1 Implementation

We wrote the model in Java, using the JADE[1] library, developed by Tilab **(check)**. We chose it because it offers a range of agent and behaviour types as well as a built-in support for communication between agents.

For the generation of random numbers, we used the uniform distribution RNG built in the Java standard library.

We also implemented the option to load the simulation set-up, including all randomly generate quantities, from a JSON file generated during another simulation.

1.1 Classes overview

The model is mainly composed of two type of classes: agents and behaviours.

TO DO

1.2 Communication protocols

The main communication protocol we designed is the one responsible for the communications between people and restaurants, which is shown in figure 1. The communication is started by a person agent, which sends a *Request* to the restaurant. If the restaurant is already full, it will answer with a *Refuse*, ending the communication. Otherwise it will answer with a *Propose*. Normally, the same person sends many requests at once to different restaurants. After receiving all the answer, it chooses the best restaurant with one of the strategies previously described. She then sends an *Accept Proposal* to this restaurant. At this point, this checks again how much space left it has. If there's none, it will send a *Failure* and the communication will end. This may happen in case of slow communications and concurrent actions by many people. If instead it still has some space left, it will send the person an *Inform* containing it's quality and set the seat as occupied. This ends the communication.

1.3 Global

To control the turn structure and initialize the model, we implemented a third type of agent called global.

Figure 2 shows the internal logic of the global agent. This can be divided in three phases (blue boxes) and some other actions. At first, the global agent initializes all the other agents and collects their data in JSON format. At the end of this first phase, it saves the JSON configuration files. Then, it signals the restaurants to reset and waits for their answer. At last, it signals the start of the turn to the people. At the end of the turn, each person sends its data to the global agent. This turn sequence is repeated for a fixed number of times, chosen at the start of the simulation. At the end of the last turn, the global agent saves on disk the log files containing all the data of the simulation.

The internal structure of a single phase is shown in figure 3. All of them start with *Confirm* performative broadcast to all the agents which signals the beginning of the phase. The different phases are differentiated based on the message ontology. Then the Global agent listens for incoming messages with

[1] **link to site**

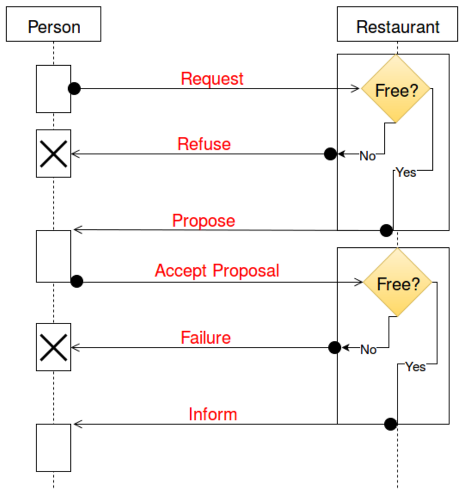

Figure 1: Diagram of the communication protocol between people and restaurants

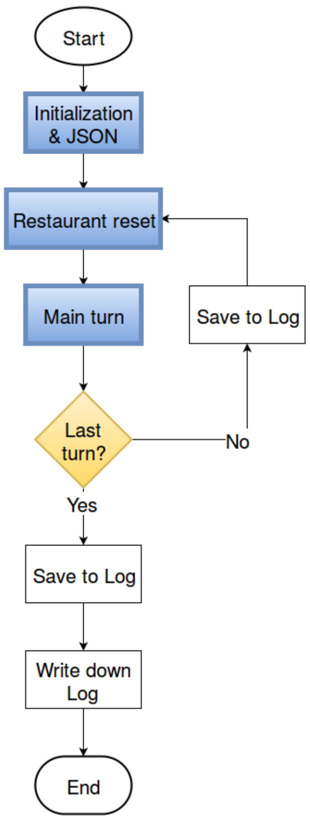

Figure 2: Diagram of the internal logic of the Global class

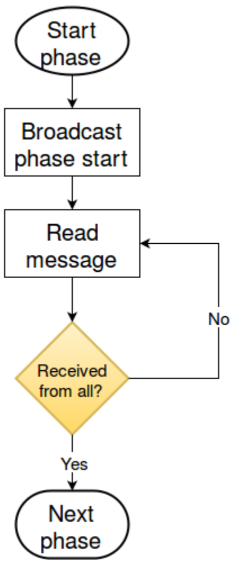

Figure 3: Diagram of the internal logic of a phase of the Global class logic

the result of the computation for that phase till it has received an answer from all the interested agents.

1.4 Restaurant

The restaurant agents work as an automatic answerer: their only task is to answer to incoming questions asked by the other agents. Their two parameters, capacity and quality, are randomly generated in a specified range on initialization.

As figure 4 shows, the internal logic of the restaurant agents is rather simple: they listen to incoming messages and answer based on the message performative. If the message performative is *Request*, the restaurant checks for free space and answers with a *Propose* if there is any or a *Refuse* if there is none. This is the first phase of the person-restaurant communication protocol.

On an *Accept Proposal*, it will again check for free space and answer with an *Inform* containing its quality if there is still unoccupied space or a *Failure* if there is none. This is the second phase of the person-restaurant communication protocol.

At last, if the message performative is *Confirm*, it will answer with a propagate only if the message ontology is ”reset”, which corresponds to the restaurant reset phase in the turn structure.

1.5 Person

Unlike the restaurant agents, the person agents play a more active role in the simulation: as it's been said earlier, they have the ability to take decisions and have a personal opinion regarding the external world in the form of a knowledge database.

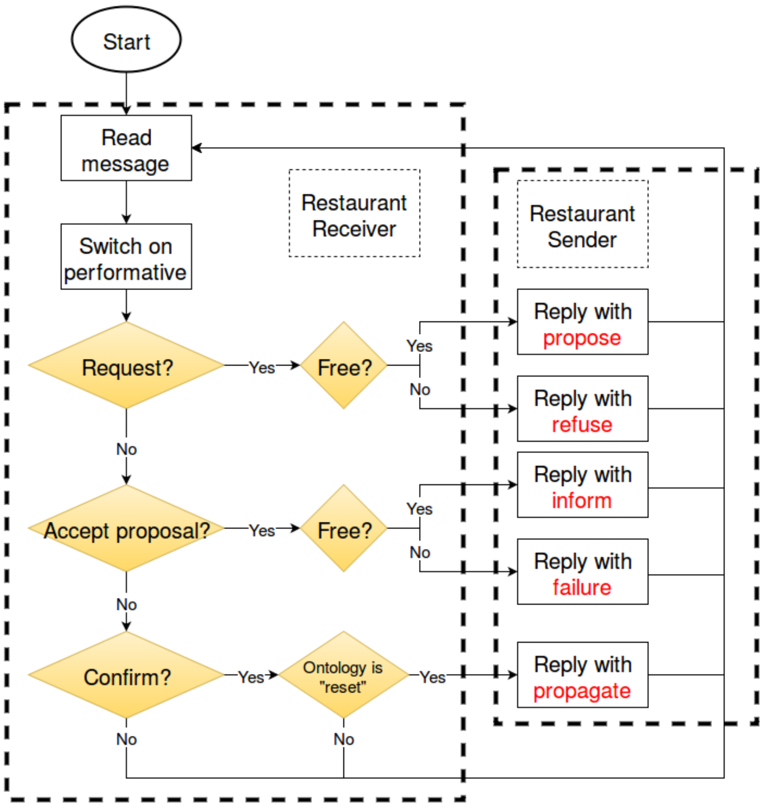

Figure 4: Diagram of the internal logic of the Restaurant class

The three properties of each person agent, *boldness*, *maximum evaluation* and its *list of friends* are randomly generate on initialization. Their range is set at the start of the simulation.

Figure 5 shows the internal logic of a person agent. After the global initialization phase described in section 1.3 the agent waits for the turn start broadcast. Afterwards it starts the *search* routine: it sends a *Request* message to every restaurant which it thinks to be better than its *maximum evaluation* parameter. Then it collects all the answers, selecting only the positive ones, containing a *Propose*, and starts the *choose* routine. At this point, it selects randomly one of the three strategies described in section **??**. The probabilities with which it will select one of them is set at the start of the simulation. Using one of these strategies, it selects the best restaurant and sends it a *Accept proposal* with its *eat* routine. If somehow in the meantime the restaurant has received enough bookings to fill all the available tables, it responds with a *Failure* and the person will start again the whole process excluding the previously chosen restaurant from its search list. Otherwise, the restaurant sends it an *Inform* containing its effective quality. The person then compares this with the restaurant record in its knowledge database and update the latter with the *evaluate* routine. The update rule is better described in section **??**. At last, the person sends its new evaluation to its friends, who will use it to update their thrust map with a similar rule when they'll eat in the same restaurant, and notifies the global agent the end of its turn.

1.6 File output

The output data is recorded at the end of the simulation in cvs tables.

TO DO

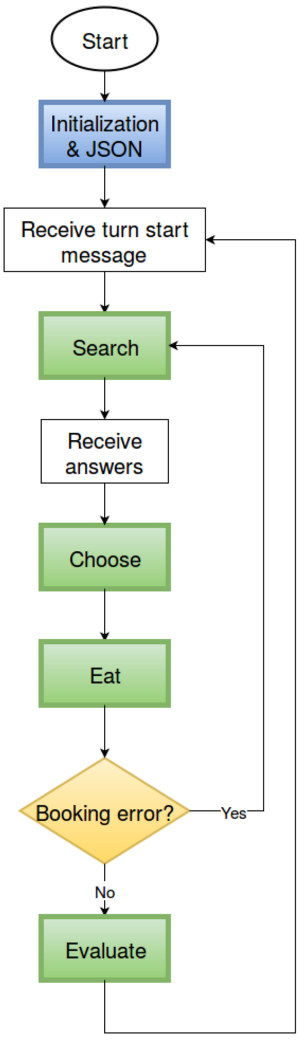

Figure 5: Diagram of the internal logic of the Person class