Developing Intelligent Agent Applications with JADE and JESS

Bala M. Balachandran

Faculty of Information Sciences and Engineering
University of Canberra
ACT 2601, Australia
bala.balachandran@canberra.edu.au

Abstract. Agent systems differ from more traditional software systems because agents are intended to be independent autonomous, reactive, pro-active and sociable software entities. Due to these unique characteristics developing agent systems has been a very challenging task for agent researchers and application developers. JADE (Java Agent DEvelopment Framework) is an agent development tool, implemented in JAVA and FIPA-compliant. Although JADE provides all the mandatory components (FIPA) for the development of autonomous agents, it lacks the ability to include intelligent behaviour to individual agents. JESS (Java Expert System Shell), a rule-based programming environment written in Java, provides a powerful tool for developing systems with intelligent reasoning abilities. This paper examines the use of both JADE and JESS for the development intelligent agent systems and shares the experience of the authors in the development of a Personal Travel Assistant (PTA).

Keywords: Intelligent Agents, AOSE, Multi-Agent Systems, JADE, JESS, FIPA, MaSE, AgentTool.

1 Introduction

An *agent* is a software entity that applies Artificial Intelligence techniques to choose the best set of actions to perform in order to achieve a goal specified by the user. An agent's characteristics include proactive, dynamic, autonomous and goal-oriented. A *multi-agent system* may be defined as a collection of autonomous agents that communicate between them to coordinate their activities in order to be able to solve collectively a problem that could not be tackled by any agent individually [17][18]. In recent times there has been a growing interest in the application of agent-technology to problems in the domain of E-commerce [8]. In this paper our interest is to design and develop a multi-agent system that is capable of producing travel packages for customers based on their personal preferences. We have chosen JADE (**J**ava **A**gent **DE**velopment Framework) which is very popular because of its open source, simplicity and compliant with the FIPA specifications [3][9][14]. JADE conceptualises an agent as an independent and autonomous process that has an identity, possibly persistent, and that requires communication (e.g. collaboration or competition) with other agents in order to fulfill its tasks. This communication is implemented through asynchronous message passing and by

I. Lovrek, R.J. Howlett, and L.C. Jain (Eds.): KES 2008, Part III, LNAI 5179, pp. 236 – 244, 2008.

using an Agent Communication Language (ACL) with a well-defined and commonly agreed semantics. The development environment incorporates a set of graphic tools, which facilitates the platform management, providing support to agent's execution, debugging and monitoring.

In order to incorporate intelligence with JADE agents, we use JESS (Java Expert System Shell), a rule engine and scripting environment written entirely in Java by Ernest Friedman-Hill at Sandia (Livermore) [10]. JESS was originally inspired by the CLIPS expert system shell, but has grown into a complete, distinct Java-influenced environment. In solving problems with rules, JESS uses the fast and efficient Rete algorithm which involves building a network of pattern-matching nodes. The strength of Rete comes from the fact that it uses a set of memories to retain information about the success or failures of pattern matches during previous attempts.

The rest of this paper is organised as follows. Section 2 describes our application domain and our case study. Section 3 presents the processes involved in the analysis and design of agent systems using MaSE and AgentTool. Section 4 describes the steps involved in developing multi-agents with JADE. Section 5 explains an approach for embedding JESS in JADE agents. Finally, the last section presents our conclusions and some lessons learned form this project.

2 Case Study: A Personal Travel Assistant (PTA)

Multi-agent systems are ideally suited to represent problems that have multiple problem solving entities. Such systems have the traditional advantages of distributed and concurrent problem solving strategies. In this research we aim to design and develop a multi-agent system called **Personal Travel Assistant (PTA).** Recently there have

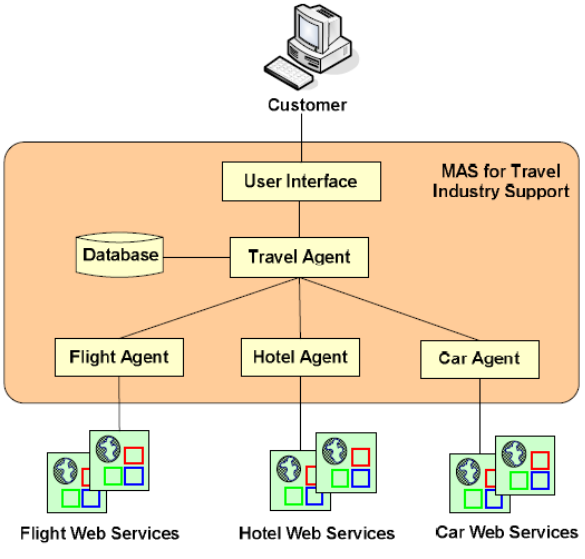

Fig. 1. The software architecture for the Personal Travel Assistant (PTA) System

been some efforts in applying agent technology for travel support systems [2][7][12]. The system will be capable of finding optimum travel packages for the customer depending on their preferences. The travel package is composed of a flight ticket, a hotel accommodation and a car rental. Figure 1 shows the proposed system architecture for our project.

The Travel Agent is the main agent that takes user preferences and requirements as input and produces a set of booking results as output. The Travel Agent communicates with other agents such as Flight Agent, Hotel Agent and Car Agent to perform its tasks. The Travel Agent also handles conflicts between the other agents in regard to fulfilling the user requirements.

3 Analysis and Design of Multi-agent Systems Using MaSE and AgentTool

There have been several proposed methodologies for analyzing, designing and multi-agent systems [13]. The most widely published methods include Gaia methodology, MaSE (Multi-agent Systems Engineering) and Prometheus [19][5][6][16]. Among these methods MaSE is more detailed and provides more guidance to system designers. We have chosen MaSE and its support tool: AgentTool for our system analysis and design. Its goal is to guide a system developer from an initial system specification to a multi-agent system implementation. The MaSE methodology consists of two phases: analysis and design. The first phase, Analysis, focuses on capturing goals, applying use cases and refining roles. The second phase, Design, focuses on creating agent classes, designing interactions, assembling agent classes, and building deployment diagrams. MaSE is supported by the agentTool [1] which is a graphically based

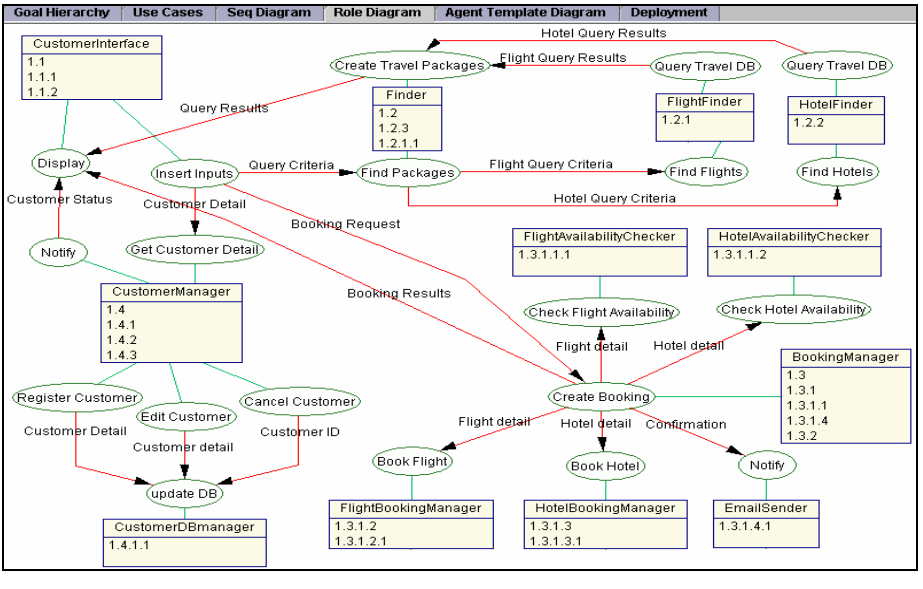

Fig. 2. The role diagram generated using the AgentTool

fully interactive software engineering tool. It also provides automated support for transforming analysis models into design artefacts. Systems designed using MaSE can be implemented using any agent-programming environment such as JADE or JACK. Figure 2 shows the role diagram developed for our case study using the agentTool.

4 Developing Multi-agent Systems with JADE

One goal of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents. It provides the following mandatory components for agent's management: [5][9]

- AMS (Agent Management System), which besides providing white page services as specified by FIPA, it also plays the role of authority in the platform.
- DF (Directory Facilitator) provides yellow pages services to other agents.
- ACC (Agent Communication Channel) which provides a Message Transport System (MTS) and is responsible for sending and receiving messages on an agent platform.

JADE has been applied to several applications successfully [3][11]. In the following subsections we discuss the various steps involved in developing a multi-agent system using JADE.

4.1 Creating Agents

Creating a JADE agent is as simple as defining a class that extends the jade.core.Agent class and overriding the default implementation of the methods that are automatically invoked by the platform during the agent lifecycle, including SetUp and TakeDown (). Consistent with the FIPA specifications, each agent instance is identified by an 'agent identifier'. In JADE an agent identifier is represented as an instance of the jade.core.AID class. The getAID () method of the Agent class allows retrieval of the local agent identifier.

4.2 Defining Agent Tasks

In JADE, behaviour represents a task that an agent can carry out and is implemented as an object of a class that extends *jade.core.behaviours.Behaviour*. Each such behaviour class must implement two abstract methods. The action () method defines the operations to be performed when the behaviour is in execution. The done () method returns a Boolean value to indicate whether or not a behaviour has completed and is to be removed from the pool of behaviours an agent is executing. To make an agent execute the tasks represented by a behaviour object, the behaviour must be added to the agent by means of the *add Behaviour ()* method of the *Agent* class.

4.3 Agent Discovery Process

The JADE platform provides a yellow pages service which allows any agent to dynamically discover other agents at a given point in time. A specialised agent called the

DF (Directory Facilitator) provides the yellow pages service in JADE. Using this service any agent can both register (publish) services and search for (discover) services.

4.4 Agent Communication

Agent communication is probably the most fundamental feature of JADE and is implemented accordance with the FIPA specifications. The JADE communication paradigm is based on asynchronous message passing. Each agent is equipped with an incoming message box and message polling can be blocking or non-blocking. A message in JADE is implemented as an object of the jade.lang.acl.ACLMessage object and then calling the send () method of the Agent class.

4.5 Defining Ontologies

Agents must share semantics if the communication is to be effective. Therefore, exchanged messages must have written in a particular language and must share the same ontology. An ontology in JADE is an instance of the jade.content.onto.Ontology class to which schemas have been added that define the types of predicates, agent actions and concepts relevant to the addressed domain. These schemas are defined as instances of the PredicateSchema, AgentAction Schema and ConceptSchema classes included in the jade.content.schema package.

5 Integrating JADE Agents with JESS

Rule-based systems have been successfully used for e-commerce applications such as order processing, user preferences, data mining and recommendations [8]. In our case study we have identified two types of knowledge that are necessary: travel-related and customer-related. Rule-based systems have also been used to assist in the negotiation process among agents [4]. Currently JADE alone does not endow agents with specific capabilities beyond those needed for communication and interaction. However,

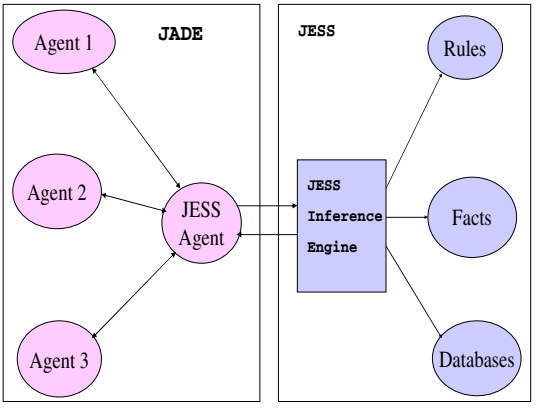

Fig. 3. The integration of JADE and JESS

the behaviour abstraction of the JADE agent model allows simple integration with JESS [15]. Figure 3 illustrates our strategy for the integration of JADE and JESS in order to enhance the JADE's capabilities.

The JESS shell provides the basic elements of an expert system including fact-list, knowledge base that contain all the rules and inference engine which controls overall execution of rules. JESS includes a special class called Rete, which implements the rule-based inference engine. To embed JESS in a JADE agent, we simply need to create a special agent called JESS agent, which incorporates jess.Rete object and manipulates it appropriately. Figure 4 illustrates how *Jess.Rete* class is used inside the JESS agent's behaviour.

```
class JessBehaviour extends CyclicBehaviour {
         private jess.Rete jess;
         JessBehaviour(Agent agent, String jessFile) {
               super(agent);
               jess = new jess.Rete();
               try {
                  FileReader fr = new FileReader(jessFile);
                  jess.Jesp j = new jess.Jesp(fr, jess);
                  try {
                      j.parse(false);
                  } catch (jess.JessException je) {
                      je.printStackTrace();
                  }
                  fr.close();
               } catch (IOException ioe) {
                  System.err.println("Error loading, Jess file");
               }
          }
…
…

  }
```

Fig. 4. A behaviour of the JESS agent includes *jess.Rete* object

The method *Rete.Run()* allows us to run the inference engine. This method will make the engine consecutively fire applicable rules, and will return only when there are no more rules to fire, that is, when the engine stops.

6 An Integrated Intelligent Agent Development Environment

We use the Eclipse platform as our programming environment, which is an open source Integrated Development Environment (IDE) that provides support for range of languages from C++, PHP to Java [20]. The Eclipse environment comes with a plug-in to integrate JADE within Eclipse. Figure 5 shows the JADE **Remote Monitoring Agent** which controls the life cycles of the agent platform and of all registered agents

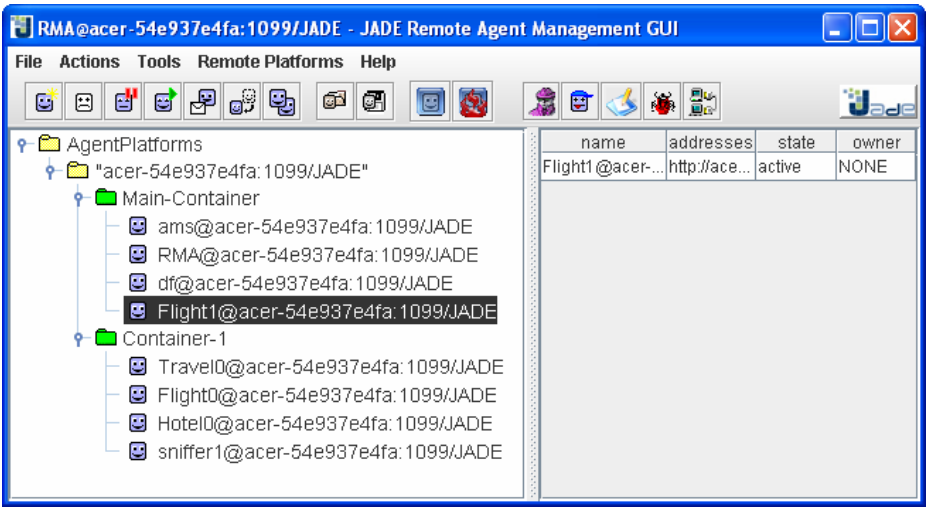

Fig. 5. JADE Remote Monitoring Agent

6.1 Running the PTA System

The PTA system can be started after creating at least one of the each agent (travel, flight, hotel and car). Travel agent is the main component of the system and the customer interacts only with this agent. Other three types of agents can be one or more depending on the web services and they have no direct interaction with the customer. For example, if we have two flight web services to connect, then two Flight agents need to be created and connected to the web services. Figure 6 shows the Travel Agent's user interface screens which is used for entering the user's travel requirements.

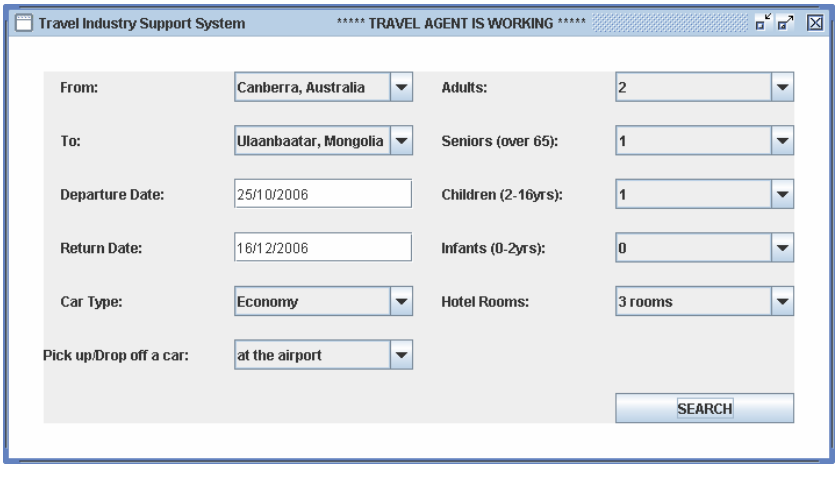

Fig. 6. Travel Agent's user interface

We have carried out some preliminary experiments with the current version of the system. The results so far are promising.

7 Conclusions and Future Work

This paper has shown the use of MaSE and agentTool in the design and development of a multi-agent system which requires communication and collaboration. By analysing the system as a set of roles and tasks, a system designer is naturally led to the definition of autonomous agents that co-ordinate their actions to solve the overall system goals.

The prototype PTA system, which integrates JADE and Jess seamlessly, has been developed and implemented. The JADE development tool provides suitable mechanisms for developing multi-agent systems, making easier the programmers' tasks when implementing agents, tasks, ontologies and communication between them. For intelligence aspects, we have exploited the facilities of JADE relating JESS integration. JADE agents embedded with JESS engines have qualities over and above those found in simple JADE agents. More importantly such hybrid agents have intelligent reasoning abilities. The PTA System has met the aims and goals expected and has been tested for travel booking in terms of hotel, flight and car requirements.

We aim to improve the ability of the agents in regard to conflict resolution using negotiation skills by incorporating more advanced negotiation protocols and negotiation rules. We will report our progress in our future publications.

Acknowledgements

The prototypes described in this paper have been mainly developed by Majigsuren Enkhsaikhan.

References

1. AgentTool, Support tool for MaSE methodology. Multiagent and Cooperative Robotics Laboratory (2006), `http://macr.cis.ksu.edu/projects/agentTool/agentool.htm`
2. Balachandran, M.B., Enkhsaikhan, M.: Development of a Multi-agent system for Travel Industry Support (CIMCA 2006 and IAWTIC 2006), Sydney, Australia (2006)
3. Bellifemine, F., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. John Wiley & Sons, UK (2007)
4. Benyoucef, M., Alj, H., Levy, K., Keller, R.K.: A Rule-Driven Approach for Defining the Behaviour of Negotiating Software Agents. In: Plaice, J., Kropf, P.G., Schulthess, P., Slonim, J. (eds.) DCW 2002. LNCS, vol. 2468, pp. 165–181. Springer, Heidelberg (2002)
5. Cuesta-Morales, P., Gomez-Rodriguez, A., Rodriguez-Martinez, F.J.: Developing a Multi-Agent System using MaSE and JADE. Upgrade 5(4), 27–31 (2004)
6. DeLoach, S.: Multiagent Systems Engineering: a Methodology and Language for Designing Agent Systems. In: Proceedings of Agent Oriented Information Systems 1999, pp. 45–57 (1999)

7. Far BH Sample Project: Travel Agency System (TAS) (2004) (Retrieved on February 2, 2006), http://www.enel.ucalgary.ca/People/far/
8. Fasli, M.: Agent Technology for e-Commerce. John Wiley and Sons, UK (2007)
9. FIPA (2006) The Foundation for Intelligent Physical Agents, http://www.fipa.org/
10. Friedman-Hill, E.: Jess in Action: Java Rule-Based Systems. Manning Publications Co., New York (2003)
11. Ganzha, M., Paprzycki, M., Pirvanescu, A., Badica, C., Abraham, A.: JADE based Multi-agent E-commerce Environment: Initial Implementation (Retrieved on July 30, 2006) (2001), http://www.ganzha.euh-e.edu.pl/agents/index.html
12. Gordon, M., Paprzycki, M.: Designing Agent Based Travel Support System (Retrieved on June 2, 2005), http://agentlab.swps.edu.pl/ISPDC_2005.pdf
13. Iglesias, C., Garijo, M., Gonzalez, J.: A Survey of Agent-Oriented Methodologies. In: Muller, J.P., Singh, M.P., Rao, A.S. (eds.) ATAL 1998. LNCS (LNAI), vol. 1555, pp. 317–330. Springer, Heidelberg (1999)
14. JADE, Java Agent Development Environment (2008), http://jade.tilab.com
15. Lopes, H.: Integrating JADE and Jess (Retrieved on March 13, 2007), http://jade.tilab.com/doc/tutorials/jade-jess/jade-jess.html
16. Padgham, L., Winikoff, M.: Developing intelligent Agent Systems: A Practical Guide. John Wiley & Sons, Chichester (2004)
17. Sycara, K.P.: Multiagent Systems. AI Magazine 19(2), 79–92 (1998)
18. Wooldridge, M.: An Introduction to Multiagent Systems. Wiley Ed., Chichester (2002)
19. Wooldridge M, Jennings NR and Kinny D.: The Gaia Methodology for Agent-Oriented Analysis and Design (Retrieved on 15 February 2006) (2000), http://www.ecs.soton.ac.uk/~nrj/download-files/jaamas2000.pdf
20. The Eclipse Platform, http://www.eclipse.org/