CHAPTER 8

DEPLOYMENT AND TESTING

After completing the steps of analysis and design, the features of JADE platform have been explored for the implementation of multi-agent system. With reference to the design phase, the following agents have been developed using Java language: Generic Agent, DB Agent, EEG Agent, ECG Agent and EMG Agent. The required behaviors and actions were implemented as per the design guidelines and FIPA recommendations. The required table has been created in Database which is accessible through Java DataBase Connectivity (JDBC).

8.1 REMOTE MONITORING AGENT

The Remote Monitoring Agent (RMA) allows controlling the life cycle of the agent platform and of all the registered agents. The distributed architecture of JADE allows also remote controlling, where the GUI is used to control the execution of agents and their life cycle from a remote host.

RMA is a Java object, instance of the class jade.tools.rma.rma and can be launched from the command line as an ordinary agent (i.e. with the command javajade.Boot myConsole:java.tools.rma.rma), or by supplying the ‘-gui’ option the command line parameters (i.e. with the command java jade.Boot –gui).

More than one RMA can be started on the same platform as long as every instance has a different local name, but only one RMA can be executed on the same agent container.
The followings are the commands that can be executed from the menu bar (or the tool bar) of the RMA GUI.

- **File menu:**
  
  This menu contains the general commands to the RMA.

  - **Close RMA Agent**
    
    Terminates the RMA agent by invoking its doDelete() method. The closure of the RMA window has the same effect as invoking this command.

  - **Exit this Container**

---

**Figure 8.1 Snapshot of the RMA GUI**

![Image of RMA GUI snapshot]

This image shows the RMA GUI with a list of containers and platforms. The menu bar and tool bar are visible at the top of the window.
Terminates the agent container where the RMA is living in, by killing the RMA and all the other agents living on that container. If the container is the Agent Platform Main-Container, then the whole platform is shut down.

- **Shut down Agent Platform**
  
  Shut down the whole agent platform, terminating all connected containers and all the living agents.

- **Actions menu:**
  
  This menu contains items to invoke all the various administrative actions needed on the platform as a whole or on a set of agents or agent containers. The requested action is performed by using the current selection of the agent tree as the target; most of these actions are also associated to and can be executed from toolbar buttons.

- **Start New Agent**
  
  This action creates a new agent. A window is prompted for the name of the new agent, the name of the Java class the new agent is an instance of, arguments, owner and container. Moreover, if an agent container is currently selected, the agent is created and started on that container; otherwise, the user can write the name of the container he wants the agent to start on. If no container is specified, the agent is launched on the Agent Platform Main-Container.

  The window looks as follows once the GenericAgent is started. The GenericAgent in turn invokes the GenericGUI where the user is prompted to enter the Social Security Number and type of signal to be processed. As the
submit button is pressed, it invokes the FileGUI in which the user has to select the data file.

![RMA window with multi-agents](image)

**Figure 8.2 RMA window with multi-agents**

Based on the type of signal, the GenericAgent invokes the specific agent available in the agent platform which in turn processes the signal available in the data file. The result is being sent to the GenericAgent and it stores / updates the result in the Database using the Java DataBase Connectivity.

- **Kill Selected Items**
  
  This action kills all the agents and agent containers currently selected. Killing an agent is equivalent to calling its doDelete() method, whereas killing an agent container kills all the agents living on the container and then de-registers that container from the platform. Of course, if the Agent Platform Main-Container is currently selected, then the whole platform is shut down.
• **Suspend Selected Agents**
  This action suspends the selected agents and is equivalent to calling the doSuspend() method. Beware that suspending a system agent, particularly the AMS, deadlocks the entire platform.

• **Resume Selected Agents**
  This action puts the selected agents back into the AP_ACTIVE state, provided they were suspended, and works just the same as calling their doActivate() method.

• **Send Custom Message to Selected Agents**
  This action allows to send an ACL message to an agent. When the user selects this menu item, a special dialog is displayed in which an ACL message can be composed and sent, as shown in the figure.

• **Migrate Agent**
  This action allows to migrate an agent. When the user selects this menu item, a special dialog is displayed in which the user must specify the container of the platform where the selected agent must migrate. Not all the agents can migrate because of lack of serialization support in their implementation. In this case the user can press the cancel button of this dialog.
8.2 DUMMYAGENT

The DummyAgent tool allows users to interact with JADE agents in a custom way. The GUI allows composing and sending ACL messages and maintains a list of all ACL messages sent and received. This list can be examined by the user and each message can be viewed in detail or even edited. Furthermore, the message list can be saved to disk and retrieved later. Many instances of the DummyAgent can be started as and where required.
The DummyAgent can both be launched from the Tool menu of the RMA and from the command line, as follows:

```
```

![Figure 8.4 Snapshot of the DummyAgent GUI](image)

### 8.3 DF GUI

A GUI of the DF can be launched from the Tools menu of the RMA. This action is actually implemented by sending an ACL message to the DF asking it to show its GUI. Therefore, the GUI can just be shown on the host where the platform (main-container) was executed.
By using this GUI, the user can interact with the DF: view the descriptions of the registered agents, register and deregister agents, modify the description of registered agent, and also search for agent descriptions.

The GUI allows also to federate the DF with other DFs and create a complex network of domains and sub-domains of yellow pages. Any federated DF, even if resident on a remote non-JADE agent platform, can also be controlled by the same GUI and the same basic operations (view/register/deregister/modify/search) can be executed on the remote DF.

![Figure 8.5 Snapshot of the GUI of the DF](image)

8.4 INTROSPECTOR AGENT

This tool allows to monitor and control the life-cycle of a running agent and its exchanged messages, both the queue of sent and received messages. It allows also to monitor the queue of behaviours, including executing them step-by-step.
Agents can be passed to the Introspector Agent in the same way as to the Sniffer Agent via the command line, or via a configuration file. The specification of performative filters, e.g. inform, agree, etc. is not supported by the Introspector Agent.

The agents were also deployed and tested on various machines of different platforms. After successful completion of Test deployment, the agents were deployed on the designated systems in order to perform System Testing. Different inputs were tried to make that the agents are performing their tasks as required. The complete Testing process has been carried out after deploying the agents on JADE Platform. At the time of deployment the naming conventions as per the FIPA recommendations has been strictly adhered.