CHAPTER FOUR: IMPLEMENTATION USING JADE

The valuation of the proposed model was done through prototyping the model into widely used open source as a platform for agent development. The prototype cope the four agents, and the messaging between the nodes. By using the jade we cane extant the domain of control into newly joined nods.

4.1 Agent Development

Agent technology is becoming increasingly important with every passing day since it promises to change software construction, as stated in (Jennings, 2001). One of the areas agent technology will impact is the software maintenance of remote systems. The Multi-agent Remote Maintenance Shell (MARMS) system is one example. The MA-RMS (Kusek et al, 2003), (Jezic et al,2003), (Lovrek et al, 2003) is an agent based framework which supports the maintenance and control of software. By using agents, the MA-RMS framework can automatically perform different operations at remote locations with very little human interaction.

4.2 Jade

Jade (Java Agent Development Framework) (JADE, 2009) is an agent platform designed to ease the development of multi-agent systems by providing standard agent services and methods. It also enables interoperability between different agent platforms since it complies with FIPA standards. The Jade platform is composed of one main container and several containers that can be launched on one or more hosts distributed across the network. Each container can run zero, one or multiple agents. The following agents and services must be present at the main container at all times: the AMS (Agent Management System), the DF (Directory Facilitator) and the MTS (Message Transport System) which is also referred to as the ACC (Agent Communication Channel).

4.2.1 Jade Messaging

Communication is a very important aspect of agent platform architecture. Jade communication is performed by using different MTPs (Message Transport Protocol). The MTP used depends on the receiver agent's location. If an agent is located on the same container, the event passing method is used. This means that the message is not sent over the network, but cloned, and a new object reference is passed to the receiver agent. If the receiver is located on another container, RMI (Remote Method Invocation) is used.
When one agent wants to send a message to another agent located on different container, a new ACL (Agent Communication Language) (ACL, 2009) message must be created. An ACL message is a FIPA standard which defines different parameters used in agent communication. Only the performative (communication act) and the receiver address parameter must be set. The remaining parameters are optional. The data being sent is then coded with the proper codec and placed in the content slot of the message. The entire ACL message is then encoded. Finally, the encoded message is placed in the envelope of the transport message.

4.2.2 Agent Communication Language (ACL) Messages

The class ACL Message represents ACL messages that can be exchanged between agents. It contains a set of attributes as defined by the FIPA specifications. An agent willing to send a message should create a new ACL Message object, fill its attributes with appropriate values, and finally call the method *Agent.send()*. Likewise, an agent willing to receive a message should call *receive ()* or *blockingReceive()* method. Sending or receiving messages can also be scheduled as independent agent activities by adding the behaviours ReceiverBehaviour and SenderBehaviour to the agent queue of tasks.

4.3 Detection Agent task

The activities or the jobs that an agent has to do are carried out within a "behavior". A behavior represent a task that an agent can carry out and it implement as an object of class that extends *jade.core.behaviors.behavior* . To make an agent execute the task implemented by a behavior object. The behavior must be added to the agent by means of the *addbehaviour()* method of the Agent class.
Functions or classes of D agent:

- Software_Executions_Detections ()
- Instruction_Detections ()
- Instructions_Analyze ()
- Instructions_Executions_Evaluations ()
- Replications_detections ()

Brief elaboration of this class is described in the sections below. A diagram to better show the prototype is presented.

- **Software Executions Detections**

Software execution detection is a cyclic behavior, thus it execute its behavior repetitively, and send a message to the next behavior. Meanwhile the other agent functions in mode of sleep in order not to consume CPU time, and it will wake up when the behavior has been activated. The SED will send a message to the appropriate operation in the D agent or to the others agent according to the

- **Instruction Detections**

Instruction detection is to identify which instruction is going to be executed, when the instruction is known, the function would send a message to another function. Furthermore, the time ID finished its functions it goes back to sleep mode again. Figure 4-1 shows the process.

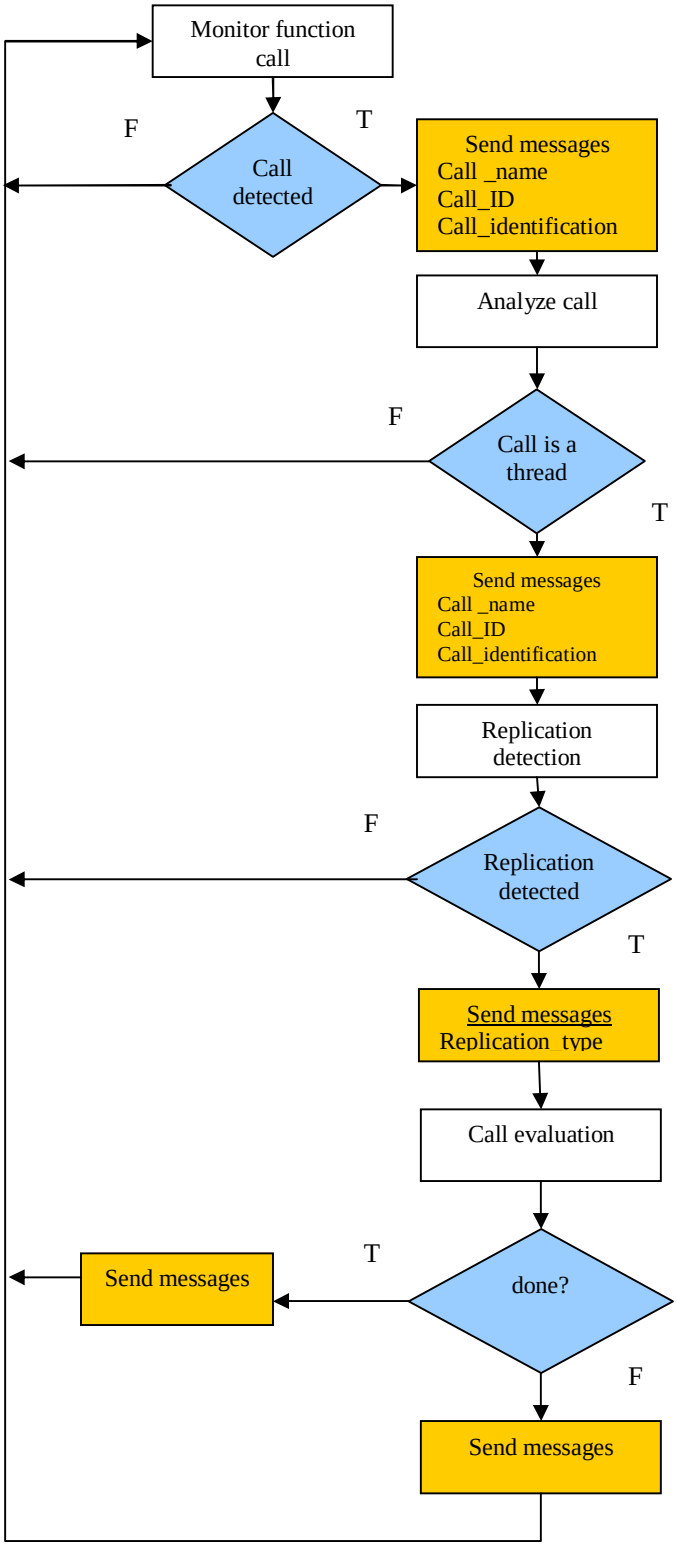

Figure 4-1: Messages flow

- **Instructions Analyze**

Analyzing instructions is to make sure that this instruction is not mean to harm any specific component, otherwise in considered as a harmful and malicious, therefore actions, must be taken in order to eliminate of regenerate. In this case IA will send according to the scenarios tow message, the first is to the preventions agent and the second to the replications and configuration agent.

IS has both receive and send message. It receive message to enter working mode and send message to other functions. After IS finishes its job it goes back to sleep mode in order not consume CPU time.

- **Instructions Executions Evaluations**

Evaluations of instructions specified by the Analyzer passed to IEE to determine whether it has been done or not, if the instructions is done then the IEE send a message to the appropriate function or agent according to the scenarios specified before, and also if the instruction is done IEE will send a message to the particular functions informing them to take actions.

- **Replications detections**

If the system experience a replications the function send message to other agent to tackle the replication or do the necessary action to replicate component in the other node.

In the communication of message interaction in JADE is implemented in accordance with the FIPA specification. The communication is based on asynchronous message passing. Each agent has its own message queue that gather every all the messages passed to it. The format of message in JADE is compliant with that define by FIPA-ACL , that mean each message include the following field:

- The sender of the messages
- The receivers
- The communicative act
- The content containing
- The ontology

4.3.1 Sending message:

A message in JADE is implemented as an object of the jade.lang.acl.ACLMessage class that provide get and set methods for accessing all field specified by the ACL format. In order to well define semantic is important to make agent make proper decision when it receive the message.

4.3.2 Receiving message:

In Jade when sending message, it has to be posted in receiver message queue. A receiver ca pick up the message from the queue by mean of receive(). Method this method returns the first method in the queue and return nul if the message in the queue is empty

4.3.3 Message signaling and broadcasting

Actually, message sending is to return a information about the events, as we sees in detection agent , when a new function call is detected , a messages would be send containing all information about the call for analysis . Furthermore, after analysis if the call is threat it has to be prevented and the post it for a replication analyzes. if replication found a message would be sent to all nodes in the network, every nodes receives messages will reply back, the ontology of the messages is depend on the availability of nodes in the networks, the nodes are given an ID . The Broadcast for replication is as follow:

Sender node: initiate message, e.g. <replication>, <replication_B>.

4.4 Replication and Configuration and Run time Management Agent

Replication in activated after receiving the thread message from detection agent as well as prevention agent. At the moment the message received and according to the ontology, the replication and configuration, the agent identifies the message if the message means to replicate component, nodes replicate component. If the message means localization of component and configuration, the agent localizes and configure. Figure below illustrate the process of receiving message and configuration

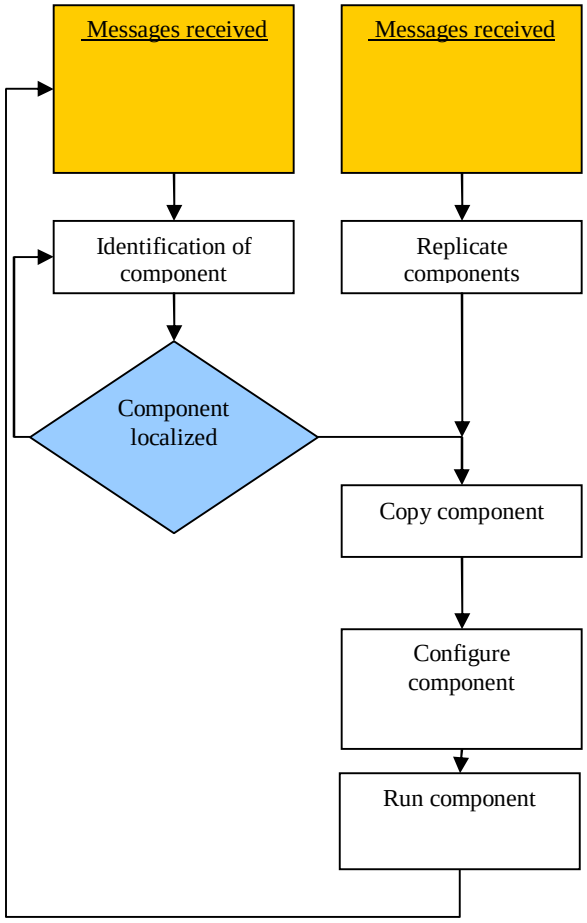

Figure 4-2: Replication and configuration and run time Messages flow

- **Message broadcasting**

Messages in this agent are received for replication of component, and also received for localization of component.

Sender node: initiate message, e.g. <request_ID>
Receiver's nodes reply: the receiver initiate respond, e.g. <speed_bandwith_ID>

In the stages of replicating component for availability, we have implemented a selection algorithm. First the replication and configuration agent send messages to all of the nodes in the network, the receiver reply a message containing the speed and bandwidth they have, based on this information the receivers select an appropriate node to replicate its component, figure 4-3 below shows the process of selecting a node.

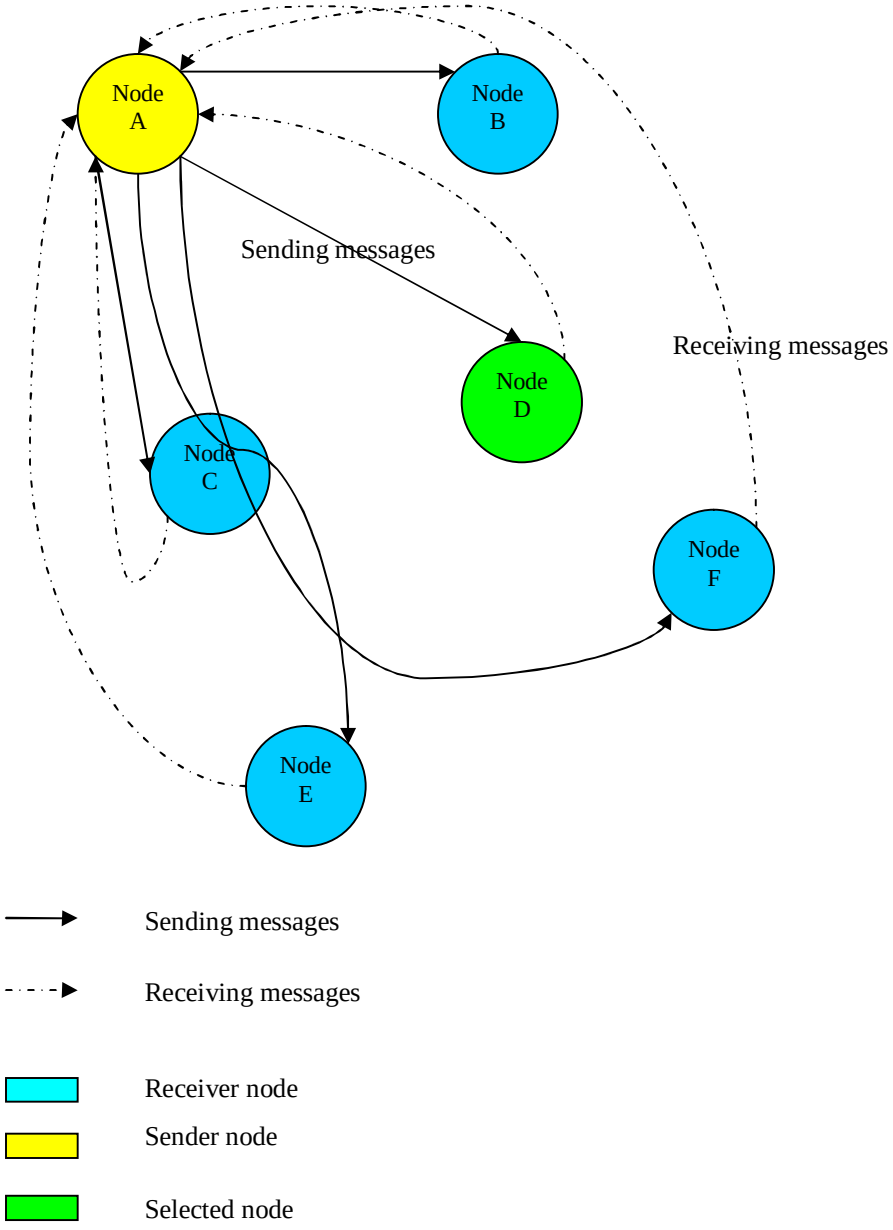

Figure4-3: the selection of node and messaging

4.5 Prevention:

The prevention requires ontology to differentiate the messages received. In this agent there tow different message to received. The first is when the thread call identified, the prevention agent received message saying to prevent the call from been executed. The second is when there is replication detected, in this case the agent prevent the replication and broadcast messages to all nodes available, the objectives of this messages is to inform that there is a attack threaten to propagate across the network. Figure below explain the function of the prevention agent.

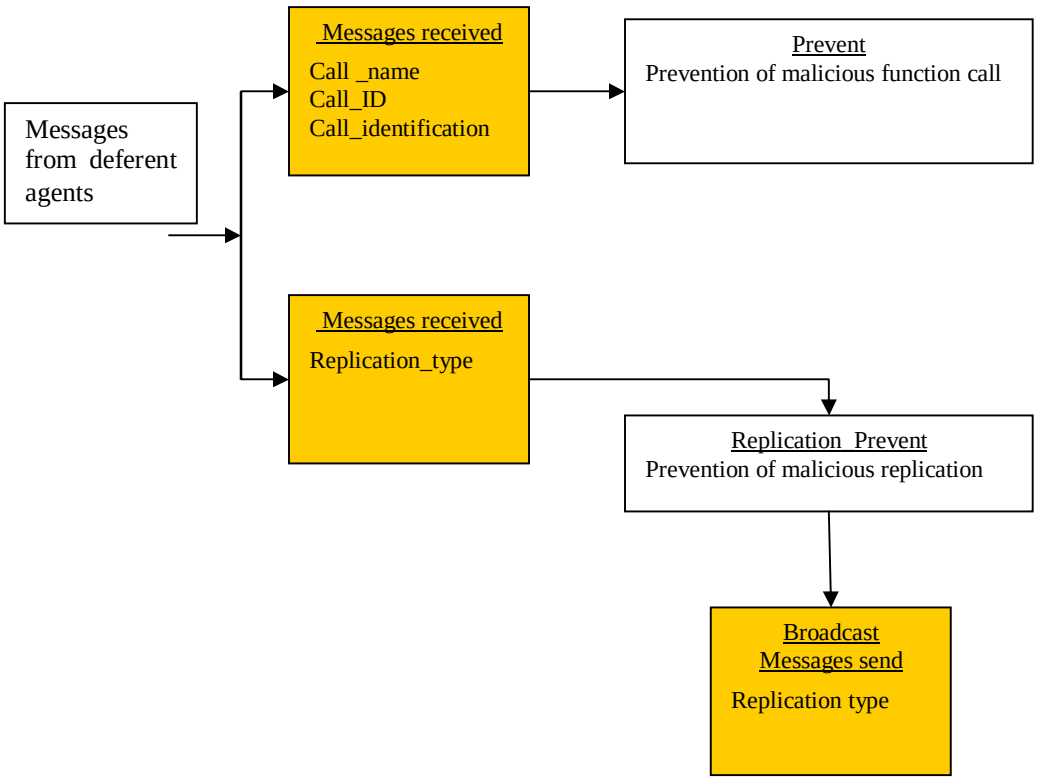

Figure 4-4: The prevention Agent messaging

- **Message broadcasting**

Apparently, message broadcasting is to all nodes in the network, every nodes receives messages will reply back, the ontology of the messages is depend on the availability of nodes in the networks; the nodes are given an ID. e.g.

- Sender node: initiate message, e.g. replication_A, replication_B.
- Receiver's nodes reply: the receiver initiate respond

4.6 Summery

In this chapter we implement the model using java agent development framework (JADE), the messaging in jade is using specific ontology, based on jade ontology we can build a powerful and complex agent messaging between nodes. This will make available and facilitate network nodes to self manage their component.