bill, generate invoice, reconcile voucher, store reconciled voucher, deliver reconciled voucher. Data for generating invoice – billing units, billing rate, agent to receive the voucher. The sequence of tasks execution. |
| Agent requirements              | The role is played by the paymentAgent that: <ul style="list-style-type: none"> <li>▪ generates and deliver the voucher to the distant PaymentAgent</li> <li>▪ generates and deliver payment instruction to the BankingAgent</li> <li>▪ receives voucher from the distant PaymentAgent</li> <li>▪ receives payment information from the BankingAgent</li> </ul>  |

**Table 4. 3:** PaymentAgent/PaymentRole

**BankingAgent/BankingRole**

|                                 |   |
|---------------------------------|---|
| Role schema                     | Banker  |
| Goals                           | FundTransferred, ReceivedFundTransferDelivered  |
| Capabilities                    | Ability to store and retrieve data from a database. Ability to generate and deliver fund transfer to the distant BankingAgent. Learning to detect any error in the received fund transfer information. Ability to monitor balance levels of all the accounts maintained by the bank.  |
| Knowledge, belief and intension | A profile of tasks to: receive payment instruction, generate fund transfer, transfer fund, receive transferred fund, delivers received fund transfer information. Account details for other banks and operators. The sequence of executing its tasks.   |
| Agent requirements              | <p>The role is played by BankingAgent that</p> <ul style="list-style-type: none"> <li>▪ receives payment instruction from the PaymentAgent</li> <li>▪ generate and deliver fund transfer information to the distant BankingAgent</li> <li>▪ receives fund transfer information from the distant BankingAgent</li> <li>▪ delivers payment information to the PaymentAgent</li> </ul> |

**Table 4. 4: BankingAgent/BankingRole**

#### 4.6.4 Agent diagrams

The agent is a model that represents each agent constituting AMBIBPSY system. Each diagram represents the role played by the agent, tasks performed, services provided and required, the goals to be achieved and external entities controlled by the agent.

##### CollectionAgent

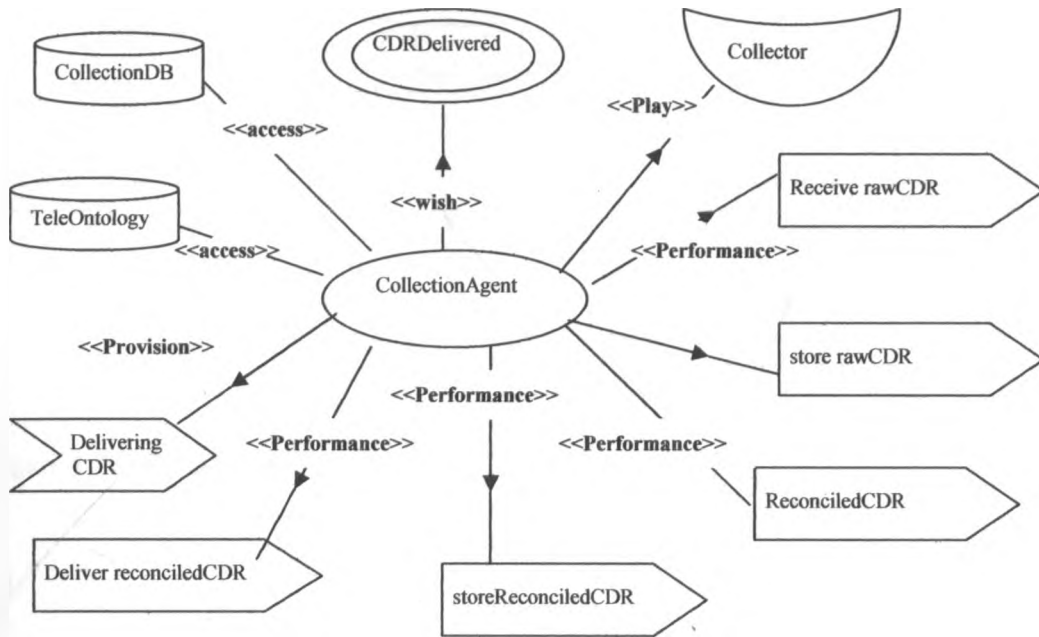


Figure 4. 7: CollectionAgent

## BillingAgent

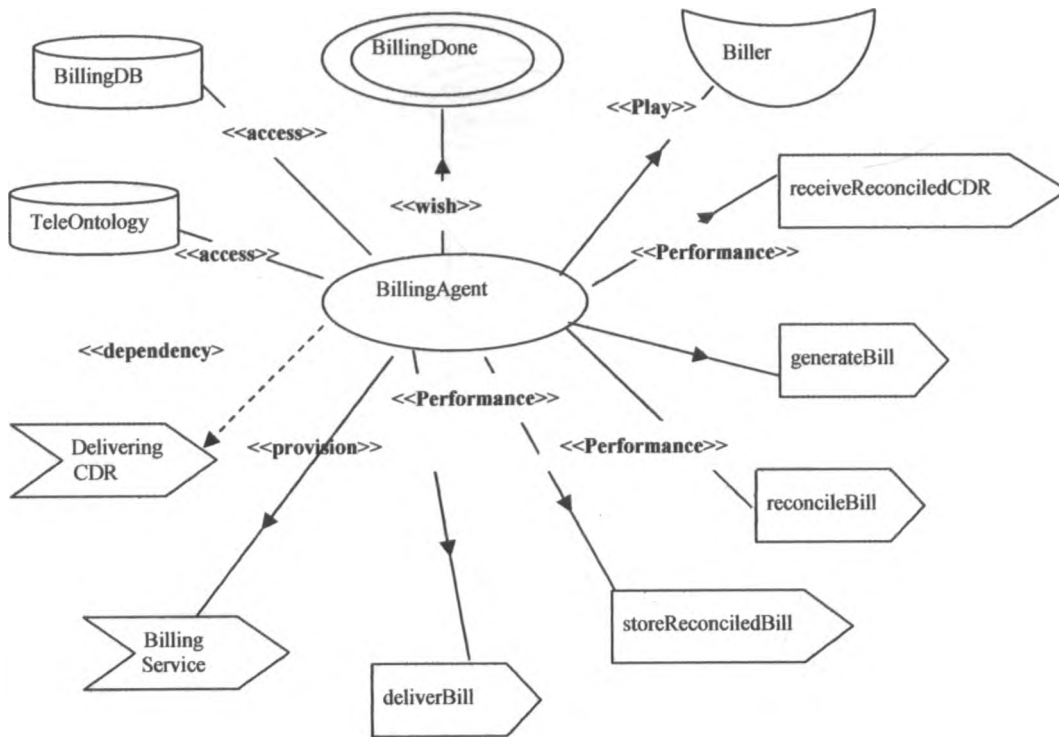


Figure 4. 8: BillingAgent



# PaymentAgent

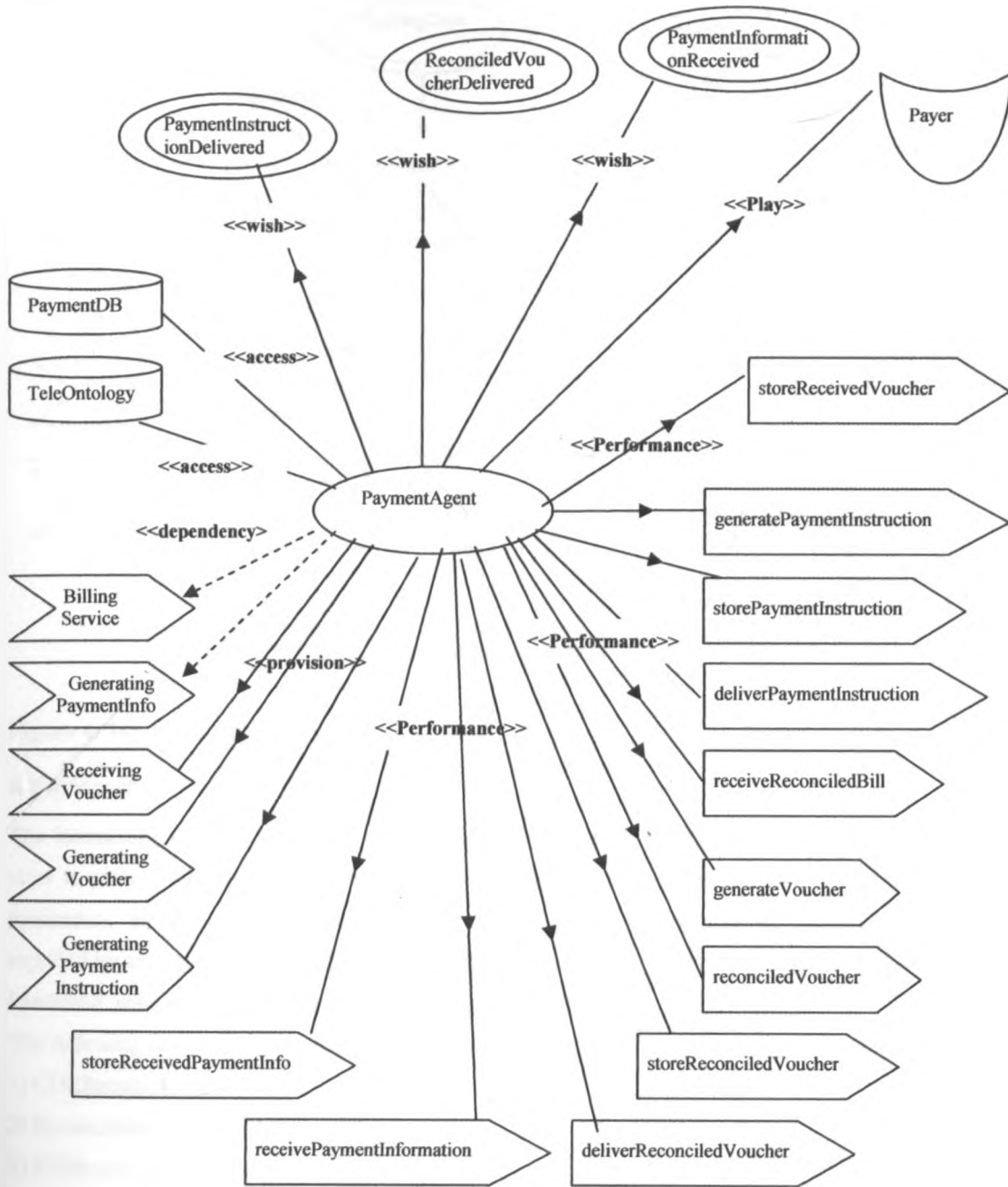


Figure 4. 9: PaymentAgent

## BankingAgent

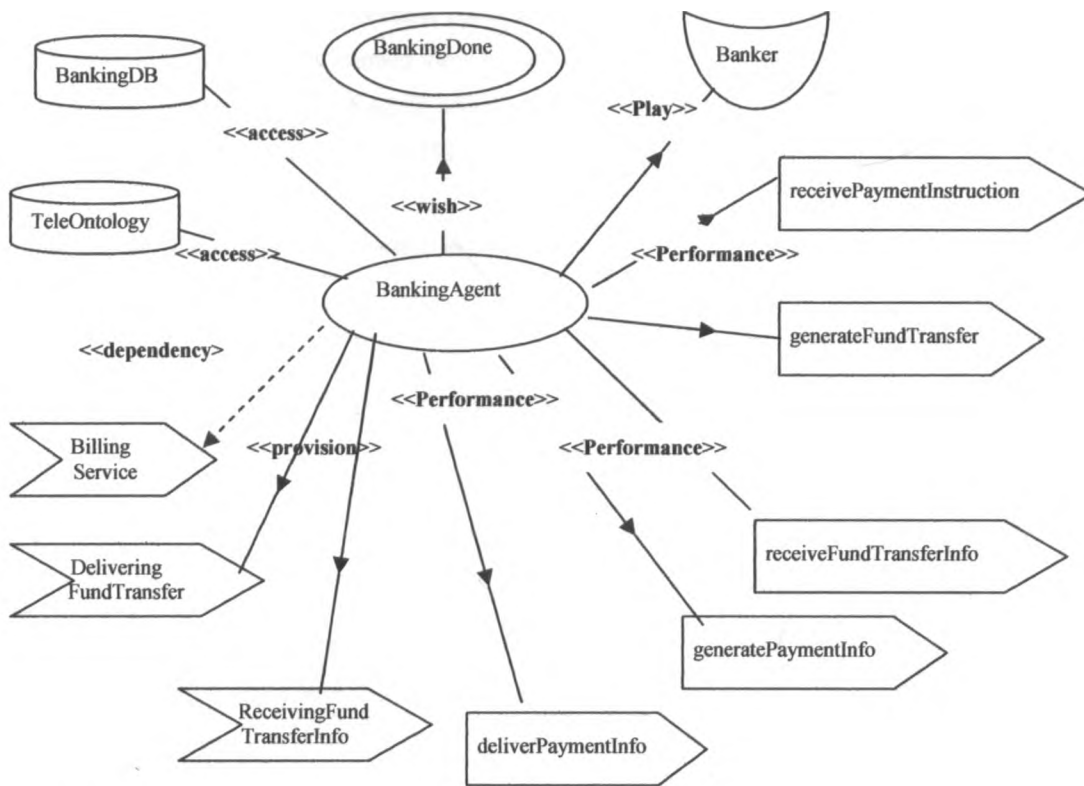


Figure 4. 10: BankingAgent

### 4.7 Interaction view

The interaction view highlights which, why and when agents/roles should communicate. The interaction view is refined through several interaction diagrams. Each interaction diagram shows the initiator, the responders, the motivator (a goal of the initiator) of an interaction and the information achieved and supplied by each participant. An interaction diagram must have at least two participants. Each interaction is illustrated using both interaction diagram and sequence diagram.

The following are the interaction instances considered;

- 1) CDRReconciliation
- 2) ReconciledCDRDelivery
- 3) BillReconciliation
- 4) ReconciledBillDelivery
- 5) VoucherReconciliation
- 6) PaymentInstructionDelivery
- 7) FundTransferDelivery
- 8) PaymentInformationDelivery

The details of these interactions are covered in design (chapter five).

### 4.8 Domain view

The domain view illustrates the domain specific concepts, agent actions and relations relevant for the system under construction. The domain view is represented by UML class diagrams. The classes represent the domain specific concepts and agent actions, named associations represent the relations. The figures below represent the domain view for AMBIBPSY

#### Domains under the control of collection agent

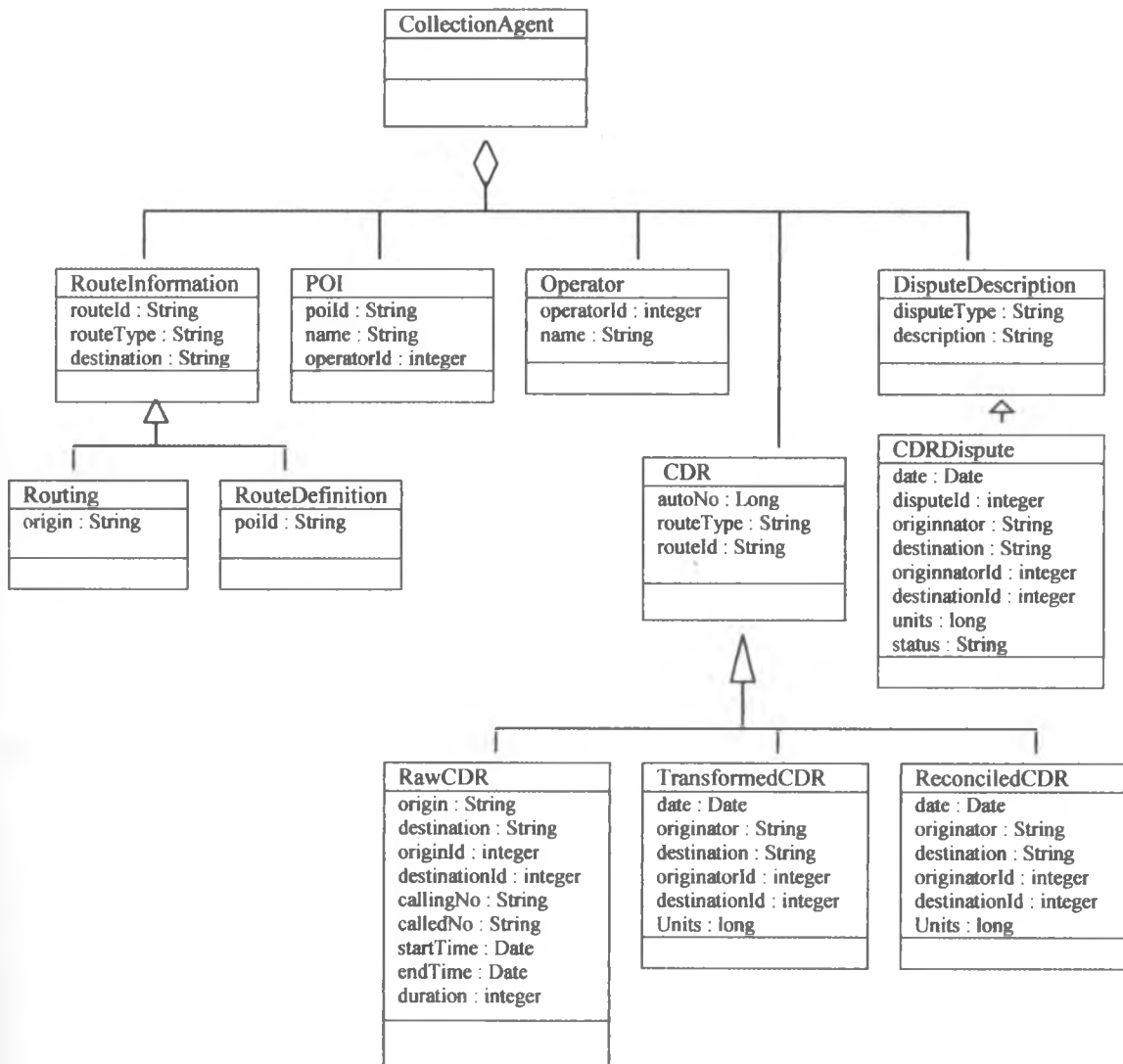
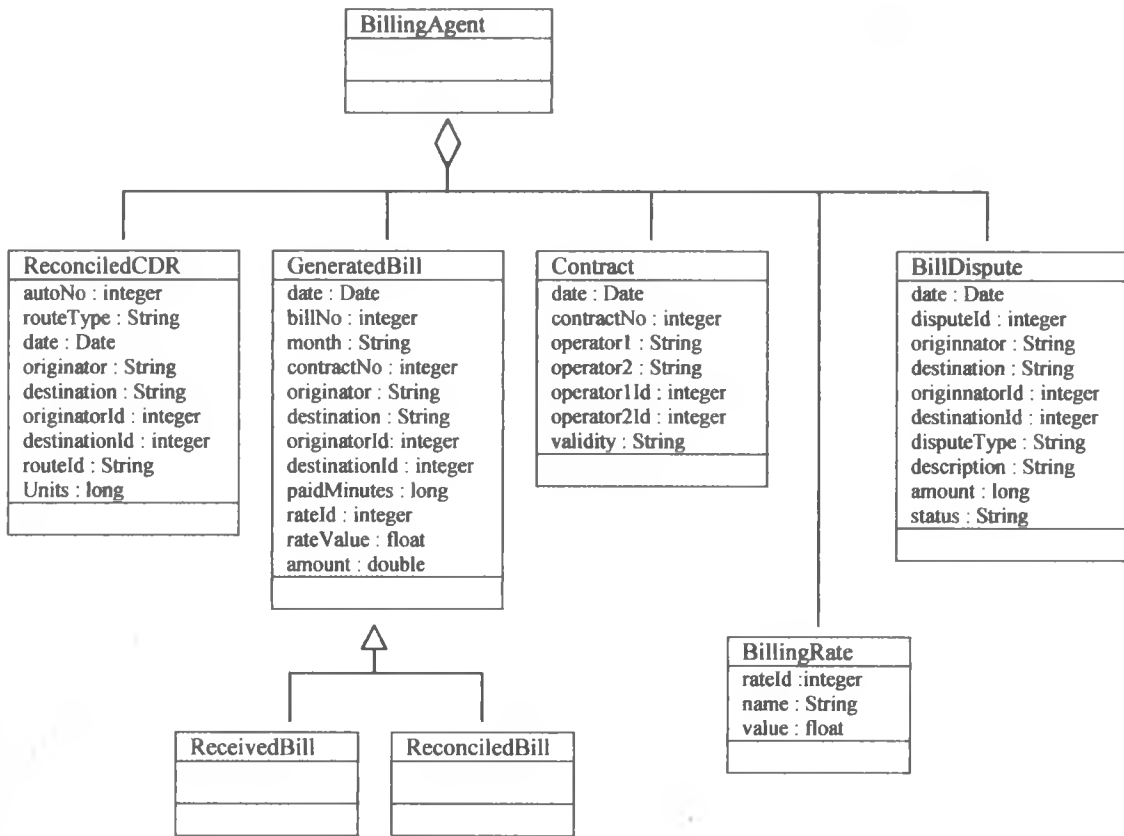


Figure 4. 11: Domains under the control of collection agent

**Domains under control of billing agent**



**Figure 4. 12:** Domains under control of billing agent

## Domains under control of payment agent

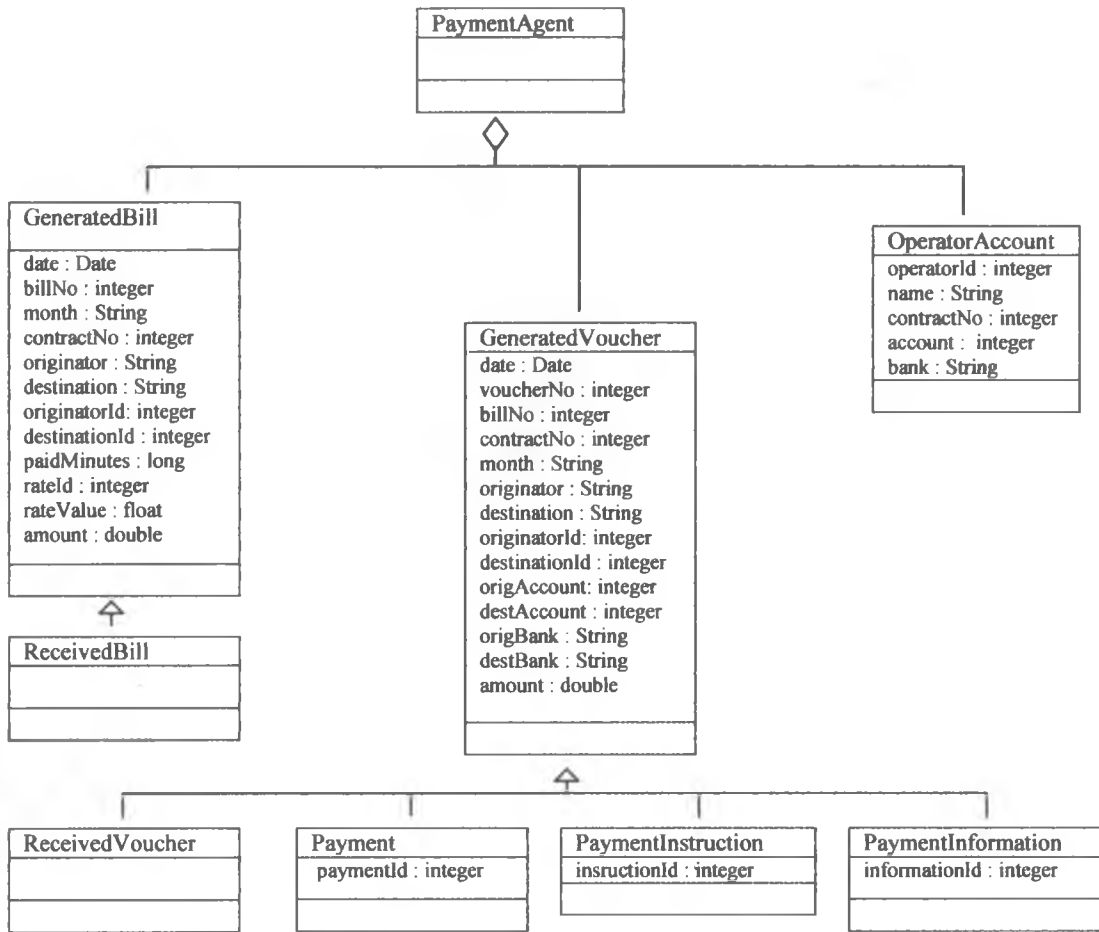
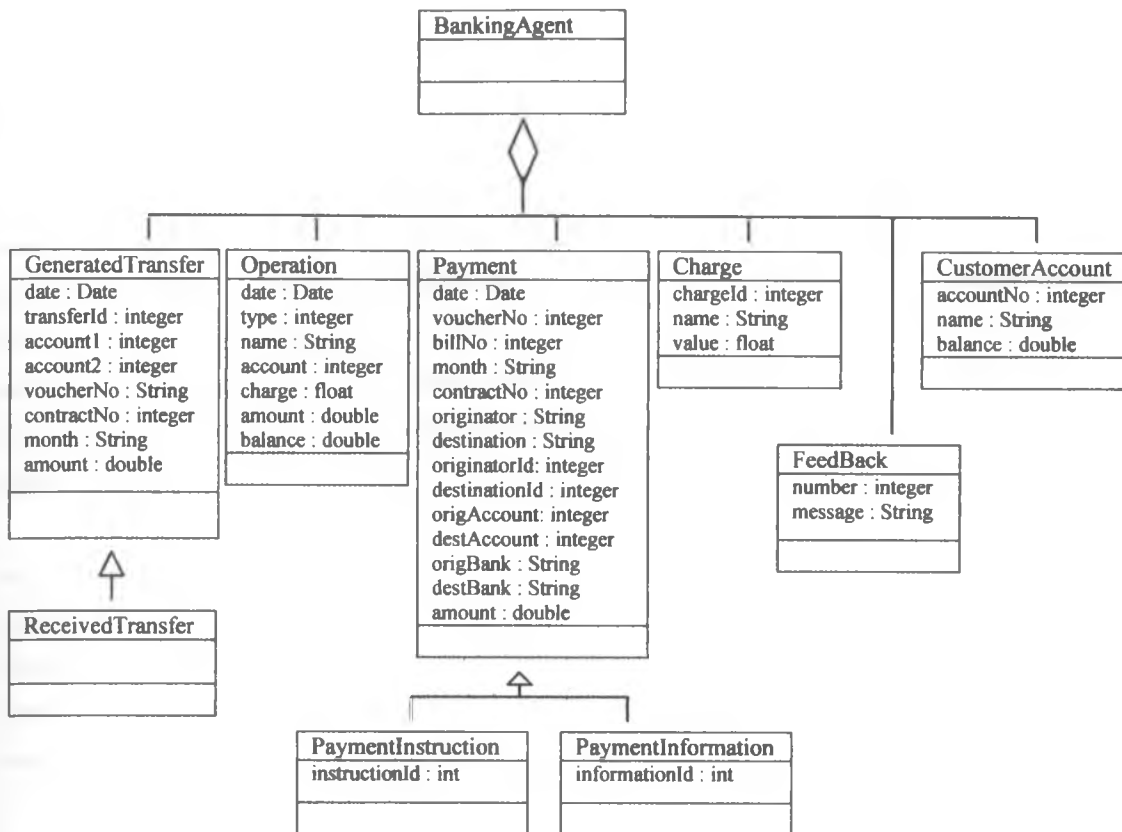


Figure 4. 13: Domains under control of payment agent

**Domains under control of banking agent**



**Figure 4. 14: Domains under control of banking agent**

## **CHAPTER 5: DESIGN**

### **5.0 Introduction**

Inter-operator billing and payment process is bidirectional between any two operators. The processes which have been used in the current inter-operator billing and payment systems are semi automatic- at some instant, there must be human intervention. The AMBIBPSY system has implemented inter-operator billing and payment process automatically without human intervention. The application has been restricted to inter-operator billing and payment for voice services. The system collects CDR, does billing, perform payment and carry out banking automatically. Due to the complexity of the system, we have developed it using multiagent as shown in the schematic diagram (figure 5.1) below. We have used four agents: CDR collection agent (CA), billing agent (BA), payment agent (PA) and banking agent (BKA).

In this chapter, we are going to discuss various design aspects that have translated the system requirements obtained from analysis into functionalities to be coded using a computer programming language. The designs that are going to be considered include: interaction design, database design, ontology design, agents design, object generation and acquaintance design, and interfaces design. The multiagent design methodology that has been used is known as MESSAGE.

#### **CDR collection agent (CA)**

This agent receives and stores raw data from the switch. The raw CDR is formatted to suite the billing format. CDR reconciliation with the external agent takes place at this level. The reconciled and formatted CDR then sent to the billing agent.

#### **Billing agent (BA)**

Reconciled and formatted CDR from all collection agents are received by the billing agent. The analysis of the reconciled CDR is done and sum of call duration is converted into paid minutes. Reconciliation of the billing data is done with the external billing agent. Once the billing data has been reconciled, a bill is prepared according to the contract rules. The prepared bill is transmitted to the external billing agent. A bill sent from the external billing agent is received by the local billing agent which then verifies the bill and forward it to the payment agent.

#### **Payment agent (PA)**

After the billing agent has received the bill from the external billing agent, the bill is transmitted to the payment agent. The payment agent generates and sends the payment voucher to the external payment agent. The payment vouchers generated from the external payment agents are received by the payment agent. To implement payment, the payment agent generates and transmits payment instruction to the local banking agent.

### Banking agent (BKA)

The banking agent receives payment instruction from the payment agent. The BKA carries out money transfer using electronic fund transfer (EFT) to the external banking agent. The fund transferred from the distant banking agent is received by the local BKA. The update of the funds received from the external banking agent is provided to the payment agent.

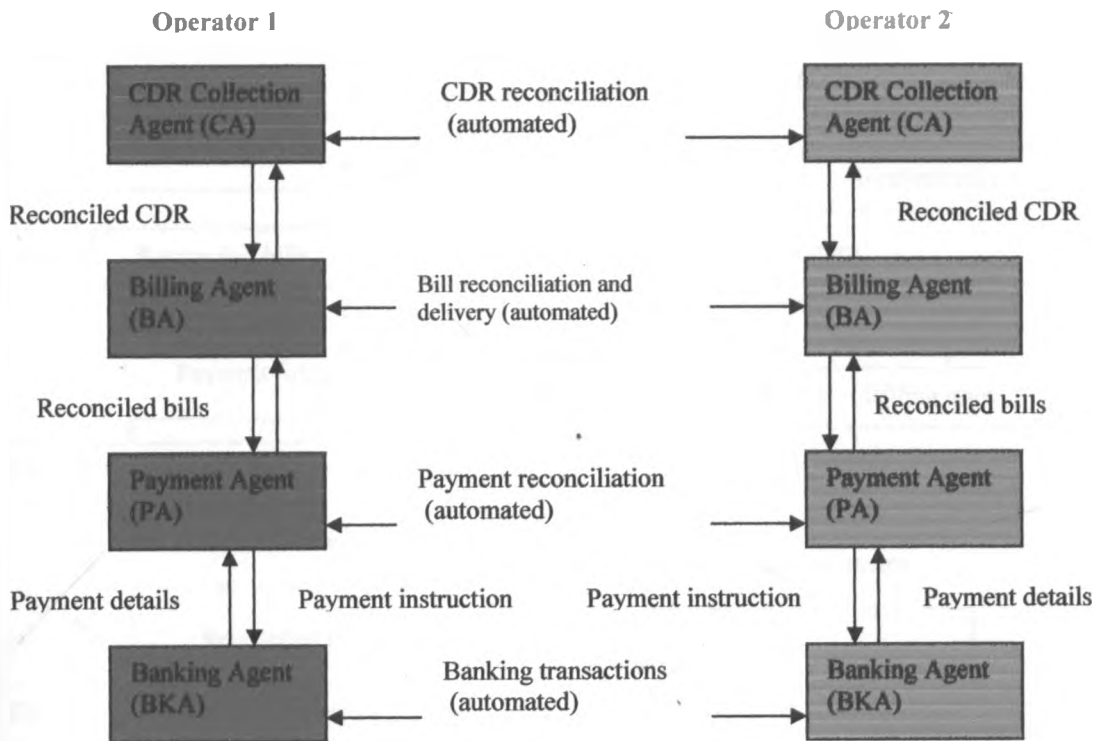


Figure 5. 1: Schematic Diagram for the system



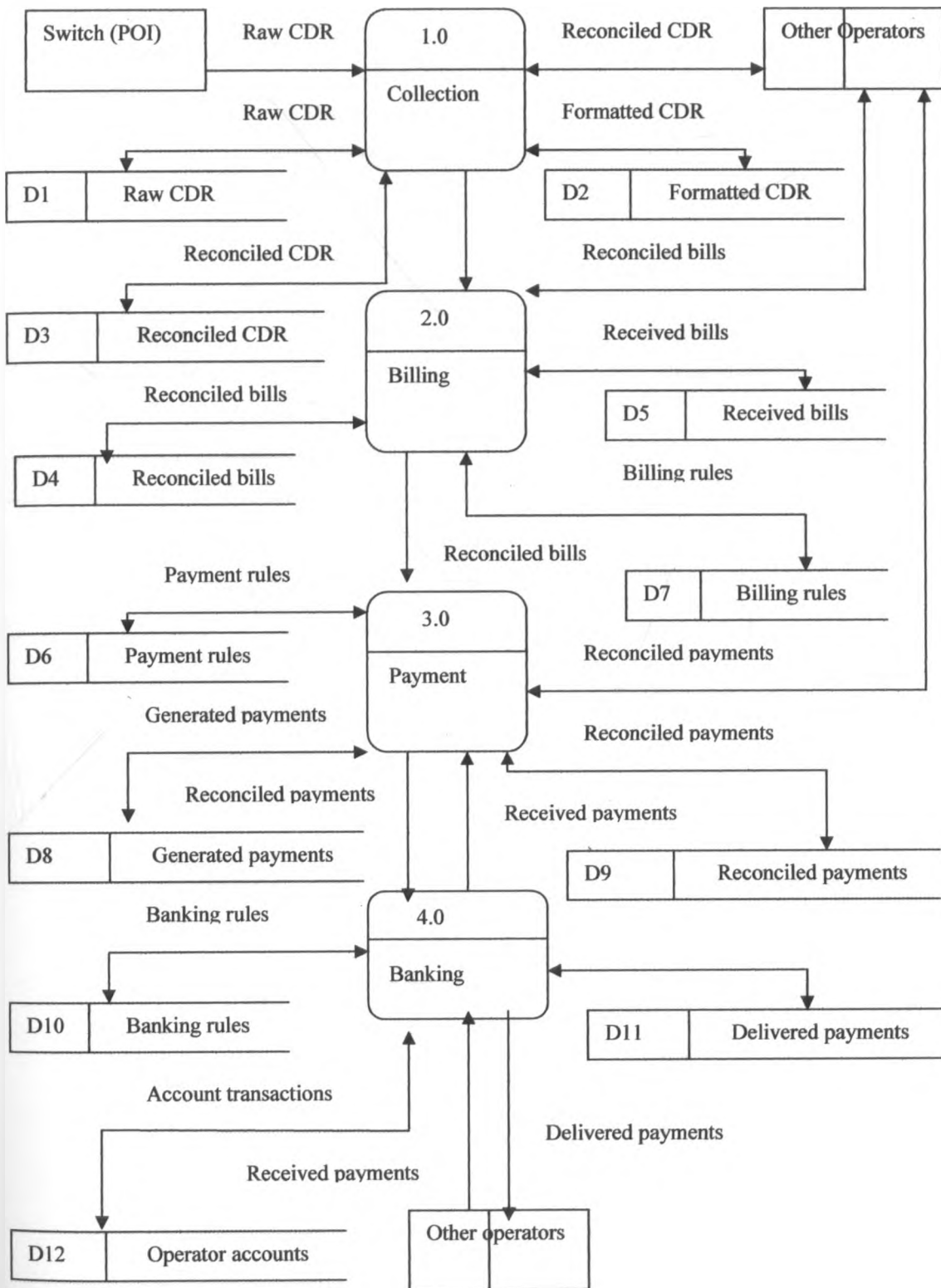


Figure 5. 2: DFD Diagram for the Proposed system

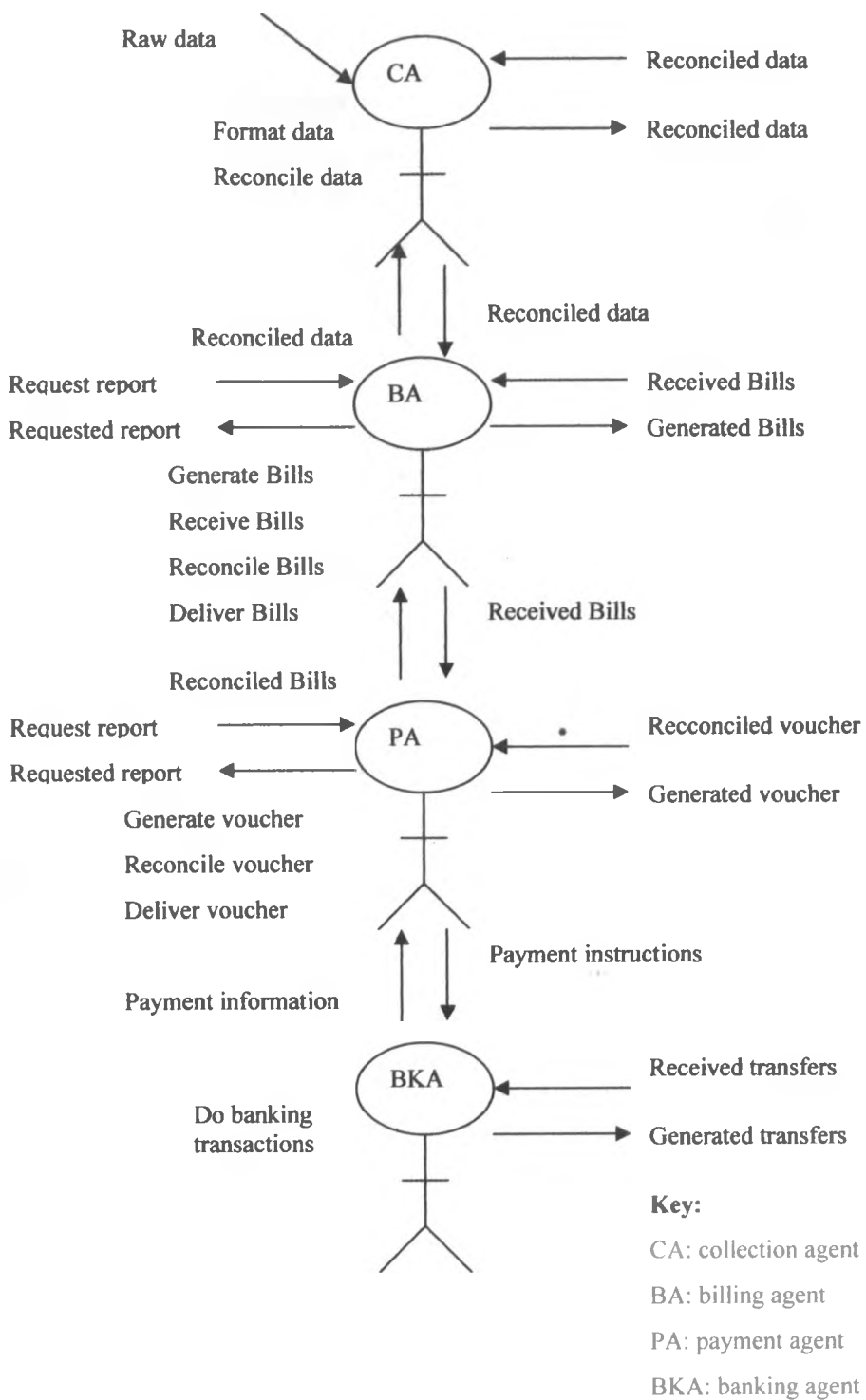


Figure 5. 3: Conceptual model of the system using agents

## 5.1 Interaction design

### 5.1.1 CDRReconciliation interaction.

This interaction involves four entities; two collector roles and two collection databases. The collector roles negotiate to reconcile the CDR and collection databases are used to provide and store the CDR being reconciled and the CDR which has been reconciled. The messages being exchanged are shown in the sequence diagram version of the interaction diagram.

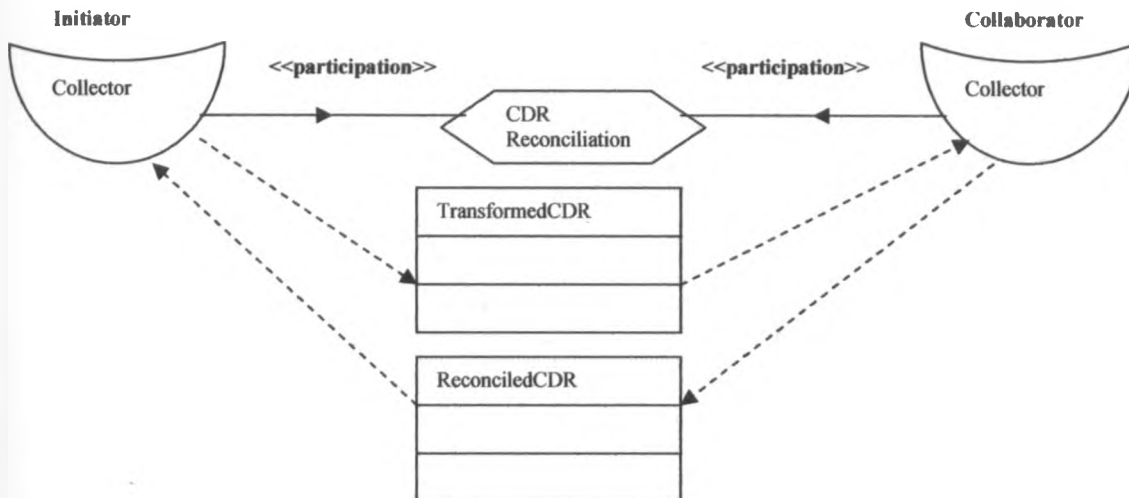


Figure 5. 4: CDRReconciliation interaction diagram

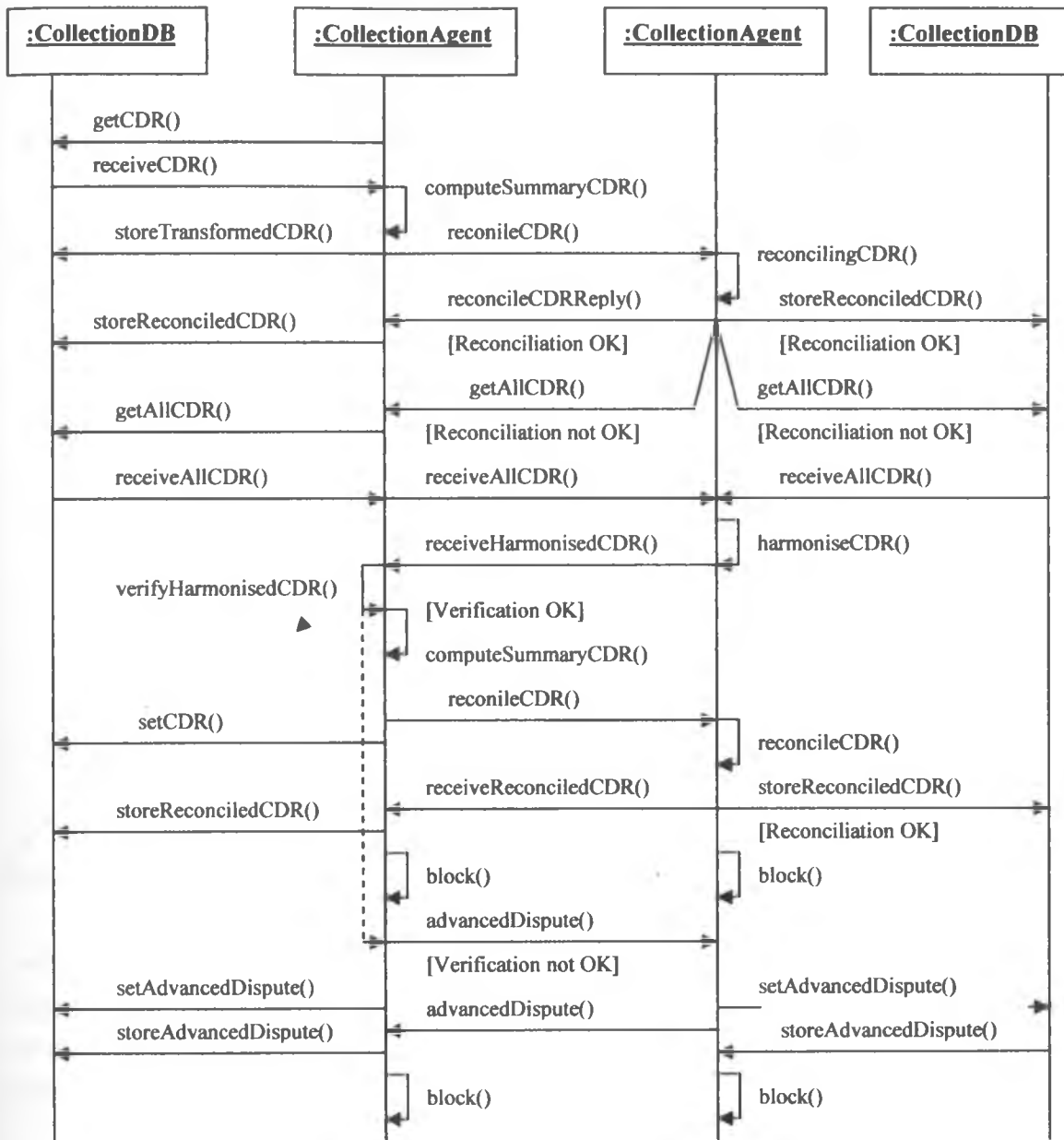


Figure 5. 5: CDRReconciliation interaction sequence diagram

### 5.1.2 ReconciledCDRDelivery interaction

This interaction takes place between the collector role and the biller role. The interaction involves delivery of reconciled CDR from the collection database to the billing database. The collection agent/role retrieves reconciled CDR from the collectionDB. The collection agent then delivers the reconciled CDR to the billing role/agent which receives the reconciled CDR then stores the reconciled CDR in the billingDB

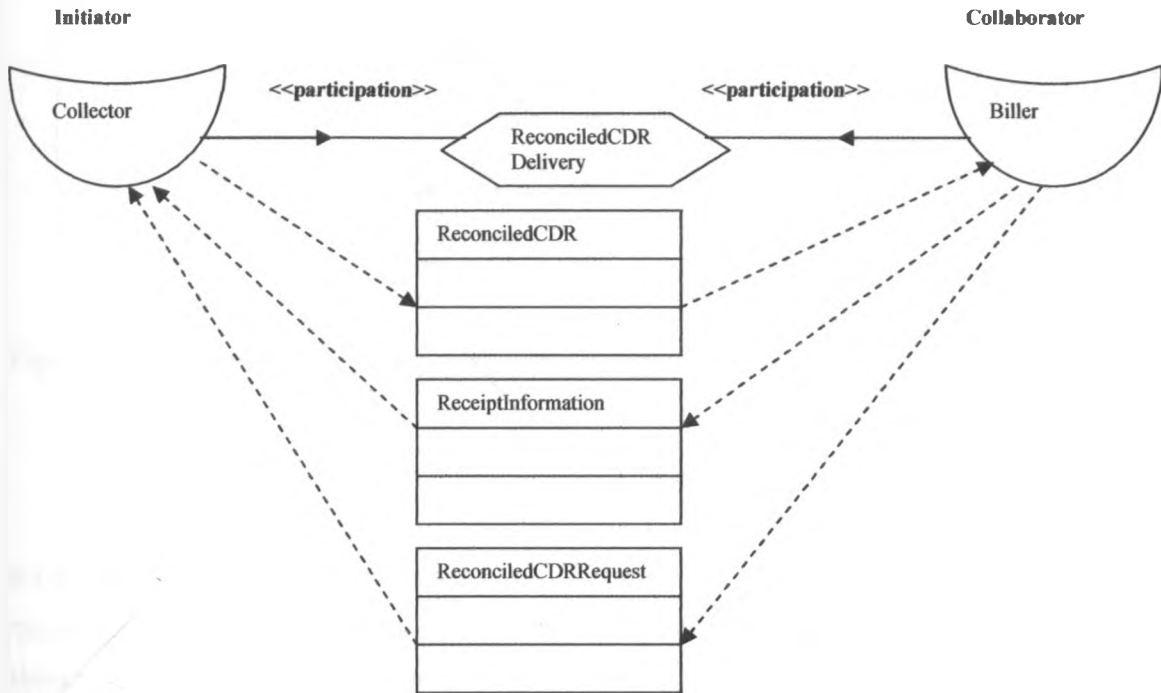


Figure 5. 6: ReconciledCDRDelivery interaction diagram

Under normal operation, the collection agent should deliver the reconciled CDR immediately after successful CDR reconciliation. The response to the billing agent is a set of reconciled CDR. Upon successful receipt of the reconciled CDR, the billing agent sends an acknowledgement receipt information to the collection agent.

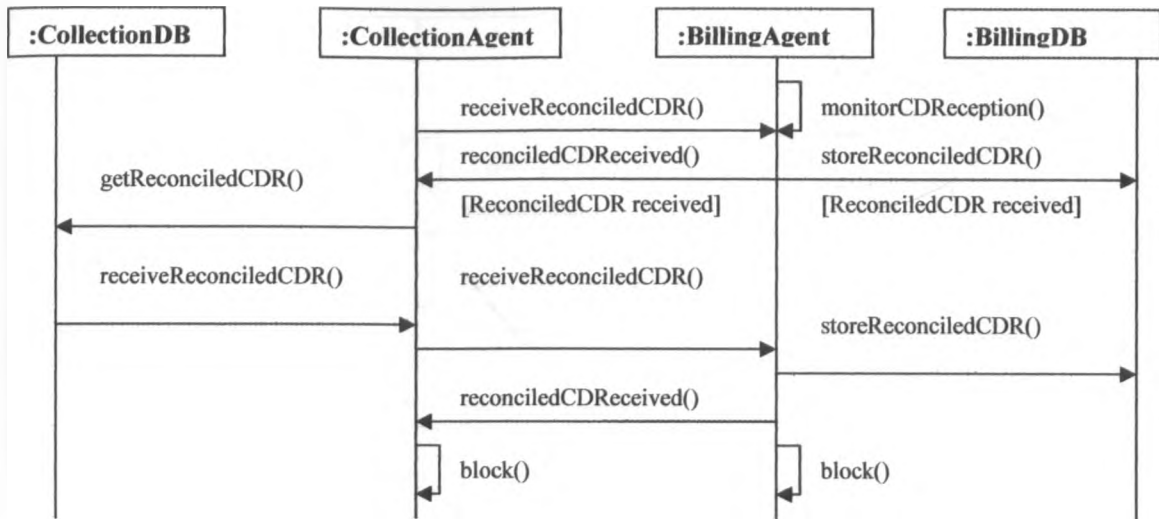


Figure 5. 7: ReconciledCDRDelivery interaction sequence diagram

### 5.1.3 BillReconciliation interaction

The bill reconciliation interaction takes place mainly between the billing agents of any two operators. This interaction facilitates the reconciliation of the generated bills to achieve a finally accepted reconciled bill. The reconciled bill is then sent to the payment agent to generate payment invoice .The interaction can be initiated by any of the two billing agents. In addition to the billing agents, the interaction incorporates billing databases of the two operators, which are used to provide and store data relevant to the interaction.

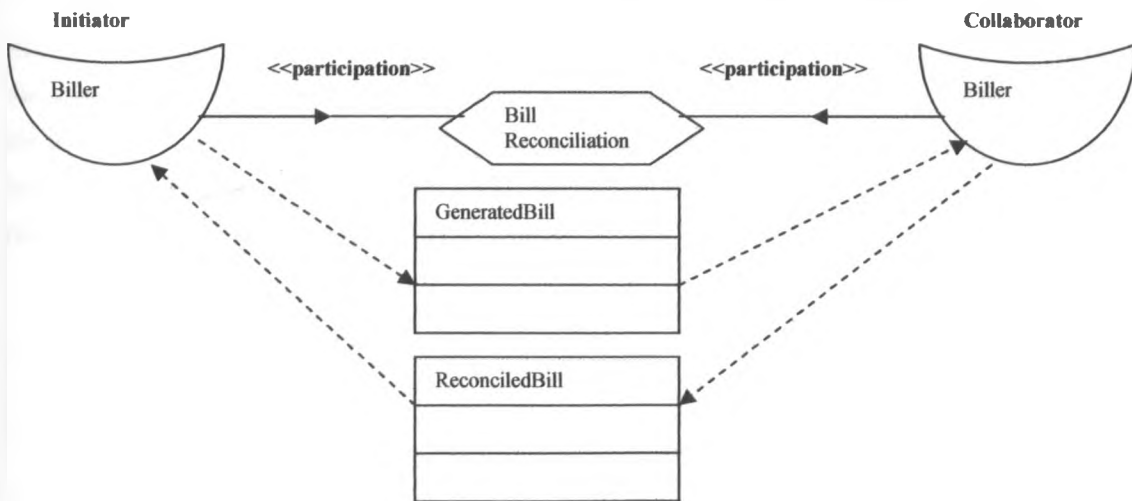


Figure 5. 8: BillReconciliation interaction diagram

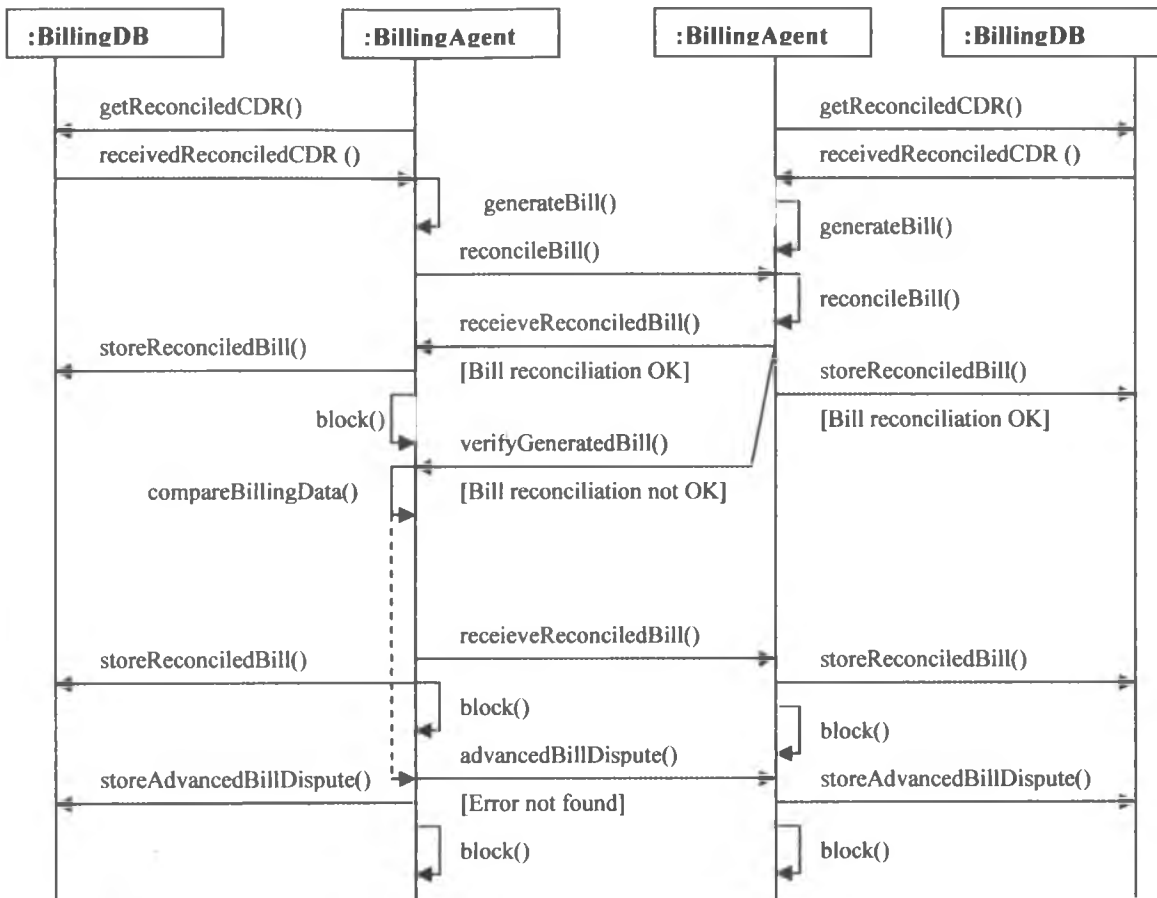


Figure 5. 9: BillReconciliation interaction sequence diagram

### 5.1.4 ReconciledBillDelivery interaction

In order to generate payment voucher or payment instruction, the payment agent requires reconciled bill. Reconciled bill delivery interaction is used to retrieve reconciled bill from the billing database and deliver the same to the payment database. The interaction involves billing agent and payment agent together with the billing database and payment database of the same operator. The billing agent retrieves the reconciled bill from the billing database. The billing agent then delivers the reconciled bill to the payment agent which receives and stores the reconciled bill in the payment database.

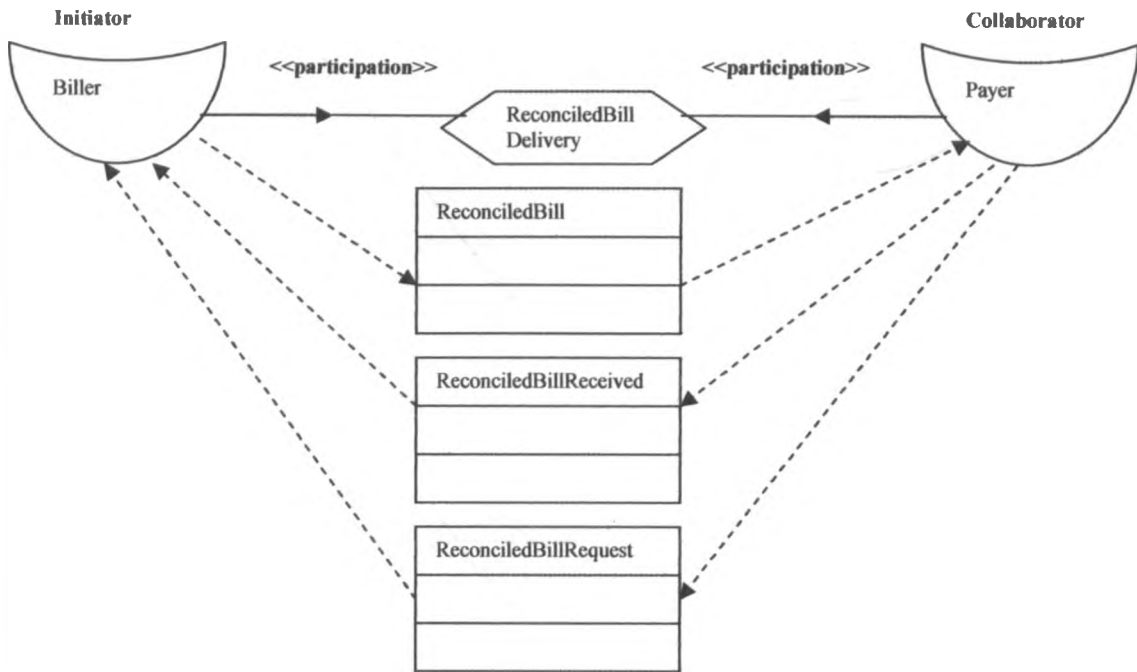


Figure 5. 10: ReconciledBillDelivery interaction diagram

The interaction is normally initiated by the billing role/agent and responded by the payment agent. It is ment to deliver the reconciled bill from the billing agent to the payment agent.

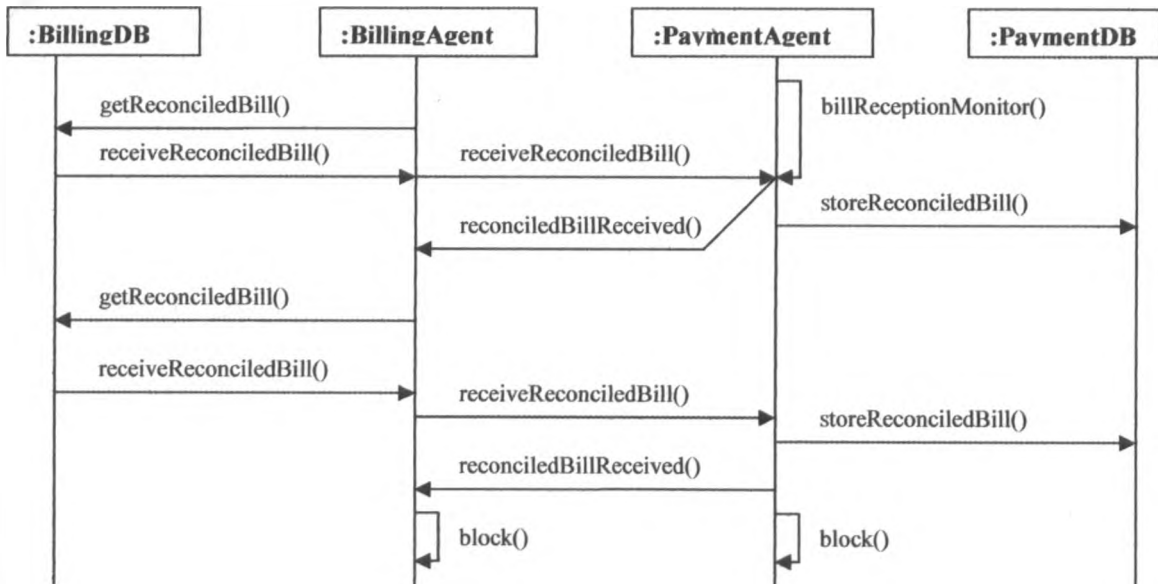


Figure 5. 11: ReconciledBillDelivery interaction sequence diagram



### 5.1.5 VoucherReconciliation interaction

Once the payment agent has received the reconciled bill, the payment agent generates payment voucher against the received reconciled bill. The generated voucher is reconciled for it to be accepted for payment instruction generation or payment information generation. Voucher Reconciliation Interaction is used to reconcile the payment voucher. The interaction involves two payment agents, each from a different operator. In addition, two payment databases, each from different operators are also included. The interaction is initiated by the payment agent that has generated the voucher and the response is provided by the distant paymentAgent.

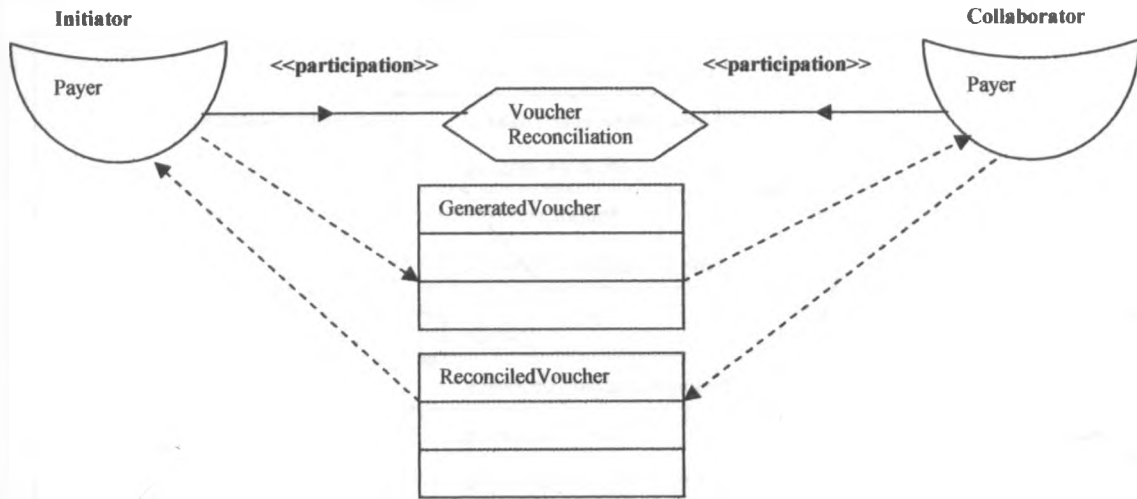


Figure 5. 12: VoucherReconciliation interaction diagram

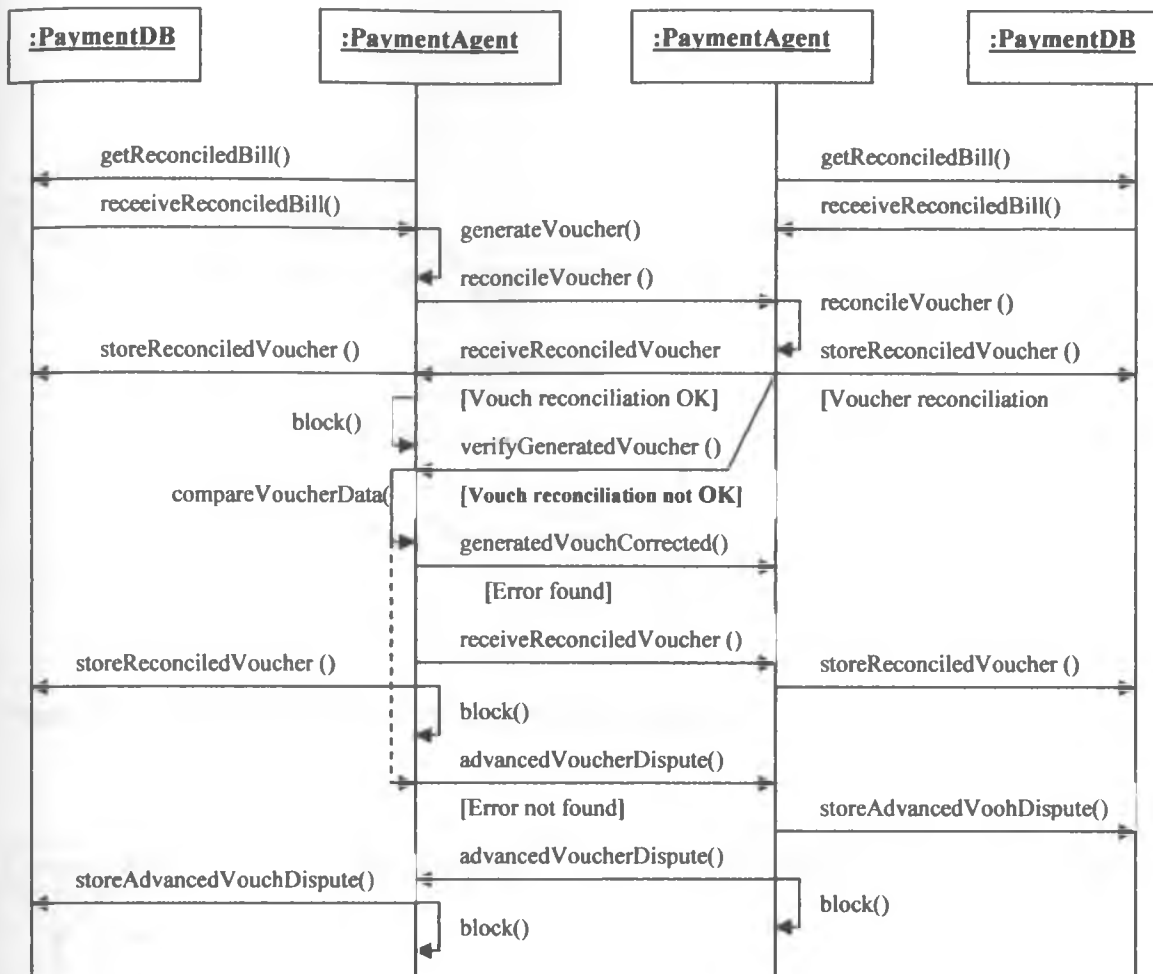


Figure 5. 13: VoucherReconciliation interaction sequence diagram

### 5.1.6 Payment instruction delivery interaction

After the payment voucher has been reconciled, the payment agent generates and deliver payment instruction to the banking agent. The banking agent receives and stores the payment instruction in the banking database. Payment instruction delivery involves payment agent and banking agent of the same operator. The interaction is initiated by the payment agent and the response is provided by the banking agent. The payment agent retrieves the reconciled voucher which is used to generate payment instruction. The generated payment instruction is stored in the payment database and is also delivered to the banking agent. The banking agent receives the payment instruction and stores it in the baking database, then acknowledges the receipt of the payment instruction to the payment agent.

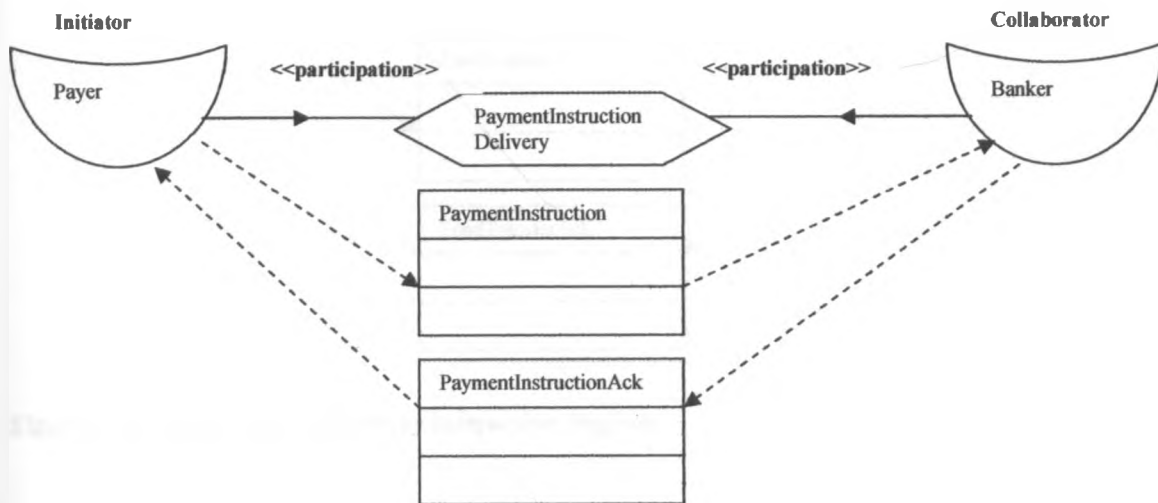


Figure 5. 14: PaymentInstructionDelivery interaction diagram

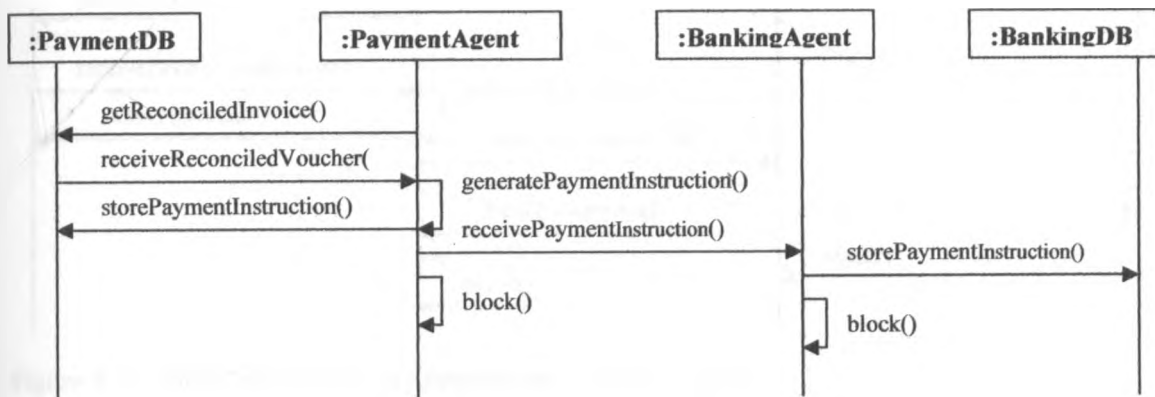


Figure 5. 15: PaymentInstructionDelivery interaction sequence diagram

### 5.1.7 FundTransfer delivery interaction

This interaction involves two banking roles from different operators. After receiving paymentInstruction from the paymentAgent, the banking agent generates and delivers fundTransfer to the distant bankingAgent. Both the banking agents store the generated fundTransfer in their banking databases. The interaction is initiated by the bankingAgent that has received the payment instruction. In response, the distant bankingAgent sends back a fundTransfer acknowledgement information to the initiating bankingAgent.

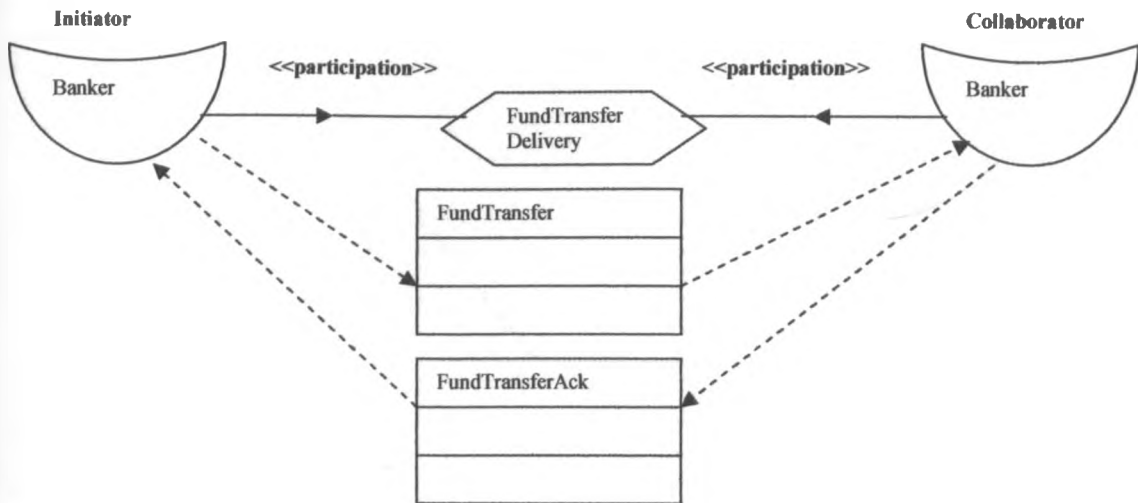


Figure 5. 16: FundTransferDelivery interaction diagram

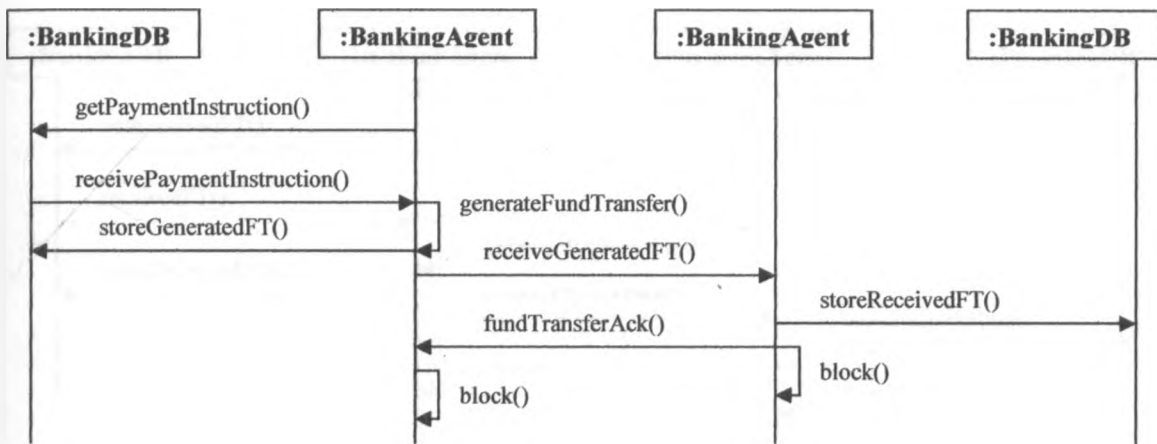


Figure 5. 17: FundTransferDelivery interaction sequence diagram

### 5.1.8 PaymentInformationDelivery interaction

Once the bankingAgent has received the fundTransfer information, it generates and delivers paymentInformation to the paymentAgent. This interaction is initiated by the bankingAgent and collaborated by the paymentAgent. The bankingAgent retrieves the fundTransfer information from the banking database, and generates paymentInformation. The generated paymentInformation is stored in the banking database and also sent to the local paymentAgent. The paymentAgent receives and stores the paymentInformation in the payment database. The paymentInformation represents the payment which has been received by the bankingAgent on behalf of the operator. The paymentAgent acknowledges the receipt of the paymentInformation by sending acknowledgement information to the bankingAgent.

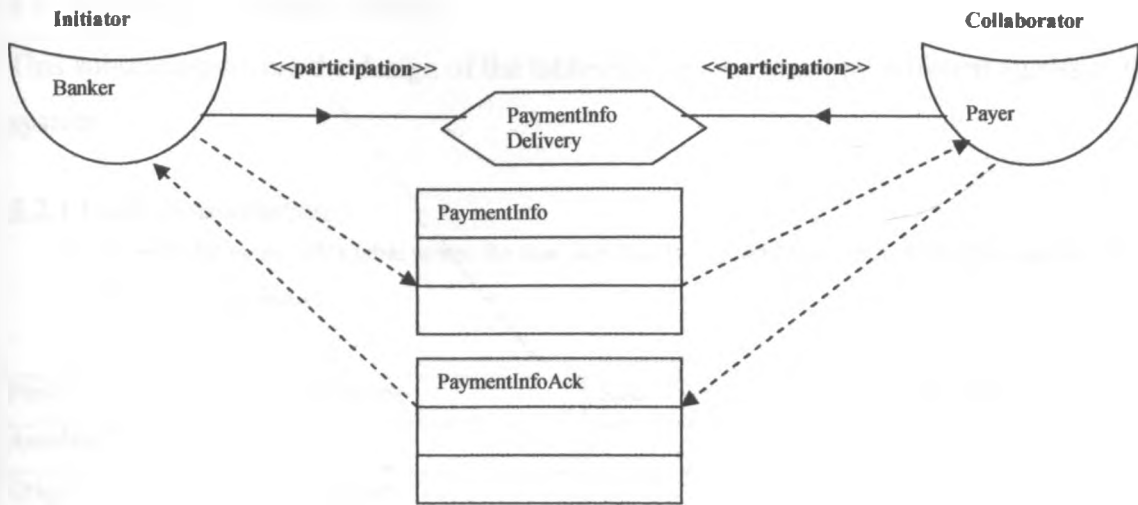


Figure 5. 18: PaymentInfoDelivery interaction diagram

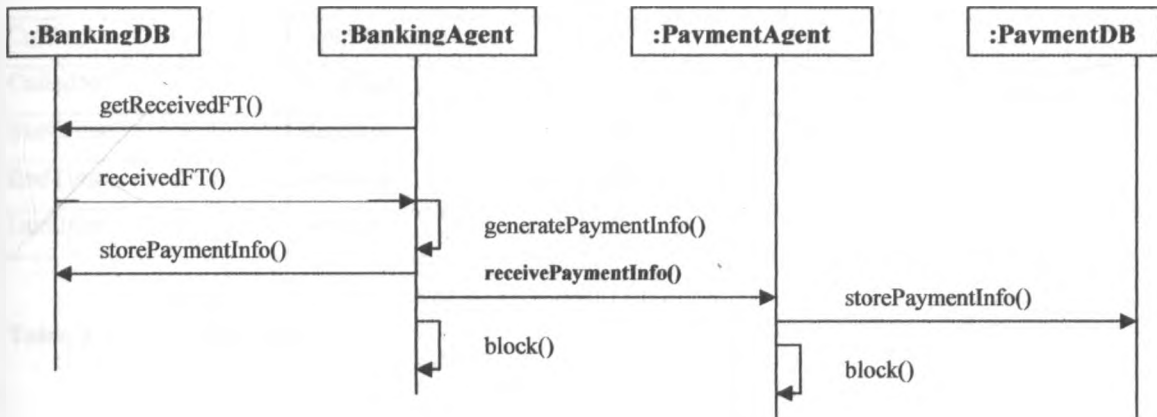


Figure 5. 19: PaymentInfoDelivery interaction sequence

## 5.2 The design of database tables

This subsection covers the design of the tables that are managed by different agents in the system.

### 5.2.1 Collection database:

- a) **Raw CDR table:** This table stores the raw data that is captured from POI. The table consists of the following fields

| Field         | Data type | Size | Constraint |
|---------------|-----------|------|------------|
| AutoNo        | bigint    |      |            |
| Origin        | nvarchar  | 20   |            |
| Destination   | nvarchar  | 20   |            |
| OriginId      | int       | 20   |            |
| DestinationId | int       | 20   |            |
| RouteId       | nvarchar  | 20   |            |
| RouteType     | nvarchar  | 5    |            |
| CallingNo     | nvarchar  | 20   |            |
| CalledNo      | nvarchar  | 20   |            |
| StartTime     | datetime  | 30   |            |
| EndTime       | datetime  | 30   |            |
| Duration      | integer   | 20   |            |

**Table 5. 1:** Raw CDR table

**b) Operators table**

This table contains the operators that are connected in the telecommunication network. It forms part of the knowledge for the agents to verify the valid operators to be served. Any time a new operator joins the service, its name and identification must be added in the operators table. The AMBIBPSY assumes that there four operators existing in the telecommunication network. The operators table consists of the following fields

| Field      | Data type | Size | Constraint  |
|------------|-----------|------|-------------|
| OperatorId | int       | 20   | Primary key |
| Name       | nvarchar  | 20   |             |

**Table 5. 2:** Operators table

c) **Routing table:** This table has the following fields

- RouteId: denotes the POI and the destined operator
- RouteType: is either in or out
- Origin: name of originating operator or originating operatorId
- DestOperator: name of destination operator or destination operatorId

| Field name  | Data type | Size | Constraint  |
|-------------|-----------|------|-------------|
| RouteId     | nvarchar  | 20   | Primary key |
| RouteType   | nvarchar  | 5    |             |
| Origin      | nvarchar  | 20   |             |
| Destination | nvarchar  | 20   |             |

**Table 5. 3:** Routing table

Naming of RouteId:

The name of the RouteId should signify the originating and destination operators, the point of interconnection (POI) number and the type of route.

Example 1: A RouteId denoted by Airtel\_010\_12 is decoded as follows

Airtel: Airtel operator route naming scheme

01: represents Airtel POI\_01

0: represents outgoing route from Airtel

12: represents destination operatorId

Example 2: A RouteId denoted by Airtel\_011\_12 is decoded as follows

Airtel: Airtel operator route naming scheme

01: represents Airtel POI\_01

1: represents incoming route into Airtel

12: represents destination operatorId

The RouteId will help in selecting the calls originating or terminating in a given POI of an operator for the purpose of CDR reconciliation. The collectionAgent should be able to retrieve the CDR from the raw data table based on the RouteId.

d) **Point of interconnection (POI) table:** As points of connections are added into the network, their details are stored in the POI table. This table has the following fields

| Field name | Data type | Size | Constraint  |
|------------|-----------|------|-------------|
| PoId       | nvarchar  | 20   | Primary key |
| Name       | nvarchar  | 20   |             |
| OperatorId | int       | 20   |             |

**Table 5. 4:** POI table

- e) **Route definition table:** This table contains the routes which have been created. It indicates the POI to which the route belongs and a description of the route showing the two POIs associated by the route. The fields in this table include

| Field name  | Data type | Size | Constraint            |
|-------------|-----------|------|-----------------------|
| RouteId     | nvarchar  | 20   | Primary key, not null |
| Poild       | nvarchar  | 20   | not null              |
| Description | nvarchar  | 30   | not null              |
| RouteType   | nvarchar  | 5    | not null              |

**Table 5. 5:** Route definition table

- f) **ReconciledCDR table:** This table contains the daily summary of both the outgoing calls from a POI and incoming calls into a POI. The data for this table come as a result of processing raw CDR by the collectionAgent. The processed data is sent to the distant collectionAgent for verification and certification. Once the summary of the CDR data has been certified, the reconciledCDR data is stored in the reconciledCDR table. The reconciledCDR table has the following fields

| Field name    | Data type | Size | Constraint |
|---------------|-----------|------|------------|
| AutoNo        | int       |      | not null   |
| Date          | datetime  |      | not null   |
| Originator    | nvarchar  | 20   | not null   |
| Destination   | nvarchar  | 20   | not null   |
| OriginatorId  | int       |      | not null   |
| DestinationId | int       |      | not null   |
| RouteId       | nvarchar  | 20   | not null   |
| RouteType     | nvarchar  | 5    | not null   |
| Units         | bigint    |      | not null   |

**Table 5. 6:** ReconciledCDR table



**g) CDRDisputes table**

During CDR reconciliation, different errors are detected and resolved. The errors detected during CDR reconciliation are registered as disputes. The disputes that are registered are stored in the CDR disputes table. This table contains the following fields

| Field name    | Data type | Size | Constraint            |
|---------------|-----------|------|-----------------------|
| Date          | datetime  |      | not null              |
| DisputeId     | integer   | 20   | Primary key, not null |
| Originator    | nvarchar  | 20   | not null              |
| Destination   | nvarchar  | 20   | not null              |
| OriginatorId  | int       |      | not null              |
| DestinationId | int       |      | not null              |
| DisputeType   | nvarchar  | 20   | not null              |
| Description   | nvarchar  | 30   | not null              |
| Units         | bigint    |      | not null              |
| Status        | nvarchar  | 5    | not null              |

**Table 5. 7:** CDRDisputes table

**h) Dispute description table**

This table contains the description of different types of dispute that can be registered by an agent. It has the following fields

| Field name  | Data type | Size | Constraint            |
|-------------|-----------|------|-----------------------|
| DisputeId   | integer   | 20   | Primary key, not null |
| Description | nvarchar  | 30   | not null              |

**Table 5. 8:** Dispute description table

- i) **TransformedCDR table:** This table contains the daily summary of both the outgoing calls from a POI and incoming calls into a POI. The data for this table come as a result of processing raw CDR by the collectionAgent. The processed data is sent to the distant collectionAgent for verification and certification. Once the summary of the CDR data has been certified, the the distant collectionAgent send back a reconciledCDR data which is then stored in the reconciledCDR table. The transformedCDR table has the following fields

| Field name    | Data type | Size | Constraint |
|---------------|-----------|------|------------|
| AutoNo        | int       |      | not null   |
| Date          | datetime  |      | not null   |
| Originator    | nvarchar  | 20   | not null   |
| Destination   | nvarchar  | 20   | not null   |
| OriginatorId  | int       |      | not null   |
| DestinationId | int       |      | not null   |
| RouteId       | nvarchar  | 20   | not null   |
| RouteType     | nvarchar  | 5    | not null   |
| Units         | bigint    |      | not null   |

**Table 5. 9:** TransformedCDR table

**j) Collection user table**

This table contains the configuration of the people who are supposed to use the Collection system.

| Field name | Data type | Size | Constraint  |
|------------|-----------|------|-------------|
| Date       | datetime  | 30   |             |
| System     | nvarchar  | 20   |             |
| Category   | nvarchar  | 20   |             |
| Name       | nvarchar  | 20   |             |
| Identity   | int       | 20   | Primary key |
| Password   | nvarchar  | 20   |             |

**Table 5. 10:** Collection user table

k) **Collection Login table:** This table captures how users login and out of the collection system. It has the following fields:

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| StartDate  | datetime  | 30   |            |
| EndDate    | datetime  | 20   |            |
| Duration   | int       | 20   |            |
| System     | nvarchar  | 20   |            |
| Category   | nvarchar  | 20   |            |
| Name       | nvarchar  | 20   |            |
| Identity   | int       | 20   |            |
| Password   | nvarchar  | 20   |            |

**Table 5. 11:** Collection Login table

### 5.2.2 Billing database

The billing database contains the files that are managed by the billingAgent. It consists of the following tables

a) **ReconciledCDR table**

This table stores the reconciledCDR sent from all the collection agents serving a given operator. The billing units for different operators are obtained from this table. The fields contained in this table are as follow

| Field name    | Data type | Size | Constraint |
|---------------|-----------|------|------------|
| AutoNo        | int       |      | not null   |
| Date          | datetime  |      | not null   |
| Originator    | nvarchar  | 20   | not null   |
| Destination   | nvarchar  | 20   | not null   |
| OriginatorId  | int       |      | not null   |
| DestinationId | int       |      | not null   |
| RouteId       | nvarchar  | 20   | not null   |
| RouteType     | nvarchar  | 5    | not null   |
| Units         | bigint    |      | not null   |

**Table 5. 12:** ReconciledCDR table

The billing agent will select the units for a particular operator from the reconciledCDR table. These are the units that are going to be used to calculate the paid minutes. The search will use the billing month, operatorId, and the route type. This will enable the paid minutes for both outgoing and incoming calls to be computed. The difference between the outgoing calls paid minutes and the incoming calls paid minutes is computed to determine the paid minutes to be used to generate the bill to be sent to the distant billing agent or expected from it.

**b) Generated bills table**

This table contains the generated bill sent to the distant billing agent. The generated bills table has the following fields:

| Field name    | Data type | Size | Constraint  |
|---------------|-----------|------|-------------|
| Date          | datetime  | 30   |             |
| BillNo        | integer   | 20   | Primary key |
| Month         | nvarchar  | 10   |             |
| ContractNo    | int       | 20   |             |
| Originator    | nvarchar  | 20   |             |
| Destination   | nvarchar  | 20   |             |
| OriginatorId  | int       | 20   |             |
| DestinationId | int       | 20   |             |
| PaidMinutes   | int       | 20   |             |
| RateId        | Int       | 10   |             |
| RateValue     | float     | 10   |             |
| Amount        | float     | 30   |             |

**Table 5. 13: Generated bills table**

- c) **Reconciled bills table:** The table stores the reconciled bill from the distant billing agent. It has the following fields.

| Field name    | Data type | Size | Constraint  |
|---------------|-----------|------|-------------|
| Date          | datetime  | 30   |             |
| BillNo        | integer   | 20   | Primary key |
| Month         | nvarchar  | 10   |             |
| ContractNo    | int       | 20   |             |
| Originator    | nvarchar  | 20   |             |
| Destination   | nvarchar  | 20   |             |
| OriginatorId  | int       | 20   |             |
| DestinationId | int       | 20   |             |
| PaidMinutes   | int       | 20   |             |
| RateId        | Int       | 10   |             |
| RateValue     | float     | 10   |             |
| Amount        | float     | 30   |             |

**Table 5. 14: Reconciled bills table**

- d) **Received bills table**

This table stores the bills received from the other billing agents. The information in this table is used to generate payment voucher. The fields contained in this table are as follow

| Field name    | Data type | Size | Constraint  |
|---------------|-----------|------|-------------|
| Date          | datetime  | 30   |             |
| BillNo        | integer   | 20   | Primary key |
| Month         | nvarchar  | 10   |             |
| ContractNo    | int       | 20   |             |
| Originator    | nvarchar  | 20   |             |
| Destination   | nvarchar  | 20   |             |
| OriginatorId  | int       | 20   |             |
| DestinationId | int       | 20   |             |
| PaidMinutes   | bigint    |      |             |
| RateId        | int       | 10   |             |
| RateValue     | float     | 10   |             |
| Amount        | float     | 30   |             |

**Table 5. 15: Received bills table**

e) **BillDispute table:** Stores disputes generated as a result of bill reconciliation. It has the following fields.

| Field name    | Data type | Size | Constraint            |
|---------------|-----------|------|-----------------------|
| Date          | datetime  |      | not null              |
| DisputeId     | integer   | 20   | Primary key, not null |
| Originator    | nvarchar  | 20   | not null              |
| Destination   | nvarchar  | 20   | not null              |
| OriginatorId  | int       |      | not null              |
| DestinationId | int       |      | not null              |
| DisputeType   | nvarchar  | 20   | not null              |
| Description   | nvarchar  | 30   | not null              |
| Amount        | float     |      | not null              |
| Status        | nvarchar  | 5    | not null              |

**Table 5. 16:** BillDisputes table

f) **Contract table**

This is the table that stores the operators involved in the telecommunication service contract. It specifies the contract number between any two operators. The table has the following fields

| Field name  | Data type | Size | Constraint |
|-------------|-----------|------|------------|
| Date        | datetime  | 30   |            |
| ContractNo  | integer   | 20   |            |
| Operator1   | nvarchar  | 20   |            |
| Operator2   | nvarchar  | 20   |            |
| Operator1Id | int       | 20   |            |
| Operator2Id | int       | 20   |            |
| Validity    | nvarchar  | 5    |            |

**Table 5. 17:** Contract table

The validity field signifies the successful configuration of the contract by both the operators. The value of the validity field defaults to NO if the other operator has not configured the same contract number. If both the operators have configured the contact number correctly, the value of the validity field should change to YES. It therefore means that during the configuration of the contract number, the two billing agents of the two operators must communicate to ensure that the contract number is agreed upon by both of them. The configuration of the contract table is done by the system administrator

**g) Billing rate table**

This table contains the billing rates for the possible services provided by the operators. During billing, the billing agent will get the relevant billing rate from this table. The fields contained in this table are as follow

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| RateId     | integer   | 10   |            |
| Name       | nvarchar  | 20   |            |
| Value      | float     | 10   |            |

**Table 5. 18: Billing rate table**

**h) Billing user table**

This table contains the configuration of the people who are supposed to use the billing system.

| Field name | Data type | Size | Constraint  |
|------------|-----------|------|-------------|
| Date       | datetime  | 30   |             |
| System     | nvarchar  | 20   |             |
| Category   | nvarchar  | 20   |             |
| Name       | nvarchar  | 20   |             |
| Identity   | integer   | 20   | Primary key |
| Password   | nvarchar  | 20   |             |

**Table 5. 19: Billing user table**

**i) Billing Login table: The table contains the users who login and logout of the Billing system.**

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| StartDate  | datetime  | 30   |            |
| EndDate    | datetime  | 20   |            |
| Duration   | integer   | 20   |            |
| System     | nvarchar  | 20   |            |
| Category   | nvarchar  | 20   |            |
| Name       | nvarchar  | 20   |            |
| Identity   | integer   | 20   |            |
| Password   | nvarchar  | 20   |            |

**Table 5. 20: Billing Login table**

### 5.2.3 Payment database

This database contains the following tables

- Generated bills
- Received bills
- Operator accounts
- Generated vouchers
- Received vouchers
- Payment instructions
- Payment information

#### a) Generated bills account table

This table stores the bills which have been generated and sent to the distant billing agent for services rendered. It has the structure as the generated bill stored in the generated bills table in the billing database. This offers data redundancy for reliability and security purposes. The data in this table is used by the payment agent to verify the content of the received voucher from the distant billing agent.

| Field name   | Data type | Size | Constraint  |
|--------------|-----------|------|-------------|
| Date         | datetime  | 30   |             |
| BillNo       | integer   | 20   | Primary key |
| Month        | nvarchar  | 10   |             |
| ContractNo   | integer   | 20   |             |
| Originator   | nvarchar  | 20   |             |
| Receiver     | nvarchar  | 20   |             |
| OriginatorId | integer   | 20   |             |
| ReceiverId   | integer   | 20   |             |
| PaidMinutes  | bigint    | 20   |             |
| RateId       | integer   | 10   |             |
| RateValue    | float     | 10   |             |
| Amount       | float     | 30   |             |

**Table 5. 21:** Generated bills account table

#### b) Received bills table

The bills received from the distant billing agent are first stored in the billing database. A copy of the received bill is sent to the payment agent and stored in the received bills table. The payment agent uses the data from this table to generate the payment voucher for the services received from the other operator. The structure of the received bill table is similar to the received bills table in the billing data base. It has the following fields



| Field name   | Data type | Size | Constraint |
|--------------|-----------|------|------------|
| Date         | datetime  | 30   |            |
| BillNo       | integer   | 20   |            |
| Month        | nvarchar  | 10   |            |
| ContractNo   | int       | 20   |            |
| Originator   | nvarchar  | 20   |            |
| Receiver     | nvarchar  | 20   |            |
| OriginatorId | int       | 20   |            |
| ReceiverId   | integer   | 20   |            |
| PaidMinutes  | bigint    | 20   |            |
| Rate         | float     | 10   |            |
| Amount       | float     | 30   |            |

**Table 5. 22: Received bills table**

**c) Generated vouchers**

Once the payment agent has stored the received bills from the billing agent, it generates a voucher against the received bill. The generated voucher is stored in the generated voucher table. This table has the following fields

| Field name    | Data type | Size | Constraint  |
|---------------|-----------|------|-------------|
| Date          | datetime  | 30   |             |
| VoucherNo     | integer   | 20   | Primary key |
| BillNo        | nvarchar  | 20   |             |
| Month         | nvarchar  | 10   |             |
| ContractNo    | int       | 20   |             |
| Originator    | nvarchar  | 20   |             |
| Destination   | nvarchar  | 20   |             |
| OriginatorId  | int       | 20   |             |
| DestinationId | int       | 20   |             |
| OrigAccount   | int       |      |             |
| DestAccount   | int       | 20   |             |
| OrigBank      | nvarchar  | 20   |             |
| DestBank      | nvarchar  | 20   |             |
| Amount        | float     | 30   |             |

**Table 5. 23: Generated vouchers**

**c) Operator accounts table**

This is the table that stores the operator's account particulars. The data in this table is used to verify the operator's account number incase of any query. The table has the following fields

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| OperatorId | integer   | 20   |            |
| Name       | nvarchar  | 20   |            |
| ContractNo | integer   | 20   |            |
| Account    | integer   | 30   |            |
| Bank       | nvarchar  | 20   |            |

**Table 5. 24: Operator accounts table**

**d) Received vouchers table**

This table stores the vouchers that have been received from the distant payment agent for the services rendered. The data in this table is used to verify the payment information received from the banking agent.

The structure of this table is similar to the generated voucher table.

| Field name    | Data type | Size | Constraint |
|---------------|-----------|------|------------|
| Date          | datetime  | 30   |            |
| VoucherNo     | integer   | 20   |            |
| BillNo        | nvarchar  | 20   |            |
| Month         | nvarchar  | 10   |            |
| ContractNo    | int       | 20   |            |
| Originator    | nvarchar  | 20   |            |
| Destination   | nvarchar  | 20   |            |
| OriginatorId  | int       | 20   |            |
| DestinationId | int       | 20   |            |
| OrigAccount   | int       |      |            |
| DestAccount   | int       | 20   |            |
| OrigBank      | nvarchar  | 20   |            |
| DestBank      | nvarchar  | 20   |            |
| Amount        | float     | 30   |            |

**Table 5. 25: Received vouchers table**

**e) Payment instruction table**

Once the payment agent has generated, reconciled and stored the payment voucher, it generates a payment instruction. The generated payment instruction is stored in the Payment instruction table. The generated payment instruction is delivered by the payment agent to the banking agent. The receipt of the payment instruction enables the banking agent to generate fund transfer against the generated invoice. The payment instruction table has the following fields

| Field name    | Data type | Size | Constraint  |
|---------------|-----------|------|-------------|
| Date          | datetime  | 30   |             |
| InstructionId | integer   | 20   | Primary key |
| VoucherNo     | nvarchar  | 20   |             |
| BillNo        | nvarchar  | 20   |             |
| Month         | nvarchar  | 10   |             |
| ContractNo    | integer   | 20   |             |
| Originator    | nvarchar  | 20   |             |
| Destination   | nvarchar  | 20   |             |
| OriginatorId  | integer   | 20   |             |
| DestinationId | integer   | 20   |             |
| OrigAccount   | integer t |      |             |
| DestAccount   | integer   | 20   |             |
| OrigBank      | nvarchar  | 20   |             |
| DestBank      | nvarchar  | 20   |             |
| Amount        | float     | 30   |             |

**Table 5. 26: Payment instruction table**

#### j) Payment information table

If the banking agent receives transferred fund on behalf of an operator, it generates and send payment information to the payment agent. The payment agent receives and stores the payment information in the payment information table. The payment information table has the following fields

| Field name    | Data type | Size | Constraint  |
|---------------|-----------|------|-------------|
| Date          | datetime  | 30   |             |
| InformationId | integer   | 20   | Primary key |
| VoucherNo     | integer   | 20   |             |
| BillNo        | nvarchar  | 20   |             |
| Month         | nvarchar  | 10   |             |
| ContractNo    | integer   | 20   |             |
| Originator    | nvarchar  | 20   |             |
| Destination   | nvarchar  | 20   |             |
| OriginatorId  | integer   | 20   |             |
| DestinationId | integer   | 20   |             |
| OrigAccount   | integer   | 20   |             |
| DestAccount   | integer   | 20   |             |
| OrigBank      | nvarchar  | 20   |             |
| DestBank      | nvarchar  | 20   |             |
| Amount        | float     | 30   |             |

Table 5. 27: Payment information table

#### j) Payment user table

This table contains the configuration of the people who are supposed to use the payment system.

| Field name | Data type | Size | Constraint  |
|------------|-----------|------|-------------|
| Date       | datetime  | 30   |             |
| System     | nvarchar  | 20   |             |
| Category   | nvarchar  | 20   |             |
| Name       | nvarchar  | 20   |             |
| Identity   | integer   | 20   | Primary key |
| Password   | nvarchar  | 20   |             |

Table 5. 28: Payment user table

**k) Payment Login table:** This table the users who login and logout of the payment system.

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| StartDate  | datetime  | 30   |            |
| EndDate    | datetime  | 20   |            |
| Duration   | integer   | 20   |            |
| System     | nvarchar  | 20   |            |
| Category   | nvarchar  | 20   |            |
| Name       | nvarchar  | 20   |            |
| Identity   | integer   | 20   |            |
| Password   | nvarchar  | 20   |            |

**Table 5. 29: Payment Login table**

### 5.2.4 Banking Database

The banking database contains the following tables

- Payment instructions table
- Payment information table
- customer accounts table
- Generated fund transfers table
- Received fund transfers table

#### a) Payment instruction table

This table contains the payment instructions received from the payment agent. The information in this table is used by the banking agent to generate fund transfer information. The structure of this table is the same as the payment instruction table in payment database. The table has the following fields

| Field name    | Data type | Size | Constraint  |
|---------------|-----------|------|-------------|
| Date          | datetime  | 30   |             |
| InstructionId | integer   | 20   | Primary key |
| VoucherNo     | nvarchar  | 20   |             |
| ContractNo    | integer   | 20   |             |
| Originator    | nvarchar  | 20   |             |
| Destination   | nvarchar  | 20   |             |
| OrigAccount   | integer   | 20   |             |
| DestAccount   | integer   | 20   |             |
| DestBank      | nvarchar  | 20   |             |
| Month         | nvarchar  | 10   |             |
| Amount        | float     | 30   |             |

**Table 5. 30: Payment instruction table**

**b) Payment information table**

The payment information generated by the banking agent is stored in this table. After storing the payment information, it is also sent to the payment agent. The payment information contains the details of the received funds from the distant banking agent on behalf of the operator. The structure of this table is similar to the payment information table in the payment database. The table has the following fields

| Field name    | Data type | Size | Constraint |
|---------------|-----------|------|------------|
| Date          | datetime  | 30   |            |
| InformationId | integer   | 20   |            |
| VoucherNo     | nvarchar  | 20   |            |
| ContractNo    | integer   | 20   |            |
| Originator    | nvarchar  | 20   |            |
| Destination   | nvarchar  | 20   |            |
| OrigAccount   | integer   | 20   |            |
| DestAccount   | nvarchar  | 20   |            |
| OrigBank      | nvarchar  | 20   |            |
| Month         | nvarchar  | 10   |            |
| Amount        | float     | 30   |            |

**Table 5. 31:** Payment information table

**c) Customer accounts table**

The details of the accounts held by different customers of a given bank are stored in customer account table. The table has the following fields

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| AccountNo  | int       | 20   |            |
| Name       | nvarchar  | 20   |            |
| Balance    | float     | 30   |            |

**Table 5. 32:** Customer accounts table

**d) Generated fund transfers table**

When the banking agent receives payment instruction from the payment agent, it generates fund transfer information. The generated fund transfer information is delivered to the distant banking agent that manages the payment information for the distant payment agent. The generated fund transfers table contains the following fields

| Field name | Data type | Size | Constraint  |
|------------|-----------|------|-------------|
| Date       | datetime  | 30   |             |
| TransferId | integer   | 20   | Primary key |
| Account1   | int       | 20   |             |
| Account2   | int       | 20   |             |
| VoucherNo  | nvarchar  | 20   |             |
| ContractNo | int       | 20   |             |
| Month      | nvarchar  | 10   |             |
| Amount     | float     | 30   |             |

**Table 5. 33:** Generated fund transfers table

#### e) Operation table

This table contains the information of the operations that can be performed by a banking agent. Some of these operations include

1. **Internal transfer:** This transfer takes place from the operator's account to local bank account. This transfer is done by the banking agent after receiving payment instruction message from the payment agent. In this transfer, funds are removed from the operator's account and added to the local bank's account. The funds include the amount to be paid to the distant operator and bank charges.
2. **External transfer:** This transfer takes place from the distant bank's account to the local operator's account. The transfer is done by the banking agent to effect the payment received from the distant banking agent on behalf of the local operator. This transfer is done when the banking agent receives fund transfer message from the distant banking agent. In this transfer, the amount to be paid to the local operator and the bank charge are deducted from the distant bank's account. The amount is added to the local operator's account and the bank charge is added to the local bank's account – as transfer service charge.
3. **Dual transfer:** In this transfer, funds are removed from the local operator's account and added directly to the distant operator's account. This transfer is done if both operators' accounts are maintained by the same bank. The transfer process is initiated when the banking agent receive payment instruction message from the payment agent. In this case, the amount to be paid to the distant operator and charge funds are deducted from the local operator's (the operator that initiated the payment instruction message) account. The amount is added to the distant operator's (the operator to be paid) account and the charge is added to the local bank's account.
4. **Withdrawal:** Removal of fund from an account
5. **Deposit:** Adding fund to an account

The operation table contains the following fields

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| Date       | datetime  | 30   |            |
| Type       | integer   | 20   |            |
| Name       | nvarchar  | 20   |            |
| Account    | integer   | 20   |            |
| Charge     | float     | 10   |            |
| Amount     | float     | 30   |            |
| Balance    | float     | 30   |            |

**Table 5. 34:** Operation table

**g) Received transfers table**

Once the local banking agent has generated fund transfer information, it generates and sends a fund transfer message to the distant banking agent. The contents of the fund transfer message are the values of the fields in the generated fund transfer instance. The fund transfer message is received by the distant banking agent and its contents are stored in the received transfer table. The fund transfer message is generated only if the two operators' accounts are maintained by different banks. Once the content of the fund transfer message is stored, the banking agent invokes the external transfer type of operation. After successful invocation of external transfer operation, the banking agent generates and send payment information message to the local payment agent. The generated payment information is stored in the payment information table. All the messages are generated in the form of objects (instances).

The structure of the received fund transfer table is similar to the generated fund transfer table. It has the following fields

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| Date       | datetime  | 30   |            |
| TransferId | integer   | 20   |            |
| Account1   | integer   | 20   |            |
| Account2   | integer   | 20   |            |
| VoucherNo  | nvarchar  | 20   |            |
| ContractNo | integer   | 20   |            |
| Month      | nvarchar  | 10   |            |
| Amount     | float     | 30   |            |

**Table 5. 35:** Received transfers table



**i) Charge table:** The tables contains the different charges levied on the services rendered. It has the following fields.

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| ChargeId   | integer   | 20   |            |
| Name       | nvarchar  | 20   |            |
| value      | float     | 10   |            |

**Table 5. 36:** Charge table

**k) Feedback table:** Contains the feedback information messages description.

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| Number     | integer   | 20   |            |
| Message    | nvarchar  | 40   |            |

**Table 5. 37:** FeedBack table

**l) OrdinaryTransfer table:** This table contains the transfers done over the counter.

| Field name | Data type | Size | Constraint  |
|------------|-----------|------|-------------|
| Date       | datetime  | 30   |             |
| TransferId | integer   | 20   | Primary key |
| Account1   | integer   | 20   |             |
| Account2   | integer   | 20   |             |
| Charge     | integer   | 10   |             |
| Amount     | float     | 30   |             |

**Table 5. 38:** Ordinary transfer table

### m) Banking user table

This table contains the configuration of the people who are supposed to use the banking system.

| Field name | Data type | Size | Constraint  |
|------------|-----------|------|-------------|
| Date       | datetime  | 20   | not null    |
| System     | nvarchar  | 20   | not null    |
| Category   | nvarchar  | 20   | not null    |
| Name       | nvarchar  | 20   | not null    |
| Identity   | integer   | 20   | Primary key |
| Password   | nvarchar  | 20   | not null    |

**Table 5. 39:** Banking user table

**n) Banking Login table:** This table contains the users who login and logout of banking system.

| Field name | Data type | Size | Constraint |
|------------|-----------|------|------------|
| StartDate  | datetime  | 20   | not null   |
| EndDate    | datetime  | 20   | not null   |
| Duration   | integer   | 20   | not null   |
| System     | nvarchar  | 20   | not null   |
| Category   | nvarchar  | 20   | not null   |
| Name       | nvarchar  | 20   | not null   |
| Identity   | integer   | 20   | Not null   |
| Password   | nvarchar  | 20   | Not null   |

**Table 5. 40:** Banking Login table

## 5.3 Agent class diagrams

In this diagram, each agent constituting the AMBIBPSY system is represented with a UML class diagram. Since we are using JADE frame work to design the agents, all these agents are designed as subclasses of a GuiAgent class that is already provided in JADE. The newly created agent inherits all public properties and behaviors of the JADE GuiAgent class. The implementation of each agent program is based on the tasks to be performed, services provided and the goals to be achieved as illustrated in agent diagram in analysis.

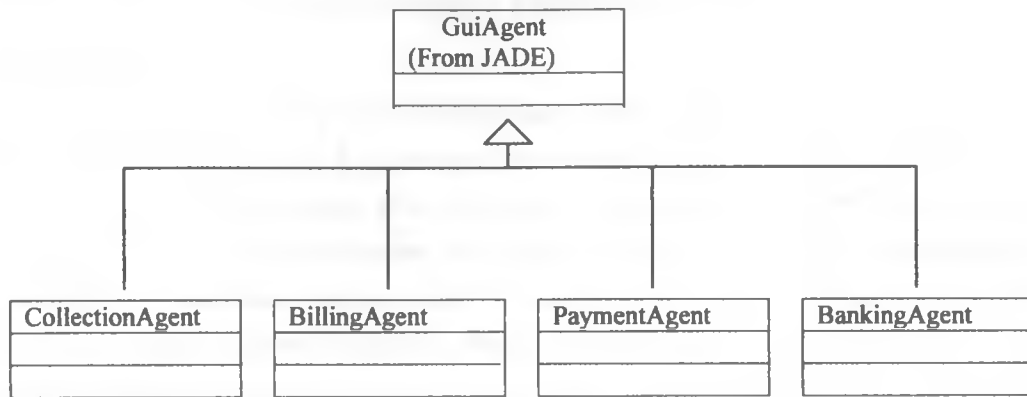


Figure 5. 20: Agent class diagram

### 5.4 Agent instance acquaintance Object diagram

This diagram represents the instances of each agent class and their acquaintances. Each instance has a link to all other instances it can interact with.

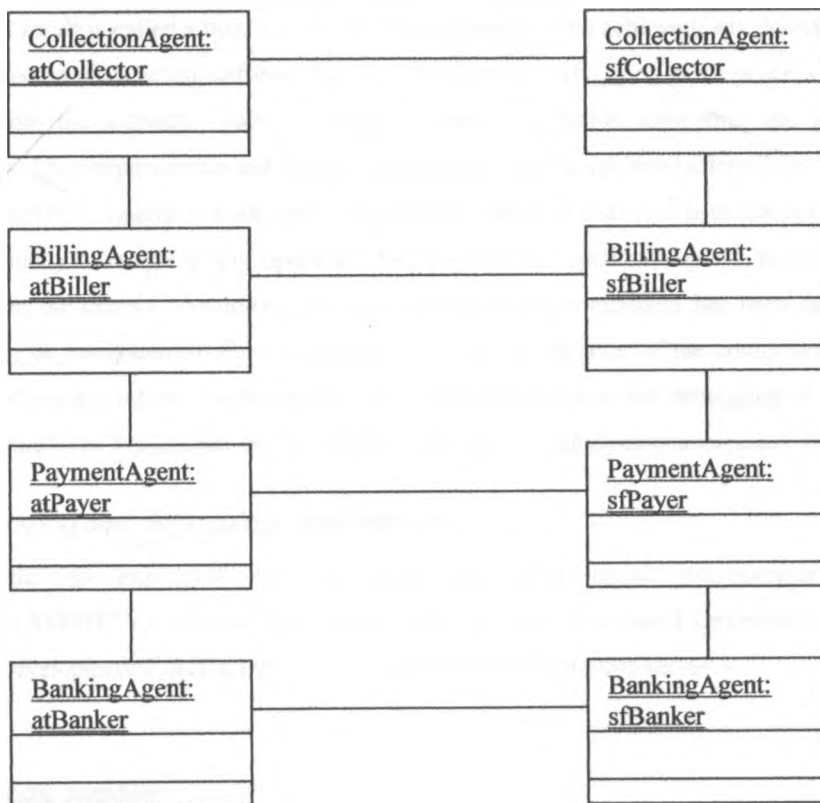


Figure 5. 21: Agent instance acquaintance Object diagram

# CHAPTER 6: CODING AND DEBUGGING

## **6.0 Introduction**

Any system is developed for the purpose of helping users perform their work with ease. The system should meet user requirements as approved by the users. For user requirements to be achieved, they are translated into system requirements and design. Once the design is successfully completed, coding and debugging process begins. A user will use and approve the system to meet his/her requirements after successful coding and debugging is completed. This is a very challenging stage of system development. It requires a programming language, the environment in which the programming language is going to be used, the database management system that the system will interact with and any other software required for the development of the system.

## **6.1 Coding and debugging**

Coding is the process of translating the system specifications into program code statement that are the compiled and executed to produce results or outputs for the user. A written program can refuse to compile, or can compile and refuse to execute. A program code can compile and execute successfully, but fail to produce correct result. The error that can prevent the program from compiling, executing and producing the correct result is called a bug. For the user requirements to be achieved, any detected bug must be removed. The process of detecting software bug is called testing. Debugging is the process of removing the detected bug from the software system to enable proper compilation, execution and generation of the correct outputs. User requirements and system requirements will be achieved after all the bugs have been removed.

AMBIPSY system is built using components called classes. Each component is coded and debugged individually to ensure proper operation. For a system to function properly, there must be some interaction between the classes constituting the system. After each component has been debugged, the components need to be integrated to allow communication. The integration of the components also requires program code which should also be debugged. After proper integration and debugging of all system components, a fully functional system that can be tested by the user for deployment acceptance is realised.

## **6.2 Coding and debugging requirements**

In order to implement the automated multiagent based interoperator billing and payment system(AMBIPSY) we have used WINDOWS XP2006, Java Agent Development(JADE 4.0) framework, Java Development Kit(JDK) version 6.1 and Microsoft SQL2005 server.

## **6.3 Code samples**

Due to enormous code used to develop the system, we could not print all of them in this space, hence code samples of some components are attached in appendix 5.

# CHAPTER 7: TESTING AND SYSTEM EVALUATION

## 7.0 Introduction

Testing is a process of detecting errors in a software program. The detected errors must be corrected to ensure that the software meets its specifications and user requirements. Any testing process applied throughout software development is either static or dynamic. Static testing process involves reading and checking documents concerning requirements, reading and checking of code without running the software. Dynamic testing process involves running the code to check its output. Both types of testing process have been done. Dynamic testing can be done using two approaches – black box testing and white box testing. Black box testing treats the system as a black box that receives inputs and produce outputs. It is based on assessment of requirements and functionality of the system. White box testing involves the coverage of code statements, branches, paths and conditions of execution. It tests the internal logic of the application's code. The documentation in this chapter covers types of dynamic testing and evaluation of the test results. In order to conduct testing process, a test plan is required. The content of a test plan includes:

- The identity of the component to be tested
- The purpose of the test
- Condition for carrying out the test
- Test data to be used(includes correct and incorrect data)
- Expected result

## 7.1 Component testing

This refers to testing of each individual module in a suite of programs. Each module is tested with correct and incorrect data to reveal loopholes in the code. It often takes place in parallel with coding to provide intermediate results that reveal logic, functionality and error handling of a unit. This testing is done at the source or code level for language, specifically for identifying programming errors such as bad syntax, logic errors or to test particular functions or code modules.

## 7.2 Functional/integration testing

Specific functional user requirements would be tested to ensure that the system meets these requirements and ensure that the developer has developed the right product. Each user specific function would be thoroughly tested to ensure that they perform their tasks properly. The test would try to find errors within inputs and outputs of these functions. The results display and format would be closely examined to make sure that it meets user requirements. This test would also include tests such as:

- The interfaces to ensure that they are functioning well by checking if the interface components are properly integrated and behaving as expected – especially verifying that the appropriate action is associated with correct interaction method of designed mouse click events, key strokes etc.

- The interaction of the system's graphical user interface (GUI) and the backend database. In this case, data would be inserted from the front end using data entry forms and check if each entry corresponds to the correct database table field, also check if data inserted is in the desired format.
- The form interaction design would also be tested to ensure that the information presentation to the user is in a manner that the users can easily find and/or perform the operation they want. The design should be an abstract representation of the abstract physical forms that the system employees use.

### **7.3 System testing**

The system would be properly tested to verify its functionality. This would be achieved by ensuring that relevant tests have been performed such as integration tests, unit tests, functional tests and acceptance tests. This test would be used as the overall testing of the system after the system has fully been developed just before it is taken to the client, where the client will perform acceptance testing with the developer and raise any adjustments they want at different system levels. This type of testing would be designed to ensure that the system meets all functional and non functional user requirements. This test would also focus on the behavior of the system once it is subjected to different inputs. It would also focus on how different components of the system interact with each other for optimal performance of the system (integration testing). Different user demand would be tested against the system which would include error message testing and several screen mappings.

### **7.4 Acceptance testing**

This is done by the system client and after all other types of testing have been performed. The system is taken to the commissioner to test whether it adheres to the agreed upon functional requirements in System Requirements Specification Document (SRS). The client also uses black box testing to evaluate the system functionality. Mission critical errors in functional requirements are corrected immediately and regression testing performed to ensure that changes made do not affect other system components or functions. Upon successful acceptance testing, the system will then be implemented; a process that will see the developed system is in place and working as intended.

### **7.5 Performance testing**

This type of testing relates to the expected level of the processing time of the transaction or response time especially when the system is being fully utilized.

### 7.6 Volume testing

This testing ensures that the system can handle the expected number of users or transaction's processing speed.

### 7.7 Regression testing

This involves ensuring that the system corrections do not result in the same or other errors as corrections are being made.

### 7.8 Evaluation testing

For accurate results on testing, once alpha test is finished an independent body will be assigned to carry out a beta test. This will guarantee software quality since the body does not have idea on the system structure. Therefore they will use different test data which may help in discovering more errors. Their recommendations will be considered and errors corrected accordingly. Later the system will be released to the end user. For the purpose of software maintenance and documentation, all the test records and test logs will be recorded and documented for future reference both for alpha and beta test.

### 7.9 Tests on Collection System

| SN | Component being tested   | Purpose for the test                   | Condition for testing          | Test data   | Expected result                                  |
|----|--------------------------|--|--------------------------------|---|--|
| 1  | CollectionAgent          | To fire CollectionAgent                | DOS command line interface     | java jade.Boot – host Safaricom sfCollector:CollectionAgent | Login GUI displays                               |
| 2  | CollectionUser           | Correct Login to display main menu GUI | Login GUI                      | User name and password                                      | Main menu form display                           |
| 3  | CollectionPasswordChange | To ensure successful password change   | Collection Change password GUI | User name, old password and new password                    | old password replaced by new password            |
| 4  | CollectionSystemMenu     | Correct access of all menu items       | Main menu form                 | Selected menu item  | Display of the selected menu item                |
| 5  | NewPOI                   | Confirm successful creation of new POI | Create New_POI GUI             | poiid, name and operatorId                                  | Creation and storage of new POI in the POI table |
| 6  | NewOperator              | Confirm successful creation of new     | Create New_Operat or GUI       | OperatorId and name   | Creation and storage of new                      |

|    |                       |  |                                    |  |   |
|----|-----------------------|--|------------------------------------|--|---|
|    |                       | operator   |                                    |  | operator in the Operator table  |
| 7  | NewRouting            | Ensure successful creation of new routing                    | Create New_Routing GUI             | RouteId, Route type, origin and destination                                | Creation and storage of new routing in the Routing table                        |
| 8  | NewRouteDefinition    | Confirm successful creation of new route definition          | Create New_Route_Definition GUI    | RouteId, PoiId, destination and route type                                 | Creation and storage of new route definition in the RouteDefinition table       |
| 9  | NewDisputeDescription | Confirm successful creation of new dispute description       | Create New_Dispute_Description GUI | DisputeId and description  | creation and storage of new dispute description in the DisputeDescription table |
| 10 | CDRDispute            | Confirm successful creation of new CDRDispute                | Create New_CDRDispute GUI          | Date, disputeId, origin, destination, dispute type, description and status | creation of new CDRDispute and storage in the CDRDispute table                  |
| 11 | NewCollectionUser     | To confirm successful addition of new user                   | Create New_Collection_user         | Date, system, category, name, identityNo and password                      | creation and storage of new collection user in the CollectionUser table         |
| 12 | ModifyPOI             | To ensure correct modification is done on the POI data       | Modify_POI GUI                     | PoiId, name and operatorId   | Successful modification of the selected data item in the POI table              |
| 13 | ModifyOperator        | To confirm the correct changes in the selected operator data | Modify_Operator GUI                | OperatorId and name  | Successful modification of the selected operator data item in Operator table    |
| 14 | ModifyRouting         | Confirmation of  | Modify Routing                     | RouteId, route   | Successful  |



|    |                          |   |                                |  |  |
|----|--------------------------|---|--------------------------------|--|--|
|    |                          | correct modification of the selected routing data item                    | ting GUI                       | type, origin and destination   | modification of the selected routing data item in Routing table                |
| 15 | ModifyRouteDefinition    | To confirm the correct changes on the selected route definition data item | Modify_Route_definition GUI    | RouteId, Poild, destination and route type                                 | Correct modification of the selected data item in the RouteDefinition table    |
| 16 | ModifyDisputeDescription | To confirm the correct changes on the selected route definition data item | Modify_Dispute_Description GUI | DisputeId and description  | Correct modification of the selected data item in the DisputeDescription table |
| 17 | ModifyCDRDispute         | To confirm correct modification of the selected CDRDispute data item      | Modify_CD RDispute GUI         | Date, disputeId, origin, destination, dispute type, description and status | Successful modification of the selected data item in the CDRDispute table      |
| 18 | ModifyCollectionUser     | Correct changes in the selected data item of the collection user          | Modify_Collection_User GUI     | Date, system, category, name, identityNo and password                      | Correct changes of the selected data item in the CollectionUser table          |
| 19 | DeletePOI                | Confirm deletion of a selected POI  | Delete_POI GUI                 | Poild  | Selected POI removed from the POI table  |
| 20 | DeleteOperator           | To confirm deletion of the selected operator                              | Delete_Operator GUI            | OperatorId   | Successful removal of the selected operator data from the Operator table       |
| 21 | DeleteRouting            | Confirm successful deletion of the selected routing data                  | Delete_Routing GUI             | RouteId  | Selected routing data deleted from the Routing table                           |
| 22 | DeleteRouteDefinition    | to confirm successful deletion of the                                     | Delete_Route_definition GUI    | RouteId  | Successful deletion of the selected  |

|    |                          |   |   |                                  |   |
|----|--------------------------|---|---|----------------------------------|---|
|    |                          | selected route definition   |   |                                  | route definition data from the RouteDefinition table  |
| 23 | DeleteDisputeDescription | confirm successful deletion of the selected dispute description data              | Delete_Dispute_Description GUI            | DisputeId                        | Successful removal of the selected dispute description from the DisputeDescription table      |
| 24 | DeleteCDRDispute         | To confirm successful removal of the chosen CDRDispute data from CDRDispute table | Delete_CDR_Dispute GUI                    | DisputeId                        | Successful deletion of the selected CDRDispute data from the CDRDispute tables                |
| 25 | DeleteCollectionUser     | To confirm successful deletion of the selected collection user                    | Delete_Collection_User GUI                | IdentityNo                       | Successful removal of the selected collection user from the CollectionUser table              |
| 26 | DeleteCollectionLogin    | To confirm successful deletion of a selected collection of login users            | Delete_Collection_Login GUI               | StartDate                        | Successful deletion of logins from CollectionLogin table as indicated in StartDate text field |
| 27 | CollectionSystemMenu     | To confirm successful display of POI in the POI table                             | Main menu, View_POI menu item             | View_POI menu item               | Display of View_POI GUI with a list of POI  |
| 28 | CollectionSystemMenu     | To confirm successful display of View_Operator GUI and a list of operators        | Main menu GUI and View_Operator menu item | Selected View_Operator menu item | Successful display of View_Operator GUI and a list of operators                               |
| 29 | CollectionSystemMenu     | To confirm successful display of View_Routing GUI and a list of routings          | main menu, View_Routing menu item         | Selected View_Routing menu item  | Successful display of View_Routing GUI and a list of routing information                      |

|    |                      |   |  |  |  |
|----|----------------------|---|--|--|--|
| 30 | CollectionSystemMenu | To confirm successful display of View_Route_Definition GUI and a list of route definition data                                      | Main menu, View_Route_Definition menu item | View_Route_Definition menu item                      | Display of View_Route_definition GUI and a list of route definition  |
| 31 | CollectionSystemMenu | To confirm successful display of View_Collection_User GUI and a list of users   | Main menu, View_Collection_User menu item  | View_Collection_User menu item                       | Display of View_Collection_User GUI and a list of users  |
| 32 | CollectionSystemMenu | To confirm display of View_Collection_Login GUI and a list of login users on specified dates  | Main menu, View_Collection_Login menu item | IdentityNo, start date and end date                  | Successful display of View_Collection_Login GUI and a list of collection login users for the specified dates |
| 33 | CollectionSystemMenu | To confirm the correct display of View_RawCDR GUI, capture and display of a list of rawCDR between two operators on specified dates | Main menu, View_RawCDR menu item           | OriginatorId, destinationId, start date and end date | Correct display of the View_RawCDR GUI and a list of rawCDR between two operators on specified dates         |
| 34 | CollectionSystemMenu | To confirm correct reconciliation of CDR, display of a list of reconciledCDR on specified dates, and View_ReconciledCDR GUI         | Main menu, View_ReconciledCDR menu item    | OriginatorId, destinationId, start date and end date | Correct display of View_ReconciledCDR GUI and a list of reconciledCDR on specified dates                     |

### 7.10 Tests on Billing system

| SN | Component being tested | Purpose for the test  | Condition for testing       | Test data  | Expected result  |
|----|------------------------|---|-----------------------------|--|--|
| 1  | BillingAgent           | To fire BillingAgent  | DOS command line interface  | java jade.Boot – host Safaricom sfBill:BillingAgent                          | Display of Login GUI   |
| 2  | BillingUser            | Correct Login to display main menu GUI                          | Login GUI                   | User name and password   | Main menu form display   |
| 3  | BillingPasswordChange  | To ensure successful password change                            | Billing Change password GUI | User name, old password and new password                                     | Ability of the user to login using the new password but not the old password |
| 4  | BillingSystemMenu      | Correct access of all menu items                                | Main menu form              | Selected menu item   | Display of the selected menu item  |
| 5  | NewContract            | Confirm successful creation of new Contract                     | Create New_Contract GUI     | Date, contactNo, operator1, operator2, operator1Id, operator2Id and validity | Creation and storage of new contract in the Contract table                   |
| 6  | NewBillingRate         | Confirm successful creation of new billing rate                 | Create New_Billing_Rate GUI | RateId, name and value   | Creation and storage of new billing rate in the BillingRate table            |
| 7  | NewBillingUser         | To confirm successful addition of new user                      | Create New_Billing_User GUI | Date, system, category, name, identityNo and password                        | creation and storage of new billing user in the BillingUser table            |
| 8  | ModifyContract         | To ensure correct modification is done on the Contract data     | Modify_Contract GUI         | Date, contactNo, operator1, operator2, operator1Id, operator2Id and validity | Successful modification of the selected data item in the Contract table      |
| 9  | ModifyBillingRate      | To ensure correct modification is done on the billing rate data | Modify_Billing_Rate GUI     | RateId, name and value   | Successful modification of the selected data item in the BillingRate table   |
| 10 | ModifyBillingUser      | To ensure correct modification is done on the billing user data | Modify_Billing_User GUI     | Date, system, category, name, identityNo and password                        | Successful modification of the selected data item in the BillingUser         |

|    |                    |  |  |  |  |
|----|--------------------|--|--|--|--|
|    |                    |  |  |  | table  |
| 11 | DeleteContract     | Confirm successful deletion of a selected contract   | Delete_Contract GUI                      | ContractNo   | Selected contract removed from the Contract table  |
| 12 | DeleteBillingRate  | Confirm successful deletion of a selected billing rate   | Delete_Billing_Rate GUI                  | RateId   | Selected billing rate removed from the BillingRate table   |
| 13 | DeleteBillingUser  | Confirm successful deletion of a selected billing user   | Delete_Billing_User GUI                  | IdentityNo   | Selected billing user removed from the BillingUser table   |
| 14 | DeleteBillingLogin | To confirm successful deletion of a selected billing login users   | Delete_Billing_Login GUI                 | Start date   | Successful deletion of logins from BillingLogin table as indicated in StartDate text field             |
| 15 | BillingSystemMenu  | To confirm successful display of contract data from the Contract table   | Main menu, View_Contract menu item       | Selected View_Contract menu item                     | Display of View_Contract GUI with a list of contracts  |
| 16 | BillingSystemMenu  | To confirm successful display of billing rates in the BillingRate table  | Main menu, View_Billing_Rate menu item   | Selected View_Billing_Rate menu item                 | Display of View_Billing_Rate GUI with a list of billing rates  |
| 17 | BillingSystemMenu  | To confirm successful display of billing users in the BillingUser table  | Main menu, View_Billing_User menu item   | Selected View_Billing_User menu item                 | Display of View_Billing_User GUI with a list of billing users  |
| 18 | BillingSystemMenu  | To confirm display of View_Billing_Login GUI and a list of login users on specified dates  | Main menu, View_Billing_Login menu item  | IdentityNo, start date and end date                  | Successful display of View_Billing_Login GUI and a list of billing login users for the specified dates |
| 19 | BillingSystemMenu  | To confirm the correct display of View_Generated_Bill GUI and a list of generated bills between two operators on specified dates | Main menu, View_Generated_Bill menu item | OriginatorId, destinationId, start date and end date | Correct display of the View_Generated_Bill GUI and a list of generated bills between                   |

|    |                   |  |   |  |   |
|----|-------------------|--|---|--|---|
|    |                   |  |   |  | two operators on specified dates  |
| 20 | BillingSystemMenu | To confirm the correct display of View_Received_Bill GUI and a list of received bills between two operators on specified dates                   | Main menu, View_Received_Bill menu item   | OriginatorId, destinationId, start date and end date | Correct display of the View_Received_Bill GUI and a list of received bills between two operators on specified dates     |
| 21 | BillingSystemMenu | To confirm the correct display of View_Reconciled_Bill GUI and a list of reconciled bills between two operators on specified dates               | Main menu, View_Reconciled_Bill menu item | OriginatorId, destinationId, start date and end date | Correct display of the View_Reconciled_Bill GUI and a list of reconciled bills between two operators on specified dates |
| 22 | BillingSystemMenu | To confirm the correct display of View_ReconciledCDR GUI, capture and display of a list of reconciledCDR from CollectionAgent on specified dates | Main menu, View_Reconciled_CDR menu item  | OriginatorId, destinationId, start date and end date | Correct display of the View_ReconciledCDR GUI and a list of reconciledCDR from CollectionAgent on specified dates       |

### 7.11 Tests on Payment system

| SN | Component being tested | Purpose for the test                              | Condition for testing      | Test data   | Expected result              |
|----|------------------------|---|----------------------------|---|------------------------------|
| 1  | PaymentAgent           | To fire PaymentAgent                              | DOS command line interface | java jade.Boot – host Safaricom sfPayer:Payment Agent | Login GUI display            |
| 2  | PaymentUser            | To confirm correct Login to display main menu GUI | Login GUI                  | User name and password                                | Main menu form display       |
| 3  | PaymentPasswordChange  | To ensure successful                              | Payment Change             | User name, old password and                           | Ability of the user to login |

|    |                       |  |                                  |  |  |
|----|-----------------------|--|----------------------------------|--|--|
|    |                       | password change  | password GUI                     | new password   | using the new password but not the old password                                |
| 4  | PaymentSystemMenu     | To ensure correct access of all menu items                                   | Main menu form                   | Selected menu item   | Display of the selected menu item  |
| 5  | NewOperatorAccount    | To confirm successful creation of a new operator account                     | Create New_Operat or_Account GUI | Operator1, operator2, operator1Id, operator2Id, operator1Account, operator2Account, Operator1Bank, operator2Bank, contractNo | Creation and storage a new operator in the OperatorAccount table               |
| 6  | NewPaymentUser        | Confirm successful creation of a new payment user                            | Create New_Payment_User GUI      | Date, system, category, name, identityNo and password  | Creation and storage a new user in the PaymentUser table                       |
| 7  | ModifyOperatorAccount | To ensure correct modification is done on the selected operator account data | Modify_Operator_Account GUI      | Operator1, operator2, operator1Id, operator2Id, operator1Account, operator2Account, Operator1Bank, operator2Bank, contractNo | Successful modification of the selected data item in the OperatorAccount table |
| 8  | ModifyPaymentUser     | To ensure correct modification is done on the selected user data             | Modify_Payment_User GUI          | Date, system, category, name, identityNo and password  | Successful modification of the selected data item in the PaymentUser table     |
| 9  | DeleteOperatorAccount | To confirm successful deletion of the selected operator account              | Delete_Operator_Account GUI      | Operator2Account   | Selected operator account is removed from the OperatorAccount table            |
| 10 | DeletePaymentUser     | To confirm successful deletion of the selected user                          | Delete_Payment_User GUI          | IdentityNo   | Selected user is removed from the PaymentUser table                            |
| 11 | DeletePaymentLogin    | To confirm successful deletion   | Delete_Payment_Login             | Start date   | Successful deletion of   |

|    |                   |  |  |  |   |
|----|-------------------|--|--|--|---|
|    |                   | of a selected payment login users  | GUI  |  | logins from PaymentLogin table as indicated in StartDate text field   |
| 12 | PaymentSystemMenu | To confirm display of operator accounts from OperatorAccount table   | Main menu, View_ Operator_Ac count menu item | Operator1Id and operaor2Id                           | Display of View_ Operator_Ac count GUI with a set of operator account data as specified by Operator1Id and Operaor2Id |
| 13 | PaymentSystemMenu | To confirm successful display of users from the PaymentUser table  | Main menu, View_Paym ent_user menu item      | Selected View_ Payment_user menu item                | Display of View_ Payment_user GUI with a list of payment users  |
| 14 | PaymentSystemMenu | To confirm display of View_ Payment_Login GUI and a list of login users on specified dates   | Main menu, View_ Payment_Login menu item     | IdentityNo, start date and end date                  | Display of View_ Payment_Login GUI and a list of payment login users for the specified dates                          |
| 15 | PaymentSystemMenu | Confirm the display of View_ Received_Bill GUI, reception and display of a list of received bills from the BillingAgent on specified dates               | Main menu, View_Recei ved_Bill menu item     | OriginatorId, destinationId, start date and end date | Correct display of the View_ Received_Bill GUI and a list of received bills on specified dates                        |
| 16 | PaymentSystemMenu | To confirm the correct display of View_Reconciled_Bill GUI, reception and display of a list of reconciled bills from the BillingAgent on specified dates | Main menu, View_Reconciled_Bill menu item    | OriginatorId, destinationId, start date and end date | Display of View_Reconciled_Bi ll GUI and a list of reconciled bills on specified dates                                |
| 17 | PaymentSystemMenu | To confirm display of View_Generated_V oucher GUI and a list of generated vouchers from the  | Main menu, View_Generated_Voucher menu item  | OriginatorId, destinationId, start date and end date | Display of View_Generated_Vo ucher GUI and a list of generated  |



|    |                   |   |   |  |  |
|----|-------------------|---|---|--|--|
|    |                   | GeneratedVoucher table on specified dates   |   |  | vouchers on specified dates  |
| 18 | PaymentSystemMenu | To confirm display of View_Received_Voucher GUI and a list of received vouchers from the distant PaymentAgent on specified dates    | Main menu, View_Received_Voucher menu item    | OriginatorId, destinationId, start date and end date | Display of View_Received_Voucher GUI and a list of received vouchers on specified dates    |
| 19 | PaymentSystemMenu | To confirm display of View_Reconciled_Voucher GUI and a list of received vouchers from the distant PaymentAgent on specified dates  | Main menu, View_Reconciled_Voucher menu item  | OriginatorId, destinationId, start date and end date | Display of View_Reconciled_Voucher GUI and a list of received vouchers on specified dates  |
| 20 | PaymentSystemMenu | To confirm display of View_Payment_Instruction GUI and a list of payment instructions to the local BankingAgent on specified dates  | Main menu, View_Payment_Instruction menu item | OriginatorId, destinationId, start date and end date | Display of View_Payment_Instruction GUI and a list payment instructions on specified dates |
| 21 | PaymentSystemMenu | To confirm display of View_Payment_Information GUI and a list of payment information from the local BankingAgent on specified dates | Main menu, View_Payment_Information menu item | OriginatorId, destinationId, start date and end date | Display of View_Payment_Information GUI and a list payment information on specified date   |

### 7.12 Tests on Banking system

| SN | Component being tested | Purpose for the test | Condition for testing      | Test data   | Expected result      |
|----|------------------------|----------------------|----------------------------|---|----------------------|
| 1  | BankingAgent           | To fire BankingAgent | DOS command line interface | java jade.Boot – host Safaricom sfBanker:BankingAgent | Display of Login GUI |
| 2  | BankingUser            | To confirm correct   | Login GUI                  | User name and   | Display of           |

|    |                       |  |                                 |   |  |
|----|-----------------------|--|---------------------------------|---|--|
|    |                       | login and display of main menu GUI   |                                 | password  | Main menu form   |
| 3  | BankingPasswordChange | To ensure successful password change   | Banking Change password GUI     | User name, old password and new password              | Ability of the user to login using the new password but not the old password   |
| 4  | BankingSystemMenu     | To confirm correct access of all menu items                                  | Main menu form                  | Selected menu item                                    | Display of the selected menu item  |
| 5  | NewCustomerAccount    | Confirm successful creation of a new customer account                        | Create New Customer Account GUI | Date, accountNo, name and balance                     | New customer account is created and stored in the CustomerAccount table        |
| 6  | NewCharge             | To confirm successful creation of a new charge                               | Create New Charge GUI           | Date, chargeId, name and value                        | New charge is created and stored in the Charge table                           |
| 7  | NewFeedback           | To confirm successful creation of a new feedback                             | Create New Feedback GUI         | Number and message                                    | New Feedback is created and stored in the Feedback table                       |
| 8  | NewBankingUser        | To confirm successful creation of a new banking user                         | Create New Banking User GUI     | Date, system, category, name, identityNo and password | New banking user is created and stored in the BankingUser table                |
| 9  | ModifyCustomerAccount | To ensure correct modification is done on the selected customer account data | Modify Customer Account GUI     | Date, accountNo, name and balance                     | Successful modification of the selected data item in the CustomerAccount table |
| 10 | ModifyCharge          | To ensure correct modification is done on the selected charge data           | Modify Charge GUI               | Date, chargeId, name and value                        | Successful modification of the selected data item in the Charge table          |
| 11 | ModifyFeedback        | To ensure correct modification is done on the selected feedback data         | Modify Feedback GUI             | Number and message                                    | Successful modification of the selected data item in the Feedback table        |
| 12 | ModifyBankingUser     | To ensure correct modification is done on the selected user data             | Modify Banking User GUI         | Date, system, category, name, identityNo and password | Successful modification of the selected data item in the                       |

|    |                       |  |  |                                      |   |
|----|-----------------------|--|--|--------------------------------------|---|
|    |                       |  |  |                                      | BankingUser table   |
| 13 | DeleteCustomerAccount | To confirm successful deletion of the selected customer account data                               | Delete_Customer_Account GUI                | AccountNo                            | Selected customer account data removed from the CustomerAccount table                                   |
| 14 | DeleteCharge          | To confirm successful deletion of the selected charge data   | Delete_Charge GUI                          | ChargeId                             | Selected charge removed from the Charge table   |
| 15 | DeleteFeedback        | To confirm successful deletion of the selected feedback data                                       | Delete_Feedback GUI                        | Number                               | Selected feedback data is removed from the Feedback table   |
| 16 | DeleteBankingUser     | To confirm successful deletion of the selected banking user data                                   | Delete_Banking_User GUI                    | IdentityNo                           | Selected banking user data is removed from the BankingUser table  |
| 17 | DeleteBankingLogin    | To confirm successful deletion of a selected collection of login users                             | Delete_Banking_Login GUI                   | Start date                           | Deletion of logins from BankingLogin table as indicated in StartDate text field                         |
| 18 | BankingSystemMenu     | To confirm display of View_Customer_Account GUI and a list of customer accounts on specified dates | Main menu, View_Customer_Account menu item | AccountNo, Start date and end date   | Successful display of View_Customer_Account GUI and a list of customer accounts for the specified dates |
| 19 | BankingSystemMenu     | To confirm successful display of charges from the Charge table                                     | Main menu, View_Charge menu item           | Selected View_Charge menu item       | Display of View_Charge GUI with a list of charges   |
| 20 | BankingSystemMenu     | To confirm display of View_Feedback GUI and a list of feedback from the Feedback table             | Main menu, View_Feedback menu item         | Selected View_Feedback menu item     | Display of View_Feedback GUI with a list of feedback data   |
| 21 | BankingSystemMenu     | To confirm successful display of   | Main menu, View_Banking_User               | Selected View_Banking_User menu item | Display of View_Banking_User GUI  |

|    |                   |   |   |   |   |
|----|-------------------|---|---|---|---|
|    |                   | View_Banking_Us<br>er GUI and a list of<br>banking users from<br>the BankingUser<br>table   | menu item   |   | with a list of<br>banking users   |
| 22 | BankingSystemMenu | To confirm display<br>of<br>View_Banking_Lo<br>gin GUI and a list<br>of login users on<br>specified dates   | Main menu,<br>View_Banki<br>ng_Login<br>menu item           | IdentityNo, start<br>date and end date                        | Display of<br>View_Banking<br>_Login GUI<br>and a list of<br>banking login<br>users for the<br>specified dates                  |
| 23 | BankingSystemMenu | To confirm display<br>of<br>View_Payment_Ins<br>truction GUI and a<br>list of payment<br>instructions from<br>the local<br>PaymentAgent on<br>specified dates       | Main menu,<br>View_Paym<br>ent_Instructi<br>on menu<br>item | OriginatorId,<br>destinationId,<br>start date and<br>end date | Correct<br>display of<br>View_<br>Payment_Inst<br>ruction GUI<br>and a list of<br>payment<br>instructions on<br>specified dates |
| 24 | BankingSystemMenu | To confirm display<br>of<br>View_Payment_Inf<br>ormation GUI and<br>a list of generated<br>payment<br>information to<br>local<br>PaymentAgent on<br>specified dates | Main menu,<br>View_Paym<br>ent_<br>Information<br>menu item | OriginatorId,<br>destinationId,<br>start date and<br>end date | Correct<br>display of the<br>View_Paymen<br>t_Information<br>GUI and a list<br>of payment<br>information on<br>specified dates  |
| 25 | BankingSystemMenu | To confirm the<br>correct display of<br>View_Ordinary_<br>Transfer GUI and a<br>list of ordinary<br>transfers on<br>specified dates                                 | Main menu,<br>View_<br>Ordinary_Tr<br>ansfer menu<br>item   | Account1,<br>account2, start<br>date and end date             | Correct<br>display of the<br>View_<br>Ordinary_Tran<br>sfer GUI and a<br>list of ordinary<br>transfers on<br>specified dates    |
| 26 | BankingSystemMenu | To confirm display<br>of<br>View_Operations<br>GUI and a list of<br>operations on<br>specified dates  | Main menu,<br>View_Opera<br>tions menu<br>item              | AccountNo, start<br>date and end date                         | Display of<br>View_<br>Operations<br>GUI and a list<br>of operations<br>on specified<br>dates                                   |
| 27 | BankingSystemMenu | To confirm the<br>correct display of<br>View_Generated_T<br>ransfer GUI and a<br>list of generated<br>transfers on<br>specified dates                               | Main menu,<br>View_Gener<br>ated_Transfe<br>r menu item     | Account1,<br>account2, start<br>date and end date             | Correct<br>display of the<br>View_Generat<br>ed_Transfer<br>GUI and a list<br>of generated<br>transfers on<br>specified dates   |
| 28 | BankingSystemMenu | To confirm display  | Main menu,  | Account1,   | Correct   |

|    |               |   |  |  |   |
|----|---------------|---|--|--|---|
|    |               | of View_Received_Transfer GUI and a list of received transfers from the distant BankingAgent on specified dates | View_Received_Transfer menu item       | account2, start date and end date                              | display of View_Received_Transfer GUI and a list of received transfers on specified dates |
| 29 | Deposit       | To confirm that the deposit amount is added to the relevant customer's account                                  | Main menu, Deposit Form                | Date, type, accountNo, accountName, charge, and amount         | Correct addition of deposit amount to the relevant customer's account balance             |
| 30 | Withdrawal    | To confirm that the withdrawal amount is deducted from the relevant customer's account                          | Main menu, Withdrawal Form             | Date, type, accountNo, accountName, charge, and amount         | Correct deduction of withdrawal amount from the relevant customer's account balance       |
| 31 | MakeOrdinary  | To confirm the correct transfer of fund from one account to another account                                     | Main menu, Make_Ordinary_Transfer Form | Date, type, transferId, account1, account2, charge, and amount | Successful transfer of the specified amount from Account1 to Account2                     |
| 32 | Balance       | To confirm the correct display of balance of the specified account  | Main menu, Balance Form                | AccountNo  | Display of balance of the specified account   |
| 33 | MiniStatement | To confirm the correct generation of mini statement of the selected customer account on specified               | Main menu, Mini_Statement Form         | Start date, end date and accountNo                             | Display of mini statement of the selected account on specified dates                      |
| 34 | Statement     | To confirm the correct generation of statement of the selected customer account on specified dates              | Main menu, Statement Form              | Start date, end date and accountNo                             | Display of statement of the selected account on specified dates                           |

### **7.13 Evaluation of the system**

System evaluation is carried out to ensure that the developed product meets user requirements and system specifications. When we were conducting the above tests, we successfully evaluated the functionalities of the system. Based on the accomplishment of the system functionalities, the system is able to achieve the following user requirements:

- Automatic collection of call detail from the point of inter-connection (POI)
- Storage CDR(raw data from the POIs)
- Formatting of data to suit the billing needs
- Carry out automatic CDR data reconciliation
- Automatically prepare and send the bill to the distant operator
- Reconcile the billing data with the distant operator
- Generate voucher against received bills
- Reconcile the received voucher
- Automatically generate payment instruction against reconciled voucher
- Send the payment instruction to bank and distant operator
- Receive payment information from the distant operator
- Reconciles payment with the distant operators
- Carry out inter-bank money transfer automatically
- Interaction with the administrator and user using GUI interface
- Provision of secure access to the system

## CHAPTER 8: CONCLUSION

### 8.1 Summary

Different telecommunication operators offer various services that are consumed by their customers. Each service offered should be billed in the most effective and efficient manner. Effective and efficient billing enables the customers to develop more confidence in the operator and its services. In this regard, each operator would strive to optimize its billing process in order to retain the current customers and also bring more clients on board. After making the customers happy by providing attractive billing process, the operators also require prompt payments of the generated bills. Synchronization of of billing and payment has been a problem. Post paid bills and inter-operator bills take time to prepare and deliver. This in turn results in delayed payments.

In order to solve inter-operator bill preparation, delivery and payment, we have developed automated multiagent based interoperator billing and payment system (AMBIBPSY). This system has been designed and implemented successfully using four agents. The four agents include: CollectionAgent, BillingAgent, PaymentAgent and BankingAgent. The multiagent system approach to this study is due to the distributed nature of points of interconnection from which the call detail records are captured. The use of agents has reduced the complexity of the system during development and when the system will require expansion afterwards. Each agent has been built and implemented successfully including all the roles it should play. CollectionAgent is used to receive and transform raw CDR from the POI into a format suitable for bill generation. BillingAgent is used to generate, store and deliver bill to the distant operator. PaymentAgent is used to generate and store voucher against the received bill from the distant operator. BankingAgent is used to effect the payment against the reconciled voucher by implementing EFT. The system is able to generate and send bill to the distant operator. Voucher generation and transmission to the distant operator has been accomplished successfully. Electronic fund transfer between the banking agents has been successful.

This system facilitates automatic inter-operator bill generation, delivery and payment. This will enable operators to spend no time and labour on bill generation, delivery and payment processing. With the incorporation of automatic fund transfer functionality, physical fund transactions will be eliminated completely as far as inter-operator billing and payment is concerned.

AMBIBPSY is capable of collecting rawCDR from points of interconnections (POIs). The rawCDR is transformed and made ready for bill generation. Since AMBIBPSY is associated with each point of interconnection, rawCDR and transformed CDR is stored in a distributed manner which enhances data security incase of point of interconnection failure. The automatically generated bills and payments are stored in the database for access and verification by the users at their convenience. The success of AMBIBPSY is therefore a major milestone in inter-operator billing and payment processing.

## **8.2 Challenges**

In most cases, researches pose a number of challenges. AMBIBPSY system development was not an exception. The area of research was considered sensitive, hence availability of some information was very hard. Some operators were very busy and had very little time for detailed discussion. This hindered the data collection process, hence overall system development time. In order to test the system, we had to use simulated data due to the considered sensitivity of billing and payment data. This made testing a bit difficult and lengthy as time had to be spent to generate the simulated data. The system looked simple during proposal stage but proved hard later due to its complexity in the coverage scope. Programming stage of the system proved to be the hardest. Programming the agents and ensuring effective communication between them posed a lot of challenges. Data type conversion, especially Date and time, during agents communication was very difficult to manage.

## **8.3 Recommendation for further study**

Since research subsists, AMBIBPSY system can also be improved to incorporate more functionalities in relation to other telecommunication billing areas. Apart from inter-operator billing and payment processing, further work can be done to include post paid billing and payment processing for individual clients. The area of post paid billing and payment processing can benefit the operators more than inter-operator billing and payment processing. This is due to the high number of customers in post paid scheme.



## REFERENCES

- Senez Raymond. Automated bill payment system. 2007. 8 Feb. 2010. <<http://www.patentstorm.us/patents/7200551/fulltext.html>>
- Billmax. Billing and provisioning system. 2008. 3 Feb. 2010. <<http://www.billmax.com/docs22/web/bxdocs.html>>
- Brian Henderson-Sellers, Paolo Girogini. Agent oriented methodologies. Idea Group Publishing, 2005
- Fabio Bellifemine, Giovanni Caire, Tiziana Trucco. Jade administrator's guide. 2010. 8 Apr. 2010. 11 May 2010. <<http://jade.tilab.com/doc/administratorsguide.pdf>>
- Fabio Bellifemine, Giovanni Caire, Tiziana Trucco. Jade programmer's guide. 2010. 8 Apr. 2010. 11 May 2010. <<http://jade.tilab.com/doc/programmersuide1.pdf>>
- Franco Zambonelli, Andrea Omicini, Paolo Ciancarini. Coordination Technologies for Internet Agents: Tutorial Coordination Technologies for Internet Agents. 1999.
- Franco Zambonelli, N. R. Jennings, Andrea Omicini, and M. Wooldridge. Agent-Oriented Software Engineering for Internet Applications. In Coordination of Internet Agents: Models, Technologies, and Applications, Springer-Verlag, Berlin, 2001.
- Giovanni Caire. Jade programmer's guide. 2009. 11 May 2010 <<http://jade.tilab.com/doc/JADEProgramming-Tutorial-for-beginners.pdf>>
- Heindel et al. Distributed electronic billing system with gateway interfacing biller and service center. 2001. 3 Feb. 2010. <<http://www.freepatentsonline.com/6304857.pdf>>
- Hilt et al. Electronic billpay system for consumers to generate messages directing financial institutions to pay a biller's bill. 2002. 5 Feb. 2010. <<http://www.freepatentsonline.com/6408284.pdf>>
- Methodology for Agent-oriented Software Engineering. 2001. 10 Feb. 2010. <<http://www.eurescom.de/~pub-deliverables/p900-series/907/T11/p907ti1.pdf>>
- Methodology for Agent-oriented Software Engineering. 2001. 10 Feb. 2010. <<http://www.eurescom.de/~pub-deliverables/p900-series/907/T12/p907ti2.pdf>>
- Michael Wooldridge. An Introduction to Multiagent Systems. JohnWiley & Sons Ltd., Indianapolis, 2002.
- Paolo Cancedda, Giovanni Caire. Creating ontologies by means of bean-ontology class. 2010. 11 May 2010. <<http://jade.tilab.com/doc/tutorials/BeanOntologyTutorial.pdf>>
- Kenya. Central Bank of Kenya. Payment system in Kenya. Sep. 2003. 8 Feb. 2010. <<http://www.centralbank.go.ke/downloads/nps%20old/psk.pdf>>

- Remington et al. Electronic bill presentment and payment system with bill dispute capabilities. 2005. 5 Feb. 2010. <<http://www.freepatentsonline.com/6968319.pdf>>
- Russel Norvig. Artificial intelligence: A modern approach, second edition, Prentice Hall, India, 2005.
- Y.Daniel Liang. Introduction to java programming, Eighth edition, Pearson Education. 2009.

# APPENDICES

## Appendix 1: Collection system test results

### 1.1 Agent activation

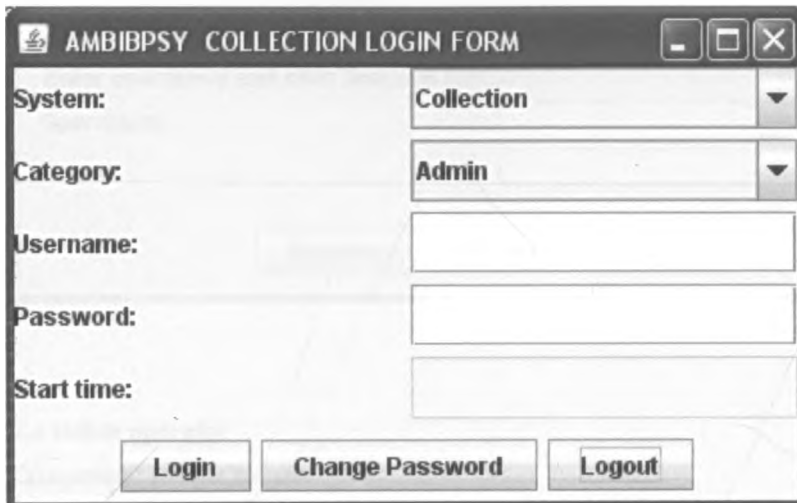
Component: CollectionAgent

Purpose: Execute Agent program and display Login GUI

Condition of testing: DOS command line interface

Test data: java jade.Boot -host Safaricom sfCollector:CollectionAgent

Expected result: Login GUI display



The screenshot shows a window titled "AMBIBPSY COLLECTION LOGIN FORM". It contains the following fields and controls:

- System:** A dropdown menu with "Collection" selected.
- Category:** A dropdown menu with "Admin" selected.
- Username:** An empty text input field.
- Password:** An empty text input field.
- Start time:** An empty text input field.
- Buttons:** "Login", "Change Password", and "Logout" buttons are located at the bottom of the form.

### 1.2 Creation of new Operator

Component: NewOperator

Purpose: Confirm successful creation of new operator

Condition of testing: Create New\_Operator GUI

Test data: As specified in the Create New\_Operator GUI

Expected result: Successful creation and storage of new operator in the NewOperator table

Use View\_Operator menu item to confirm the result.



The screenshot shows a window titled "Create New\_Operator Form". It contains the following fields and controls:

- Operation:** A text input field containing "NEW\_OPERATOR".
- Enter operator details:** A section header above two input fields.
- OperatorId:** An empty text input field.
- Name:** An empty text input field.
- Buttons:** "Save" and "Cancel" buttons are located at the bottom of the form.

### 1.3 Modify operator

Component: ModifyOperator

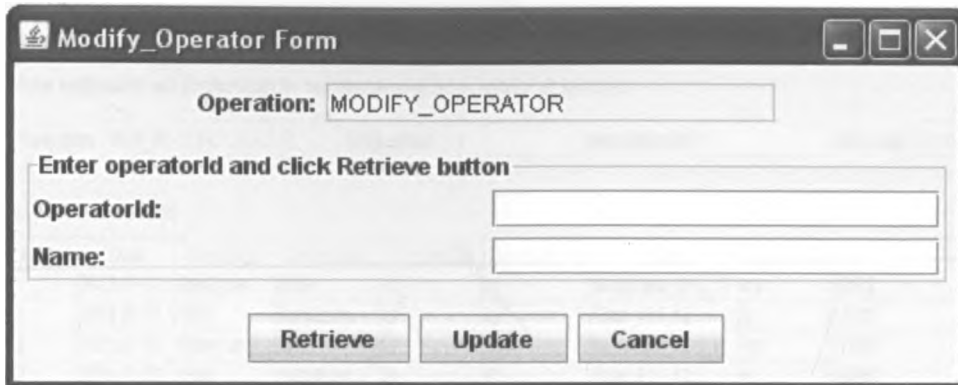
Purpose: To confirm the correct changes in the selected operator data

Condition of testing: Modify\_Operator GUI

Test data: As chosen from the Modify\_Operator GUI

Expected result: Successful modification of the selected operator data item in Operator table

Result confirmation: Retrieve the selected operator data to see the changes



The screenshot shows a window titled "Modify\_Operator Form". At the top, there is a text field labeled "Operation:" containing the text "MODIFY\_OPERATOR". Below this, there is a section titled "Enter operatorId and click Retrieve button". This section contains two input fields: "OperatorId:" and "Name:". At the bottom of the form, there are three buttons: "Retrieve", "Update", and "Cancel".

### 1.4 Delete operator

Component: DeleteOperator

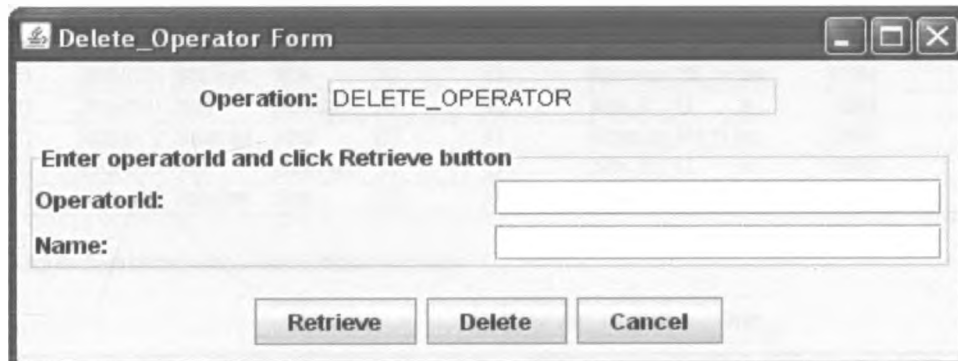
Purpose: To confirm successful deletion of the selected operator

Condition of testing: Delete\_Operator GUI

Test data: As selected from the Delete\_Operator GUI

Expected result: Successful removal of the selected operator data from the Operator table

Result confirmation: Retrieve data of the selected Operator to confirm the result



The screenshot shows a window titled "Delete\_Operator Form". At the top, there is a text field labeled "Operation:" containing the text "DELETE\_OPERATOR". Below this, there is a section titled "Enter operatorId and click Retrieve button". This section contains two input fields: "OperatorId:" and "Name:". At the bottom of the form, there are three buttons: "Retrieve", "Delete", and "Cancel".

### 1.5 View reconciledCDR

Component: CollectionSystemMenu

Purpose: To confirm correct reconciliation of CDR, display of a list of reconciledCDR on specified dates, and View\_ReconciledCDR GUI

Condition of testing: Main menu, View\_ReconciledCDR menu item

Test data: As specified in the View\_ReconciledCDR GUI

Expected result: Display of View\_ReconciledCDR GUI and a list of reconciledCDR on specified dates

**View\_ReconciledCDR Form**

Enter originatorId and destinationId for two operators or dates only for all operators

Operation: VIEW\_RECONCILEDCCR    OriginatorId: 12    DestinationId: 11    Start date: 2010-1-1    End date: 2010-1-31

**List of reconciledCDR**

| AutoNo | Date       | Originator | Destination | OriginatorId | DestinationId | RouteId          | RouteType | PaidMinutes |
|--------|------------|------------|-------------|--------------|---------------|------------------|-----------|-------------|
| 1      | 2010-01-01 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 16512       |
| 1      | 2010-01-01 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 13230       |
| 2      | 2010-01-02 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17568       |
| 2      | 2010-01-02 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14760       |
| 3      | 2010-01-03 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 16344       |
| 3      | 2010-01-03 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14310       |
| 4      | 2010-01-04 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14710       |
| 4      | 2010-01-04 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17424       |
| 5      | 2010-01-05 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17496       |
| 5      | 2010-01-05 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14240       |
| 6      | 2010-01-06 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17256       |
| 6      | 2010-01-06 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14320       |
| 7      | 2010-01-07 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17424       |
| 7      | 2010-01-07 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14560       |
| 8      | 2010-01-08 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 16956       |
| 8      | 2010-01-08 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14080       |
| 9      | 2010-01-09 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17304       |
| 9      | 2010-01-09 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14420       |
| 10     | 2010-01-10 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17628       |
| 10     | 2010-01-10 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14700       |
| 11     | 2010-01-11 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17304       |
| 11     | 2010-01-11 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 13880       |
| 12     | 2010-01-12 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 16944       |
| 12     | 2010-01-12 | Airtel     | Safaricom   | 11           | 12            | Airtel_011_12    | in        | 14730       |
| 13     | 2010-01-13 | Safaricom  | Airtel      | 12           | 11            | Safaricom_010_11 | out       | 17040       |

Click Refresh button then view menu to display information

## Appendix 2: Billing system test results

### 2.1 View generated bill

Component: BillingSystemMenu

Purpose: To confirm the correct display of View\_Generated\_Bill GUI, capture and display of a list of generated bills between two operators on specified dates

Condition of testing: Main menu, View\_Generated\_Bill menu item

Test data: As specified in the View\_Generated\_Bill GUI

Expected result: Correct display of the View\_Generated\_Bill GUI and a list of generated bills between two operators on specified dates

View\_Generated\_Bill Form

Enter originatorId and destinationId for two operators or dates only for all operators

Operation: VIEW\_GENERATED\_BILL    OriginatorId: 12    DestinationId: 11    Start date: 2010-1-1    End date: 2010-12-31

List of Generated bills

| Date       | BillNo | Month     | ContractNo | Originator | Destination | OriginatorId | DestinationId | PaidMinutes | RateId | RateValue | Amount     |
|------------|--------|-----------|------------|------------|-------------|--------------|---------------|-------------|--------|-----------|------------|
| 2010-01-31 | 1      | January   | 333222     | Safaricom  | Airtel      | 12           | 11            | -88848      | 10     | 2.10      | -186580.80 |
| 2010-02-28 | 2      | February  | 333222     | Safaricom  | Airtel      | 12           | 11            | -120623     | 10     | 2.10      | -253308.28 |
| 2010-03-31 | 3      | March     | 333222     | Safaricom  | Airtel      | 12           | 11            | 177084      | 10     | 2.10      | 371876.38  |
| 2010-04-30 | 4      | April     | 333222     | Safaricom  | Airtel      | 12           | 11            | 43872       | 10     | 2.10      | 92131.20   |
| 2010-05-31 | 5      | May       | 333222     | Safaricom  | Airtel      | 12           | 11            | 222804      | 10     | 2.10      | 467888.38  |
| 2010-06-30 | 6      | June      | 333222     | Safaricom  | Airtel      | 12           | 11            | -170850     | 10     | 2.10      | -358784.97 |
| 2010-07-31 | 7      | July      | 333222     | Safaricom  | Airtel      | 12           | 11            | 132047      | 10     | 2.10      | 277298.69  |
| 2010-08-31 | 8      | August    | 333222     | Safaricom  | Airtel      | 12           | 11            | -221481     | 10     | 2.10      | -465110.09 |
| 2010-09-30 | 9      | September | 333222     | Safaricom  | Airtel      | 12           | 11            | 170929      | 10     | 2.10      | 358950.88  |
| 2010-10-31 | 10     | October   | 333222     | Safaricom  | Airtel      | 12           | 11            | 134833      | 10     | 2.10      | 283149.28  |
| 2010-11-30 | 11     | November  | 333222     | Safaricom  | Airtel      | 12           | 11            | -42644      | 10     | 2.10      | -89552.40  |
| 2010-12-31 | 12     | December  | 333222     | Safaricom  | Airtel      | 12           | 11            | -90488      | 10     | 2.10      | -190024.80 |

Click Refresh button then view menu to display information

Refresh    Clear    Print

## 2.2 View received bill

Component: BillingSystemMenu

Purpose: To confirm the correct display of View\_Received\_Bill GUI, capture and display of a list of received bills between two operators on specified dates

Condition of testing: Main menu, View\_Received\_Bill menu item

Test data: As specified in the View\_Received\_Bill GUI

Expected result: Correct display of the View\_Received\_Bill GUI and a list of received bills between two operators on specified dates

View\_Received\_Bill Form

Enter originatorid and destinationid for two operators or dates only for all operators

Operation: VIEW\_RECEIVED\_BILL    OriginatorId: 11    DestinationId: 12    Start date: 2010-1-1    End date: 2010-12-31

List of Received bills

| Date       | BillNo | Month    | ContractNo | Originator | Destination | OriginatorId | DestinationId | PaidMinutes | RateId | RateValue | Amount    |
|------------|--------|----------|------------|------------|-------------|--------------|---------------|-------------|--------|-----------|-----------|
| 2010-01-31 | 1      | January  | 333222     | Airtel     | Safaricom   | 11           | 12            | 88848       | 10     | 2.10      | 186580.80 |
| 2010-02-28 | 2      | February | 333222     | Airtel     | Safaricom   | 11           | 12            | 120521      | 10     | 2.10      | 253094.09 |
| 2010-06-30 | 6      | June     | 333222     | Airtel     | Safaricom   | 11           | 12            | 170850      | 10     | 2.10      | 358784.97 |
| 2010-08-31 | 8      | August   | 333222     | Airtel     | Safaricom   | 11           | 12            | 221481      | 10     | 2.10      | 465110.09 |
| 2010-11-30 | 11     | November | 333222     | Airtel     | Safaricom   | 11           | 12            | 42542       | 10     | 2.10      | 89338.20  |
| 2010-12-31 | 12     | December | 333222     | Airtel     | Safaricom   | 11           | 12            | 90488       | 10     | 2.10      | 190024.80 |

Click Refresh button then view menu to display information

Refresh    Clear    Print

### 2.3 View reconciled bill

Component: BillingSystemMenu

Purpose: To confirm the correct display of View\_Reconciled\_Bill GUI, capture and display of a list of reconciled bills between two operators on specified dates

Condition of testing: Main menu, View\_Reconciled\_Bill menu item

Test data: As specified in the View\_Reconciled\_Bill GUI

Expected result: Correct display of the View\_Reconciled\_Bill GUI and a list of reconciled bills between two operators on specified dates

View\_Reconciled\_Bill Form

Enter originatorId and destinationId for two operators or dates only for all operators

Operation: VIEW\_RECONCILED\_BILL    OriginatorId: 12    DestinationId: 11    Start date: 2010-1-1    End date: 2010-12-31

List of Reconciled bills

| Date       | BillNo | Month     | ContractNo | Originator | Destination | OriginatorId | DestinationId | PaidMinutes | RateId | RateValue | Amount    |
|------------|--------|-----------|------------|------------|-------------|--------------|---------------|-------------|--------|-----------|-----------|
| 2010-03-31 | 3      | March     | 333222     | Safaricom  | Airtel      | 12           | 11            | 177084      | 10     | 2.10      | 371876.38 |
| 2010-04-30 | 4      | April     | 333222     | Safaricom  | Airtel      | 12           | 11            | 43872       | 10     | 2.10      | 92131.20  |
| 2010-05-31 | 5      | May       | 333222     | Safaricom  | Airtel      | 12           | 11            | 222804      | 10     | 2.10      | 467888.38 |
| 2010-07-31 | 7      | July      | 333222     | Safaricom  | Airtel      | 12           | 11            | 132047      | 10     | 2.10      | 277298.69 |
| 2010-09-30 | 9      | September | 333222     | Safaricom  | Airtel      | 12           | 11            | 170929      | 10     | 2.10      | 358950.88 |
| 2010-10-31 | 10     | October   | 333222     | Safaricom  | Airtel      | 12           | 11            | 134833      | 10     | 2.10      | 283149.28 |

Click Refresh button then view menu to display information

Refresh    Clear    Print



### Appendix 3: Payment system test results

#### 3.1 View generated voucher

Component: PaymentSystemMenu

Purpose: To confirm the correct display of View\_Generated\_Voucher GUI and a list of generated vouchers from the GeneratedVoucher table on specified dates

Condition of testing: Main menu, View\_Generated\_Voucher menu item

Test data: As specified in the View\_Generated\_Voucher GUI

Expected result: Correct display of the View\_Generated\_Voucher GUI and a list of generated vouchers on specified dates

View\_Generated\_Voucher Form

Enter originatorId and destinationId for two operators or dates only for all operators

Operation: VIEW\_GENERATED\_VOUCHER    OriginatorId: 12    DestinationId: 11    Start date: 2010-1-1    End date: 2010-12-31

List of Generated vouchers

| Date       | VoucherNo | BillNo | Month    | ContractNo | Originator | Destination | OrigId | DestId | OrigAccount | DestAccount | OrigBank | DestBank | Amount    |
|------------|-----------|--------|----------|------------|------------|-------------|--------|--------|-------------|-------------|----------|----------|-----------|
| 2010-01-31 | 1         | 1      | January  | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 186580.80 |
| 2010-02-28 | 2         | 2      | February | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 253094.09 |
| 2010-06-30 | 3         | 6      | June     | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 358784.97 |
| 2010-08-31 | 4         | 8      | August   | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 465110.09 |
| 2010-11-30 | 5         | 11     | November | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 89338.20  |
| 2010-12-31 | 6         | 12     | December | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 190024.80 |

Click Refresh button then view menu to display information

Refresh    Clear    Print

### 3.2 View received voucher

Component: PaymentSystemMenu

Purpose: To confirm the correct display of View\_Received\_Voucher GUI and a list of received vouchers from the distant PaymentAgent on specified dates

Condition of testing: Main menu, View\_Received\_Voucher menu item

Test data: As specified in the View\_Received\_Voucher GUI

Expected result: Correct display of the View\_Received\_Voucher GUI and a list of received vouchers on specified dates

View\_Received\_Voucher Form

Enter originatorId and destinationId for two operators or dates only for all operators

Operation: VIEW\_RECEIVED\_VOUCHER    OriginatorId: 11    DestinationId: 12    Start date: 2010-1-1    End date: 2010-12-31

List of Received vouchers

| Date       | VoucherNo | BillNo | Month     | ContractNo | Originator | Destination | OrigId | DestId | OrigAccount | DestAccount | OrigBank | DestBank | Amount    |
|------------|-----------|--------|-----------|------------|------------|-------------|--------|--------|-------------|-------------|----------|----------|-----------|
| 2010-03-31 | 1         | 3      | March     | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 371876.38 |
| 2010-04-30 | 2         | 4      | April     | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 92131.20  |
| 2010-05-31 | 3         | 5      | May       | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 467888.38 |
| 2010-07-31 | 4         | 7      | July      | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 277298.69 |
| 2010-09-30 | 5         | 9      | September | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 358950.88 |
| 2010-10-31 | 6         | 10     | October   | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 283149.28 |

Click Refresh button then view menu to display information

Refresh    Clear    Print

### 3.3 View reconciled voucher

Component: PaymentSystemMenu

Purpose: To confirm the correct display of View\_Reconciled\_Voucher GUI and a list of received vouchers from the distant PaymentAgent on specified dates

Condition of testing: Main menu, View\_Reconciled\_Voucher menu item

Test data: As specified in the View\_Reconciled\_Voucher GUI

Expected result: Correct display of the View\_Reconciled\_Voucher GUI and a list of received vouchers on specified dates

Enter originatorId and destinationId for two operators or dates only for all operators

Operation: VIEW\_RECONCILED\_VOUCHER    OriginatorId: 12    DestinationId: 11    Start date: 2010-1-1    End date: 2010-12-31

List of reconciled vouchers

| Date       | VoucherNo | BillNo | Month    | ContractNo | Originator | Destination | OrigId | DestId | OrigAccount | DestAccount | OrigBank | DestBank | Amount    |
|------------|-----------|--------|----------|------------|------------|-------------|--------|--------|-------------|-------------|----------|----------|-----------|
| 2010-01-31 | 1         | 1      | January  | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 186580.80 |
| 2010-02-28 | 2         | 2      | February | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 253094.09 |
| 2010-06-30 | 3         | 6      | June     | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 358784.97 |
| 2010-08-31 | 4         | 8      | August   | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 465110.09 |
| 2010-11-30 | 5         | 11     | November | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 89338.20  |
| 2010-12-31 | 6         | 12     | December | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 190024.80 |

Click Refresh button then view menu to display information

Refresh    Clear    Print

### 3.4 View payment instruction

Component: PaymentSystemMenu

Purpose: To confirm the correct display of View\_Payment\_Instruction GUI and a list of payment instructions to the local BankingAgent on specified dates

Condition of testing: Main menu, View\_Payment\_Instruction menu item

Test data: As specified in the View\_Payment\_Instruction GUI

Expected result: Correct display of the View\_Payment\_Instruction GUI and a list payment instructions on specified dates

View\_Payment\_Instruction Form

Enter originatorId and destinationId for two operators or dates only for all operators

Operation: VIEW\_PAYMENT\_INSTRUCTION    OriginatorId: 12    DestinationId: 11    Start date: 2010-1-1    End date: 2010-12-31

List of Payment instruction

| Date       | InstructionId | VoucherNo | BillNo | Month    | ContractNo | Originator | Destination | OrigId | DestId | OrigAccount | DestAccount | OrigBank | DestBank | Amount    |
|------------|---------------|-----------|--------|----------|------------|------------|-------------|--------|--------|-------------|-------------|----------|----------|-----------|
| 2010-01-31 | 1             | 1         | 1      | January  | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 186580.80 |
| 2010-02-28 | 2             | 2         | 2      | February | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 253094.09 |
| 2010-06-30 | 3             | 3         | 6      | June     | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 358784.97 |
| 2010-08-31 | 4             | 4         | 8      | August   | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 465110.09 |
| 2010-11-30 | 5             | 5         | 11     | November | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 89338.20  |
| 2010-12-31 | 6             | 6         | 12     | December | 333222     | Safaricom  | Airtel      | 12     | 11     | 111222333   | 666111888   | Barclays | Family   | 190024.80 |

Click Refresh button then view menu to display information

Refresh    Clear    Print

### 3.5 View payment information

Component: PaymentSystemMenu

Purpose: To confirm the correct display of View\_Payment\_Information GUI and a list of payment information from the local BankingAgent on specified dates

Condition of testing: Main menu, View\_Payment\_Information menu item

Test data: As specified in the View\_Payment\_Information GUI

Expected result: Correct display of the View\_Payment\_Information GUI and a list payment information on specified dates

**View\_Payment\_Information Form**

Enter originatorId and destinationId for two operators or dates only for all operators

Operation: VIEW\_PAYMENT\_INFORMATION    OriginatorId: 11    DestinationId: 12    Start date: 2010-1-1    End date: 2010-12-31

List of Payment information

| Date       | InformationId | VoucherNo | BillNo | Month     | ContractNo | Originator | Destination | OrigId | DestId | OrigAccount | DestAccount | OrigBank | DestBank | Amount    |
|------------|---------------|-----------|--------|-----------|------------|------------|-------------|--------|--------|-------------|-------------|----------|----------|-----------|
| 2010-03-31 | 1             | 1         | 3      | March     | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 371876.38 |
| 2010-04-30 | 2             | 2         | 4      | April     | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 92131.20  |
| 2010-05-31 | 3             | 3         | 5      | May       | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 467888.38 |
| 2010-07-31 | 4             | 4         | 7      | July      | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 277298.69 |
| 2010-09-30 | 5             | 5         | 9      | September | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 358950.88 |
| 2010-10-31 | 6             | 6         | 10     | October   | 333222     | Airtel     | Safaricom   | 11     | 12     | 666111888   | 111222333   | Family   | Barclays | 283149.28 |

Click Refresh button then view menu to display information

Refresh    Clear    Print

## Appendix 4: Banking system test results

### 4.1 View operations

Component: BankingSystemMenu

Purpose: To confirm the correct display of View\_Operation GUI and a list of operations on specified dates

Condition of testing: Main menu, View\_Operation menu item

Test data: As specified in the View\_Operation GUI

Expected result: Correct display of the View\_Operation GUI and a list of operations on specified dates

**View\_Operations Form**

Enter accountNo for one customer or dates only for all all customers

Operation:  AccountNo:  Start date:  End date:

**List of operations**

| Date                 | OperationId | Name             | AccountNo | Charge | Amount    | Balance        |
|----------------------|-------------|------------------|-----------|--------|-----------|----------------|
| 2010-01-31 00:00:... | 32          | InternalTransfer | 111222333 | 250.00 | 186580.80 | 8491431940.06  |
| 2010-01-31 00:00:... | 32          | InternalTransfer | 111222000 | 250.00 | 186580.80 | 69547878754.95 |
| 2010-02-28 00:00:... | 32          | InternalTransfer | 111222333 | 250.00 | 253094.09 | 8491178595.97  |
| 2010-02-28 00:00:... | 32          | InternalTransfer | 111222000 | 250.00 | 253094.09 | 69548132099.05 |
| 2010-03-31 00:00:... | 34          | ExternalTransfer | 111222888 | 400.00 | 371876.38 | 9486992316.59  |
| 2010-03-31 00:00:... | 34          | ExternalTransfer | 111222333 | 400.00 | 371876.38 | 8491550072.34  |
| 2010-03-31 00:00:... | 34          | ExternalTransfer | 111222000 | 400.00 | 0.00      | 69548132499.05 |
| 2010-03-31 00:00:... | 34          | ExternalTransfer | 111222000 | 400.00 | 0.00      | 69548132899.05 |
| 2010-04-30 00:00:... | 34          | ExternalTransfer | 111222888 | 400.00 | 92131.20  | 9486899785.40  |
| 2010-04-30 00:00:... | 34          | ExternalTransfer | 111222333 | 400.00 | 92131.20  | 8491641803.54  |
| 2010-04-30 00:00:... | 34          | ExternalTransfer | 111222000 | 400.00 | 0.00      | 69548133299.05 |
| 2010-04-30 00:00:... | 34          | ExternalTransfer | 111222000 | 400.00 | 0.00      | 69548133699.05 |
| 2010-05-31 00:00:... | 34          | ExternalTransfer | 111222888 | 400.00 | 467888.38 | 9486431497.02  |
| 2010-05-31 00:00:... | 34          | ExternalTransfer | 111222333 | 400.00 | 467888.38 | 8492109291.91  |
| 2010-05-31 00:00:... | 34          | ExternalTransfer | 111222000 | 400.00 | 0.00      | 69548134099.05 |
| 2010-05-31 00:00:... | 34          | ExternalTransfer | 111222000 | 400.00 | 0.00      | 69548134499.05 |
| 2010-06-30 00:00:... | 32          | InternalTransfer | 111222333 | 250.00 | 358784.97 | 8491750256.95  |
| 2010-06-30 00:00:... | 32          | InternalTransfer | 111222000 | 250.00 | 358784.97 | 69548493534.02 |
| 2010-07-31 00:00:... | 34          | ExternalTransfer | 111222888 | 400.00 | 277298.69 | 9486153798.34  |
| 2010-07-31 00:00:... | 34          | ExternalTransfer | 111222333 | 400.00 | 277298.69 | 8492027155.63  |
| 2010-07-31 00:00:... | 34          | ExternalTransfer | 111222000 | 400.00 | 0.00      | 69548493934.02 |
| 2010-07-31 00:00:... | 34          | ExternalTransfer | 111222000 | 400.00 | 0.00      | 69548494334.02 |
| 2010-08-31 00:00:... | 32          | InternalTransfer | 111222333 | 250.00 | 465110.09 | 8491561795.54  |
| 2010-08-31 00:00:... | 32          | InternalTransfer | 111222000 | 250.00 | 465110.09 | 69548959694.11 |
| 2010-09-30 00:00:... | 34          | ExternalTransfer | 111222888 | 400.00 | 358950.88 | 9485794447.46  |

Click Refresh button then View menu to add new operation to the list

#### 4.2 View generated transfer

Component: BankingSystemMenu

Purpose: To confirm the correct display of View\_Generated\_Transfer GUI and a list of generated transfers on specified dates

Condition of testing: Main menu, View\_Generated\_Transfer menu item

Test data: As specified in the View\_Generated\_Transfer GUI

Expected result: Correct display of the View\_Generated\_Transfer GUI and a list of generated transfers on specified dates

The screenshot shows a software window titled "View\_Generated\_Transfer Form". At the top, there is a text prompt: "Enter account1 and account2 for two customers or dates only for all customers". Below this, there are input fields for "Operation" (set to "VIEW\_GENERATED\_TRANSFER"), "Account1", "Account2", "Start date" (set to "2010-1-1"), and "End date" (set to "2010-12-31").

Below the input fields is a section titled "List of generated transfers" containing a table with the following data:

| Date       | TransferId | Account1  | Account2  | VoucherNo | ContractNo | Month    | Amount    |
|------------|------------|-----------|-----------|-----------|------------|----------|-----------|
| 2010-01-31 | 1          | 111222333 | 111222000 | 1         | 333222     | January  | 186580.80 |
| 2010-02-28 | 2          | 111222333 | 111222000 | 2         | 333222     | February | 253094.09 |
| 2010-06-30 | 3          | 111222333 | 111222000 | 3         | 333222     | June     | 358784.97 |
| 2010-08-31 | 4          | 111222333 | 111222000 | 4         | 333222     | August   | 465110.09 |
| 2010-11-30 | 5          | 111222333 | 111222000 | 5         | 333222     | November | 89338.20  |
| 2010-12-31 | 6          | 111222333 | 111222000 | 6         | 333222     | December | 190024.80 |

Below the table, there is a large empty rectangular area. At the bottom of the window, there is a text prompt: "Click Refresh button then view menu to display information". Below this prompt are three buttons: "Refresh", "Clear", and "Print".

### 4.3 View received transfer

Component: BankingSystemMenu

Purpose: To confirm the correct display of View\_Received\_Transfer GUI and a list of received transfers from the distant BankingAgent on specified dates

Condition of testing: Main menu, View\_Received\_Transfer menu item

Test data: As specified in the View\_Received\_Transfer GUI

Expected result: Correct display of the View\_Received\_Transfer GUI and a list of received transfers on specified dates

View\_Received\_Transfer Form

Enter account1 and account2 for two customers or dates only for all customers

Operation: VIEW\_RECEIVED\_TRANSFER Account1: Account2: Start date: 2010-1-1 End date: 2010-12-31

List of received transfers

| Date       | TransferId | Account1  | Account2  | VoucherNo | ContractNo | Month     | Amount    |
|------------|------------|-----------|-----------|-----------|------------|-----------|-----------|
| 2010-03-31 | 1          | 111222888 | 111222333 | 1         | 333222     | March     | 371876.38 |
| 2010-04-30 | 2          | 111222888 | 111222333 | 2         | 333222     | April     | 92131.20  |
| 2010-05-31 | 3          | 111222888 | 111222333 | 3         | 333222     | May       | 467888.38 |
| 2010-07-31 | 4          | 111222888 | 111222333 | 4         | 333222     | July      | 277298.69 |
| 2010-09-30 | 5          | 111222888 | 111222333 | 5         | 333222     | September | 358950.88 |
| 2010-10-31 | 6          | 111222888 | 111222333 | 6         | 333222     | October   | 283149.28 |

Click Refresh button then view menu to display information

Refresh Clear Print



#### 4.4 Statement

Component: Statement

Purpose: To confirm the correct generation of statement of the selected customer account on specified dates

Condition of testing: Main menu, Statement Form

Test data: As specified in the Statement Form

Expected result: Display of statement of the selected account on specified dates

Statement Form

Enter StartDate, EndDate and Account then click Retrieve button

Operation:  CurrentDate:  StartDate:

EndDate:  AccountNo:  Name:

**Bank Statement**

| Date                 | OperationId | Type             | Charge | Amount    | Balance       |
|----------------------|-------------|------------------|--------|-----------|---------------|
| 2010-01-31 00:00:... | 32          | InternalTransfer | 250.00 | 186580.80 | 8491431940.06 |
| 2010-02-28 00:00:... | 32          | InternalTransfer | 250.00 | 253094.09 | 8491178595.97 |
| 2010-03-31 00:00:... | 34          | ExternalTransfer | 400.00 | 371876.38 | 8491550072.34 |
| 2010-04-30 00:00:... | 34          | ExternalTransfer | 400.00 | 92131.20  | 8491641803.54 |
| 2010-05-31 00:00:... | 34          | ExternalTransfer | 400.00 | 467888.38 | 8492109291.91 |
| 2010-06-30 00:00:... | 32          | InternalTransfer | 250.00 | 358784.97 | 8491750256.95 |
| 2010-07-31 00:00:... | 34          | ExternalTransfer | 400.00 | 277298.69 | 8492027155.63 |
| 2010-08-31 00:00:... | 32          | InternalTransfer | 250.00 | 465110.09 | 8491561795.54 |
| 2010-09-30 00:00:... | 34          | ExternalTransfer | 400.00 | 358950.88 | 8491920346.41 |
| 2010-10-31 00:00:... | 34          | ExternalTransfer | 400.00 | 283149.28 | 8492203095.70 |
| 2010-11-30 00:00:... | 32          | InternalTransfer | 250.00 | 89338.20  | 8492113507.50 |
| 2010-12-31 00:00:... | 32          | InternalTransfer | 250.00 | 190024.80 | 8491923232.70 |

Click Retrieve button then operation menu to display statement

## Appendix 5: Code samples

### 5.1 Code to display Login GUI

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.table.*;
import javax.swing.border.*;
import java.sql.*;
import java.io.*;
import java.util.*;
import java.text.*;
import jade.core.*;
import jade.gui.*;
import teleOntology.*;
class CollectionAgentGui extends JFrame implements ActionListener, TeleBankVocabulary
{
//Object declaration
JPanel panel1, panel2,panel3;
JLabel label1,label2,label3,label4,label5;
final JTextField text1,text2, text3;
JComboBox combo1,combo2;
JButton btLogin, btChangepw, btLogout;
CollectionSystemMenu2 menu2 = new CollectionSystemMenu2();
CollectionSystemMenu menu = new CollectionSystemMenu ();
SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
java.util.Date cDate = new java.util.Date();
String date, sDate, eDate, system1, category1, name1, password1;
int identityNo;
Object category;
JPanel p1,p2;
private CollectionAgent collAgent;
Connection con;
```

```

Statement st;
ResultSet rs;
public CollectionAgentGui(CollectionAgent ag)
{
    collAgent = ag;
    menu = new CollectionSystemMenu ();
    menu2 = new CollectionSystemMenu2 ();
    setTitle(" AMBIBPSY COLLECTION LOGIN FORM");
    setSize(400,250);
    label1 = new JLabel();
    label1.setText("System:");
    combo1 = new JComboBox(new String[] {"Collection", "Billing", "Payment", "Banking"});
    label2 = new JLabel();
    label2.setText("Category:");
    combo2 = new JComboBox(new String[] {"Admin", "User"});
    label3 = new JLabel();
    label3.setText("Username:");
    text1 = new JTextField(30);
    text1.requestFocus(true);
    label4 = new JLabel();
    label4.setText("Password:");
    text2 = new JPasswordField(30);
    label5 = new JLabel();
    label5.setText("Start time:");
    text3 = new JTextField(30);
    text3.setEditable(false);
    btLogin = new JButton("Login");
    btChangepw = new JButton("Change Password");
    btLogout = new JButton("Logout");
    panel1=new JPanel(new GridLayout(5,2,5,5));
    panel1.add(label1);
    panel1.add(combo1);
    panel1.add(label2);
    panel1.add(combo2);
    panel1.add(label3);
    panel1.add(text1);
    panel1.add(label4);

```

```

panel1.add(text2);
panel1.add(label5);
panel1.add(text3);
panel2=new JPanel(new FlowLayout());
panel2.add(btLogin);
panel2.add(btChangepw);
panel2.add(btLogout);
panel3 = new JPanel(new BorderLayout(5,5));
panel3.setSize(350,150);
panel3.add(panel1,BorderLayout.CENTER);
panel3.add(panel2,BorderLayout.SOUTH);
setLayout(new BorderLayout(10,10));
add(panel3,BorderLayout.CENTER);
btLogin.addActionListener(this);
btChangepw.addActionListener(this);
btLogout.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
Object system = combo1.getSelectedItem();
category = combo2.getSelectedItem();
String name = text1.getText();
String password = text2.getText();
if(ae.getSource()==btLogin)
{
//Connection con;
try
{
//Loads JDBC Driver
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
// Create connection
con = DriverManager.getConnection("jdbc:odbc:CollectionDB");
// Create Statement
st = con.createStatement();
java.util.Date cDate = new java.util.Date();
if(category.equals("Admin"))
{

```

```

rs = st.executeQuery(" SELECT * FROM CollectionUser WHERE  System = '"+system+"' and
Category = '"+category+"' and Name = '"+name+"' and Password = '"+password+"'");
while(rs.next())
{
date = rs.getString(1);
system1 = rs.getString(2);
category1 = rs.getString(3);
name1 = rs.getString(4);
identityNo = rs.getInt(5);
password1 = rs.getString(6);
sDate = df.format(cDate.getTime());
text3.setText(sDate);
    menu.setVisible(true);
    menu.setTitle(category + " " + name + " Welcome to AMBIBPSY " + system + " System : Select
an activity from the menu");
}
}
else if(category.equals("User"))
{
rs = st.executeQuery(" SELECT * FROM CollectionUser WHERE  System = '"+system+"' and
Category = '"+category+"' and Name = '"+name+"' and Password = '"+password+"'");
while(rs.next())
{
date = rs.getString(1);
system1 = rs.getString(2);
category1 = rs.getString(3);
name1 = rs.getString(4);
identityNo = rs.getInt(5);
password1 = rs.getString(6);
sDate = df.format(cDate.getTime());
text3.setText(sDate);
    menu2.setVisible(true);
    menu2.setTitle(category + " " + name + " Welcome to AMBIBPSY " + system + " System :
Select an activity from the menu");
}
}
con.close();

```

```

}
catch(Exception e)
{
    e.printStackTrace();
}
}
if(ae.getSource()==btChangepw)
{
    CollectionPasswordChange pwChange= new CollectionPasswordChange();
    pwChange.setVisible(true);
    pwChange.setSize(400,300);
}
if(ae.getSource()==btLogout)
{
    try
    {
        //Loads JDBC Driver
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        // Create connection
        con = DriverManager.getConnection("jdbc:odbc:CollectionDB");
        // Create Staement
        st = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
        java.util.Date cDate = new java.util.Date();
        sDate = text3.getText();
        eDate = df.format(cDate.getTime());
        name = text1.getText();
        password = text2.getText();
        ResultSet rs = st.executeQuery(" SELECT * FROM CollectionUser WHERE Name = '"+name+"'
        AND Password = '"+password+"'");
        while(rs.next())
        {
            date = rs.getString(1);
            system1 = rs.getString(2);
            category1 = rs.getString(3);
            name1 = rs.getString(4);
            identityNo = rs.getInt(5);

```

```

password1 = rs.getString(6);
    }
int rs1 = st.executeUpdate("INSERT INTO CollectionLogin values('"+sDate+"', '"+eDate+"', 0,
"+system1+"', '"+category1+"', '"+name1+"', '"+identityNo+"', '"+password1+"')");
    rs1 = st.executeUpdate("UPDATE CollectionLogin SET Duration = DATEDIFF(ss, StartDate,
EndDate) WHERE StartDate = '"+sDate+"' AND EndDate = '"+eDate+"' AND IdentityNo =
"+identityNo+" ");
    text1.setText("");
    text2.setText("");
    text3.setText("");
    menu.setVisible(false);
    menu2.setVisible(false);
    con.close();
}
catch(Exception e)
{
    e.printStackTrace();
}
}
}
}
}

```

## Appendix 6: Installation manual

AMBIBPSY system is developed using java programming language in JADE 4.0 environment. The databases are developed using Microsoft SQL server 2005 database management system. In order to install AMBIBPSY, the following steps are followed:

1. Install Microsoft SQL server 2005 database management system if not existing in the computer.
2. Set the name of the computer to Safaricom
3. Install JDK 6.0 and above if not existing in the computer.
4. Create CollectionDB, BillingDB, PaymentDB and BankingDB.
5. For each of the above database, configure the SQL driver as follows:
  - a) From the task bar, click start then control panel
  - b) In the pick a category list, click performance and maintenance, then administrative tools
  - c) In the administrative tools, double click data source (ODBC)
  - d) From ODBC data source administrator tab, click System DSN then double click Add button.
  - e) From create new data source, double click SQL server
  - f) From create a new data source to SQL server: Write the name of the database in the name text field; give a description of the database in the description text field; write the name of the computer in the server text field then click next, then click next, then click finish, then click test data source, then click OK, then click OK.
6. Insert AMBIBPSY CD in the CD drive
7. Copy jade and Safaricom folders from AMBIBPSY CD and paste them in the C drive.
8. From the command prompt, change the directory path as follows: CD\Safaricom\bin then press return key.
9. From the command prompt, set the classpath to the jade environment as follows: set classpath=%classpath%;.;C:\jade\lib\jade.jar;C:\jade\lib\commons-codec\commons-codec-1.3.0.jar
10. Use the following command to create the required tables in the databases: java Tables then press return key.
11. Fire the agent by using the following command: java jade.Boot -host Safaricom sfCollector:CollectionAgent then press return key.
12. Installation for Airtel computer is done by replacing where Safaricom is by Airtel and firing the agent using: java jade.Boot -host Airtel atCollector:CollectionAgent then press return key.
13. You can login using admin as default for name and password.