JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No. 8, Novmeber - December 2008

Extension of Object-Oriented Software Testing Techniques to Agent Oriented Software Testing

Praveen Ranjan Srivastava, Karthik Anand V, **Mayuri Rastogi**, **Vikrant Yadav**, **G Raghurama**
Birla Institue of Technology and Science, Pilani, India

Abstract

In recent years, agent-based systems have received considerable attention in both academics and industry. The agent-oriented paradigm can be considered a natural extension to the object-oriented (OO) paradigm. Agents differ from objects in many issues which require special modeling elements but have some similarities. Although there is a well-defined OO testing technique, agent-oriented development has neither a standard development process nor a standard testing technique. In this paper, we propose extensions of OO testing techniques to test agent oriented systems. For illustration purpose a multi agent air ticket booking system is implemented using JADE 3.5 and tested using our proposed method.

1 INTRODUCTION

As the technology advancing, the more we are driven towards abstraction and generalization. The software systems now âdays need to be adaptive, autonomous and dynamic to serve the needs of varied user community. These systems are evolved very fast in past few decades. Software agents are an abstraction to describe computer programs that act for a user or another program. They can be dedicated to a particular task or, if endowed with enough intelligence and can act on behalf of a client. The agent oriented methodologies provide us a platform for making our system abstract, generalize, dynamic and autonomous. However, many methodologies like MASE, Prometheus, Tropos do exist for the agent oriented framework but on contrary to it the testing techniques for the methodologies are very few [Dam]. This paper is indented to extend the object oriented software testing techniques to agent oriented systems.

Agents autonomously work in dynamic and uncertain environments. Each agent senses the environment and acts accordingly. Since the environment in which the agent reside change dynamically, the construction of the agents should be such that it is able to

Praveen Ranjan Srivastava, Karthik Anand V, Mayuri Rastogi, Vikrant Yadav, G Raghurama: "Extension of Object-Oriented Software Testing Techniques to Agent Oriented Software Testing", in *Journal of Object Technology*, vol. 7, no. 8, November-December 2008, pp. 155-163. http://www.jot.fm/issues/issue_2008_11/article5/

accomplish the desired tasks by collaborating with other agents. There are very few literature that describe software testing for agents. The simulation approach suggested by Himmelspach et al. can be utilized to test the behavior of the agent system interacting with environment [Himmelspach]. The virtual environment is used in contrast to the real time environment because it will reduce the cost and effort involved. Since the environment changes dynamically, we need to monitor the agents at each point of time. JAMES (Java based agent modeling environment for simulation), is the simulation system that is used for creating the virtual environments and generating the dynamic test cases. Nguyen et al. [Nguyen] suggested the goal testing approach based on tropos methodology. Goals are classified as mainley hard goal and soft goal. Goals are executed by plans and a goal can have sub goal. Test cases are derived from goals.

Agents have higher level of abstraction as compared to objects. Agents encapsulate mental state and behavior also. On the other hand, objects encapsulate data and algorithm. Agents can change their behavior according to the environment while objects can perform only trained tasks. Each agent has its own thread of control whereas each object need not have its own thread of control. Even though the agents and objects have the above mentioned differences, the modeling techniques used for analysis and design of object oriented techniques are being extended to support agent oriented software development [Yim]. Hoongsoon Yim et al. has extended UML to support development of agent systems [Zied].

An example of air ticket booking agent system is described in section two. This example is used as a reference and the testing techniques are applied on that. Section three and four discusses about the random testing and behavioral testing techniques respectively. Section five focusses on the partition technique at the agent level. The paper concludes with a summary of the work and some suggestions for the potential future work.

2 AIR TICKET BOOKING AGENT SYSTEM

The paper presents the implementation of an agent based air ticket booking system using JADE 3.5 (Java Agent Development Framework). JADE simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases [Jade]. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required [Dam]. The air ticket booking system is a multi agent system comprising of buyer and a seller agent. The seller agent registers with the service directory. The buyer agent will read the flight details through the command line and search the corresponding seller agents which are satisfying the flight requirements. The directory service maintains the list of all available sellers. Furthermore, the buyer receives the proposals from all the seller agents and will select the seller with best price

offer. Moreover it confirms the ticket with the seller. The seller agent will respond to the queries made by the buyer agent and updates its state on successful transaction. The buyer agent re-executes its plan if there are no seller agents. The buyer agent is blocked until there is a seller of tickets. The sequence diagram for the buyer and seller agent is shown in the Figure 1.

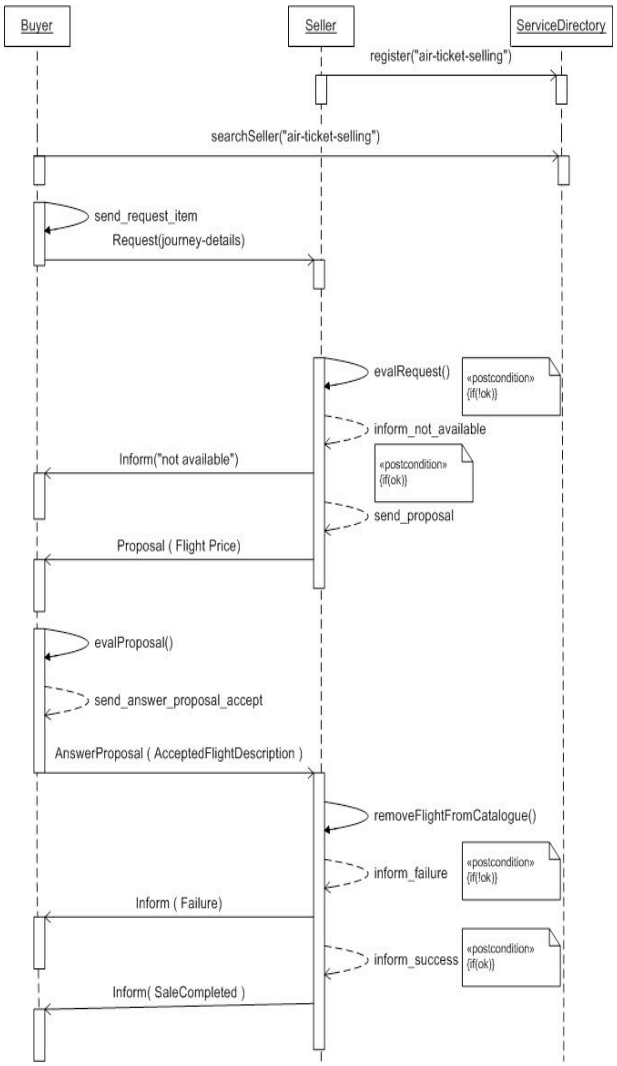


Figure 1- Sequence diagram for air ticket booking system showing the interaction between buyer, seller and search directory agent

In object oriented paradigm, the communication between the objects is via method calls i.e. an object is actually sends a message to other object. An agent does not have any publicly visible method which the other agents can call. The communication is achieved by agent communication language. The communication between the agents in JADE is done by sending ACLMessage like CFP (call for purpose), propose, failure, inform etc.

The testing techniques like random testing, behavioral testing and partition testing are black box testing techniques that can be utilized to test the agents individually. Stubs have to be written for implementing the test cases for those techniques.

3 RANDOM TESTING FOR AGENTS

The random testing proposed in the paper for agent oriented systems is analogous to that of the object oriented framework. One agent is considered at a time. The list of all possible messages which the agent can receive is formulated. The normal sequence of the messages which can be sent to the agent is formed. The agent is tested by sending random messages in that sequence and the response of the agent corresponding to the message sequence is checked.

In the flight air ticket booking example, the sequence of messages that the buyer agent expects is:

(Propose – Inform | Failure) | Refuse

Test cases for different message sequences are generated randomly. For example:

- Test Case 1: Propose – Inform
- Test Case 2: Propose – Failure
- Test Case 3: Refuse

The agent can also be tested under test case which sends other kinds of messages which are not known to the agent. The following is the pseudo code for testing Propose – Inform (Test Case 1) message sequence. Using this method the following test are generated.

Test Case 1:	
Agent under test	Buyer Agent
Agent goal tested	Buy tickets with the cheapest seller
Collaboration agents involved	Seller agent
Testing technique	Random Testing
Scenario	Testing Buyer by sending PROPOSE and INFORM messages
Expected result	Buyer agent will reject the proposal.
Observed result	When selecting the best seller, the buyer did not consider the price to be greater than zero. The buyer selected the seller even though it offered a price less than zero.
Test Case result	Failed

Test Case 2:	
Agent under test	Buyer Agent
Goal Tested	Buy tickets with the cheapest seller
Collaboration agents involved	Seller agent
Testing technique	Random Testing
Scenario	Testing buyer agent by sending REJECT messages for accept proposals
Expected result	Buyer agent will not consider the seller agent as the seller tickets for the required route.
Observed result	Buyer agent did not consider the seller agent as the seller tickets for the required route.
Test Case result	Passed

Test Case 3:	
Agent under test	Buyer Agent
Goal Tested	Buy tickets with the cheapest seller
Collaboration agents involved	Seller agent
Testing technique	Random Testing
Scenario	Testing buyer agent by sending messages not known to the buyer after receiving a ACCEPT_PROPOSAL message.
Expected result	Buyer will ignore the message and continue with its activity.
Observed result	Buyer ignored the message and continued with its activity.
Test Case result	Passed

4 BEHAVIOR BASED TESTING FOR AGENTS

Agents can have any number of behaviors like one-shot, cyclic, parallel, sequential, FSM behaviors. Each behavior is seen as a black box. Apart from this the programmers can write their own behaviors. Each behavior can send or receive any number of messages. Test cases must be designed in such a way so as to test the behaviors of the agent by sending messages. In the flight air ticket booking example, the seller agent has two cyclic behaviors.

The figure 2 shows the types of messages passed between the buyer and seller agents.

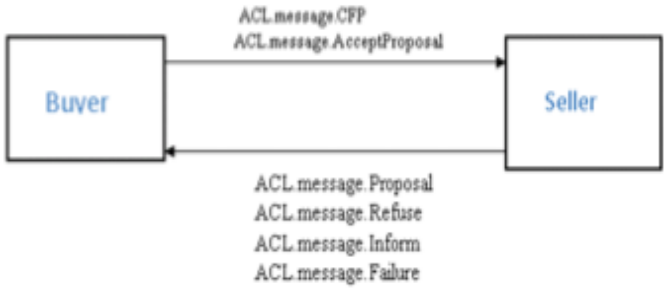


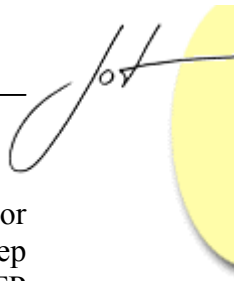
Figure 2 -Communication messages between buyer and seller agents

Example 1: One cyclic behavior is for offering the flight information request and the other is for confirming the order request. The confirm order cyclic behavior responds to ACCEPT_PROPOSAL (order request message with order details like flight info, number of seats required) message with either INFORM (confirmation for booking) or REFUSE (booking failure) message. The following test cases can be designed.

Test Case: Make a stub agent which acts like a buyer and sends ACCEPT_PROPOSAL messages to the seller agent. The messages should be sent at regular intervals as the behavior under test is a cyclic behavior.

Test Case 1:	
Agent under test	Seller Agent
Goal Tested	Make proposals
Collaboration agents involved	Buyer agent
Testing technique	Behavior Testing
Behavior under test	Cyclic behavior of the seller agent to make proposals to the buyer agents
Scenario	Testing Seller agent by sending a CFP message periodically.
Expected result	Seller agent will de-register from directory facilitator and quit itself if it had no more tickets to sell.
Observed result	Seller agent sent a refuse message. It was registered with the directory facilitator even though it had no ticket to sell.
Test Case result	Failed

Example 2: Testing one shot behavior is simple. The message of appropriate kind should be sent only one time. User defined behaviors are the difficult to test. It requires the



understanding of the sequence of messages expected by the behavior (if any). For example the buyer agent has a user defined behavior called 'RequestInformer'. It is a step behavior. At each step, messages are either sent or received. At step one it sends the CFP message. In step two it waits for PROPOSE message and on selecting the best seller in step three it sends ACCEPT_PROPOSAL message and in step four waits for INFORM or FAILURE message.

The difference between behavior and random testing is that, in behavior testing the messages are sent behavior wise to test them individually. In random testing behaviors are not considered and the agent is considered as a whole.

5 PARTITION TESTING AT AGENT LEVEL

Normally in functional and object oriented approaches, the partition testing is done to reduce the number of test cases by selecting partition categories based on input, output , state, attributes , function type. Agents do have input, output, state and attributes and so partition can be based on these criteria. But agents do not have externally visible functions. Each agent has its own thread of control and behaviors. Partitions can be made on the various types of behaviors like one-shot, cyclic, parallel, sequential, FSM behaviors.

For example all the one shot behaviors can be tested first and then all the cyclic behaviors can be tested. Messages can have parameters. Test cases can be designed based on the partition on the input values of the parameters. In the flight air ticket booking example, buyer agent the seller expects an ACCEPT_PROPOSAL message in the 'PurchaseOrdersServer' cyclic behavior. It requires the parameters like flight number, flight date, starting place, destination and number of tickets required. Equivalence partitioning can be made on these required parameters. For example a stub can be created that will act like a buyer and sends a ACCEPT_PROPOSAL message which has the number of required tickets as -1 or flight date as 31st February. The following test case is generated using the partition testing.

Test Case 1:	
Agent under test	Seller Agent
Goal Tested	Accept Offers
Collaboration agents involved	Buyer agent
Testing technique	Partition Testing
Partition base	Cyclic behaviors of the seller agent. The valid range of values for required ticket count is > 0. The invalid range of values for required ticket count is < 0
Scenario	Testing Seller agent by sending an ACCEPT_PROPOSAL message with the required ticket count as a negative value. Tickets are available with the seller.
Expected result	Seller sends a failure message as the required ticket count is a negative value.
Observed result	Seller sent a booking successful 'INFORM' message and increased the ticket count. Error: freeSeats-=requestrequiredticketcount caused the number of free seats to be increased.
Test Case result	Failed

6 CONCLUSION

In this paper, we have proposed three extensions of OO testing techniques to test agent oriented systems. These techniques are applied to a simple dual agent air ticket booking system. Random testing technique is used to generate test cases which send random messages to the agent under test. Behavior testing technique is utilized to generate the test cases for analyzing the type of messages which can be sent to each behavior of the agents. Partition technique is used to limit the number of test cases by choosing partitioning categories such as input, output, state, attributes, behaviors, message types. Furthermore this technology can be applied to other simple agent based information systems. But these techniques have to be studied in detail when applying to complex multi agent systems.

REFERENCES

[Nguyen] C D. Nguyen, A. Perini, and P. Tonella, "A goal oriented software testing methodology"

[Yim] Yim H, Cho K, Kim J, and Park S,: "Architecture-Centric Object-Oriented Design Method for Multi-Agent Systems"

[Zied] Zeid A: "A UML Extension for Agent-Oriented Analysis an Design, Department of Computer Science, the American University in Cairo

[Dam] Dam K H, Winikoff M, "Comparing Agent-Oriented Methodologies"

[Himmelspach] Jan Himmelspach, "Simulation for testing software agents – An Exploration based on JAMES"

[Jade] http://jade.tilab.com/

About the authors

Praveen Ranjan Srivastava is a working as a lecturer in computer science and information systems group at Birla Institute of Techonology and Science(BITS),Pilani India.He is currently doing research in the area of Software Testing.He has a several publications in the area of software testing. His research area is software testing, Quality assurance ,agent oriented modeling, Software architecture framework and data mining.He is reachable at praveenrsrivastava@gmail.com

Karthik Anand V is a Teaching Assistant in computer science and information systems group at Birla Institute of Techonology and Science. He is presently doing M.E. in Software Systems at BITS, Pilani, India. Contact him at kathikbits@gmail.com.

Mayuri Rastogi is a Teaching Assistant in computer science and information systems department at Birla Institute of Techonology and Science. She is presently doing M.E. in Software Systems at BITS, Pilani, India. Contact her at mayuri.b5bits@gmail.com.

Vikrant Yadav is doing M.E. in Software Systems at BITS, Pilani, India. Contact him at mailmevicky@gmail.com.

G Raghurama is working in Electrical and Electronic Group at Birla Institute of Techonology and Science (BITS),Pilani, India.