AGENT COMMUNICATION AND NEGOTIATION IN A SUPPLY CHAIN

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Rajat Upadhyay

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

March 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

AGENT COMMUNICATION AND NEGOTIATION IN A SUPPLY CHAIN

By

Rajat Upadhyay

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall E. Nygard
Chair

Dr. Simone Ludwig

Dr. Limin Zhang

Approved:

April 08, 2013	Dr. Brian Slator
Date	Department Chair

ABSTRACT

Supply Chain Management is an essential management paradigm for almost every organization. Effective implementation of Supply Chain Management (SCM) can significantly increase an organization's profit. In this paper, a simple SCM is implemented using multi-agent modeling. A typical SCM consists of the supplier, manufacturer, inventory, seller, and customer. A multi-agent system provides a natural solution for SCM because various entities involved in SCM can be represented as intelligent agents. In the present approach various agents representing various entities of SCM, negotiate with each other in order to achieve their goals.

Apart from the entities associated with typical SCM, additional entities have been added to the proposed system. This system consists of six agents representing the SCM entities: Customer Agent, Seller Agent, Coordinator Agent, Inventory Agent, Manufacturer Agent, and Supplier Agent. The system also consists of other agents to initialize the system and the database.

ACKNOWLEDGEMENTS

I am heartily thankful to my adviser, Dr. Kendall Nygard, whose encouragement, guidance, and support from the initial to the final level enabled my work to be more organized and achievable.

I would like to sincerely thank my supervisory committee members, Dr. Simone Ludwig and Dr. Limin Zhang, for being on my committee and for providing their help and support.

Finally, I would like to give special thanks my family and loved ones for their support throughout my education. Without their help and encouragement, I would not be here to pursue my degree.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER 1. INTRODUCTION 1

1.1. Simple Supply Chain 1

1.2. Agent Roles and Responsibilities 2

1.2.1. Customer Agent 2

1.2.2. Seller Agent 2

1.2.3. Inventory Agent 3

1.2.4. Coordinator Agent 3

1.2.5. Manufacturer Agent 3

1.2.6. Supplier Agent 3

1.2.7. Persistence Agent 4

1.2.8. Email Agent 4

CHAPTER 2. LITERATURE REVIEW 5

CHAPTER 3. DESIGN AND DEVELOPMENT 7

3.1. Java Agent Development Framework (JADE) 7

3.1.1. JADE History 7

3.1.2. JADE Architecture 7

3.1.3. Creating Agents 8

3.1.4. Agent Identifiers 8

3.1.5. Agent Communication 8

3.1.6. Agent Discovery: The Yellow Pages Service 10
3.1.7. Content Language 11
3.1.8. Ontology 11
3.2. System Requirements 11
3.2.1. Functional Requirements 11
3.2.2. Non-Functional Requirements 15
3.2.3. Interface Requirements 15
3.3. System Design 16
3.3.1. High-Level Use-Case Diagrams 16
3.3.2. Activity Diagram 22
3.3.3. Class Diagram 24
3.3.4. Sequence Diagram 29
3.3.5. Low-Level Design Details 30
3.4. System Testing 41
3.5. Screen Shots 50
CHAPTER 4. CONCLUSION 53
REFERENCES 55

LIST OF TABLES

Table Page

1. Functional Requirements of the Application 12

2. Non-Functional Requirements of the Application 15

3. Interface Requirements of the Application 16

4. Methods of SCMOntology Class 30

5. Methods of Persistence Class 31

6. Methods of RecieveAndSaveQueries Class 31

7. Methods of StaticSql Class 32

8. Methods of CoordinatorAgent Class 32

9. Methods of Coordinate Class 33

10. Methods of InventoryAgent Class 33

11. Methods of ManageSellerInventory Class 34

12. Methods of AbstractVendorAgent Class 34

13. Methods of CustomerAgent Class 34

14. Methods of CustomerStartNegotiation Class 35

15. Methods of ManufactureAgent Class 35

16. Methods of Manufacture Class 36

17. Methods of SupplierAgent Class 36

18. Methods of Negotiate Class 37

19. Test Cases 41

LIST OF FIGURES

Figure Page

1. JADE Architecture 8

2. Agent Communication 9

3. The Yellow Pages Service. 10

4. Use-Case Diagram Illustrating the Customer Agent's Actions. 17

5. Use-Case Diagram Illustrating the Seller Agent's Actions. 18

6. Use-Case Diagram Illustrating the Persistence Agent's Action. 19

7. Use-Case Diagram Illustrating the Inventory Agent's Actions. 20

8. Use-Case Diagram Illustrating the Coordinator Agent's Actions. 21

9. Use-Case Diagram Illustrating the Manufacturer Agent's Action. 22

10. Activity Diagram Showing the Negotiation Flow. 23

11. Class Diagram Showing the AgentMonitor Class and Its Inner Classes. 24

12. Class Diagram Showing the Classes Responsible for Transactions. 25

13. Class Diagram Showing Agents in the Marketplace. 26

14. Class Diagram Showing the Agents Responsible for Sending Emails. 27

15. Class Diagram Showing the Coordinator and Inventory Agents. 28

16. Class Diagram Showing the Persistence Agent and the Classes that Are Responsible for Accessing the Database. 28

17. Sequence Diagram of the System. 29

18. WSIG Architecture. 39

19. The Website's Home Page. 50

20. Customer Details' Entry Page. 50

21. Seller, Manufacturer, and Supplier Creation Page. 51

22. JADE Framework's Remote Agent Management GUI Showing Agents in the Active State..... 51

23. Sniffer Agent's Message Box GUI. 52

CHAPTER 1. INTRODUCTION

New scientific development, such as programming languages and paradigms, typically emerge from the ideas and terms that are prevalent at that time, and multi-agent systems are no exception. Multi-agent systems arrive from the five basic trends in the history of computing: ubiquity, interconnection, intelligence, delegation, and human orientation.

An agent is a computer system that is situated in some environment and is capable of autonomous action in this environment on behalf of its owner in order to meet its design objectives. The agent takes input from its environment and produces output against that input. Multi-agent systems are comprised of such intelligent agents and their environments, and the key issue here is for the agents cooperate, coordinate, and negotiate as humans do.

1.1. Simple Supply Chain

A simple supply chain consists of entities such as the seller, manufacturer, and supplier. These entities are individual organizations that interact with each other to move the product from the manufacturer to the end customer. In order to meet the market requirements and to balance the supply and demand, it is very important for these entities to effectively find the other supply-chain entities. Also, it is important for these entities to effectively negotiate with each other, under predefined criteria, so that each entity can make profit.

In this paper, a simple supply chain is implemented using intelligent agents. This system represents a worldwide network of customers, sellers, manufacturers, and suppliers. These entities are represented by the intelligent agents. The agents discover other agents that are selling, manufacturing, or supplying the desired product and then perform the negotiation.

1.2. Agent Roles and Responsibilities

This section describes the roles and responsibilities of various agents that are implemented in this system.

1.2.1. Customer Agent

The customer agent is responsible for buying the final product on the behalf of the human customer. A human customer can provide details about the product, desired price, time limit under which the customer agent is supposed to buy the product, and the email address. The time-limit factor does not play any role in the negotiation but defines the agent's lifetime. If the agent is not able to find a seller agent or is not able to perform negotiation within this time, it will die without buying the product. A web-based user interface is provided for entering the buying product details.

Customer agents get the list of seller agents that are selling a desired product from the Directory Facilitator Agent (DF) and then start negotiation by sending the proposal and counter proposals.

1.2.2. Seller Agent

The seller agent is responsible for selling the final product on behalf of a human seller. The human seller can provide details about the product, maximum selling price, minimum selling price, time limit under which agent is supposed to sell the product, and the email address. Similar to the customer agent's time limit, the seller agent also has a time limit which is known as its life time. Once this time limit is over, the seller agent will die. A web-based user interface is provided to enter the product details.

The seller agent receives the proposal from the customer agents. Negotiation takes place in the form of a proposal and counter proposals.

1.2.3. Inventory Agent

The inventory agent keeps track of the available products; also, it maintains the inventory in such a way that product availability does not increase or decrease from a given marker. Every time that a seller agent sells a product, the persistence agent updates the database, and the inventory agent checks the inventory status. If the current inventory is lower than the defined limit, it requests that the coordinator agent purchase the product from the manufacturer.

1.2.4. Coordinator Agent

The coordinator agent is contacted by the inventory agent once the inventory level goes below the predefined limit. The coordinator agent discovers all the manufacturers selling the desired product and sends a proposal to them. Once the coordinator agent and the manufacturer agent agree on a price, negotiation is completed.

1.2.5. Manufacturer Agent

The manufacturer agent is involved with the product's manufacturing process. The manufacturer agent gets the list of suppliers that are selling the desired raw material from the DF agent. A graphical user interface is provided to enter the desired raw material, raw material quantity, minimum price, maximum price, production quantity, and the time limit after which the predefined number of products will be produced periodically.

1.2.6. Supplier Agent

Supplier agents provide the raw materials to manufacturer agents. Supplier agents receive proposals from the manufacturer agents, and after a negotiation, the raw material is sold. A graphical user interface is provided to enter the raw material's details.

1.2.7. Persistence Agent

The persistence agent is responsible for storing and retrieving all information to and from the database. Every time the seller agent, manufacturer agent, and supplier agent sell or buy a product, the persistence agent is informed, and it updates the database.

1.2.8. Email Agent

The email agent is responsible for sending the email to the agent's owner. Whenever a negotiation is finished, the result of that negotiation is emailed to the agents' owners.

CHAPTER 2. LITERATURE REVIEW

This chapter presents the work done in the area of multi-agent systems and supply chains using multi-agent systems.

A multi-issue negotiation protocol for a one-buyer-to-many-sellers interaction which suits the electronic commerce framework is presented [1]. In their approach, the authors use the traditional Consumer Buying Behavior (CBB) model. This paper identifies three broad topics for research on negotiation. Those topics are *negotiation protocol (set of rules that govern the interaction), negotiation objects (range of issues over which agreement must be reached),* and *agents' reasoning models (decision-making apparatus by which participants attempt to achieve their objectives)*.

Misra et al. [2] present a survey of implemented supply-chain systems and presents the requirements for the next generation of supply-chain management systems. Also, this paper discusses the six characteristics of the supply-chain management philosophy:

- Shared Information
- Organizational Relationship
- Inventory Management
- Total Pipeline Coordination
- Readiness to adopt Flexibility
- Costing Issues

In a paper [3], the authors suggest a new negotiation algorithm in the SCM environment that uses multi-agent technology. The algorithm is based on multiple factors, such as price review point and delivery point. The suggested system is implemented using the JADE framework.

A paper [4] explores the similarities between the multi-agent system and the supply-chain system to justify using multi-agent technology as an appropriate approach to support supply-chain collaboration. This paper also presents a framework for an agent-based collaborative supply-chain system.

Magrid et al. [5] propose a methodology for developing multi-agent systems using the JADE platform. The proposed methodology focuses on the key issues in the analysis and design of a multi-agent system. The proposed methodology does not extend the object-oriented techniques, instead focusing on agents specifically and the abstractions provided by the agent paradigm. It combines a top-down and bottom-up approach so that both existing system capabilities (including those provided by legacy software and people) and the application's overall needs (based on the requirements) can be accounted for.

A paper [11] looks at the supply-chain system as a group of companies where every company in the supply chain has its own interests and goals even though the companies may also have intentions to deal with each other. The presented work also examines the problem of modeling agent cooperation as a pure scheduling problem, and in a real business environment, there are no obligations for companies to remain with a supply chain for a certain time period. To address these problems, a framework is presented; there is no preset relationship between functional agents. When an order comes, a virtual supply chain may emerge through negotiation processes. Even after the order has been accepted, the components of the chain may change according to the external situation.

CHAPTER 3. DESIGN AND DEVELOPMENT

The purpose of this chapter is to present the functional and non-functional requirements. Also, this chapter provides the use case diagrams, class diagram, activity diagram, and sequence diagram.

3.1. Java Agent Development Framework (JADE)

JADE is a Java-based software framework for implementing intelligent agents. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that support the debugging and deployment phases. The agent platform can be distributed across machines (which do not need to share the same operating system), and the configuration can be controlled via a remote graphical user interface (GUI). The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. JADE is completely implemented in the Java language, and the minimal system requirement is version 1.4 of JAVA (the run-time environment or the JDK).

3.1.1. JADE History

The JADE project was started by Telecom Italia in late 1998 and was motivated by the need to validate the early FIPA specifications. JADE went open source in 2000 and was distributed by Telecom Italia under the Library Gnu Public License (LGPL) license [6].

3.1.2. JADE Architecture

The JADE platform is composed of containers, and these containers can be distributed over the network. As shown in Figure 1, the Main Container represents the boot strap. It is the first container to be launched, and all other containers must join a main container by registering with it [6].

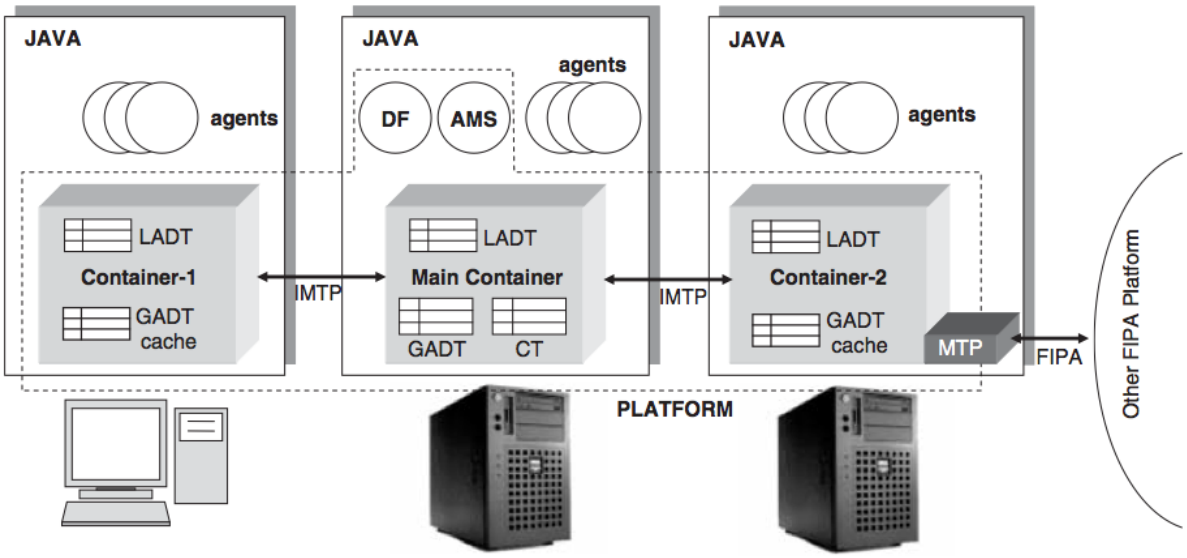

Figure 1. JADE Architecture.

3.1.3. Creating Agents

The JADE framework provides an agent class which can be extended in order to create an agent class. The agent class provides common behaviors that an agent should have according to the FIPA specification. The agent class is present in the *jade.core* package [6].

3.1.4. Agent Identifiers

In accordance with FIPA, each agent instance is identified by an agent identifier. In JADE, an agent identifier is represented as an instance of the *jade.core.AID* class [6].

3.1.5. Agent Communication

Agent communication in JADE is implemented in accordance with FIPA specifications. The communication paradigm is based on *Asynchronous message passing*. Each agent has a mailbox where the JADE runtime posts messages sent by other agents [6].

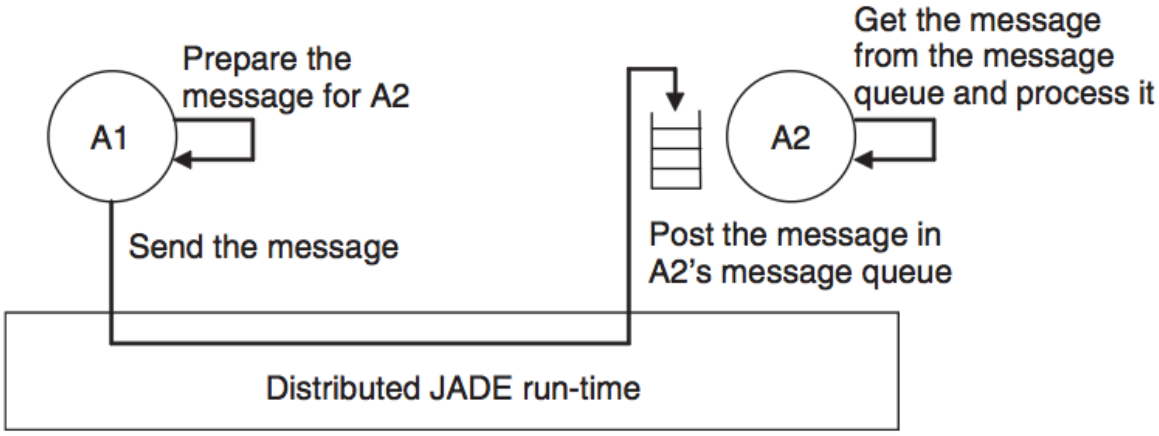

Figure 2. Agent Communication.

As shown in Figure 2, agent A1 prepares the message and sends it to JADE run-time. JADE run-time posts that message to agent A2's mailbox (message queue). Each message includes [6]

- The sender of the message
- The list of receivers
- The communicative act (also called the "performative")
- The content
- The content language
- The ontology

Agent A2 receives the message by calling the receive() method provided by the agent class.

3.1.6. Agent Discovery: The Yellow Pages Service

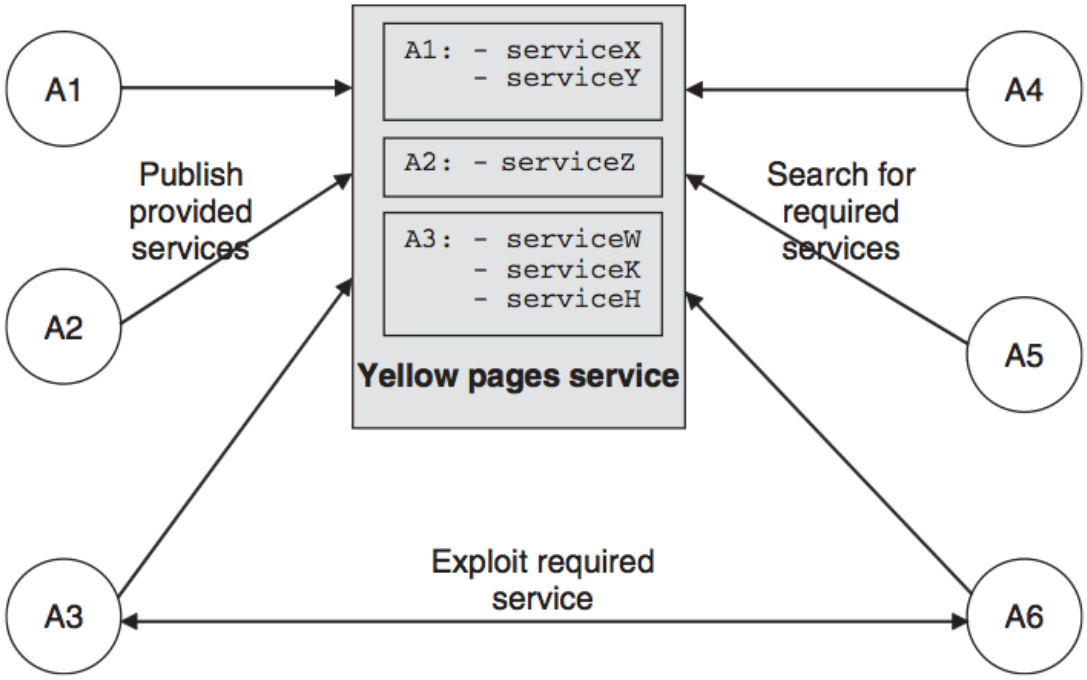

Figure 3. The Yellow Pages Service.

As shown in Figure 3, a "yellow pages" service allows agents to publish descriptions of one or more services that they provide so that other agents can easily discover and exploit the services. Any agent can both register (publish) services and search for (discover) services. Registrations, deregistration, modifications, and searches can be performed at any time during an agent's lifetime. The yellow pages service in JADE, in accordance with the FIPA Agent Management specification, is provided by a specialized agent called the Directory Facilitator (DF). Every FIPA-compliant platform should host a default DF agent (where the local name is "df@<platform-name>"). Other DF agents can be deployed, if required, and several DF agents (including the default) can be federated to provide a single, distributed yellow-pages catalog [6].

3.1.7. Content Language

According to FIPA, the value of the content slot should be a string or raw sequence bytes. In order to communicate more complex information, such as an object, it is necessary to adopt a well-defined syntax so that the receiver can parse the message. This syntax is known as the content language. FIPA defines and recommends SL language to be used when communicating [6].

3.1.8. Ontology

When SL syntax is received by the receiver agent, it must have a shared understanding of the concept and symbols. This set of concepts and symbols is known as ontology. Each time a message is exchanged:

- The sender needs to convert its internal representation to the corresponding ACL content's expression representation, and the receiver needs to perform the opposite conversion.
- The receiver should perform a number of semantic checks to verify that the received information complies with the ontology rules shared by the communicating agents.

3.2. System Requirements

System requirements are divided into functional requirements, non-functional requirements, and interface requirements.

3.2.1. Functional Requirements

Functional requirements capture the expected behavior of all the agents described in Section 1.2 (Table 1):

- Commit (C): Requirements that will be implemented.
- Target (T): Requirements that will targeted but not guaranteed.

Table 1. Functional Requirements of the Application

REQ-ID	Description	C	T
FR-1	Monitor agent must launch email agent after its successful launch	X	
FR-2	Monitor agent must launch persistence agent after its successful launch	X	
FR-3	Monitor agent must launch all vendor agents available in the database after its successful launch	X	
FR-4	Supplier agent must replenish the raw-material inventory at the given time.	X	
FR-5	Supplier agent must replenish the raw-material inventory with the given quantity.	X	
FR-6	Manufacturer agent must produce the product at the given time.	X	
FR-7	Manufacturer agent must discover all the supplier agents supplying the required raw materials	X	
FR-8	Manufacturer agent must negotiate with all available supplier agents supplying the required raw material	X	
FR-8.1	Manufacturer agent must send a proposal to the known supplier agents.	X	
FR-8.2	Supplier agent must send a counter proposal to the manufacturer agent if the proposal from the manufacturer is less than the supplier's minimum price.	X	
FR-8.3	Manufacturer agent must send a counter proposal to the supplier agent if the supplier agent's proposal is greater than the manufacturer's maximum price.	X	

(continued)

Table 1. Functional Requirements of the Application (continued)

REQ-ID	Description	C	T
FR-8.4	Supplier agent must accept the proposal from the manufacturer agent if the manufacturer agent's counter proposal is within the supplier's price range.	X	
FR-8.5	Supplier agent must reject the manufacturer's proposal if the proposal is lower than the supplier's minimum price.	X	
FR-8.6	Supplier agent must send an email to the supplier after completing the negotiation, by sending a request to an email agent.	X	
FR-8.7	Manufacturer agent must send an email to the manufacturer after completing the negotiation, by sending a request to an email agent.	X	
FR-9	Coordinator agent must negotiate with the manufacturer agent when requested by the inventory agent.	X	
FR-10	Inventory agent must inform the coordinator agent when the seller's inventory is lower than the defined limit.	X	
FR-11	Customer agent must discover all seller agents that are selling the required product.	X	
FR-12	Customer agent must negotiate with all known seller agents.	X	
FR-12.1	Customer agent must send a proposal to the seller agents.	X	
FR-12.2	Seller agent must send a counter proposal to the customer agent if the customer's proposal is lower than the seller's price limit.	X	

(continued)

Table 1. Functional Requirements of the Application (continued)

REQ-ID	Description	C	T
FR-12.3	Customer agent must send a counter proposal to the seller agent if the seller agent's counter proposal is greater than the customer's price limit.	X	
FR-12.4	Seller agent must accept the customer agent's proposal if the proposal is within the seller's price range.	X	
FR-12.5	Seller agent must reject the customer agent's proposal if it is lower than the seller's minimum price.	X	
FR-13	Customer agent must send an email to the customer after completing the negotiation, by sending a request to an email agent.	X	
FR-14	Persistence agent must update the database after each transaction.	X	
FR-15	Email agent must send an email to email address defined by the negotiating agents.	X	

3.2.2. Non-Functional Requirements

Non-functional requirements capture the non-functional characteristics of the function (Table 2).

Table 2. Non-Functional Requirements of the Application

REQ-ID	Description	C	T
NFR-1	All kinds of negotiation must be performed within the specified time.	X	
NFR-2	The system must be available using a simple HTTP request.	X	
NFR-3	The system must not crash, except when the server is not running.	X	
NFR-4	The system shall include the specification document, application code, and test suits.	X	
NFR-5	The system shall run on any Java application server.	X	
NFR-6	The system shall not require its users to have any specific training.	X	

3.2.3. Interface Requirements

Interface requirements capture the user-interface behavior (Table 3).

Table 3. Interface Requirements of the Application

REQ-ID	Description	C	T
IR-1	A web-based form shall be provided to the customer where the customer can select the desired product and can provide the minimum price, maximum price, quantity, negotiation time, and email address.	X	
IR-2	The customer shall be able to submit the form using a submit button.	X	
IR-3	In case of an incomplete form submission, a message box stating "Please complete the form" shall be displayed.	X	
IR-4	A web-based form shall be provided to the vendors (sellers, manufacturers, and suppliers) where the vendors shall be able to select existing agents and edit them.	X	
IR-5	A web-based form shall be provided to the vendors (sellers, manufacturers, and suppliers) where the vendors shall be able to create new agent types with their details.	X	

3.3. System Design

This section provides the detailed architectural and component-level design to implement the system. This includes a high-level use-case diagram, an activity diagram, a class diagram, a sequence diagram, and low-level class details.

3.3.1. High-Level Use-Case Diagrams

This section provides the high-level use-case diagrams, illustrating agents' interaction with the system.

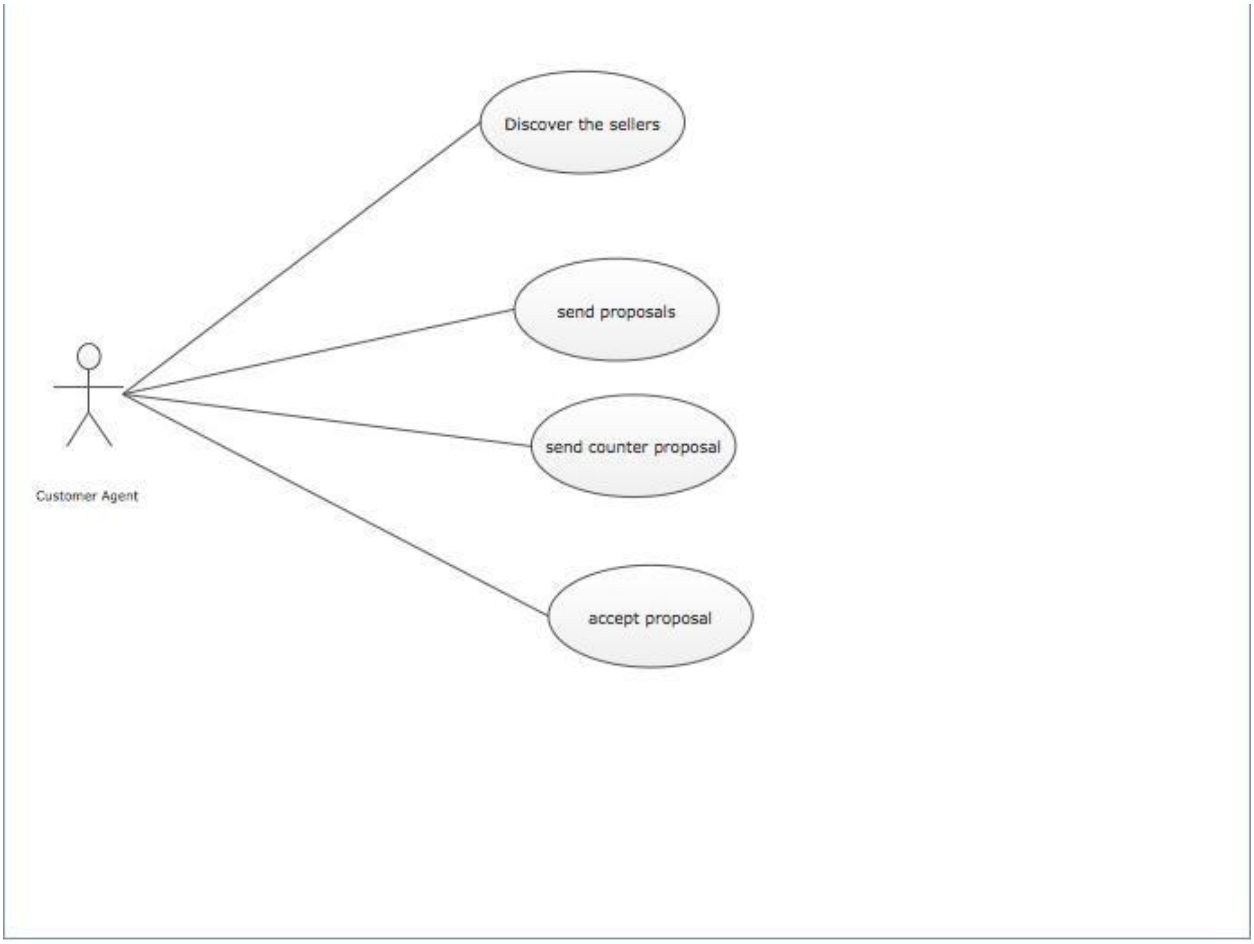

Figure 4. Use-Case Diagram Illustrating the Customer Agent's Actions.

Figure 4 illustrates the customer agent's actions during its lifetime. The customer agent discovers seller agents that are selling the desired product and then sends the proposal and counter proposals until agreeable terms are reached.

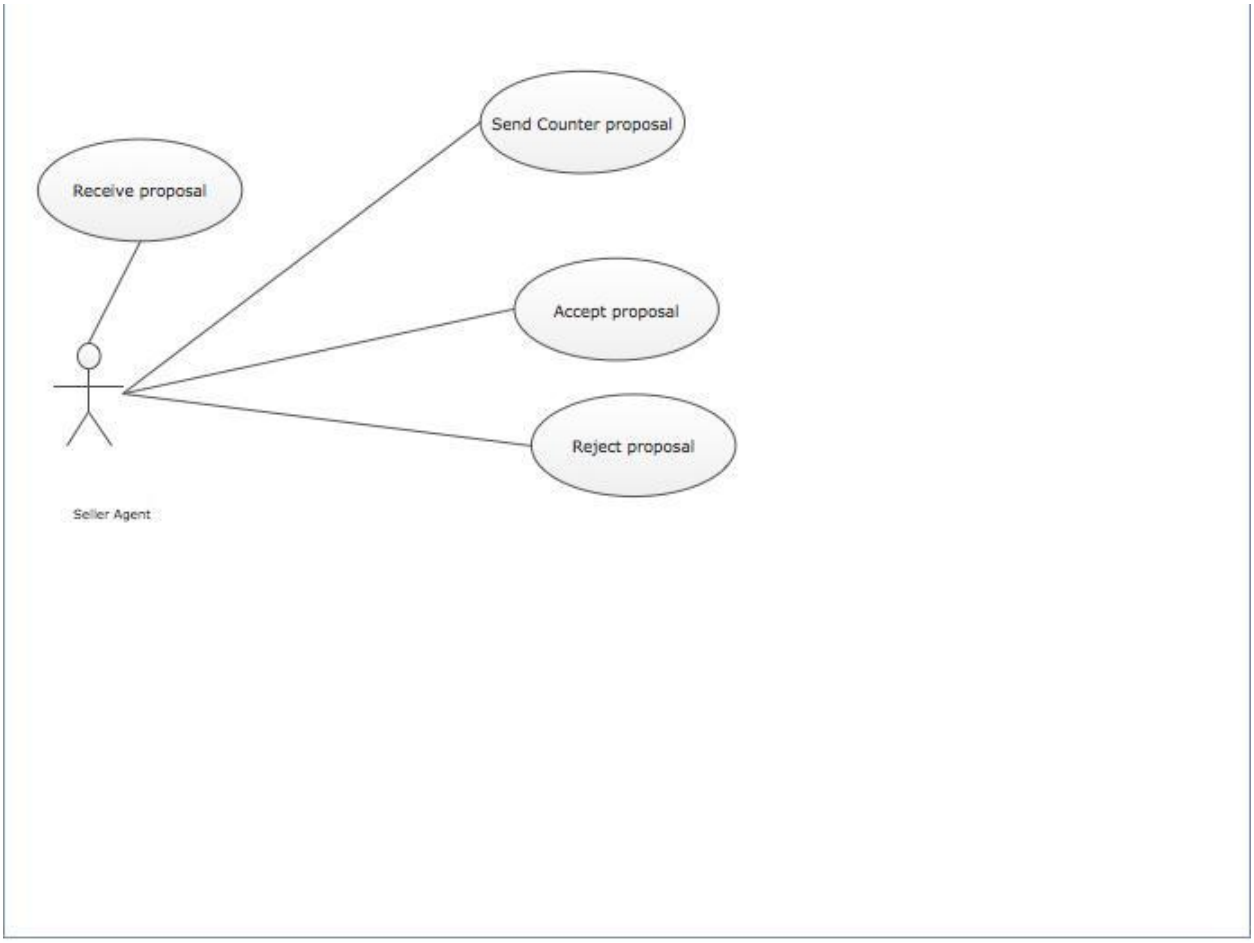

Figure 5. Use-Case Diagram Illustrating the Seller Agent's Actions.

Figure 5 illustrates the seller agent's actions. The seller agent receives proposals from the customer agents and then sends counter proposals, either accepting or rejecting the proposal.

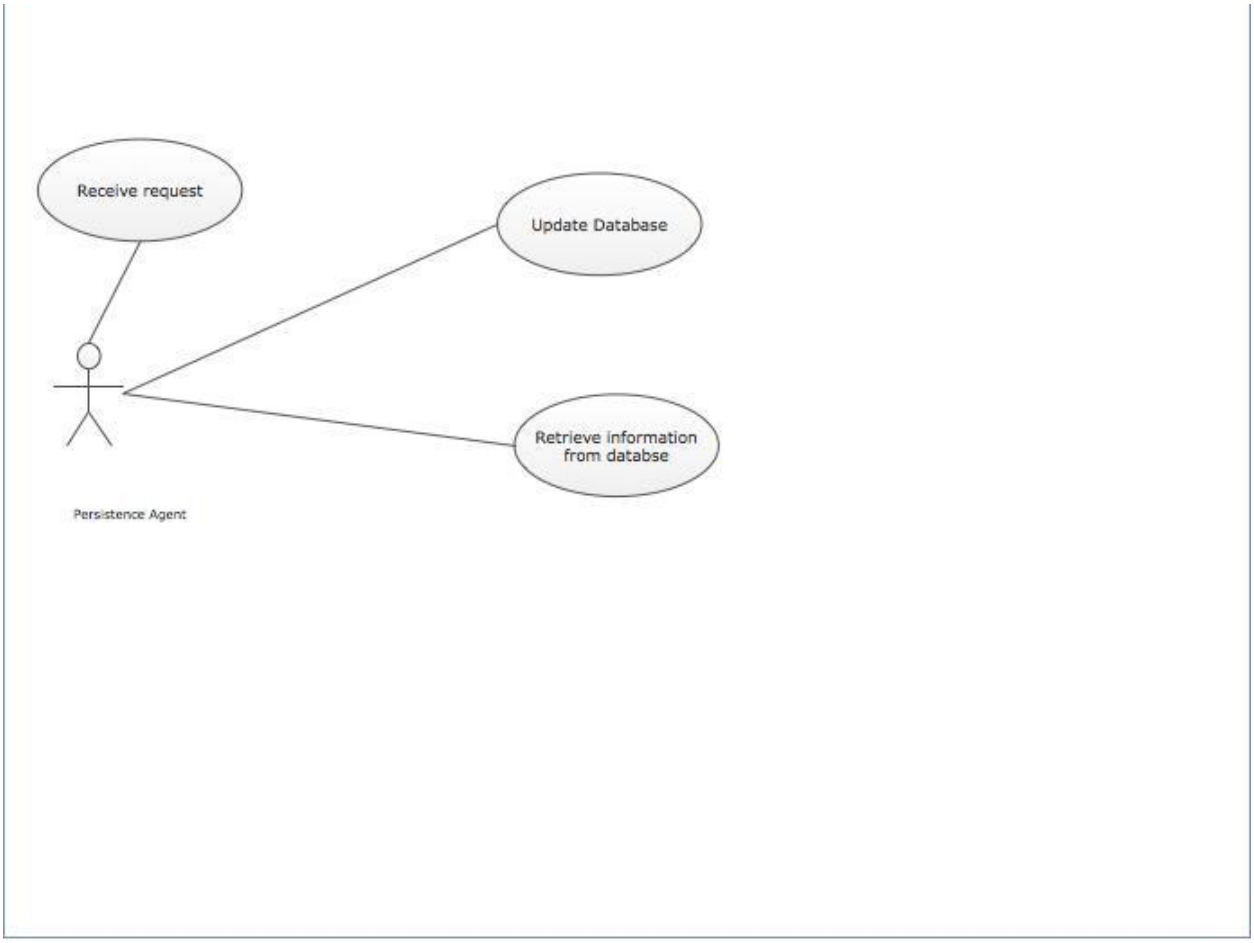

Figure 6. Use-Case Diagram Illustrating the Persistence Agent's Action.

As shown in Figure 6, the persistence agent's sole purpose is to interact with the database whenever it receives a request from other agents.

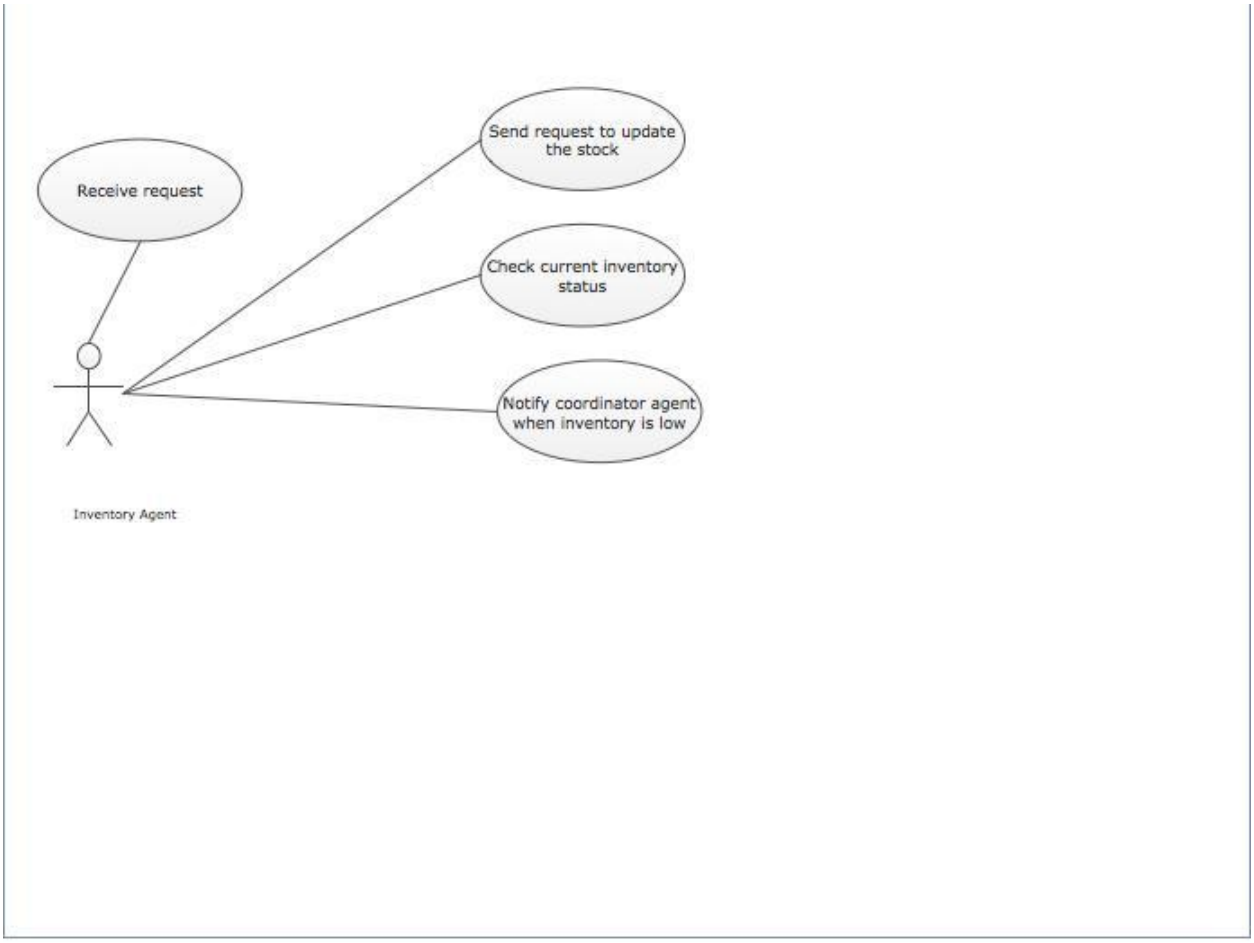

Figure 7. Use-Case Diagram Illustrating the Inventory Agent's Actions.

As shown in Figure 7, the inventory agent receives a notification from the seller agent every time the seller agent sells a product. The inventory agent then checks the current inventory status and notifies the coordinator agent if the inventory level is lower than the predefined criteria.

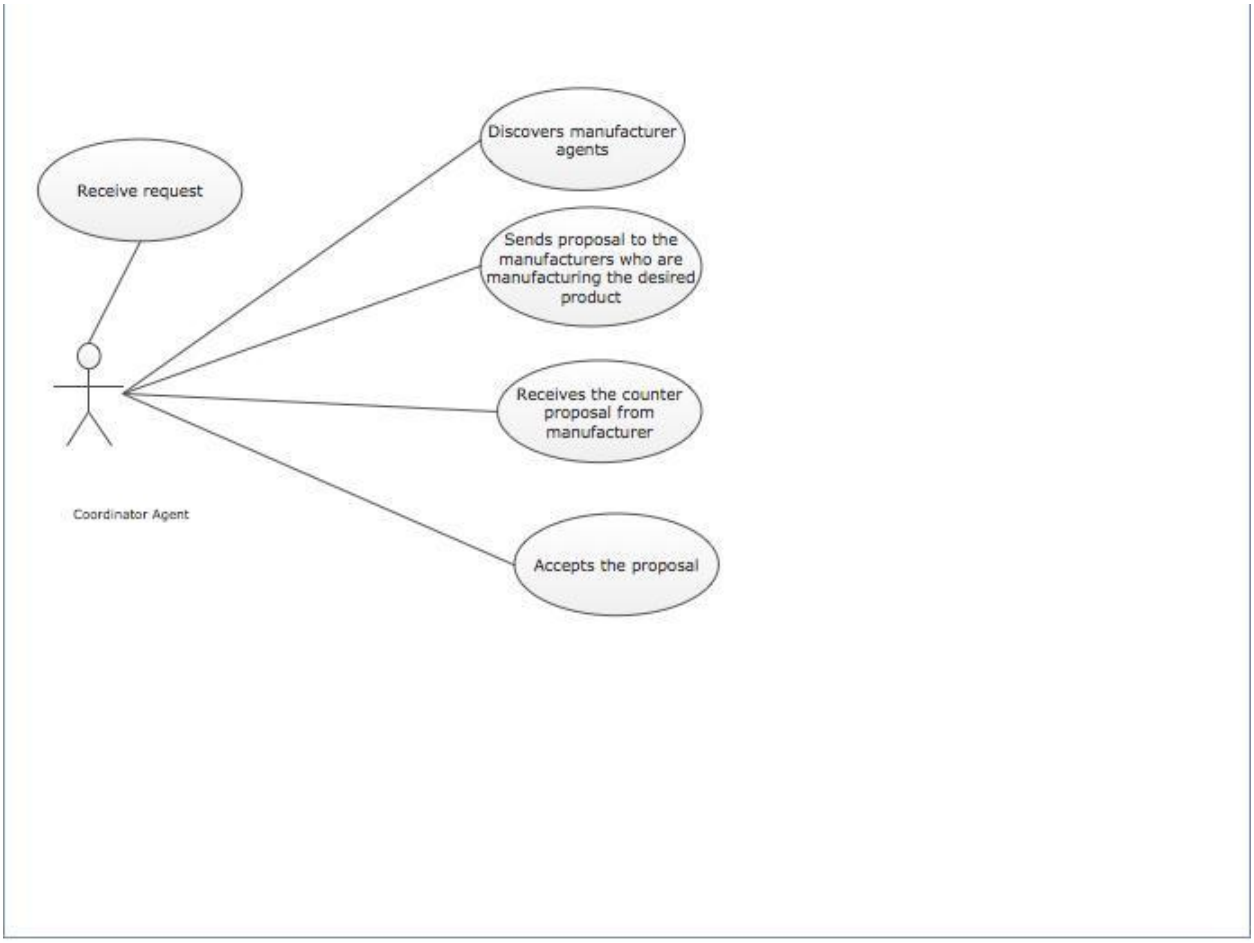

Figure 8. Use-Case Diagram Illustrating the Coordinator Agent's Actions.

Figure 8 shows that the coordinator agent receives the request from the inventory agent and then discovers all the manufacturer agents that are making the desired product. Once it has the list of all manufacturer agents, it sends the proposal; receives the counter proposals, if any, from manufacturer agents; and accepts the best proposal.

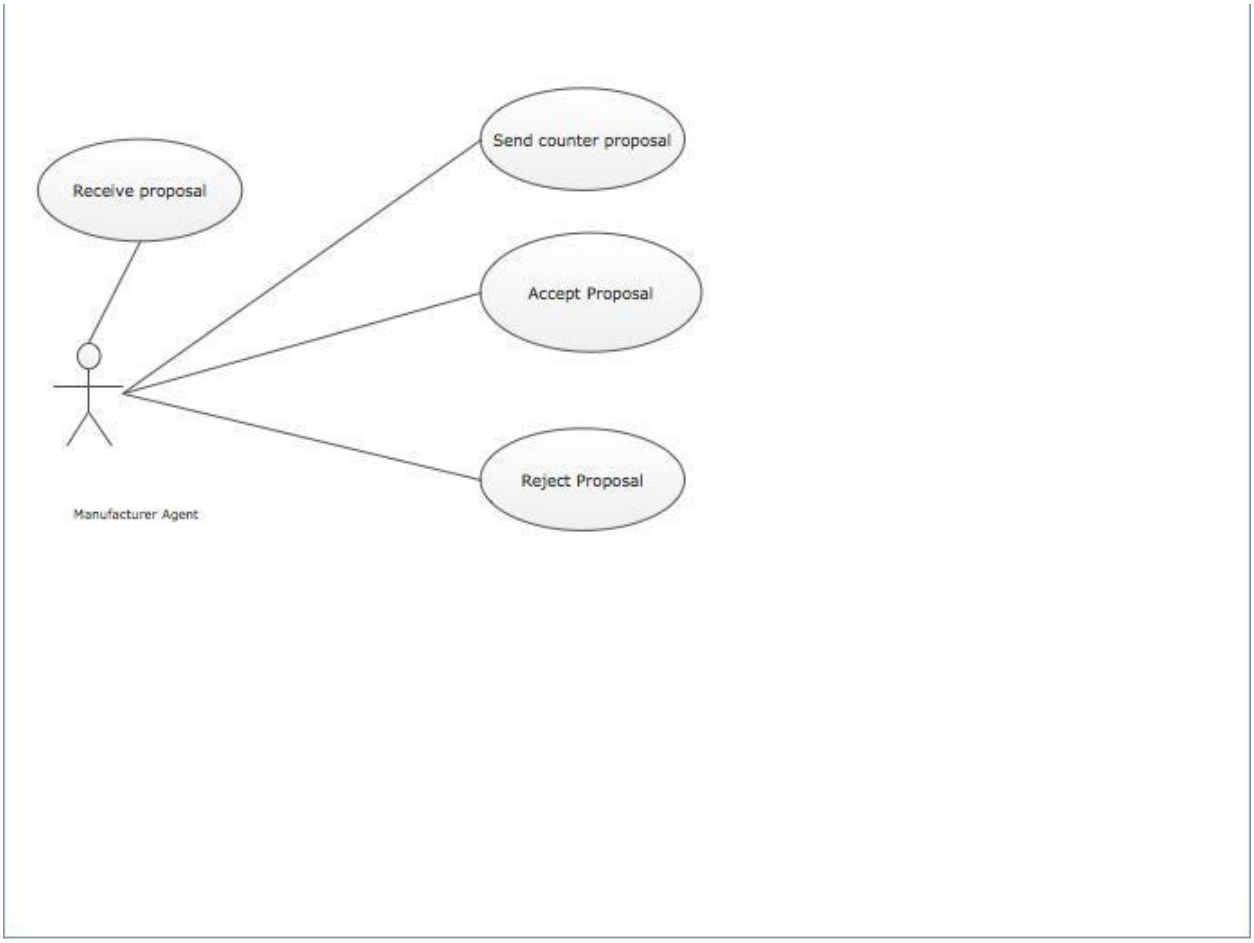

Figure 9. Use-Case Diagram Illustrating the Manufacturer Agent's Action.

Figure 9 illustrates the manufacturer agent's actions. The manufacturer agent receives the proposal from the coordinator agent and, if required, sends a counter proposal to the coordinator. If they agree to the terms, the manufacturer agent accepts the proposal; otherwise, the manufacturer agent rejects the proposal.

3.3.2. Activity Diagram

The activity diagram in Figure 10 provides the stepwise activities, actions, and decisions made by the agents during the negotiation process.

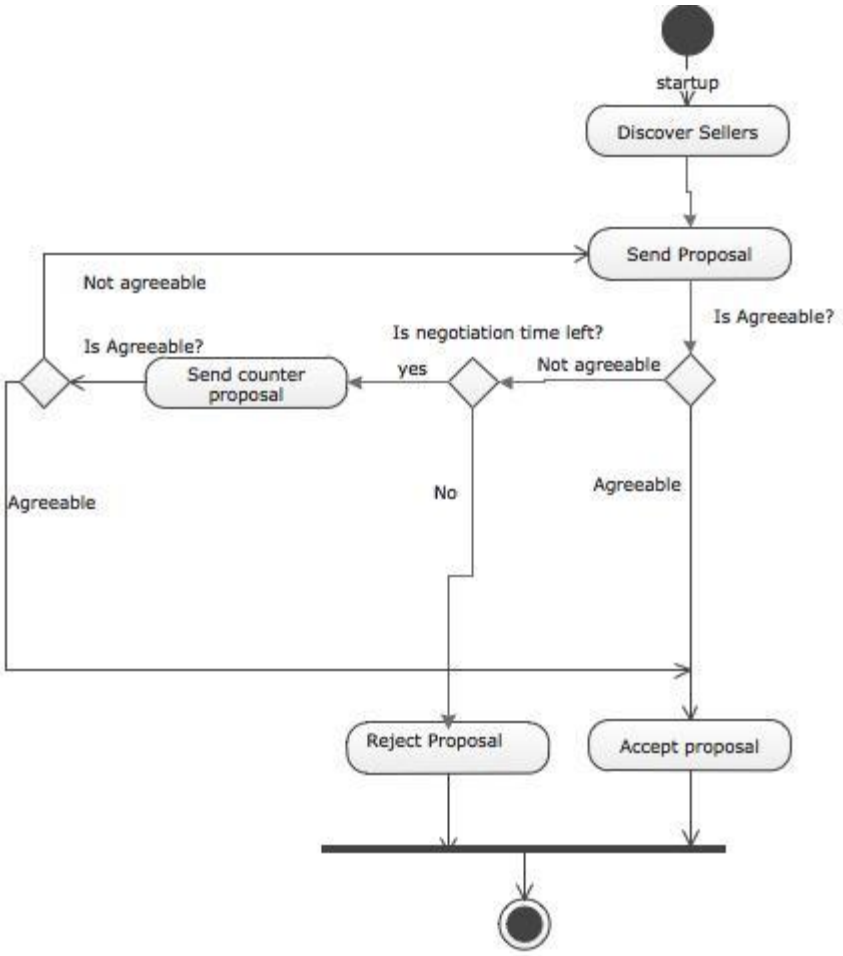

Figure 10. Activity Diagram Showing the Negotiation Flow.

Figure 10 illustrates that, before the negotiation process starts, the customer agent discovers all available sellers that are selling the customer's desired product. After receiving the list of available seller agents, the customer agent sends a proposal to all sellers on the list. If the seller agent accepts the customer agent's proposal, the seller agent sends the accept proposal message to the customer agent, and negotiation completes; otherwise, the seller agent sends a counter proposal to the customer agent. This process lasts until they reach an agreement or the customer agent's negotiation time ends.

3.3.3. Class Diagram

In this section, class diagrams that describe the system's structure are provided by showing the system's classes. Relationships among important classes are shown along with their attributes and their methods.

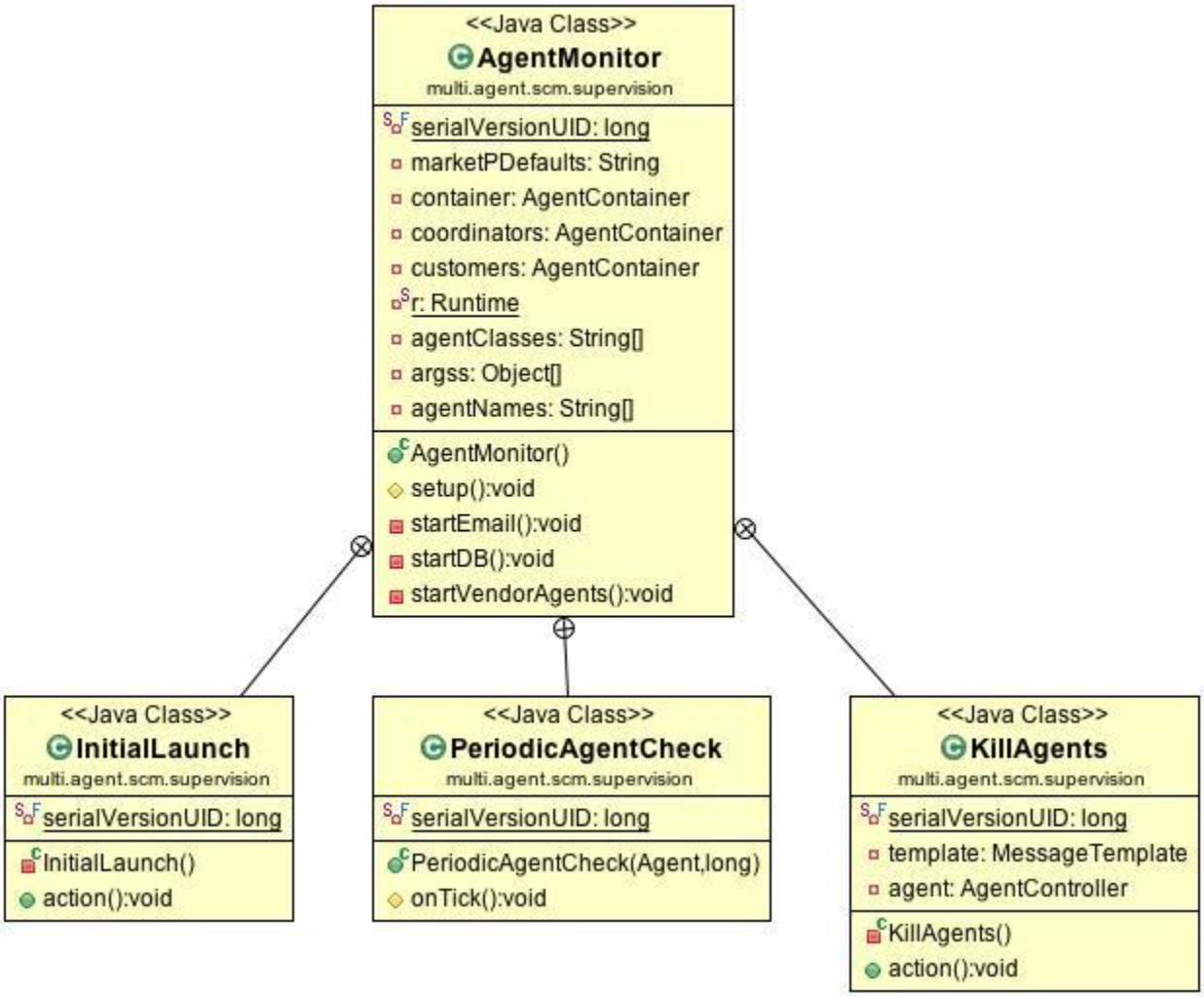

Figure 11. Class Diagram Showing the AgentMonitor Class and Its Inner Classes.

Figure 11 shows the AgentMonitor class and its inner classes. The AgentMonitor class is loaded after the JADE framework starts. This class is responsible for deploying the seller agent,

persistence agent, email agent, manufacturer agent, and supplier agent. The class acts as a bootstrap for the system.

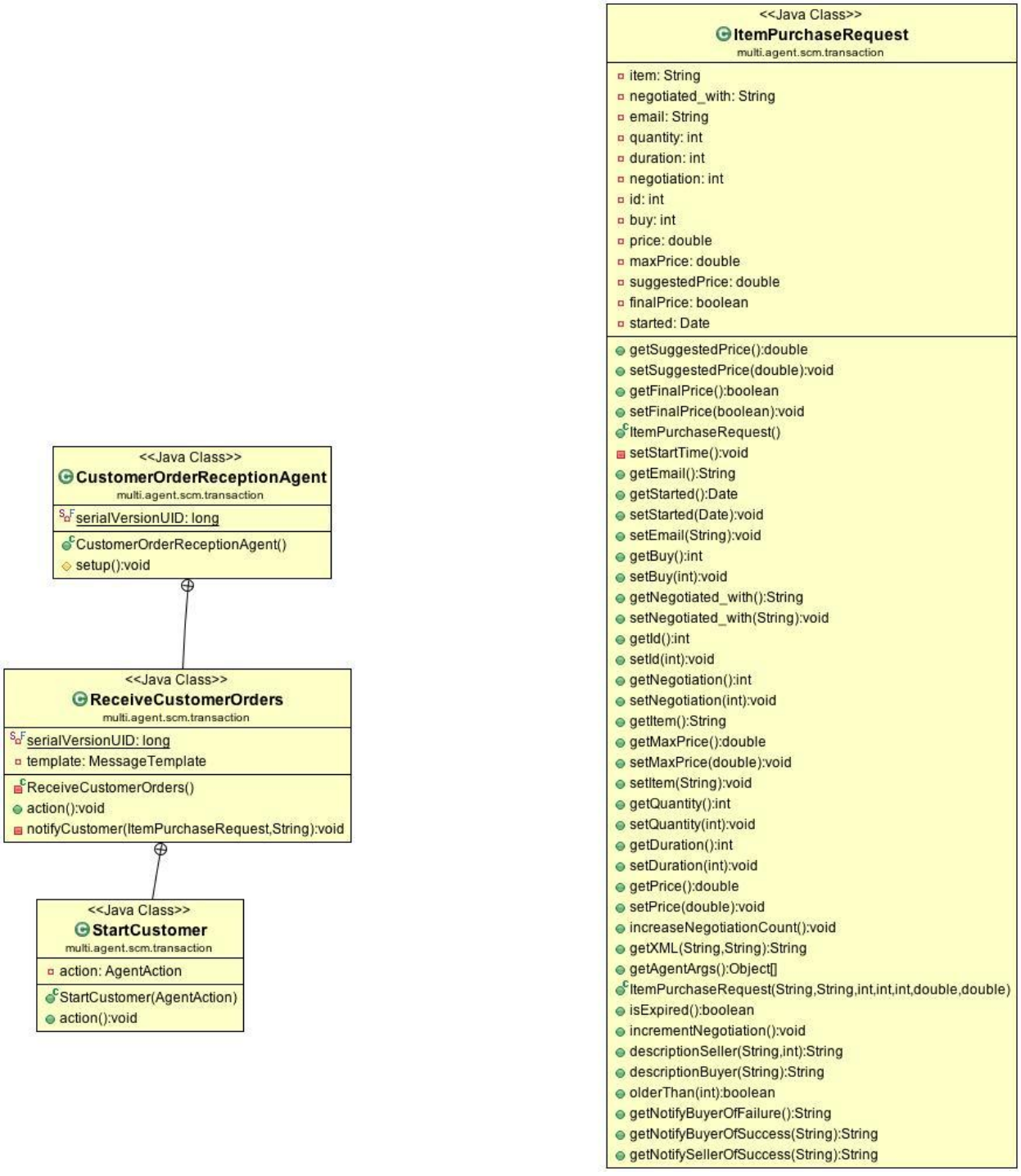

Figure 12. Class Diagram Showing the Classes Responsible for Transactions.

Figure 12 displays the ItemPurchaseRequest class and the CustomerOrderReception class. The ItemPurchaseRequest class acts as a container for requests submitted by the customer and seller agents. All classes that take part in the negotiation process have an instance of this class. The CustomerOrderReception class is responsible for launching a new customer agent and receiving orders from the customer. After receiving the orders from customers, it passes that to the CustomerAgent class, and then, negotiation starts.

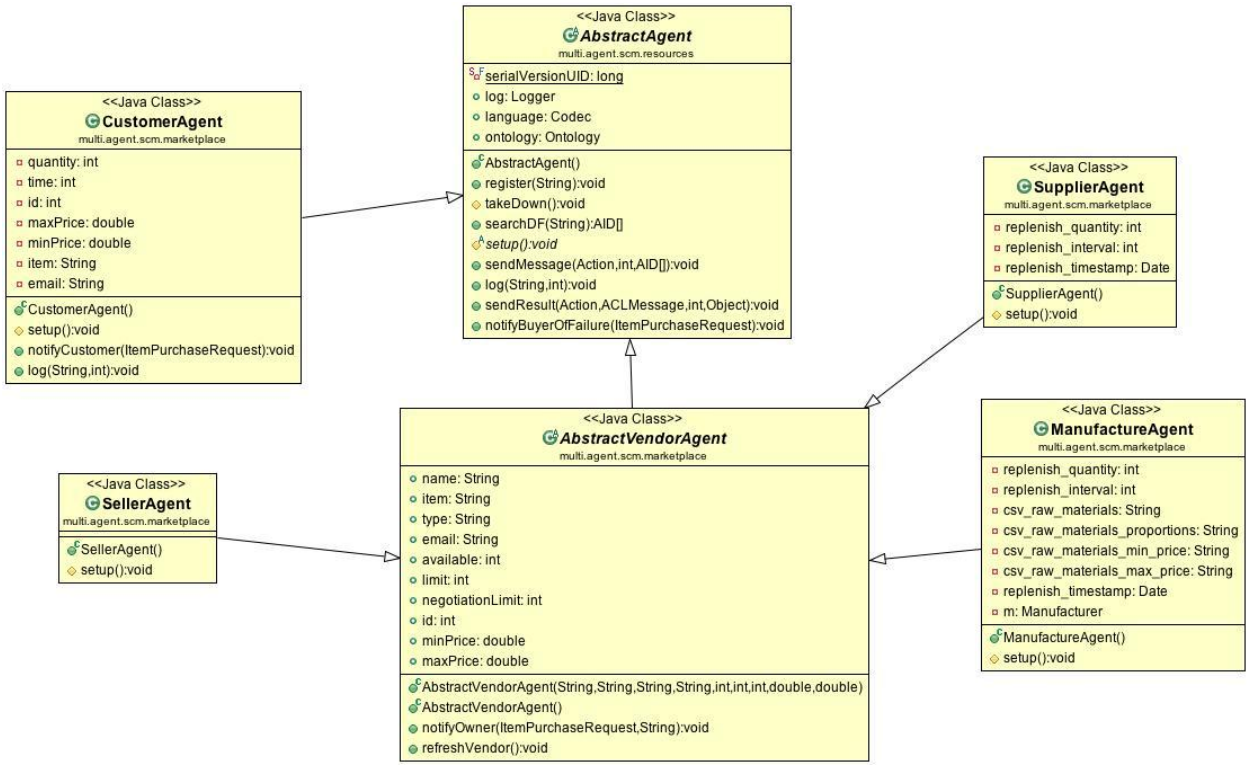

Figure 13. Class Diagram Showing Agents in the Marketplace.

Figure 13 shows very important classes from the architecture's point of view. It consists of the AbstractAgent class, AbstractVendorAgent class, CustomerAgent class, SellerAgent class, ManufacturerAgent class, and SupplierAgent class. The AbstractAgent class is an abstract class and contains all the basic methods that are common among the previously mentioned classes. The AbstractVendorAgent class is also an abstract class which extends the AbstractAgent class

and encapsulates all the methods that are common among the SellerAgent class, ManufacturerAgent class, and SupplierAgent class.

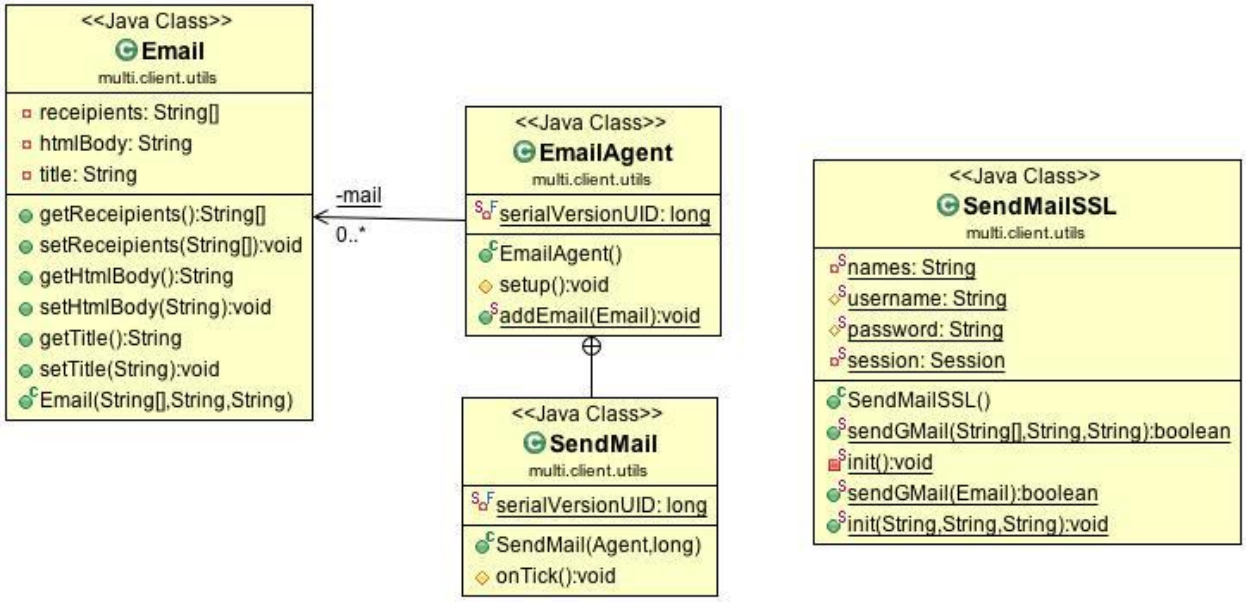

Figure 14. Class Diagram Showing the Agents Responsible for Sending Emails.

Figure 14 shows the Email class, SendMailSSL class, EmailAgent class, and its inner class. The Email class contains the list of recipients, the email body, and the subject. The EmailAgent class has an inner class, SendMail class, as its behavior. The SendMailSSL class has a static method, sendGMail(), which is responsible for sending the mail.

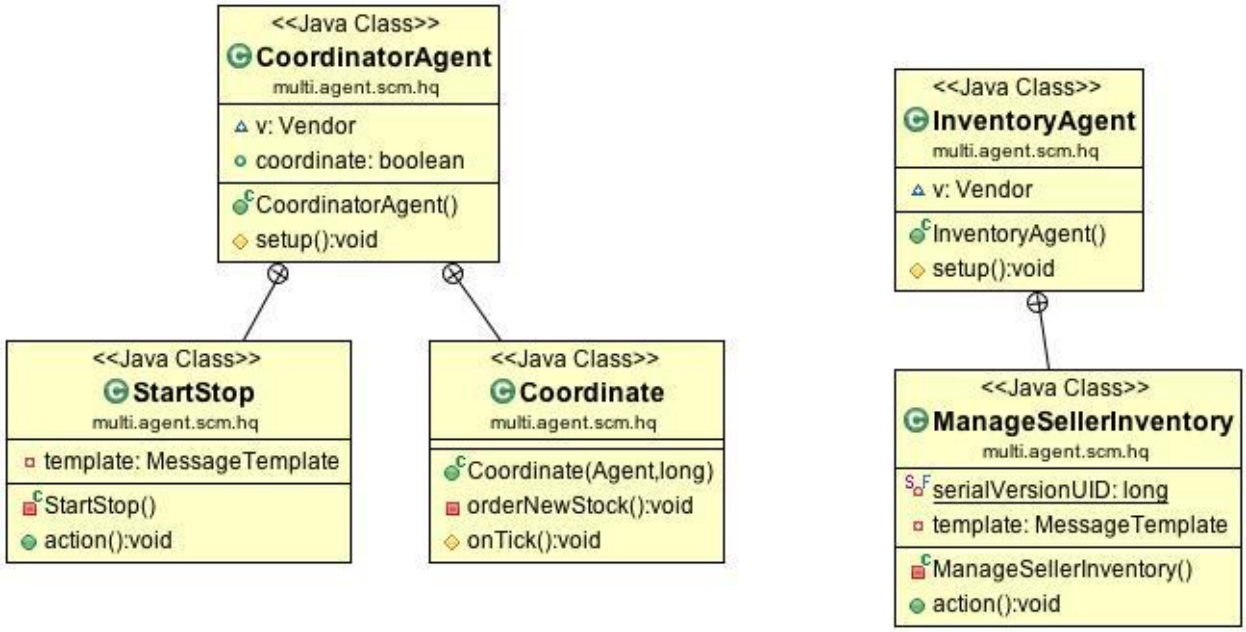

Figure 15. Class Diagram Showing the Coordinator and Inventory Agents.

Figure 15 shows the CoordinatorAgent class and its inner classes. Figure 15 also shows the InventoryAgent class and its inner class. The InventoryAgent class is responsible for keeping track of the inventory level, a predefined quantity.

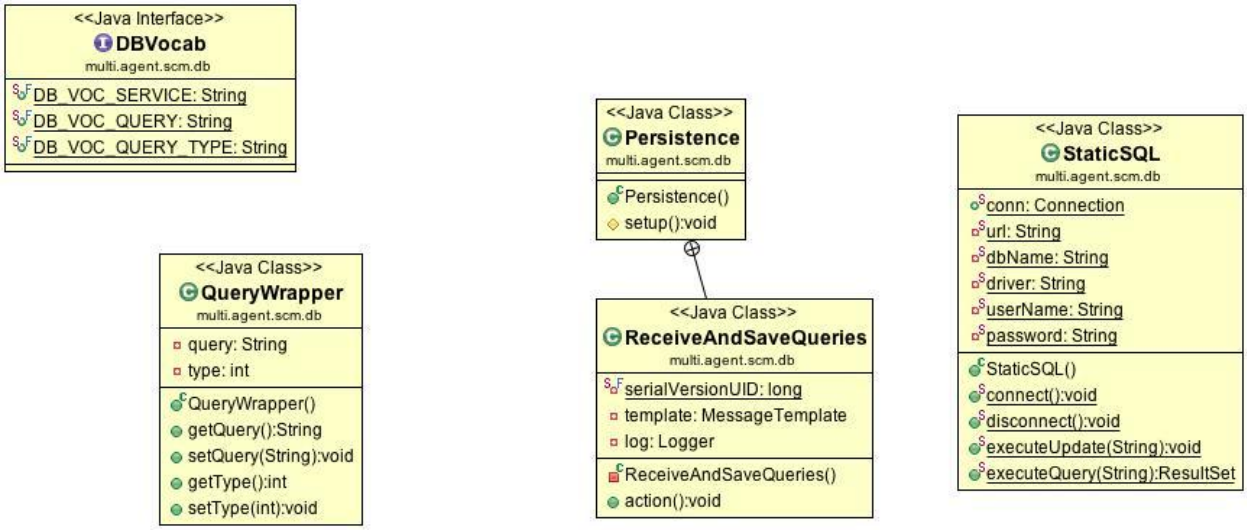

Figure 16. Class Diagram Showing the Persistence Agent and the Classes that Are Responsible for Accessing the Database.

As shown in Figure 16, we have the Persistence class and its inner class, StaticSql class and Query Wrapper class. The Persistence class is an agent class and implements its behavior as the ReceiveAndSaveQueries class which is responsible for creating queries using the QueryWrapper class. The StaticSQL class is responsible for connecting and disconnecting the application; also, it provides the executeUpdate() and executeQuery() methods to execute the queries.

3.3.4. Sequence Diagram

This diagram (Figure 17) illustrates the interaction among the customer, seller, inventory, coordinator, manufacturer, and supplier agents. Various agents send proposal and counter-proposal messages to each other in order to negotiate.

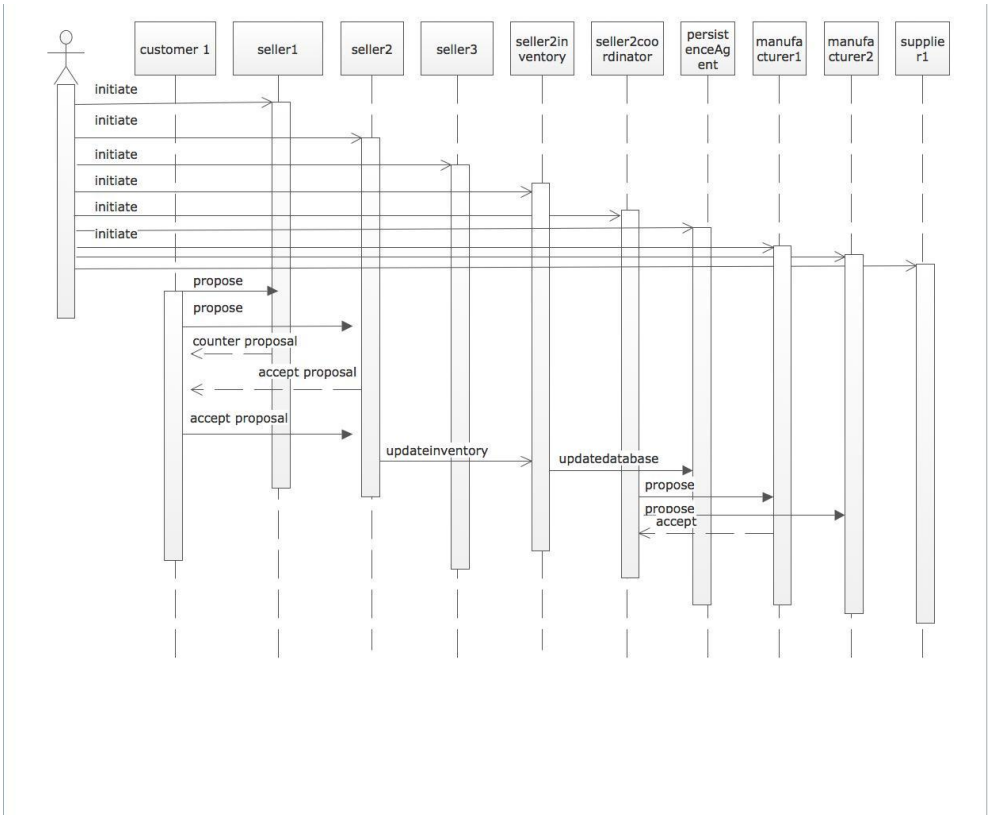

Figure 17. Sequence Diagram of the System.

3.3.5. Low-Level Design Details

This section provides the low-level design details which contain information about the important classes of the proposed system. For ease of illustration, this section is divided into packages that contain the related classes.

3.3.5.1. package multi.agent.scm

3.3.5.1.1. AgentNames Interface: This interface contains predefined names for the agents which are used for communication.

3.3.5.1.2. SCMOntology Class: This singleton class provides the ontology definition for the entire system.

Table 4. Methods of SCMOntology Class

Method Name	Description	Return Type	Parameter	Data Type
getInstance()	Provides an instance of this class.	jade.content.onto.Ontology	None	None
SCMOntology()	Creates agent-action schema and adds all the services provided by the agents.	None	None	None

3.3.5.1.3. Services Interface: This interface provides the standardized values for services registered by agents of a certain type in DF.

3.3.5.1.4. SystemVocabulary Interface: This interface contains the system's ontology name.

3.3.5.2. package multi.agent.scm.db

3.3.5.2.1. Persistence Class: This class implements persistence-agent behavior. It extends the AbstractAgent class which contains the common agent behaviors.

Table 5. Methods of Persistence Class

Method	Description	Return Type	Paramete	Data Type
setup()	Registers the service provided by this agent into DF and adds the behavior.	None	None	None

3.3.5.2.2. RecieveAndSaveQueries Class: This class is the inner class for the Persistence class. This class extends the CyclicBehaviour class defined by the JADE framework.

Table 6. Methods of RecieveAndSaveQueries Class

Method Name	Description	Return	Paramete	Data Type
action ()	Defines the action of updating the database.	None	None	None

3.3.5.2.3. QueryWrapper Class: This class implements the AgentAction interface defined by the JADE interface. AgentAction is a generic interface that is implemented by classes associated wiht agent actions in an ontology.

3.3.5.2.4. StaticSql Class: This class contains the static methods and objects for accessing the database.

Table 7. Methods of StaticSql Class

Method Name	Description	Return	Parameter	Data Type
connect()	Used to connect to the database.	None	None	None
disConnect()	Used to release the connection to the database.	None	None	None
executeUpdate()	Used for updating the database.	None	updateQuery	String
executeQuery()	Used for executing the standard sql for retrieving the information.	ResultSet	Query	String

3.3.5.3. package multi.agent.scm.hq: This package represents the headquarters where the coordinator and inventory agents reside.

3.3.5.3.1. CoordinatorAgent Class: This class is responsible for ordering new products for the sellers. This class contacts manufacturer agents and performs negotiation.

Table 8. Methods of CoordinatorAgent Class

Method	Description	Return Type	Paramet	Data Type
setup()	This method registers the service provided by this agent to the DF agent and adds the behaviors defined by its inner classes.	None	None	None

3.3.5.3.2. Coordinate Class: It is an inner class of the CoordinatorAgent class and extends the CyclicBehavious class from the JADE framework. This class receives a message from the inventory agent; it then initiates a purchase order by searching the platform for all manufacturers that produce the required product. It then sends a purchase request to all of them, and then, the negotiation takes place.

Table 9. Methods of Coordinate Class

Method Name	Description	Return Type	Parame	Data Type
orderNewStock()	This method discovers all existing manufacturer agents that are selling the desired product and negotiates with them.	None	None	None

3.3.5.3.3. InventoryAgent Class: This class is responsible for monitoring the seller's inventory. It makes sure that the seller's inventory is always maintained at a predefined quantity. If inventory goes down, it informs the coordinator agent to buy the product from a seller.

Table 10. Methods of InventoryAgent Class

Method	Description	Return	Paramete	Data Type
setup()	Registers the service and adds the behavior.	None	None	None

3.3.5.3.4. ManageSellerInventory: This inner class of the InventoryAgent class extends the CyclicBehaviour class defined in the JADE framework.

Table 11. Methods of ManageSellerInventory Class

Method	Description	Return Type	Paramet	Data Type
action()	Monitors the inventory level.	None	None	None

3.3.5.4. package multi.agent.scm.marketplace: This package contains all the entities that exist in the market and take part in buying, selling, manufacturing, and supplying the raw material and final products.

3.3.5.4.1. AbstractVendorAgent Class: This class provides a level of abstraction for the vendors. The seller, manufacturer, and supplier share some common functionality which makes them a vendor. This class contains the common features shared by these vendors.

Table 12. Methods of AbstractVendorAgent Class

Method Name	Description	Return Type	Paramet	Data Type
refreshVendor()	Fetches the current list of vendors.	None	None	None

3.3.5.4.2. CustomerAgent Class: This class negotiates in the market on behalf of its owner or a human customer.

Table 13. Methods of CustomerAgent Class

Method	Description	Return	Paramet	Data Type
setup()	Registers the agent into the DF and adds behavior to the agent.	None	None	None

3.3.5.4.3. CustomerStartNegotiation Class: It is an inner class of the CustomerAgent class. This class represents the behavior which is added to the CustomerAgent class. This class extends the OneShotBehaviour class which is defined in the JADE framework.

Table 14. Methods of CustomerStartNegotiation Class

Method Name	Description	Return	Parameter	Data Type
action()	This method is responsible for negotiation.	None	None	None
notifyBuyerOfFailure()	This method is responsible for notifying the user if the negotiation fails.	None	Item	ItemPurchaseRequest

3.3.5.4.4. ManufactureAgent Class: This class represents the real-world manufacturer that makes the products. This class is responsible for creating new products, and it negotiates with the supplier when it is low on raw materials.

Table 15. Methods of ManufactureAgent Class

Method Name	Description	Return Type	Paramet	Data Type
setup()	This method registers the agent to the DF and initializes the manufacturer's properties.	None	None	None

3.3.5.4.5. Manufacture Class: It is the inner class of the ManufactureAgent class. It represents the behavior which is responsible for producing the new products and ordering the raw material.

Table 16. Methods of Manufacture Class

Method Name	Description	Return	Parameter	Data Type
orderNewStock()	This method negotiates with suppliers and buys the raw material.	None	Item,no,max,min	String, int, double, double
onTick()	This method makes new products after every predefined time period.	None	None	None

3.3.5.4.6. SupplierAgent Class: This class represents the supplier that is responsible for supplying the raw material.

Table 17. Methods of SupplierAgent Class

Method Name	Description	Return Type	Parameter	Data Type
setup()	This method registers the agent to the DF and initializes the supplier's properties.	None	None	None

3.3.5.4.7. ReplenishSupply Class: This inner class of the SupplierAgent class implements the behavior responsible for creating a new raw material after a given time period.

3.3.5.4.8. Negotiate Class: This class is the main class that is responsible for all negotiations in this system. All the entities delegate the negotiation task to this class, and based on the message it receives, it performs the negotiation.

Table 18. Methods of Negotiate Class

Method	Description	Return	Paramet	Data
onStart()	This method adds the behavior to this agent class based on the agent class which sends the message to it.	None	None	None
action()	This method is responsible for performing the negotiation.	None	None	None

3.3.5.5. Jade APIs: This section describes the JADE APIs used by the system. Mainly, these behaviors are defined by the JADE framework to enable the agents achieve their goals. This section also briefly describes the JADE Web Service Integration Gateway (WSIG), an add-on that provides support for the invocation of JADE agent services from web-service clients.

3.3.5.5.1. Cyclic Behavior: This class provides the behavior that must be executed forever. It is an abstract class and is extended by the agent classes that want to create behavior that should keep executing continuously. Simple, reactive behavior is an example of such behavior.

3.3.5.5.2. OneShot Behavior: This class provides the behavior that only executes once. This class must be extended by the agent classes that want to create the behavior for operations that only need to be done one time.

3.3.5.5.3. Ticker Behavior: This class provides the behavior that periodically executes the user-defined code. Agent classes that want to extend this class must override the onTick() method and include the code that must be executed periodically.

3.3.5.5.4. Web Service Integration Gateway (WSIG): The objective of WSIG is to expose services that are provided by agents and published in the JADE DF as web services with no minimal or no additional effort while giving developers enough flexibility to meet the specific requirements that they may have. The process involves generating a suitable WSDL for each service-description registered with the DF and, possibly, publication of the exposed services in a UDDI registry.

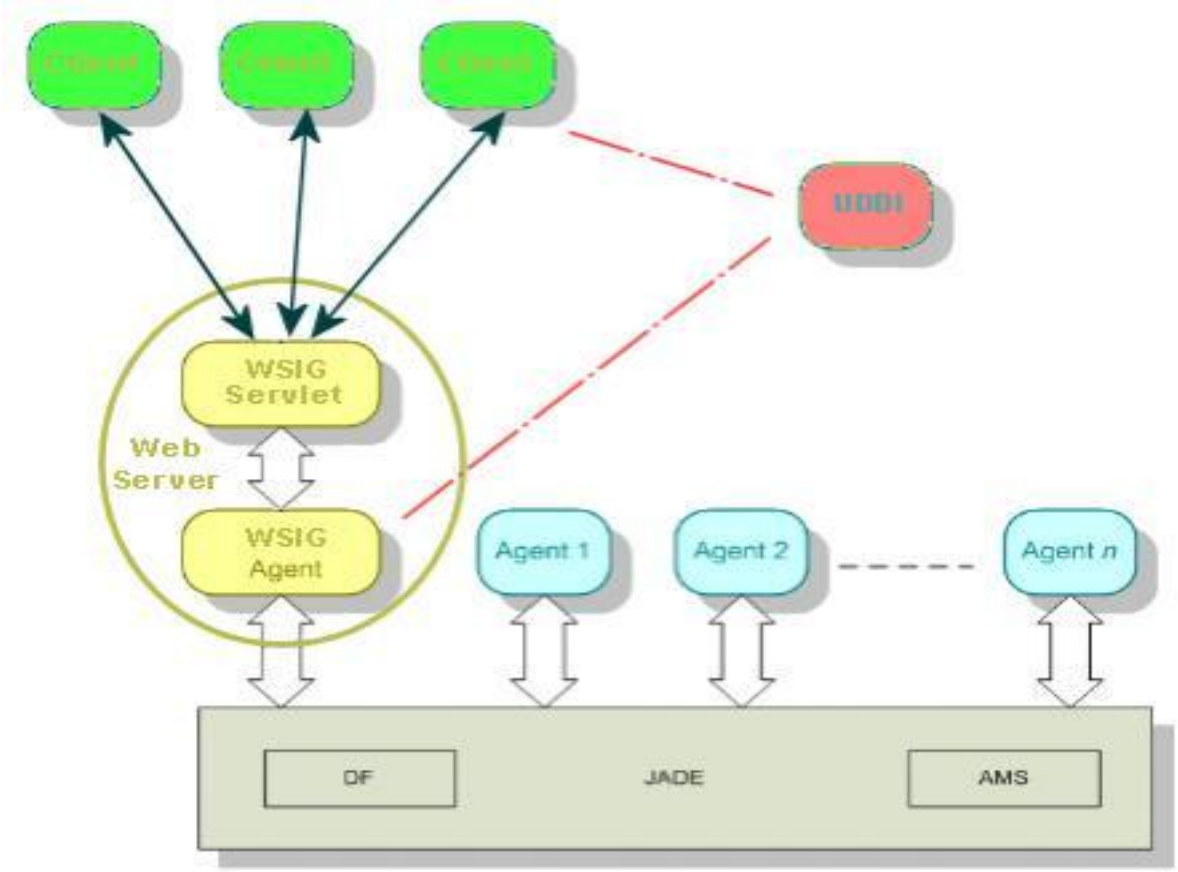

Figure 18. WSIG Architecture.

As shown in Figure 18, the WSIG is composed of two main components:

- WSIG Servlet
- WSIG Agent

The WSIG servlet serves the HTTP/SOAP requests and extracst the SOAP message. It then creates the corresponding agent action and passes it to WSIG agent.

The WSIG agent is the gateway between the web and the agent world. It forwards the agent actions, received from the WSIG servlet, to the agents that are actually able to serve the request and gets a response. The WSIG agent subscribes to the DF service so that it can be

notified when a new agent is registered and an existing agent deregisters. It also creates a Web Service Definition Language (WSDL) corresponding to each agent service registered with the DF and publishes the service in a UDDI registry, if needed.

3.3.5.5.5. Negotiation Algorithm:

-- An agent that wants to purchase an item searches the DF for all agents selling the required item.

-- It then sends purchase requests to all found agents.

-- All agents not in a position to supply the required items reject this transaction.

-- All other agents proceed with the negotiation.

-- If a selling agent does not accept the proposed terms,

-If that was the buyer's last offer, then negotiation ends.

-Otherwise, the selling agent will propose different terms.

-If the buyer does not agree with the seller's terms, a counter proposal is made.

-This process continues until the agents either die or they do agree.

-- If a selling agent accepts the buyer's terms, it sends a proposal acceptance.

-If the buying agent has not accepted another offer, it will respond to this seller with instructions to effect the transaction; otherwise, it will be ignored.

-The selling agent will then effect the transaction and send a confirmation to the buying agent.

-The agent updates its database via the persistence agent and emails its owner about the transaction.

3.4. System Testing

This section provides the cases developed to test the system. Test cases are run by creating various agents and setting their properties in such a way that they negotiate with each other. Test validity is checked by observing the sniffer agent's message window which shows the agent's interaction message by message.

Table 19. Test Cases

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass /Fail	Req. Id
TC1	AgentMonitor launches the EmailAgent at the start of the application.	The database must be up and running	The following command should be entered at the command line: java –cp <classpath> jade.Boot –gui Supervisor:multi.agent.scm.supervision.AgentMonitor	The JADE GUI agent must start, and the EmailAgent must be deployed.	The JADE GUI agent starts, and EmailAgent is deployed.	Pass	FR-1

(continued)

Table 19. Test Cases (continued)

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass/Fail	Req. Id
TC2	AgentMonitor launches the persistence agent at the start of the application.	The database must be up and running.	The following command should be entered at the command line: java –cp <classpath> jade.Boot –gui Supervisor:multi.agent.scm.supervision.AgentMonitor	The JADE GUI agent must start, and the persistence agent must be deployed.	The JADE GUI agent starts, and the persistence agent is deployed.	Pass	FR-2
TC3	AgentMonitor launches the vendor agents at the start of the application.	The database must be up and running.	The following command should be entered at the command line:java –cp <classpath> jade.Boot –gui Supervisor:multi.agent.scm.supervision.AgentMonitor	The JADE GUI agent must start, and it creates a container named MarketPlace which contains all the vendor agents.	The JADE GUI agent starts, and it creates a container named MarketPlace which contains all the vendor agents.	Pass	FR-3

(continued)

Table 19. Test Cases (continued)

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass /Fail	Req. Id
TC4	The supplier agent replenishes the raw material at the predefined time.	The database must be up and running.	Create supplier with the replenish time and replenish quantity	The supplier must replenish its stock with the predefined quantity at the predefined time.	The supplier replenishes its stock with the predefined quantity at the predefined time.	Pass	FR-4,FR-5
TC5	The manufacturer agent makes a product at the given time and with the predefined quantity.	The database must be up and running.	Create manufacturer with the replenish time and replenish quantity	The manufacturer must replenish its stock with the predefined quantity at the predefined time.	The manufacturer replenishes its stock with the predefined quantity at the predefined time.	Pass	FR-6

(continued)

Table 19. Test Cases (continued)

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass /Fail	Req. Id
TC6	The manufacturer agent discovers all supplier agents that have the desired raw material.	The database must be up and running. More than 1 supplier agent is deployed with the desired raw material supplying	Create manufacturer with raw material 0 quantity and production time and production quantity > 0	The manufacturer agent must send the proposal to all suppliers that have the desired raw material. This should be verified by observing the sniffer agent's message window.	The manufacturer agent sends the proposal to all the supplier agents that have the desired product. This is verified by observing the sniffer agent's message window.	Pass	FR-7

(continued)

Table 19. Test Cases (continued)

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass/ Fail	Req. Id
TC7	The manufacturer and supplier agents negotiate with each other.	The database must be up and running. More than 1 supplier agent is deployed with the desired raw material supplying	Create a manufacturer with raw-material quantity and production time of 0 as well as a production quantity > 0. Set the manufacturer's minimum raw material price < the maximum supplier's price and the manufacturer's maximum raw-material price > the supplier's minimum price.	The manufacturer must send a proposal to the supplier agent, and the supplier agent must send a counter proposal to the manufacturer agent. This continues until both reach to a point agreement. This is verified by checking the sniffer agent's status.	The manufacturer and supplier agents exchange proposals and counter proposals as expected. This is verified by observing the sniffer agent's message window.	Pass	FR-8,FR-8.1 to FR-8.4

(continued)

Table19. Test Cases (continued)

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass/ Fail	Req. Id
TC8	The supplier agent rejects the manufacturer agent's proposal if the they do not agree to the negotiation terms.	The database must be up and running. More than 1 supplier agent is deployed with the desired raw material supplying	Create a manufacturer with a raw-material quantity and production time of 0 as well as a production quantity > 0. Set the manufacturer's maximum raw material price < the supplier's minimum price.	The supplier must reject the manufacturer's proposal after sending proposals and counter proposals. This must be verified by observing the sniffer agent's message window.	The supplier agent rejects the manufacturer agents's proposal after sending proposals and counter proposals. This is verified by observing the sniffer agent's message window.	Pass	FR-8.5

(continued)

Table 19. Test Cases (continued)

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass/ Fail	Req. Id
TC9	The supplier agent and manufacturer agent send email to their owners by requesting the email agent.	The database must be up and running. More than 1 supplier agent is deployed with the desired raw material supplying	Create a manufacturer agent with a raw-material quantity and production time of 0 as well as a production quantity > 0. Set the manufacturer agent's minimum raw-material price < the maximum supplier's price and the manufacturer agent's maximum raw-material price > the supplier agent's minimum	The supplier agent and manufacturer agent send a request to the email agent after negotiation is completed. An email must be sent to the email address which is associated with the supplier agent. This must be verified by checking the sniffer agent's status and the email.	The supplier agent and manufacturer agent send a request to the email agent after negotiation is completed, and an email is received by the supplier's owner and the manufacturer agent's owner.	Pass	FR-8.6, FR-8.7

(continued)

Table 19. Test Cases (continued)

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass/ Fail	Req. Id
TC 10	The coordinator agent and manufacturer agent negotiate with each other.	The database must be up and running. More than 1 manufacturer agent is deployed with the desired product manufacturing	Create a seller agent with a stock quantity < the inventory limit.	The coordinator agent associated with the seller agent and the manufacturer agent sends a proposal and counter proposal to each other until they reach agreeable terms. This is verified by observing the sniffer agent's window.	The xpected behavior is achieved and verified by observing the sniffer agent's message window.	Pass	FR-9,FR-10

(continued)

Table 19. Test Cases (continued)

Test ID	Description	Pre-Condition	Input	Expected output	Actual output	Pass /Fail	Req. Id
TC 11	The customer agent negotiates with seller agents and sends an email to their owners after the negotiation is completed, by sending a request to the email agent.	The database must be up and running. More than 1 seller agent is deployed selling the desired product	Create a customer agent with a quantity property > 0.	Customer agents send the proposal to all seller agents that have the desired product. Customer and seller agents exchange counter proposals until they reach to an agreement proposal. This is verified by checking the sniffer and checking the owner's email	Negotiation takes place as expected and is verified by observing the sniffer agent's message window. Emails are received by the agent's owners.	Pass	FR-12 to FR-15

3.5. Screen Shots

This section provides screen shots of the user interface. It also provides screen shots of the JADE framework's Remote Agent Management GUI and the sniffer agent's message box GUI.

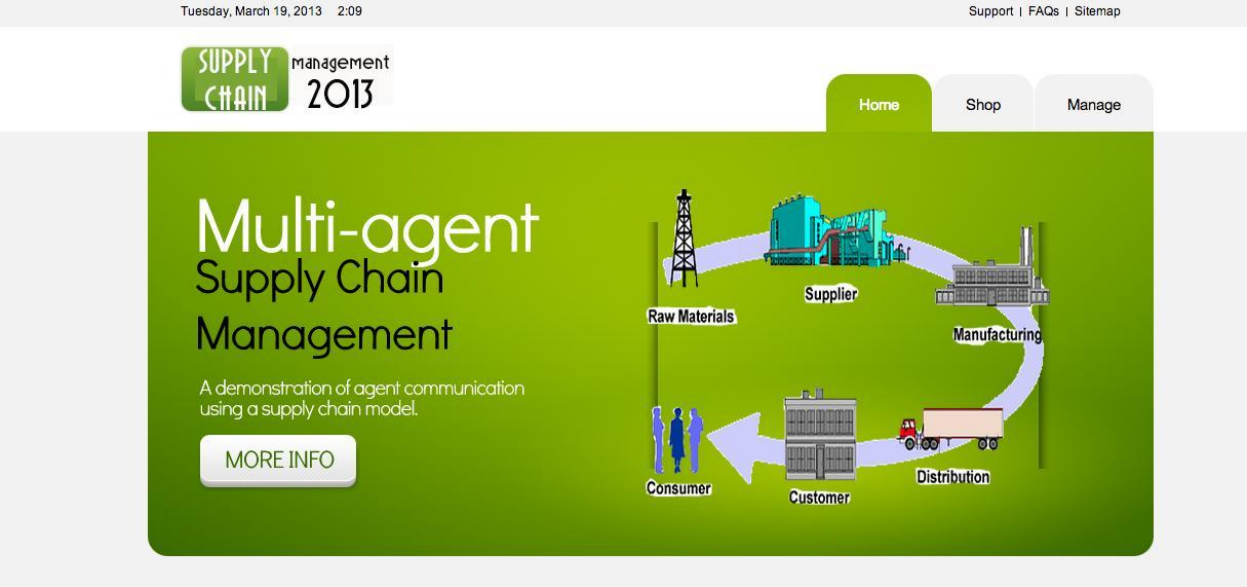

Figure 19. The Website's Home Page.

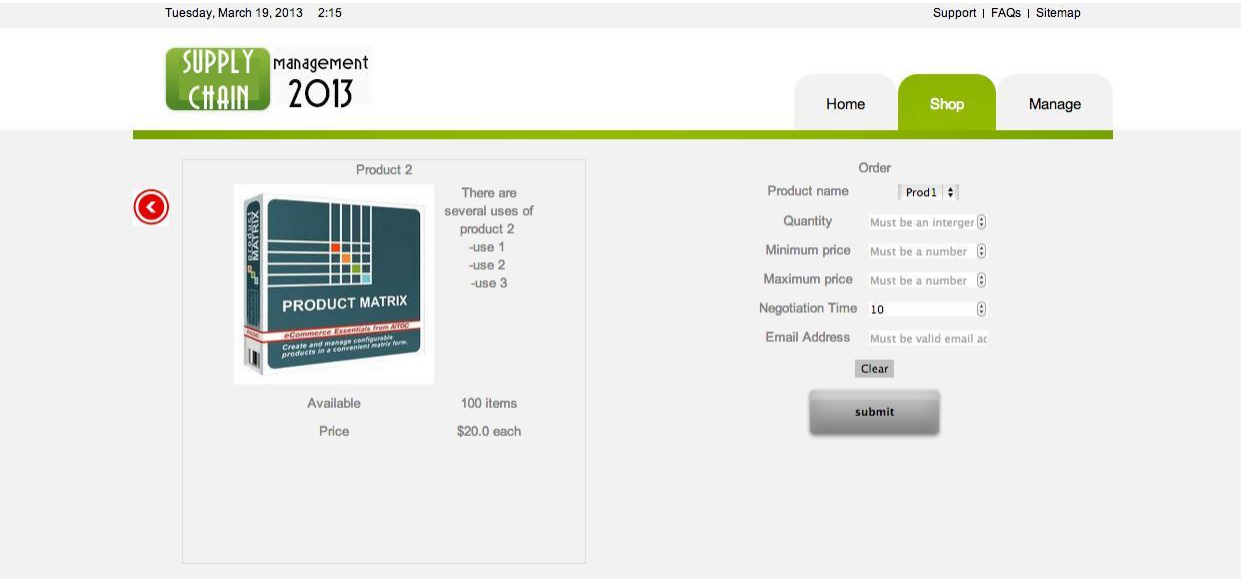

Figure 20. Customer Details' Entry Page.

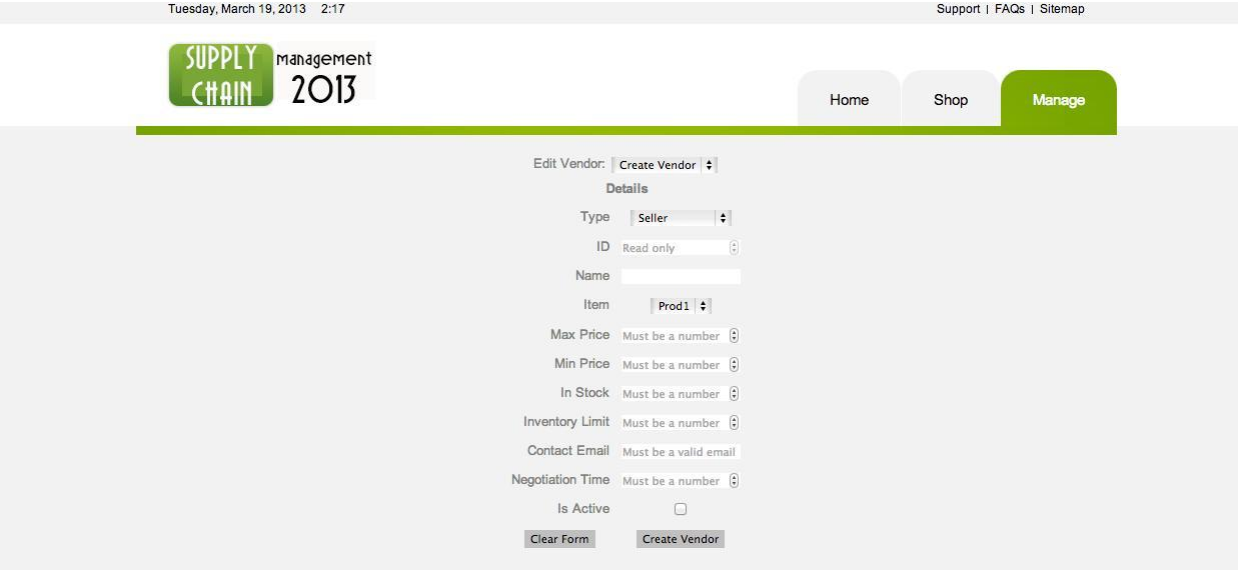

Figure 21. Seller, Manufacturer, and Supplier Creation Page.

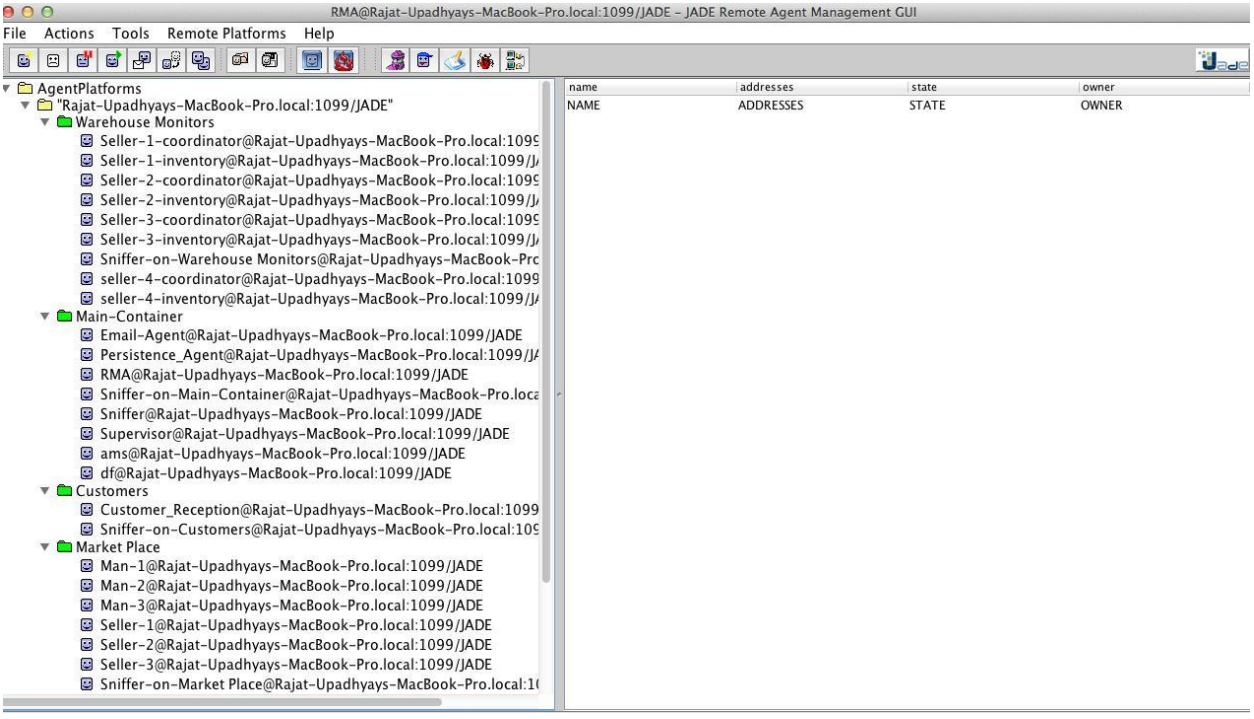

Figure 22. JADE Framework's Remote Agent Management GUI Showing Agents in the Active State.

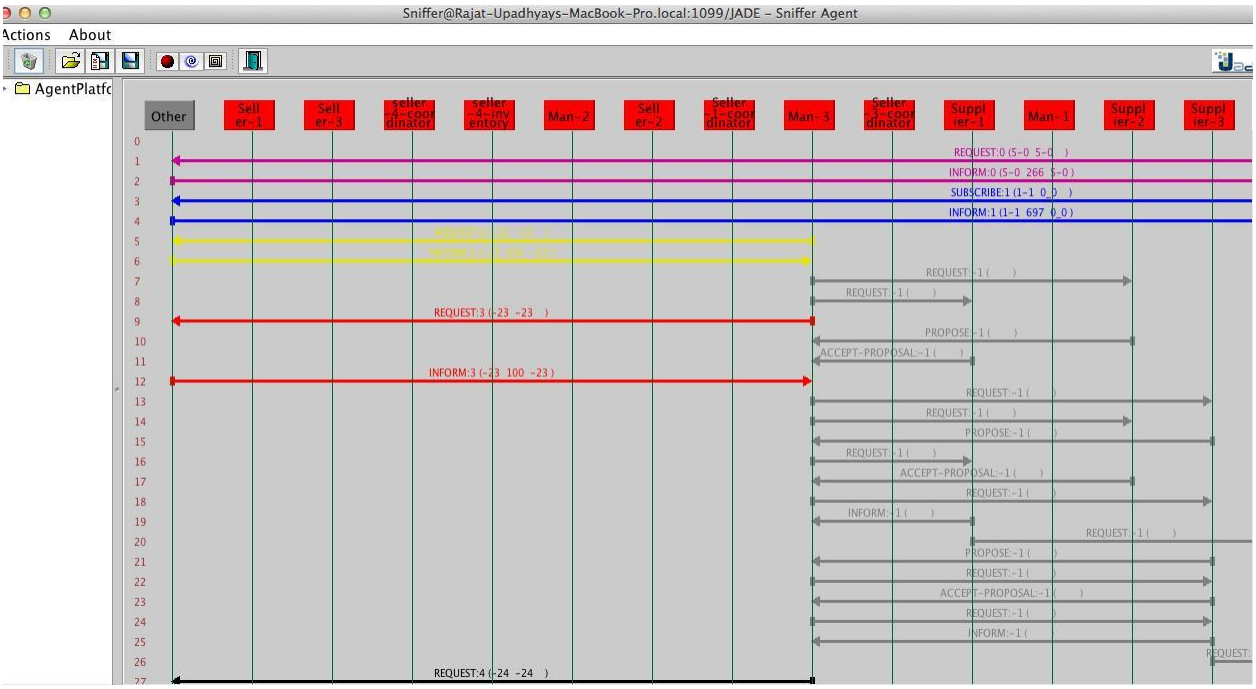

Figure 23. Sniffer Agent's Message Box GUI.

CHAPTER 4. CONCLUSION

A simple supply chain is implemented in the present paper using multi-agent modeling. Entities related to a simple supply chain, such as the seller, inventory, manufacturer, and supplier, are implemented as intelligent agents. When these agents are created with their specific goals, they communicate and negotiate with each other successfully by sending ACL messages (FIPA compliant).

This application effectively demonstrates the capabilities of intelligent agents in the field of supply chain management. It also demonstrates that agents can independently carry out goals by making their own decisions in a collaborative manner.

This paper presents an approach using which supply chain management can be implemented like an e-commerce system. In order to keep things less complex and to demonstrate agent negotiation effectively, entities of the supply chain are kept very simple. In real life, a supply chain is a very complex; this system can be extended very easily to accommodate complex models.

As proposed, this implementation is for a very basic supply chain model. However, it can be converted to a complex supply chain system because agents are capable of making complex decisions and are able to communicate seamlessly.

In this paper, a simple negotiation algorithm that is based on the price and the negotiation time is provided. In the real world, a new attribute credibility for sellers and buyers can be introduced. This attribute must be based on the rating system provided by various entities of the supply chain, and the rating system can be used as one of the variables in the negotiation process.

This paper also discusses the Web Service Integration Gateway, a very important feature of the JADE framework. The Internet has become all-pervasive in the modern world, and multi-

agent systems are no the exception to this. This paper successfully demonstrates the integration of web services with the multi-agent framework which can be utilized as providing the interactive web pages or can be exposed to other business entities.

REFERENCES

1. M. Beer, M d' Inverno, M. Luck, N. Jennings, C. Preist, and M. Schroeder, "Negotiation in Multi-Agent Systems." Workshop of the UK Special Interest Group on Multi-Agent Systems, UKMAS'98
2. V. Misra, M. Khan, and U. Singh, "Supply Chain Management Systems: Architecture, Design and Vision." *Journal of Strategic Innovation and Sustainability,* 6(4), 2010
3. WanSup Um, Huitian Lu, and Teresa J. K. Hall, "A Study of Multi-Agent Based Supply Chain Modeling and Management." *iBusiness*, 2010
4. Xin Li and Sim Kim Lau, "A Multi-Agent Approach Towards Collaborative Supply Chain Management." Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 929-935.
5. Nikraz Magid, Caire Giovanni, and Bahri A. Parisa "A Methodology for the Analysis and Design of Multi-Agent Systems Using JADE." May 2006
6. Caire Giovanni, *Developing Multi-Agent Systems with JADE*. John Wiley & Sons Ltd. 2007.
7. Michael Wooldridge, *An Introduction to Multi Agent Systems*. John Wiley & Sons Ltd. 2004.
8. David Grimshaw, "JADE Administration Tutorial", Internet: http://jade.tilab.com/doc/tutorials/JADEAdmin/ March, 26, 2010 [October 2012]
9. The Foundation for Intelligent Physical Agents. Internet: http://www.fipa.org/, [September 2012]
10. Ye Chen, Yun Peng, Tim Finin, Yannis Labrou, Cost Scott, Chu Bill, Yao Jian, Sun Rongming, and BobWilhelm, "A Negotiation Based Multi-Agent System for Supply Chain Management." 1999

11. R. Kalakota, J. Stallaert, and A. B. Whinston, Implementing Real Time Supply Chain Optimiaztion System. *In Proceedings of the Conference on Supply Chain Management, Hong Kong*, 1995

12. Java 2 Platform Standard Edition 5.0 API Specification. Internet: http://docs.oracle.com/javase/1.5.0/docs/api/, 2010 [September 2012]

13. R. H. Bordini, M. Dastani, and M. Winikoff, Current issues in multi-agent systems development (invited paper). In *Post-proceedings of the Seventh Annual International Workshop on Engineering Societies in the Agents World*, volume 4457 of *Lecture Notes in Artificial Intelligence*, pages 38--61, 2007.

14. Java Agent Development Framework. Internet: http://jade.tilab.com/, January 2010 [October 2012]

15. Autonomous Agents and Multi-Agent Systems for Healthcare. Internet: http://www.openclinical.org/agents.html, 06 March 2005 [September 2012]