UNIVERSITY OF NAIROBI
SCHOOL OF COMPUTING AND INFORMATICS

A MODEL FOR SUPPORTING DECISIONS USING MULTI-AGENT SYSTEMS CO-OPERATION: A SIMULATION OF DONOR FUNDING IN HEALTH ENVIRONMENT

BY

ORIEDI DAVID OPONDO

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE MASTER OF SCIENCE IN COMPUTER SCIENCE OF THE UNIVERSITY OF NAIROBI

AUGUST 2011

DECLARATION

STUDENT

I declare that this project is my original work and has not been presented anywhere for academic or any other purposes

ORIEDI DAVID OPONDO

P58/9074/2006

Sign ……[illegible]……………… Date …8/9/2011……

SUPERVISOR

I declare that this project has been presented with my approval as the supervisor

ELISHA OPIYO

SCHOOL OF COMPUTING AND INFORMATICS

Sign ……[illegible]…………… **Date** …13/9/2011……

DEDICATION

To my beautiful and lovely wife, Sophie and

the handsome gift from the Almighty God, my son Lance.

ACKNOWLEDGEMENT

I return all the glory to the Almighty God of heavens, for giving me a sound mind, strength and dedication that led to the research and compilation of this work.

I acknowledge the unwavering intellectual support given to me by my supervisor Elisha Opiyo, Senior Lecturer in the School of Computing and Informatics, University of Nairobi.

To my brother, Mr. Paul Otieno, who gave me this dream long before I even thought of undertaking it. I also appreciate my parents, Mr. and Mrs. Eliakim Oriedi, for exceptional patience and understanding during this period.
Special thanks to my lovely wife Sophie and son Lance for the emotional support I got through the valleys and mountains I encountered in this journey. And to all those who contributed to the success of this work in one way or another, your input has been invaluable.

God Bless You All.

Abstract

This study addresses the development of a multi-agent system model for use in supporting group decision making and the challenges that face the implementation of such a model. As a proof of concept, the model has been applied in a simulated environment in the health domain to help in supporting decisions needed to be made by a donor regarding the funding of a health facility.

One of the challenges that comes up in implementing models for group decision making is how to co-ordinate the different parties (such as people, business partners or software agents) that are involved in making the decision and how to make sure that the system of decision makers functions as a unit with a significant level of coherence.

For example a model can be developed to represent the different levels in an organization structure and how decisions can be supported at such levels or a multi-agent cooperative model can be created for crisis management.

In this project it has been demonstrated that a model for a multi-agent system can use cooperation and coordination to support decision making in a coherent manner – and therefore solve a problem. This has been done by having several agents communicating with each other and a coordination point to ensure the system functions as a unit. An agent model has been implemented using the Java Agent Development Framework (JADE) technology which provides several tools packaged in a framework for implementing agent architecture, behavior and communication. The interaction among the agents has been achieved through agent communication implemented using the Agent Communication Language (ACL) component of the JADE framework. Web based interfaces for input capture and display of output have been implemented using Java Server Pages (JSP). The complete model demonstrates that cooperation and coordination in a multi-agent system is applicable and can be used in providing solutions in group decision making scenarios.

Keywords: Agent, Multi-Agent System, Model, JADE, Cooperation, Coordination

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background

The idea of agency runs across several disciplines. It centers on the concept of delegating tasks by an entity to other entities. An entity in this case could be a person, an object or a software module. This delegation of tasks comes out of the fact that either the delegator does not have enough expertise to handle the task in question or it does not have enough power in terms of processing parameters (memory, processor time).

An agent is defined as anything that is situated in an environment and acts, based on its observation, its interpretation and its knowledge about its situation in such an environment to fulfill a particular action. A software agent is an autonomous software entity that basically carries out tasks on behalf of users. Examples include agents for e-commerce, email readers and search engines.

To maximize the efficiency, performance and optimality of an agent, agents can be organized into groups or societies of agents. This is particularly necessary when a single agent is unable to address a problem because of its inadequate competence, or lack of expertise needed in solving that particular problem. If this is the case, then several software agents can be grouped to form a distributed loosely coupled network and work together to solve the problem. Such an arrangement is described as a Multi Agent System (MAS). MAS are used in such applications and problem solving scenarios that are distributed in nature. In this context, a distributed problem is a problem that requires the agents to work together in a cooperative and a coordinated manner in order to provide a solution. In Multi Agent Systems, the task is decomposed into several subtasks; subtasks are distributed among various agents, and agents interact with each other regarding the best approach to adopt in solving the problem. Because different agents are of the different capabilities, there is need to build the MAS in such a way that the agents may cooperate, coordinate and negotiate in order to achieve the task.

Decision making is the process of selecting a specific action out of multiple alternatives. This process occurs continuously in daily life. For example, humans have to decide on what clothes to wear or what food to eat. In solving a distributed problem, agents in MAS have to make a decision. The decision is made regarding whether or not each agent has the capacity to solve the part of the problem it has been assigned. In order to do this, agents must get knowledge that allow them to be aware of what actions they can or cannot perform. This choice is determined by a set of parameters that define the agents proper understanding of its environment. This parameters – known as Decision Axes – include the agents environmental conditions, the agent's physical features such as its structure and specification and the agent's trust value, meaning its ability to interact and work with other agents.

Decision support involves providing necessary information to enhance the process of decision making. In the context of agents, this includes the methods, mechanisms and techniques that allow agents to be aware of the fact that they must plan and coordinate their actions before a decision or a commitment to carry out a task is made.

In general therefore, to solve a task in MAS, the agents must be able to cooperate in performing the tasks allocated to them according to their individual abilities and expertise and also the agents in the system must be coordinated so that the overall objective of the system becomes to solve the problem at hand. This is the major focus of this research study.

1.2 Group Decision Making

A group denotes a designated set of entities, such as people, business partners or software agents that are working towards a common goal. Group Decision Making is an approach to decision making that involves two or more entities making a decision. In MAS, group decision making takes different forms including negotiation, bargaining, voting and auctions. These models provide different approaches to decision making. In this study, group decision making scenario will involve a number of agents interacting and exchanging information required for making a particular decision – the building of a health facility.

Decisions can be individual – in which case an individual makes a decision without seeking or incorporating the opinion of other parties – or group where the decision made involves a process of consultations with the various parties whose inputs are expected to contribute to the kind of decision made. Every decision maker follows a particular decision style. Decision style is a term used to describe the manner in which a decision maker makes decisions. A manager's design style, for example, is reflected in the way he or she reacts to a given decision making context – what is believed to be of value or importance, how the information is interpreted, how the externalities and forces are dealt with. (Marakas G., 2004).

1.3 Donor Funded Projects

A project has been defined in so many ways. One of the most common characteristics of a project is that it has a beginning and an end. In the context of this study, the term project will be used to describe the building of a health facility which in this case could be an hospital, dispensary or a local initiative meant to bring health services to the surrounding community. Such a facility can be funded by a donor, who in the context of developing countries is usually some western government or donor body. When funding the building of health facilities, such governments and donor bodies make it an obligation that all rules pertaining to the execution of such a project are adhered to. In the building of a health facility, of particular importance is usually the environmental impact assessment carried out to determine the effect of such a facility on water, land, air and the resident population. If such an assessment yields results below levels that are acceptable then the project is not funded and therefore terminated. In the same manner, there are rules regarding the procedures followed in receiving funds from such donors. These rules could have something to do with budgetary priorities regarding the building of health facilities, existing trade relations between the countries or whether the donor is fully funding the project or partially doing so.

What is clear therefore is that for such a project to be allowed to begin, three parties must come to an agreement: the donor, the body/individual carrying out environmental assessment and the financial ministry that deal with issues of external funding. If the three come to a unanimous agreement that the project be allowed to begin, then it will. If they do not agree then the project will not be allowed to proceed. In other words, each of them needs to work with the other (cooperate) and they are brought together by a common objective (coordinate) so that the project can be successful.

In this study, these three parties have been modeled as independent and autonomous software programs – called agents – which have been made to cooperate and coordinate in making decisions that lead to the funding of the health facility or rejection of the project by the donor.

1.4 Problem Statement

Making a decision involves making a choice among a number of alternatives. The decisions can be made by an individual or entity. An entity in this case would be an administrative hierarchy of an organization, a government ministry or an intelligent agent. The contributions made by the various entities into the decision making process are received at a point of coordination at which they are analyzed or aggregated before the final decision is made. If the receipt of these different inputs is not coordinated then the final decision made becomes uncoordinated or flawed. This is the scenario encountered most of the times in cases where there is no co-ordination and therefore no co-operation among the different parties whose input is expected to guide the decision making process. This may lead to a wrong decision taken at the end of the day.

Co-operation and co-ordination are most important when there are people or organizations with different (or possibly conflicting) goals with proprietary information yet their input is needed in making an overall decision. In such a case, it becomes important to have a means of ensuring that the conflicts in individual goals that exist do not override the overall objective of the entire system. For example, in a partnership business with three partners, decision making involves all the three partners in the business. If they cannot co-operate, then a decision will not be made and if it is made then it will not be all inclusive. The lack of unanimity – due to lack of co-operation and co-ordination – therefore derails the entire decision making process. Such challenges are common place in most real life situations where decision making involves different parties. The problem tackled in this research project is that of cooperation and coordination among agents in a MAS. This research project demonstrates that despite the differences in goals and objectives, different agents in a MAS can contribute to the overall objective of the system as a whole by way of being cooperative and coordinated. The coordination and corporation is shown by having the agents carry out their tasks according to their specialties and communicating their output to a coordinator agent. The coordinator agent in turn uses the inputs from various agents to make a request to another agent. In this respect, the agents therefore must cooperate otherwise the task will not be accomplished. The strategy will therefore be to make sure that a final decision cannot be made unless and until the inputs are received from all the agents have been incorporated. The output at the decision making point, in the context of this research project, will be either to proceed with a particular action or not to proceed with the intended action, in this case, either to fund the building of the health facility or not.

1.5 Objectives of the study

1. To construct a model for agent cooperation and coordination in group decision making
2. To test the prototype of the model in donor funding in the health environment
3. To analyze the results and report the findings
4. To develop a model for supporting decisions using Multi-Agent Cooperation

1.6 Significance of the study

Multi-Agent based models for supporting decision making are needed in real life. This is because we make decisions all the time. The challenge however, has been on how to successfully implement such models to fit exactly in real life scenarios. This model will be expected to provide an opportunity for further work in this area in terms of how it can be adapted in the various decision making scenarios in real life other than the domain of health.

1.7 Assumptions and Limitations of the Study

In the development of the model, it has been assumed that the decision making parties represented in the model and the variables used do not take extreme values. For example during the assessment of the environment to give a recommendation, there will be no unpredicted floods or earthquakes. In the same way, there will not be say, drastic economic emergencies that will reverse the decision of the donor or change their priorities. The only circumstances expected to influence the response and therefore the decision of the decision making parties will be the result of individual observation and understanding of their environment and the use of the knowledge at their disposal. It is also assumed that the number of agents in the multi-agent system will not change during the time the system is running.

The scope of this study will not include discussions on the dependability and reliability of this model. This will be left for the extension of this work later on.

1.8 Definition of Important Terms

Agent – an autonomous software entity that performs some task on behalf of the user

Cooperation – working together towards a common goal

Coordination – managing dependencies among decision making parties

Decision – a choice made among two or more alternatives.

Decision Support System – a collection of components with knowledge that can be used in supporting decision making process

Decision Axes – a collection of decision factors such as environmental conditions, physical knowledge and the trust value of the agent that

Group Decision Making (GDM) – a scenario where there are more than one entity, such as people, involved in decision making.

Multi-Agent System (MAS) - a group of agents that are functioning as one entity in a given environment

Negotiation – a situation in which agents interact by exchanging information with a view to reducing conflicts among them influence how it reaches a particular decision.

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 Introduction

Agent technology has been incorporated in computing for some time now. The motivation has been majorly due to the fact that there is a need to have tasks "delegated" so that many tasks can actually be accomplished by autonomous software entities working in a coordinated and cooperative manner. Agent-based systems have indeed emerged as an appropriate alternative to improving traditional computing and current algorithms and software applications especially in dynamic and open environments, where heterogeneous systems must interact effectively to achieve specific goals (Marakas G., 2004). As a result of this therefore, agent-based systems technology has become a new paradigm for conceptualizing, designing, and implementing software systems (Zoltán B., 2008).

J. Cabestany et al (2009) observes that the agents in a multi-agent system 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. These features include autonomy, collaborative behavior, adaptability and reactivity.

2.2 Agent Concept

The agent concept is a general abstraction appropriated to a large range of applications in very specific contexts. This is majorly because the various attributes associated with agency are of differing importance for different domains. More specifically however, agents can be defined as autonomous and problem solving computational entities capable of effective operation and flexible autonomous actions in dynamic, unpredictable and open environments (Salvador, 2008). The aspect of environment brings to focus the fact that agents do not function in a vacuum but rather in an environment where they perceive the environment and respond appropriately to changes taking place in that environment. In addition, an agent denotes a software-based computer system that has several properties such as autonomy, social ability, pro-activeness, reactivity, mobility, rationality etc., which is capable of independent actions to achieve some goals or desires (Wooldridge, 2002).

The potential in agents can best be utilized in scenarios where the agents are functioning as a group. And so agents are often deployed in environments in which they interact, and may be co-operate, with other agents that have possibly conflicting objectives. Such environments are known as multi-agent systems (Luck et al., 2003).

2.3 Agent Modeling

An agent model has two components; an agent class model, which defines abstract and concrete (instantiable) agent classes and captures the inheritance and aggregation relationships between them, and an agent instance model, which identifies agent instances and their properties. (Kinny and Michael,1996). A system with a collection of agents (multi-agent system) will therefore have a collection agent classes basically being instantiated in order to implement the various functionalities expected out of that system. This is the approach of modeling adopted in implementing the model of multi-agent system in this study.

2.4 Agent Coordination

Agent coordination involves managing inter-dependencies between the activities of agents. In order to achieve this, some coordination mechanism is essential. Coordination is interested in fully cooperative multi – agent systems in which all agents share a common goal and their actions are beneficial for the whole system (Ibarra M. S., 08). In this light, agents can select the actions they can execute singly, in a suitable way. Coordination ensures that the individual decisions of the agents result in optimal decisions for the group of agents as a single unit.

2.4.1 Agent Coordination Models

When a multi-agent system is made up of a large number of independently designed components, it may be very difficult to correctly design and manage the system as a whole. In this regard therefore, there is need to define a general framework of coordination for the multi-agent system. This framework is provided by the agent coordination model. A coordination model provides a formal framework in which the interaction of software agents can be expressed. Generally speaking, a coordination model deals with the creation and destruction of agents, their communication activities, their distribution and mobility in space as well as the synchronization and distribution of their actions overtime. From a software engineering viewpoint, a coordination model works as a source for design metaphors, abstractions and mechanisms effectively supporting the definition of the software architecture and the development process of a multi-component software system. According to Paolo et al (1999) a coordination model consists of three elements:

- The *coordinables,* which are the entities whose mutual interaction is ruled by the model. These could be processes, threads, objects, users and, in this context, agents.
- The *coordination media*, which are the abstractions enabling agent interactions, as well as the core around which the components of a coordinated system are organized. Examples are the classic media like semaphores, monitors or channels. In this research project, the coordination media has been implemented using remote method invocation and message passing.
- The *coordination laws*, which define the behavior of the coordination media in response to interaction events. The laws can be defined in terms of a communication language, that is a syntax used to express and exchange data structures and a coordination language, that is a set of interaction primitives and their semantics. The laws in this work have been expressed by well defined communication semantics in form of agent communication language in the Java Agent Development Framework (JADE).

All the above three therefore form a coordination model as depicted in figure 1.0 below.

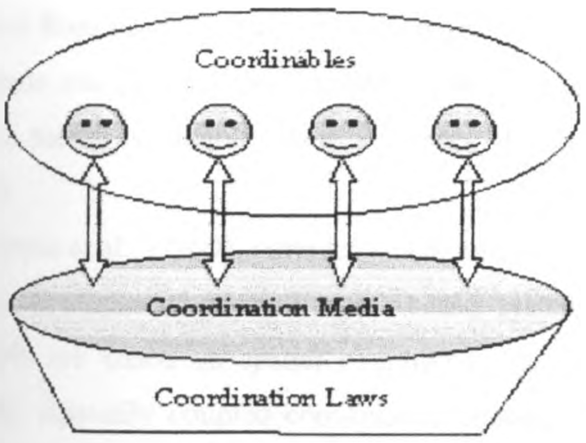


Figure 1: Components of a coordination model – ***from*** *www.agentlink.org*

2.4.2 Control Driven Coordination Model

Agent coordination models can either be data driven or control driven (Paolo et al., 1999). In a control-driven coordination model, agents typically open themselves to the external world and interact with it through events occurring on well-defined input/output ports. The observable behavior of the agents from the point of view of the coordination media is then the one of state changes and events occurring on these ports. The coordination laws establish how events and state changes can occur and how they should propagate. Therefore the coordination media handle the topology of the interaction space among agents, without paying any attention to the data possibly exchanged be include between processes. In general, control driven coordination models suits better those systems made up of a well defined number of entities in which the flow of control and the dependencies between the components have to be regulated and in which the data exchanged is not so important. These include for example computation intensive parallel systems and distributed management systems.

2.4.3 Data Driven Coordination Model

In data-driven coordination models, agents interact with the external world by exchanging data structures through the coordination media which basically acts as a shared data space. The coordination laws establish how data structures should be represented and how they should be stored and extracted from the data space. The coordination media has no perception of the state changes of the agents and does not provide for any virtual connection among the agents. Data-driven coordination model is suitable where a number of autonomous agents have to cooperate (Paolo et al., 1999).

Additionally, Giacomo et al., (1999), provides a taxonomy of agent coordination models that can be applied in the development of multi-agent based systems. According to this taxonomy, coordination models are based on spatial coupling or temporal coupling as induced by the coordination model. Spatially coupled coordination models require the interacting entities to share common namespace and conversely, spatially uncoupled models enforce anonymous interactions. Temporally coupled coordination models imply synchronization of the entities involved and conversely temporally uncoupled coordination models achieve asynchronous interactions. All these models are however designed for mobile agent coordination. They include direct coordination model, meeting-oriented coordination model, blackboard based coordination model and Linda-like coordination model.

2.4.4 Direct Coordination Model

In direct coordination models, agents start a communication by explicitly naming the partners involved (spatial coupling). In the case of inter-agent coordination, two agents are to agree on a communication protocol, typically peer to peer. The access to the local resources generally uses client-server coordination, since a hosting environment usually provides local servers for the management of its resources.

2.4.5 Meeting Oriented Coordination Model

Meeting oriented coordination aims at defining spatially uncoupled models where agents interact in the context of meetings without needing to explicitly name the partners involved. Agents join known meeting points; afterwards, they can communicate and synchronize with the other agents participating in such meetings.

2.4.6 Black-board based Coordination Model

In black board based coordination models, agents interact via shared data spaces used as common repositories to store and retrieve messages. In this sense, interactions are fully temporally uncoupled but since agents have to agree on a common message identifier to communicate and exchange data via a blackboard, they are spatially coupled.

2.4.7 Linda-like Coordination Model

Linda-like coordination models use local tuple spaces as containers of messages. A tuple space is similar to a blackboard, but in addition the accesses are based on associative mechanisms. Information is organized in tuples and retrieved in an associative way via a pattern matching mechanism. This enforces full uncoupling requiring neither temporal nor spatial agreement.

2.5 Coordination Model Applied

In this research project, the data-driven coordination model has been adopted in implementing the coordination taking place in the model. This is because all the agents i.e health agent, environment agent, finance agent and donor agent all share the same data about a particular project being considered for funding. The project details provide a shared data space which is accessible by all the agents. The data in this case is being received through message passing, initially from a control agent to the health agent and then from the health agent to the rest of the agents. This scenario is depicted in figure 1.2 below.

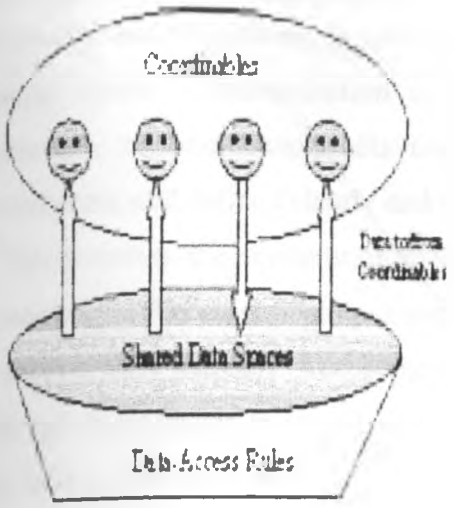


Figure 2: Data Driven Coordination Model (adapted from www.agentlink.com)

2.6 Agent Cooperation

Cooperation refers to coordination with a common goal in mind (Luck et al 2005). Cooperation embraces the allocation and coordination of tasks. Cooperation is necessary and essential for multi-agent systems. According to Mao et al., (2004), service request and offer are the main cooperation manners among agents, where an agent either requests to get services or offers services to other agents. Such a system can be taken as a service oriented system. Request/service is the simplest and most effective cooperation manner. There are two important aspects of a cooperation process namely data movement and computation pattern (Rio, 2007). Data movement refers to the communication process in the cooperation protocol while computation pattern refers to the actual computation functions that are executed to process the communicated data. This research mainly deals with the problem of defining a cooperation and coordination framework to enable the agents in a multi-agent system to solve a problem by way of group decision making process.

2.6.1 Agent Cooperation Models

In general, cooperation needs explicit interactions, and it is a dialog process based on interaction among agents. Communication is the main technology used in accomplishing interaction. Therefore, communication acts and dialog processes in the cooperation model should be investigated and defined clearly and formally.

In this research, the cooperation models considered are the one identified in Mao et al.,(2004). According to this classification, cooperation models can either be based on service request forces or service offer manners. According to the service request forces that the service applicant brings forward the cooperation models are passive, active terminating and active non-terminating. From the service offer point of view, a cooperation model can either be direct or in direct.

2.6.2 Passive Cooperation Model

In passive cooperation model, the service applicant agent requests service offer agent to provide services and the cooperation relationship between the two agents will be terminated when the service is provided by the service offer agent.

2.6.3 Active Terminating Cooperation Model

In this model, the service applicant agent requests service offer agent to provide services actively when some specified conditions are satisfied and the cooperation relationship between the two agents will be terminated when the service is provided by the service offer agent.

2.6.4 Active Non-Terminating Cooperation Model

The service applicant agent requests service offer agent to provide services when some specified conditions are satisfied. However the cooperation relationship between the two agents will not be terminated when the service is provided by the service offer agent. If the specified conditions are satisfied again, service offer agent should provide service again.

2.6.5 Direct Cooperation Model

In direct cooperation model, the service provider agent is just the agent that promises to offer services.

2.6.6 Indirect Cooperation Model

In this model, it is the third party agent, not the service promising agent which provides the service for request agent.

2.7 Cooperation Model Adopted

In this research study, the passive cooperation model has been adopted. The reason for this is that the health agent sends requests to the various agents without fulfilling any specified conditions. The request, which includes a provision of project details, is sent to the environment agent to perform environmental assessment on the location of the project. It is also sent to the finance agent to analyze the financial issues regarding the project and it is finally sent to the donor to decide whether the funding to the project can be granted or not. In all these cases, the cooperation relationship is terminated as soon as the service requested by the health agent has been provided by the respective agent providing that service.

2.8 Agent Cooperation Protocols

2.8.1 Negotiation

Negotiation is the process of reaching agreements on matters of common interest. Any negotiation setting will have four components: (Tuomas W. Sandholm 1999).

- A negotiation set: possible proposals that agents can make.
- A protocol.
- Strategies, one for each agent, which are private.
- A rule that determines when a deal has been struck and what the agreement deal is.

Negotiation usually proceeds in a series of rounds, with every agent making a proposal at every round and involves the following protocol actions: propose, evaluate, revise and accept (Mark et al (1992)).

2.8.2 Bargaining

In bargaining agents agree on outcomes that are mutually beneficial. Agents however still have conflicts of interest on which outcomes to agree on. In monopoly one agent gets all the benefits of the interaction. In a perfect competition no agent gets the benefits of the interaction. Real world situation is neither necessarily purely monopolistic, nor supporting perfect competition.

2.8.3 Auctions

An auction takes place between an agent known as the *auctioneer* and a collection of agents known as the *bidders*. There are three auction settings namely private value auctions, common value auctions and correlated value auctions.

- *Private value auctions*. The value of goods depends only on the agents. Only the winning agents utilize the goods and do not re-sell them. Agents are assumed to know the values exactly.
- *Common value auctions.* The value of the goods depends entirely on others value of the good. For example, where the goods are re-sold such as in treasury bills etc, tea or coffee auctions.
- *Correlated value auctions.* Value of goods depend partly on agents own preferences and partly on value of others such as in negotiation within a contract.

2.8.4 Voting

Voting occurs on some social choice such as a swimming pool, public health centre, etc. All agents give inputs to a mechanism. The mechanism then makes a choice that will apply to all agents

2.9 Group Decisions and Decision making

A group refers to a designated collection of people, objects or software agents. Group decision making is the process of settling for a particular action by a group after having reached an agreement that the action can be carried out. This involves input from every party in the decision making process. Group decisions are therefore joint decisions made by the group.

2.9.1 Multi-Agent Systems

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 M. S., 2008). In multi-agent systems, "the intelligence" arises from the aggregation of simple competitions as well as the task assigned to every individual is as important as the collective task. An intelligent agent in this context is one that has characteristics such as reactivity, pro-activeness and social ability (Wooldridge, 2002). In this research work, the interaction will be between a number of agents as depicted below:

2.9.2 Multi Agent Systems and Group Decision Making

Research in Multi-Agent Systems is concerned with the study, behavior and the construction of a collection of possibly preexisting autonomous agents that interact with each other and their environments (Sycara 1998). According to (Wooldridge 2002), multi-agent systems can be integrated by a group of autonomous agents with different capabilities such that the ability to communicate among themselves and to make collective decisions aims to improve cooperative agents' performance in dynamic and unpredictable environments. In fact multiple cooperating agents hold the promise of improved performance and increased fault tolerance for large scale problems such as planetary survey (Haldemann et al., 2007) and habitat construction (Howard 2005). To reach a decision, the agents in a MAS must be aware of their roles in the decision making process and act only to the extent allowed by their capabilities. This means that the agents must be able to communicate and exchange as much information as possible during the decision making process.

2.9.3 Multi-Agent Decision Support

Decision making process has stages which it goes through before a final decision is made. There are three distinct stages for every decision making process. (Simon (2007):

Intelligence: Fact finding, problem and opportunity sensing, analysis, and exploration.

Design: Formulation of solutions, generation of alternatives, modeling and simulation.

Choice: Goal maximization, alternative selection, decision making, and implementation.

The skills of an agent, however, are limited by its knowledge and the perspective of its situation on an environment especially when the agents must coordinate among themselves. An agent's situation refers to all the information that an agent has to decide if it can or cannot execute any proposed action. Specifically, these information elements are directly estimated from three points of view (Salvador, 08):

- *Agents' environmental conditions (World)* – composed of information about the state of the environment, directly involved in the performance of a cooperation action.
- *Agents' physical knowledge (awareness)*- meaning the specification, the structure and other relevant details related to the agents' physical skills and characteristics.
- *Agents' trust value (interaction)* - related to the capability of an agent to communicate, to interact and other relevant details to enable it work together with other agents.

The above, known as the decision axes, indicates that a complex problem will be decentralized when agents will be able to have both an individual perception of its situation in the world and therefore use such information in their decision making aiming to make correct decisions and to achieve trustworthy commitments in cooperative environments.

In this sense, agents support decisions based on the knowledge they have about the problem domain and the amount of information that has been made available to them. This decision support finally leads to a joint decision made by the group of agents hence provision of a solution to the problem.

2.9.4 Multi-Agent System Models for Decision Support

A number of models have been developed to show how the concept of multi-agent systems can be used to represent real life decision making parties in group decision making circumstances. For example, they have been used in the management of urban traffic (Sascha O. et al (1998)), in hospital management (Eunyoung K. et al (2008)) and in stock trading (Yuan L. et al (2008)).

2.9.5 Challenges in Multi Agent System Model Implementation

In implementing a model for MAS cooperation and coordination, a number of challenges come up. The challenges exist both in the aspect of coordination and cooperation. Mark et al (1992), notes that coordination raises the question of cost (of coordinating different agents), risk of failure (during the coordination process) and misinterpretation (among different agents). If any of these occur, then the entire coordination process might be derailed.

Cooperation also has a challenge in implementation and the challenge comes in form of the manner in which it can be implemented. Mark et al (1992), points out that the issue is whether to implement the MAS as a framework according to the domain of the problem, work together to improve individual performance or work together to improve the overall performance of the system.

This study demonstrates cooperation and coordination by tackling these challenges. This will be done by having the agents operate according to a defined measure of performance while negotiating.

2.9.6 Theoretical Framework

The area of agents and particularly MAS is based on a number of theories. These theories form the theoretical foundation upon which the study of multi-agent systems is done. The theories not only affect the way individual agents are designed and implemented in a MAS but also how the behavior of those agents are implemented. This becomes particularly important in cases where the agents are interacting through cooperation and coordination.

2.9.7 Multi-Agent Theory

Multi-Agent Theory provides the framework within which every MAS is built. This theory provides a way of having several agents implemented and operating in the context of a system. In this sense, several agents will be having their objectives geared towards the overall goal of the system of agents rather than their objectives as individual agents. The theory lays a foundation for implementing agent aspects such as coordination, cooperation and distributed problem solution.

Adel A. and Mohamed A. (2006) describe Multi-Agent Theory as a combination of agent cooperation, coordination, cooperative problem solving, coalition formation and negotiation. All these aspects work together to ensure the success of a multi-agent system and they form the foundation of multi-agent systems. The agents constituting the MAS can be a software routine, robot, sensor, process or person, which performs actions, works and makes decision (Arenas & Sanabria, 2003). As a system of agents, a MAS has been found to greatly improve performance, productivity, efficiency and effectiveness of the systems in which they are in-cooperated.

However, Changhong et al., (2002), divides Multi-Agent Theory into two parts: Theory and Application phases. The theory phase deals with cooperation taxonomy, cooperation structure and cooperation forming procedure. The taxonomy of the cooperation defines the manner in which groups of agents come together depending on factors such as design objectives, goals and their environments. Cooperation structure deals with the overall framework of the cooperation of the MAS, for example, the number of agents involved in a particularly MAS and whether the MAS is a closed system or an open one. These factors influence the operations within the MAS by either restricting what the agents can do in terms of when, where and with which agent or whether other agents can get into this system or not. Cooperation forming procedures defines the protocols of operation within the MAS such as protocols of communication and decision making.

The application phase involves mobile agent cooperation, information gathering, sensor information and communication.

The Multi-Agent Theory is the framework within which the agents in this research project have been implemented. All the agents in the MAS are cooperating towards a joint decision – the funding of a health facility. The implementation of this kind and level of cooperation has borrowed heavily from the Multi-Agent Theory particularly in the cooperation taxonomy, cooperation structure and cooperation forming procedure of the agents in this system.

2.9.8 Game Theory

A game is a formal description of a strategic situation. Game Theory is the formal study of decision making where several players must make choices that potentially affect the interests of the other players. Game Theory provides a framework for a formal study of conflict and cooperation. Game theoretic concepts apply whenever the actions of several agents are interdependent. These agents may be groups, individuals, firms or combinations of these. The concepts of game theory provide a language to formulate, structure, analyze and understand scenarios. In Game Theory, every agent makes decisions based on the relative payoffs of such decisions. A payoff in this context is a number, also called utility, that reflects the desirability of an outcome to an agent for whatever reason. When the outcome is random, payoffs are usually weighted with their probabilities and the expected payoff reflects the agent's attitude towards the risk associated with the decision made.

The classic game theoretic question to ask in any multi-agent encounter is, according to Parsons and Wooldridge (2000), what is the most rational thing an agent can do? In most multi-agent encounters, the overall outcome will depend critically on the choices made by all agents in the scenario. This implies that in order for an agent to make a choice that optimizes its outcome, it must reason strategically. That is, it must take into account the decisions that other agents may take and must assume that they will act so as to optimize their own outcome. Game Theory gives a way of formalizing and analyzing such concerns. In particular Game Theory provides a foundation on how interaction strategies can be designed in order to maximize the welfare of an agent in a multi-agent encounter, and how protocols or mechanisms can be designed that have certain desirable properties.

The agents in this research project are self interested agents, in other words, each of them is working in order that its interests are taken care of. However, the decision of each of the agents is affecting the outcome of the decision from the entire MAS. This means that if one or more agents maintain a hard line position concerning its interests – and if this position happens to be against the desired outcome – then the outcome from the MAS will be negative. In this respect, the agents modeled here as the players in this system have been designed with a level of flexibility concerning their interests. This has been based on randomly generated values which determine the kind of decision an agent comes up with. So while there is a clear scenario of winning and losing as far as the joint decision making process is concerned, the design approach has allowed room for a renegotiation process which might compromise on a few aspects of the decision for the sake of the overall outcome of the system as a whole. For example – in this case - the decisions made by the environment agent and the finance budget will influence the decision of the donor agent for funding the facility. But there are times when the need for a health facility in a particular community could actually override the budgetary hindrances that might arise in such cases. In such a case, as much as the finance agent may have a reason to oppose the funding, the agents might go back to renegotiate for the sake of the overall benefit. The renegotiation process will then be expected to influence the finance agent to reason in favor of the overall outcome. This element of flexibility in the negotiation process is based on Drama Theory.

2.9.9 Drama Theory

Different from Game Theory, Drama Theory focuses on how conflict that happens during agent interaction can change because the agents want to eliminate dilemmas using positive or negative emotions. Drama Theory depicts agent interactions as evolving characters each seeking simultaneously to have others adopt their positions in collaborative situations. The dilemmas represent the challenges that each party seeks to overcome either by managing conflicts and establishing a shared solution or managing dilemmas faced in characteristic and repeatable ways. Drama Theory asserts that full conflict resolution generally requires players to engage in rational emotional process of redefining both the game and their positions in it until there exists a fully satisfactory resolution on which they all agree. In redefining the game, the players must eliminate the dilemmas, each of which tends to cause emotions and rationalizations tending towards its elimination. Drama Theory recognizes six dilemmas including threat, rejection, positioning, persuasion, cooperation and trust dilemma. Of the six dilemmas, cooperation and trust dilemmas are collaboration dilemmas. According to Howard (2009), Drama Theory proves that in general a satisfactory resolution cannot be found if preferences and beliefs in MAS are fixed. The agents' "minds and hearts" have to change depending on the prevailing circumstances. In this sense, the drama theoretic framework assumes that first, each party openly states a position (the position that it advocates for). Then if positions differ, it states a fallback strategy (this being what it will unilaterally do if positions do not change). This is the framework assumed by Drama Theory. In this framework therefore, it is proved that no disagreement and no distrust requires the non existence of six independent dilemmas. Drama Theory therefore predicts that each party will use emotion and argument to try to change hearts in a way that eliminates the six dilemmas.

In the context of this research work, Drama Theory is applied because of this element in the change of heart during the negotiation and decision making process. The agents have been designed to consider cases where the overall objective overrides individual objectives and, based on their level of flexibility, slowly gravitate towards a decision that favors the overall outcome. Ideally, Drama theory is as in figure 1.2 below.

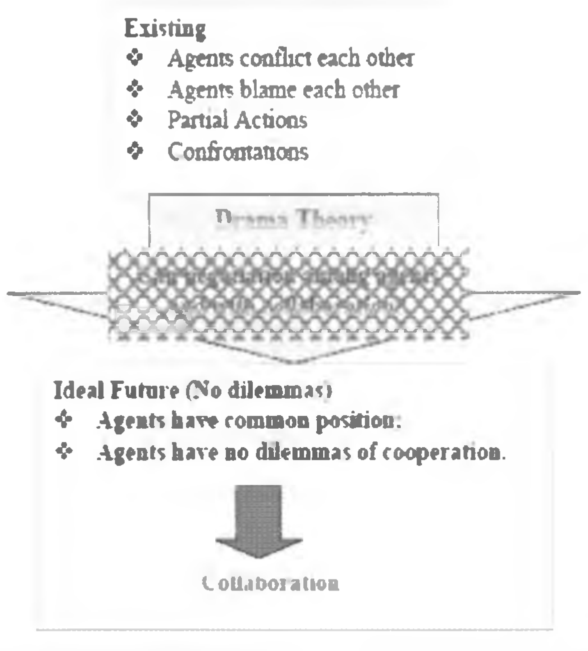

Existing
❖ Agents conflict each other
❖ Agents blame each other
❖ Partial Actions
❖ Confrontations
Drama Theory
Ideal Future (No dilemmas)
❖ Agents have common position.
❖ Agents have no dilemmas of cooperation.
Collaboration

Figure 3: Agent Based Simulation of Negotiation process using Drama Theory (*from Howard 2009*)

2.9.9.1 Conceptual Framework

Conceptually, the agents interact as depicted in figure 1.0 below

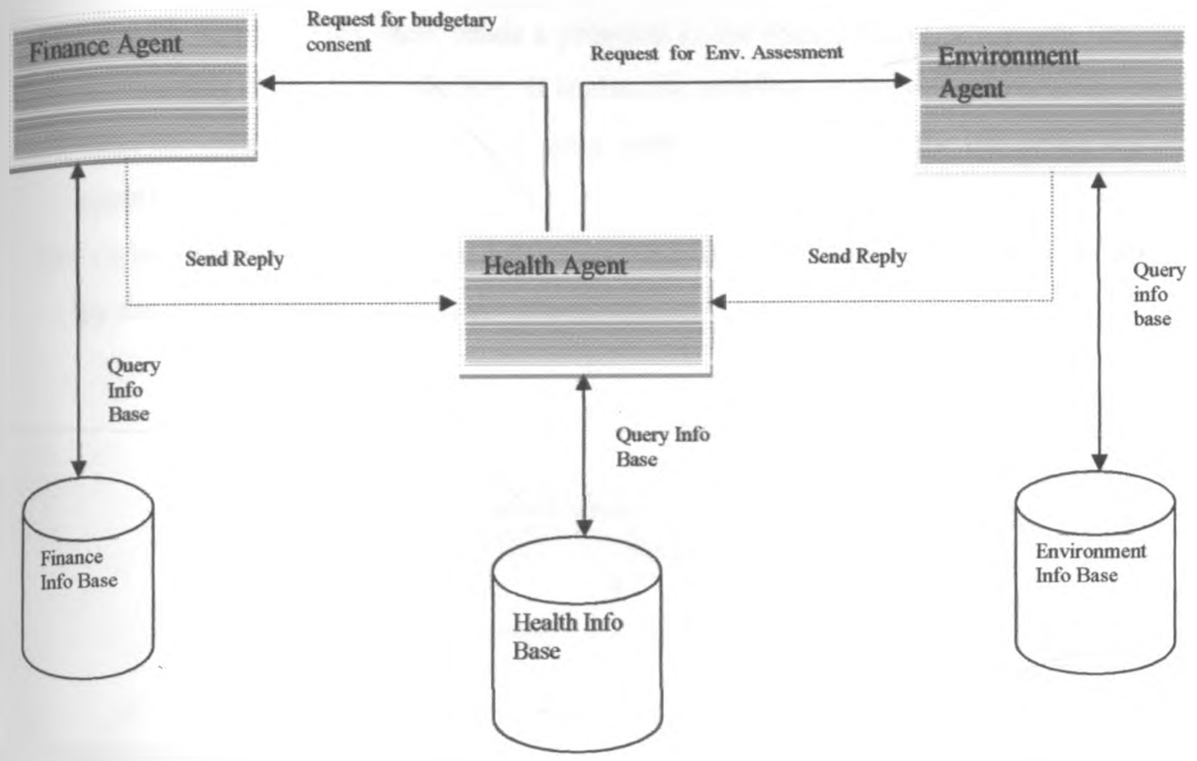


Figure 4: Conceptual Framework - Agent Interaction

This model is based on the idea of group decision making in a multi-agent environment and the decision making process is supported by the agents through their co-operation and coordination.

The model will have a number of components that will be working together to support the final decision made by the system.

The parties participating in the decision making process have been modeled as agents with specific roles as outlined below:

The Donor Agent – This is the provider of the funds to facilitate the building of the health facility. The funds are released only if the recommendations received from all the parties conform to the expectations of the donor agent.

The Finance Agent – This agent facilitates the funding of the project by analyzing policy issues associated with external funding and budgetary allocations.

The Environment Agent - This agent performs environmental assessments regarding the building of the health facility and sends an appropriate recommendation based on the suggested location of the health facility.

The Health Agent – This agent sends a proposal to the Donor Agent requesting funding for the building of the health facility. It is also the recipient of the various recommendations from the other agents. It will acts as a point of coordination for the entire system of agents.

The various requests and proposal between all the agents leading to the release of the funds by the donor is as shown below:

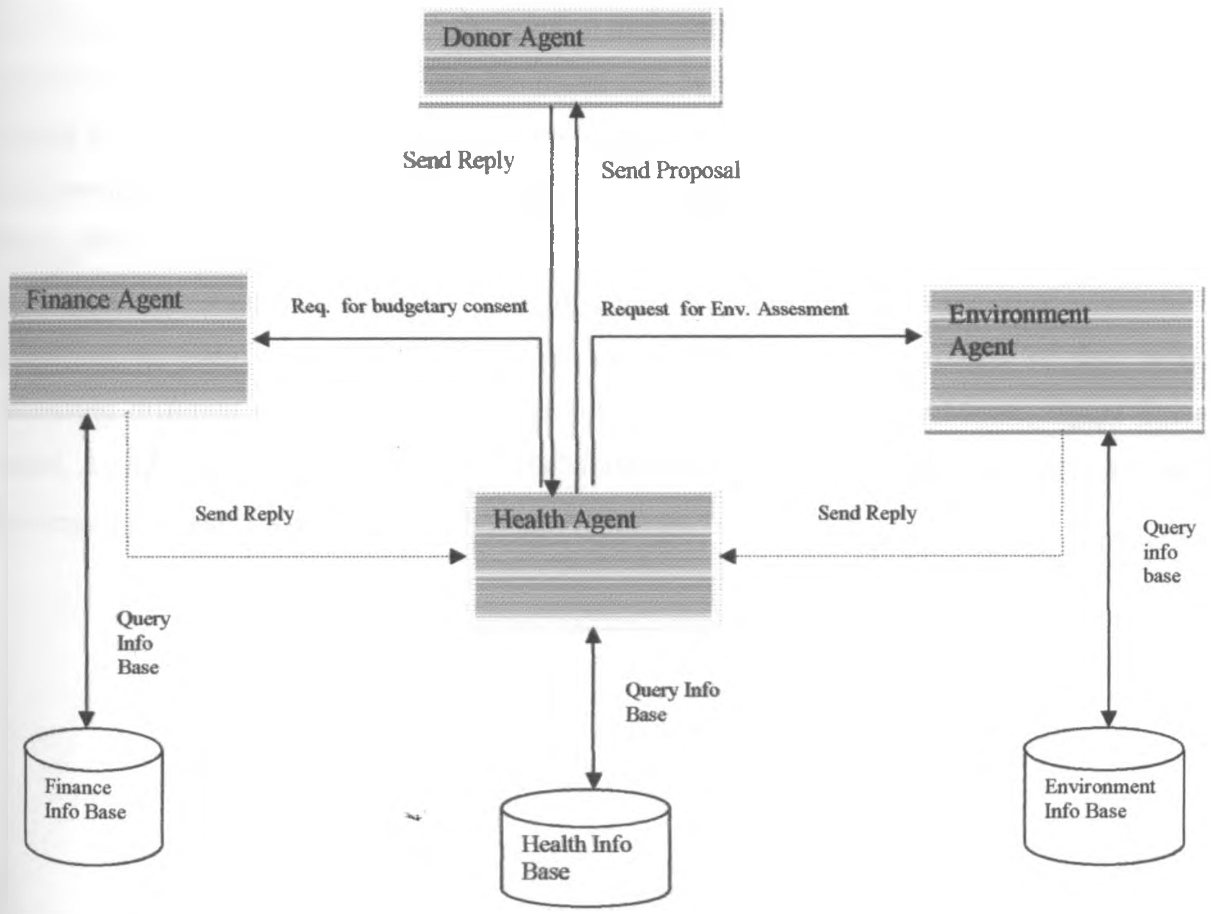


Figure 5: Model Application in Donor Funding

two software engineering tools: the first is the software development methodology that can design the cooperative agent-based system; the second is the agent cooperation model that can manage the team behavior at runtime.

3.2 Research Design

The approach to this study begins with a review of the work that has been done before on multi-agent system and decision making. A number of existing models have been reviewed and the challenges highlighted. The model is applied in a donor funding scenario to test its applicability. To create a group decision making scenario, there are a number of interacting agents which are exchanging information in order to reach a decision based on their individual expertise and the information available at their disposal.

3.3 The PASSI Agent Methodology

The Process for Agent Societies Specification and Implementation (PASSI) is a step by step requirement-to-code methodology for designing and developing multi-agent societies, using the Unified Modeling Language (UML) notation (Cossentino, 2005). 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 a system. According to Cabri et al., (2005), the PASSI methodology is composed of five models addressing different design levels of abstraction: System Requirements Model, Agent Society Model, Agent Implementation Model, Code Model and Deployment Model. Below is a diagram showing the various phases of the PASSI methodology.

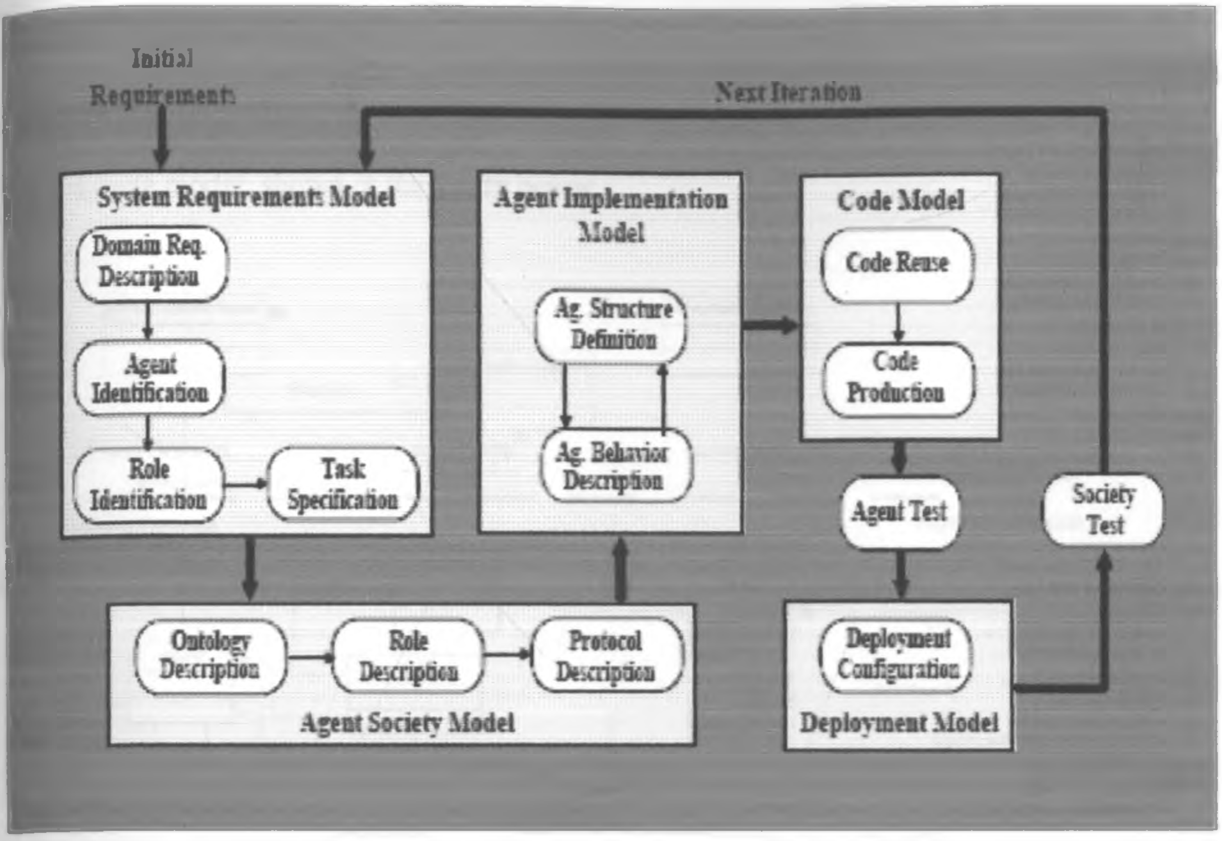


Figure 6: Phases of PASSI Methodology (*from: Massimo Cossentino (2005)*)

In the System Requirements Model, there is the role identification phase, where a series of sequence diagrams exploring the responsibilities of each agent through role specification scenarios, are created. In the Agent Society Model, there is the role description step where class diagrams are used to show the role played by agents, the tasks involved, communication capabilities and inter-agent dependencies. Roles are very important in this methodology and so they are very well addressed.

In the ontology description step of the Agent Society Model, constraints are used to describe the knowledge of each agent, with the help of Object Constraints Language (OCL). Society rules are then introduced in the role description phase.

An agent can play several functional roles during interaction with other agents to achieve its goals. A role is a social concept: a collection of tasks performed by the agent in following a sub goal or offering some services to the other members of the society.

Furthermore, the PASSI meta-model is divided into three logical areas: the problem domain, which is directly related to the System Requirement Model, the solution domain and the agency domain, which are related to the other models. The three logical areas together constitute the PASSI meta model, shown in the figure below.

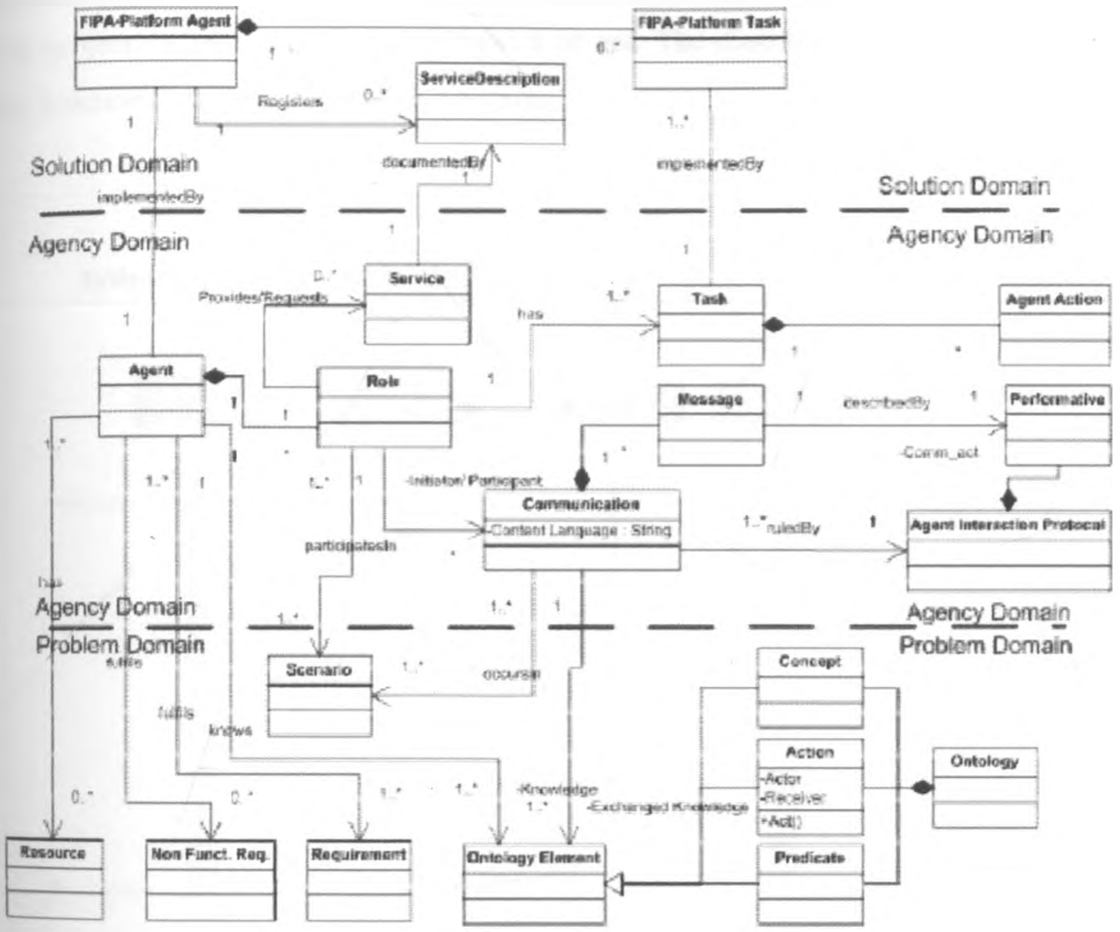


Figure 7: The PASSI Meta Model *(Cabri et al 2005)*

3.3.1 System Requirement Model

In this research study, there are four major agents in the multi-agent system. The others agents are providing control, coordination and initiation services to the main agents. The Health agent receives input in form of project parameter specification. It then sends this input to the Finance agent which performs budgetary analysis on the request made. This analysis involves checking if the donor country interested in funding the project has trade agreements with the country in which the project is being done. It also checks on the priority of project in relation to other projects and whether the funds are released to the Health agent directly or if the funds will be released through the Finance agent. The Health agent also sends the project input to the Environment agent. The Environment agent performs an assessment of the environment on which the facility will be built. This assessment involves air pollution levels, water pollution, water pollution and the impact on the resident population. Both the Health agent and the Environment agent then send the results of their analysis back to the Health agent. These responses together with the project are sent to the Donor agent as a proposal. The Donor checks at the project and decides whether to fund it or not. The diagram below is a modeling of these agent functionalities using sequence diagrams.

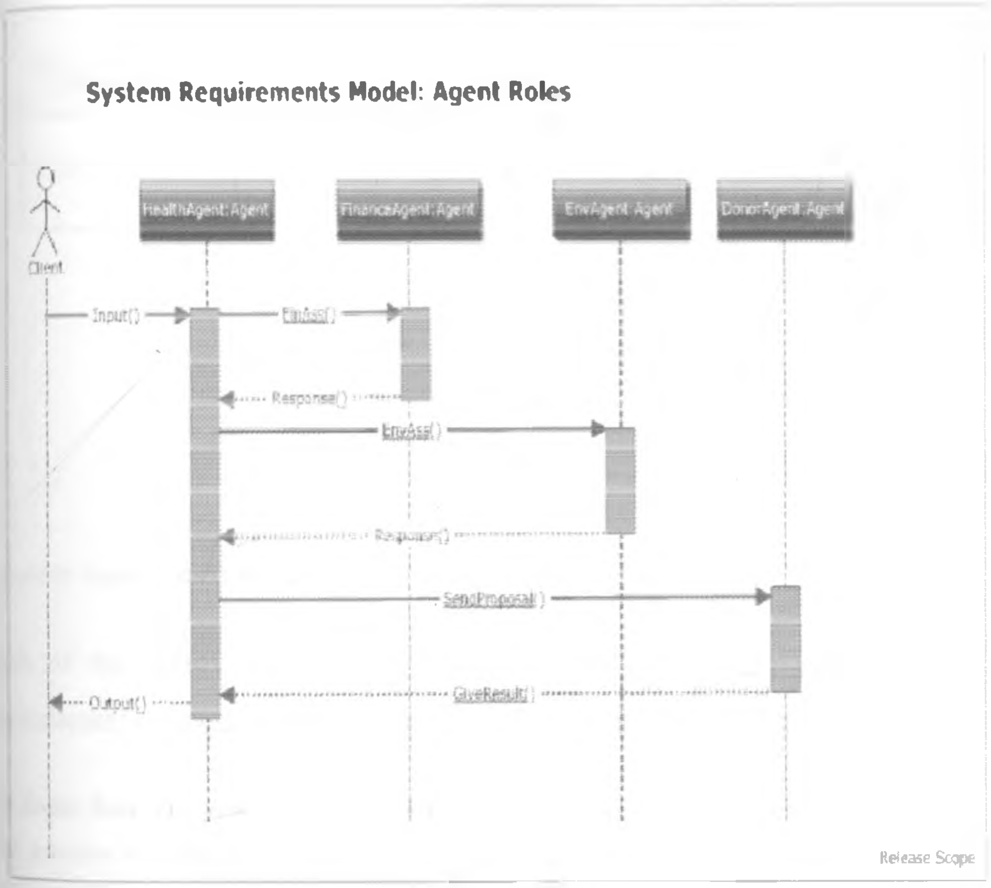


Figure 8: Roles of agents in the model

3.3.2 Agent Society Model

As a society of agents, this multi-agent system is composed of the Health, Finance, Environment and Donor agents. Each of these agents has a specific role to play within the system. The Health agent provides the point of coordination by sending requests to the various concerned agents and receiving responses from these agents. This is to say that communication takes place through the

Health agent. The Environment agent is concerned with the environmental assessment regarding the building of the health facility while the Finance agent is concerned with the financial policies that need to be observed before the project is cleared for execution. The roles assumed by the different agents in this system are represented in the diagram below.

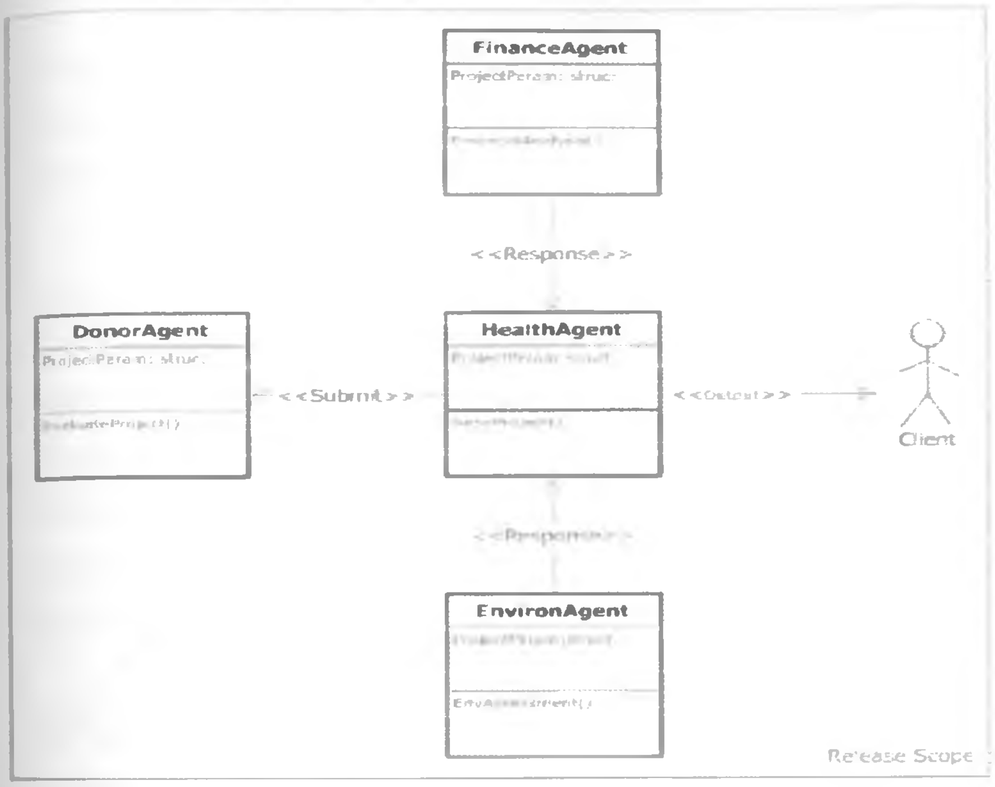


Figure 9: Agent Society Model

Each of the agents therefore has a role to play in the multi-agent system environment. The implementation, code and deployment models are discussed in the next chapter.

3.4 Data Sources and Collection techniques

The source of data for testing the model has been provided through simulation of data for the agents which are representing the ministries. However the simulation is based on simulated scenarios rather than actual data from the ministries.

3.5 Development Tools and Platforms

The framework of development used in developing this model is the Java Agent Development Framework (JADE). JADE is a software framework for developing distributed agent-based applications in compliance with the Foundation for Intelligent Physical Agents (FIPA) specifications for interoperable intelligent multi-agents systems. An application based on JADE is composed of a set of agents implementing the pieces of functionality required by the application. JADE primarily provides agents, their behaviors (tasks to be executed by the agent), transparent distribution of agents across a wide range of devices, peer-to-peer communication between agents and a publish-subscribe mechanisms that allows agents to find each other. JADE also provides a number of additional features such as agent mobility, ontologies, content language support and web services integration.

Apart from JADE, a number of tools have also been used in implementing the various aspects of this model.

3.5.1 Java and Net Beans

The development platform used is Java virtual machine platform with Java Netbeans as the application programming interface. Netbeans provides a rich set of tools that make agent development much easier. The framework of agent development is within the Java Agent Development framework (JADE). The agents have been created as separate entities within a project with each of them able to interact with the other through the various libraries available in the JADE framework.

3.5.2 Java Server Pages

The Java Server Pages (JSP) has been used in building the web portal interface through which input is provided to the model. Input data is fed through a web page after which it is sent to the health agent to the various agents that perform their respective roles with the input data. After the project has been evaluated by the Donor agent, the decision made is communicated to the Health agent which sends it to the web interface through the control agent.

3.4.3 Java Swing Components

A set of graphical tools on the Netbeans development framework has provided the tools used in designing the user interfaces for the database input and other interfaces. The swing components are tied to the database table values. This has made it possible for reading details about a particular project due for submission to the donor for approval (or disapproval) for funding.

3.4.4 Glass Fish Server

This is the server software platform on which remote method invocation has been executed. It ensures that the individual agents are actively communicating with one another and are able to exchange the various data elements in the process of decision making.

3.5 Model Input

The input for this model will is at two different levels. One level is at the point where the parties contributing into the decision making process have to make a decision on what kind of recommendation to give regarding the building of the health facility. At this level, simulated values have been used in providing the inputs particularly for the Environment Ministry Agent and Finance Ministry Agent. Such simulated inputs have provided mean values for the various parameters of interest considered in the decision making process for each of the parties.

Individual agents will query respective knowledge bases in order to make the necessary decisions.

The other level at which inputs are received is at the level where the coordinating agent receives inputs from individual decision makers to the coordination point or to the donor and the responses. This provides a two-way communication so that all agents know whether a decision was made for or against the building of the health facility.

3.6 Model Output

The output from the model is in the form of a decision on whether or not the health facility will be funded. This output depends on how well the entire set of agents cooperates in providing the input needed by the donor agent in making this decision. The output also specifies the project details and a history to enable the status of previous projects submitted for consideration.

3.7 Model Testing

The model has been tested using simulated values. The testing process tests its applicability in the context of its development i.e providing predefined parameter values of a project to the health agent and the health agent sending to the various agents to perform their respective roles on the project parameters provided.

3.8 Model Performance Measures

The success of the model has been measured using two important parameters: Accuracy and Coherence (Wooldridge 2002). Accuracy has been used to determine how accurate the decision made by the donor is after receiving input from the various agents in the system.

Coherence has been used in gauging how well the multi-agent is functioning as a unit. If none of these measures is successfully achieved, then the model may be considered not to have attained the objectives.

CHAPTER FOUR

4.0 MODEL ANALYSIS, DESIGN AND IMPLEMENTATION

4.1 Model Analysis

The analysis of this model has been done by carefully considering what constitutes cooperation among agent entities. To facilitate cooperation among the agents, agents need and must have a means of communication because they will be expected to exchange information among themselves. Communication is significant because it is through communication that agents can exchange information that they have. The challenge in implementing coordinated communication is sometimes in the way in which it is implemented (Salvador, 2008). As long as communication remains uncoordinated in a multi-agent system, the entire system may be derailed in terms of how it is expected to operate. In this study therefore, communication has been implemented using agent communication constructs available in JADE development framework. This is offered through the agent communication language ontology. This method of communication ensures, under the circumstance, that every agent in this system uses a well defined ontology that enhances clarity and coordination in the way communication is carried out among the agents in the system. This is because from the very beginning all the agents involved in decision making are invoked and made to receive the data that they are using in the process of making their individual decisions. This has ensured that the communication among the agents is fully coordinated such that every agent is involved in what is going in the system.

The other issue that arises in a multi agent system is the issue of the system working together as a unit – coherence. This is where the system of agents is able to give the output as a system rather than an individual agent. Coherence in this research project has been addressed by having one output point where the output is given only if the input from all the participating agents have been considered otherwise all the agents are called to the negotiation table once again.

The other challenge that arises in a multi-agent system has to do with agent coordination. Coordinating a group of agents becomes a challenge if there is no particular point in the multi-agent system where the activities of the agents are coordinated. In such cases, it becomes extremely difficult to have the system of agents operating in a coherent manner. When a system of agents has no point of coordination it is like all the agents in that system are operating as individual agents and not as agents in which case the essence therefore of a multi-agent system is lost. According to Wooldrige (2002), this leads to a breakdown in the general pursuit of the objective of such a system. To have coordination therefore in this model, there is a coordinator agent that is sending requests for environmental and financial assessments to be performed and for the proposal to be considered for funding or otherwise. In other words, no agent can communicate to the other without going through the coordinating agent. In this model, the coordinating agent is the health agent which submits a proposal for facility building and also receives results of environment and financial assessments.

4.2 Model design

An intelligent agent operates in an environment within which it perceives changes and takes action that might transform that environment from one state to the other. A number of environment types exist. These include deterministic/stochastic, fully observable/partially observable, static/dynamic and episodic/non episodic.

The environment in which the agents in this model are operating in is dynamic and fully observable. This is because the final decision made depends on the independent analysis carried out by the agents. The outcome of the analysis however cannot be told beforehand hence dynamic. The information the agents are using in making the decision is fully made available to them making that environment fully observable.

4.2.1 Agent Architecture

The architectural design adopted for each of the agents is that of simple reflex agents. In this architecture, the condition-action rules allow the agent to make a connection from percept to action. The percepts in this model are made up of the data about the project that is passed to each agent. Once the agent receives that data, then it performs its role using the data provided and immediately returns the output to the coordinator agent. This architecture is presented in the schematic diagram below.

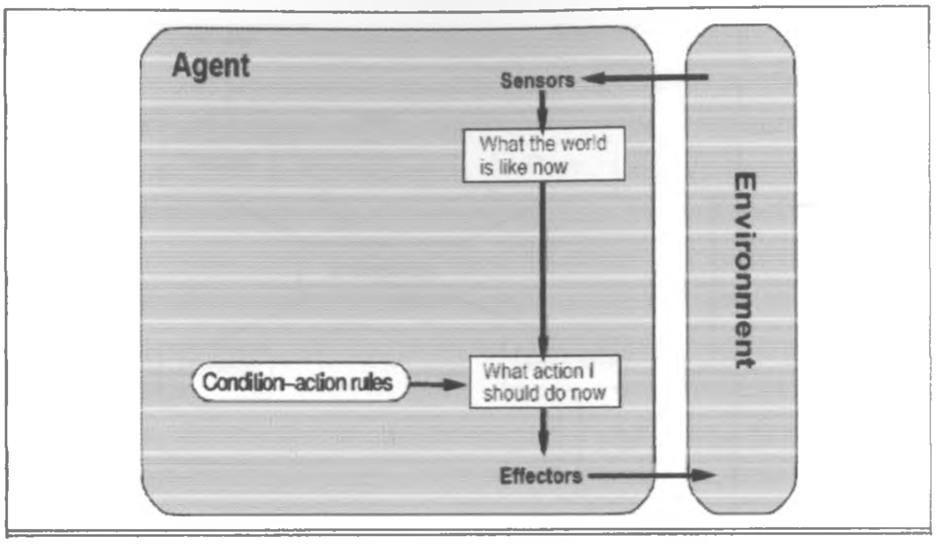


Figure 10: Schematic Diagram of a simple reflex agent *(from Peter N. 1995)*

4.2.2 Belief-Desire-Intention Agent Design

A Belief-Desire-Intention (BDI) agent is able to continuously reason about beliefs, goals and intentions and act accordingly. A BDI model represents both present uncertainties due to limitations in perception and future uncertainties due to dynamism. (Mehdi and Ali, 2007). BDI also distinguishes between success and failure in the execution of events. There are four major concepts in the BDI architecture (Mehdi and Ali, 2007). These are:
Beliefs of an agent – which are the information about the environment. They are subject to uncertainty and error.
Desires – are goals assigned to the agent.
Intentions – are commitments by an agent to achieve particular goals. In other words, they are plans that are currently being executed.
Plans – are choices available to the agent at any moment of time to achieve goals.
In this model, each agent has been designed as a BDI agent. Each one of them has a belief which is stored in its knowledge base. The belief describes what the agent believes in terms its goals in relation to the existing environment. Based on the belief, each agent is pursuing a particular desire, which in this case is tied to its expertise depending in the role it is designed to play in the system. The desire leads to the attainment of the goal of every agent. The intentions of the agents are defined by the type of goals they are pursuing and their plans are the decision choices. The BDI design architecture is shown in the diagram below.

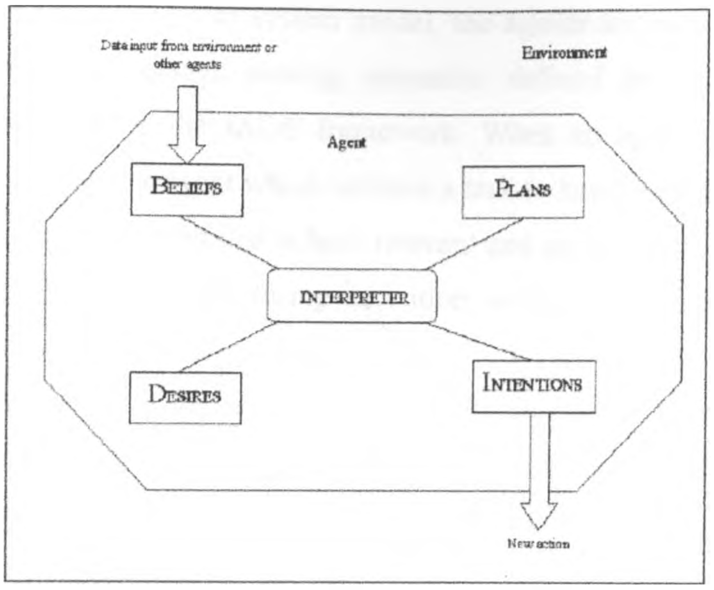


Figure 11: The BDI Agent Architecture *(from Mehdi & Ali 2007)*

So therefore, a BDI agent programming model will be made up of a series logical activities involving the perception of the environment, a change in the belief of the agent which leads to the definition of new goals. In this case, a goal could be to allow the building of a health facility or not. After this an action is then taken, which is sending the decision back to the coordinating agent via a message. This iteration of events is as shown below.

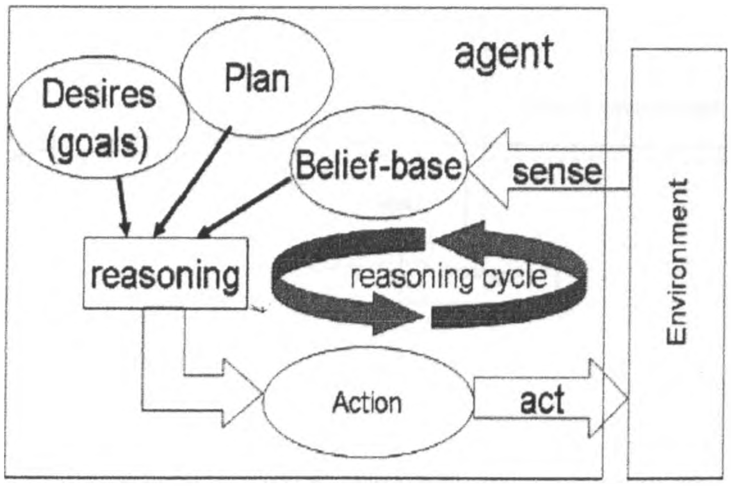


Figure 12: The BDI programming model *(from Yong 2006)*

In the multi-agent system model, the agents are receiving input and sending responses through message passing semantics defined by the agent communication language available in the JADE framework. When an agent receives a message, this message becomes the event which initiates a task to handle a plan. The task involves selecting and executing a plan that is both relevant and applicable to the event. Plans are defined at an abstract level and during execution, as more information becomes available, they are refined to fill out the details.

A change in the agent's beliefs (new observations) under some predefined conditions may cause may cause new goals or desires to be adopted. A new desire consequently causes an appropriate plan to be invoked and executed. This scenario is as represented in the diagram below:

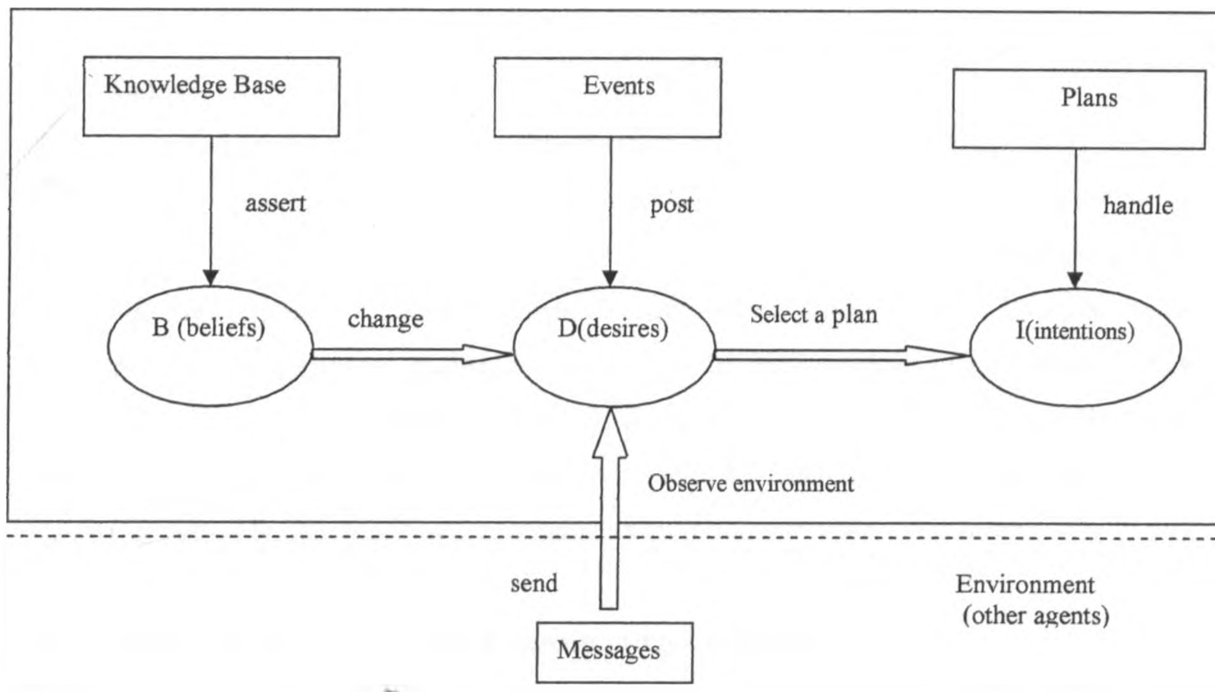


Figure 13: The BDI sequence in a MAS

4.3 Agent Development Platforms and Frameworks

A variety of frameworks and platforms exist for agent development. Each of the frameworks has unique features that enable the programmer in designing and implementing agents. The various frameworks that have been considered in this work include the following.

4.3.1 Java Agent Development Framework (JADE)

JADE is a Java framework for the development of distributed multi-agent applications. It represents an agent middleware providing a set of available and easy-to-use services and several graphical tools for debugging and testing. One of the main objectives of the platform is to support interoperability by strictly adhering to the FIPA specifications concerning the platform architecture. Moreover, JADE is very flexible and can be adapted to be used on devices with limited resources such as PDAs and mobile phones (Costin et al. 2011).

4.3.2 Jadex

Jadex is a software framework for the creation of goal oriented agents following the belief-desire-intention (BDI) model. The framework is realized as a rational agents layer that sits on top of a middleware agent infrastructure and supports agent development with well established technologies such as Java and XML (Bordini et al 2006). The Jadex reasoning engine addresses traditional limitations of BDI systems by introducing new concepts such as explicit goals and goal deliberation mechanisms, making results from goal oriented analysis and design methods. Besides the framework and additional development tools, the distribution contains an introductory tutorial for users.

4.3.3 3APL (An Abstract Agent Programming Language)

This is a programming language and framework for implementing cognitive agents that have beliefs, goals and plans as mental attitudes. The agents can generate and revise their plans to achieve their goals and are able to interact with each other and with the environments they share with other agents. It provides constructs for implementing mental attitudes of agents as well as the deliberation process which manipulates the agents (Dastani et al 2003).

4.3.4 JACK Agent Language

JACK is a commercial agent platform provided by Autonomous Decision Making Software (AOS). Its components include the language, compiler, kernel and a development environment. It supports the development of distributed agent applications by allowing agents to be deployed in separate processes, possibly running on different networked machines. JACK agents are able to exchange messages in a peer-to-peer fashion as well as they are able to find each other using name servers (Costin et al 2011).

4.3.5 2APL

This framework provides a clear separation of multi-agent and individual agent concerns. The multi-agent part addresses the specification of a set of agents, a set of external environments and the relations between them. The individual agent concepts in 2APL cover beliefs, goals, plans, events, messages and rules.

2APL amalgamates declarative and imperative programming styles so it can be also described as a hybrid.

4.4 Implementation Framework

This model has been implemented within the JADE development framework. JADE is a middleware that facilitates the development of multi-agent systems. It provides a runtime environment where JADE agents can exist and the environment must be active on a given host before one or more agents can be executed on that host. JADE also provides a library of classes that can be used in developing the agents and a suite of graphical tools that allows for the administration and monitoring of running agents.

4.4.1 Why JADE

JADE provides a set of features that makes agent development much easier. These features are discussed in the JADE 4.0 programmers' guide and include:

i. Distributed Agent platform which can be split among several hosts with only one java virtual machine executed on each host.
ii. Graphical user interface to manage several agents and agent containers from a remote host.

iii. Debugging tools to help in developing multi-agent applications based on JADE.

iv. Intra-platform agent mobility including transfer of both the state and the code of the agent.

v. Support to the execution of multiple, parallel and concurrent agent activities via the behavior model. JADE schedules the agent behaviors in a non preemptive fashion.

vi. Provides FIPA compliant platform, which includes the Agent Management System (AMS) and the Directory Facilitator (DF). These components have specific roles and are automatically activated at the agent platform start up.

vii. Efficient transport of Agent Communication Language (ACL) messages inside the same agent platform. The messages are sent as encoded java objects.

viii. Library of FIPA interaction protocols ready to be used.

ix. Automatic registration and deregistration of agents with the Agent Management System.

4.4.2 JADE Agent Platform

The standard model of an agent platform, as defined by FIPA, has a number of components as shown in the figure below (www.FIPA.org).

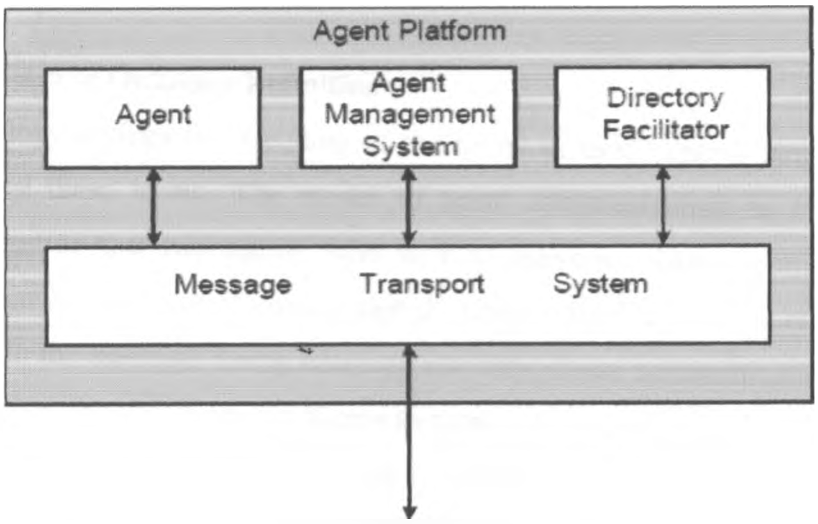


Figure 14: Reference architecture of a FIPA Agent Platform *(adapted from www.FIPA.org)*

The Agent Management System (AMS) is the agent which exerts supervisory control over access to and use of the agent platform. Only one AMS will exist in a single platform. The AMS provides a life-cycle service, maintains a directory of agent identifiers (AID) and agent state. Each agent must register with an AMS in order to get a valid AID.

The Directory Facilitator (DF) is the agent who provides the default yellow page service in the platform, that is, it allows other agents to register their services so that other agents that need those services can get them from them.

The Message Transport System (Agent Communication Channel) is the software component controlling all the exchange of messages within the platform including messages to/from remote platforms.

JADE fully complies with this architecture and when a JADE platform is launched, the AMS and DF are immediately created. Furthermore, the messaging service is always activated to allow message based communication. The agent platform can be easily split into several hosts each of which executing as a Java Virtual Machine (JVM).

Each JVM is a basic container of agents that provides a complete runtime environment for agent execution a d allows several agents to concurrently execute on the same host. The main container is the container where the AMS and the DF live. The other containers, instead, connect for the execution to the main container and provide a complete run-time environment for the execution of any set of JADE agents.

4.4.3 Agent Ontology Definition

Ontology defines the meaning of the terms in used content language and the relation among these terms. The model of agent communication in FIPA is based on the assumption that two agents, who wish to converse, share a common ontology for the domain of discourse. It ensures that the agents ascribe the same meaning to the symbols used in the message. Using ontology not only allows communication between agents but also gives the possibility for agents to reason about the concept.

The ontology implementation used in building this model is defined within the JADE agent system in which ontology elements and its relation and properties are described as real java objects. This is important particularly when implementing agent behavior for manipulating the agents.

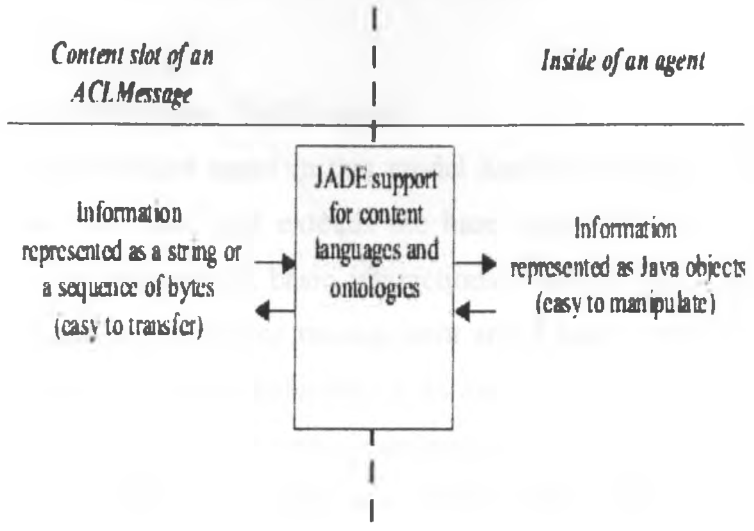

Content slot of an ACLMessage
Inside of an agent
Information represented as a string or a sequence of bytes (easy to transfer)
JADE support for content languages and ontologies
Information represented as Java objects (easy to manipulate)

Figure 15: The conversion performed by the JADE support for content languages and ontologies *(from: JADE tutorial for programmers, www.tilab.com)*

In this implementation, the ontology defined here facilitates the communication between agents. This is enabling them to interpret the terms and concepts in the same way to enhance their cooperation and coordination. Below is a description of this ontology for the agents.

```
public interface EnvironmentVocabulary {
  public static final String ENVIRONMENTAL_ASSESSMENT = "EnvironmentalAssessment";
  public static final String ENVIRONMENTAL_ASSESSMENT_BODY = "body";
  public static final String ENVIRONMENTAL_ASSESSMENT_LAND_POLLUTION = "land_pollution";
  public static final String ENVIRONMENTAL_ASSESSMENT_AIR_POLLUTION = "air_pollution";
  public static final String ENVIRONMENTAL_ASSESSMENT_WATER_POLLUTION = "water_pollution";
  public static final String ENVIRONMENTAL_ASSESSMENT_POPULATION_DISPLACEMENTS =
"displacement";
  public static final String ENVIRONMENTAL_ASSESSMENT_POPULATION_MOVEMENTS = "movement";
  public static final String ENVIRONMENTAL_ASSESSMENT_POPULATION_COMPENSATION =
"compensation";
  public static final String ENVIRONMENTAL_ASSESSMENT_OVERALL_RULES_OBSERVED =
"rules_observed";
  public static final String ENVIRONMENTAL_ASSESSMENT_PROCEED = "proceed";
  public static final String ENVIRONMENTAL_ASSESSMENT_NAME = "name";
  public static final String ENVIRONMENTAL_ASSESSMENT_ATTEMPTS = "attempts";
}

public interface FinanceVocabulary {
  public static final String BUDGETARY_CONSIDERATION = "BudgetaryConsideration";
  public static final String BUDGETARY_CONSIDERATION_AGREEMENT = "agreement";
  public static final String BUDGETARY_CONSIDERATION_PRIORITY = "priority";
  public static final String BUDGETARY_CONSIDERATION_FUNDS_RECEIPIENT = "receipient";
  public static final String BUDGETARY_CONSIDERATION_OVERALL_RULES_OBSERVED =
"rules_observed";
```

```
public static final String BUDGETARY_CONSIDERATION_PROCEED = "proceed";
public static final String BUDGETARY_CONSIDERATION_NAME = "name";
public static final String BUDGETARY_CONSIDERATION_ATTEMPTS = "attempts";
```

4.4.4 Implementing JADE Agent

Each user defined agent in this model has been implemented as an instance of a user defined Java class that extends the base Agent class. This implies the inheritance of features to accomplish basic interactions with the agent platform such as registration, configuration and remote management and a basic set of methods that can be called to implement the custom behavior of the agent like sending and receiving of messages, use of standard interaction protocols and registration with several domains.

The functionality of each agent has been implemented as a behavior. The behaviors are managed by a scheduler which controls when each behavior is executed.

In JADE, an agent environment is implemented as a container. A container can have more than one agent and every agent must be registered in a container for it to be recognized. In this model, all the agents are in the main container. Apart from the major agents constituting the multi-agent system such as the Health, Finance, Donor and Environment agents, the following agents have been created for the management and monitoring of the rest of the agents in the container.

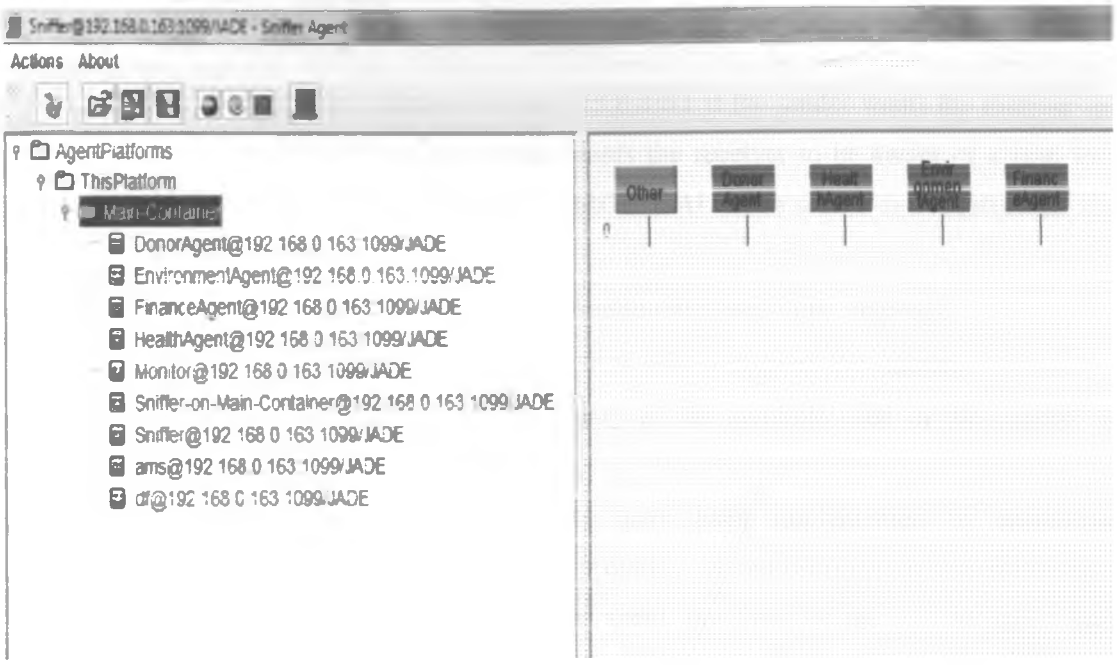
Remote Management Agent (RMA) – acts as a graphical console for platform management and control. The RMA is able to start other JADE tools.

The Sniffer Agent – intercepts agent communication language messages while they are in transit and displays them graphically using a notation similar to Unified Modeling Language (UML) sequence diagrams.

The Agent Management System (AMS)- is the agent which exerts supervisory control over access to and use of the agent platform. Only one AMS will exist in a single platform. The AMS provides a life-cycle service, maintains a directory of agent identifiers (AID) and agent state. Each agent must register with an AMS in order to get a valid AID.

The Directory Facilitator (DF)- is the agent who provides the default yellow page service in the platform, that is, it allows other agents to register their services so that other agents that need those services can get them from them.

The Monitor Agent – which monitors the activities of other agents within the container. The figure below shows how each of the agents is started when the model is executing.



AgentPlatforms
ThisPlatform
Main-Container
DonorAgent@192.168.0.163:1099/JADE
EnvironmentAgent@192.168.0.163:1099/JADE
FinanceAgent@192.168.0.163:1099/JADE
HealthAgent@192.168.0.163:1099/JADE
Monitor@192.168.0.163:1099/JADE
Sniffer-on-Main-Container@192.168.0.163:1099/JADE
Sniffer@192.168.0.163:1099/JADE
ams@192.168.0.163:1099/JADE
df@192.168.0.163:1099/JADE

Figure 16: Agent start up

4.4.5 Agent Communication

In JADE, communication between agents is implemented using the Agent Communication Language (ACL) message construct. The communication paradigm adopted is the asynchronous message passing. Each agent has a message queue where the agent runtime posts messages sent by other agents. Whenever a message is posted in the message queue, the receiving agent is notified. This is shown in the diagram below.

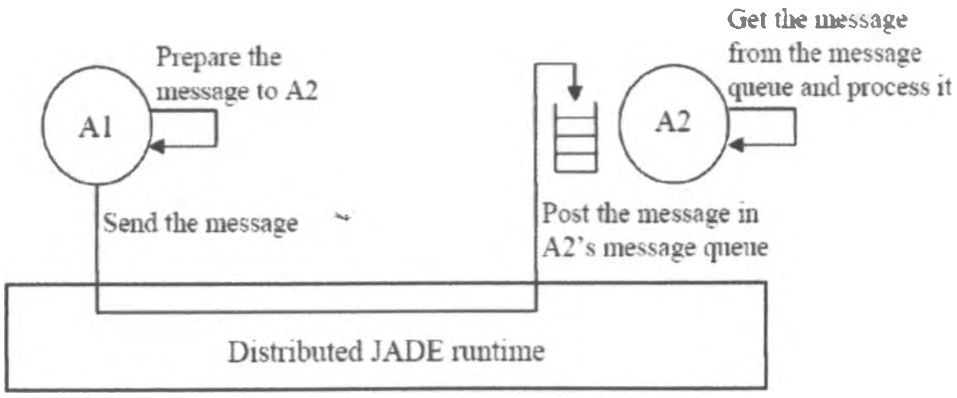

Prepare the message to A2
A1
Send the message
Post the message in A2's message queue
A2
Get the message from the message queue and process it
Distributed JADE runtime

Figure 17: The JADE asynchronous message passing paradigm *(from: JADE Beginners tutorial, www.tilab.com)*

Messages exchanged by JADE agents have a format specified by the ACL language. This format comprises a number of fields such as (www.FIPA.org):

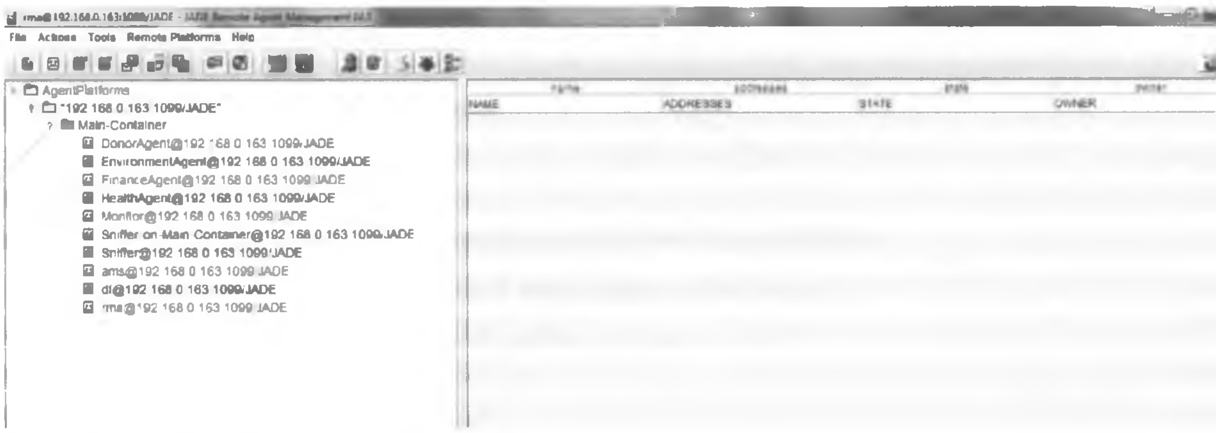
- The sender of the message
- The list of receivers
- The communicative intention indicating what the sender intends to achieve by sending the message. The intention can be REQUEST if the sender wants the receiver to perform an action, INFORM if the sender wants the receiver to be aware of a fact, PROPOSE, ACCEPT_PROPOSAL, REJECT_PROPOSAL if the sender and receiver are engaged in a negotiation.
- The content which is the actual information included in the message
- The content language which is the syntax used to express the content.
- The ontology which is the vocabulary of the symbols used in the content and their meaning.

In this model, messages between agents have been implemented by instantiating the *jade.lang.acl.ACLMessage* java class which provides get and set methods which handle all fields of a message. To send a message to another agent, the fields of the ACLMessage object have been field with the right arguments and the send () method of the Agent class called to do the sending.

CHAPTER FIVE

5.0 MODEL TESTING AND DISCUSSION OF RESULTS

When the model is executed, the *Remote Management Agent (RMA)* starts a graphical user interface with components for executing the agent environment. The Remote Management Agent (RMA), the Directory Facilitator (DF) agent, the Sniffing agent and the Agent Management System agents are all created as soon as the interface starts to execute. In addition the Health, Donor, Environment and Environment agents instantiated and started. All these agents are created and executed in the same container, which provides the execution environment. This is as shown in the figure below.



```
AgentPlatforms
  "192 168 0 163 1099/JADE"
    Main-Container
      DonorAgent@192 168 0 163 1099/JADE
      EnvironmentAgent@192 168 0 163 1099/JADE
      FinanceAgent@192 168 0 163 1099 JADE
      HealthAgent@192 168 0 163 1099/JADE
      Monitor@192 168 0 163 1099 JADE
      Sniffer on Main Container@192 168 0 163 1099/JADE
      Sniffer@192 168 0 163 1099/JADE
      ams@192 168 0 163 1099 JADE
      df@192 168 0 163 1099/JADE
      rma@192 168 0 163 1099 JADE
```

Figure 18: Runtime screen showing RAM interface

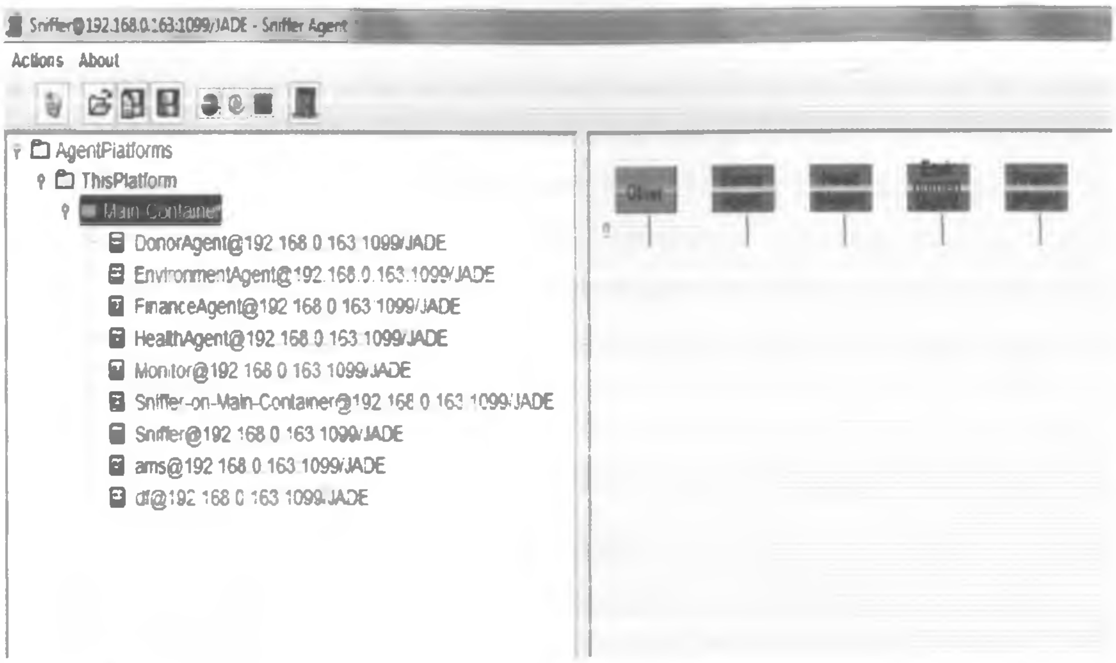
5.1 Testing of the Model

Testing of the model has been performed using simulated values as input for each of the agents in the model. The Environment agent has simulated values supplied to it to represent a scenario where there are a number of environmental assessment parameters considered and whose values have been used as input while the Finance agent also has values simulated for use as input.

Simulated inputs have been used to capture what would happen in a real life scenario in cases of both environmental and financial assessments.

The main input to the model is a project with parameters which are analyzed by both Environment and the Finance agents before sending their respective recommendations to the

Health agent which then sends the respective recommendations to the Donor agent for possible approval for funding.



```
AgentPlatforms
  ThisPlatform
    Main Container
      DonorAgent@192 168 0 163 1099/JADE
      EnvironmentAgent@192.168.0.163.1099/JADE
      FinanceAgent@192 168 0 163 1099/JADE
      HealthAgent@192 168 0 163 1099/JADE
      Monitor@192 168 0 163 1099/JADE
      Sniffer-on-Main-Container@192 168 0 163 1099/JADE
      Sniffer@192 168 0 163 1099/JADE
      ams@192 168 0 163 1099/JADE
      df@192 168 0 163 1099/JADE
```

Figure 19: Active agents created

5.2 Project Parameters as Input

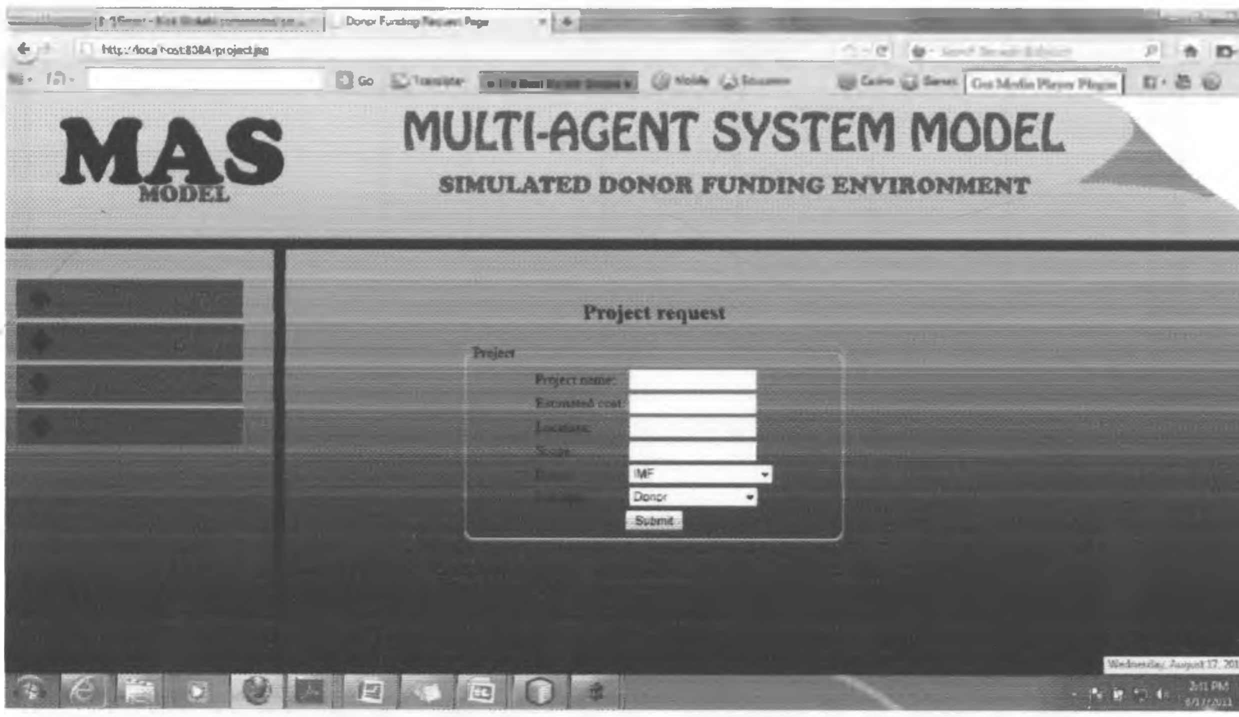
A typical project for building a health facility has the following parameters:

- *Name of the project* – this is what is used to refer to the project
- *Project Cost* – this is how much the project is going to cost
- *Project Location* – is a description of the physical location of the project. A location has, associated with it, air pollution levels, water pollution, land pollution, population displacements and overall effects on the environment. These aspects are analyzed by the Environment agent, and based on the outcome of this analysis, the agent provides a recommendation for or against the building of the health facility.
- *Project Funding* – this is how the project is going to be funded. The funding can be done either fully by a donor or it can be a cost sharing scenario where the cost is being split between the donor and the government.
- *Donor* – is the source of the funds. The Donor agent finds out whether there are trade agreements existing between the donor states and the country in which the project is being funded. This, and other factors, such as whether budgetary priorities are well placed influence the decision given by the Finance agent regarding the building of the facility.

- *Scope* – describes the level of the project. A project may be a national, regional or local project.

5.3 Results

A user interface designed using the Java Server Pages (JSP) has been provided to capture the project parameters provided as input. The input is first sent to the Health agent. The Health agent sends the input as a proposal to the Donor agent and to both the Environment agent and Health agent to carry out respective analysis. The Health agent therefore receives details of a particular project and sends those details to the Environment agent and Finance agent for environmental assessment and financial assessment respectively. Below is a runtime screen capturing the details of a particular project.



Project

Project name:
Estimated cost:
Location:
Scope:
[illegible]: IMF
[illegible]: Donor

Figure 20: Project input capture screen

After the project details have been captured, the details are sent to the other two agents by the Health agent for assessments. Each of the agents do private assessments of the suitability of the project being funded by the donor based on the conditions specified. The agents then return the results of their assessments to the health agent which then submits the proposal to the donor agent together with the recommendations of the assessing agents. This collective information is what the donor agent uses to determine whether to fund the facility or not. The output below shows, project input sent to the Finance, Environment and Donor agents.

The Health and the Finance agents then send feed back to the Health agent regarding the analysis they have carried out. The Health agent forwards this feedback to the Donor gent. The Donor agent finally sends back a decision on whether the funding of the project allowed or not. This is shown below.

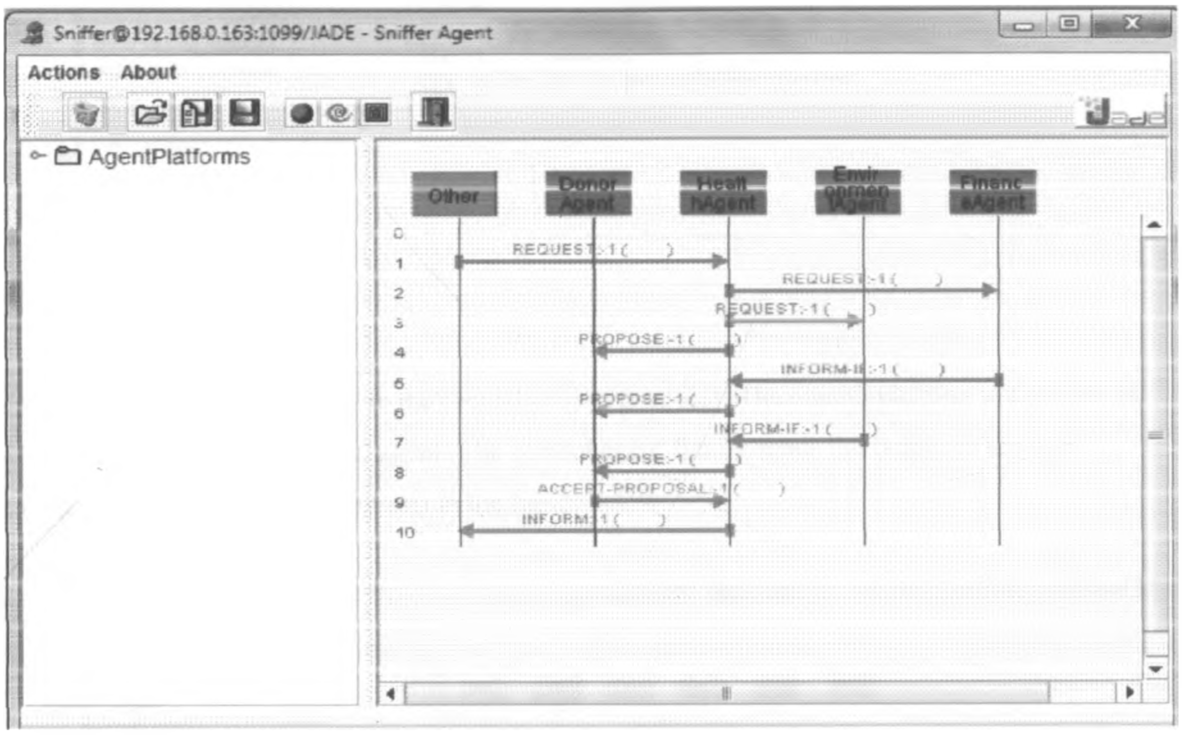
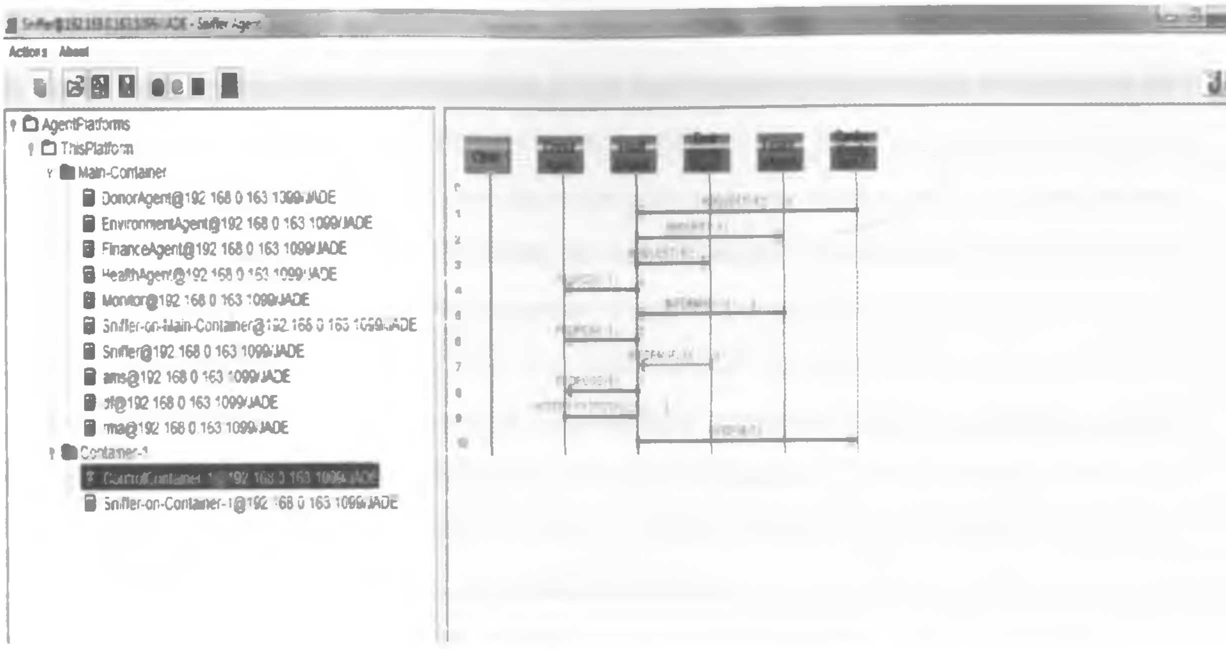


Figure 21: Project proposal and donor response

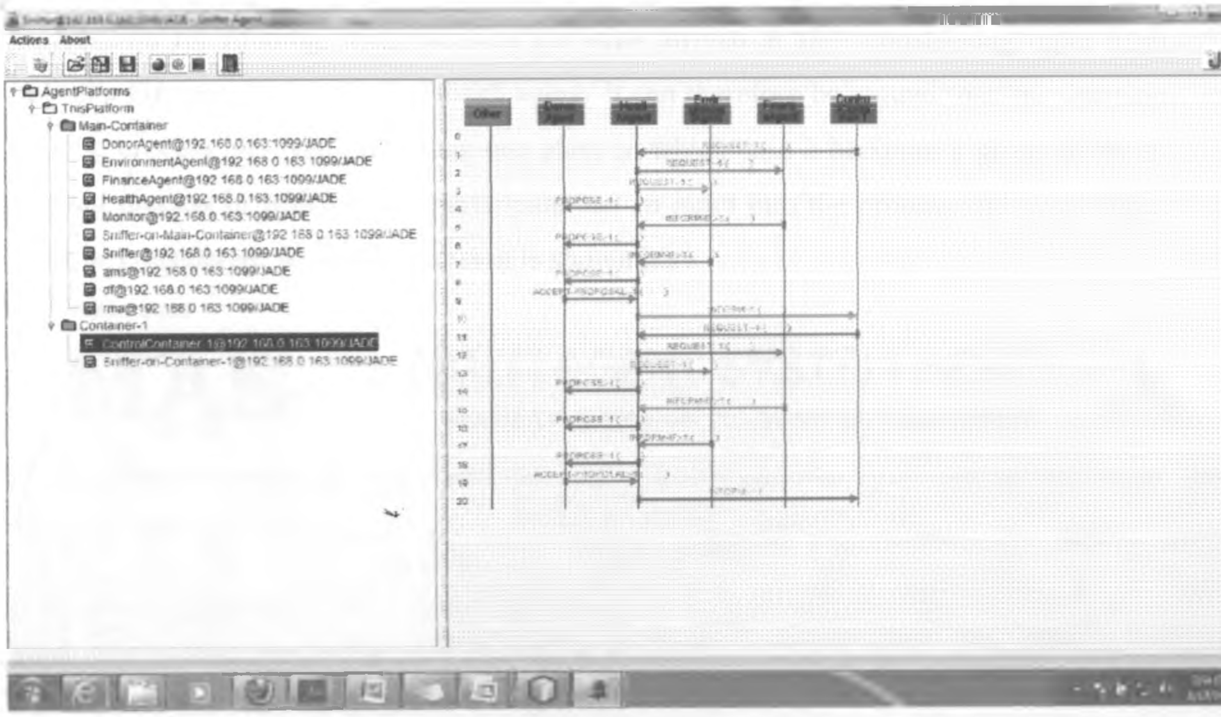
Initially, when the input is read, the control agent has not been sniffed and so as much as there is communication among the agents, it is not shown on the interface. When it is sniffed however (figure below), it is then shown to have been the one coordinating the capturing of input and the communication of the decision to the user on the interface.



AgentPlatforms
ThisPlatform
Main-Container
DonorAgent@192.168.0.163:1099/JADE
EnvironmentAgent@192.168.0.163:1099/JADE
FinanceAgent@192.168.0.163:1099/JADE
HealthAgent@192.168.0.163:1099/JADE
Monitor@192.168.0.163:1099/JADE
Sniffer-on-Main-Container@192.168.0.163:1099/JADE
Sniffer@192.168.0.163:1099/JADE
ams@192.168.0.163:1099/JADE
df@192.168.0.163:1099/JADE
rma@192.168.0.163:1099/JADE
Container-1
ControlContainer-1@192.168.0.163:1099/JADE
Sniffer-on-Container-1@192.168.0.163:1099/JADE

Figure 22: Control agent

This communication continues in the second project proposal. The communication in this case is fully coordinated by the control agent. The control agent finally sends the output to the user through the interface. This is shown in the figure below.



AgentPlatforms
ThisPlatform
Main-Container
DonorAgent@192.168.0.163:1099/JADE
EnvironmentAgent@192.168.0.163:1099/JADE
FinanceAgent@192.168.0.163:1099/JADE
HealthAgent@192.168.0.163:1099/JADE
Monitor@192.168.0.163:1099/JADE
Sniffer-on-Main-Container@192.168.0.163:1099/JADE
Sniffer@192.168.0.163:1099/JADE
ams@192.168.0.163:1099/JADE
df@192.168.0.163:1099/JADE
rma@192.168.0.163:1099/JADE
Container-1
ControlContainer-1@192.168.0.163:1099/JADE
Sniffer-on-Container-1@192.168.0.163:1099/JADE

Figure 23: Subsequent project proposal and response

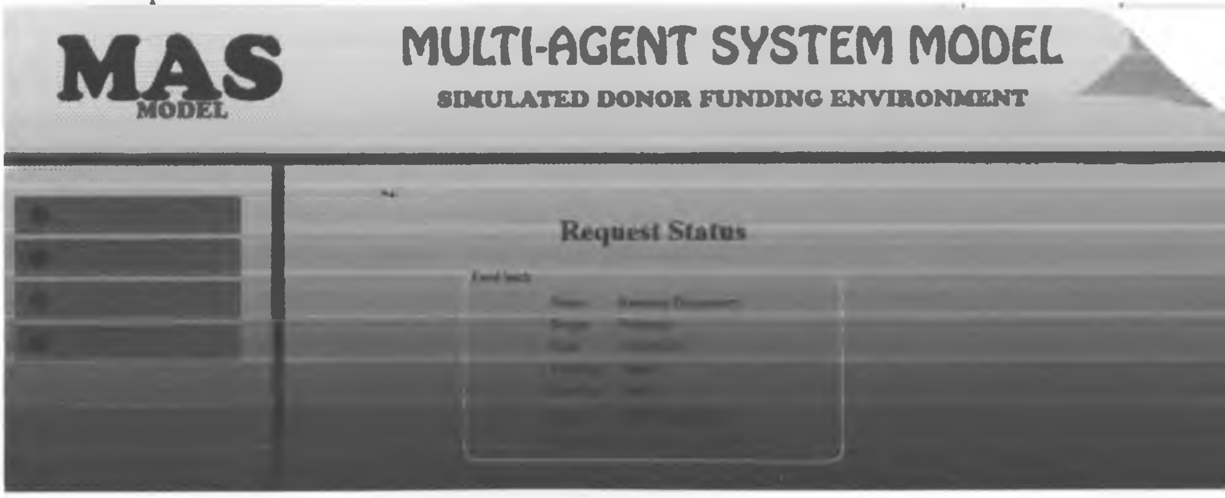
5.4 Discussion of Results

As has been shown, there is cooperation in the multi-agent system model necessitated by the need to exchange information particularly between the Health agent and the other agents. This cooperation has taken place through message exchange as defined in the ACL message ontology. It can therefore be said that agent cooperation particularly in a multi-agent system environment is important if agents are going to work together. This is irrespective of whether the agents are benevolent or self interested. It is important to note that in this model, if one or more of the agents fails to provide input towards the overall objective of the system, then a decision will never be reached. This is because the donor agent needs input from all the agents before it makes a decision on whether to fund the health facility or not. This is what cooperation is about.

Agent coordination has also been shown to be important if the actions of individual agents are to contribute to the general objective of the multi-agent system. In this case, it has been shown that coordination can be used in bringing together the various agents under one agent through which all communication and exchanges are allowed to take place. In this case, this role has been played by the Health agent.

The output from the model is a decision which can either be an approval for funding or disapproval for funding. This decision has been arrived at after considering input from both the Finance agent and the Health agent. It can therefore be argued that this decision making process involved more than one party in which case it becomes a group decision making process in which every participant has an input into the process and every input must be considered. A sample decision is shown here.

Figure 24: Model output



CHAPTER SIX

6.0 CONCLUSIONS AND RECOMMENDATION

In conclusion, the constructed model has demonstrated how cooperation and coordination in a group decision making environment in a multi-agent system environment takes place. It has been shown that agent coordination provides a way of managing the interdependencies that exist among agents in a multi-agent system. Successful management of these dependencies allows such a system to function as a unit thereby manifesting the concept of coherence. When agents in a multi-agent system are functioning as a unit, then it becomes easier to achieve the objective of such a system as a whole rather than each individual agent pulling in its own direction. Coordination therefore, it can be argued, enhances the functioning of a system of agents as a unit and this unity enables the system to perform better than individual agents would in the same circumstances. Agent cooperation in this model has been shown to majorly facilitate exchange of information among existing agents in a multi-agent system. In a multi-agent system, cooperation is key in enabling agents to communicate intentions and exchange the data that they need to enable each of them to pursue their individual objectives yet still contribute to the general objective of the entire system For example in this model the Health agent cooperates with the Donor, Finance and Environment agents to have the funds released for the building of a health facility. Without the cooperation of all the agents it would be difficult for the donor agent to make a decision.

The tests conducted on the model reveal the dynamics of cooperation and coordination through the communication taking place among the agents.

The analysis of the results shows that problem solution in a multi-agent system is optimized when agents cooperate via communication and work together in a coordinated manner.

The contribution of this model is in the fact that through the exploitation of agent coordination and cooperation, solutions can be found in decision making scenarios where there are many parties whose input need to be factored into the decision making process in order to arrive at a final decision that will lead to the solution of a problem.

As a proof of concept, it has been shown here, through model construction and execution, that this model can work in an environment where a donor needs to fund the building of a health facility. But before this can happen, the donor must receive recommendations regarding environmental assessment and budgetary assessment from the Environment agent and the Finance agent respectively. There is cooperation and coordination leading to a solution in this case.

Further work however can be done in testing this model in its scalability and application in different environments. In this model, only four agents have been used. This means that coordination and cooperation has been done with agents that are not so many. It would be interesting to see how the model behaves in cases where the decision makers are more than four. Also this model has been tested in an environment where a donor is funding a health facility. It can be tested in different environments, say for example, in a military recruitment to see whether it will be applicable outside the health domain or not.

CHAPTER SEVEN

7.0 REFERENCES

1. Abdelkader A. (2007). *A Distributed Architecture for Cooperative Intelligent Decision Support Systems*, in, *IEEE Multidisciplinary Engineering Education Magazine*, VOL. 2, NO. 2, JUNE 2007.
2. 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.
3. Axtell R. (2000). Why Agents*? On the Varied Motivations for Agent Computing in the Social Sciences*, in, *Center on Social and Economic Dynamics*, Working Paper No. 17, November 2000.
4. Changhong, L., Minqiang, L., & Jisong, K.; (2002). *Cooperation Structure of Multi-agent and Algorithms*, *Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems*, pp. 303–307, ISBN:0-7695-1733-1, September 2002, IEEE, Divnomorskoe, Russia.
5. Eunyoung K., Hee Y. and Ungmo K. (2008). *Mining Based Decision Support Multi-agent System for Personalized e-Healthcare Service*, KES-AMSTA 2008, LNAI 4953, pp. 733–742, 2008.
6. Far T., Wanyama T. and Soueina S. O., (2006). *A Negotiation Model for Large Scale Multi-Agent Systems*, in proc. of, *The IEEE International Conference on Information Re-use and Integration*, pp. 589-594.
7. Ferber J., (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley.
8. Harrison R., Yuaxing W., Nguyen H., Xiongmin L., (2007). *A Decision Support System for Filtering and Analysis of Carbon Dioxide Capture Data*. In Proc. of the *Canadian Conference on Electrical and Computer Engineering*, pp 1380-1383.
9. Howard N. (2009). *Resolving conflicts in a tree. Drama Theory in the Extensive form*, vol 2 pp 6
10. Ibarra M. S., (2008). *Physical Multi Agent Systems: A new Theory aimed at Coordination*, in proc of, *The 2nd Informatics Spanish Congress*, ISBN: 978-84-9732-597-4, vol. 1, pp. 97-104.
11. Liping S., (2005). *Decision Support Systems Based on Knowledge Management*. In Proc. of the International Conference on Services Systems and Services Management, vol 2, pp. 1153-1156.

12. Luck M., McBurney P. and Preist C., (2003). *Agent Technology: Enabling Next Generation Computing,* in *A Road Map for Agent Based Computing,* ISBN 0854327886 ver. 1.0, Southampton: AgeLink.
13. Luck M., McBurney P., Shehory O., and Willmott S., (2005). *Agent Technology: Computing as Interaction,* in *A Road Map for Agent Based Computing,* compiled, written and edited by Luck M., McBurney P., Shehory O., Willmott S. and the AgentLink Community, pp. 11-12.
14. M. Dastani, F. de Boer, F. Dignum, and J.-J. Meyer. *Programming agent deliberation: An approach illustrated using the 3APL language.* In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS' 03)*, pages 97–104. ACM, 2003.
15. Marakas G. M. (2004). *Decision Support Systems in the 21st Century, 2nd Edition*, Prentice Hall, 2004.
16. Marko B. (2001): *What is Decision Support System*, Prentice-Hall, 2001.
17. Mas A., (2005). *Agent Software and Multi Agent Systems: Concepts, Architectures and Applications*. ISBN: 84-205-4367-5, Pearson Education, S.A., Madrid Spain.
18. Parker L. E., (2008). *Distributed Intelligence: Overview of the field and its application in Multi-Robot Systems*, in, *Journal of Physical Agents*, vol. 2, issue 1, pp. 5-14.
19. Parsons S. & Wooldridge M. (2000). *Game Theory and Decision Theory in Multi-Agent Systems*, Kluwer Academic Publishers, Netherlands.
20. Power D. J. (2009). *Supporting Decision-Makers: An Expanded Framework*, Available: http://dssresources.com, 2009.
21. Power D. J., (2002). *Web Based and Model Driven Decision Support Systems: Concepts and Issues.* In Proc. of the Americas Conference on Information Systems, Long Beach, California-USA.
22. Quintero C. G., (2007). "*Introspection on Control grounded Capabilities: An Agent Inspired Approach for Control.* In proc. of the European Control Conference, vol 1, pp. 35-42".
23. Rao A. S. and Georgeff M. P. (1995). *BDI Agents from Theory to Practice*, in proc. of the *First International Conference on Multi Agent Systems*, ICMAS95.
24. Salvador, I. M.,(2008). *A Formalization for Multi-Agent Decision Support in Cooperative Environments: A Framework for Situated Agents.* Girona, Catalina, Spain.
25. Sarjono U., Manahan S. & Novani S. (2007). *Agent Based Simulation of Negotiation Process using Drama Theory*, vol 4, pp 12.
26. Sascha O., Josė C. and Ana G. (1998). *Using Autonomous Agents for Urban Traffic Control*, in, IBERAMIA'98, LNAI 1484, pp. 100-111, 1998.
27. Sascha O., Jose O. and Ana G. (1998). *A Case of Multiagent Decision Support:*

28. Simon A. H., (2007). *The New Science of Management Decision*, Prentice-Hall, 2007.
29. Stone P. and Veloso M., (2000) *Multi Agent Systems: A Survey from a Machine Learning Perspective*, in, *Autonomous Robots*, vol 8, no. 3., pp. 34-383.
30. Sycara, K.(1998). *Multiagent systems. AI Magazine* 10(2), 79–93 (1998)
31. *The Java Agent Development Framework Release*, www.jade.tilab.com – visited on 20 March 2011.
32. Turban E. and Aronson J., (2001). *Decision support Systems and Intelligent Systems*, Prentice-Hall International, Upper Saddle River, New Jersey, 2001.
33. Wanng D., Goldenberg A. & Liu G. (2007). *Development of Control System Architecture for Modular and Reconfigurable Robot Manipulators*, in proc of, *The International Conference on Mechatronics and Automation*, pp. 20-25.
34. Wooldridge M, (2002). *An Introduction to Multi-Agent Systems.* Published in Feb. 2002 by John Wiley and Sons in Chichester England.
35. Yong S. and Bo W., (2006). *Agent Hybrid Architecture and its Decision Processes*, in proc of, *The International Conference on Machine Learning and Cybernetics*, pp 641-644.
36. Yuan L., Darryl N. D. and Kedung L. (2008). *A Multi-Agent System Framework for Decision Support in Stock Trading*, in proc. of the *36th Hawaii Int. Conf. System Sciences, IEEE*, 2008.
37. Zoltàn B., Michal L. and Ladislav H. (2008) . *Multi Agent System for Negotiation and Decision Support*, in, *The International Arab Journal of Information Technology*, vol. 3, No. 2 pp. 56-62.

APPENDICES

Appendix 1A

Sample Code

<u>Creating the Donor Agent</u>

```
package donor;

import java.util.Hashtable;
import java.util.logging.Level;
import java.util.logging.Logger;
import resources.AbstractAgent;
import resources.behaviours.ReceiveProposals;

/**
 *
 * @author Oriedi
 */
public class DonorAgent extends AbstractAgent{
   Hashtable data = new Hashtable();
   @Override
   protected void setup() {
      register("Donor");
      addBehaviour(new ReceiveProposals(this,data));

//       while(true){
//       try {
//          Thread.sleep(1000);
//       } catch (InterruptedException ex) {
//          Logger.getLogger(DonorAgent.class.getName()).log(Level.SEVERE, null, ex);
//       }
//       System.out.println(data);
//   }
   }
```

Creating the Donor Agent Behaviour

```
package donor;

import jade.content.AgentAction;

/**
 *
 * @author Oriedi
 */
public class DonorRequest implements AgentAction{
   private String cost;
   private String funding;
   private String location;
   private String scope;
   private String name;
   private String status;
   private String country;
   private String reason;
   private String attempts;

   public String getAttempts() {
      return attempts;
   }

   public void setAttempts(String attempts) {
      this.attempts = attempts;
   }


   public String getReason() {
      return reason;
   }

   public void setReason(String reason) {
      this.reason = reason;
   }


   public String getCountry() {
      return country;
   }

   public void setCountry(String country) {
      this.country = country;
   }


   public String getStatus() {
      return status;
```

```
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getCost() {
        return cost;
    }

    public void setCost(String cost) {
        this.cost = cost;
    }

    public String getFunding() {
        return funding;
    }

    public void setFunding(String funding) {
        this.funding = funding;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getScope() {
        return scope;
    }

    public void setScope(String scope) {
        this.scope = scope;
    }

}
```

Creating the Environment Agent

```
package environment;

import java.util.Hashtable;
import resources.AbstractAgent;
import resources.behaviours.ReceiveRequests;

/**
 *
 * @author oriedi
 */
public class EnvironmentAgent extends AbstractAgent {

   @Override
   protected void setup() {
      register("Environment");
      addBehaviour(new ReceiveRequests(this));

   }}
```

Creating the Environment Agent

```
package environment;

import jade.content.AgentAction;

/**
 *
 * @author oriedi
 */
public class EnvironmentalAssessment implements AgentAction{
   private String body;
   private String land_pollution;
   private String air_pollution;
   private String water_pollution;
   private String displacement;
   private String movement;
   private String compensation;
   private String rules_observed;
   private String proceed;
   private String name;
   private String attempts;

   public String getAttempts() {
      return attempts;
   }
```

```
public void setAttempts(String attempts) {
   this.attempts = attempts;
}


public String getName() {
   return name;
}

public void setName(String name) {
   this.name = name;
}



public String getAir_pollution() {
   return air_pollution;
}

public void setAir_pollution(String air_pollution) {
   this.air_pollution = air_pollution;
}

public String getBody() {
   return body;
}

public void setBody(String body) {
   this.body = body;
}

public String getCompensation() {
   return compensation;
}

public void setCompensation(String compensation) {
   this.compensation = compensation;
}

public String getDisplacement() {
   return displacement;
}

public void setDisplacement(String displacement) {
   this.displacement = displacement;
}

public String getLand_pollution() {
   return land_pollution;
}

public void setLand_pollution(String land_pollution) {
        this.land_pollution = land_pollution;
    }

    public String getMovement() {
        return movement;
    }

    public void setMovement(String movement) {
        this.movement = movement;
    }

    public String getProceed() {
        return proceed;
    }

    public void setProceed(String proceed) {
        this.proceed = proceed;
    }

    public String getRules_observed() {
        return rules_observed;
    }

    public void setRules_observed(String rules_observed) {
        this.rules_observed = rules_observed;
    }

    public String getWater_pollution() {
        return water_pollution;
    }

    public void setWater_pollution(String water_pollution) {
        this.water_pollution = water_pollution;
    }


}
```

Creating the Finance Agent

```
package finance;

import resources.AbstractAgent;
import resources.behaviours.ReceiveRequests;

/**
 *
 * @author oriedi
 */
public class FinanceAgent extends AbstractAgent {

    @Override
    protected void setup() {
      register("Finance");
      addBehaviour(new ReceiveRequests(this));  }}
```
Finance Agent Behaviour

```
package finance;

import jade.content.AgentAction;

/**
 *
 * @author oriedi
 */
public class BudgetaryConsideration implements AgentAction{

   private String agreement;
   private String rules_observed;
   private String receipient;
   private String priority;
   private String proceed;
   private String name;
   private String attempts;

   public String getAttempts() {
      return attempts;
   }

   public void setAttempts(String attempts) {
      this.attempts = attempts;
   }


   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }


   public String getAgreement() {
      return agreement;
   }

   public void setAgreement(String agreement) {
      this.agreement = agreement;
   }

   public String getPriority() {
      return priority;
   }
```

```
    public void setPriority(String priority) {
        this.priority = priority;
    }

    public String getProceed() {
        return proceed;
    }

    public void setProceed(String proceed) {
        this.proceed = proceed;
    }

    public String getReceipient() {
        return receipient;
    }

    public void setReceipient(String receipient) {
        this.receipient = receipient;
    }

    public String getRules_observed() {
        return rules_observed;
    }

    public void setRules_observed(String rules_observed) {
        this.rules_observed = rules_observed;
    }

}
```

Creating the Health Agent and its Behaviour

```
package health;

import donor.DonorRequest;
import environment.EnvironmentalAssessment;
import finance.BudgetaryConsideration;
import jade.content.Concept;
import jade.content.ContentElement;
import jade.content.lang.Codec.CodecException;
import jade.content.onto.OntologyException;
import jade.content.onto.UngroundedException;
import jade.content.onto.basic.Action;
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import java.util.Hashtable;
import java.util.logging.Level;
import java.util.logging.Logger;
import resources.AbstractAgent;
import resources.MyDelay;
import resources.behaviours.PersistData;
import resources.behaviours.ReceiveDonorReply;
import resources.behaviours.ReceiveInformation;
import resources.behaviours.ReceiveRequests;
import resources.behaviours.SendMessage;

/**
 *
 * @author oriedi
 */
public class HealthAgent extends AbstractAgent {

   @Override
   protected void setup() {
      register("Health");
      addBehaviour(new ReceiveInformation(this));
      addBehaviour(new ReceiveDonorReply(this));
      addBehaviour(new ReceiveRequests(this));
      Hashtable data = new Hashtable();
      addBehaviour(new HandleAnalysisInfo(this, data));
   }
}

class HandleAnalysisInfo extends CyclicBehaviour {

   Hashtable data;

   public HandleAnalysisInfo(Agent a, Hashtable data) {
        super(a);
        this.data = data;
    }

    @Override
    public void action() {
        try {
            ACLMessage received =
myAgent.receive(MessageTemplate.MatchPerformative(ACLMessage.INFORM_IF));

            if (null == received) {
                block();
                return;
            }
            ContentElement messageContent = myAgent.getContentManager().extractContent(received);
            Concept action = ((Action) messageContent).getAction();

            if (action instanceof EnvironmentalAssessment) {
                EnvironmentalAssessment env = (EnvironmentalAssessment) ((Action)
messageContent).getAction();
//                System.out.println(received.getSender().getLocalName() + " " + env.getProceed() + " " +
env.getAir_pollution() + " " + env.getLand_pollution()
//                        + " " + env.getWater_pollution() + " " + env.getBody() + " " + env.getMovement() + " "
+ env.getCompensation());
                MyDelay.delay(5);
                if (env.getProceed().startsWith("NO") && Integer.parseInt(env.getAttempts()) < 2) {

                    DonorRequest dr = new DonorRequest();
                    dr.setAttempts("1");
                    dr.setCost("000");
                    dr.setCountry(env.getRules_observed());
                    dr.setFunding("");
                    dr.setLocation(env.getName());
                    dr.setName(env.getName());
                    dr.setReason("");
                    dr.setScope("");
                    dr.setStatus("");

                    myAgent.addBehaviour(new SendMessage(myAgent, "FinanceAgent",
ACLMessage.REQUEST, dr));
                } else {
                    myAgent.addBehaviour(new PersistData(myAgent, env));
                    myAgent.addBehaviour(new SendMessage(myAgent, "DonorAgent",
ACLMessage.PROPOSE, env));

                }
            } else if (action instanceof BudgetaryConsideration) {
                BudgetaryConsideration bgt = (BudgetaryConsideration) ((Action)
messageContent).getAction();
//              System.out.println(received.getSender().getLocalName() + " " + bgt.getProceed());
                MyDelay.delay(5);
                if (bgt.getProceed().startsWith("NO") && Integer.parseInt(bgt.getAttempts()) < 2) {
```

```
                DonorRequest dr = new DonorRequest();
                dr.setAttempts("1");
                dr.setCost("000");
                dr.setCountry(bgt.getRules_observed());
                dr.setFunding("");
                dr.setLocation(bgt.getName());
                dr.setName(bgt.getName());
                dr.setReason("");
                dr.setScope("");
                dr.setStatus("");

                myAgent.addBehaviour(new SendMessage(myAgent, "FinanceAgent",
ACLMessage.REQUEST, dr));
            } else {
                myAgent.addBehaviour(new PersistData(myAgent, bgt));
                myAgent.addBehaviour(new SendMessage(myAgent, "DonorAgent",
ACLMessage.PROPOSE, bgt));
            }
        }



    } catch (CodecException ex) {
        Logger.getLogger(HandleAnalysisInfo.class.getName()).log(Level.SEVERE, null, ex);
    } catch (UngroundedException ex) {
        Logger.getLogger(HandleAnalysisInfo.class.getName()).log(Level.SEVERE, null, ex);
    } catch (OntologyException ex) {
        Logger.getLogger(HandleAnalysisInfo.class.getName()).log(Level.SEVERE, null, ex);
    }

  }
}
```