THE UNIVERSITY OF NAIROBI

COLLEGE OF BIOLOGICAL AND PHYSICAL SCIENCES

SCHOOL OF COMPUTING AND INFORMATICS

A LOGISTICS PLANNING SYSTEM BASED ON BELIEF-DESIRE-INTENTION AGENT MODEL

BY

PATRICK MARIONYA OSINDI

A RESEARCH PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT FOR A MASTERôS DEGREE IN COMPUTER SCIENCE IN THE SCHOOL OF COMPUTING AND INFORMATICS IN THE UNIVERSITY OF NAIROBI

SEPTEMBER 2014

DECLARATION

This research project report is my original work and has not been presented for an award in any other university.

_____ _____

Patrick Marionya Osindi Date

APPROVAL

This research project report has been submitted for examination with my approval as the University Supervisor.

_____ _____

Dr.Opiyo T Omulo Date

SeniorLecturer
School of Computing and Informatics,
UniversityofNairobi

ABSTRACT

The aim of this research project was to research, design and develop a multi-agent logistics planning system prototype based on a BDI agent model. In most cases, logistics planning is done by repeated manual calculations that are error-prone, cumbersome and unnecessarily long. Theoretical literature describes the BDI multi agents and their suitability for the logistics planning problems. Empirical literature reviewed the contributions of other researchers to the research topic. BDI-Agent Software Development Process (BDI-ASDP) methodology was employed in the analysis and design of the logistics planning system prototype. The system design was implemented in Jason open source agent programming platform. The product was subjected to a thorough evaluation using System Usability Score (SUS) evaluation tool and registered an above average SUS score of 77.We expect that the BDI agent architecture can provide solutions to an otherwise long, tiresome and sophisticated manual logistics planning process.

ACKNOWLEDGEMENT

Foremost, I thank the Almighty God for the gift of life and good health. His daily providences have made this project a possibility.

I would like to express my sincere gratitude to my supervisor Dr. Elisha Opiyo for his valuable support in my graduate studies and in the research project. His guidance helped me all the time of my research and in the writing of this report.

Last but not least, I would like to thank my late father for his support and blessings in my formative years. I appreciate my mother for her unconditional love, prayers of good will and support that she has tirelessly accorded me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. AI Artificial Intelligence
2. AO Agent Oriented
3. AOP Agent-Oriented Programming
4. AOSE Agent-Oriented Software Engineering
5. BDI Belief-Desire-Intention (model)
6. DFD Data Flow Diagram
7. JADE Java Agent DEvelopment language
8. MAS Multi-Agent Systems
9. OO Object-Oriented
10. OR Operations Research
11. ASDP Agent Software Development Process
12. UML Unified Modeling Language

1 CHAPTER ONE: INTRODUCTION

1.1 Definition of Logistics

According to Vitasek(2013), logistics management is the part of supply chain management that plans, implements, and controls the efficient, effective, forward, and reverse flow and storage of goods, services, and related information between the point of origin and the point of consumption in order to meet customer's requirements.

Security operations and logistics are inextricably connected. Logistics is defined as ñthe practical art of moving armies and keeping them suppliedò(Creveld, 1977). Logistics can also be ñthat branch of administration which embraces the management and provision of supply, evacuation and hospitalization, transportation and service. It envisages getting the right people and appropriate supplies to the right place at the right time and in the proper conditionò(Creveld, 1977).

This capacity to concentrate ammo, transportation, food, troops, and other resources at a particular place and time is what gives commanders advantage over their foes.

1.2 Logistics and the Military

In military it is the work of logistics planners to ensure that the campaign plan from operational planners is actualized. Logistics planners ensure that the troops involved in an operation are deployedto their operation field, and are well supported until they accomplish the campaign. They do this by availing them the supplies they require to achieve the campaign, such as food, fuel, ammo, water, equipment, maintenance and medical assistance. The first step is to determine what resources are required to adequately support the operation.

In most cases, this is done by many repeated calculations that are error-prone if carried out manually. This results in a schedule which is a matrix of supplies and means of transportation. The last step is to acquire the needed supplies; and transport them and the troops to the operation area.

The process of coming up with a logistics plan is not linear as logistics planners must satisfy many constraints. For instance, troops must be deployed within the operation time constraints, resources must be availed periodically before they run out, the most suitable means of transport must be employed depending on the terrain of the operation area, and the weather must not be an impediment to the chosen means of transport. This process is complicated in that unsatisfying and overlooking one constraint could lean in an unworkable logistics plan, and hence impact on the overall operation plan.

1.3 Problem Statement

Military logistics planning is essentially concerned with the supply and transport of resources to aid military operations. For instance in order to deploy a force element to conduct a military operation, logistics planners must establish the supply needed to adequately sustain them throughout the operation. Currently in most military organizations in Africa, most of this planning is carried out manually by those soldiers trained in logistics matters. This sophisticated, tiring and long process involves a number of arithmetic to fulfill constraints, and combining of possible plans to achieve desired logistics goals. Logistics planning being an important aspect of any military operation, a logistics management information system could have the potential to improve the process of logistics planning and the output logistics plans. Furthermore, a logistics management information system could assist the military logistics planning soldiers by automating most of these complex logistics planning processes.

Transport department needs to optimize the available means of transport to move soldiers, supplies and services at the right time and in required condition from the point of origin to the point of destination. This will involve selecting the appropriate vehicle depending on the size of troops or size of goods. It may also involve choosing the routes that will ensure optimum results in terms of delivery times.

There is need for a system that can help commanders at the strategic levels in planning and forecasting logistics. Much wastage especially in supply of foods and fuels are experienced because there is no platform to help in making decisions that will avoid overestimations or delays in shipping and delivery of perishable commodities.

There is lack of operational transparency among all the players in the logistics arm of the military. No one player can access real-time or near real-time state of the other player. For instance, the contracted supplier of say wheat flour cannot tell the stock levels of the same commodity in the military primary depot. The transport department cannot know what is due for transportation to make prior arrangements and avail the appropriate means of transport. The commanders in field need a platform to support them in making orders/requisitions and closely monitor the state of their orders. They need to know when their order has been received, whether it is being processed or delivered.

To add spanner to the works, military planning has increasingly become decentralized, open and dynamic. Military has traditionally used its own assets to fulfill its logistics demands. In this approach, all data concerning logistics assets in collected and processed by a single decision making entity. This resultant plan is then applied onto the assets to be implemented. This has however changed due to deregulation, joint operations and outsourcing and military must procure services from other civilian outfits to achieve their logistics needs. This implies that military no longer has control over the assets they employ to achieve their business goals. It also creates a dynamic and open environment where organizations may enter or quit the system promptly, and their goals and abilities are prone to change without warning during the planning process.

1.4 The Purposeof study

The primary aim of this study was toinnovate the use of Belief-Desire-Intention agent model to developa multi-agent logistics planning system prototype that can help commanders within any military in effective and efficient logistics planning. The research led to the design and development of a multi-agent logisticsplanning system prototype with a BDI decision engine. The research sought to demonstrate how most of military logistics planning and management problems will addressed using BDI agent model. I believe the outcome of the research and the resultant prototype will be of interest to future logistics system developers, to stakeholders in the domain of logistics management and to military formations around the world. The research will evaluate the developed system prototype for learning purposes and particularly explore the benefits of BDI software agent model in the development of distributed logistics planning systems.

1.5 Objectives of the study

1.5.1 General Objective

1. Design and develop a multi-agent logistics planning system prototype based on the BDI agent architecture

1.5.2 Specific Objectives

1. Design a multi-agent logistics planning system prototype based on the BDI agent architecture.
2. Implement the designed logistics planning system prototype using a suitable technology for experimentation, testing and evaluations.
3. Test and experiment with the developed system for learning purposes and present the findings

1.6 Research Questions

1. How do you represent the proposed logistics planning system in the BDI agent architecture?
2. Which is the best technology to implement a logistics planning system prototype for experimentation?
3. What are the results of the prototype tests and evaluations?

1.7 Assumptions and limitations of the study

The study assumed that the time constraints of the academic project will not affect the quality of the research findings

2 CHAPTER TWO: LITERATURE REVIEW

2.1 Theoretical Literature

2.1.1 Agents

A software **agent** is a computational system that is situated in a dynamic environment and iscapable of exhibiting autonomous and intelligent behavior. An agent may have an environment that includes other agents. The community of interacting agents, as a whole, operates as a **multi-agent system**.

The most important common properties of computational agents are as follows:

Purposeful: Agents act on behalf of their designer or the user they represent in order to meet a particular purpose.

Autonomous: Agents are autonomous in the sense that they control both their internal state and behavior in the environment.

Intelligent: Agents exhibit some kind of **intelligence**, from applying fixed rules to reasoning, planning and learning capabilities.

Interactive: Agents **interact** with their environment, and in a community, with other agents.

Agents are ideally **adaptive**, that is they are capable of tailoring their behavior to the changes of the environment without the intervention of their designer.

Mobility: an agent can transport itself to another environment to access remote resources or to meet other agents

Genuineness: An agent does not falsify its identity

Transparency/ trustworthiness: An agent does not communicate false information willfully.

2.1.2 Limitations of agents

Even though they exhibit only some of the above properties, agents relax several strong assumptions of classical computational intelligence: they typically have incomplete and inconsistent knowledge as well as limited reasoning capabilities and resources.

2.1.3 Multi-Agent Systems

A MAS is formed by a group of software agents that interact and typically communicate with each other to solve a given computational problem. In a multi-agent system environment, individual agents have to interact, communicate and collaborate towards achieving some shared goals.

Decentralized organization: MAS are inherently distributed software systems, enabling agents to transparently communicate regardless of their location. This transparency makes MAS subject to inherently distributed organizations, where the physical location is abstracted and systems operate in a decentralized network of distributed application components.

The inherently decentralized nature of MAS based applications is a major building block for the MAS characteristics discussed below and contributes to fault tolerance and scalability, since local failures only have minor effects on the software application itself and new components or agents can be connected or removed at run-time.

This decentralized infrastructure is particularly attractive for open environments where agents and hosts enter and leave the system at run-time. Concerning logistics applications, this feature, can facilitate the additionand removal of automatic guided vehicles or manufacturing machines.

Environment abstraction: While the physical environment is transparently hidden from agent developers, agents themselves are expected to inhabit an application dependent environment. The MAS environment can either be implicitly perceivable (only message passing agents) or explicitly represented (situated MAS). The environment provides a first class abstraction to interact with MAS external components and software frameworks are available that support modeling environment properties and agent interactions (Viroli *et al.* 2007). In case of situated MAS, the agents can interact indirectly by concurrently modifying their shared environment.

These indirect interactions are particularly useful for exploiting self-organizing phenomena (Serugendo *et al.* 2006), that is to achieve large-scale coordination solely by local interactions.

2.2 Operations Research

Operations Research (OR) is a branch of applied mathematics that uses algorithms, simulation, modeling, queuing, and stochastic methods to optimize or improve a real-world situation. OR was developed by a group of British and American mathematicians who were studying strategic logistics problems during World War II. Since that war, this branch of mathematics has been used in a variety of industrial and military applications.
Available OR Techniques:

2.2.1 Analyzing a Network

A useful OR technique is finding optimal solutions to problems involving start nodes, arcs, and destination nodes. A basic problem in OR is the ñtransportation problem,ò in which there are known supply bases, known customer demands, and known costs to each route from supply base to customer. (The costs could be in time, risk, shipping costs, or something else that is considered important.) The objective is to minimize total cost while meeting all demands. Networks, such as maximum flow networks (which are useful in describing port activities), decision trees, lattices, and other deterministic flows, can be used in many ways to solve logistics problems.

2.2.2 Programming

In OR, programming is used to quantify a problem involving an objective function that is subject to one or more constraints in the system. An objective function attempts to perform actions that affect the output of a system, such as minimizing shipping cost, maximizing throughput, or maximizing material shipped to an area. Constraints are functions that place limits on the range of the objective function. These can include limitations on infrastructure capacity, warehouse space, cost, trucks available, and integer, or non-negativity, limits.

2.2.3 Simulation

Simulation involves using a combination of deterministic and probabilistic functions to model the problem and then predict actual system improvements after changes. Because of the large number of calculations involved and the need for multiple runs, simulations are almost always run on a computer.

A number of excellent simulation programs are available, such as ProModel or Arena that use graphical interfaces to help model an actual system. Simple models, however, can be run from Microsoft Excel.

The important characteristics of the simulation technique are as follows:

a. Any number of variables can be handled
b. The data to be processed can be empirically derived and do not have to be smoothed or changed into equation form
c. The relationship between variables can be complex ie linear restrictions do not have to be maintained
d. The essential nature of simulation is that the model should vary in time, so that the process is a step-by-step re-enactment of the physical or qualitative system

Disadvantages of Operations Research

High computational requirement

The first drawback relates to the magnitude of mathematical and computing requirements. OR techniques try to find out an optimal solution taking into account all the factors involved. In modern business environment, these factors are enormous and expressing them in quantity and establishing relationships among them requires voluminous calculations that can only be handled by computers (UniversalTeacher, 2010).

Deals only with quantifiable factors

Even from the discussions in the previous sections of this paper, it can be inferred that OR isapplied only on problems which have quantifiable decision factors. It excludes other complex factors such as human behavior. Which means decisions will continue to be made based on personal judgment and experience. Here it should be pointed out that probabilities and approximations are being substituted for factors that could not be measured. Yet, a major proportion of managerial decisions involve qualitative factors (CiteMAN, 2007).

Time and cost

Priyanka points out that in order to carry out an effective research and implementation, a company needs to invest time and effort. A team of professionals must be hired to conduct research and that comes with high cost. As a result OR is not feasible for problems which do not involve big amount of money. Data collection by itself may consume a large portion of time and money (Priyanka, 2012).

2.3 Programming paradigms and their benefits

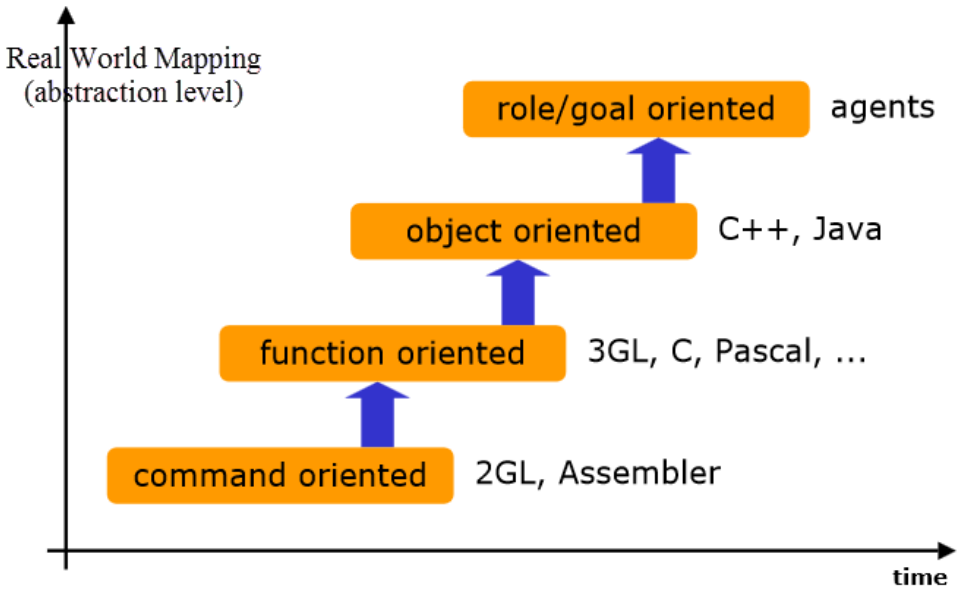

Figure 1: Abstraction Level vs Time graph

Software designers of large-scale embedded systems have had to deal with the challenge of managing complexity. The need for reliable, maintainable and extensible systems that conform to user specifications demands for use of design methodologies and modeling techniques that support abstraction, inheritance, structuring , modularity and other mechanisms that well manage complexities inherent in those systems. The following sections present some of those methodologies, how they work and some inherent limitations in the same methodologies.

2.3.1 Object-Oriented Methodology

This is a methodology that exists for the design, specification, and programming of software systems that are OO in their approach. They are based on the notion of objects which encapsulate state information as a collection of data values and provide behaviors through well-defined interfaces for operations upon that information. They provide key steps of object identification, design, refinement, permitting abstraction through object classes and inheritance within class hierarchies.

This methodology has well matured and is widely accepted due to its numerous advantages.

The OO methodology decomposes a system by identifying key object classes in the application domain, focusing upon their behavior and their relationships with other classes. The details of a system can be captured using three models:

An Object Model captures information about objects within the system, describing their data structure, relationships and the operations they support

A dynamic model describes the states, transitions, events, actions, activities and interactions that characterize system behavior.

2.3.2 Agent-Oriented Paradigm

ñAgent-based programming is programming based on agents. A program consists of a set of agents and their collaboration. Agent-oriented approach involves agents that learn themselves by experience and adapt themselves based on the previous learning to the current environment.ò(Jo, 2001)

This is AI paradigm that is founded upon the notion of reactive, internally-motivated and autonomous entities embedded in dynamic, uncertain worlds which they perceive and in which they act.

AO methodology supports the decomposition of a system based on the key roles in the system. The identification of roles and their relationships guides the specification of an agent class hierarchy. A further analysis of the responsibilities of each agent class leads to identification of the services provided and used by an agent, and hence its external interactions.

These details are captured in two models:

An agent model which describes the hierarchical relationship among different agent classes and identifies the agent instances which may exist within the system, their multiplicity and when they come into existence

An interaction model which describes the responsibilities of an agent class, the services it provides, associated interactions, and control relationships between agent classes. This will include the syntax and semantics of messages used for inter-agent communication and communication between agent and other system components, such as user interfaces.

Roles, responsibilities, and services are just descriptions of purposeful behavior at different levels of abstraction.

2.4 BDI Agents

The BDI model was conceived by Bratman as a theory of human practical reasoning(Bratman, 1987). BDI agent model is an event-driven execution model providing both reactive and proactive behavior. The BDI agent model is built on a simplified view of human intelligence. In it, agents have a view of the world (Beliefs), certain goals they wish to achieve (Desires), and they form Plans (Intentions) to act on these using their accumulated experience.

The BDI model enables to view an agent as a goal-directed entity that acts in a rational manner.

ñAgents have explicit goals to achieve or events to handle (desires). A set of plans (intentions) is used to describe how agents achieve their goals. Each plan describes how to achieve a goal under varying environments (beliefs). A set of data called belief describes the state of the environment.ò (Jo, 2001).

Agents can be seen as individual problem-solvers with capability of sensing and acting upon their environment, for deciding their course of action and communicating with other agents. Depending on the challenge at hand, agents can apply various means of problem solving among them searching, reasoning, planning and learning(Jo, 2001). In the reasoning category is the knowledge-based reasoning and sophisticated BDI model. It has been described as the most expressive model of an agent and its knowledge about the environment. The model assumes the agent has both certain and uncertain knowledge ï beliefs(B) ï regarding the states of its environment.

In BDI terms, *Beliefs* represent knowledge of the world. However, in computational terms, beliefs represent the state of the world, such as the value of a variable, a relational database, or symbolic expressions in predicate calculus. *Desires* (or goals) form another essential component of system state. In computational terms, a goal may simply be the value of a variable, a record structure, or a symbolic expression in some logic. The important point is that a goal represents some desired end state. The committed plans or procedures are called *Intentions,* which represent the third necessary component of system state. Computationally, intentions may simply be a set of executing threads in a process that achieve the goals (desires) of the system.

Beliefs, Desires, Intentions are the basic components of an agent system designed for a dynamic, uncertain world. So, as BDI agents, they must have explicit goals to achieve or events to handle (desires); a set of plans (intentions) is used to describe how agents achieve their goals; a set of data called belief describes the state of the environment(Jo, 2001).

Rao & Georgeff (1995)give another justiýcation and explanation for the need for intentions. They say that beliefs are needed as the informative component of a system state. Desires provide objectives for the system to accomplish; the ómotivationalô state of the system.To control the balance between always re-planning and never re-planning (until the whole plan has been executed), the system must represent the currently chosen course of action called the systemôs intention: the ódeliberativeô component of the system state.

Belief is necessary for the usual reasons of the necessity for representationð the ability to keep information that is not directly perceivable and to use this information to make more effective decisions. A desire is thoughtofas a goal. George sets goal-orientated computation against task-oriented computation. Task-oriented computation ñis executed without any memory of what is being executed,ò (Georgeff *et al.* 1999)

Practical reasoning can be divided into deciding what to do and determining how to do it (Bratman, 1987).Wooldridge (2002)calls these two processes deliberation and means-ends reasoning respectively. In the context of practical reasoning, deliberation means deciding on goals to pursue and means-ends reasoning means determining plans to achieve those goals.

2.4.1 Advantages and Benefits of BDI agents

The BDImodel is a popular and well-studied architecture of agency for intelligent agents situated in complex and dynamic environments. The model has its roots in philosophy with Bratmanôs (1987) theory of practical reasoning and Dennettôs theory of intentional systems (Georgeff *et al.* 1999). BDI AO systems are extremely þexible and responsive to the environment, and as a result, well suited for complex applications with real-time reasoning and control requirements.

2.4.2 Limitations

However, a limitation of these systems is that they normally do no look ahead or planning in the traditional sense; execution is based on a user-provided ñplan libraryò to achieve goals.

2.5 Characteristics of Logistic Problems

Davidson & Kowalczyk, (1997)define logistics as follows: ñLogistics is the process of managing the flow and storage of materials and information across the entire organization with the aim to provide the best customer service in the shortest time at the lowest cost.ò This is to mean that Logistics must provide solutions for resource planning and transport in the broadest sense. (Alexander*et al, 2002*), give examples of logistics problems as fleet management, order management, route planning, fleet scheduling, and cargo management.

Most of these Logistics problems are NP-hard and there are no efficient algorithms available, which can provide optimal solutions in sophisticated real-world scenarios. The following sections highlight the common characteristics of Logistic problems,(Peruginі*et al*, 2003).

High complexity:

Logistic problems in most cases involve a number of components, which demonstrate sophisticated behavior that are interrelated in different ways. To solve a logistic problem needs a good understanding these dynamics in order to provide ways for handling the complexity.

Large decision space:

The solution strategies for solving logistics problems can have multiple alternatives at their disposal and many varied decision variables need to be taken into account. Besides, these alternatives are in most cases difficult to evaluate and prioritize.

Utilization of real-time data:

Logistic departments these days are faced with a dynamic world, in which a number of unexpected situations can arise. For these departments to remain competitive, data has to be harnessed and analyzed in real time.

Uncertainty:

Business environments are characterized by partial or incomplete information. Sometimes, this imperfect knowledge can be used as a basis to make important decisions. Furthermore, unanticipated events might occur for instance emergencies and machine breakdowns which could bear a severe impact on the flow of goods and services and they have to be dealt with.

Numerous decision makers:

Logistic processes in most cases involve many decision makers, who work in concert for the processes to run smoothly.

Highly constrained:

Many are constraints that are required to be satisfied to plan and accomplish logistic activities. These may include but not limited to available storage and machine capacities as well as business objectives such as production efficiency or customer satisfaction.

Distributed domains:

This may take the form of complex settings consisting of physically distributed entities and/or data. Next, it may involve stakeholders who have individual desires such as sticking to their rest times, which have to be synchronized with institutional objectives for instance delivering goods on time.

Even if logistic settings might not exhibit all the above mentioned properties at once, ***logistics solutions and software*** have to embrace the existing characteristics and handle them in an intelligent way.

Understandability:

The complexity of the logistics problems notwithstanding, the provided software should attempt to mask this complexity as far as possible and provide usable interfaces.

Furthermore, it is often beneficial that a software system makes transparent what it does and allows users to understand how the applied solution strategy works. If decision support systems are considered this may lead to an increased acceptance of the software

Seamless software / operator interaction: In many logistics scenarios manual operators work hand in hand with software tools supporting them.

As software cannot always be aware of all currently relevant knowledge and additionally the operator might have long experience with certain tasks, the software should in those scenarios play the role of a subordinated assistant.

This means that the software should make autonomous decisions only if explicitly authorized by the operator. Otherwise the software should make recommendations leaving the final decisions about its execution by the human operator (Dorer & Calisti, 2005).

Robust system behavior: Logistics software systems should exhibit robust system behavior also in unanticipated situations especially due to the great amount of uncertainty in the domain.

Concretely the software should be able to cope with unexpected situations and produce acceptable results also in those situations.

2.6 How Logistics can benefit from Agent-based solutions

In this section it will be discussed how the previously presented inherent agent properties make agent-based approaches suitable and appropriate for solving typical logistic problems. In general, multi-agent systems provide natural important metaphors that facilitate a high-level and understandable description of the problem domain and the aspired solution

Autonomy: In Logistic Problem, each decision maker can be represented by an agent, which has the purpose to act on behalf of its principal.

Despite the possibility of autonomous action the degree of autonomy is controllable and should be adapted to the specifics of the concrete application domain and the responsibilities of the agent in the system.

Reactivity: this should Regarding the logistics domain reactivity is extremely important for coping with uncertainties. One important aspect of these uncertainties is unexpected occurrences such as breakdown of machines or delays in delivery of goods, which need to be considered by the logistics system as soon as possible. If the environment is monitored and occurrences are propagated to agents with reactive capabilities a timely handling can be enforced.

Proactivity: Concerning the logistics domain proactivity allows to specify the individual objectives of the different participating entities.

This means that e.g. in a transportation scenario the vehicles as well as the hubs could be represented as agents, which are seeking to fulfill their aims. In this respect, one important vehicle objective could be to perform transportations with high utilization.

Social abilities: The social abilities combined with the decision freedom of agents allow them to communicate with others whenever they see need for it. In a transport setting, truck agents could e.g. proactively communicate to other trucks nearby that the used highway is jammed. This new knowledge gives the other truck agents the chance to re-plan their current route and possibly avoid the jam.

BDI Model: In the context of logistic scenarios mentalist agent descriptions can help managing the complexity of behavior descriptions.

As an example one can consider the scenario in which one top-level goal of a truck agent is to bring a packet to the main station. Depending on the delivery context different routes may be applicable, but this does not to be considered on the highest abstraction level.

Instead, lower level plans can handle the route planning according to the delivery context and e.g. prefer freeways if cost effectiveness is important or also consider routes liable to charges for time-critical deliveries.

2.7 Multi-Agent characteristics of logistic problems

Decentralized organization: Concerning Logistics applications, this feature, e.g. facilitates the addition and removal of automatic guided vehicles or manufacturing machines.

Environment abstraction: Since logistics is often intrinsically related to the spatial movement of vehicles, it is particularly attractive for developers to represent the system context explicitly.

Concerning logistics the availability of routes will be influenced by external factors like traffic jams. These application internal events are to be represented in environment models allowing the agent population to perceive and adjust.

Self-organizing Behavior: Particularly, for logistic settings it is interesting to allocate resources and tasks in adaptive ways. E.g. transportation routes can be subject to adaptation as to react on vehicles unavailabilityôs (e.g. repairs) and availability of new transporters to address high workloads as well as to allocate trucks to specific routes. In manufacturing line control, working examples are available that show the benefits of the self-organizing adaptation of the routes of items in production lines.

Coordination Mechanism: this is to in transportation logistics one usually wants to maximize utilization of trucks (i.e. avoid tours of only partially loaded trucks), but also wants to minimize packet delivery time. By representing individual resources (e.g. trucks and packets) as agents that negotiate with each other, appropriate trade-offs between conflicting goals can be established using suitable coordination strategies that move solutions in the direction of a global optimum.

Organization Structures: In logistic scenarios organizational ideas can e.g. be used for naturally mapping real-word settings. In military transport logistics the existing hierarchical troop structure consisting of groups, subgroups and individual vehicles can be directly used in the software design. Also, in manufacturing logistics different production cells and their contained machines can be modeled as groups and agents. This allows viewing the design at different levels and different aspects can be emphasized if the top-level or lower-level layers are under consideration.

2.8 Empirical Literature

Logistics planning and management is vast area and therefore literature abounds on these subjects. Some of this literature focuses on the traditional approaches to logistics planning and management i.e OR and AI. Solutions to logistics problems using these approaches assume a static and centralized logistics environment. These approaches cannot be relied upon to appropriately solve problems in militaryôs devolved, open and dynamic logistics environment.

Traillandier *et al* (2012), propose a BDI architecture dedicated to cognitive agents that is based on the belief theory. The aim of the paradigm is to minimize complexity and agent execution time by using belief theory in choosing the most relevant action for an agent to pursue. The architecture is composed of four databases namely: desires, plans, beliefs and intentions. In this, the agent selects a plan through three multi-criteria decision-making processes i.e. choice of plan, choice of actions and plan execution control. Also an application of the architecture to a real model concerning cropping plan decision-making is presented. The model aims at simulating the behaviors of farmers in their choice of crops and their day-life activities.

The model constitute of two main agents: field and farmer. For the farmer agent, a plan represents complete assignments of crop rotations to its fields. A plan is defined by a set of actions, action application rules and plan update rules. This research reiterates on the suitability of BDI agents in developing multi-agent systems that can exhibit complex reasoning abilities while interacting with its biophysical environment and other agents. This architecture is also helpful as it provides some insights that can be useful in designing the multi-agent logistics planning system.

Nitin *et al*, (2010)developed RACT-C in the context of a multi-agent programming contest which is a multi-agent system built following the BDI agent-oriented paradigm. The Prometheus methodology and its corresponding Prometheus Design Tool (PDT) were used in designing the system. The actual system implementation was carried out in the JACK programming language, which is an extension of Java to support BDI agent features. The concept behind its development was to realize MAS to solve a cooperative task in a grid-like dynamically changing world where agents in a team can move from one cell to a neighboring cell. System specification is done in terms of how (external) actors interact with the scenarios of the system via input percepts and output actions. The specification is then further refined into a goal hierarchy and set of main roles. The second development stage involves the architectural design of the system. Based on the roles from the specification phase, the required data, agents, and communication protocols among them are detailed. Agents are meant to autonomously and proactively be able to address the goals associated with their corresponding roles. The communication that shall be required is encoded in the protocols. Coordination and team-work arises as agents take roles with shared goals in the goal hierarchy.

The agent system has specific plans encoding simple strategies for various tasks, such as, navigating the grid, pushing cows towards a direction, going through fences in groups, and finding objects like fenceôs switches or cows. The PDT functionality of generating skeleton code of BDI agent is only limited to JACK programming language.

Gavin &Feb (2010) presentBDI architecture with a logic based planner that can be applicable in stochastic domains. The motivation behind the research is to improve reasoning of BDI agents by not only using a library of plans but also generating suitable plans on demand to pursue different goals under circumstances. This is done by integrating a Partially Observable Markov Decision Process (POMDP) into the BDI architecture to combine the benefits of the architecture with the ability to generate plans. The POMDP planner is a program written in Golog programming language which can be executed directly by an agent.

TeleTruck system by Hans-Jurgen *et al* (1997)is an application prototype developed in close collaboration with a forwarding company. It schedules realistic orders using heterogeneous agents modeling different forms of vehicles. A central idea underlying the TeleTruck approach is to model the basic physical objects (drivers, trucks, trailers, containers, of the transportation domain explicitly by basic agents. These agents have to join together and form *holonic* agents that act in a corporated way. These composed agents represent the physical transportation entities (e.g. road trains or articulated vehicles together with their drivers) which are able to execute the orders. The contract net protocol (CNP) is used as a basic problem solving paradigm for the assignment of orders to trucks. The shipping company agent announces orders to the set of trucks and waits for bids regarding which costs would be produced if a specific order was executed by a specific truck.

The shipping company agent evaluates all bids, selects the best one, and gives the truck which submitted the best bid the award to execute the order. Although each individual assignment of orders to trucks is an optimal choice regarding the current situation. A sequence of such choices is not an optimal solution to the problem that is given to the system. This is a general problem of CNP which one needs to take care of in practice. This approach solves only two problems within the logistics domain i.e pick-up and delivery problem and vehicle routing problem. This prototype nevertheless does not capture any element of BDI reasoning engine.

Janis Grundspenkis *et al*(2003) developed a multi-agent based simulation tool in Borland ++ and MS Access for decision support in transportation and logistics domain. The multi-agent system consists of clientsô agents and logistics companies agents which may participate in four types of auctions namely English auction, Dutch auction, First-price sealed-bid auction and Vickrey auction. A client is an auctioneer who is making decision about the best offer of delivering goods. In as much as this solution is multi-agent based it provided through Borland C++ which is not well adapted for agent programming. The solution may work in a situation where Military poses as a client that wants the optimal transportation option from the many available in the market. However it is far from what we want as the solution is narrow in scope and does not take the BDI agent model architecture.

3 CHAPTER THREE: RESEARCH METHODOLOGY

3.1 Introduction

This chapter outlines the **BDI Agent Software Development Process** method that was used todesign and analyze the proposed logistics planning system. In this method, the belief-desire-intention (BDI) agent model is adopted to define the system. In BDI terms, *Beliefs* represent knowledge of the world. However, in computational terms, beliefs represent the state of the world, such as the value of a variable, a relational database, or symbolic expressions in predicate calculus. *Desires* (or goals) form another essential component of system state. In computational terms, a goal may simply be the value of a variable, a record structure, or a symbolic expression in some logic. The important point is that a goal represents some desired end state. The committed plans or procedures are called *Intentions*, which represent the third necessary component of system state. Computationally, intentions may simply be a set of executing threads in a process that achieve the goals (desires) of the system.

Beliefs, Desires, Intentions are the primary building blocks of an agent system designed for a dynamic, uncertain world. So, as BDI agents, they must have explicit goals to achieve or events to handle (desires); a set of plans (intentions) is used to describe how agents achieve their goals; a set of data called belief describes the state of the environment.

There is a few methodologies for agent oriented programming, such as Deloachô MaSE, Frankôs methodology, and Wooldridgeôs Gaia, but none of them is a simple and thereby an efficient method to analyze and design agent-based software. In our approach, we settle for BDI-ASDP with the aid of the Use Case, the method of OO, to find the Desires and Intentions, and the DFD, the method of functionality, to find the Beliefs. By following the BDI Agent Software Development Process approach, the logistics planning system can be decomposed into Belief, Desire, and Intention (BDI) agent models.

3.2 BDI Agent Software Development Process (BDI- ASDP)

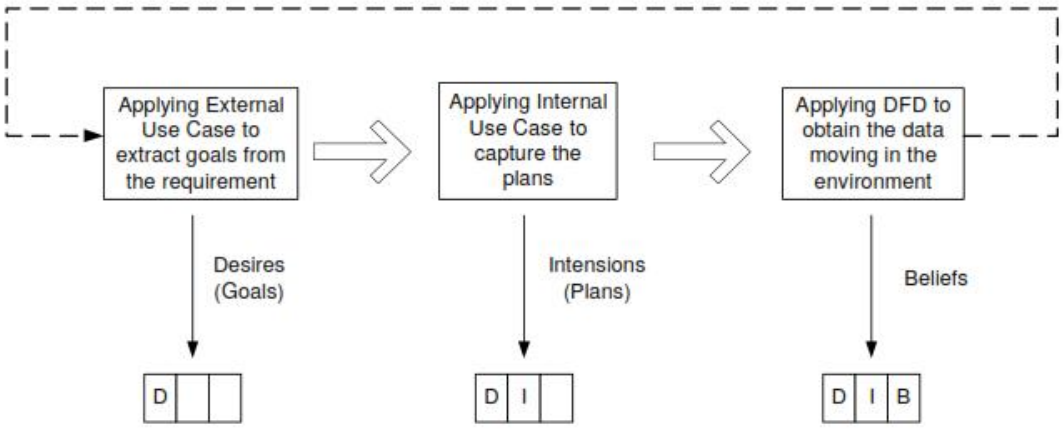

Figure 2: BDI - ASDP 3 Steps

This approach emphasized on ***analysis and design phases*** ofthe logistics planning system development cycle. In the analysis phase, External Use Case was applied to extract the desires from the external point of view as well as to fully understand the requirements and prepare for the design phase. In the next phase, the design phase, the Internal Use Case was utilized to capture the intentions while DFDwas employed to find the beliefs. After the beliefs, desires and intentions are discovered, they were collected together to form the BDI Agent Cards.

This methodology embraces three main steps as in Figure 2 and 5 and ten detailed steps as listed below and illustrated in Figure 3 below.

a. Initial Problem Statement
b. Enterprise Software Assessment
c. Brief External Use Case
d. Detailed External Use Case
e. Structuring Goals
f. Internal Use Case
g. Sequence Diagram
h. Agent Activity Diagram
i. Data Flow Diagram
j. BDI Agent Cards

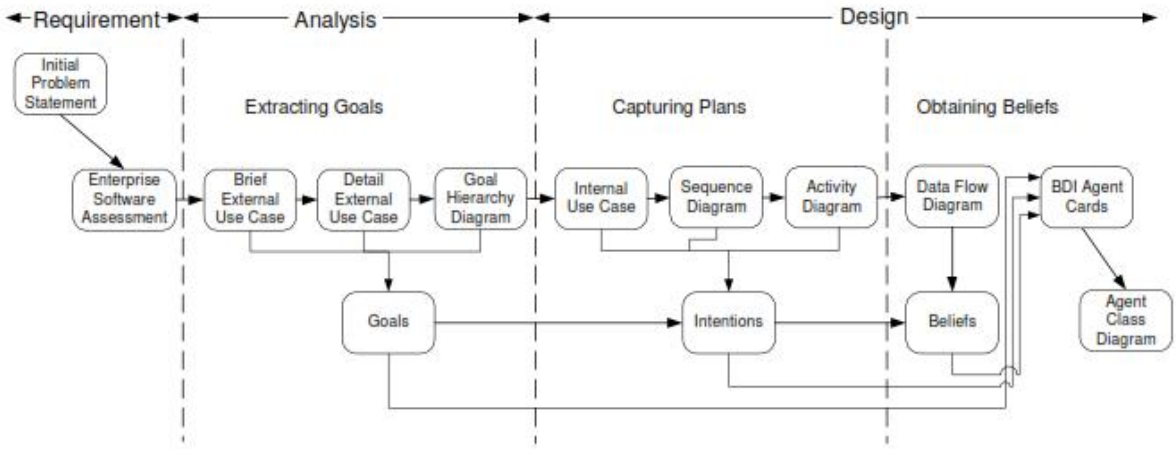

Figure 3: ASDP Requirement Analysis and Design Stages

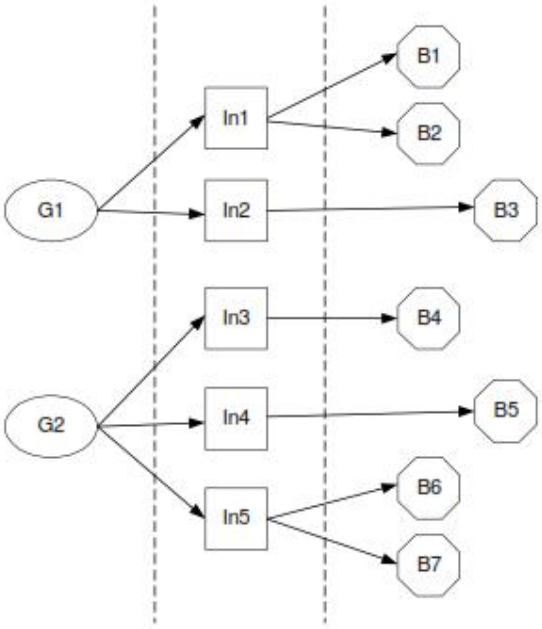

Figure 4: ASDP Goal Intention and Belief extraction process

3.3 Requirements Analysis

The requirement analysis was the initial part of the proposed logistics planning ASDP. It aided in understanding the system and to have a clear thought on how to construct it.

3.3.1 Step1: Initial Problem Statement

i. The system provides a web-based interface for the user to input the details of an operation plus constraints
   a) Peacetime or wartime operation ï training or otherwise
   b) The number of troops required
   c) The duration of operation

ii. Through the same interface, the user then prompts the system to draft a logistics plan for the operation

iii. The system calculates the supply requirement for the operation ï Food, Water, Ammo etc

iv. The system decides whether to supply from the military stores or civilian suppliers based on the costing and availability

v. Civilian suppliers are invited to make bids to supply various items. The supplier with the lowest cost of supply & transportation is chosen

vi. The chosen suppliers are notified (and should acknowledge) make the deliveries to meet the operation constraint. They should acknowledge receipt of the notification.

vii. The system selects the most appropriate mean of transporting troops and supply to the operation area while meeting the operation constraints (could be civilian or military)

viii. The system establishes how the supplies are to be stored in the operation area depending on their perishability or otherwise

ix. Where need be, the system should be able to seek external information helpful in building the plan

x. The system then drafts the logistics plan that is presented on an interface.

3.3.2 Step 2: Enterprise Software Assessment

There was not any requirements or constrain to an already developed logistics planning system, because the proposed system is the first version. The new software did not impact on any ancestor.

3.4 System Analysis: Capturing Goals

Use Case	Logistics Plan
Goal	Generate a logistics plan
Primary Actor	Logistics Planner
Stakeholders	Transport Stores Supplies
Preconditions	There is need for a logistics plan for an operation
Postconditions	The system drafts a logistics plan and schedule
Main Success Scenario	A logistics plan is generated and displayed
Extensions	the system fails to draft a plan

Table 1: External Use Case

Analysis phase emphasized on the*investigation* of the problem and requirements, rather than a solution. During analysis phase, we emphasized on finding the first element of agentsð *Desire*, in the proposed logistics planning problem domain.

3.4.1 Step 3, Step 4 and Step 5: External Use Case

Goals

a. Capture the Operations Constraints
b. Estimate the supply needed to support the operation
c. Find the suppliers
d. Find the optimum and appropriate means of transport for the supplies
e. Draft a logistics plan and display it

3.5 System Design

From this point, we will shift from the analysis to the design phase. During this phase, we will emphasis on defining proposed software agents and how they will collaborate to fulfill the requirements.

3.5.1 Step 6: Internal Use Cases

The Internal Use Case was concerned with the interactions among elements inside the proposed system. The *internal* Use Cases shows how entities interact in the system prototype internally and how the entities *use* each other to get things done. The purpose of this step was to identify the *intentions* (plans). The resultants of this stage are illustrated in the following tables:

Use Case	**Estimate the supply requirements for the operation**
Goal	Generate the operation supplies estimates
Primary Actor	Logistics Planner
Stakeholders	Logistics Planner
Preconditions	The operation constraints are given
Postconditions	The operation supplies estimates are generated
Main Success Scenario	The operation supplies estimates are generated
Extensions	The operation estimates are not generated

Table 2: Estimate generation process internal use case

Use Case	**Find Suppliers**
Goal	Find the optimum suppliers of the supplies
Primary Actor	Logistics Planner
Stakeholders	Logistics Planner Suppliers
Preconditions	The quantity of supplies
Postconditions	Selected suppliers and their details
Main Success Scenario	The optimum suppliers are selected
Extensions	The system fails to find optimum suppliers

Table 3: Supplier finding process internal use case

Use Case	Find Transport
Goal	Find the suitablemeans of transport for supplies
Primary Actor	Logistics Planner
Stakeholders	Logistics Transporters
Preconditions	The quantity of supplies
Postconditions	Selected means of transport and their details
Main Success Scenario	The optimum transporters are selected
Extensions	The system fails to find optimum tranporters

Table 4: Transport finding process internal use case

Use Case	Draft and Display Logistics Plan
Goal	Generate a logistics plan
Primary Actor	Logistics Planner
Stakeholders	Transport Suppliers
Preconditions	Operation constraints are provided
Postconditions	The operation plan is generated and displayed With suppliers and transport
Main Success Scenario	The operation plan is generated and displayed With suppliers and transport
Extensions	The logistics plan is generated and displayed

Table 5: Plan drafting process internal use case

3.5.2 Step 7: Sequence Diagram

Based upon the Use Case scenarios, the Sequence Diagram was used to illustrate the sequence of events that are transmitted and the relationship between *roles* (a role will include a particular goal or set of goals and a set of intentions). It showed how the roles communicate with one another over time. Below are sequence diagrams from the use case scenarios generated above.

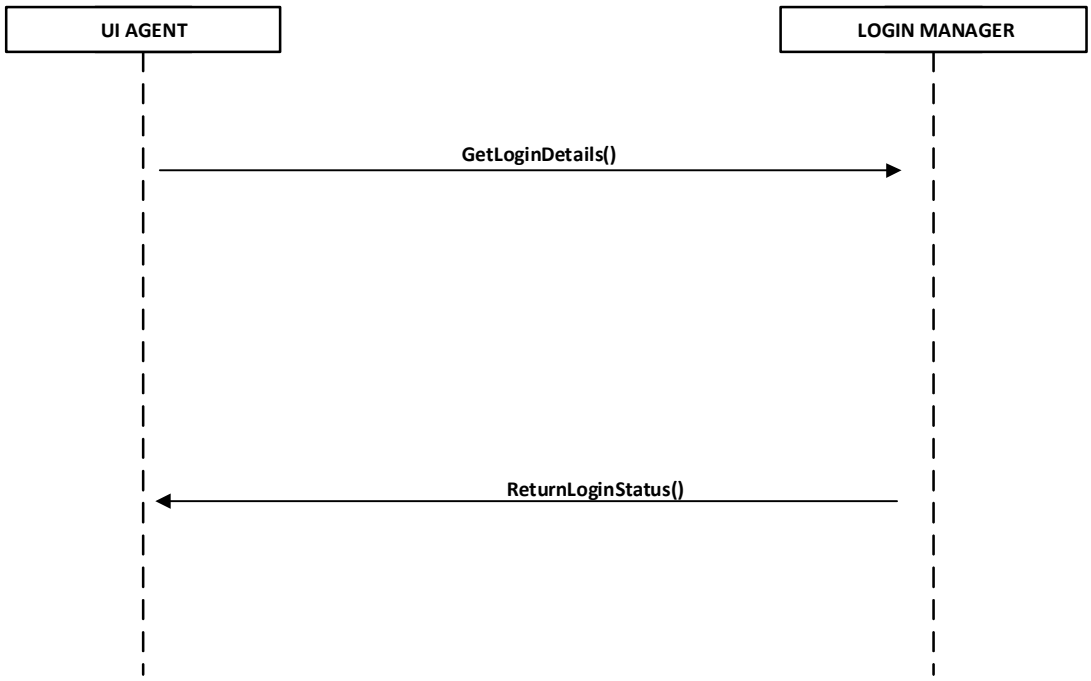

Figure 5: Login manager sequence diagram

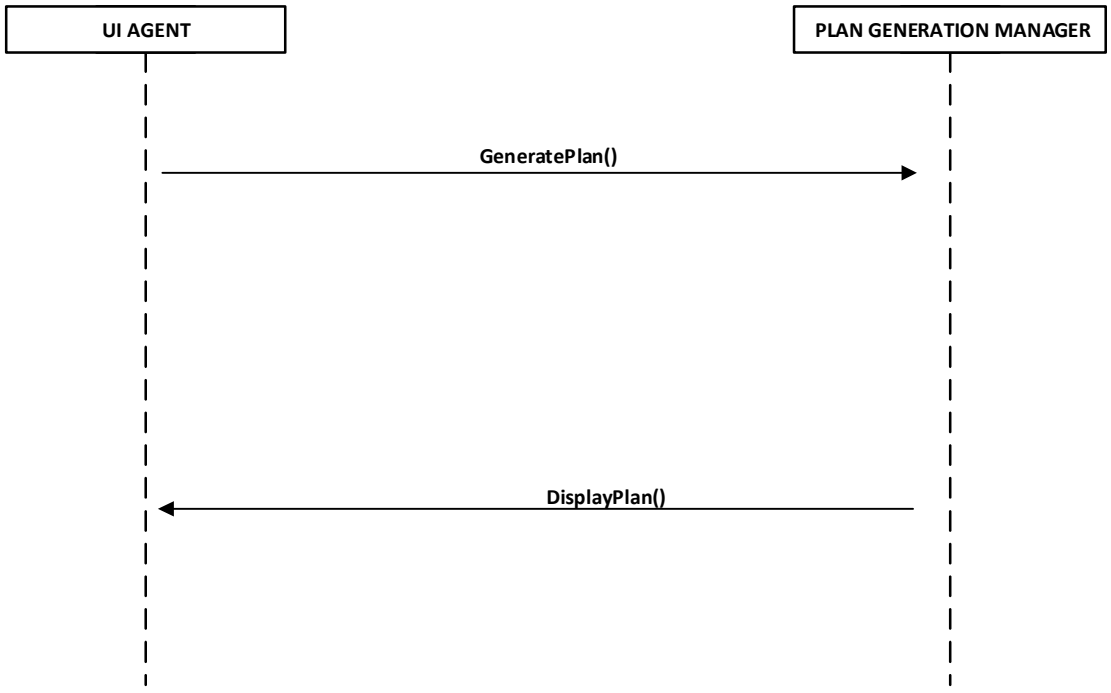

Figure 6: Plan generation manager sequence diagram

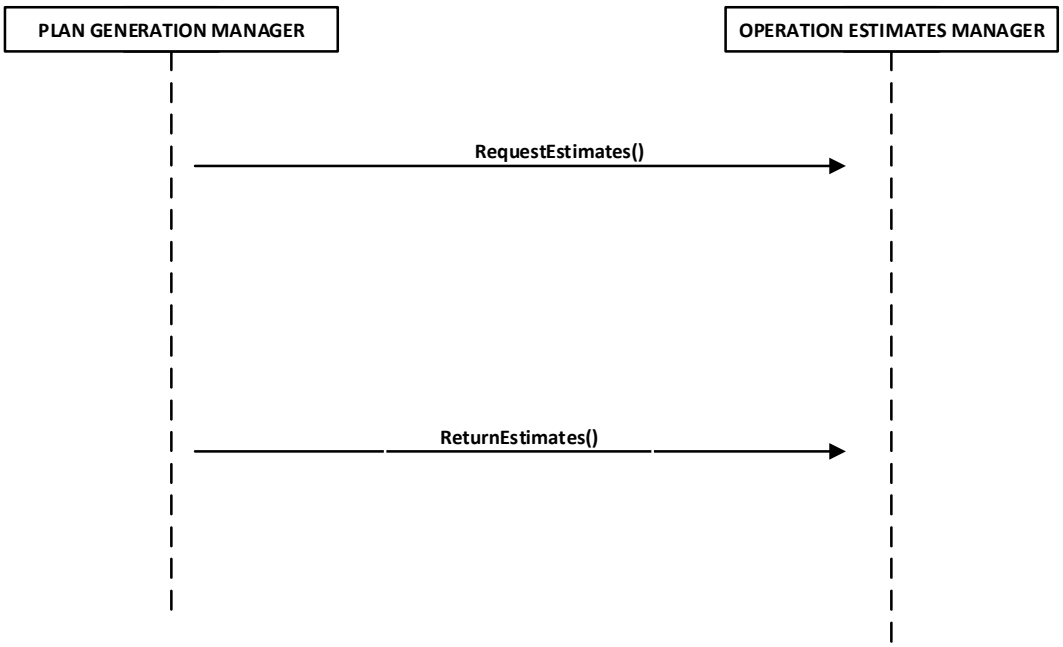

Figure 7: Operation estimates manager sequence diagram

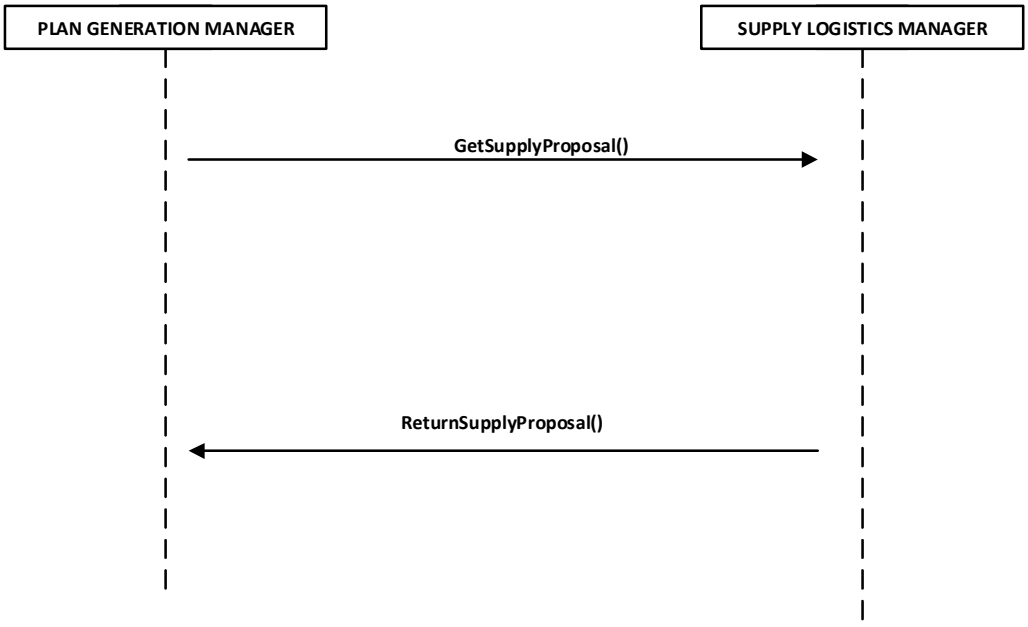

Figure 8: Supply logistics manager sequence diagram

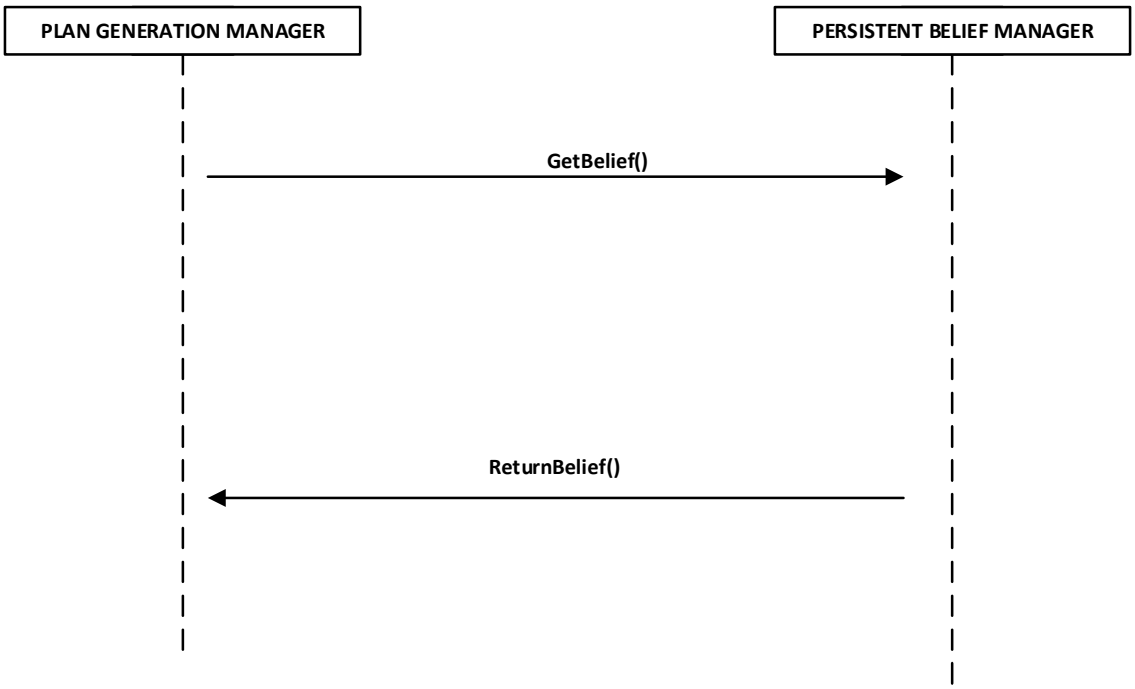

Figure 9: Persistent belief manager sequence diagram

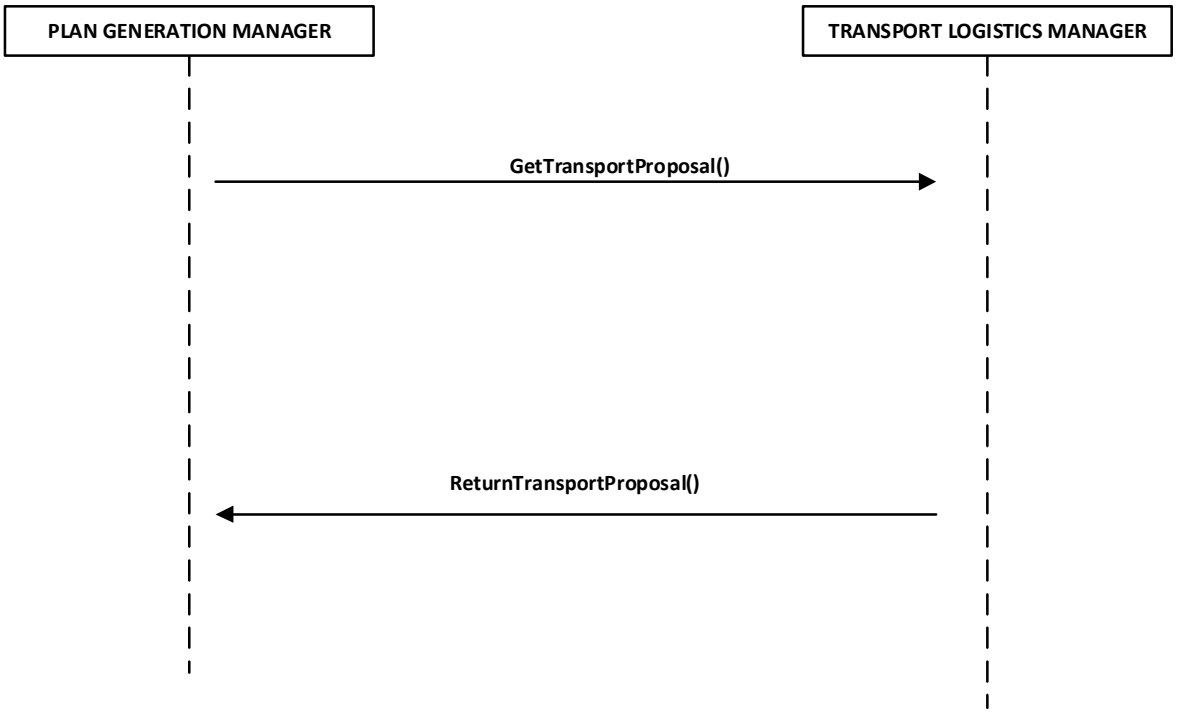

Figure 10: Transport logistics manager sequence diagram

3.5.3 Step 8: Agent Activity Diagrams

An Agent Activity Diagram (Agent Event Diagram) expressed operations and the events that triggered agents. The Agent Activity Diagramsare much like the flowcharts of old. They show steps (called *activities*) as well as decision points and branches. Activity Diagrams put the spotlight on the events. The following Agent Activity Diagramswere derived from the UML Activity Diagrams.

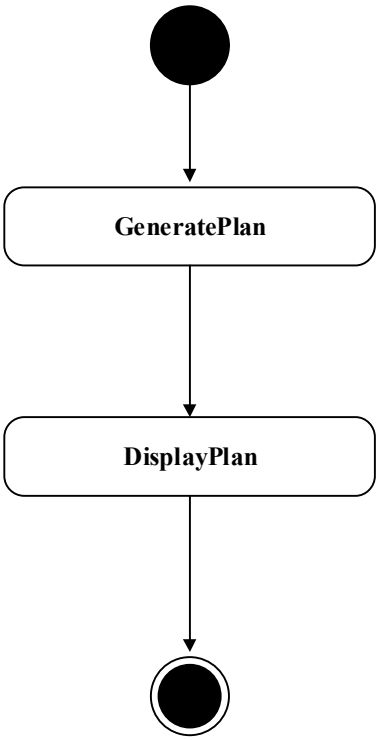

Figure 11: Generate and display plan process activity diagram

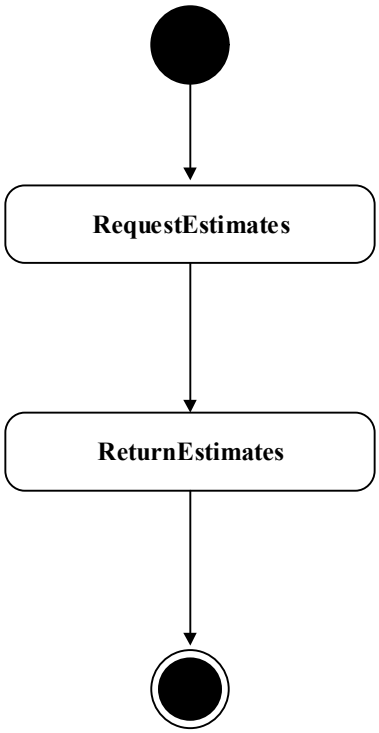

Figure 12: Request and return estimates process activity diagram

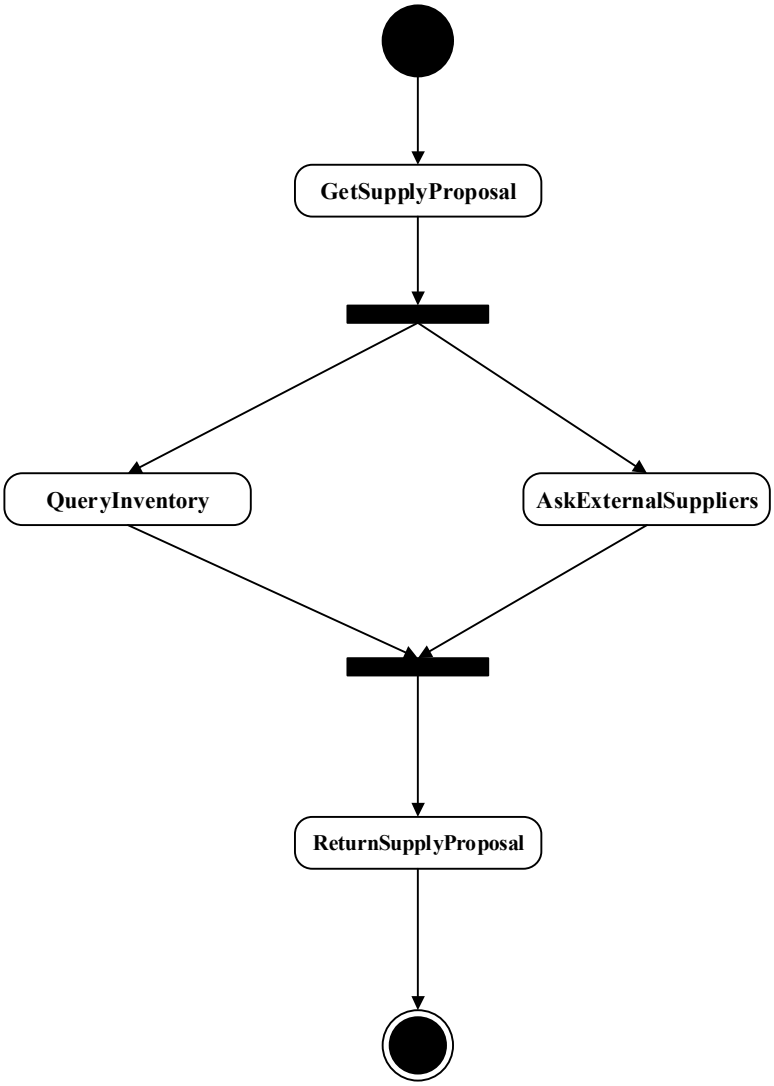

Figure 13: Find supply process activity diagram

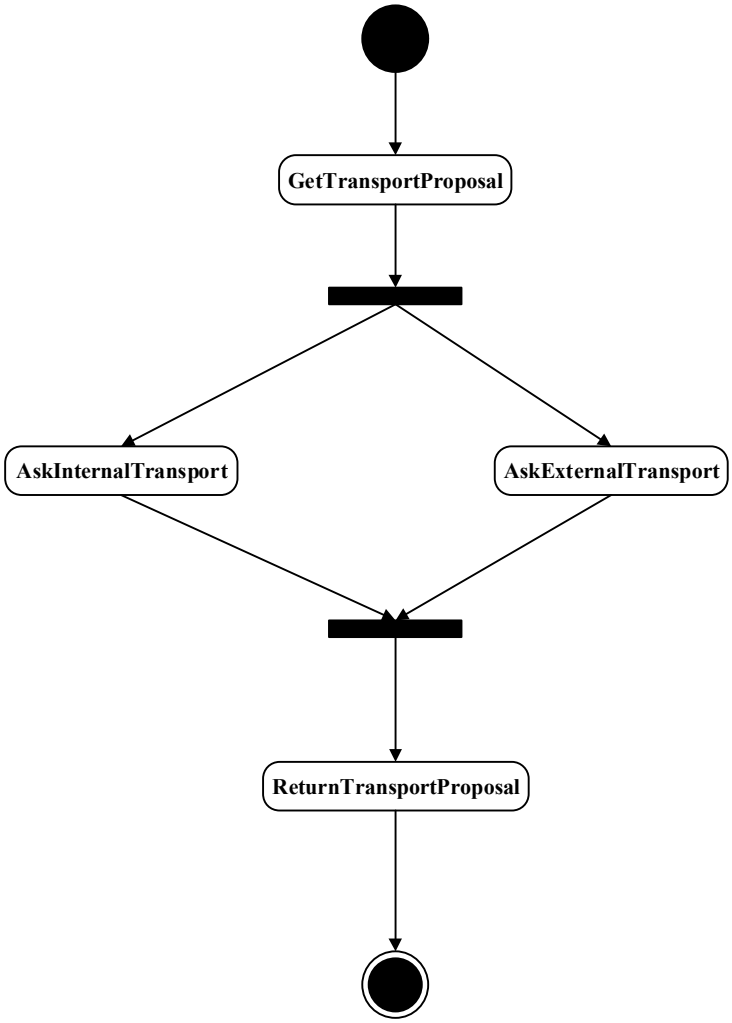

Figure 14: Find transport process activity diagram

3.5.4 Step8: Agent Belief List

An agentôs *beliefs* are a set of data describing the state of the environment. They are the knowledge that *intentions* use to fulfill their goals (*desires*). In order to find those data that agents needed, the DFD was applied in our approach.

Figure 15: Summary Data Flow Diagram

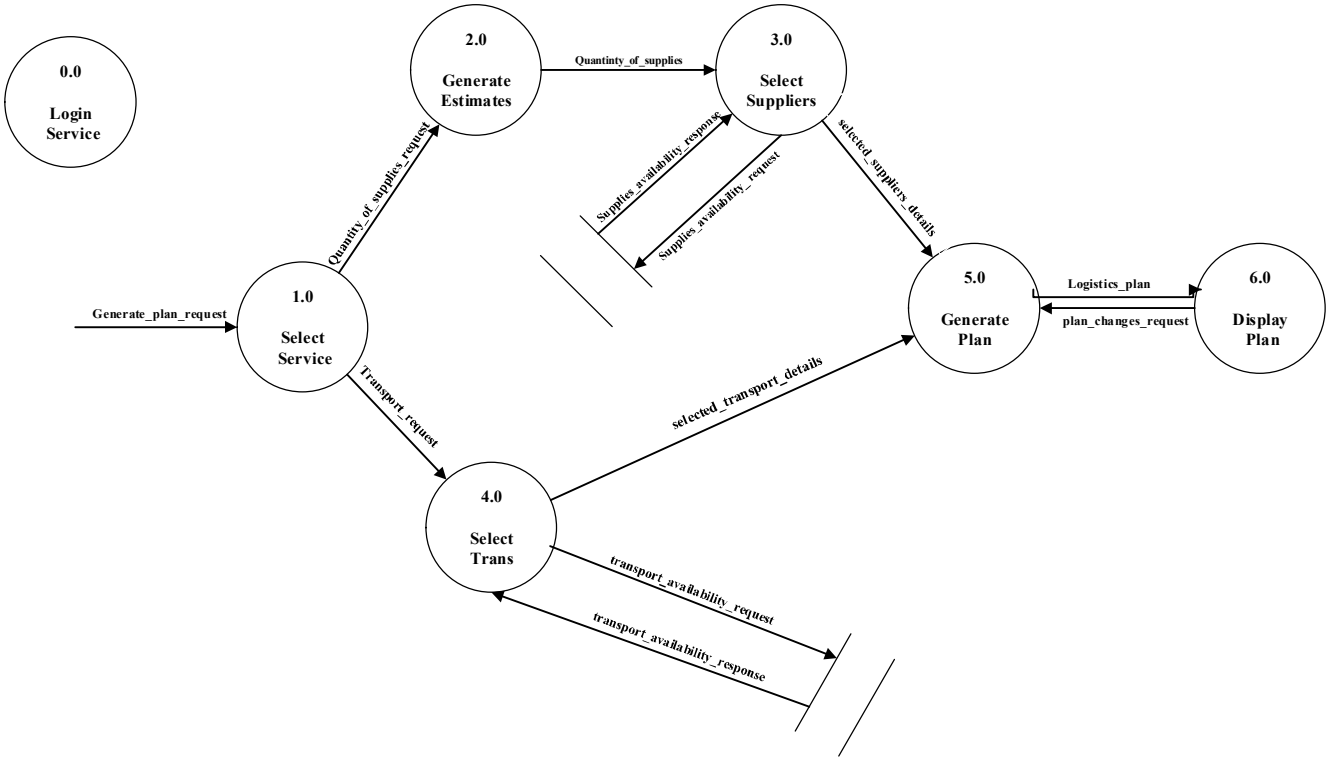

Figure 16: Detailed Data Flow Diagram

The data that was collected from the above Data Flow Diagrams inserted in the following use case form

Service 1

EstimatesManager

Goal

To generate item estimates of an operation

Belief

quantity_of_supply_request

size_of_force

time_of_deployment

quantity_of_supplies

Service 2

SuppliesManager

Goal

To find the suppliers of an operation supplies

Belief

quantity_of_supplies_request

quantity_of_supplies

supplies_availability_request

supplies_availability_response

selected_suppliers_details

Service 3

TransportManager

Goal

To find transport for the operation supplies

Belief

transport_request

transport_availability_request

transport_availability_response

selected_transport_details

Service 4

planGenerationManager

Goal

To generate and display plan for an operation

Belief

plan_request

quantity_of_supply_request

suppliers_details_request

transport_details_request

selected_suppliers_details

selected_transport_details

3.5.5 Step9: BDI Agent Class

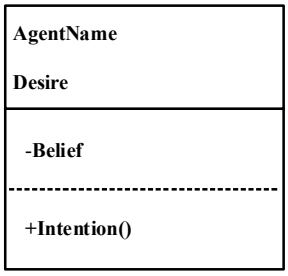

Figure 17: Agent Class Format

Ultimately an agent class diagram in Figure 19 below was generated after the format given in Figure 18 above.

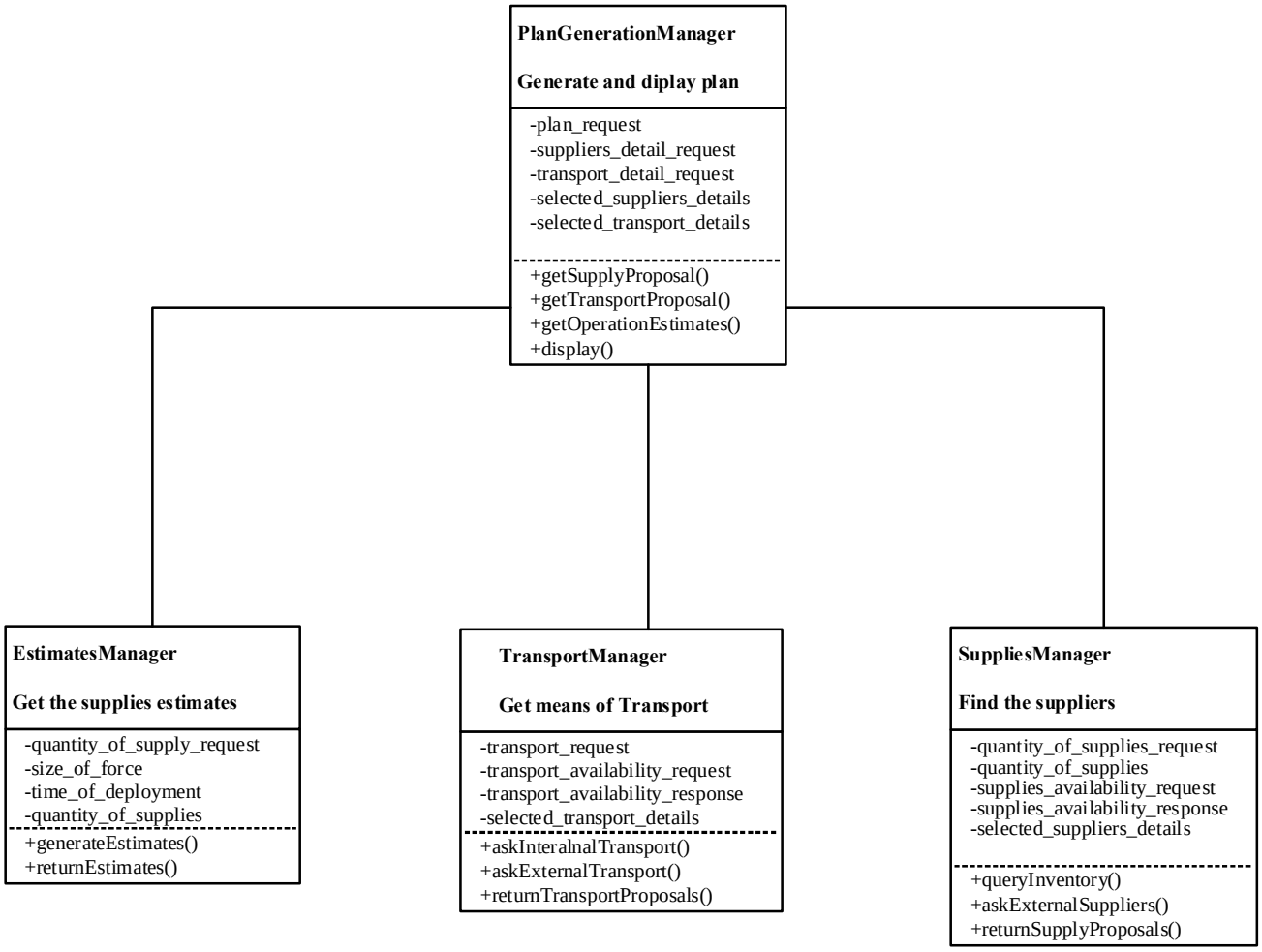

Figure 18: Agent Class Diagram

4 CHAPTER FOUR: IMPLEMENTATION AND DISCUSSION

4.1 Implementation overview

The implementation of the logistics planning system closely follows the BDI-ASDP methodology described in chapter 3. In particular, the four agents shown in Agent Class Diagram in figure 19 are implemented. Additional agents are also implemented to augment on the previous four agents. Each of the agents has the following basic capabilities:

a. **transportManager:** To find and select the most suitable means of transport to ferry goods and persons from one place to another
b. **estimatesManager:** To estimate the quantity/volume of items required to support an operation for a given duration
c. **suppliesManager:**Tofind the suitable suppliers for various items
d. **planGenerationManager:**To coordinate plan generation function
e. **home:** It acts as a text based interface to the system
f. **login:** It authenticates users into the system
g. **dbmanager:** used to establish connection to JDBC persistent Belief Base and to manage the same Belief Base.

Sample code can be seen in Appendix A.

4.2 Implementation technologies

4.2.1 Jason Agent Programming Language

Jason is an open-source platform written in Java which provides support for development of multi-agent systems. It is an implementation and extension of the BDI (Beliefs ï Desires ï Intentions) architecture based AgentSpeak(L) agent-oriented abstract programming language, and it can be used as a plugin for jEdit or Eclipse; further it provides high level of support and customization for most parts of the development of a complex multi-agent system. The Jason extensions to AgentSpeak (L) include extending the belief base to support arbitrary PROLOG clauses, support for both strong and default negation, and allowing annotations on plans and beliefs for instance stating the source of

a belief. Jason was selected as it gives a relatively high level of abstraction and facilities such as persistent belief bases that are well suited for the logistics planning system.

4.2.2 MySQL database and PhpMyAdmin

The two tools are used together to create and manage the Logistics Planning System persistent belief bases. The database is necessary to store persistent beliefs for querying when necessary.

4.2.3 Eclipse IDE

Eclipse IDE was used to write the code for the Logistics Planning System. This program is adapted for Jason programming with the installation of Jason plugin for Eclipse. This gives Eclipse the ability to recognize Jason code and highlight code errors for debugging. The plugin also makes it possible to compile and run any Jason script from the Eclipse environment.

4.2.4 JADE

This open-source java platform is used to complement Jason. It allows for full transaction control, multi-user object access, concurrency control with object-level locking, publish-and-subscribe events, and automatic object caching across distributed application servers. Through this Java API, all of the capabilities of the JADE object manager become available to Java applications. This platform was particularly adopted for the logistics planning system to make it fully distributed.

4.3 Testing

After the implementation of the logistics planning system the agentsô functionalities were tested individually and collectively using the decision table shown in Table 6 below and the system met the expected results.

NO.	Component Tested	Purpose	Condition	Test Data	Expected Results
1	Login	Authenticate user to generate plan	An interface BeliefBase with usernames and passwords	Username and password	Sign in user
2	estimatesManager	Generate operations requirement estimates	Logged in Duration Size of force	Items Items requirement/day Duration	Generate estimates for items required
3	suppliesManager	Find optimum suppliers for items	Operation Requirements	Quantity of goods List of suppliers	Select suppliers with optimum cost
4	transportManager	Find optimum means of transport for goods and persons	Size of force, Quantity of goods to be shipped	Quantity of goods, number of people, Transport belief base	Select affordable and suitable transporter
5	planGenerationManager	Generate and display plan	Operation requirements Suppliers Transporters	Items Suppliers Transporters	Display plan

Table 6: Testing Matrix

4.4 Evaluation

Independent review of the implemented logistics planning system prototype was conducted by respondents drawn from a research and development department in state organization. The respondents were professionals with sufficient knowledge for evaluation but without prior knowledge of the design and implementation of the system. This was as a result of the high challenges faced in an attempt to conduct evaluation in the military logistics planning formation due to sensitivity and secrecy of real data. Appendix C has the sample questionnaire presented to the respondents.

4.4.1 Evaluation Tool: System Usability Scale

The System Usability Scale (SUS) is a simple, ten-item scale giving a global view of subjective assessments of usability (Brooke, 1996). It uses a Likert scale ranging from 1 to 5 where 1 = Strongly Disagree and 5 = Strongly Agree.

The ten items on the scale are:

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly 15
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

Sample questionnaire of SUS method can be found in Appendix C

Brooke instructs the use of questionnaire as follows: ñTo calculate the SUS score, first sum the score contributions from each item. Each item's score contribution will range from 0 to 4. For items 1, 3, 5, 7, and 9 the score contribution is the scale position minus 1. For items 2,4,6,8 and 10, the contribution is 5 minus the scale position. Multiply the sum of the scores by 2.5 to obtain the overall value of SUò.

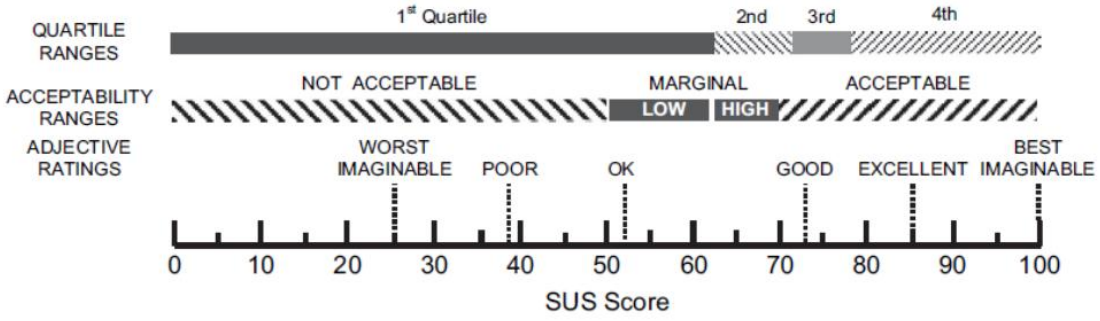

Figure 19: A comparison of mean System Usability Scale (SUS) scores by quartile, adjective ratings, and the acceptability of the overall SUS score" (Bangor et al. 2008 p. 592)

Table 7: Table with Score Contributions from each respondent

Item No	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10
1	4	3	4	5	3	4	3	5	5	4
2	2	1	1	2	1	2	3	2	1	1
3	5	4	3	4	5	3	3	4	3	5
4	1	2	1	2	2	1	1	3	3	2
5	5	4	5	4	3	4	4	3	4	3
6	2	1	1	2	1	2	1	3	1	1
7	5	4	3	4	5	3	2	4	4	4
8	2	3	1	1	3	1	1	2	3	1
9	4	4	5	3	4	5	2	3	3	3
10	2	1	1	2	1	2	3	1	1	3

Table 8: Table with SUS score from each respondent

Item No	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10	Total	Ave (Total/10)
1	3	2	3	4	2	3	2	4	4	3	30	3
2	3	4	4	3	4	3	2	3	4	4	34	3.4
3	4	3	2	3	4	2	2	3	2	4	29	2.9
4	4	3	4	3	3	4	4	2	2	3	32	3.2
5	4	3	4	3	2	3	3	2	3	2	29	2.9
6	3	4	4	3	4	3	4	2	4	4	35	3.5
7	4	3	2	3	4	2	1	3	3	3	28	2.8
8	3	2	4	4	2	4	4	3	2	4	32	3.2
9	3	3	4	2	3	4	1	2	2	2	26	2.6
10	3	4	4	3	4	3	2	4	4	2	33	3.3
Total	34	31	35	31	32	31	25	28	30	31	308	30.8

Total Average score =30.8

SUS Score=30.8*2.5=77

4.4.2 Discussion

Table 7 shows the scores of each of the ten-item scale from the ten respondents. Table 8 shows the SUS scores of the same items. According to Bangor *et al* (2008) ratings shown on Figure 20 above, the logistics planning system prototype with a SUS score of 77 can be considered a good product and hence passable. However, this score also indicates that further adjustments can be done on the system prototype to elevate it to either best or superior ratings with scores of above 80 and 90 respectively. From the average score of each item, the items with score of less than 3 should be relooked unto for adjustments in the future versions of this product.

5 CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This study involved theory review, design and analysis, implementation, testing and evaluation of the proposed logistics planning system prototype based on the BDI agent architecture. It had the following as main objectives:

a. How do you represent the proposed logistics planning system as a Multi-Agent System with BDI agent architecture?
b. Which is the best technology to implement a logistics planning system prototype for experimentation?
c. What are the results of the prototype testing and evaluations?

The theory review established the characteristics of logistics planning problem. Logistics planning problems are highly complex, have large decision space, they utilize real-time data, they are quite uncertain, they involve many decision makers, they are highly constrained and they occur in distributed domains. The inherent MAS characteristics like autonomy, reactivity, proactivity, decentralized organization, environment abstraction, social organization and social abilities will greatly benefit a logistics planning problem.

The traditional programming paradigms for instance command oriented, function oriented and OO do not have sufficient abstraction to adequately define and programa logistics planning system. The Agent oriented paradigm stands out as the best with the best structures to fill this gap. The addition of BDI model makes the definition of a logistics planning system even better. This model describes an agent as a goal-directed entity that acts in a rational manner. Agents have defined goals to achieve (desires), a set of plans (intentions) to describe how to achieve the goals and a set of data called beliefs which describe the state of the environment in which the system is running.

5.2 Summary of Achievements

Logistics planning problem was defined as a Multi-Agent system with BDI agent model. The bigger logistics planning problem was decomposed into multiple agents with Beliefs, Desires and Intentions which are building blocks of an agent system designed for uncertain dynamic world. This was made possible using the BDI-ASDP methodology.

The Logistics planning system design was implemented in Jason Agent programming language which provides support for BDI agent model and evaluated using System Usability Scale (SUS) method.

5.3 Choice of Design Methodology

There are a number of agent-based methodologies out there for instance Gaia and MaSE but none of them proved to be simple and efficient method to design and analyse an agent based software like the DBI-ASDP methodology. In this methodology, the belief-desire-intention (BDI) agent is employed to define the logistics planning system. It uses the existing proven methods and tools used in object-oriented modelling techniques with some refinement, extensions to adapt them into the BDI agent-based software construction. It provides a sound, realistic and practical modelling for an agent-oriented software development. However the methodology does not support full-cycle software development process. The aspects like software testing and evaluation are missing in the architecture.

5.4 Choice of Technology

There was an early contemplation to usecommand oriented, function oriented, OO and ORas alternative methodologies to solve the logistics planning problem. OR methodology uses algorithms, simulation, modelling and queuing techniques in problem solving. However this methodology was not without drawbacks. For a logistics planning problem this methodology will result in high computational requirements. The methodology deals only with quantifiable factors and not all logistics planning factors can be quantified. It would have required a lot of time and cost to implement this methodology. Command oriented, function oriented and OO together they lack abstraction and structures to concisely define any logistics planning problem.

Besides the above discussed methodologies, there is agent oriented paradigm. This paradigm decomposes any system based on key roles. This decomposition is followed by the identification of roles and their relationships and this guides the specification of an agent class hierarchy.

The technology that was adopted in the implementation of the logistics planning system was open-source Jason Agent Programming language. It was selected because if provides a concise and powerful notation to program multi-agent systems based on BDI model. It is an agent-oriented programming language and therefore it has the necessary abstraction to define and implement the system. However, there are a number of limitations to the use of Jason:

- The integration of PROLOG into Jason for the belief base while flexible, it results in a challenge of executing queries into the belief base. The queries are only successful for the belief bases that have been created from the Jason environment.
- Jason lacks adequate support given that it is not popular and mostly used for academic purposes. The examples, demo files and documentation that is part of Jason package is not a sufficient support for serious programming. The online forum is not proactive and it takes time before your concerns are addressed. This results in long learning curves for Jason newbies.

Despite these limitations, Jason is still a promising approach to programming logistics planning systems.

5.5 Comments

Individual agent testing and overall system testing has proved that the system has met the specified user requirements. This implementation has shown that BDI agent architecture can provide solutions to an otherwise long, tiresome and sophisticated manual logistics planning process. The distributed nature of the prototype due to JADE platform means that logistics planning can be done in an environment where components and players are geographically distributed.

The study has also shown that it is possible to model logistics planning problem using the BDI architecture. Given that the BDI agent architecture inherits most of the multi-agent properties for instance the fact that agents are continually running and exchanging information where necessary and when applicable. The BDI agent architecture modularizes the system making it easy to understand code and maintain. The BDI agent architecture is comparable to Model-View-Control (MVC) framework employed in other programming paradigms.

The fact that Jason platform is based on Java implies that it can be extended and be made as complex as it can possibly be. Jason has being tailored for multi-agent programming making the agent programming process quick and simple. You require less code to achieve module in Jason as compared to other non-agent programming platforms. The platform independence nature of Jason is a plus as it possible to realize a multi-agents system with individual agents residing in different platforms. Jason being open-source provides source codes for enhanced customizations.

5.6 Future Work and Recommendations

The current approach makes use of a library of plans as intentions, a text based and relational database persistent belief bases as beliefs and clearly defined goals as desires. However, the system can be made more dynamic by giving it the ability to create plans dynamically and using these plans to achieve defined goals.

The system can also be extended to allow bidding and finally selecting an optimal transporter or supplier. The current implementation relies on plans that select an optimal supplier or transporter from a persistent belief base based on their costing.

Work is also underway to make the system fully distributed. This can be possible by replacing the text based interface with a GUI interface. The GUI interface may be browser based, desktop based or mobile devices based. This will enable users to access the logistics planning system from different platforms. This will enable suppliers to bid to supply items required for an operation. This will also enable external transporters to bid to transport the required items to an operation area.

REFERENCES

1. Alexander, P., Lars, B., Jan, S., R, W., & L, W. (n.d.). Simulation and Implementation of Logistics Systems based on Agent Technology.

2. Bangor, A., Kortum, P., & Miller, J. (2008). An Empirical Evaluation of the System Usability. *International Journal Of Human-Computer Interaction*, Vol. 24, No. 6, pp. 574-594.

3. Bratman, M. E. (1987). *Intentions, Plans and Practical Reasoning.* Cambridge: Havard University Press.

4. Brook, J. (1996). SUS - A quick and dirty usability scale. Available from http://cui.unige.ch/isi/icle-wiki/_media/ipm:test-suschapt.pdf.

5. CiteMAN. (2007). Limitations of operations research. *CiteMAN*, http://www.citeman.com/1913-limitations-of-operations-research.ht ml .

6. Creveld, M. V. (1977). *Supplying War.* Cambridge: United Kingdom: Cambridge University Press.

7. Davidson, I., & Kowalczyk, R. (1997). *Towards Better Approaches to Decision Support in Logistics, Industrial Logistics.*

8. Dorer, K., & Calisti, M. (2005). An adaptive solution to dynamic transport optimization AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. *ACM*, 45-51.

9. Gavin, B. R. (Feb 2010). *A Belief-Desire-Intention Architecture with A Logic-Based Planner for Agents in Stochastic Domains.*

10. Georgeff, M., Pell, B., Pollack, M., Tambe , M., & Wooldridge, M. (1999). *The Belief-Desire-Intention Model of Agency, Proceedings of the 5th International Workshop on Intelligent Agents V: Agent Theories, Architectures and Languages.*

11. Hans-Jurgen, B., Klaus, F., & Gero, V. (1997). *TeleTruck: A Holonic Fleet Management System.*

12. Janis , G., & Egons, L. (2003). *Multiagent Based Simulation Tool for Transportation and Logistics Decision Support.*

13. Jo, C. H. (2001). A Seamless Approach to the Agent Development. *ACM 2001 15th Annual symposium on Applied Computing (ACM)*, 641-647.

14. Monostori, L., Vancza, J., & Kumara, S. R. (2006). *Agent-Based Systems for Manufacturing* (Vol. Vol 55/2/2006). Annals of the CIRP.

15. Nitin, Y., Chenguang, Z., Sebastian, S., & Raiph, R. (2010). *A BDI agent system for the cow herding domain.* Springer Science+Business Media B.V.

16. Perugini, D., Zschorn, A., Lambert, D., Sterling, L., & Pearce, A. (2003). Agents in logistics planning, experiences with coalition agents experimental project. Agents at Work AAMAS Workshop.

17. Priyanka. (2012). Disadvantages of Operations Research. http://www.ehow.com/info_8480135_disadvantagesoperations-research.html.

18. Rao, A., & Georgeff, M. (1995). BDI agents: From theory to practice. *In Proc. of ICMAS-95*, 312-319.

19. Serugendo, G., Gleizes, M., & Karageorgos, A. (2006). Self-Organisation and Emergence in MAS: An Overview.

20. Traillandier, P., Oliever , T., & Benoit, G. (2012). A new BDI agent architecture based on the belief theory. Application to the modelling of cropping plan decision-making. *PRIMA*.

21. UniversalTeacher. (2010). Advantages and Limitations of Operations Research. http://www.universalteacherpublications.com/univ/ebooks/or/Ch1/advlim.htm.

22. Viroli, M., Holvoet, T., Ricci, A., Shelfthout, K., & Zambonelli, F. (2007). *Infrastructures for the environment of multiagent systems, Autonomous Agents.*

23. Vitasek, K. (2013). Supply Chain Management Terms and Glossary.

24. Wooldridge, M. J. (2002). An introduction to multiagent systems. John Wiley & Sons, Chichester,. *John Wiley & Sons, Chichester,*.

APPENDIX A: SAMPLE CODE

Sample code for ***home*** agent:

```
// Agent test in project logplanner

/* Initial beliefs and rules */
operation_count(0).

/* Initial goals */

!start.

/* Plans */

+!start:true
<-
        .send(login,askOne,user_logon(patrick,"patrick"),Ans);
        .print("Login = ",Ans);
Ans \== false; // only continues if Ans was ok
.print("Start operation and insert its details");
.send(dbmanager,tell,operation(1,"Mpeketoni Op","Lamu County","Kenya",2,30,"12-08-2014"));
        .send(dbmanager,tell,operation(2,"Nairobi Op","Nairobi County","Kenya",2,30,"30-08-2014"));
        .send(dbmanager,tell,force_structure(1,"Section","9-10"));
        .send(dbmanager,tell,force_structure(2,"Platoon","16-44"));
        .send(dbmanager,tell,force_item_req(1,2,1,9));
        .send(dbmanager,tell,force_item_req(2,2,2,4));
        .send(dbmanager,tell,force_item_req(3,2,3,5));
        .send(dbmanager,tell,force_item_req(4,2,4,10));
        .send(dbmanager,tell,item(1,"Flour",1,1));
        .send(dbmanager,tell,item(2,"Meat",1,1));
        .send(dbmanager,tell,item(3,"Potatoes",1,1));
        .send(dbmanager,tell,item(4,"Milk",1,2));
        .send(dbmanager,tell,item(5,"Eggs",1,3));
        .send(dbmanager,tell,item(6,"Onion",5,1));
        .send(dbmanager,tell,item(7,"Petrol",2,2));
        .send(dbmanager,tell,item(8,"Diesel",2,2));
        .send(dbmanager,tell,item(9,"Firewood",2,1));
        .send(dbmanager,tell,item(10,"Carrot",5,1));
        .send(dbmanager,tell,item(11,"Sugar",1,1));
```

```
        .send(dbmanager,tell,item(12,"Tea",6,1));
        .send(dbmanager,tell,item(13,"Salt",7,1));
        .send(dbmanager,tell,item(14,"Gas",2,2));
        .send(dbmanager,tell,item_type(1,"Food"));
        .send(dbmanager,tell,item_type(2,"Fuel"));
                .send(dbmanager,tell,item_type(3,"Ammo"));
        .send(dbmanager,tell,item_type(4,"Others"));
        .send(dbmanager,tell,item_type(5,"Ingredients"));
        .send(dbmanager,tell,item_type(6,"Beverages"));
        .send(dbmanager,tell,item_type(7,"Food Additives"));
        .send(dbmanager,tell,unit(1,"Kilograms","Kgs"));
        .send(dbmanager,tell,unit(2,"Litres","Lts"));
        .send(dbmanager,tell,unit(3,"Crates",""));
        .send(dbmanager,tell,supplier(1,"EverFresh Ltd","Nairobi County","P.O. Box 89767
(0300)","Nairobi","(020) 09876","info@evergreen.co.ke"));
        .send(dbmanager,tell,supplier(2,"Integrity Groceries Ltd","Thika County","P.O. Box 8767
(0300)","Thika","(020) 09876","customer@integrocery.co.ke"));
        .send(dbmanager,tell,supplier(3,"Limuru Fresh Suppliers Ltd","Nairobi County","P.O. Box 89767
(0300)","Nairobi","(020) 09876","info@evergreen.co.ke"));
        .send(dbmanager,tell,supplier(4,"Prompt Suppliers Ltd","Nairobi County","P.O. Box 89767
(0300)","Nairobi","(020) 09876","info@prompt.co.ke"));
        .send(dbmanager,tell,supplier(5,"Efficient Suppliers Ltd","Limuru","P.O. Box 89767
(0300)","Limuru","(020) 09876","jgrace@efficientsups.co.ke"));
        .send(dbmanager,tell,supplier(6,"Quality Suppliers Ltd","Kiambu County","P.O. Box 89767
(0300)","Kiambu","(020) 09876","john@qualitysups.co.ke"));
        .send(dbmanager,tell,supplier_item(1,1,1,140.0));
        .send(dbmanager,tell,supplier_item(2,1,2,400.0));
        .send(dbmanager,tell,supplier_item(3,2,3,100.0));
        .send(dbmanager,tell,supplier_item(4,2,4,50.0));
        .send(dbmanager,tell,supplier_item(5,3,5,400.0));
        .send(dbmanager,tell,supplier_item(6,3,6,150.0));
        .send(dbmanager,tell,supplier_item(7,4,7,100.0));
        .send(dbmanager,tell,supplier_item(8,4,8,100.0));
        .send(dbmanager,tell,supplier_item(9,5,9,100.0));
        .send(dbmanager,tell,supplier_item(10,5,10,30.0));
        .send(dbmanager,tell,supplier_item(11,6,11,130.0));
        .send(dbmanager,tell,supplier_item(12,6,12,100.0));
        .send(dbmanager,tell,supplier_item(13,1,13,30.0));
        .send(dbmanager,tell,supplier_item(14,2,14,400.0));
        .send(dbmanager,tell,supplier_item(15,3,2,380.0));
```

```
        .send(dbmanager,tell,supplier_item(16,5,5,350.0));
        .send(dbmanager,tell,supplier_item(17,4,14,379.0));
        .send(dbmanager,tell,supplier_item(18,1,4,55.0));
        .send(dbmanager,tell,supplier_item(19,6,3,80.0));
        .send(dbmanager,tell,supplier_item(20,4,1,130.0));
        .send(patrick,askOne,force_item_reqs(2,L),force_item_reqs(2,L));
        .print(L);
        .print(" (----Name -Type -Quantity -Units---)");
        !show_items(L).

        +!show_items([]).
        +!show_items([C|R])
        <-.send(dbmanager,askOne,req_details(C,B),req_details(C,B));

.print("  - ",B);
        !show_items(R).
```

APPENDIX B:SYSTEM SCREEN SHOTS

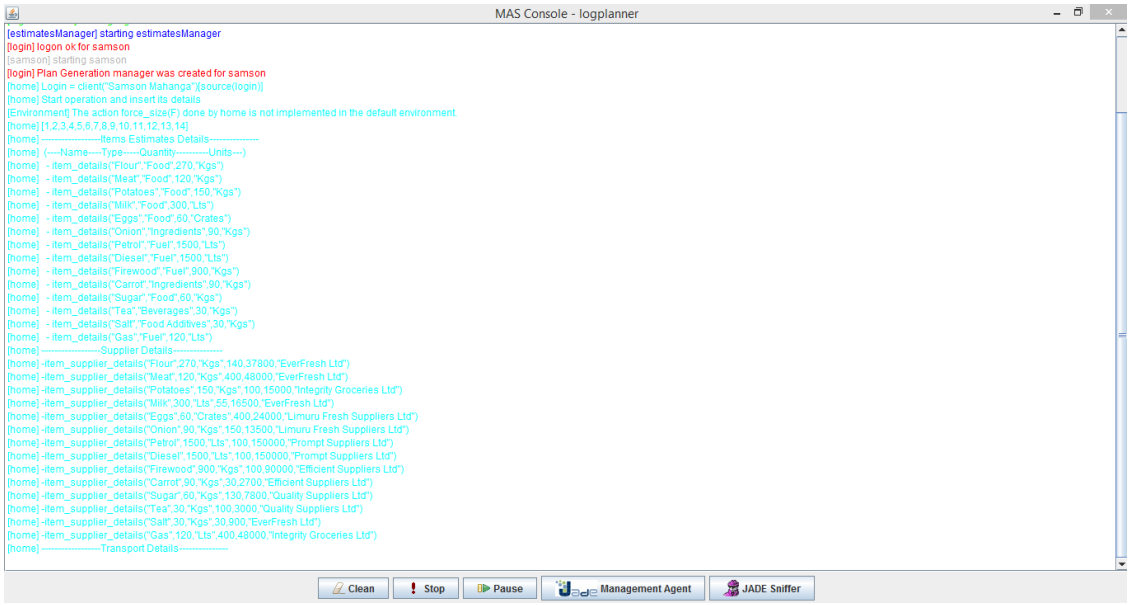

Figure 20: System in Jason Screenshot

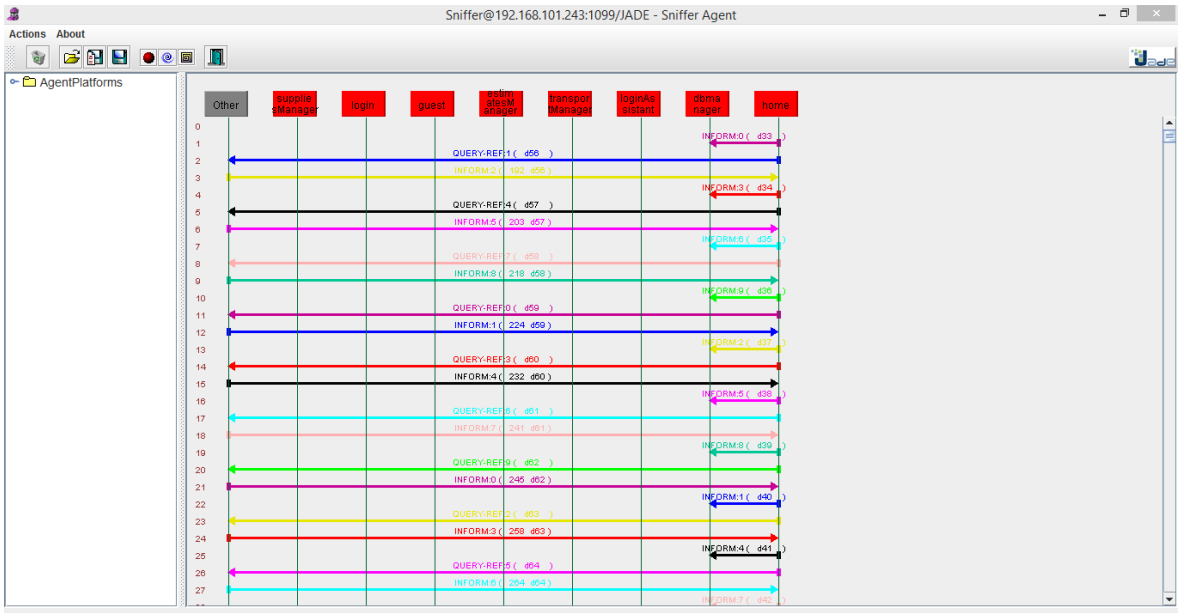

Figure 21: Agents Interaction Screenshot

APPENDIX C: SAMPLE QUESTIONNAIRE

System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5