STATIONARY AGENTS COMMUNICATION COMPLAINT WITH THE FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS SPECIFICATIONS

Dhuha Basheer Abdullah¹, Manar Talaat Ahmed²

¹Department of Computer Science, College of Computer Science and Mathematics, University of Mosul, Iraq
  1rtdm2005@yahoo.com
  2manar_20_10_89@yahoo.com

ABSTRACT

This paper describes a fundamental characteristic of multi-agent systems, which is individual agents, communicate and interact. The focus of this research on the communication between stationary agents (non-mobile agents), this accomplished through the exchange of messages and, to understand each other, it is crucial agents agree on the format and semantic of these messages. JADE chosen (Java Agent DEvelopment framework) as a software framework to facilitate the development of the stationary agents-based applications in compliance with the FIPA (Foundation for Intelligent Physical Agents) specifications for interoperable intelligent multi-agent systems. We are adopting the stationary agents to engage in the activities usually encountered in a real e-commerce environment in simplified form to show the role of the communication that can be provided by the stationary agents in this environment.

Keywords: FIPA (Foundation for Intelligent Physical Agents), JADE (Java Agent DEvelopment framework), MAS (Multi-Agent System), Stationary Agent, DF (Directory Facilitator), ACL (Agent Communication Language), speech-act theory.

1. INTRODUCTION

The growth of applications, ranging from small systems for personal assistance to complex and mission-critical systems for industrial applications which include process control, system diagnostic, manufacturing and network management, as well as the growth in networked information resources, require information systems that can be distributed on a network and interoperate with other systems. Such systems cannot be easily realized with traditional software technologies because of the limits of these technologies in coping with distribution and interoperability. The agent-based technologies seem to be a promising answer to facilitate the realization of such systems because they were to cope with distribution and interoperability [3].

One of the key components of multi-agent systems is a communication [4]. In fact, the stationary agents need to be able to communicate with the users, with the system resources, and with each other if the need to cooperate, collaborate, negation, and so on. There are many development environments provide some predefined agent models and tools to make easy the development of systems. Moreover, some of them try to allow interoperability with other agent systems through the use of a well-known agent communication language such as KQML (Knowledge Query Manipulation Language) [9]. However, the use of a common communication language is not enough to easily support interoperability between different agent systems. The standardization work of FIPA (Foundation for Intelligent Physical Agents) is in the direction to allow an easy interoperability between agent systems, because FIPA beyond the agent communication language, specifies also the key agents necessary for the management of an agent system, the ontology necessary for the interaction between systems, and it defines also the transport level of the protocols [3], so that we used JADE (Java Agent DEvelopment framework), probably the most widespread agent-
oriented middleware in use today [4], and compliant with the FIPA specification to support communication between agents by using some special communication languages, called Agent Communication Language (ACL). Currently the most used and studied agent communication language is the FIPA-ACL [4].

2. THE FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS (FIPA)

The Foundation for Intelligent Physical Agents was established in 1996 as an international non-profit association of companies and organizations sharing the effort to develop a collection of standard relating to software agent technology [3][4]. FIPA specifications are developed through direct involvement of FIPA membership [6]. The first output documents of FIPA, called FIPA97 specifications, specify the normative rules that allow a society of agents to interoperate [4].

The fundamental aspect for any agent system addressed by the early FIPA specifications, and it is a very important for stationary agents is the agent management, so that FIPA supports a reference model of an agent platform, as shown in Fig.1, which consists the following components:

- **Agent platform (AP):** This provides the physical infrastructure in which agents are deployed. The AP consists of the machines, operating system, FIPA agent management components (DF, AMS and MTS) , agents themselves and any additional support software [6].
- **Agent Management System (AMS):** Is a mandatory component of an AP and responsible for managing the operations of an AP, such as providing the naming service (i.e. ensure that each agent in the platform has a unique name), represent the authority in the platform (for instance it is possible to create/kill agents on remote containers) [7], and each agent should be register with an AMS in order to obtain an AID. And we should not that only a single AMS can exist in each AP [6], and if AP spans multiple machines, the AMS is the authority across all those machines [4].
- **Directory Facilitator (DF):** Is an optional component of an AP, that provides a Yellow Pages service by means of which an agent can find other agents providing the services he requires in order to achieve its goal [7].
- **Agent Communication Channel (ACC):** Is the agent that provides the path for basic contact between agents inside and outside the platform.
- **Message Transport Service (MTS):** Is a service provided to transport FIPA-ACL messages between agents on any given AP and between agents on different APs [4]. Messages are providing a transport envelope that comprises the set of parameters detailing, for example, to whom the message is to be sent [3]. The general structure of a FIPA-compliant message is depicted in Fig.2.

The standard specifies also is the Agent Communication Language (ACL), that is used by agents to communicate based on message passing, where the stationary agents formulating and sending individual messages to each other. The FIPA-ACL specifies a standard message language by setting out the encoding, semantics and pragmatics of the messages [3], and this will be explained in the future in this paper.

It is importuned to point here to the requirements of the ACL , that it should be reliable and secure, where provisions for secure and private exchanges between agents should be supported and also must be robust to inappropriate or malformed messages and support reasonable mechanisms for identifying and signaling errors and warnings [9].

The communication paradigm adopted is the asynchronous message passing, where each agent has
a sort of mailbox (the agent message queue) where the JADE runtime post messages sent by other agents. Whenever a message is posted in the message queue the receiving agent is notified. If and when the agent actually picks up the message from the message queue to process it and this belong to the programmer [4].

![Fig.2 FIPA message structure](image)

### 3. JAVA AGENT DEVELOPMENT FRAMEWORK (JADE)

Java Agent DEvelopment framework, is a software development framework aimed at developing multi-agent systems and applications conforming to FIPA standards for intelligent agents, and JADE has fully coded in java because of its many attractive features, particularly geared towards object-oriented programming in distributed heterogeneous environment [5], and JADE was chosen in this paper as a development framework to implement the communication between stationary agents, because JADE provide programmers with the following ready-to-use and easy-to-customize core functionalities [4]:

- **A fully distributed system inhibited by agents**, where each running as a separate thread, may be on different distributed machines, and it has the ability of transparently communicating with one another.

- **FIPA-compliant Agent Platforms**, which includes AMS, DF, and ACC. All these three components are automatically activated at the agent platform start-up.

- **Automatic registration and deregistration of agents with AMS.**

- **A simple, yet effective, agent life cycle management.**

- **FIPA-compliant IIOP-based message transport system to connect different agent platforms.**

- **Efficient transport of ACL messages, inside the same agent platform.** In fact, messages are transferred encoded as java objects, rather than strings, in order to prevent marshalling and unmarshalling procedures.

- **Support for the usage of application-defined content languages and ontologies.**

- **FIPA-compliant naming service, where each agent at start-up obtain a GUID (Globally Unique Identifier).**

- **Support to the execution of multiple, parallel and concurrent agent activities via the behavior model [5].**

The main architectural elements of a JADE platform shown in the Fig.3, the software architecture
is based on the coexistence of several Java Virtual Machine (JVM), where each VM is a basic container of agent/agents that provides a complete runtime environment for agent execution and allows several agents to concurrently execute on the same host, and the architecture allows also several VMs to be executed on the same host, however this is discourage because of the increase in overhead and the lack of whatever benefit [3], therefore, we must take into consideration the number of cores in the computer To avoid the load on the CPU.

The software architecture provide a special container, called the main-container, which is the first container to be lunched and all other containers must join to a main container by registering to it, and the main-container has the following special responsibilities, which are very important for the stationary agents communication:

- Managing the container table (CT), which is the registry of the object references and transport addresses of all container nodes composing the platform.
- Managing the global agent descriptor table (GADT), which is the registry of all agents present in the platform, including their current status and location?
- Hosting the AMS and the DF, the two special agents that provides the agent management and white page service, and the default yellow page service of the platform [4].

JADE provides a cache of the GADT that each container manages locally to prevent bottleneck in the main-container. Platform operations, in general, do not involve the main-container, but instead just the local cache and the two containers hosting the agents which are the subject and the object of the operation (e.g. sender and receiver of the message). When a container must discover where the recipient of a message lives, it first searches its LADT (local agent descriptor table) and then, only if the search fails, is the main-container contacted in order to obtain the proper remote reference which, consequently, is cached locally for future usages [4].

4. AGENT COMMUNICATION LANGUAGE (ACL)

Agent communication language is probably the most fundamental feature of JADE and is implemented in accordance with the FIPA specifications. Most ACL are based on the speech-act theory [8]. Speech-acts are expressed by using standard keywords, also called "performative" indicating what the sender intends to achieve by sending the message such as (REQUEST, INFORM, PROPOSE, etc.). Each message includes the following parameters:

- The sender of the message.
- The list of receivers.
- Performative, FIPA message type.
• Content, which is the actual information to be exchanged.

• ConversationID, used to link messages in same conversation.

• Language, specifies which language is used in the conversation.

• Ontology, specifies which ontology is used in the content.

• Protocol, specifies the protocol is used in the conversation.

• ReplayWith, another field to help distinguish the answers.

• InReplayTo, Sender uses to help distinguish answers.

• ReplayBy, Used to set a time limit on an answer.

All messages in JADE are implemented as an object of the jade.lang.acl.ACLMessage class that provides get and set methods for accessing all fields specified by the ACL format [4].

5. WHAT IS AN AGENT?

There are several definitions of agent, but the most acceptable one is: “An agent is a software entity has the ability to assist people and act on their behalf and have the ability to interact with their execution environment, and act asynchronously and autonomously upon it” [3].

Agent is autonomous, because it operates without the direct intervention of humans and has control over its actions and internal state. An agent is social, because it cooperates with humans or other agents in order to achieve its tasks. An agent is reactive, because it perceives its environment and responds in a timely fashion to changes that occur in the environment. And an agent is proactive, because it does not simply act in response to its environment but is able to exhibit goal-directed behavior by taking initiative [4].

In particular, this research focuses on implementing the communication between stationary agents (non-mobile agents) also called static agents, which are execute only on the system where they begin execution and if need information that is not on that system or need to interact with an agent on different system, they are normally used a communication mechanism such as RPC (Remote Procedure Call) and ACL messages [2][9]. However, stationary agents have all the previous properties of the agent to achieve the goal that designed for it. We should note that, each stationary agent in the system has its own thread of control, and JVM is a container of agent/agents that provides a complete runtime environment for agent communication.

6. WHAT IS A MULTI-AGENT SYSTEM?

A Multi-Agent system (MAS) is a system composed of multiple autonomous agents that cooperate to solve a complex problem [8]. One of the most important application fields of multi-agent systems is information management. In particular, the Internet has been shown as an ideal domain for multi-agent systems due to its distributed nature and the mass of information available [4].

The Internet has also adopting the use of agent technologies in the e-commerce and business process management fields. In fact, before the spread of Internet e-commerce, business process management was almost entirely driven by human interactions: humans deciding when to buy goods, how much they are willing to pay, and so on [4]. In this research, focus of the work is on the importance of the stationary agents in communication, and we are adopting these agents to engage the activities usually encountered in a real e-commerce environment in simplified form to show the role of communication by the stationary agents as we mentioned that previously.

7. SYSTEM DESCRIPTION

E-commerce involves complex processes spanning in different areas that cover business modeling, information technology, and social aspects, taking into account the high diversity of e-commerce activities involving electronic payments, business document processing (orders, bills, requests for quotes, etc.) [1]. Since this is a long-term
undertaking, at this research the focus is on implementing a simplistic skeleton for an e-commerce simulation by adopting stationary agents that are capable of adaptive behavior in context of price negotiation to show the communication that can be done by these agents on JADE platform.

7.1 System Architecture

Basically, the following e-commerce model acts as a distributed marketplace that hosts seller agents of goods and allows buyer agents to negotiate with the seller agents and choose the seller agent from which to make purchase.

This model implementing on books trading as an example of e-commerce to show the communication between the stationary agents. However the architecture of this model consists one main-container which contains DF agent, which provides the default yellow pages service in the platform, and the AMS, is the agent who exerts supervisory control over access to and use of the agent platform as we mentioned that previously [6], and other containers distributed on different computers linked in LAN, this model can be developed in the future to be implement in WAN. Where each container can host one agent, except the main-container can host a buyer agent with its default agent (AMS, and DF), as shown in the Fig.4.

This model uses the stationary agents to facilitate communication between the system and the real-world users (shoppers and vendors), let us now describe the role of buyer and seller agents appearing in Fig.4. The vendor employs its seller agent to register his service in the DF to advertise this service to all buyers in the platform (for ex: Book-Seller service), register his products in a catalogue of pairs (products, reserve price), and implementing negotiation with the buyer agents, which need a specific product from it depending on the ACL messages to communicate with each other’s, while the shopper employs its buyer agent to search in DF for seller agents that provide a specific service and implements negotiation with them for a specific product that determined by the shopper also depending on the ACL messages. The communication between users (buyers and sellers) and their agents can be done through the GUI interfaces.

7.2 System Implementation

The model has been implemented within JADE 4.3.1. We chose JADE, because its architecture matches well with the requirements. Negotiation between seller agents and buyer agents take place in JADE containers, each container hosts either buyer agent or seller agent. This model implemented based on several Java classes organized in the following categories:

- Agent classes, used for describing the agent type. We used two types of classes one for buyer agent named (Buyer) and other for seller agent named (seller), both type extend the GuiAgent class provided by JADE, because they are associated with the class of the GUI.

- Agent activity classes, also called behaviours, used for describing the activities performed by agents in the system [7]. A behavior is implemented in JADE by extending the provided Behaviour abstract class. The class Behaviour is the root of a class hierarchy abstracting various agent behavior type [4]. It is useful to use TickerBehaviour,
SequentialBehaviour, CyclicBehaviour, SimpleBehaviour, and ParallelBehaviour in This model.

- Other classes, for implementing a graphical user interface for buyer and seller users.

As mentioned previously, the model adopting stationary agents for communication and representing the role of buyer and seller using FIPA-ACL messages. We have used the following messages: QUERY_REF, PROPOSE, ACCEPT_PROPOSAL, FAILURE, and INFORM.

7.3 System Algorithms

A conversation is a sequence of behaviours which work together toward a common goal. It is possible to shorten the role of each agent (buyer and seller agent) by the following algorithms:

7.3.1 Buyer agent algorithm:

Input: Service name and product information

Output: Result of the negotiation

- **Step1:** The user should input the service name, which is needed for specifying the seller agents, which are Specialized in a specific trading.

- **Step2:** Searching for all seller agents in the directory facilitator, those have the same entered service name.

- **Step3:** The user (buyer) enters the product name, and the budget to purchase the product within him/her budget.

- **Step4:** Sending a QUERY_REF message about a specific product that specified by the user to get the prices quotations from all seller agents distributed on the platform.

- **Step5:** Waiting few second.

- **Step6:** Receiving a PROPOSE or a FAILUR message form seller agents.

- **Step7:** Choosing the minimum price among seller agents, which have the product and its price within the budget.

- **Step8:** Waiting few second.

- **Step9:** Sending an ACCEPT-POROSEAL message to the seller agent that has the minimum price.

- **Step10:** Waiting few seconds.

- **Step11:** Receiving an INFORM message from the seller agent about the success of purchasing the product.

- **Step12:** Printing the result of the negotiation on the user interface.

7.3.2 Seller agent algorithm:

Input: Service name and catalogue of the products which contains product name and its price

Output: printing the Queries that sent by all buyers.

- **Step1:** The user enters pairs of (product, reserve price).

- **Step2:** Receiving a QUERY-REF message from the buyer agents.

- **Step3:** Waiting few seconds.

- **Step4:** Sending a PROPOSE message to the buyer agent whether the product found in the catalogue, otherwise sending a FAILUR message.

- **Step5:** Waiting few seconds.

- **Step6:** Receiving a QUERY-REF message from the buyer agents.

- **Step7:** Sending an ACCEPT-PROPOSAL message from the buyer agent.

- **Step8:** Sending an INFORM message to the buyer agent about the success of purchasing the order.

- **Step9:** View queries sent by all buyer agents on the user interface.

7.3.3 Removing unwanted message algorithm
We should pointed to a very important point, to prevent infinite waiting we using timeouts, but Some messages may be arrived late and slow down operations, because every receive with a template will have to examine all messages in the queue before getting to the ones they looking for. These messages can be removed by using TickerBehaviour class. TickerBehaviour class, periodically, checks the queue and if the message seen in the previous, it is considered an unwanted message and removing it from the queue, otherwise it should return it to the queue and this behaviour works independently, we can shorten its work as the following algorithm:

**Input:** Read Messages from the mailbox message queue.

**Output:** Return untreated message.

- **Step1:** Reading a message from the mailbox (message queue) of the agent.
- **Step2:** Removing the message whether it was seen in the previous, otherwise put it into a temporary queue.
- **Step3:** Redo Step1 until the mailbox (message queue) is empty.
- **Step4:** Put back all messages in the temporary queue to the mailbox (message queue).

### 7.4 The flow of the system

The flow of the process for removing unwanted messages and the implementation of the buyer, and seller agent shown in the Fig.5, Fig.6, and Fig.7, respectively, to clarify the implementation for each one of them by using flowcharts to describe the progress of the implementation.
7.5 System Application

Here, we introduce a simple experiment to illustrate main features of this implementation. In order to run the experiment we set-up JADE platform on three computers: comp1, comp2 and comp3. On computer comp1 the main-container is initialized, and on computers comp2 and comp3 second
containers (container-1 and container-2) respectively will be initialized and register with the main-container dynamically.

We have chosen a simple scenario that implemented on book trading as an example of e-commerce applications. In this scenario one book-buyer agent is hosted in the main-container, and two book-seller agents, one of them is hosted in the container-1 and other in the container-2. Each book-seller agent must register its service such as a (Book-Seller service) in the DF to be able searched by each book-buyer agent need this service to buy a specific book from it. The vendor uses the GUI to enter his service and to add pairs of (product, reserve price) in a specific catalogue and can be retrieved these products by using also the same GUI. By this GUI the vendor can be shown all the queries that sent to him from different buyer agents in the platform. Each shopper needs to buy a specific book should be searching for all seller agents in the platform, which are support the Book-Seller service, and then specifies the book that need to buy it with him/her Budget (i.e. maximum price for this book) and then press on the negotiation button to implement the negotiation process with the book-seller agents, then the result of the negotiation will be presented in the GUI of the shopper to learn him/her about the success or the failure of the negotiation. The GUI interfaces of the implementation are shown in Fig. 8 and Fig. 9 respectively.

![Fig.8 Screen caption showing buyer GUI.](image)
8. CONCLUSIONS

This paper aims to show the importance of the stationary agents for communication in Multi-Agent systems in accordance with the FIPA specifications by using Agent communication language (ACL) to exchange messages between agents. A simple scenario for an e-commerce simulation implemented by adopting the stationary agents that have adaptive behavior in context of price negotiation to show the communication that can be done by these agents on JADE platform, this implementation is done on JADE 4.3.1. JADE was chosen, because its architecture matches well with all needed requirements in the implementation of this model that mentioned in this paper.

9. REFERENCES


10. AUTHOR’S PROFILE

Dhuha Albazaz is the head of Computer Science Department, Collage of Computer and Mathematics, University of Mosul. She received her PhD degree in computer science in 2004 in the specialty of computer architecture and operating system. She supervised many Master degree students in operating system, computer architecture, dataflow machines, mobile computing, real time, and distributed database. She supervised three PhD student in FPGA field, distributed real time systems, and Linux clustering. Currently she supervises PhD student in cloud computing field. She leads and teaches modules at both BSc, MSc, and PhD levels in computer science. Also, she teaches many subjects for PhD and master students.

Manar Seyala is a master student in Computer Science Department, College of Computer and Mathematics, University of Mosul. She interested with network security, operating system and distributed databases subjects.