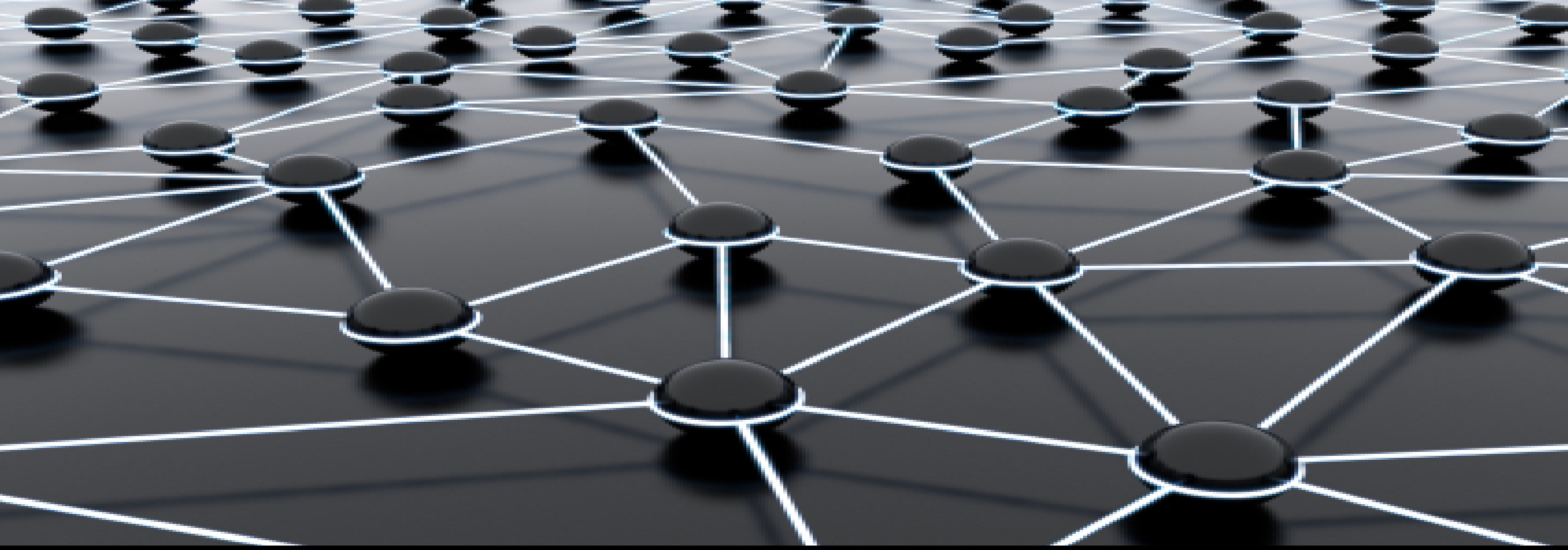


UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Multiagent Simulation

Lab Work #2

STÉPHANE GALLAND

Ref.: UTBM-INFO-IA54-LW2
Ver.: 1.1
Date: 2016/01/20
Status: Public

Laboratoire Systèmes et Transport
Multiagent Group
Université Bourgogne Franche-Comté
Université de Technologie de Belfort-Montbéliard
13, rue Ernest Thierry-Mieg
90010 Belfort cedex, France
http://www.multiagent.fr

1/ Goal of this Lab Work Session

The goal of this lab work session is to write a simulator of the famous Pacman game. You shall learn:

- How to write the agent perception computation.
- How to write the agent perception action mechanism.
- How to write the Ghost behavior.
- How to synchronize the agents.

2/ First Step: prepare the development environment

In this section, you will find the tasks to do for preparing your development environment.

2.1/ Installation of the Eclipse Tools

Recommended version of SARL : 0.3.1-SNAPSHOT (stable)
Recommended version of Janus : 2.0.3.1-SNAPSHOT (release/stable version)

1. Download the **Eclipse product** that contains the compilation tools for the SARL programming language : http://www.sarl.io
2. Uncompress the Eclipse product for SARL.
3. Download the **development version** of the Janus agent execution platform : http://www.janusproject.io
4. Do not uncompress the Jar file of the Janus platform.
5. Launch the downloaded Eclipse product.
6. Open the wizard for creating a SARL project, with the menu:
   > `File` > `New` > `Project` > `SARL` > `Project`
7. Enter the name of the project.
8. Click on the tab with the name `Libraries`.
9. In the list of the libraries, remove the `SARL Libraries`.
10. In the list of the libraries, add the download file of the Janus platform as an external Jar file.
11. Click on ”Finish”, the SARL project should be created.
12. You must ensure that the configuration of your SARL project is correct (may be still a bug in the SARL environment):

a) Open the dialog of the properties of the SARL project by clicking on:
`Right click on project > Properties > SARL > Compiler > Output Folder`

b) Check if the field "Directory" is set to a source folder that is existing in your SARL project. If not change the property with):
`src/main/generated-sources/xtend`

You Eclipse is now ready for the lab work session. You should now install the code skeleton provided by the teachers.

2.2/ Installation of the Code Skeleton

The teachers provide a code skeleton that should be completed by you for terminating the tasks related to this lab work session. The steps to follow for installing the code skeleton are:

1. Download the file with the name `IA54_LW2_skeleton.jar`.
2. Do not uncompress the downloaded Jar file.
3. Open the wizard for importing the code skeleton into the SARL project:
   `Right click on project > Import > General > Archive File`
4. Select on the local file system the downloaded file of the code skeleton; and click on "Finish".
5. The source folders of your project shall contains SARL and Java code.
   Some errors are appearing since they are related to the missed part of the code that must be provided by you.
6. Clean the workspace for ensuring that each existing file is compiled:
   `Menu > Project > Clean`

Figure 1 on the page 5 gives an example of the structure of the SARL project that you should obtain.

2.3/ SARL Documentation

The documentation for the SARL syntax, and the provided elements is available at: http://www.sarl.io/docs/suite/io/sarl/docs/SARLDocumentationSuite.html

2.4/ Bug Report and Questions

In case you have discovered an issue in the SARL tools, you could submit it to the SARL development team via the Github interface: https://github.com/sarl/sarl

In case you cannot discuss with your teacher, you could submit your questions to the SARL development team via the Github interface: https://github.com/sarl/sarl

3/ Brief Description of the Code Skeleton

The skeleton contains an implementation of the environment model (a maze) `fr.utbm.info.ia54.environment`, and the agent that is managing the environment. The subpackages are or will be:

- `fr.utbm.info.ia54.environment.maze` is the package that contains the definition of the maze, and the objects inside.
- `fr.utbm.info.ia54.environment.agent` is the package in which you could find the definition of the agent and the interaction space that are needed for supported the maze environment in the simulation.

It is recommended to read this code and the associated Javadoc.

The package `fr.utbm.info.ia54.players` contains the playing agents, i.e. the ghosts.

The packages `fr.utbm.info.ia54.math` and `fr.utbm.info.ia54.ui` contain mathematical tools and the graphical user interface. The file `fr/utbm/info/ia54/PacManSimulator.java` contains the main program.

4/ Work to be Done during the Lab Work Session

The following sections describe the work to be done during this lab work session.

4.1/ Capacity to Manage a Maze

The agent `Environment` is in charge of managing the maze and the interactions with the playing agents (the ghosts), and the pacman. The agent `Environment` has the knowledge of the maze. It means that this agent has the capacity to manage the maze. This capacity is `MazeManager`, defined in the file `fr/utbm/info/ia54/environment/agent/Capacities.sarl`.

You must create a skill, i.e. an implementation of a capacity, for the capacity `MazeManager`. In the code skeleton, this skill is named `DefaultMazeManagerSkill`, and all the functions inside must be code. The skill contains a reference to the maze.

You should:

a) implement the functions in `DefaultMazeManagerSkill` that implements the functions defined in `MazeManager`.

Two functions may be overriden for coding the skill:

- `def install`: it is invoked when the agent is starting to use the skill; and
- `def uninstall`: it is invoked when the agent is stopping to use the skill.

4.2/ Capacity to be the Front-End to Players

The agent `Environment` is also in charge of providing the perceptions to the playing agents, i.e. the ghosts, and to accept the actions from them. It means that this agent has the capacity to be the front-end to the player. This capacity is `MazeFrontEnd`, defined in the file `fr/utbm/info/ia54/environment/agent/Capacities.sarl`.

You should:

a) implement the functions in `DefaultMazeFrontEndSkill` that implements the functions defined in `MazeFrontEnd`.

Be sure that your capacity is creating an interaction space from the space specification `MazeSpaceSpecification`, and with the proper ID.

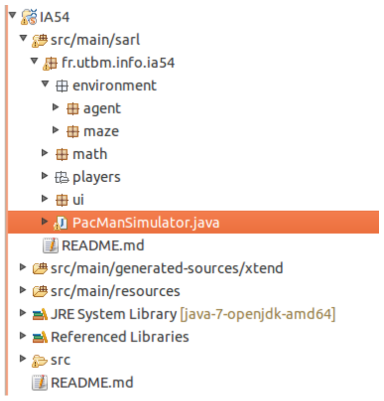
You could take a look in the definition of `DefaultMazeMotionSkill` that is the counter part (in the players) of the `DefaultMazeFrontEndSkill` capacity.

4.3/ Ghost Behavior

The agent `Ghost` must be defined in the file `fr/utbm/info/ia54/players/Players.sarl`

This agent should:

- Pursue the pacman if it is in the field-of-view of the ghost, and the pacman has no super power.
- Evade the pacman if it is in the field-of-view of the ghost, and the pacman has the super power.
- Move randomly, but avoiding to turn back in the middle of a corridor.



```
▼ IA54
  ▼ src/main/sarl
    ▼ fr.utbm.info.ia54
      ▼ environment
        ▶ agent
        ▶ maze
      ▶ math
      ▶ players
      ▶ ui
      ▶ PacManSimulator.java
    README.md
  ▶ src/main/generated-sources/xtend
  ▶ src/main/resources
  ▶ JRE System Library [java-7-openjdk-amd64]
  ▶ Referenced Libraries
  ▶ src
  README.md
```

Figure 1: Example of the structure of a SARL project

Stéphane Galland

Multiagent Simulation
Lab Work #2

Ref.: UTBM-INFO-IA54-LW2
Ver.: 1.1
Date: 2016/01/20
Status: Public

Laboratoire Systèmes et Transport
Multiagent Group
Université Bourgogne Franche-Comté
Université de Technologie de Belfort-Montbéliard
13, rue Ernest Thierry-Mieg
90010 Belfort cedex, France

Contact
Stéphane Galland, Ph.D.
✆ +33 384 583 418
stephane.galland@utbm.fr
http://www.multiagent.fr