ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Homework 1

Course Coordinator:
Mihhail Matskin

Course Assistant:
Cosar Ghandeharioon
Shatha Jaradat

Óttar Guðmundsson

Xin Ren

2017-11-15

How to run

Import the folder as a new java project in Eclipse. Start a new agent platform and start the agents in the order CuratorAgent, TourGuideAgent and ProfilerAgent. Start the tour guide with any of these 5 parameters: era, genre, country, narrow, all. If no parameters are used, all will be selected as an artifact matching preference. The Profiler can also be started with the same parameters and more than one of them. Wait for the system to send a request to the TourGuideAgent about interesting artifacts. After the artifacts have appeared on the screen, select an artifact by double clicking on an item to make a request to the CuratorAgent which results in a new screen that opens up with detailed information about that artifact.

ProfilerAgent

The agent that is responsible for the user viewing the system. It starts up by registering to the DFAgent and finding all possible *TourGuideAgent* or *CuratorAgent* that exists. After that a CyclicBehaviour is added to take care of incoming messages, notifications from DF for other agents that might be started and replies from *TourGuideAgent* and *CuratorAgent,* while the session is running. A parallel behavior is used to keep track of all different types of TourGuideAgent. OneShotBehaviour is used for initiating a new user profile followed by a WakerBehaviour of 2 seconds that requests tours if the *ProfilerAgent* doesn't see any artifacts in the application. TickerBehaviour is used to refresh an ad application at the bottom on a fixed interval. SenderBehaviour is used when sending message to *TourGuideAgent* and *CuratorAgent.* If no tour guides are available for that the profile matching preferences after 10 seconds, initiated by a WakerBehaviour, it will be closed.

TourGuideAgent

This agent is quite complex due to its FSMBehaviour. It starts pretty similar as the *ProfilerAgent* by registering and receiving notification when a new *CuratorAgent* joins the DFAgent. After that, FSMBehaviour is added to a CyclicBehaviour to manage state transitions when receiving request from *ProfilerAgent* and sending to the *CuratorAgent.* This is important to make sure that the agent will return the matching artifacts from the initial request. The agent will only search for types of artifacts he is supposed to find.

Curator Agent

SequentalBehaviour is kicked off to make sure that the following behaviors will be added in the correct order. This is important since we don't want to register the agent to the DFAgent unless it has already created artifacts to find. Thus, the first sub behavior added is an OneShotBehaviour that creates a new Markov chain and creates a random set of artifacts. After that a new OneShotBehaviour is used to register the agent so other agents can find it and finally a CyclicBehaviour to receive and reply to requests from other agents. The agent has two other methods available, one that creates a tour based on a received user profile from the *TourGuideAgent* and then a function to return details regarding an artifact.

Other classes

ArtObject

Class for defining a new artifact in a museum. On creation it receives a random id number, individual name and artist provided by the Markov chain. The artifact will be also be given a place of origin as well as a genre it belongs to and when it was created. The latter two attributes are used for matching against the user profile.

ArtDisplay

A smaller class of the ArtObject that only keeps the id, name and artist that is displayed for the *ProfilerAgent.*

MarkovChain

A simple Markov chain is created for each *CuratorAgent* started and trained on more than 141000 real artifacts that actually exist. On creation, a random number of artifacts are generated with random names and artists.

UserProfile

When a new *ProfilerAgent* is created it also creates a random profile attached to it. The profile has random age, gender as well as a particular era in human history that he is interesting within a random range. He can also like one or two genres out of six possible genres defined by the system. The *CuratorAgent* uses a function from this class to compare artifacts and the users' preferences provided by the *TourGuideAgent.*

Tasks

So by that description we have satisfied all sub categories in Task 1 and Task 2.

Task 1 - Implementation of Agent Behaviors

Behaviors should correspond to each category below:

- Simple Behavior (at least 5 different behaviors):
  - **CyclicBehaviour**
  - MsgReceiver
  - **OneShotBehaviour**
  - SimpleAchieveREInitiator
  - SimpleAchieveREResponder
  - **TickerBehaviour**
  - **WakerBehaviour**
  - **SenderBehaviour** (not included in list by we implemented it)

- Composite Behaviors (at least 2 different behaviors):
  - **ParallelBehaviour**
  - **FSMBehaviour**
  - **SequentialBehaviour**

Task 2 - Using DF (Directory Facilitator) agent.

Implement an agent that provides some service and registers it at DF **– all agents have some connection to the DF.**

Implement an agent that can discover all registered services at DF, asks user for its choice and then displays the parameters expected to use that service **- ProfilerAgent and TourGuideAgent both do that for the curator.**

Implement an agent that subscribes to DF and gets notified each time the desired service is published at DF **- both the ProfileAgent and the TourGuideAgent are notified when a new agent joins the DF.**