A MULTI- AGENT BASED HOSPITAL INSURANCE SYSTEM (MAHIS)

BY

SALIM IBRAHIM MURGANI

P58/70492/2008

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF AWARD OF THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE OF THE UNIVERSITY OF NAIROBI

2016

DECLARATION

This project presented in this report is my original work and has been done with full support of my supervisor as part of the fulfilment for Master of Science in Computer Science.

Signed: ………………………….

Date: ……………………………..

Salim Ibrahim Murgani

P58/70492/2008

This project has been submitted as partial fulfilment of requirements for the Masters of Science in Computer Science with my approval as a lecturer in the University of Nairobi, School of Computing and Informatics and as the Project Supervisor.

Signed: …………………………

Date:.……………………………

Mr Eric Ayienga

DEDICATION

This project is dedicated to my family for the support and guidance they gave me and the sacrifice they had to make for me to realize my dream and reach this far.

ACKNOWLEDGEMENT

Firstly I wish to thank God the Almighty for having given me strength, good health and patience all this time.

Secondly I wish to thank my parents and my family for their support and encouragement all this years.

My sincere thanks to my supervisor Mr. Eric Ayienga for his guidance and support through my project. Thanks also to the members of the project defence panel for their valuable time, effort and feedback in the examination and evaluation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

GRMAS- Genatric residence multi-Agent system

MAHIS- Multi-Agent Hospital Insurance System

MAS- Multiagent systems

FIPA- Foundation for Intelligent Physical Agents

JADE- Java Agent Development Framework

DF- Directory Facilitator

AMS- Agent Management System

MTS- Message transport service

PASSI- Process for Agent societies specification implementation

UML- Unified Modelling Language

ACC- Agent Communication channel

Jdbc- Java database connector

ABSTRACT

The process of handling claims by insurance firms is a very expensive process that requires the attention of different experts for different stages of the claim process. With the different process comes the problem of having huge amounts of information in unstructured manner. The domain also requires managing enormous amounts of information, which is many cases is heterogenous, distributed and unstructured, due to the different panties involves and different in their internal systems. This combination requires autonomous, intelligent and adaptive technologies to implement, therefore, the need of agent based technology is highly required.

The multiagent based hospital insurance system is efficient in handling registration and claims. The system is able to handle large amounts of information in a structured manner. It also helps in saving the amount of time that a client takes to submit and have the claim processed. The system is able to register promptly new members and automatically allocates them new identification numbers to use. The members are also able to view their statements and registration details.

CHAPTER 1: INTRODUCTION

1.1 Background

Insurance is method of spreading the risk, rather than lose everything, for a premium or payment; the insurance company protects each individual from a large financial loss. Therefore, hospital insurance is a body that covers its members for their inpatient medical needs. Workers earning more than a defined amount must contribute to the fund through payroll deductions. Membership is voluntary for self-employed workers. The fund then pays hospital benefits to members and their declared dependants out of the contributions received.

The insurance market relies heavily on the traditional way of handling claims. Every aspect of a claim will often be dealt with by different expert working at different department of the company. The input, processing and distribution of data are treated by each pant of the organization in their own traditional way, making the process very costly.

More than ever, the insurance market is currently searching for ways of making the process of handling claims more economical. Because the process of handling claims involves many different parties such as the patients, the insurance company and the healthcare centres, there is a growing need for agent based systems that can handle all the claims from the parties involved.

Research on applications utilizing agent technology is on the use. The primary focus for this has been in applying agent technology to application domains for which it would have a major impact. Agents are small, autonomous or semi-autonomous software programs that perform a set of specialized functions to meet a specific set of goals and then provide results to a customer or end user in a format readily acceptable by that customer or end user (Minh, 2008) et al.

The study of multi-agent systems is concerned with the development and analysis of sophisticated artificial intelligence, problem solving and control architectures for both single agent and multiple agent systems. Agent based systems technology has generated lots of excitement in recent years because of its promise as a new paradigm for conceptualizing, designing, implementing software systems. The promise is particularly attractive for creating software that operates in environments that are distributed and open, such as the internet. (Cabestay, 2009) et al observes that the agents in a multi-agent system. Must have some individual features that facilitate their integration.

1.2 Problem Statement

The insurance sector is quite dynamic with changing guidelines. The domain also required managing enormous amounts of information, which in many cases is heterogeneous distributed and unstructured, due to the different parties involved and different in their internal systems. This combination requires autonomous, intelligent and adaptive technologies to implement; therefore, the need of agent based technology is highly required.

1.3 Problem Objectives

This project aims to demonstrate the applicability of agent technology in Hospital insurance systems. This aim is to build and deploy a prototype multi-agent based hospital insurance system (MAHIS) that could be applied to provide services to the clients that are the public users and the healthcare centres.

1.3.1 Research Objectives

Research on agent applications and utilization of agent technologies in hospital insurance.

1.3.2 System Development Objectives

1. Develop a functional specification for the proposed agent-based system.
2. Design and implement an agent based Hospital Insurance system.
3. Test and evaluate the prototype.

1.4 Significance of the Project

The multi-agent agent based hospital insurance system is able to assist organizations that provide insurance cover to handle claims efficiently and save costs on hiring of experts that are more professional to handle claims. The agents developed must possess this expertise and it is desirable that they have learning ability. With learning ability they will posses

knowledge of handling different claims, comprehension in the understanding of facts, application in the use of new knowledge to solve problems. This will ultimately lead to cost savings.
The project will be important to the academicians, scholars and researchers as it will provide in depth information in terms of literature review and form a reference point in examining different aspects of multi-agent systems in the field of hospital insurance and other related fields.

The system itself will have agents responsible for handling claims and applications from both the health centres and the public or members insured; in the process reduce favoursim in handling cases whereby certain members are highly favoured due to status

1.5 Chapter Outline

This document is organized as follows, chapter 2 discusses the literature reviews, chapter 3 discusses the proposed methodology. Chapter 4 is the analysis and design, chapter 5 is the implementation and evaluation and chapter 6 is the conclusion

1.6 Scope of the research

The system is only for health insurance dealing with hospitalisation for the members. However this can be extended by further research and include other products like, car insurance and other insurance that exist.

The scope of the research includes the agent features, the design and implementation of the Agent-based Hospital Insurance System. The system will only be used by the public users and the Hospitals.

It is also assumed that the number of agents will not change during the time the system will be running.

CHAPTER 2: LITERATURE REVIEW

2.1 Hospital or health Insurance

Health insurance schemes are mechanisms wherein people prepay for some component of health care and there is some pooling of revenues and risks such that the healthy cross-subsidies health-care for the sick. Policy makers generally see health insurance as means of improving access to effective health care, particularly among the poor, preventing indebtedness because of trying to access such care [WHO, 2000].

In Kenya like any other developing country, healthcare is of high importance to its citizens. There are many hospital insurers but there is also the government owned National Hospital Insurance Fund that handles claims for both the formal and informal sectors. For those in the formal sector, it is compulsory to be a member.

For those in the informal sector and retirees, membership is open and voluntary. Majority of the informal sector employees do not subscribe to insurance cover due to lack of information, unaffordable premiums and priorities that are more important or immediate than health and medical insurance. According to [Bennet, 1998] et al, lack of participation in insurance schemes was due to unaffordable premiums.

2.1.1 Benefits of Hospital Insurance

Hospital insurance has many advantages to individuals and families. Benefits include paying for expensive medical procedures that can occur when there is an emergency. Individuals typically have various benefits that are provided by a group hospital insurance plan or an individual health insurance policy.

In the group health insurance plan, Individuals typically receive health insurance offered by their employer. This type of health care insurance has many benefits that include all physical examinations and no exclusions for pre-existing health conditions. Insurance companies that provide group health insurance are prohibited by law from denying coverage based on an individual's current health or health history. The cost of this type of insurance is also cheaper than other type of health insurance.

Individual health insurance plans are typically more expensive than other types of insurance plans but do have certain advantages, for individuals who obtain this type of insurance can choose the deductible and co-payments for their policy, the primary health care provider or doctor can also be selected when an individual healthcare policy is used.

General benefits for both individual and group health insurance are;

1. Upon admission in hospital, the member is accorded services and the hospital makes a claim to the fund for reimbursement.
2. An in-patient covers for the contributor, declared spouse and children.
3. Provides comprehensive medical cover in majority of accredited hospitals.
4. Provide in-patient services in private and high cost hospitals on a co-payment basis.
5. Comprehensive maternity and CS (Caesarean) package in accredited hospitals.
6. Family planning- vasectomy and Tubal Ligation
7. Most of the insurers cover majority of the diseases

2.1.2 Contribution and Deductions

Formal sector employee's contribution are deducted and remitted to the insurance fund by their employers. This is done by cheque or through E-banking. Contributions are made as per their income. For members under voluntary category, they pay a specified amount agreed.

2.1.3 Payment of Claims

Claims are submitted by hospitals directly to the insurer after the contributors have been discharged from hospitals. The claims are examined to ensure validity before payment.

A claim can however be rejected and the hospital informed accordingly to incorporate either the missing documents or to address the abnormalities identified.

2.2 Agents concept

An agent as a small, autonomous or semi-autonomous software programs that performs a set of specialized functions to meet a specific set of goals and then provide results to a customer, in a format readily acceptable by that customer. Agents can operate on their own without human guidance. They have control over their own actions and internal states. An agent is a software entity that applies artificial intelligent techniques to choose the best set of actions to perform in order to reach a goal specified by the user [Minh, 2008] et al.

Agent technology has been incorporated in computing for some time now. The motivation has been majorly due to the fact that there is need to have task's delegated so that many tasks can be actually be accomplished by autonomous software entities working in a coordinated and cooperative manner. Agent based systems technology has become a new paradigm for conceptualization, designing and implementing software systems [Zoltan, 2008].

In a multi-agent environment, agents must have some individual features that facilitate their integration into a larger system without reducing their ability to work as stand-alone devices and to satisfy the user [Cabestany, 2009] et al. These features include anatomy, collaborative behaviour, adaptability and reactivity.

2.2.1 Agent properties

It is normally assumed that agents have the following properties;

1. Reactivity- they can react timely and flexibly to the dynamic and unexpected changes in their environment.
2. Autonomy- they have an autonomous and independent behaviour, which is not controlled by any external entity.
3. Proactively- they can take the initiative and perform proactively actions that may help them to reach their goals.

4. Communication- they can communicate with users or other agents. Thus, they can exchange information, engage in complex negotiations, and coordinate their activities to cooperate in the joint resolution of a problem.

Agents usually have reasoning, planning and learning capabilities that allow them to display an intelligent behavioural.

2.2.2 Agent Classification

Agents are classified according to:

1. Collaborative- group of agents that corporate in the joint solution of a problem.
2. Interface- Collaborations with user to solve a task.
3. Internet- Manage the search and manipulation of information through the internet.
4. Mobile- Physical movement through different machines
5. Hybrid- combination of some of the previous types

2.3 Multi-agent Concept

[Cabestany, 2009] et al observes that the agents in a multi-agent system must have some individual features that facilitate their integration into a feature that facilitates their integration into a larger system, without reducing their ability to work as stand-alone devices and to satisfy the user. These features include autonomous, collaborative behaviour, adaptability and reactivity.

Multi-agent system can be defined as a system with varying number of interacting, autonomous agents that communicate with each other using flexible and complex protocols, in order to achieve particular goals or perform some set of tasks [Ibarra, 2008]. Multi-agent as a combination of agent's cooperation, coordination, cooperative problem solving, coalition formation and negotiation. A Multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. MAS are used to solve problems that are difficult or impossible for an individual agent to solve. MAS systems are also referred to as 'self organized systems', tend to find the best solution for their problems 'without intervention'. There is high similarity here to physical phenomena, such as energy minimizing, where physical objects tend to reach the lowest energy possible, within the physical constrained world [Adel and Mohamed, 2006]. For instance many cars entering Nairobi central district in the morning, will be available for leaving that same central district in the evening.

The main feature, which is achieved when developing multi-agent systems, if they work, is flexibility, since a multi-agent system can be added to, modified and reconstructed, without the need for detailed rewriting of the application. The study of multi-agent systems is concerned with the development and analysis of sophisticated AI problem solving and control architectures for both single-agent and multiple agent systems.

2.3.1. Advantages of Multi-Agent Approach

MAS have the following advantages over a single agent or centralized approach:

1. MAS distribute computational resources and capabilities across a network of interconnected agents. Whereas a centralized system may be plagued by resource limitations, performance bottlenecks, or critical failures, MAS is decentralized and thus does not suffer from “single point of failure” problem associated with centralized systems.
2. MAS allows for interconnection and interoperation of multiple existing legacy systems. By building an agent wrapper around such systems, they can be incorporated into an agent society.
3. MAS models problems in terms of autonomous interacting component-agents, which is providing to be a more natural way of representing task allocation, team planning, user preferences, open environments.
4. MAS efficiently retrieve, filters, and globally coordinate information from sources that are spatially distributed.
5. MAS provide solutions in situations where expertise is spatially and temporary distributed.
6. MAS enhance overall system performance, specifically along the dimensions of computational efficiency, reliability, maintainability, responsiveness, flexibility and reuse.

2.3.2 Motivation for Using MAS

MAS offer an appropriate tool to tackle problems in Health care especially in Hospital insurance. Some of the reasons to support this claim are;

1. The components of a multi-agent system may be running in different machines located in many different places. Each of the agents may keep part of the knowledge required to solve the problem, therefore multi-agent systems offer a natural way of attacking distributed problems.
2. One of the main properties of an intelligent agent is sociability. Agents are able to communicate between themselves, using some kind of agent communication language, in order to exchange any kind of information. In that way they can engage in complex dialogues, in which they negotiate, coordinate their actions and collaborate in the solution of a problem.
3. When a problem is too complex to be solved in a single system, it is usual to decompose it in sub problems. In multi-agent systems there are techniques of distributed problem solving, in which a group of agents may dynamically discuss how to partition a problem, how to distribute the different subtasks to be solved among them, how to exchange information to solve possible dependencies between partial solutions, and how to combine the partial results into the solution of the original problem.
4. Another important property of agents is their proactively. Their ability of performing tasks that may be beneficial for the user, even if the user has not explicitly demanded those tasks to be executed. Using this property they may find relevant information and show it to the user before the user requests it

2.3.3 Applications in the real life

MAS are systems are applied in the real world to graphical applications such as computer games, agents have also been used in films, coordinated defence systems. Other applications include transportation, logistics, graphics, GIS as well as in many other fields. It is widely advocated for use in networking and mobile technologies, to achieve automatic and dynamic load balancing, high scalability and self-healing networks.

2.4. The frameworks

While ad hoc multi-agent systems are often created from scratch by researchers and developers, some frameworks have arisen that implement common standards such as the FIPA (Foundation for Intelligent Physical Agents) agent system platforms and communication languages. These frameworks save developers time and aid in the standardization of MAS development.

2.4.1 FIPA (Foundation for Intelligent Physical Agents)

The foundation for intelligent physical Agents (FIPA) is an international non-profit organisation sharing the effort to produce specifications of generic agent technologies. FIPA is envisaged not just as a technology for one application but as generic technologies for different application areas, and not just as independent technologies but as s set of basic technologies that can be integrated by developers to make complex systems with a high degree of interoperability.

FIPA is based on two main assumptions. The first is that the time to reach consensus and to complete the standard should not be long, and, mainly, it should not act as a brake on progress rather than an enabler, before industries make commitments. The second is that only the external behaviour of system components should be specified, leaving implementation details and internal architectures to agent developers. In fact, the internal architecture of JADE is a proprietary even if it complies with the interfaces specified by FIPA. According to the FIPA97 specifications, specify the normative rules that allow a society of agents to inter-operate, that is effectively exist, operate and be managed. First of all they describe the reference model of an agent platform as shown in the figure 1.

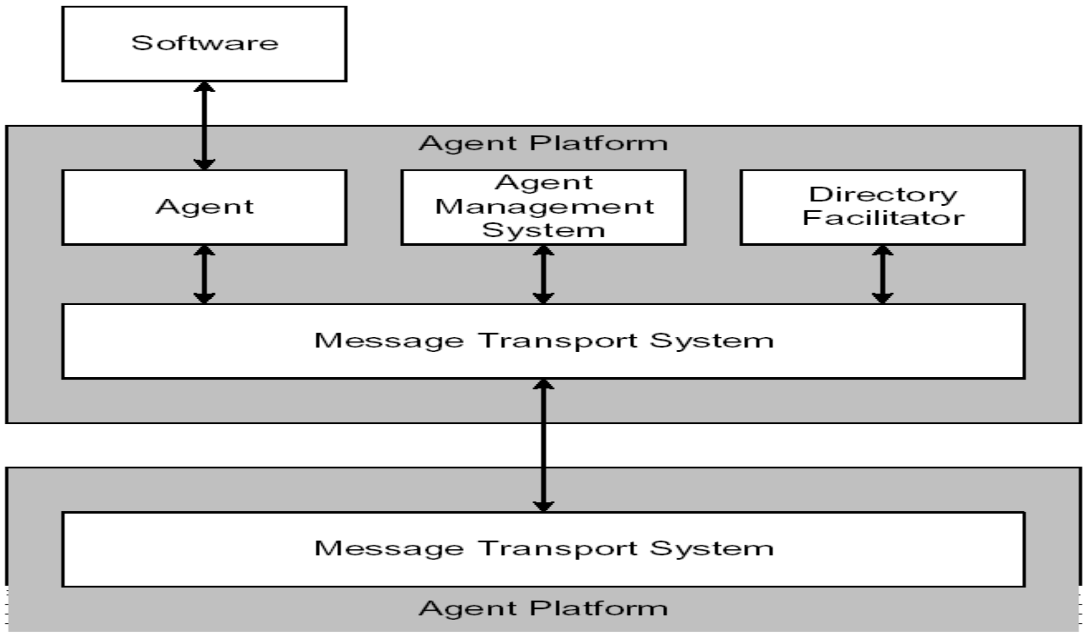


Figure 1- FIPA Architecture of MAS [Bellifemine, 2003]

Platform Elements 1 consists of the Agent- an application that provides a set of services, for instance when claims are made, the agent will be able to offer services required by the user and the directory facilitator (DF) - DF is an agent that offers a yellow pages service inside a system. DF agent knows the services that can be offered by the other agents of MAS.

Platform Elements 2 consists of the Agent Management System (AMS) - AMS is an agent that controls the access and the use of the agent's platform. AMS knows the addresses of the platform agents and offers a white page service and Message Transport Service (MTS) - MTS is used to communicate to agents in different platforms, although most research projects have only one platform.

There are two types of communication used. They are;

1. Blackboard systems- The agents communicate information to each other using a common data structure in the form of a blackboard which all of them can access.
2. Message Communication- The agents communicate with each other directly via messages by either using;- Common communication language (e.g. FIPA-ACL) and Communication protocol, Message format (e.g. FIPA protocols)

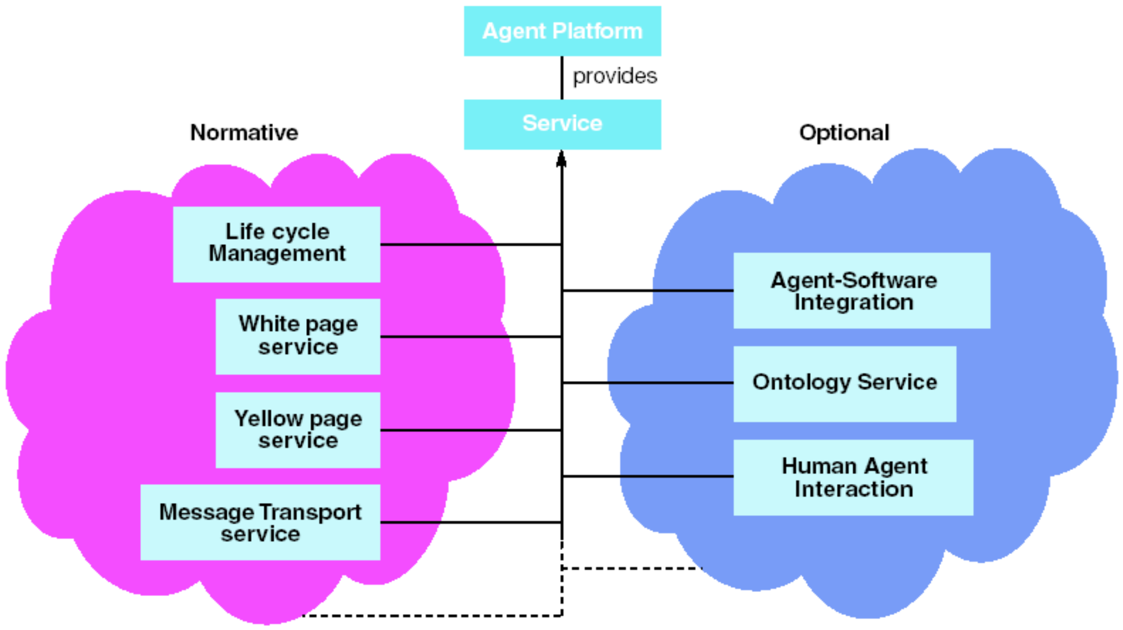


Figure 2: FIPA standard. Services provided by a platform (F.Bellifemine, 2002)

2.4.2 JADE

JADE is the middleware developed by TILAB for the development of distributed multi-agent applications based on the peer to peer communication architecture. The intelligence, the initiative, the information, the resources and the control can be fully distributed on mobile terminals as well as on computers in the fixed networks. The environment can evolve dynamically with peers that in JADE are called agents that appear and disappear in the system according to the needs and the requirements of the applications environment. Communication between the peers, regardless of whether they are running in the wireless or wire line network, is completely symmetric with each peer being able to play both the initiator and the responder role.

JADE is fully developed in Java and is based of the following driving principles;

1. Interoperability- JADE is compliant with the FIPA specifications. As a consequence, JADE agents can interoperate with other agents, provided that they comply with the same standard.
2. Uniformity and portability- JADE provides a homogenous set of APIs that are independent from the underlying network and Java versions. More in details, the

JADE runtime provides the same APIs both for the J2EE, J2SE and J2ME environment. In theory, application developers could decide the Java run time environment at deploy-time.

3. Easy to use- The complexity of the middleware is hidden behind a simple and intuitive set of APIs.
4. Pay-as-you-go philosophy-Programmers do not need to use all the features provided by the middleware. Features that are not used do not require programmers to know anything about them, neither add any computational overhead.

CHAPTER 3: METHODOLOGY

3.1 Introduction

A methodology is a body of methods employed by a discipline. A method is procedure for attaining something. Most developers of agent-based systems use an ad hoc approach- minimising guidelines and assembling some set of notation and terminology. This approach has maximum flexibility but the quality of the resulting application is questionable, since any knowledge and experience gained cannot be easily transferred to other projects [Odell, 2005].

MAS differ from non-agent based systems because agents are intended to be autonomous units of intelligent functionality. As a consequence, agent based software engineering methods must complement standard design activities and representations with models of the agent society. There exist a number of design methodologies. Some of the methodologies are; The TROPOS methodology which places an emphasis on modelling goals and their relationship with the systems actors, tasks, and resources; The MAS CommonKADS which enables the developer to build agent-based systems while applying the experiences of pre-agent methodologies and employing familiar techniques and diagrams.

3.2 The Proposed Methodology

The PASSI methodology is another example of agent development methodologies. This is the proposed methodology for a multi-agent based hospital insurance system (MAHIS). PASSI methodology brings a particularly rich development lifecycle that spans initial requirements though deployment and, in addition, emphasises the social model of agent-based systems.

3.3 Overall view of the PASSI Methodology

A definition of what an agent is can help explain PASSI as a methodology. An agent can be viewed as the instance of an agent class that is software implementation of an autonomous entity capable of pursuing an objective through its autonomous decisions, actions, and social relationships. An agent may occupy several functional roles to achieve its goals. A role is a function temporarily assumed by the agent in the society while pursuing a sub goal. During this activity the agent uses one or more of its tasks. A task is a series of elementary pieces of behaviour or actions necessary for the specific purpose. Each task carries out one of the agent's decisions, actions and social relationships. In this definition we can find the concepts of agent's role, task and action.

3.4 The PASSI methodology

Process for Agent Societies Specification Implementation (PASSI) is a step-by-step requirement to code methodology for: Designing and developing multi-agent societies integrating design models and concepts from both object oriented (OO) software engineering and MAS, using the unified modelling language (UML) notation.

PASSI aims at using standards whenever it is possible. It considers two different aspects of agents: during the initial step of design, they are seen as autonomous entities capable of pursuing an objective through autonomous decisions, actions and relationships; then they are considered as part of the system. According to [Cabri, 2005] et al, the PASSI methodology is composed of five models addressing different design levels of abstraction; system requirement model, agent society model, agent implementation model, code model and deployment model. The diagram below shows the phases of PASSI methodology:

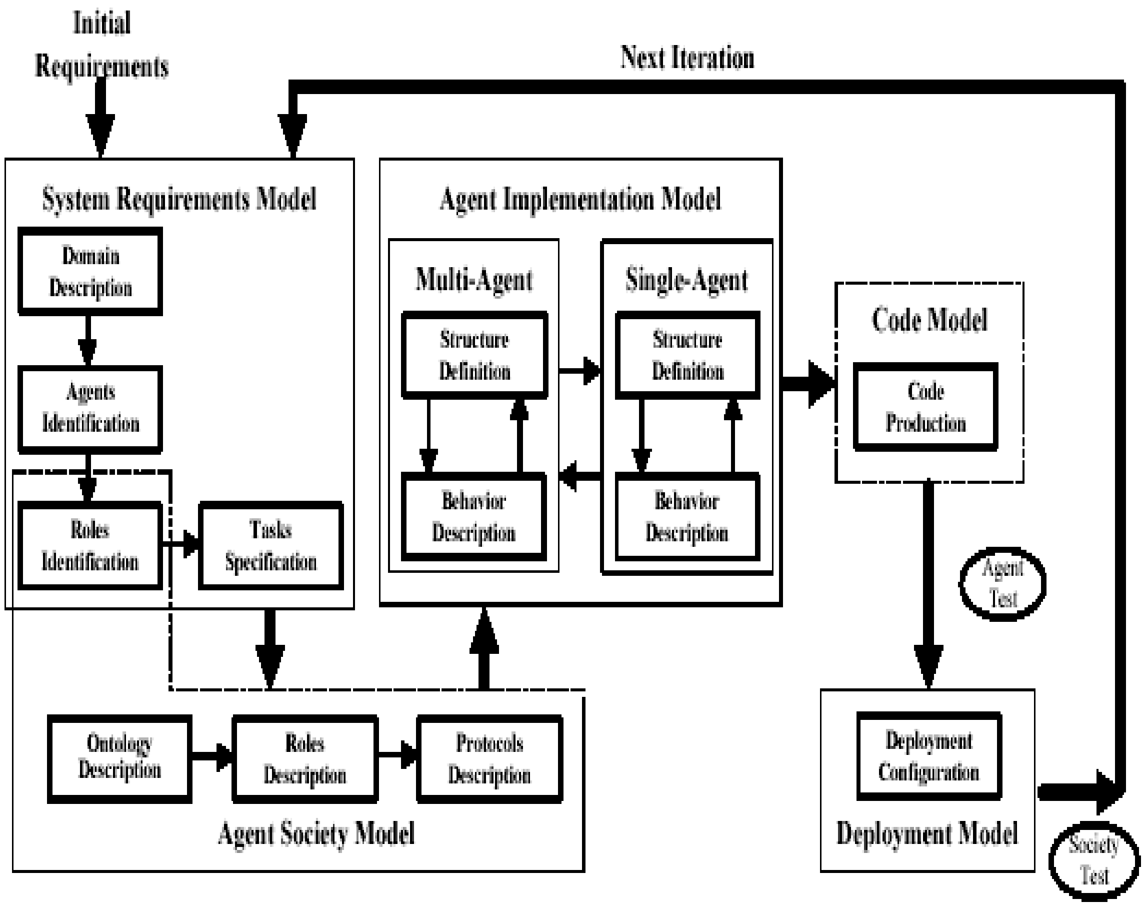


Figure 3. Phases of PASSI methodology [Cossentino, 2005]

3.4.1 Systems Requirement Model

A model of the systems requirements in terms of agency and purpose. It is composed of the following phases namely the; domain description, agent identification, role identification and task specification.

3.4.2 Agent Society Model

A model of social interactions and dependencies among the agents involved in the solution. Developing this model involves three steps;

1. Role identification
2. Ontology description- description of the knowledge ascribed to individual agents and the pragmatics of their interaction.

3. Role description- Used to show the roles played by agents, the tasks involved, communication capabilities and inter agent dependencies.

3.4.3 Agent Implementation Model

A classical model of the solution architecture in terms of classes and method, the most important difference with common object oriented approach is that we have two different abstractions, the social level and the single agent level.

3.4.4 Code Model and Deployment Model

A model of the solution at the code level requiring the following steps to produce;

1. Generation of code from the model using one of the following functionalities of the PASSI add ins. It is possible to generate not only the skeletons but also largely reusable parts of the methods implementation based on a library of code and associated design descriptions.
2. Manual completion of the source code.

The Deployment Model is the model's description of the parts of the system across hardware processing units, and their migration between processing units.

3.5 JADE Application Platform

JADE (Java Agent Development Environment) is a software framework to make easy the development of agent applications in compliance with FIPA specifications for interoperable intelligent multi-agent systems. JADE is an open source project, and the complete system can be downloadable from JADE home page. The goal of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents.

The JADE system can be described from two different points of view. On the one hand, JADE is a runtime system for FIPA-compliant Multi-Agent Systems, supporting application agents whenever they need to exploit some feature covered by the FIPA standard specification (message passing, agent life cycle management). On the other hand, JADE is a java framework for developing FIPA-complaint agent applications, making FIPA standard assets available to the programmer through object oriented abstractions.

3.6 The JADE Runtime System

A running agent platform must provide several services to the applications, when looking at the parts of FIPA97 specification, it can be seen that these services fall into two main areas, that is, message passing support with FIPA ACL and agent management with life-cycle, white and yellow pages.

3.6.1 Distributed Agent Platform

JADE complies with FIPA (& specifications and includes all the system agents that manage the platform that is the ACC (Agent Communication Channel), the AMS (Agent Management System) and the default DF (Directory Facilitator). All agent communication is performed through message passing, where FIPA ACL (Agent Communication Language) is the language used to represent messages.

While appearing as a single entity to the outside world, a JADE agent platform is itself a distributed system, since it can be split over several hosts with one among them acting as a front end for inter platform IIOP communication. A JADE system is made by one or more Agent Container, each one living in a separate Java Virtual Machine and communicating using Java RMI. IIOP is used to forward outgoing messages to foreign agent platforms. A special front end container is also an IIOP server, listening at the official agent platform ACC address for incoming messages from other platforms. The figure below shows the architecture of a JADE Agent platform.

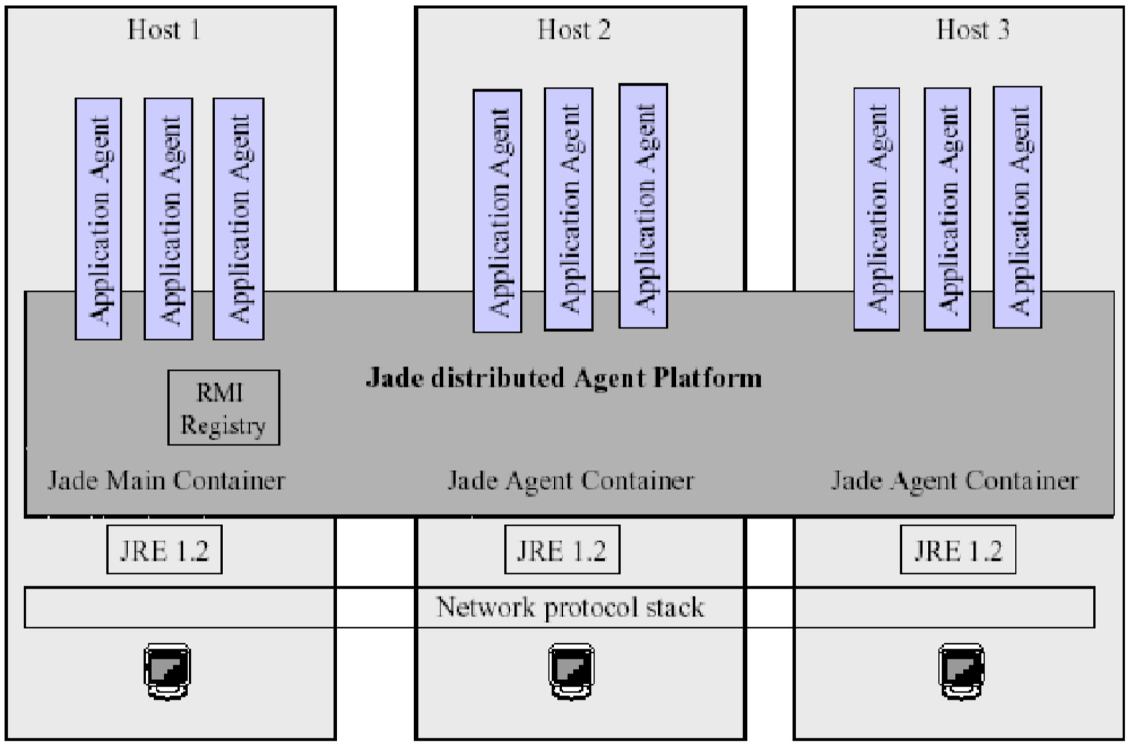


Figure 4: JADE distributed Agent platforms [Fabio Bellifemine, 2002]

3.6.2 Message Delivery subsystem

FIPA agent communication model is a peer-to-peer through multi-message context is provided by interaction protocols and conversations identifiers. JADE uses transport technologies and event dispatching which are typically associated with reactive systems. Clearly, there is some gap to bridge to map the explicitly addressed FIPA message-passing model into the request/response communication model of distributed objects. This is why in JADE ordinary agents are not distributed objects, but agent containers.

3.6.3 User-Defined Ontologies and Content Languages

According to the FIPA standard, achieving agent level interoperability requires the different agents share much more than a simple on-the-wire-protocol. While FIPA mandates a single agent communication language, the FIPA ACL, it explicitly allows application dependent content languages and ontologies. The FIPA specifications themselves now contain a content language library, whereas various mandatory ontologies are defined and used within the different parts of the FIPA standard.

Every JADE agent keeps a capability table where the known languages and ontologies are listed, user defined codec's must be able to translate back and forth between the string format and a frame based representation.

3.7. JADE Agent Development Model

FIPA specifications state nothing about agent internals, but when JADE was designed and built they had to be addressed. A major design issue is the execution model for an agent platform, both affecting performance and imposing specific programming styles on agent developers. JADE solution stems from the balancing of forces from ordinary software engineering guidelines and theoretical agent properties.

A distinguishing property of a software agent is its autonomy, an agent is not limited to react to external stimuli, but it's also able to start new communicative acts of its own. A software agent, besides being autonomous, is said to be social, because it can interact with other agents in order to pursue its goals or can even develop an overall strategy together with its peers. The autonomy property requires each agent to be an active object with at least a Java thread, to proactively start new conversations, make plans and pursue goals. The need for sociality has the outcome of allowing an agent to engage in many conversations simultaneously, dealing with a significant amount of concurrency.

3.8 Data Collection

The data source for testing the model was provided through simulation of data for the agents in the system. The simulation was aided by observation of the different activities carried out by the clients and the insurance officers. Questionnaire was given out to different people who are within the area of study.

CHAPTER 4: ANALYSIS AND DESIGN

4.1 MAHIS Model Design

Multi-agent Hospital Insurance is developed using PHP as the graphical user interface for both the hospital user and the public, JADE as the middleware where the agents are situated and MySQL has been used to develop the database for the insurance company.

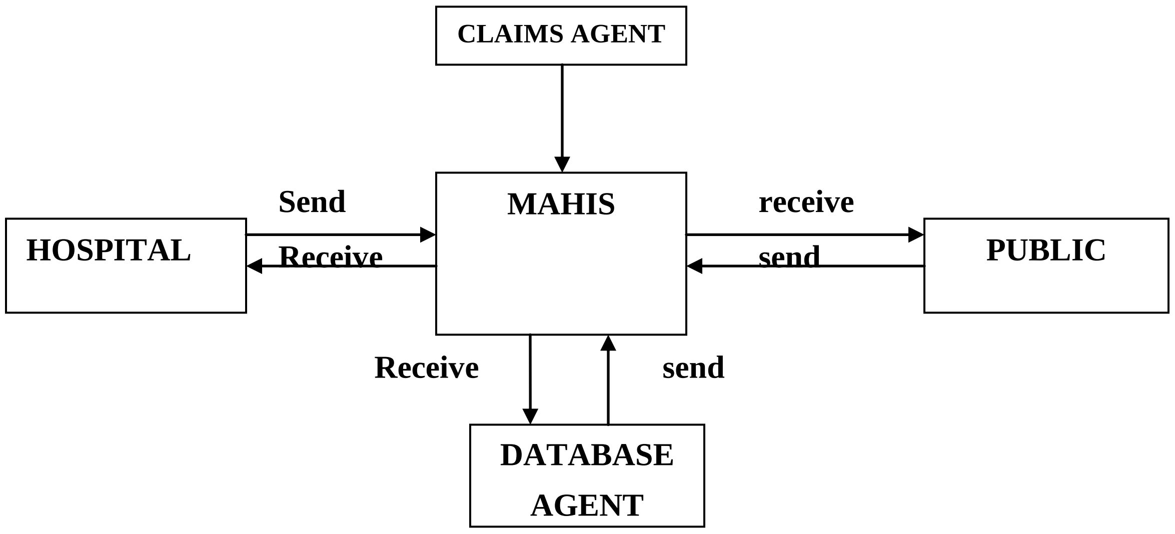


Figure 5: MAHIS system representation

The MAHIS is the internal system that controls the requests it receives and send its responses. The claim is like an external system that fetches the claim requests whereas the public and the hospital are also external but they receive and send requests to MAHIS.

4.2 Model Analysis

Model analysis is the detail examination of the requirements and the conditions that were met in developing the system. In chapter 3 we introduced the PASSI methodology and explained the five models of PASSI namely the Systems requirements, Agent society, Agent implementation, Code model and deployment. In each of the models there were phases that were analysed.

4.2.1 System Requirements Model

It is composed of four phases namely the Domain Description, Agent identification, Role Identification and Task specification. In Domain description Phase, we give the functional description of the system. A public user who is member in the insurance company makes an application and claim to the insurance firm where as the hospital providing the service to the public user, requires the same services of application and claims from the insurance body. Below is a diagrammatic representation;

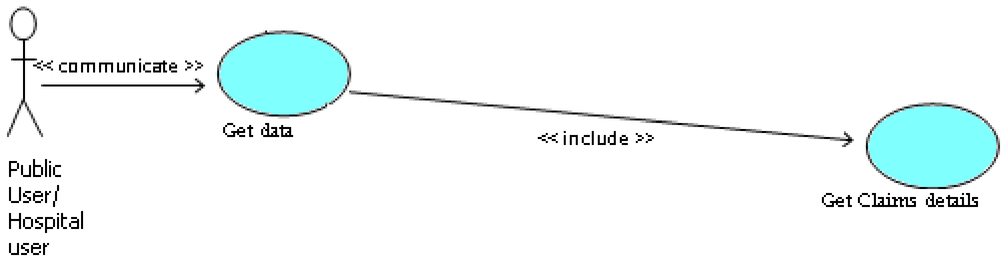


Figure 6: A diagrammatic representation of the domain description of MAHIS

The second phase in the system requirement model is the Agent identification model. According to our domain description diagram we identify the agents. The agents in MAHIS are two namely; the Claims agent and the Data agent (DBconnect agent).

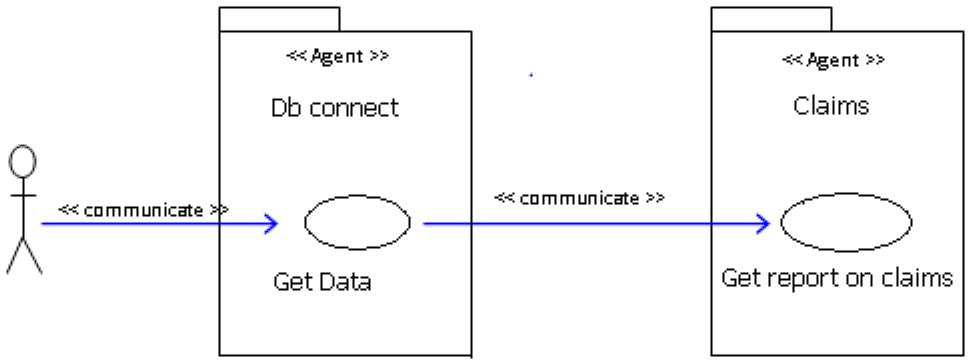


Figure 7: Agent identification model for MAHIS

The third and fourth phases are the role and tasks identification for the agents. The Data agent (DBconnect agent) connects and fetches client data from the MySQL database and forwards requests to the claims agent. It is the intermediary between the front end and the claim processor. The second agent is the Claims agent, it deals with the claims made by both the public and the healthcare centers. The public will make a request for refund if for whatever reason they had to pay for services at the hospital despite the fact they have a membership to the insurance firm. The health centers will make requests of refund to the insurance firm for the services rendered to the patient. A health insurance claim is a bill for health care services that your health care provider turns to the insurance company for payment. After services the doctor sends your bills to an insurance claims processing center. The processing centre gathers all relevant information from the doctor, the patient information sheet, intake forms and any other proper documentation. These are compared to the insurer’s explanation of benefits to see if the policy covers the services. If it does your insurance carrier will submit payments.

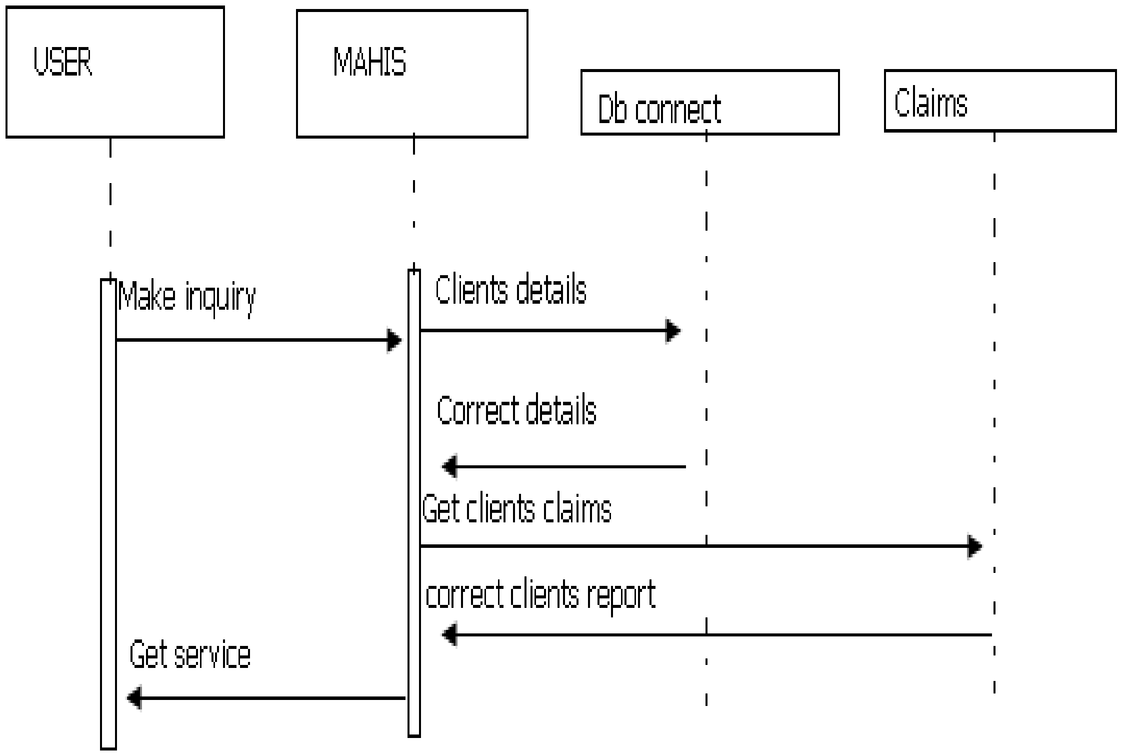


Figure 8: Role identification diagram of MAHIS

4.2.2 Agent Society Model

This is the model of the social interactions and dependencies on claims agent and data agent. The following diagram gives the interactions and dependencies of the MAHIS system and its agents.

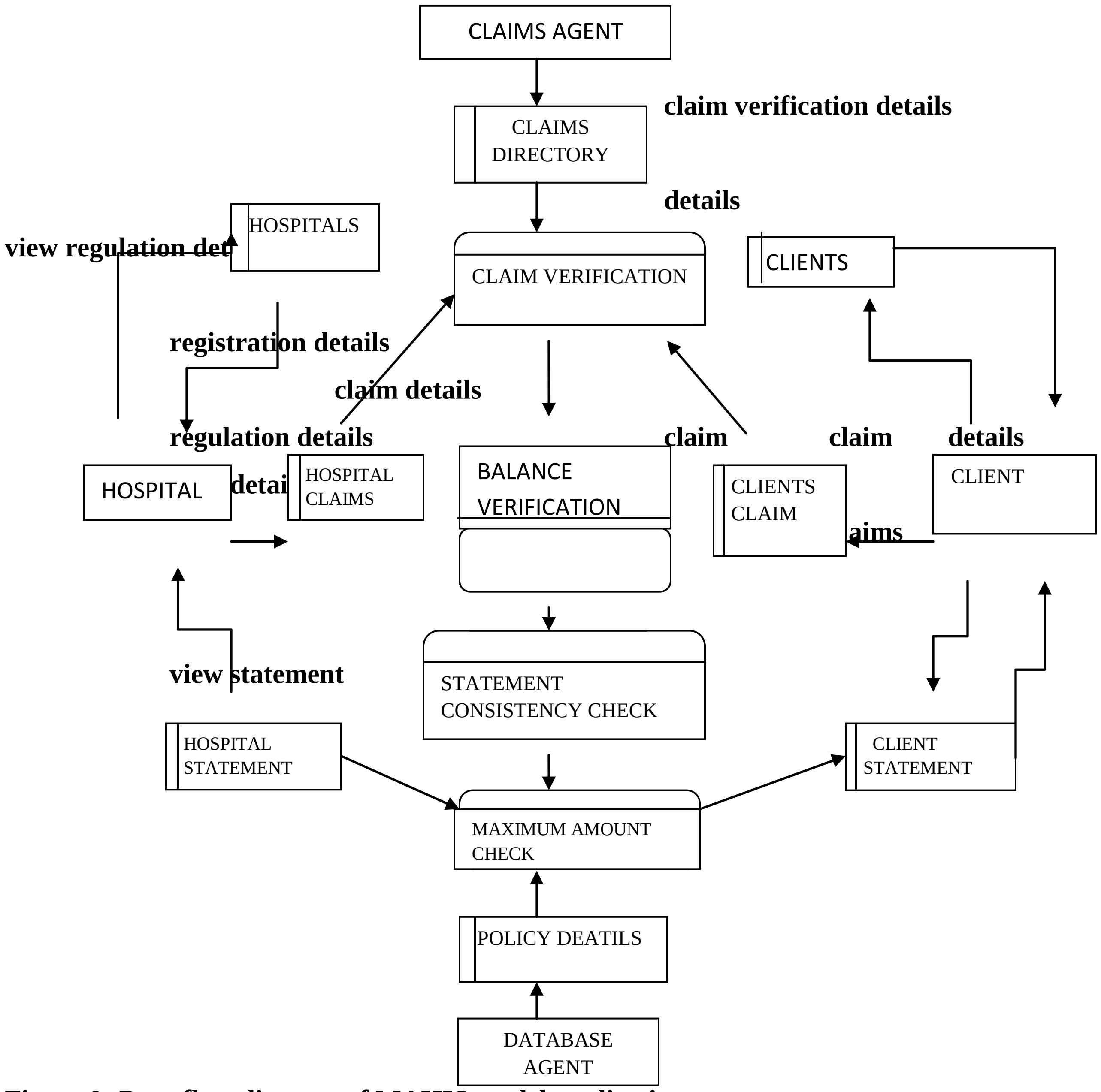


Figure 9: Data flow diagram of MAHIS model application

4.2. Requirements Specification

The system requirements define how user requirements shall be met by the system. The design of the system incorporates several factors, the functional and non-functional requirements.

The following functional requirements must be met by MAHIS.

1. The process should

- i. Enable users to initiate claims, have them approved and show the statement.
- ii. The system should maintain up to date records
- iii. Allow the insurance personnel to monitor and update progress of claims and applications as needed.
- iv. Allow users to make applications.

2. Inputs

There are two inputs

- i. The client user makes the application to the insurance company and receives a password after successful application.
- ii. The hospital user makes claims to the insurance company and if successful the hospital receives a statement which is taken to the bank for payment.

3. Output

- i. The password is the output of the system for the client user and it is in form of a number after application.
- ii. After sending the claim by the hospital their claim is verified and once successful then they receive a statement. The statement is the output for the hospital users.

The Non-Functional Requirements are requirements that do not directly define the core functionalities the system is expected to deliver. For this system they include:-

1. This system should exhibit good performance in the speed of operation, in that whenever a request for a claim is made, the response should fast.
2. The system should provide high security during the claim processing. The data being transmitted should only be accessed by the rightful users.

CHAPTER 5: IMPLEMENTATION AND EVALUATION

5.1 Choice of Implementation Platform

MAHIS has been implemented using a Java-based platform, Java agent development framework (JADE) for agents, PHP for the user interface and MySQL platform has been used to implement the database.

JADE (Java Agent Development Framework) is a software framework to make easy the development of multi-agent applications in compliance with the FIPA specifications. JADE can be considered a middleware that implements an efficient agent platform and supports the development of multi-agent systems. JADE agent platform tries to keep high the performance of a distributed agent system implemented with the java language. In particular its communicating architecture tries to offer flexible and efficient messaging, transperately choosing the best transport available and leveraging state-of-the art distributed object technology embedded within Java runtime environment. JADE uses an agent model and Java implementation that allow good runtime efficiency, software reuse, agent mobility and the realisation of different agent architectures.

Why JADE? Because with JADE, there is an integration of programming languages and creation of one application that would go from end-to-end instead of having to write three separate applications for the database server, application server and presentation client and then write a code for them to communicate with each other.

JADE programs are developed using a user interface that allows programmers to visually create classes and define their properties and methods. Instead of locating methods in large files, programmers select the method they would like to edit and only the code for that particular method is displayed. Also instead of compiling all the code of a program at once, in JADE, each method is compiled individually as soon as the method is completed, meaning code can be checked immediately.

All the code for a JADE application is stored in its object-oriented database. This allows for multi-user development as the database maintains concurrency control, and with each piece of the code being a separate object in the database, it is often possible to recode a system while it is live and online as long as the parts of the system being changed are not in use.
PHP is a programming language designed to generate web pages interactively on a web server. It can be embedded into HTML and is very compatible with the MySQL Database. Unlike HTML, where the web browser uses tags and mark-up to generate a page, PHP code runs between the requested page and the web server, adding to and changing the basic HTML output.

MySQL is a free yet full-featured relational database. MySQL supports several different database engines. The current production release of MySQL is the 5.0*x* version. MySQL 5.0 provides performance that is comparable to any of the much more expensive enterprise databases such as Oracle, Informix, DB2 (IBM), and SQL Server (Microsoft).

5.2 Program Coding

The various form modules were developed independently and each tested for errors and achieving the intended functionality. The stepwise refinement approach was used whereby each form module was developed and errors fixed as the system development advanced. This was the main task in implementation. It involved implementing the following modules:

1. Web Application and Database, This was done on apache web server for the web application and MySQL database management system for the database.
   The initial step was implementing the database in tandem with the ERD that had been designed. The web application interface was then implemented in html as par the input and output design. The application logic was implemented in PHP. This module allowed the interfacing of the application's user interface and the database.
2. Back end JADE server; Jade libraries were imported into the Java project. The default main container was used for holding the agents that were to be implemented. The default container had, The *ams* agent which manages the other

agents and the *df* agent which is used to locate and identify agents. The following two agents were implemented.

3. DB-Connect Agent: This is the main agent which deals with fetching information from the mySQL database and sending request to the other agent which processes claims.

ACL (Agent Communication Language) was used in the communication between the agents. This was implemented in the agent behaviour class RequestProcessor.

4. Claim-Processor Agent

This agent is the main agent that handled claim processing based on client balance, maximum withdraw-able amount as per the policy, client statement consistency and transaction credibility.

5.2.1 Database Connection

Connection with the database was through the java database connector (jdbc) drivers which were imported into the project. The code for database connection is shown below:

5.2.2 User Interface Implementation

All input and output of the system was implemented using web-based forms, using PHP. There are two logins, one for the public user and the other one for the Hospital use.

Public Login

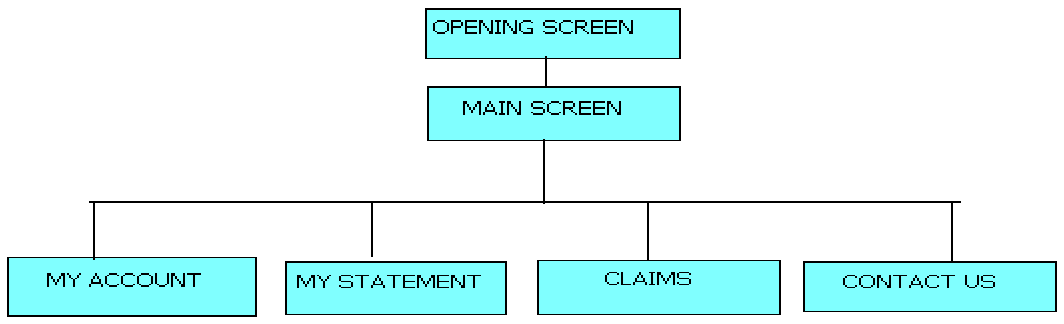


Figure 10: A Public Login schematic diagram

The Public login portal is the Opening screen and has the capability to register and also to login the client to the system. On application the client can either insert a login type of choice depending on the easier to remember. At the login type there is the policy number use or the ID number and then the password of choice. On successful login the client access the main screen that is the home screen which includes links to my account, my statement, claims and contact details.

In my account the user can access personal information like name, ID number, Pin number, telephone number, email, marital status, profession, address, town and country. In my statement the user can access information on the contributions and request for print out of the statement from the site. The claims site provides the user with information on claims status and a platform to submit claim. The contact us is the site where the user can get information on how to contact the insurance company and be able to send a message from the site for any help required.

Hospital Login

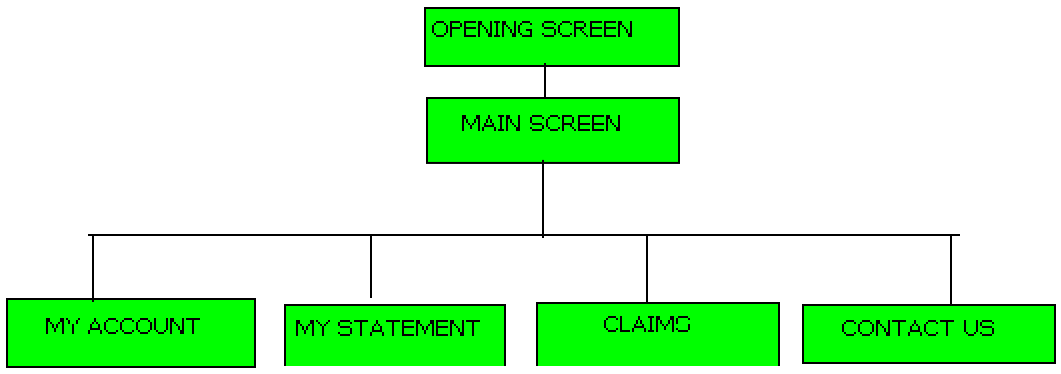


Figure 11: A Hospital Login Schematic Diagram

The hospital portal login which is the opening screen has the option of login using the Hospital number given for the hospital or ID number of the Hospital and then use the preferred password that was used during registration. Once successful the hospital can access the system for posting of claims in order to receive a refund inform of a statement that will be taken to the bank for payment.

The hospital can then access its account details on successful login, the statement on its contributions and deductions, the claims status and contact details.

5.3 Evaluation

Testing is done at four different levels of abstraction: at algorithmic level, class level, cluster/integration level and system level. The algorithmic level, similar to conventional program testing, considers the code for each operation in a class. Class level testing tests the code for each operation as well as method interactions within the class. At cluster level testing, the interactions among cooperating classes are tested. System level testing is done on the complete system where all the clusters are combined.

The processes that were involved during the development of the system involved a series of production activities that were in themselves opportunities for injection of human fallibilities. Thus software testing is a quality assurance activity and presents the ultimate review of specification, design and code generation

The following testing goals were used;

1. Test whether the prototype system is running
2. Test the ability of agents to subscribe to the market and engage in communication
3. Testing the ability of agent system to facilitate administrative work supporting approval of claims, payment monitoring and update of new applications.

5.4 Results Summary

After the above tests were completed, which ran during the development cycle, validation of the entire system was done. Validation succeeds when the application functions in a manner that can be reasonably expected by the end user. Validation criteria used was in line with the system requirements define earlier. The test cases for system testing are shown below:

11	Module tested	Module description	Expected result		Actual result
1	Claim Processing JADE server	This is the multi agent system that processes the claims	**Positive**	Valid claims are accepted and invalid ones declined	All claims were processed within a minute, valid ones were accepted and invalid ones were rejected
			Negative	Valid claims are accepted, invalid ones rejected or claims are not processed at all	
2	Web Login Module	This module is responsible for client and hospital registration and authentication	**Positive**	Valid clients/hospitals are registered and login in well. Invalid users cannot register nor login	Positive
			Negative	Registration of invalid clients/hospitals and invalid users can log into the system	

3	Claim submission	This module is responsible for submission of claims by both the client users and the hospital users.	**Positive**	The system allows clients/hospitals to submit claims	Positive
			Negative	Denial of service in claim submission	
4	Statement	The module allows clients to view statements of their contributions and hospitals to view approved claims	**positive**	Correct statement details are viewed by the user	Positive
			Negative	The user cannot view the statement or the details viewed are incorrect	

Figure 12 : System Test Cases Table

5.5 Running the system

To deploy the system, it required that wampserver is installed in order to have the platform for both the PHP and the MySQL server compounded into one system. The Java netbeans are required for the JADE to be implemented in a Java environment.

CHAPTER 6: CONCLUSIONS

6.0 Introduction

The system demonstrates that indeed you can use agents in the domain of health insurance and that can reduce the number of human experts in an organisation. The MAHIS is able to process claims, reducing the whole process of hospitals or clients claim processing. The Insurance firms are more dependants on the traditional manual system in either their applications process or the claim process. This method is tiring and time consuming. With MAHIS the insurance firm will have a model of a system that can greatly enhance the efficiency and quality of service provided by the healthcare insurance in their service delivery.

The insurance firms can greatly cut cost of hiring specialist to handle the process of claims and registration. MAHIS is able to tackle the process on its own.

6.1 Achievements

The main achievement is the development of a web framework that allows users to know their application status get passwords from the system as a security measure and have statements of their accounts. The MAHIS offers the following functionalities to the users:

1. Making accessible web services
2. Access to information and other services specified through the web framework

6.2 Constraints

1. It was quite difficult to fully subject the MAHIS model to different users (different hospitals that have insurance and clients depending on age group, region, level of education) for testing due to the time constraints on the project.
2. Due to time and resource constraints it was difficult to conduct a thorough research with real users.
3. Developing and the understanding the interactions required for these diversity of users require much of user studies and field research to be carried out. This requires enough time and resources which were limited for this project

According to the history of insurance sector, the way information has been handled has been very tedious and thus there was a need for radical change.
The change is to implement a flexible, efficient and effective way of handling claims and application process.

The project's main objectives were achieved. The development of the system was done within the stipulated schedule. The system was implemented in the most general way based on general requirements. Hopefully it would be acceptable to most of the users. However it is basically a skeleton that can be easily modified to suite the user's unique requirements. This is due to the diversification of people's, business' and organizations' communication needs.

It is believed that the system will come in handy to solve the communication and information needs of its users as per their requirements.
For future improvements it is recommended that other types of insurance covers be integrated into the system to offer a complete insurance system. The web service should be extended fully to support other insurance products and services.

REFERENCES

Adel A. and Mohamed A (2006), *Multi-agent System: Concepts, Theory and Application phases in mobile Robots:* Moving intelligence, ISBN: 3-86611-284-x,pp. 576,ARS/PIV December 2006.

Axtell R. (2000). *Why agents? On the varied motivations for agent computing in the social sciences, in, Centre on Social and Economic Dynamics*, Working paper No. 17, November 2000.

Bellifemine F (2003), JADE A White paper”, http://esp.teleconitalialab.com/, Last Assesed March 4th 2012

Caire G. and Cabanillas D. (2004), ‘ JADE tutorial creating and using applications specific ontologies’, *http://jade.tilab.com/doc/CLOntoSupport.pdf*.

Campo C (2002), “Directory Faciliator and Service Discoveryu Agent ”, FIPA Document Repository, *http://www.fipa.org/docs/input/f-in-00070/f-in-00070.pdf* case of reminders and todos, too (R2Do2), Artificial Intelligence in Medicine,

Chepkoech Caroline (2006), Road Traffic Information System using multi-agents. University of Nairobi

Euyoung K., Hee Y. and Ungmo K (2008). *Mining Based Decision Support multi-agent for personalized e-Healthcare services,* KES-AMSTA 2008,LNAI 4953,PP. 733-742,2008

Feber J.,(1999). *Multi-agent systems: An introduction to distributed Artificial Intelligence,* Addison-Wesley

Foundation for Intelligent Physical Agents. Specifications. 1997. Available from http://www.fipa.org IASTED International Conference on Applied Informatics, Austria, (2000)

J. Vissers and R. Beech, editors. *Health Operations Management: Patient Flow logistics in health care*. Health Management Series. Routledge, 2005.

J.M.H. Vissers, I.J.B.F. Adan, and J.A. Bekkers. Patient mix optimization in tactical cardiothoracic surgery planning: A case study. *IMA Journal of Management Mathematics,* 16(3):281-304, 2005.

James J. Odell (2005), *Foreword in* Brian Henderson-sellers, Paolo Girogini (2005). *Agent oriented methodologies.* Idea Group Publishing.

Joan Cabestany (2009), Bio Inspired Systems: Computational and Ambient Intelligence.

Luck M., Ashri R. and D'Inverno M. (2004),'Agent-Based Software Development',*Artech house Publishers, 2004.*

Minh Tuan Nguyeu, (2008) Enhancing E- Health Information Systems with Agent Technology, Department of Computer Science, University of Fribourg, Received 30 April 2008 September 2008

Ochieng P. O. (2008), Multi-Agent based traffic flow control System-Section of Nairobi. University of Nairobi

Ochola E.O. (2007), A Prototype of share trading system using agents. University of Nairobi

P.R. Harper and A.K. Shahani. Modelling for the planning and management of bed capacities in hospitals. *Journal of the Operational Research Society*, 53: 11-18, 2002.

Rosemary Thomas, (2010), Agent Based Systems for Prediction and Prevention of infectious diseases, University of Windsor, September 20, 2010

Shankararaman, V., Ambrosiadou, V., Robinson, B. Agents in Medical Informatics, Silverman, B.G., Andonyadis, C., Morales, A. Web-based health care agents; the sMas A., (2005) *Agent software and Multi-agent systems: Concepte, Architectures and Applications:* ISBN: 84-205-4367-5, Pearson Education, S.A.,Madrid Spain.

T.O. Paulussen, N.R. Jenniings, K.S. Decker, and A. Heinzl. Distributed patient Scheduling in Hospitals. *In proceedings of the Eighteeen International Joint Conference on Artificial Intelligence* (IJCAI-03), 2003.

Wooldridge M,(2002). *Anintroduction to multi-agents systems,* Published in Feb 2002 by John Wiley and sons in Chichester England.

APPENDICES

Appendix A: Installation Manual

Setup and System Requirements

This section contains information on various requirements the system should meet before you can install this software package.

Minimum Requirements for all systems

1. CD-ROM or DVD ROM drive
2. Available USB port and USB cable
3. 100MB available hard disk space for software installation (WampServer 2.0)
4. 100MB available hard disk space for software installation (NetBeans IDE 7.1.2)
5. Microsoft Internet Explorer 8 or higher

General Installation Information

Install the WampServer 2.0 by creating a folder where it will be installed. Then click install to continue with installation. WAMPs are packages of independently-created programs installed on computers that use a Microsoft Windows operating system.

WAMP is an acronym formed from the initials of the operating system Microsoft Windows and the principal components of the package: Apache, MySQL and one of PHP, Perl or Python. Apache is a web server. MySQL is an open-source database. PHP is a scripting language that can manipulate information held in a database and generate web pages dynamically each time content is requested by a browser. Other programs may also be included in a package, such as phpMyAdmin which provides a graphical user interface for the MySQL database manager, or the alternative scripting languages Python or Perl. Equivalent packages are MAMP (for the Apple Mac) and LAMP (for the Linux operating system). Next step install the NetBeans IDE 7.1.2

Appendix B: User Manual

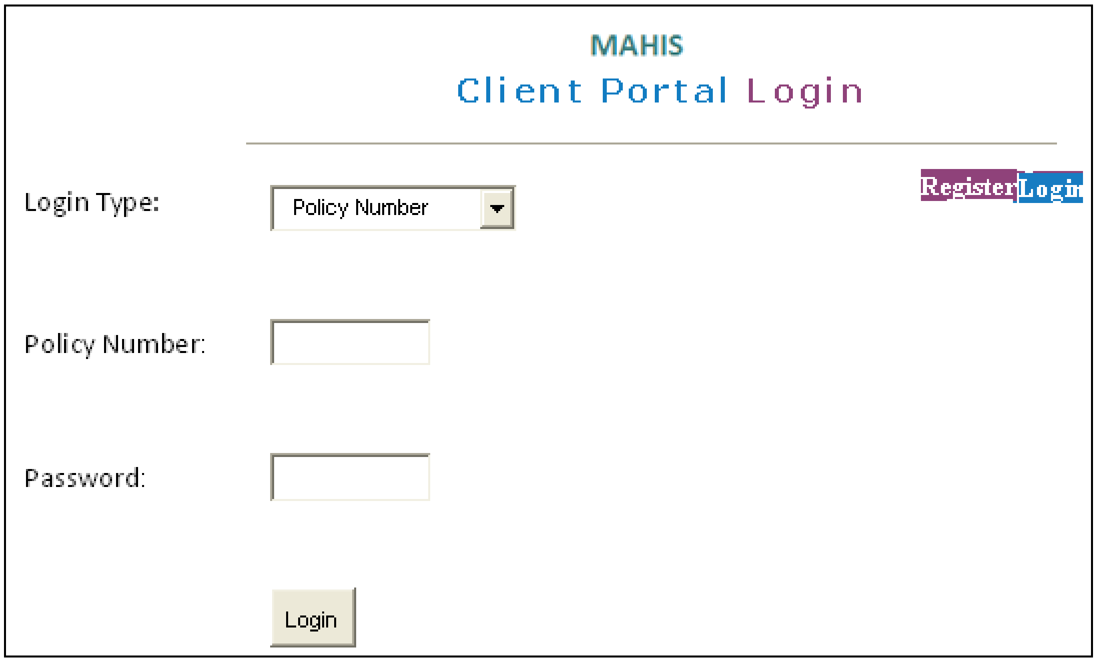


Figure 13: Clients User's Registration and Login process

There are two logins for MAHIS, the first one is the login for individual clients and the second one is for the hospital logins.

The above figure shows the client user's login. This indicates new user's registration process in which the user's profile is accepted into the system so that all activities carried by the user when he logs in are monitored by the agents.

The login type can either be by use of the policy number for the already successfully registered user's or by the ID number for the new users registering for the first time. The new user's use the button clearly indicated register to access the registration page whereby they give their ID number and their new password is confirmed.

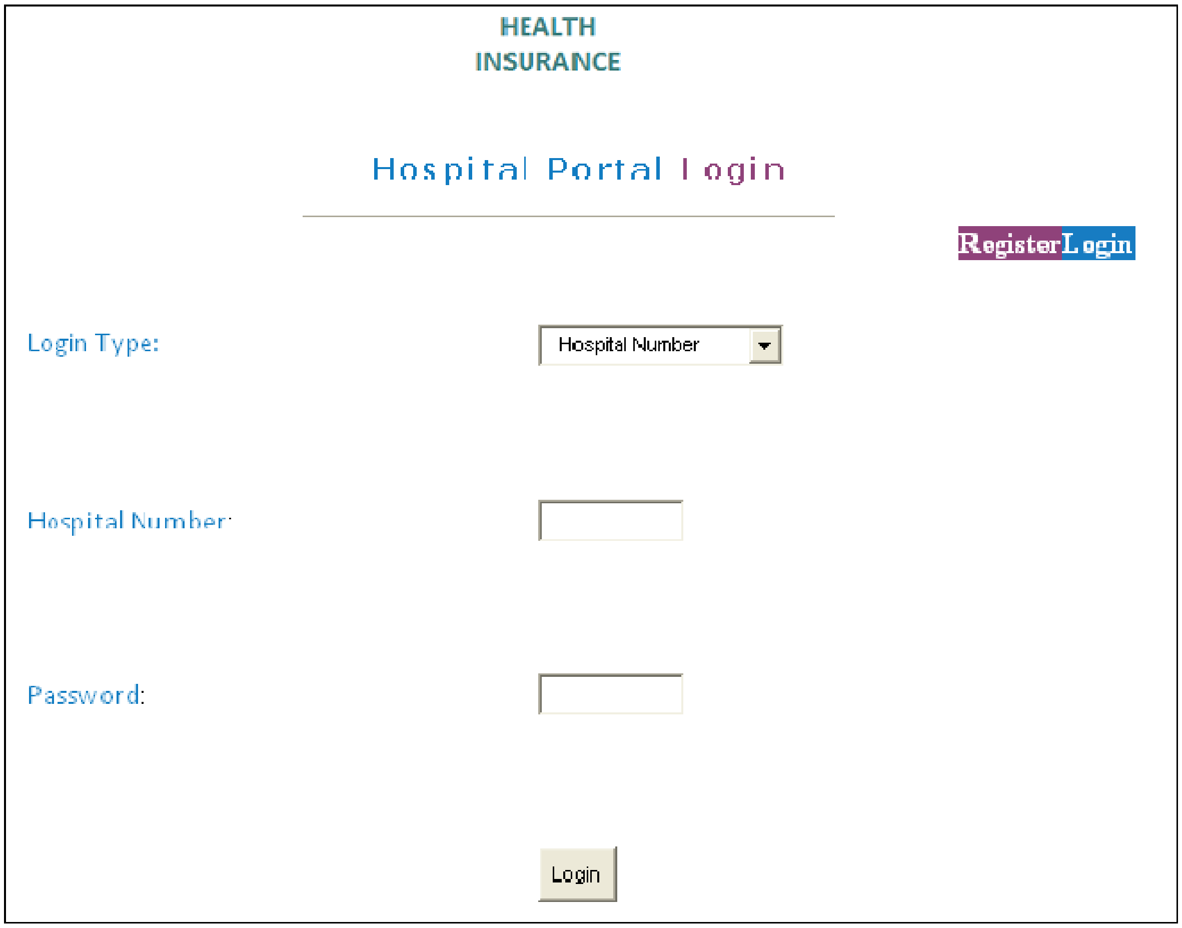
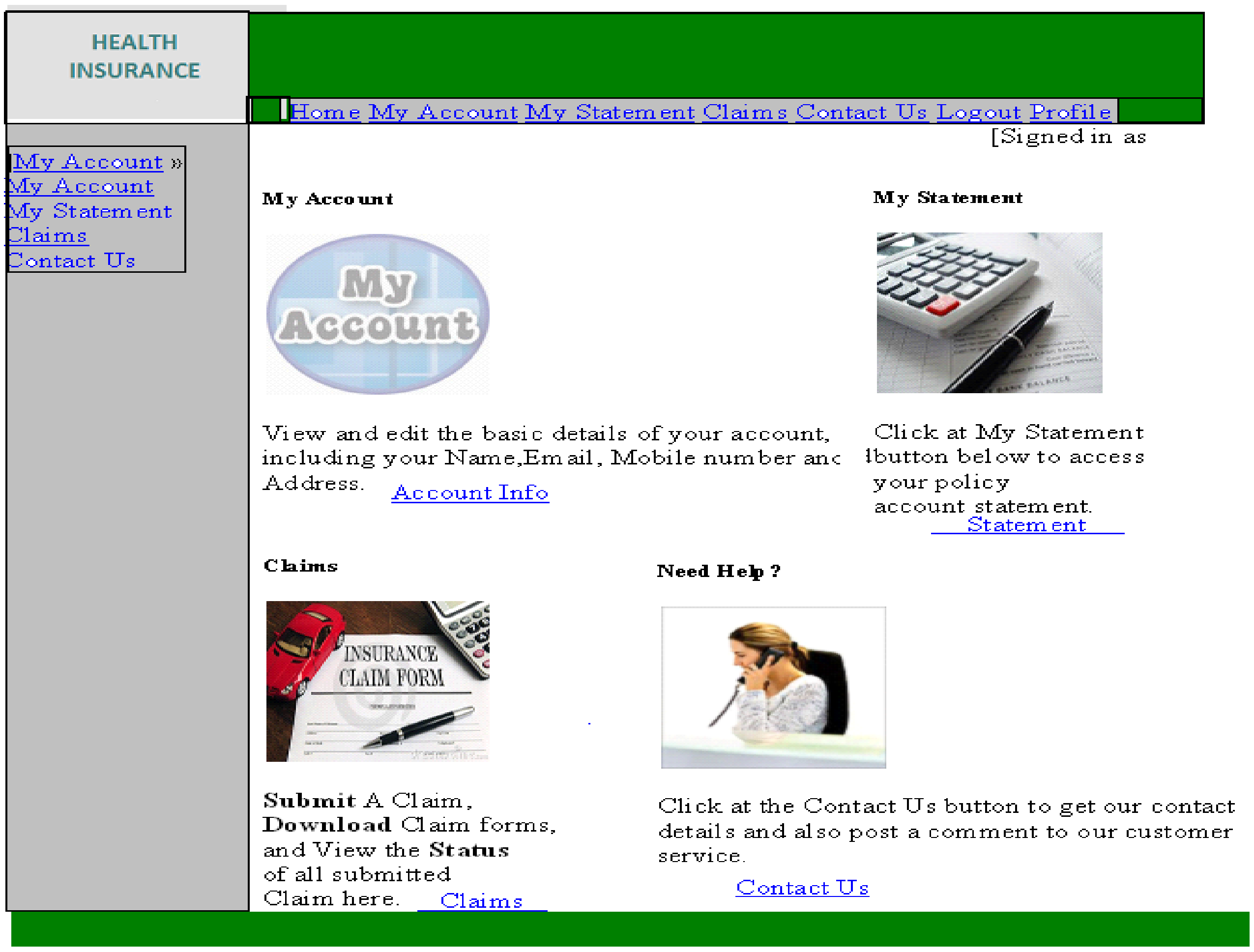


Figure 14: Hospital Registration and Login

This indicates the hospital registration for the new hospitals and the login for the member's hospitals that have already registered. For the new members they register and upon successful completion are given a unique hospital number. The hospital number will be used to access the database.



HEALTH INSURANCE

Home My Account My Statement Claims Contact Us Logout Profile

[Signed in as

My Account »
My Account
My Statement
Claims
Contact Us

My Account

View and edit the basic details of your account, including your Name,Email, Mobile number and Address. Account Info

My Statement

Click at My Statement button below to access your policy account statement.
Statement

Claims

Submit A Claim, **Download** Claim forms, and View the **Status** of all submitted Claim here. Claims

Need Help ?

Click at the Contact Us button to get our contact details and also post a comment to our customer service.
Contact Us

Figure 15: Opening Webpage for MAHIS

The user then logs in after which the various choices are displayed. This includes the user's account, statement and claims.

Appendix C: Codes

ClaimProcessingAgent

```
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class ClaimProcessorAgent extends Agent {

    @Override
    protected void setup() {
        System.out.println("Hello! ClaimProcessor-Agent " + getAID().getName() + " is ready.
Waiting for requests...");
        // Register the service in the yellow pages
        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());
        ServiceDescription sd = new ServiceDescription();
        sd.setType("claim");
        sd.setName("claim_processor");
        dfd.addServices(sd);
        try {
            DFService.register(this, dfd);
        } catch (FIPAException fe) {
        }
        // Add the behaviour serving queries from db agents
        addBehaviour(new ClaimProcessingServer());
    }

    // Put agent clean-up operations here
    @Override
    protected void takeDown() {
        // Deregister from the yellow pages
        try {
            DFService.deregister(this);
        } catch (FIPAException fe) {
        }
        // Printout a dismissal message
        System.out.println("Claim-agent " + getAID().getName() + " terminating.");
    }

    /**
     * Inner class ClaimProcessingServer. This is the behavior used by claim
     * agents to evaluate the correctness and validity of claims and to decide
     * whether to honor or reject the claims.
     */
    private class ClaimProcessingServer extends CyclicBehaviour {

        @Override
        public void action() {
            MessageTemplate                          mt                          =
MessageTemplate.MatchPerformative(ACLMessage.PROPOSE);
            ACLMessage msg = myAgent.receive(mt);
            if (msg != null) {
                // proposal Message received. Process it
                int amount = Integer.parseInt(msg.getContent());
                ACLMessage reply = msg.createReply();
                if (amount <= 1500) {
                    // The amount is less than the maximum, therefore allow claim
                    reply.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
                    reply.setContent("accepted");
                } else {
                    // The amount is more therefore decline claim
                    reply.setPerformative(ACLMessage.REJECT_PROPOSAL);
                    reply.setContent("declined");
                }
                myAgent.send(reply);
            } else {
                block();
            }
        }
    }  // End of inner class
}
```

DBconnect Agent

```
import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.Behaviour;
import jade.core.behaviours.TickerBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DBConnectAgent extends Agent {

    public AID claimprocessor;
    private Connection conn = null;
    private Statement st = null;
    private ResultSet rs = null;
    private ResultSet result = null;
    private ResultSet result1 = null;
    public int status;
    public String reason = new String();
    public int amount;
    public int claimid;
    public int claimby;
    public String userid;

    @Override
    protected void setup() {
        System.out.println("Hallo! DB-agent " + getAID().getName() + " is ready.");

        // Add a TickerBehaviour that checks fot claims every minute and forwards them to the
claimprocessor
```

```
addBehaviour(new TickerBehaviour(this, 60000) {

    @Override
    protected void onTick() {

        DFAgentDescription template = new DFAgentDescription();
        ServiceDescription sd = new ServiceDescription();
        sd.setType("claim");
        template.addServices(sd);
        try {
            DFAgentDescription[] result;
                result = DFService.search(myAgent, template);

            System.out.println("Found the following claim agents:");
                claimprocessor = result[0].getName();
                System.out.println(claimprocessor.getName());
            } catch (FIPAException fe) {
        }

        try {
            conn = mysqlconnect.ConnectDb();
            st = conn.createStatement();
            rs = st.executeQuery("select * from client_claims where status = 0");

            if (rs.next()) {
                claimid = rs.getInt(1);
                String date = rs.getString(2);
                userid = rs.getString(3);
                String hospitalid = rs.getString(4);
                amount = rs.getInt(5);
```

```
                        claimby = rs.getInt(8);

                        result = st.executeQuery("select * from claim_directory where user_id = '" +
userid + "' and hospital_id = '" + hospitalid + "' and amount = '" + amount + "' and date <= '"
+ date + "' ");
                        if (result == null) {
                            status = 2;
                            reason = "no record of use";
                        } else {
                            result1 = st.executeQuery("select * from statement where user_id = '" +
userid + "'");
                            int balance = 0;
                            if (result1.next()) {
                                balance = balance + result1.getInt(5) + result1.getInt(6);
                            }
                            if (balance < amount) {
                                status = 2;
                                reason = "insufficient balance";
                            }
                        }

                        if (status == 0) {
                            // Perform the request
                            myAgent.addBehaviour(new RequestPerformer());
                        }
                    }
                } catch (SQLException ex) {
                    Logger.getLogger(DBConnectAgent.class.getName()).log(Level.SEVERE,
null, ex);
                } finally {
                    try {
                        if (rs != null) {
                            rs.close();
                        }
                        if (st != null) {
                            st.close();
                        }
                        if (conn != null) {
                            conn.close();
                        }

                    } catch (SQLException ex) {
                    }
                }
            }
        });
    }

// Put agent clean-up operations here
    @Override
    protected void takeDown() {
        // Printout a dismissal message
        System.out.println("DB-agent " + getAID().getName() + " terminating.");
    }

    /**
     * Inner class RequestPerformer. This is the behavior used by DB agents to
     * send requests to the claim processor agents
     */
    private class RequestPerformer extends Behaviour {

        private Connection conn = null;
        private Statement st = null;
        private MessageTemplate mt; // The template to receive replies
        private int step = 0;

        @Override
        public void action() {
            switch (step) {
                case 0:
                    // Send the cfp to all sellers
                    ACLMessage propose = new ACLMessage(ACLMessage.PROPOSE);
                    propose.addReceiver(claimprocessor);
                    propose.setContent("" + amount + "");
                    propose.setConversationId("claim_p");
                    propose.setReplyWith("propose" + System.currentTimeMillis()); // Unique
value
                    myAgent.send(propose);
                    // Prepare the template to get proposals
                    mt                                                              =
MessageTemplate.and(MessageTemplate.MatchConversationId("claim_p"),
                            MessageTemplate.MatchInReplyTo(propose.getReplyWith()));
                    step = 1;
                    break;
                case 1:
                    // Receive all proposals/refusals from claim agents
                    ACLMessage reply = myAgent.receive(mt);
                    if (reply != null) {
                        // Reply received
                        if (reply.getPerformative() == ACLMessage.ACCEPT_PROPOSAL) {
                            // Claim has been accepted
                            status = 1;
                            reason = "approved";
                        } else if (reply.getPerformative() == ACLMessage.REJECT_PROPOSAL) {
                            // Claim has been rejected
                            status = 2;
                            reason = "Claim exceeds Maximum Amount";
                        }

                    } else {
                        block();
                    }
                    try {
                        conn = mysqlconnect.ConnectDb();
                        st = conn.createStatement();
                        int rs = st.executeUpdate("update client_claims set status = '" + status + "'
where claim_id = '" + claimid + "'");
                        rs = st.executeUpdate("update client_claims set reason = '" + reason + "'
where claim_id = '" + claimid + "'");
                        Date date = new Date();
                        date.getDate();
                        if (claimby == 0) {
                            st.executeUpdate("insert into statement(user_id, date, type, debit) values('"
+ userid + "','" + date + "','Debit','" + -amount + "')");
                        } else if (claimby == 1) {
                            st.executeUpdate("insert into hospital_statement(hospital_id, date, type,
credit) values('" + userid + "','" + date + "','Credit','" + amount + "')");
                        }
                    } catch (SQLException ex) {
```

```
                    Logger.getLogger(DBConnectAgent.class.getName()).log(Level.SEVERE,
null, ex);
                } finally {

                    try {
                        if (rs != null) {
                            rs.close();
                        }
                        if (st != null) {
                            st.close();
                        }
                        if (conn != null) {
                            conn.close();
                        }

                    } catch (SQLException ex) {
                    }
                }
                step = 2;
                break;
            }
        }

        @Override
        public boolean done() {
            return (step == 2);
        }
    }  // End of inner class RequestPerformer
}
```

MySQL connect

```
import java.awt.HeadlessException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class mysqlconnect {
    Connection conn = null;
    public static Connection ConnectDb(){
        try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection                          conn                          =
DriverManager.getConnection("jdbc:mysql://localhost:3306/healthinsurance", "root", "");
            JOptionPane.showMessageDialog(null, "Connection succeed");
            return conn;
        }catch (ClassNotFoundException | SQLException | HeadlessException e){
            JOptionPane.showMessageDialog(null, e);
            return null;
        }
```

Appendix D: Questionnaire

Name :..(Optional)

Date: ....

(Mark the preferred choice)

1. How long does it take to register with an insurance company?

    a. A day

    b. Two days

    c. A week

    d. A month

2. How effective is the information provided by the insurance officers?

    a. Excellent

    b. Good

    c. Poor

    d. Not helpful at all

3. Do you use any ICT system to access information? Y/N....If yes how do you rate the kind of information you access from the system?

    a. Fair

    b. Poor

    c. Good

    d. Excellent

4. How long does it take to submit a claim manually at an insurance company?

    a. Hours

    b. Weekly

    c. Monthly

    d. More than a month

5. Was it easy to use the MAHIS system? By using a scale of 1-5, 1 being strongly disagree and 5 being you strongly agree.

    a. 1

    b. 2

    c. 3

d. 4

e. 5

6. How fast is the system in registration and submitting a claim? In terms of time taken for the process to complete?

a. One minute

b. 5 minutes

c. 30 minutes

d. An hour

7. How is the organisation of information on the system screen?

a. clear

b. Not clear

c. very clear

d. poor