Assessment Brief Proforma

1. Module number	SET10111
2. Module title	Multi-Agent Systems
3. Module leader	Simon Powers
4. Tutor with responsibility for this Assessment Student's first point of contact	Simon Powers
5. Assessment	Report
6. Weighting	60% of the total module mark.
7. Size and/or time limits for assessment	**Report of no more than 4000 words. You should spend approximately 48 hours working on this assessment.**
8. Deadline of submission	Report and code to be submitted to Moodle by **4th December 2020 at 15:00**. Demonstration during the week beginning 7th December. No demonstration may result in a mark of 0.
9. Arrangements for submission	The report should be submitted to Moodle as a .PDF document. All of your source code files should be uploaded to Moodle as a .zip archive. You are advised to keep your own copy of the assessment.
10. Assessment Regulations	All assessments are subject to the University Regulations

11. The requirements for the assessment	***See attached assessment brief.***
12. Special instructions	Your multi-agent system must be implemented in JADE.
13. Return of work and feedback	Written feedback will be made available within three working weeks of submission.
14. Assessment criteria	See attached. Please note that checks for plagiarism will be made on electronic submissions. Students may be required to attend a further demonstration if there exists doubts as to the authorship of work.

Your task is to design and implement a prototype of a multi-agent system to assist with allocating School of Computing students into timetabled tutorial groups for their modules. This must be evidenced in a report consisting of no more than 4000 words (excluding code, diagrams, figures, tables and references).

Problem specification

You must develop a system where each student is represented by an agent. The agent stores the preferences of its student for the times when they would or would not like to attend tutorials. The agent will then try and meet these preferences, by negotiating with the agents representing other students to swap between tutorial groups. This should be done in a decentralised manner, such that the preferences of each student remain private to their student agent. For further background on the agent-based approach to timetabling, you may read Urquhart & Powers (2019).[1]

Tutorials take place in one hour *timeslots* from Monday to Friday between 0900-1800 (the first timeslot on a day begins at 0900, and the last timeslot begins at 1700). In your system students should initially be allocated into one tutorial group for each of their three modules by a timetabling agent (this may be done randomly). The timetabling agent is responsible for ensuring that each student is allocated to exactly one tutorial group for each module at all times (i.e. for ensuring the consistency of the timetable). It stores a record of which students are allocated to which tutorial group for each module.

Each student must be represented by its own agent. A student agent stores the preferences of its student. The minimal requirements for the preferences that the student agent should be able to store are:

1. *Timeslots* on each day where the student is *unable* to attend any tutorial (e.g. due to outside work or caring commitments).
2. *Timeslots* on each day where the student would *prefer not to* attend any tutorial.
3. *Timeslots* on each day where the student would *like to* have a tutorial.

The goal of the student agent is to try to satisfy its student's preferences, by negotiating with other student agents to swap tutorial slots. A student agent should be able to send a request to another student agent to swap their slots. If the other student agent agrees, then the timetabling agent should be notified that the swap has occurred so that it can update its records.

[1] Urquhart N., Powers S.T. (2019) An Agent Based Technique for Improving Multi-stakeholder Optimisation Problems. In: Demazeau Y., Matson E., Corchado J., De la Prieta F. (eds) Advances in Practical Applications of Survivable Agents and Multi-Agent Systems: The PAAMS Collection. PAAMS 2019. Lecture Notes in Computer Science, vol 11523. Springer, Cham.

Design of your multi-agent system

You must design a multi-agent system that, at a minimum, has the functionality and agents described above. You are free to include additional types of agents, if required by your design. Communication between agents must follow the FIPA specifications (use of FIPA performatives and the FIPA SL content language).

In your design you will need to develop:

1. An ontology that allows the agents to represent relevant information and pass this to each other in the contents of messages. The ontology should extend the base JADE ontology and include any necessary actions, concepts, and predicates. You should state the properties that these have and any restrictions on their values.
2. A communication protocol to allow student agents to communicate with the timetabling agent and with each other (and any other agents in your system). How will student agents find other student agents that they might be able to swap with, without revealing their preferences? You should consider alternatives for this and justify your choice. You should also specify the types of conversations that need to take place between agents. For each type of conversation, you should detail the sequence in which the messages will be exchanged, the FIPA performatives of the messages, and examples of the message content. You should represent this using a sequence diagram for each type of conversation. You should also show the order in which the conversations will take place in your system.
3. A utility function for the student agent. This should be written out mathematically. Remember that utility is a number, where a larger number indicates that the agent is closer to satisfying its goal (meeting the student's preferences). You should justify your choice of utility function, and consider other possible options.
4. A strategy for the student agent that determines which exchange requests it should make, which it should accept from others, and which it should reject. This strategy should aim to maximise the utility of the student.
5. A metric to evaluate the overall effectiveness of the system in satisfying student preferences. You should define and justify how you will measure this, and consider alternative options.

Implementation of your multi-agent system

You should implement your design in JADE.

For each of the conversation types in your design, you should take a screenshot from the JADE sniffer illustrating this conversation (for this you should use the minimal number of student agents necessary to illustrate the conversation). The source code for your entire system (agents, ontology, main, any other supporting classes) will be submitted to Moodle as a .zip file. To facilitate marking and plagiarism detection, the source code for the agent and ontology classes will also be submitted as an appendix at the end of your report.

Testing your multi-agent system

You should design test cases to demonstrate your multi-agent system performing swaps and satisfying student preferences. For each test case you should detail:

1. The total number of students in the system and their preferences (you might hard code the preferences or allocate them randomly, for example).
2. The number of tutorial groups for each module and their timeslots.
3. The number of students each tutorial can hold.

You should have at least three test cases, and they should have varying complexity, e.g. you could start with a basic case with one only one module and two tutorial groups, and then increase the complexity of the problem. You should design the test cases to illustrate how effectively your system performs as the problem becomes more difficult, i.e. student preferences become harder to satisfy. You should think about what metrics you will record as output.

Submission specification

Your report should be a maximum of 4000 words (excluding code, diagrams, figures, tables and references), and must be submitted in .pdf format to the "Coursework report submission" link on Moodle.
All of your source code must be submitted to the "Coursework code submission" link on Moodle inside a .zip file.

The marking rubric is shown at the end of this document.

Your report must contain the following sections:

Design
This must address points 1-5 in the "Design of your multi-agent system" section.
It is recommended that you use one subsection to address each of these points.

Implementation
Marks for this section will come from the evidence asked for below alongside evidence obtained from your demonstration. Marks will be deducted where your implemented system does not fully match your design and/or where it does not fully meet the problem specification. *This section must be supported by screenshots of each conversation in your communication protocol running from the JADE sniffer, and code listings for the agent and ontology classes in an appendix at the end of the report.*

This section must cover the following points.

- State where in the code the student agent calculates its utility, and decides which swap requests to make, which to accept, and which to reject. State where in the code the student's preferences are represented.

- State where in the code the timetabling agent ensures that each student attends exactly one tutorial for each module.

- For each conversation in your communication protocol, you should reference the relevant screenshot from the JADE sniffer, and state which agent behaviours implement it.

Testing
Describe your test cases, and justify why you have chosen them. Present the results from running your system on each test case, and justify the output metrics that you have chosen.

Evaluation and future work
This section should address the following points:
1. How will the effectiveness of your system in will change as the problem becomes more difficult?
2. What are the advantages and disadvantages of taking a multi-agent systems approach to this problem?
3. In light of 1 and 2, suggest and justify an improvement to your system.

References
Include bibliographic details of any sources that you cite. This should follow APA style.

You must also demonstrate your system running (via a short WebEx meeting) after submission during Week 14. Your work will not be marked without a demonstration.

Your code and report must be written by yourself, specifically for this module. Where have you referred to any sources these should be cited in your report. Because this is an individual assessment, working together with other students is not permitted and will be treated as plagiarism under the university’s academic regulations. Any code obtained from the internet must be cited with a URL. If it is suspected that your work is not your own then you will be referred to the Academic Conduct Officer for investigation. **If you use version control such as GitHub be sure to store your work in a private repository to prevent access by others.**

Marking rubric

The total mark available is 60. This will be broken down as follows.

	<40%	40-50%	50-60%	60-70%	70%+
Design (35% of total mark)	Not present, or fails to use a multi-agent systems approach, or fails to meet minimal requirements of exchanging timeslots.	The design presents a multi-agent system that exchanges timeslots, but the design choices may be inappropriate. Justification of choices will largely be lacking.	A good design that meets the minimal requirements with appropriate design choices in most areas. Some justification for the design choices is given.	Very good with appropriate design choices in all areas. All design choices are justified, although comparison with alternatives may be superficial.	The design choices are appropriate in all areas. Justification of design choices is excellent, possibly going beyond the taught material by considering issues such as scalability or applicability to a more realistic scenario.
Implementation (40% of total mark)	Not present in the report or not able to demonstrate exchange of timeslots between student agents.	A minimal functioning implementation is provided that demonstrates exchange between student agents. Some functionality may be missing, e.g. the timetabling agent, or the student agent strategy.	The core functionality of the student and timetabling agents is working, so that the implemented system fully meets the problem specification. There may be significant errors in the implementation of the communication protocol or ontology, or the	The problem specification is fully met. The implementation mostly matches the design. There may be some minor issues with the communication and ontology, e.g. around FIPA compliance.	The implementation fully meets both the design and the problem specification. The communication protocol and ontology are fully implemented and are FIPA compliant.

			implementation may differ significantly from the design.		
Testing (15% of total mark)	No evidence of test cases.	Test cases are presented but there may not be evidence of them running, or they may not be appropriate.	At least three test cases are presented and executed. There is some attempt at justifying them, but the output metrics may be unclear.	At least three test cases are justified and executed. Output metrics are clear and appropriate, although full justification may be lacking.	At least three test cases are chosen where their justification demonstrates an excellent understanding of the problem. The results are clearly presented with appropriate and clearly explained output metrics. The test cases clearly demonstrate how the system performs with increasing difficulty of the problem.
Evaluation and future work (10% of total mark)	Not present or fails to address points 1-3 above.	An attempt is made at addressing points 1-3, but it may be superficial or not linked to the student's system.	Points 1-3 are covered and linked to the student's system, demonstrating a good understanding of multi-agent system technologies	Points 1-3 are clearly covered and linked to the student's system, demonstrating a very good understandi	Points 1-3 are fully covered, demonstrating an excellent understanding of multi-agent system technologies which may

			in the context of this problem. The suggestion for further work may be superficial	ng of multi-agent system technologie s and their implications for this problem. The suggestion for further work is appropriate.	go beyond the taught material.