Practical 5: Using ticker agents to synchronise agent behaviours

By the end of this practical you will be able to:

1. Use ticker agents to synchronise activities in an asynchronous multi-agent system
2. Produce a multi-agent system that runs for a number of simulated days without user input. Use appropriate agent communication protocols in JADE (the Contract Net protocol)
3. Implement and evaluate simple agent control strategies experimentally

Exercise 1: Become familiar with ticker agents to synchronise agent activities

Read Unit 2 of the module practical textbook. Pay particular attention to the use of ticker agents. Study carefully the example in Section 2.2. Implement it in Eclipse and check that you understand how it works.

You should then read Section 2.3, and download the file practical05.zip from Moodle, under "Practical 5 supporting material". You should import this as a project into your Eclipse workspace (choose File- >Import->Existing projects into workspace, and then choose archive file). You may need to add your copy of JADE.jar to the buildpath. The supplied project is a modified version of the code presented in Section 2.3, where the buyer agent now stores every offer that it receives that day in a HashMap, and then prints out all of those offers. Study the supplied code and check that you understand what each agent and each behaviour is doing. Ask your tutor if you are unsure about anything.

Run the application and check that you understand the console output. Note the use of the doWait(5000) method call in BuyerSellerTicker. This pauses the ticker agent at the start, to allow time for all of the other agents in the simulation to start. Increase the wait time from 5 to 30 seconds. Then run the application and start the JADE sniffer agent (Tools->Start sniffer) from the JADE GUI. You should sniff the Buyer, Sellers, Ticker, and Directory Facilitator agents by right clicking on them and choosing sniff agent. A wait of 30 seconds will allow you to do this before they start exchanging messages. Look at the sequence of messages sent. You can double click on a message to view its contents. Using the JADE sniffer to watch messages is a very useful aid to debugging.

Change the number of seller agents from 1 to 5 in the main method and check the output.

Exercise 2: Completing the sale

Add a behaviour to the buyer agent which executes after CollateOffers (i.e. you should add it to the SequentialBehaviour between CollateOffers and EndDay). This should reply to each offer received. If you wish to accept the offer and buy that book from that seller you should send an ACLMessage.ACCEPT_PROPOSAL to the seller, otherwise you should send an ACLMessage.REJECT_PROPOSAL. Use the book title as the conversation ID, so that the seller knows which book you are referring to. At the moment you may simply randomly decide whether to accept an offer or not (and you should only accept one offer for each book).

Modify the seller agent so that it listens for these replies and sends an ACLMessage.INFORM to confirm that the book has been sold. The buyer should listen for the replies. When it receives the inform (purchase successful) message, it should remove that book from the list of books that it needs to buy. It should also add the cost of the book to its total expenditure (you will need to add an instance variable to the buyer agent to record the total amount of money spent). At the end of the simulation the buyer should print the total amount of money spend (hint: do this just before the call to myAgent.doDelete()). Finally, add a penalty of £30 for each book that has not been purchased at the end of 30 days.

Exercise 3: Experimenting with buyer agent control strategies

Run the system several times to get a feel for the performance of the strategy that randomly decides whether to buy a book. See how the performance changes when you change the range of prices that sellers can offer books for (this is in the BookGenerator behaviour of the seller agent). Can you think of a more effective strategy? Have a go at implementing one and comparing its performance to the random strategy.