ID2209 Distributed Artificial Intelligence and Intelligent Agents
Homework 1: Virtual Exhibitions…

Taylor Mordan
Jan Zikeš

Introduction

The aim of this first homework was to implement three agents, a profiler agent, a tour guide agent and a curator agent, in order to create a virtual museum tour for a customer according to his interests. This was to be done using the Jade platform to support and run the multiagent system.

It was made to get a first hands-on experience with programming agents on the Jade platform ant to think about how communications between agents can help to solve distributed problems such as the one suggested.

Summary of the events

To complete the task, the agents are acting as follows:

1. All available museums are registered on the DF by their curator agents.
2. The tour guide agent records the list of all curator agents using the DF, subscribes to it so that it can keep an updated list at any time, and finally registers also to the DF.
3. A profiler agent requests a tour to the tour guide agent (found on the DF), providing the user's interests.
4. The tour guide agent asks all the curator agents their catalogs, makes a selection and then sends the list of suitable items to the profiler agent.
5. The profiler agent asks for the user to input this choice of museum, and gets information on the interesting artifacts in this museum by asking it to the right curator agent.

Description of all agents' behaviors

Profiler agent

After searching the DF for any tour guide offer and finding one, the profiler agent uses a S*equentialBehavior* to order its actions. The first subbehavior is a *OneShotBehavior* which asks for generating a museum tour based on the user's information and interests that it provides at the same time. Once finished, the agent then uses a *MsgReceiver* behavior to get the names of museums with their interesting artifacts from the tour guide agent, after this one has found all relevant information.

It then offers this list to the user, asking him to input his choice, based on the names of the museums and their items. Once done, the agent gets the detailed characteristics of all the artifacts form the selected museum using a *SimpleAchieveREInitiator* to send a request to the associated curator agent and wait for its response.

Tour guide agent

Its first feature is that it subscribes to the DF in order to get notified each time a new museum register, and therefore can easily keep a list of all available museums up-to-date. To do this, it uses a *CyclicBehavior* that simply adds the name of the new museum to the existing list. After registration to the DF, the second behavior is also a *CyclicBehavior* and is used to handle all requests from diver profiler agents. Once such a request is received, a *SequentialBehavior* is run to effectively generate the corresponding tour. This way, the tour guide agent is able to deal with several customers at the same time.

The first step of this sequence is a *ParallelBehavior* that will run as many subbehaviors as there are curator agents it knows. Each of these is a *SimpleAchieveREInitiator* that requests the catalog to a curator agent and waits for it, and then stores the relevant information. The ParallelBehavior only finishes when all its subbehaviors are done, that is to say, when the agent has got all what it needs. After this one, the following behavior is a *OneShotBehaviour* which is to select which of all the received items are suitable for the customer, based on its interests. This was done here using a simple matching method between the interests and the types of the artifacts – and should be replaced by advanced methods of Artificial Intelligence and Machine Learning! Lastly, the final step of the SequentialBehavior is a *OneShotBehaviour* again. Its goal is simply to send the chosen collection of suitable artifacts along with the museums which hold them to the profiler agent.

Curator agent

The first action of this agent is to register two services to the DF, which are offering of the catalog of the museum and offering of detailed information on a specified artifact. Then, its only one behavior is a *SimpleAchieveREResponder* which handles both services. When a request is received, it will inspect the message to figure out which one of the two offered services is asked for. The response sent back is therefore the suitable information, namely the list of the names of all items in the museum or the detailed characteristics of a particular artifact.

Running the program

You have several options how to run the code:

1. Download the source code and unpack it in NetBeans, for this is recommended to use standard approach, import project from zip. Then based on the jade documentation install download jade and add it as a jar to the project. After that you need to update properties, set –gui as a parameters and jade.Boot as the main class. Then you can just run the project from NetBeans and use jade's GUI to add agents to the system.
   THIS WAY OF RUNNING THE CODE IS STRONGLY RECCOMENDED!!!
2. You can also compile source files from the command line and run the jade with for example the following parameters:
   -gui -agents
   "curator:homework1.curator.CuratorAgent(0);profiler:homework1.profiler.ProfilerAgent;tour_guide:homework1.tourguide.TourGuideAgent"

Conclusion

In this homework we have realized a working virtual museum tour guide agent. Since this task is inherently distributed, it has been greatly simplified by using different agents which communicate between each other in order to exchange information not accessible by everyone. It also directly allows to have several independent museums and to work with several customers. The use of Jade has enabled us to build a simple mulitagent system and to handle communications within it in a flexible and easy way, and is thus perfectly suited for such a task.