UNIVERSITAT ROVIRA i VIRGILI

Introduction to **M**ulti **A**gents **S**ystems

First Implementation

Course 2020-2021
November 18th, 2020

Students:

Roger Marrugat	-	*roger.marrugat@estudiantat.upc.edu*
Albert Ferrando	-	*albert.ferrando.garrido@estudiantat.upc.edu*
Victor Badenas	-	*victor.badenas.crespo@estudiantat.upc.edu*
Ronald Rivera	-	*ronald.rivera@estudiantat.upc.edu*
Laia Seijas	-	*laia.seijas@estudiantat.upc.edu*

Abstract

The aim of this second implementation is to develop the user, manager and classifier agent with the architecture established in the first activity of Introduction to Multi Agents Systems subject. In addition, we develop logic to evaluate different scenarios and data variation.

Contents

1. Introduction

This document contains all the information related to the First Implementation of the IMAS practical work. It contains 5 different sections divided as follows: on Section 1 a brief introduction of the delivery is presented, on Section 2 a description of the methodology followed in the implementation for each agent can be found, on Section 3 a brief description of the future work is detailed, on Section 4 a description of how to execute the system is described and finally, on Section 5 the E-portfolio developed for this project is illustrated.

2. Tasks Done

2.1 User agent

The User agent interacts with the human in order to start the MAS (Multi Agent System) according to the requirements that the user specifies and provides the user an interface to send requests to the system and receive an appropriate response. User agent implementation correctly handles:

- Setup: The agent properly sets up its registration in JADE DF (Directory Facilitator).
- TakeDown: The agent properly takes down its registration in JADE DF (Directory Facilitator).
- Behaviour: This agent has a cyclic behaviour where 1) It waits for human petitions and check they are valid, 2) It sends the request to the Manager agents and 3) Waits for the Manager agent confirmation. It distinguishes between initialization and query petitions.

2.2 Manager Agent

The Manager agent manages the requests received from the *User Agent* and creates *Fuzzy Agents* to handle them. Acts as the gateway between the *User Agent* and the set of *Fuzzy Agents*.
In our implementation, this agent is able to create different Fuzzy agents in a dynamic way, it means that it will create these agents depending on the number of Fuzzy agents specified in the configuration file. Manager agent implementation correctly handles:

- Setup: The agent properly sets up its registration in JADE DF (Directory Facilitator).
- TakeDown: The agent properly takes down its registration in JADE DF (Directory Facilitator).

2.3 Fuzzy Agent

The Fuzzy agent is defined by a *Fuzzy Set*,they embed a *Fuzzy Inference Engine* and represent the set of available resources which are used to handle the data and infer the final response that solves the problem.

- Setup: The agent properly sets up its registration in JADE DF (Directory Facilitator). Additionally, the agent has to load the fis file requested by the Manager Agent and read which inputs and outputs it needs to handle.

- Takedown: The agent properly takes down its registration in JADE DF (Directory Facilitator). It also removes the fis data.

- Behaviour: This agent has a cyclic behaviour in which waits for a message, if the message is a request and its not null, tries to infer from the data. If it is not capable to do so, it will return a FAILURE, but if it’s able, it will return a SUCCESS message with the inferend value to the manager agent.

3. Future Work

As a future work to do in general for all the agents we have identified the following tasks at this moment:

- Implement the exceptions control properly for all the agents in order to obtain the desired behaviour for the practical work (e.g. IO exceptions generated from reading / writing files).

- Implement a routine for the treatment of special or undesirable situations.

- Define behaviour for unexpected ACLMessages.

- Make it robust against user mistakes (mispellings in the file names, etc)

- Improve the logger format in order to give the user a better understanding about what is happening in the project.

4. System Execution

In this section is detailed how the project should be executed in both systems: Windows and Linux. Also, is detailed how to execute the project with the ID used in our team: IntellIJ.

4.1 Instructions for executing in Windows

1) Compile the agents

```
javac -cp libs\jade.jar:libs\jFuzzyLogic.jar src\Agents\*.java src\Behaviours\*.java src\Utils\*.java
```

2) Execute the project

```
java -cp libs\jade.jar;libs\jFuzzyLogic.jar;src jade.Boot -gui -name imas-platform -agents "UserAgent”
```

4.2 Instructions for executing in Linux

1) Compile the agents

```
javac -cp libs/jade.jar:libs/jFuzzyLogic.jar src/Agents/*.java src/Behaviours/*.java src/Utils/*.java
```

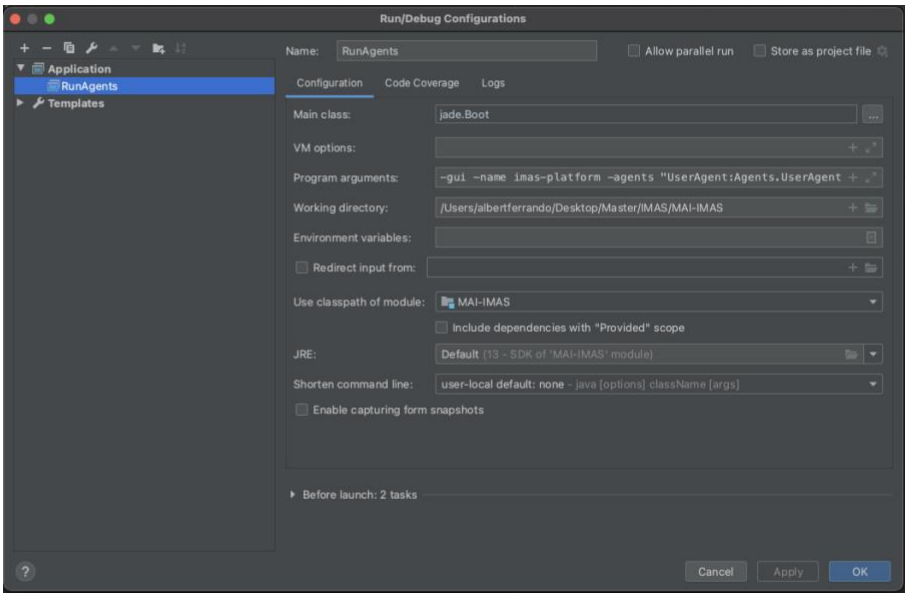
2) Execute the project

```
java -cp libs/jade.jar:libs/jFuzzyLogic.jar:src jade.Boot -gui -name imas-platform -agents "UserAgent”
```

4.3 Instructions for executing with IntelliJ

1) Open the project with IntelliJ IDEA

2) Create a new Run configuration:



Run/Debug Configurations

Name: RunAgents

Main class: jade.Boot

VM options:

Program arguments: `-gui -name imas-platform -agents "UserAgent:Agents.UserAgent`

Working directory: /Users/albertferrando/Desktop/Master/IMAS/MAI-IMAS

Environment variables:

Redirect input from:

Use classpath of module: MAI-IMAS

Include dependencies with "Provided" scope

JRE: Default (13 - SDK of 'MAI-IMAS' module)

Shorten command line: user-local default: none - java [options] className [args]

Enable capturing form snapshots

Before launch: 2 tasks

3) As in the image, the Main class should be 'jade.Boot', and the program arguments '-gui -name imas-platform -agents "UserAgent:Agents.UserAgent;ManagerAgent:Agents.ManagerAgent"'.
4) Run the project using the Run configuration created.

5. E-Portfolio

Our E-Portfolio has been implemented using Github. Our repository is hosted at: https://github.com/vbadenas/MAI-IMAS. There, you can find the following items of the project:

- The source project code
- The documents delivered in the IMAS subject
- A wiki page with a recording of the meetings performed in the subject
- A repository history with all the commits done during the project[1]

5.1 Meetings

A record of the team meetings can be found in the repository wiki page: https://github.com/vbadenas/MAI-IMAS/wiki

5.2 Tasks Assignation

Due to the actual COVID-19 situation, it's not possible for the team to meet in person and work physically together, so in the different meetings performed during this First implementation task, the work has been splitted into subgroups and later revised by all the team members.

We have decided to divide the implementation into 4 different parts:

- Communication (Albert Ferrando and Laia Seijas)
- User Agent (Albert Ferrando and Laia Seijas)
- Manager Agent (Roger Marrugat and Ronald Rivera)
- Fuzzy Agent (Victor Vadenas)

Once the whole implementation has been done and tested, this report has been written by all the team members during the last meeting.

[1] Needs to be taken into account that as some work was splitting in pairs, it could happen that one of the pair members does not appear in the commits history. Also needs to be taken into account that as the implementation was not completely independent, the integration of the code has been made in the meetings, so it's possible that one of the members appears more than the others because of this integration commits.

However, we would like to highlight the fact that although the work has been splitted into different parts, all the team has participated in the different implementation decisions discussed along the different meetings.

For holding the tasks, one project per task has been created in our github: https://github.com/vbadenas/MAI-IMAS/projects.