The Development of Multi-Agent System using Finite State Machine

Yazed Al-Saawy, Ajlan Al-Ajlan, Khalid Aldrawiesh and Abdullah Bajahzer
Software Technology Research Laboratory (STRL)
De Montfort University
The Gateway, Leicester, LE1 9BH, UK.
yazed,ajlan,khalid,bajahzer@dmu.ac.uk

Abstract

This paper sets out to develop and create a search agent that can retrieve requested information from two databases; the user will be able to search for a book by inputting variables such as title, author, ISBN and more significantly for this study the price. The system will identify the books requested, where they can be found and will show the cheapest available. SAS will work from the Application software and will link to the two databases, the design will allow more databases to be incorporated in the future. Each of the two databases has the same names of the titles however each database may contain a different price. The agent will get the required information quickly, helping the user to find the cheapest and possibly nearest. The agent will connect to the database using the Agent Interface; there are two agents in the system, known as MAS. These agents collect the required data and send it to the Agent Handler.

Keywords: Finite State Machine, Multi-Agent System JFLAP, Java, and UML.

1. Introduction

The main contribution of this paper is to create Search Agent System (SAS), which helps the user to perform a search for a book title quickly; the information about the books will be stored on two different databases. The user can search for the book by name; author or ISBN number and the system will present all options as well as identify the cheapest price for the user. The system uses a multi-agent configuration that works together with a special platform, the JADE program.

In order to model and demonstrate the software the Finite State Machine (FSM) is used. It will demonstrate how the software will run with multi agents to search for the requested book and give the cheapest price. The user can start the search by selecting either the book name, ISBN, author or price and also chooses the preferred host from which the agent begins the search. Also, the model will be created using the JFLAP software.

After the modelling will be the analysis using UML which analyses the requirements both functional and non-functional using the class diagram, class and sequence diagrams will demonstrate the best software design which helps to create and develop the software system. Finally, will be the design of the software application using JADE and JAVA which will create the type of the Agent Handler, Agent Interface and Book Agent System which will search according to the users search criteria. The system will have two databases. However, the design will be expandable, allowing more databases to be added.

This paper will examine how to use agent and Multi-Agent System (MAS) in a search system. MAS have provoked more interest in research because of the advantages found within such systems, for example they are able to deal with problems that are too large for a single-agent system are faster and more reliable. Additionally, they are able to cope with uncertain knowledge and data. Key areas of the research into the ability of MAS to solve problems focus on communication, coordination and negotiation. Examining how multi agents function in a virtual system. The virtual system exposes how behaviour of the components affects the system as a whole.

This paper is organized as follows: Section 2 provides MAS. FSM is described in Section 3. Section 4 presents requirements and Analysis for SAS. Section 5 focuses on the implementation of SAS. Testing & evaluation are described in Section 6. Finally, conclusion and future work are presented in Section 7.

2. Multi-Agent System

An agent is a software program that communicates with other software programs, interacting and responding to behaviour, acting and linking with available resources on demand. Each agent has size, capability and speed; all these are attributes of each agent. The environment of each agent has to be considered in MAS in order to coordinate the activities between a numbers of agents. An example of this could be robots. the agents, (A) are the robots, the geometrical space (E) is where the robots move, and O is obviously made up of agents, but also of physical objects placed here and there which the robots have to avoid, pick up or manipulate. The operations, Op, are the actions that the robots can take in moving themselves, moving the other objects or communicating, and R is the assembly of relations that link certain agents to others [1, 2, 3].

The domains of application for multi-agent systems are particularly numerous. We shall refer only to the main trends, as an attempt to address a wider list could only end in stagnation; the research of domains is evolving all the time. Multi-agent systems can be divided into four main categories: multi-agent simulation, building artificial worlds, collective robotics and program design as in Figure 1.

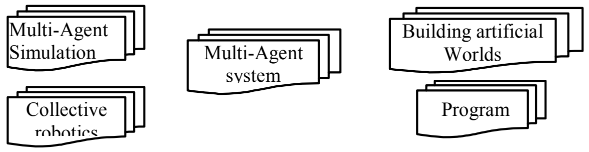


Figure 1: Application domains in MAS

In computer science, there is a branch known as simulation, which consists of analysing the properties of theoretical models. Simulation is currently an active area of computer science which is concerned with the analysis of the properties of theoretical models. The models are constructed of the mathematical relationships between the variables which represent the physical values which can then be measured. The commonly used models are transition matrices and differential equations [4, 2].

Brahms is a set of software tools used to develop and simulate MAS of human and machine behaviour. In addition, it is rule-based and multi-agent activity based on programming language. The Brahms language shows how the situated activities of agents are represented in a geographical relationship to the world. Situated activities are actions that occur in a specific situation; therefore their function is not only constrained by the reasoning abilities of the agent, but also by the agent's perception of the world [5].

Java Agent Development framework (JADE) agent can be split over several hosts with one of them acting as the front end for inter-platform IIOP communication. JADE is made up of one or more Agent. Each Jade lives in a separate Java Virtual Machine and communicates using java RMI. The Figure 2 shows the architecture of this agent [6, 7].

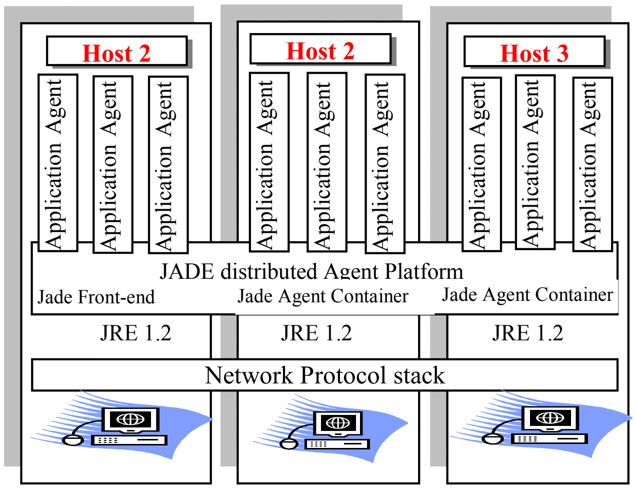


Figure 2. Software Architecture of JADE Agent [17]

The idea behind JADE communication architecture is that it attempts to offer efficient and flexible messaging; it openly chooses the best transport and leverages distributed object technology which is found within the Java runtime environment. JADE recognises the difference between inter-platform messaging where the sender and the receiver agents operate within different platforms and intra-platform messaging where the two agents are on the same platform. Inter-platform messaging has to comply with FIPA requirements; however intra-platform message delivery is solely an issue for JADE, so a more convenient transport can be utilised. JADE uses Java RMI in intra-platform communication [6, 8]. Java and JADE have attributes that support multi-agent applications: 1) Autonomy 2) Intelligence 3 Mobility)

3. Finite State Machine

This section will examine modelling of the software using FSM and how the Search Agent will search for name, ISBN, price and author or maybe just two of these elements or search all elements together depending on what the user wants [9]. FSM comprises of a data structure used to show actions with a sequence of events. In both Java and JADE applications, FSM is able to activate and deactivate certain behaviours in time. This section will define how to describe FSM using JFLAP as in Figure 3, and then the coding using JADE. JFLAP is instructional software used to experiment with grammars automata. However it also allows experimentation with applications and proofs. The main feature of JFLAP is that it can experiment with grammars and theoretical machines. It allows the building and running of user-defined input on pushdown automata, finite automata, regular grammars [10].

Here, we will examine the design of FSM, which has eight states; the first is to start the search using the name, ISBN, price and author. Afterwards the search will compare and find the cheapest; this is one of the final states. The other final state is the normal state which gives all possible prices of the book. This covers all possibilities of the search as in Figure 3.

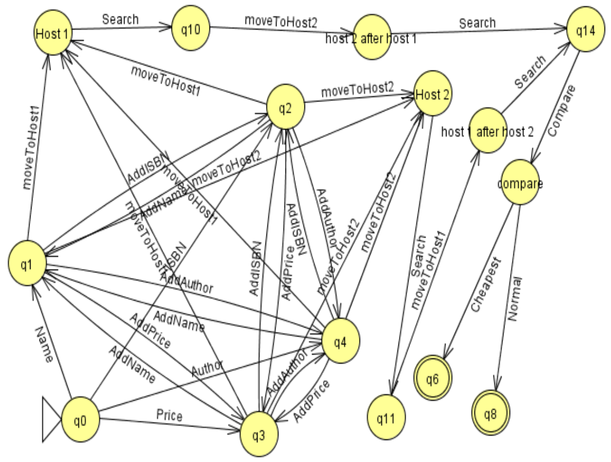


Figure 3: Search Agent Model of Finite State Machine

4. Requirements and Analysis

This section will examine the requirements and analysis before creating the application software for the Search Agent. These requirements have to be identified, giving clear guidelines as to what is required in the finished, usable software that links multi-agents. For our framework, the three stages shown in Figure 4 will be followed. The properties of MAS will be incorporated and a suggestion of required architecture which reflect the properties will be made. The first phase in the approach is known as the 'Problem Analysis.' This phase attempts to gain an understanding of how the system works in the abstract, this will be a starting point for the development process of the architecture. After gaining an understanding we then move to the next phase 'Agent Modelling", here agents will be identified to satisfy the goals and the goals relationships. Finally the 'MAS architecting' phase, here the focus in on the architecture of the agents and the setting up of a federation of agents that are able to cooperate with each other, at the same time, to a large extent, maintaining autonomy [11].

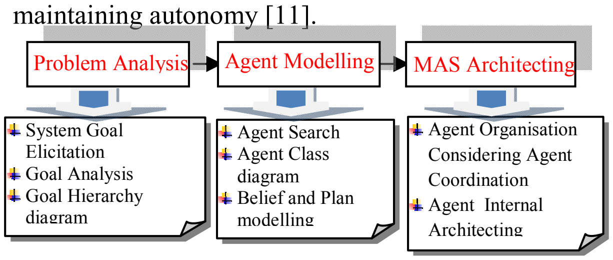
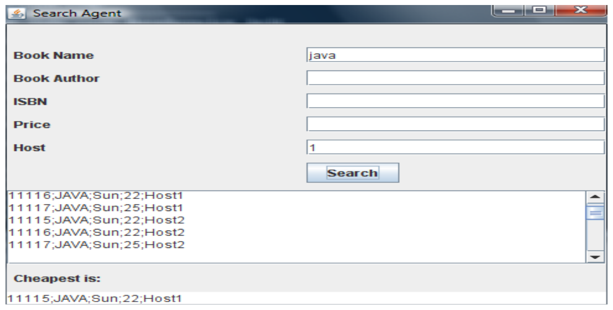


Figure 4: Architecture development process for MAS

As in Figure 4, the problem analysis is important for analyzing the user's requirements and to set up the system boundary. End users of a system frequently find it difficult to express abstract requirements for the system, a goal oriented approach to the requirements analysis is an effective way of improving the elicitation process. The agents in the system need to be identified with the relationships between them by agent modelling to be modelled, according to the relationship between goals and agents. Therefore, we have developed a set of heuristic guidelines to create and search agents to identified major goal, create a corresponding agent when necessary. A sequence diagram shows the interactions between objects in a time sequence. More specifically, it shows the objects and their interactions with the sequence of messages exchanged. Sequence diagrams are different from collaboration diagrams, as they show time sequences but do not show the relationships between objects. A sequence diagram can be in generic form, showing all possible scenarios or in instance form, focusing on one specific scenario [11].

5. Implementation

This section examines the implementation of the SAS whilst running, and the functions that a tool built by the study can perform. The system needs to work on any PC that have java environment; therefore there are some requirements to insulate the software, as follows, JADE 3.4 and JDK. Before running the SAS of Book Agent has to be put into the JADE folder in C drive so it is easy to read that class. After that, inside the folder Book Agent there is a command, the command is opened and writes Javac the BookAgentSys.java to compile the class in folder. After compilation JADE will be run using the JADE tag which is java jade.Boot t1: RunAgent. t1 is the name of the agent in the pc. After running the system, it will show the user interface so the user can start the search.



Book Name: java
Book Author:
ISBN:
Price:
Host: 1

```
11116;JAVA;Sun;22;Host1
11117;JAVA;Sun;25;Host1
11115;JAVA;Sun;22;Host2
11116;JAVA;Sun;22;Host2
11117;JAVA;Sun;25;Host2
```

Cheapest is:

```
11115;JAVA;Sun;22;Host1
```

Figure 5: search By Book Name

Figure 5 shows the Book Name as 'java'. The search will begin in the Host 1, set as the default Host, however the search can change looking at Host 2 if the user requires. Any type of search from Book Name, Author, Price, and ISBN can be performed simultaneously. Figure 5 explains all the books with 'java' in the title in a list that shows the host, price, ISBN and author. The next label shows the user which is the cheapest book and where to find the book, this saves time looking for it from the entire list.

6. Discussion

This paper used two types of testing, which are Black Box and White Box testing tools. The first testing is the functional testing, that examines whether the system behaved according to the set-out objectives of the search system, for example whether or not the system was successful in identifying the cheapest book as in Table 1.

Table 1. Black Test

Book Name	Author	Host	Price	Result	Cheapest
JAVA	SUN	1	25	Found list	The cheapest is 22
JAVA	SUN	1	27	Found list	
JAVA	SUN	1	22	Found list	

The second testing is the usability testing that test examined if the system met the requirements set out for 'user friendliness', it examines specifically the user interface and whether or not it is suited to the type of user i.e. a person searching for a book name and being given the result including the price and location. Finally, user acceptance testing was carried out by giving the system to a user to see if the system met their expectations and behaved according to what was expected. Overall, the system was deemed a success in that in functioned according to the desired objectives. The functionality, usability testing concluded that the user was able to input the search variables and was given the desired result i.e. the book they were looking for, where it was located and identification of the cheapest book on offer. The mobility agent performed very well and it is able to search in two hosts at the same time.

The system was able to search more than one search variable simultaneously i.e. book name and host and still identify the cheapest book. A slight criticism, from the tested usability perspective (Black Box testing), is the terminology used in the interface; for example the term 'host' is a technical term used by the developer and the term 'location' would be a more user-friendly, understandable term. FSM helps to modelling in useful for programmer to understand the requirement. However, FSM give the mobility of each state how to be worked and create each of state as class.

7. Conclusion and Future Work

The overall aim of the paper was to create and develop a Search Agent which uses DAI using MAS. The system was to enable users to search for book titles using various search criteria giving the result according to book found, price and location. It was evaluated using user-friendly functionality testing and performed well against desired objectives. The paper demonstrated how multi-agents functioned well in making decisions.

For future development, the design of the system will be able to handle more than two hosts, a function that is necessary for overall implementation in a real-life system. This system can be further improved by incorporation of a facility to show the user the location of the book stockiest that have the requested title. Further usability testing can be carried out in collaboration with a wider spectrum of typical users, as many users may highlight a multitude of desires and expectations.

8. Acknowledgment

The authors wish to acknowledge contributions to Hussein Zedan, Hongji Yang and Antonio Cau.

9. References

[1] J. Ferber, Multi agent systems, an introduction to distributed artificial intelligence, Addison Wesley/ Pearson, London, 1999.

[2] A. Roberto, Multi agent, available from: http://www.acm.org/crossroads/xrds5-4/multiagent.html

[3] H. Fran Available from: http://www.it-director.com/article.php?articleid=11774, 2004.

[4] M. Dastani, and J. Gomez-sanz, Programming multi-agent systems. Knowl. Eng. Rev., Cambridge University Press, vol. 20(2), pp. 151-164, 2005.

[5] Agent Solution, retrieve on 2/1/2009 from: http://www.agentisolutions.com/brahms.htm.

[6] F Bellifemine, et al., JADE: a FIPA2000 compliant agent development environment, In Proc. of the Fifth on Autonomous Agents, ACM, New York, pp: 216-217, 2001.

[7] F. Bellifemine, et al., JADE, White Paper, V.3(3), 2003.

[8] Bellifemine, et al., Developing Multi-agent Systems with JADE. In Proc. of the 7th international Workshop on intelligent Agents,. Springer, London, pp: 89-103,2001

[9] B. Lee and E. Lee, Interaction of Finite State Machines and Concurrency Models, in Proc. of Thirty Second Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, California, 1998.

[10] S. Rodger, et al., Increasing interaction and support in the formal languages and automata theory course, SIGCSE Bull, vol.39(3) ACM New York, NY, USA, pp:58-62, 2007

[11] S. Parka and V. Sugumaranb, Designing multi-agent systems: a framework and application, Expert Systems with Applications, Seoul, Korea vol. 28(2), pp: 259–271, 2005.