

CM30174/CM50206: Intelligent Agents: Assessed CourseWork 2: Implementing a Supply Chain

Marina De Vos and Julian Padget
{mdv, jap}@cs.bath.ac.uk

October 9, 2009

1 Introduction

The goal of this course work is to design and implement the agents and the agent communication language for a supply chain in which the (recursive) contract net protocol is used for reaching agreements. The work is to be carried out individually.

2 Objectives

At the end of this course work you will be able to

- develop a multi-agent system model of a supply chain
- develop an ontology
- use a specified protocol for agent interaction
- build on existing code

3 Project

The following subsections provide the project description, functional and non-functional requirements, the guidelines for installing the necessary code to start your project and the requirements for the report to be submitted for assessment.

3.1 Background Information

The situation is a trading scenario where you must develop client and supplier agents so that a client may purchase a specified quantity of:

- Nuts
- Bolts
- Washers
- Widgets
- Doobries

from a *single* supplier.

Your agent system will consist of a number of supplier agents, and a single client agent implemented as follows. For each run of the scenario:

- Each supplier agent is allocated a supply of a subset of the goods at a particular cost (see the oracle section for information on determining allocation) and a fixed cost for making a whole shipment to the client (i.e. shipment costs are not dependent on quantity).
- The client agent is allocated an order which must be fulfilled (see the oracle for information on allocation).

The client agent and supplier agents should implement the Contract Net Protocol, with the client playing the role of initiator and the suppliers playing the role of responders as follows:

- The client issues a request (call for proposals or CFP) to all of the suppliers indicating their desired bundle.
- The suppliers wait for a CFP and then generate a response to the client, which may be either:
 - a proposal: If the supplier can satisfy the request, the supplier should issue a proposal referring to or containing the original request, and a proposed price. The price should indicate the cost of the goods, and the cost of shipping the goods to the client. *or*
 - a rejection: indicating that the supplier is unable to supply the goods.
- The client waits for responses from the suppliers. If one or more proposal responses is received, then the client must choose a preferred proposal by cost and indicate to the winning supplier that they accept their proposal, and to the losing suppliers that they reject their proposals.
- The winning supplier then acknowledges this to the client.

The Oracle External information about the supply and demand for goods can be determined by a given supplier or client by interacting with the “Oracle” agent. Suppliers receive information about what they can supply by issuing a “SupplierInformationRequest” message to the oracle, who will then respond with a “SupplierInformationResponse” message indicating what the supplier can supply. The client receives information about the order for a given run by issuing a “ClientOrderRequest” message to the oracle, who responds with a “ClientOrderResponse” message indicating the required quantities of particular goods desired.

The Basic System In the basic system provided, you will receive a full oracle agent implementation, a simple client agent implementation which only queries the oracle for an order, and a simple supplier agent implementation which queries the oracle for its supply information. The oracle will supply configurations of Orders and Supplier information such that client requests will be satisfiable approximately 60% of the time.

3.2 Functional Requirements

Scenario A Using Jade, your task is to:

1. Extend the basic ontology given for the scenario to include AgentAction concept classes for the message contents of:
 - (a) The call for proposals
 - (b) The proposals from suppliers.

It is recommended that you use Protégé and the JadeBeanGenerator for this.

2. Provide a Jade agent implementation for a client and a supplier (all of the suppliers should share the same implementation class), which instantiates the contract net protocol described above. You should use the Drools4Jade behaviours in conjunction with Drools rule-sets to implement this protocol.

Scenario B This is an extension of the previous problem. Whereas in the previous scenario the probability that a given request can be satisfied is relatively high, in this scenario suppliers will be allocated supplies and orders generated such that the probability of a given supplier being able to completely satisfy an order on its own will be low.

Your second task is to extend the supplier agents to sub-contract requests to other suppliers, in the case that they are unable to satisfy requests.

In addition to the previous scenario there will be an asymmetric fixed shipment cost between each pair of suppliers: Suppliers will be able to determine transport costs between themselves and another supplier by sending a “SupplierTransportCostRequest” message containing a source supplier and a target supplier to the oracle. The oracle will reply with a “SupplierTransportCostResponse” message indicating the cost of making a shipment between these two suppliers.

The supplier implementation should be extended to support issuing calls and proposals to other suppliers to sub-contract the portion of the requested order that the original supplier could not meet. After a successful sub-contract is agreed the supplier should then reply to the client with a proposal in the same way as in scenario A. You may consider further subcontracting in a recursive fashion, although you should ensure that transactions terminate eventually.

3.3 Non-functional Requirements

- The program must be entirely written in Java and Drools.
- The program should be written in an object-oriented fashion. Methods consisting of more than 30 lines of code will be looked at with suspicion.
- The design should be as modular as possible, allowing reuse of code. Furthermore, the design should anticipate possible extensions of the software.
- The program should work correctly on the BUCS system.
- Only the standard JADE toolkit may be used in conjunction with the skeletal agents provided and the initial ontology.

3.4 Software Supplied

The following components are made available for download:

- the Jade agent platform (version 3.5, download from jade.tilab.com)
- the Protégé ontology editor (version 3.3.1, download from protege.stanford.edu)
- the Beangenerator plugin for Protégé (version 3.2.1, download from <http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator>)
- the Drools Rules Engine (version 4.0.1, download from <http://labs.jboss.com/drools/>)
- the Drools4Jade package.
- the Drools Eclipse Workbench (version 4.0.1, download from <http://labs.jboss.com/drools>)
- A basic ontology in Protégé: and the generated Java code
- An “oracle” agent which determines initial allocations for the suppliers and an order for the client.
- Skeletal client and supplier implementations which contact the oracle and get the initial allocation and orders.

The first week of the assessment period is to be used for familiarisation with Jade (see the Jade tutorial) and with Protégé.

3.5 Test Cases

Your agents will be run in a series of tests with different conditions to ensure that they act correctly in each situation. For Scenario A, the test cases are:

1. One supplier capable of supplying the goods.
2. One supplier not capable of supplying the goods.
3. Two suppliers, one capable of supplying the goods, one not.
4. Two suppliers, both incapable of supplying the goods.
5. Two suppliers, both capable of supplying the goods.

And for Scenario B, the cases are:

1. Two suppliers capable of supplying the goods. One level of recursion is required.
2. Two suppliers not capable of supplying the goods.
3. Three suppliers capable of supplying the goods. Multiple levels of recursion required.
4. Three suppliers not capable of supplying the goods.
5. Five suppliers randomly capable of supplying the goods.

3.6 The Report

The documentation that accompanies your agent submission should contain the following sections (ordered and numbered as listed here):

1. A project description
2. A detailed description of the different agents you implemented
3. A justification for the decisions made
4. A critical analysis of your implementation
5. Directions for further improvements
6. Evidence and discussion of test case results
7. The source code

4 Assessment

4.1 Conditions

The coursework will be carried out individually. Attention is drawn to the University and Departmental rules on plagiarism (page 49 of the Programmes Handbook). The coursework can be worked on during the tutorial sessions. The coursework also involves work during one's own study time.

4.2 Marking

This coursework counts for 25% (20% for MSc students) of your final mark for this unit. Marks are awarded on the basis of the submitted documentation and the running of the submitted code against the test cases identified earlier. The breakdown of marks follows:

- **60 percent** – Documentation.
- **15 percent** – Scenario A (3 percent for each test case).
- **25 percent** – Scenario B (5 percent for each test case).

The following is a guideline so that you are aware of what you may typically need to do for the documentation:

- 1st. The deliverables reach the standard for a 2:1 student. In addition, the documentation demonstrates deep understanding of the problem and pulls in ideas from background reading. The documentation demonstrates a clear appreciation of other areas of knowledge and sources of expertise that might be brought to bear on the coursework. Evidence and discussion of passing a significant number of scenario B test cases.
- 2:1. The coursework idea is clearly well understood. Evidence of the identification of background reading sources is shown. Evidence and discussion of passing all scenario A test cases. Furthermore, demonstrates an understanding of the requirements needed to implement Scenario B.
- 2:2. The students started to expand/modify the original coursework idea to suit their own skills and understanding. Some evidence of background reading is shown. The description of the agent behaviours and justification of decisions made demonstrates effort to make the different kinds of agent meet the requirements in the time-frame of the coursework. Evidence and/or discussion of passing a significant number of scenario A test cases.
- 3rd. The students have hardly expanded on the supplied agent skeletons and the description of agent behaviours and the ontology demonstrates a lack of understanding of the code that was given to them. Very little effort is made to extend in the direction identified by the requirements. Evidence and/or discussion of passing some scenario B test cases.

In borderline cases, the unit lecturers may call students for viva/demonstration to reach a final decision.

4.3 Deadlines

The coursework is published on **Tuesday 10th November**. Copies of the description are also available via Moodle. The deadline for submission is **Tuesday 15th December, noon**.

4.4 Coursework Submission

You should submit your coursework using the unit's Moodle page (<http://moodle.bath.ac.uk/moodle5/course/view.php?id=400>). You should submit your report in pdf format along with two zip files, one for scenario A and one for scenario B. Each of these files should contain:

- A directory containing the source code for your agents.
- A jar-file containing the compiled code for your agents.

4.5 Feedback

Feedback will be given in the form of individual written comments on Moodle within three weeks of the submission deadline.