West University of Timişoara
April, 2016

Bogdan Moldoveanu
Kristine Varga
Cristian Schuszter
Contora Sebastian

Jade application with Jess
Intelligent systems

1. Introduction

The purpose of this application is to display the way the JADE (Java Agent Development) framework can be connected to the Jess rule engine in order to provide an intelligent distributed platform, one that can make complex decisions in systems.

2. Installation

To install the application, you must link the jade and jess jars from the lib folder of the app to the java classpath. Afterwards, the main class of the project must be jade.Boot, starting up the local agent platform.

The arguments used in our example are:

```
-gui
jess:jess.JessEngineAgent;sensor1:temp.TemperatureSensorAgent("35.0");sensor2:temp.Tem
peratureSensorAgent("11.0");master:jess.Master
```

The ***Master*** agent, jess engine agent and at least one ***TemperatureSensorAgent*** have to be instantiated for the entire application to work as designed.

3. Case study

The goal of this application is to present the potential management of a network of temperature sensors automatically. The sensors are queried by a central master which then “asks” the Jess Agent to determine which locations should have their heating enabled or disabled, based on the sensor temperature readings.

Starting the application, you will notice that each sensor prints out status information about itself in the console, as such:

```
Agent: sensor1@192.168.0.115:1099/JADE [My current temperature -> 24.0]
Agent: sensor2@192.168.0.115:1099/JADE [My current temperature -> 11.5]
Agent: sensor1@192.168.0.115:1099/JADE [My current temperature -> 23.5]
```

When one of the agents' temperature falls below 10 degrees, the heating starts for that agent and the temperature begins to rise slowly. When it gets above 30 degrees, the cooling for that agent starts.

4. Architecture

The Jess API is encapsulated by an agent, one that listens for incoming messages telling it to assert facts or run queries and then sending the answer back to whoever wanted it.

The ***JessAgent*** implements two behaviors:

- ***JessBehaviour*** which runs the main engine and asserts facts or runs queries

- ***JessListener*** which listens for messages from the outside and parses them, sending the Jess code to the ***JessBehaviour .***

The ***JessBehaviourBase*** class has methods defined for the discovery of other agents in the network. For example,the Master agent needs when he communicates to the temperature sensors, finding them first.

TemperatureSensorAgents - they represent the sensors registering the temperature at each site. The Agent has three behaviours

- Two cyclic behaviours, one for receiving SET messages from the jess engine and one for sending their current temperature to the master
- One custom ***TemperatureChangeBehaviour*** which acts like a ticker and changes the temperature of the agents.

The normal flow of the application is as follows:

1. Each agent ***TemperatureSensorAgent*** lowers its temperature by default, provided by the ***TemperatureChangeBehaviour***.

2. The master polls all the agents for their temperatures.

3. The temperatures are all collected and sent over to the ***JessEngineAgent***.

4. The ***JessEngineAgent*** asserts facts into memory and then queries it, finding all the
    - `(modify-temperature) facts provided by the (defrule set-is-heating)`

5. The “decisions” are sent over to each agent by a ***CalibrateSensorTemperatureBehaviour*** and he modifies the temperature according to the setting from the rule engine.

6. The ***TemperatureSensorAgent*** sends back a reply to the ***JessEngineAgent***, informing it that it has changed the temperature and that it can clean up after it so he doesn’t get excessive messages.