AIAD

Library Management Simulation with Distributed Agents

GROUP 01:
Ricardo Cardoso - 201604686
João Lemos - 201000660
Simão Silva - 201605422

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Master's Degree in Informatics and Computing Engineering

1. Problem Description

- The project was made to represent a student allocation problem on the faculty's library;
- Each Student (client) requests allocation to a table;
- The Librarian sends the student's information to each floor in order to get an approval rating from each floor's Security;
- Upon receiving the Librarian request, the floor's Security consults with each of the tables in his floor to see if any table is free to receive the Student. The Security also calculates it's desire to accept that Student (approval rating) based on the number of free tables and the floor's main course.
- This information is passed to the Librarian that then shows the floor with the higher approval rating to the Student;
- Upon receiving the floor information, the Student looks for a free table in that floor and sits.

2. Scheme representation (1/2)

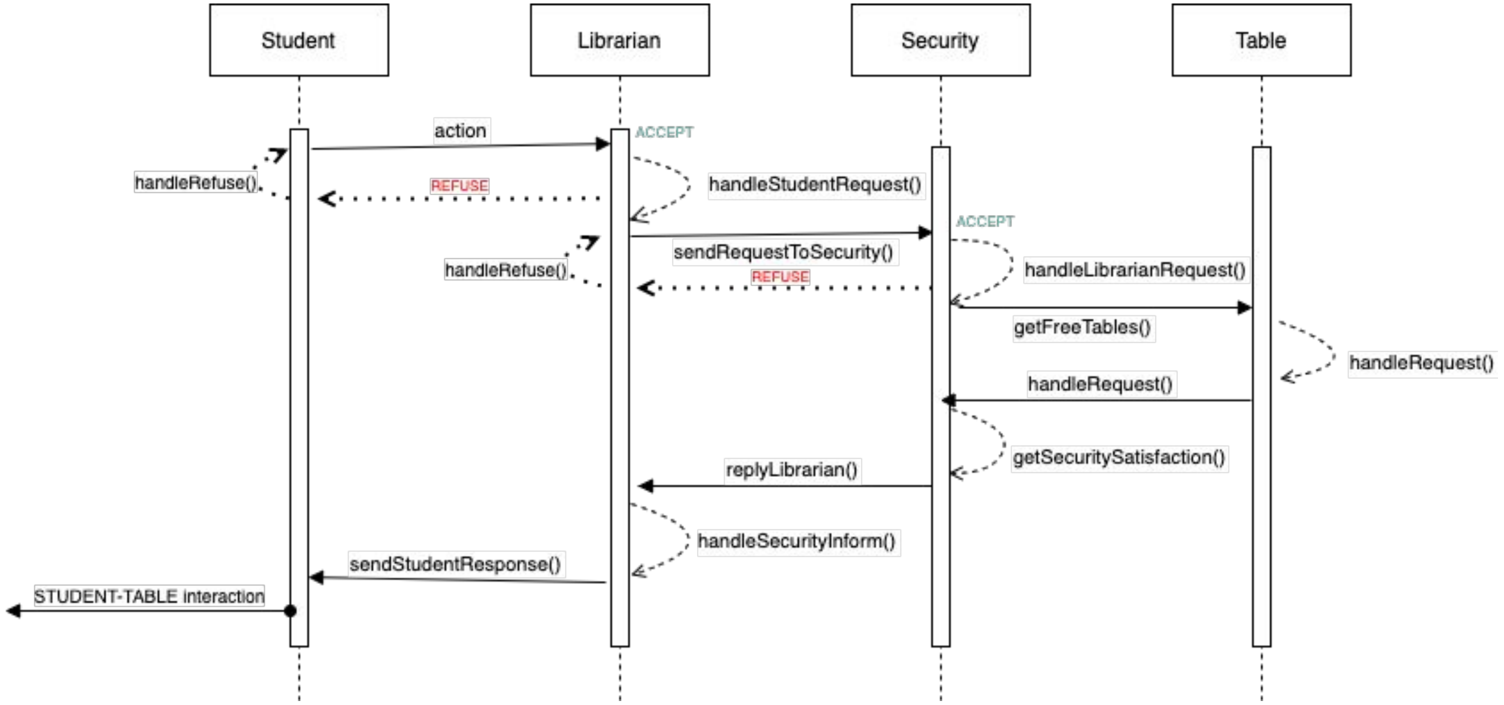


Fig. 1 Representation of Table Request

2. Scheme representation (1/2)

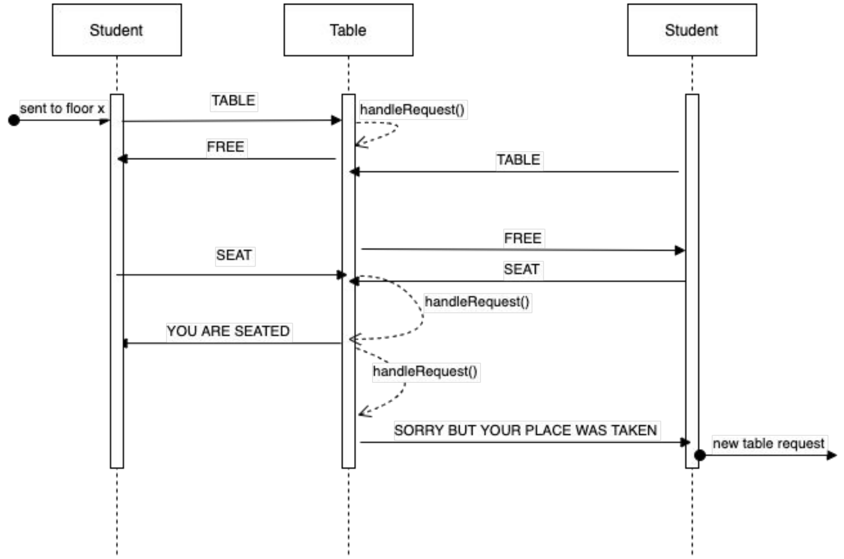


Fig. 2 Representation of Student-Table interaction

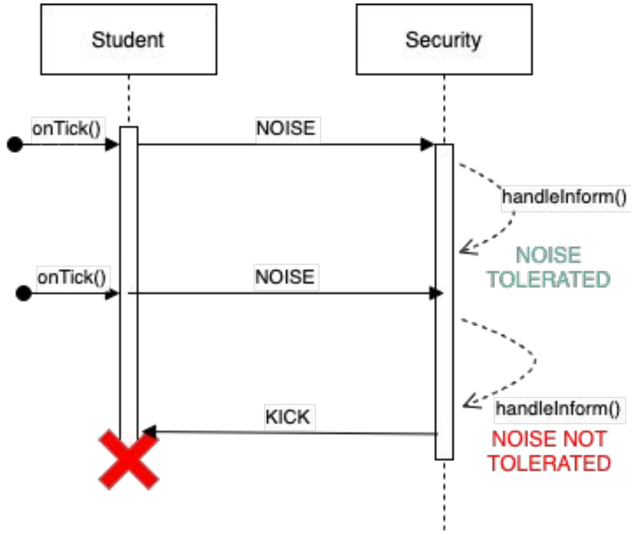
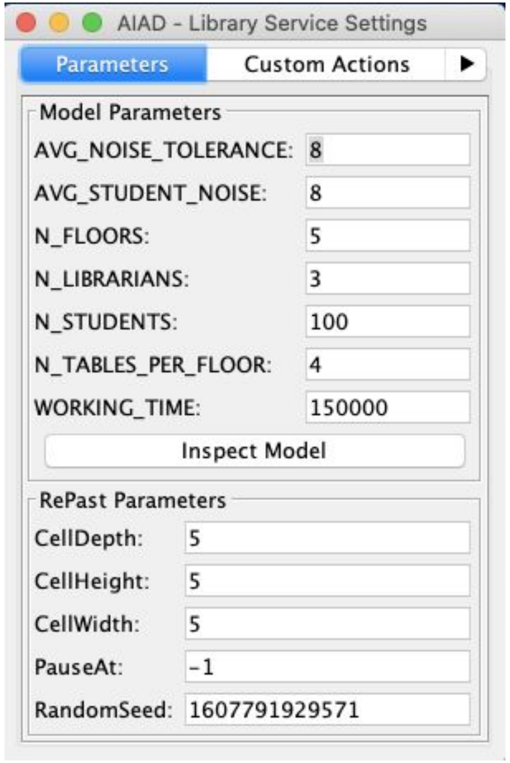


Fig. 3 Representation of Student-Security interaction

3. Independent variables



Model Parameters

AVG_NOISE_TOLERANCE:	8
AVG_STUDENT_NOISE:	8
N_FLOORS:	5
N_LIBRARIANS:	3
N_STUDENTS:	100
N_TABLES_PER_FLOOR:	4
WORKING_TIME:	150000

RePast Parameters

CellDepth:	5
CellHeight:	5
CellWidth:	5
PauseAt:	-1
RandomSeed:	1607791929571

Fig 4. Execution Parameters

- **AVG_NOISE_TOLERANCE**: the average tolerance to noise of all the securities: each security will have a tolerance close or equal to this value (1-10).
- **AVG_STUDENT_NOISE**: the average probability of all students making noise: each student will have a probability close or equal to this value (1-10).
- **N_FLOORS**: the number of floors the library has (1-10). Each floor will have one security and an associated course.
- **N_LIBRARIANS**: the number of librarians attending student's requests.
- **N_STUDENTS**: the number of students that will visit the library during the simulation, each one having a random course associated.
- **N_TABLES_PER_FLOOR**: the number of tables/seats available on each floor.
- **WORKING_TIME**: the simulation time, corresponding to a full working day, in seconds.

4. Dependent Variables

During the simulation runtime, 4 data variables are kept and plotted into a graphical interface:

- **Tables Occupation**: presented in red traced, describes the percentage of occupied tables at each tick.
- **Student Satisfaction**: drawn in blue, always starts at 100%, decreasing when a student doesn't have a seat available on the floor of its' course.
- **Kicked Students**: percentage of kicked students from the library, uses only the students that are/were seated as sample base (in green).
- **Seated Students**: percentage of students who are/were seated, in relation to the full database of students (black trace).

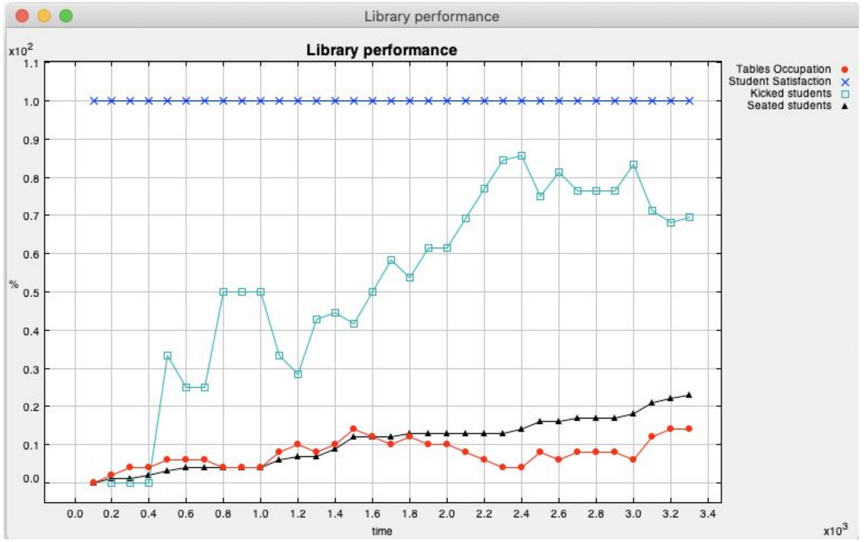


Fig 5. Performance plot

5. Interaction Space (1/2)

The interaction between agents in the simulation is displayed on a new window that opens during the simulation runtime.

The display is divided into two distinct zones: the first draws an X by Y matrix where X=N_FLOORS and Y=N_TABLES_PER_FLOOR. Each cell represents a table, and its color shows what is happening in the environment. If a cell is green (initial state), the table illustrated in that cell is free. If it shows a blue cell, it means that a student currently occupies the table. In the case of a red cell, it means that the student presently seated at that table is making noise.

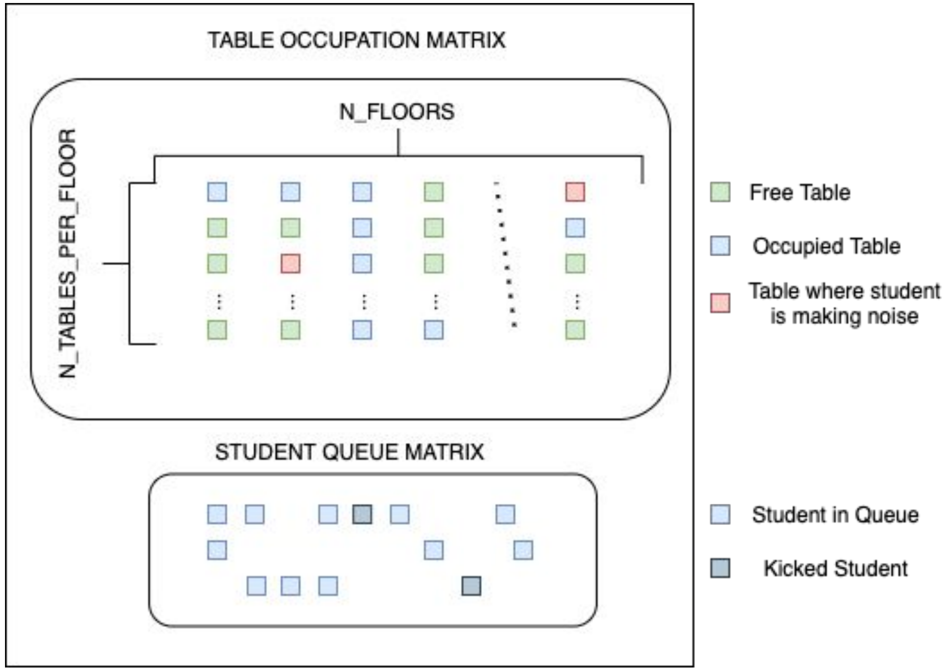


Fig 6. Space mockup

5. Interaction Space (2/2)

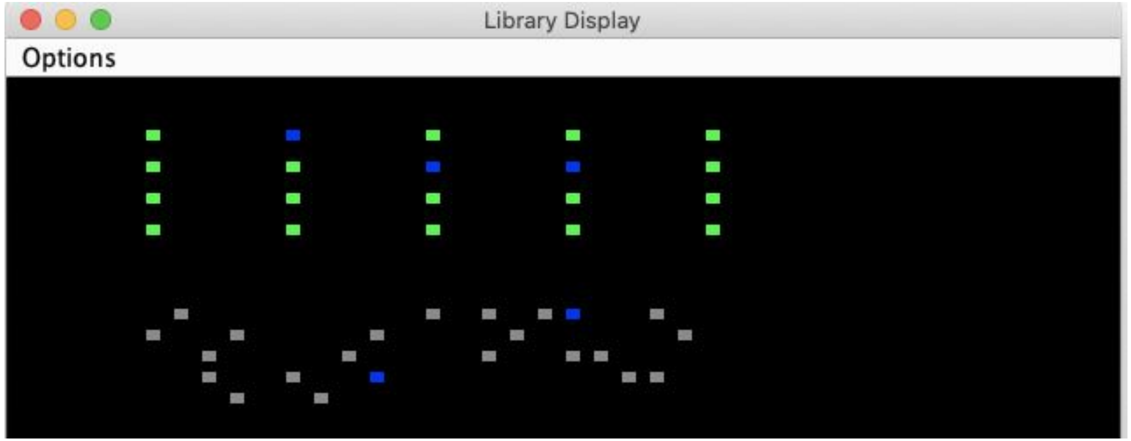
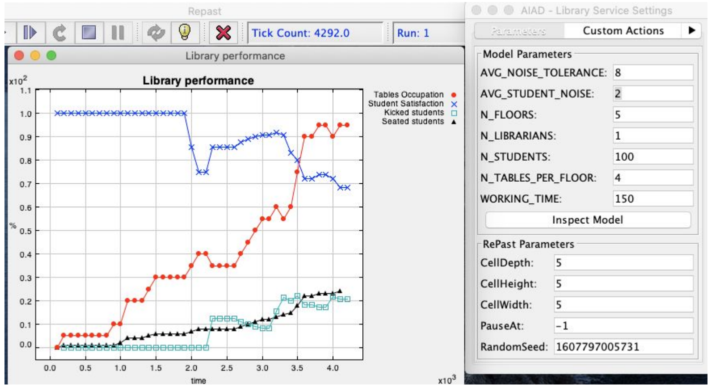


Fig 7. Object2DGrid Space

In the Student Queue Matrix, each position corresponds to the student_id. This means that the student with id=0 will always occupy the first cell in the matrix. If the cell is empty, the student has either not yet arrived at the library or has arrived and is currently seated. If a student is kicked, it returns to its designated spot on the Queue Matrix, changing its' color to grey.

6. Results and Conclusion (1/2)

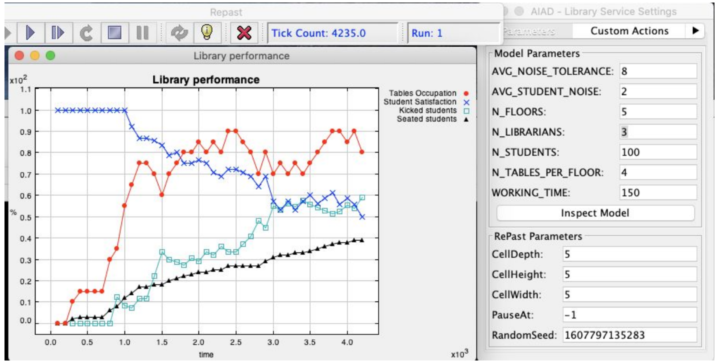


Model Parameters

AVG_NOISE_TOLERANCE:	8
AVG_STUDENT_NOISE:	2
N_FLOORS:	5
N_LIBRARIANS:	1
N_STUDENTS:	100
N_TABLES_PER_FLOOR:	4
WORKING_TIME:	150

RePast Parameters

CellDepth:	5
CellHeight:	5
CellWidth:	5
PauseAt:	-1
RandomSeed:	1607797005731



Model Parameters

AVG_NOISE_TOLERANCE:	8
AVG_STUDENT_NOISE:	2
N_FLOORS:	5
N_LIBRARIANS:	3
N_STUDENTS:	100
N_TABLES_PER_FLOOR:	4
WORKING_TIME:	150

RePast Parameters

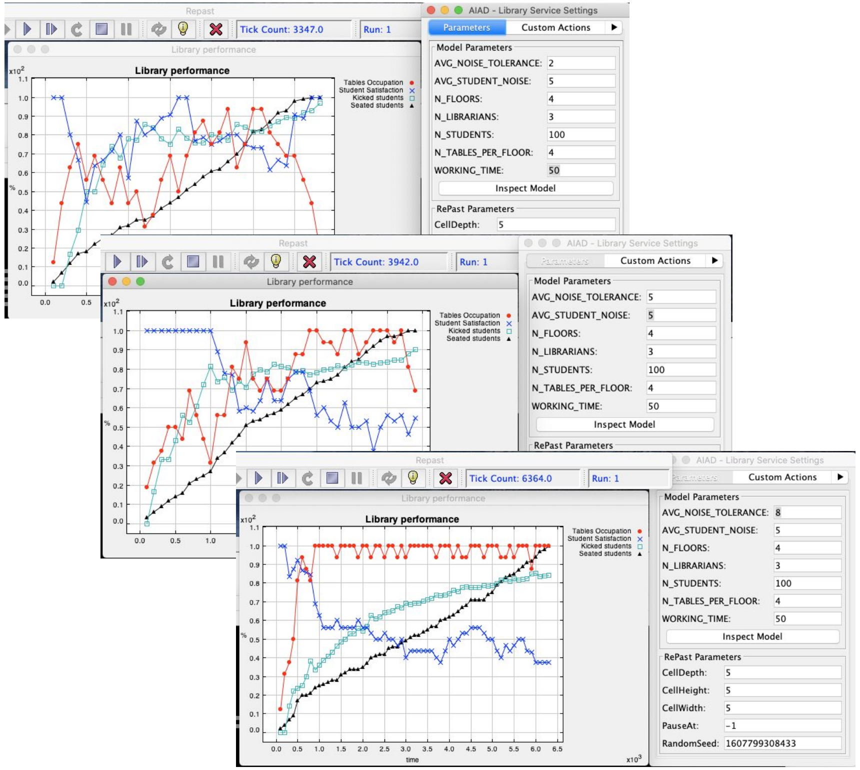
CellDepth:	5
CellHeight:	5
CellWidth:	5
PauseAt:	-1
RandomSeed:	1607797135283

Fig 8: Experiment 1 with a variable N_LIBRARIANS parameter.

N_LIBRARIANS influence:

For the following experiment, two environments were created, both with the same levels on all parameters except N_LIBRARIANS. While in the first trial, only one librarian was present, the second had 3. This increase in the number of librarians generates a faster growth in table occupation rate and everything that naturally comes with that value increase. This means that, as the simulation is single-threaded, having more librarians helps to distribute better incoming requests from students, reducing the bottleneck that otherwise would occur. As a consequence of getting positive responses faster, students can proceed to their tables more efficiently timewise.

6. Results and Conclusion (2/2)



Tick Count: 3347.0 Run: 1

Model Parameters

AVG_NOISE_TOLERANCE:	2
AVG_STUDENT_NOISE:	5
N_FLOORS:	4
N_LIBRARIANS:	3
N_STUDENTS:	100
N_TABLES_PER_FLOOR:	4
WORKING_TIME:	50

RePast Parameters

CellDepth:	5

Tick Count: 3942.0 Run: 1

Model Parameters

AVG_NOISE_TOLERANCE:	5
AVG_STUDENT_NOISE:	5
N_FLOORS:	4
N_LIBRARIANS:	3
N_STUDENTS:	100
N_TABLES_PER_FLOOR:	4
WORKING_TIME:	50

Tick Count: 6364.0 Run: 1

Model Parameters

AVG_NOISE_TOLERANCE:	8
AVG_STUDENT_NOISE:	5
N_FLOORS:	4
N_LIBRARIANS:	3
N_STUDENTS:	100
N_TABLES_PER_FLOOR:	4
WORKING_TIME:	50

RePast Parameters

CellDepth:	5
CellHeight:	5
CellWidth:	5
PauseAt:	-1
RandomSeed:	1607799308433

Fig 9: Experiment 2 with variable AVG_NOISE_TOLERANCE.

AVG_NOISE_TOLERANCE influence:

For the following experiment, two environments were created, both with the same levels on all parameters except for AVG_NOISE_TOLERANCE, and were kept running until all students entered the library. The AVG_NOISE_TOLERANCE varies between low (1st trial) and high (3rd trial). We can see that if securities are less tolerant to noise, more students can use the library and with higher satisfaction. However, this generates a lot of empty tables.

On the other hand, if the securities are more tolerant, students aren’t kicked at a fast enough rate to ensure a satisfactory flux of students in the library entering a “first come - first served” policy. As can be seen, the simulation takes a lot more time to process all the students. The Option with a medium security noise tolerance and a high rate of student noise ensures a healthy flux of new students keeping the quiet ones and sending the loud ones away.

AIAD

Library Management Simulation with Distributed Agents

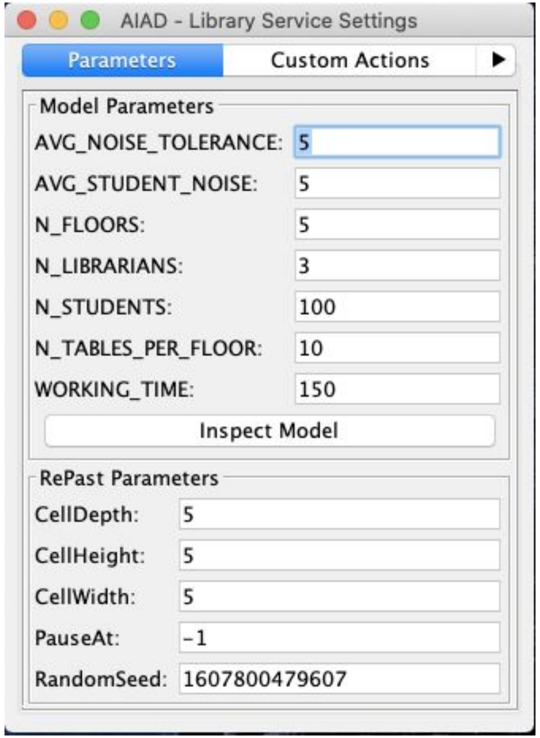
Additional Information

GROUP 01:
Ricardo Cardoso - 201604686
João Lemos - 201000660
Simão Silva - 201605422

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Master's Degree in Informatics and Computing Engineering

1. Detailed Example (1/3)



AVG_NOISE_TOLERANCE:	5
AVG_STUDENT_NOISE:	5
N_FLOORS:	5
N_LIBRARIANS:	3
N_STUDENTS:	100
N_TABLES_PER_FLOOR:	10
WORKING_TIME:	150

CellDepth:	5
CellHeight:	5
CellWidth:	5
PauseAt:	-1
RandomSeed:	1607800479607

Fig 10: Input parameters.

Running a simulation is simple. After running the project, we are faced with the input parameters window. In this window, the simulation's model parameters can be changed. Default values are already loaded in. These parameters are:

- AVG_NOISE_TOLERANCE
- AVG_STUDENT_NOISE
- N_FLOORS
- N_LIBRARIANS
- N_STUDENTS
- N_TABLES_PER_FLOOR
- WORKING_TIME

The Repast command interface is also shown. To start the simulation, press the play button.

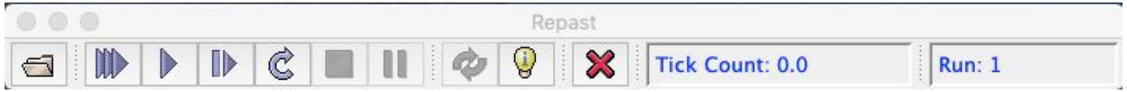


Fig 11: Repast simulation controller.

1. Detailed Example (2/3)

After starting the simulation run, two new windows are shown, the Library Display and the Library performance.
Library Display shows the interaction between agents in the simulation while Library performance charts the 4 data variables calculated in the simulation (Table Occupation, Student satisfaction, Kicked Students, and Seated Students). More information on the functionality of these windows can be found on pages 6-7 of this document.

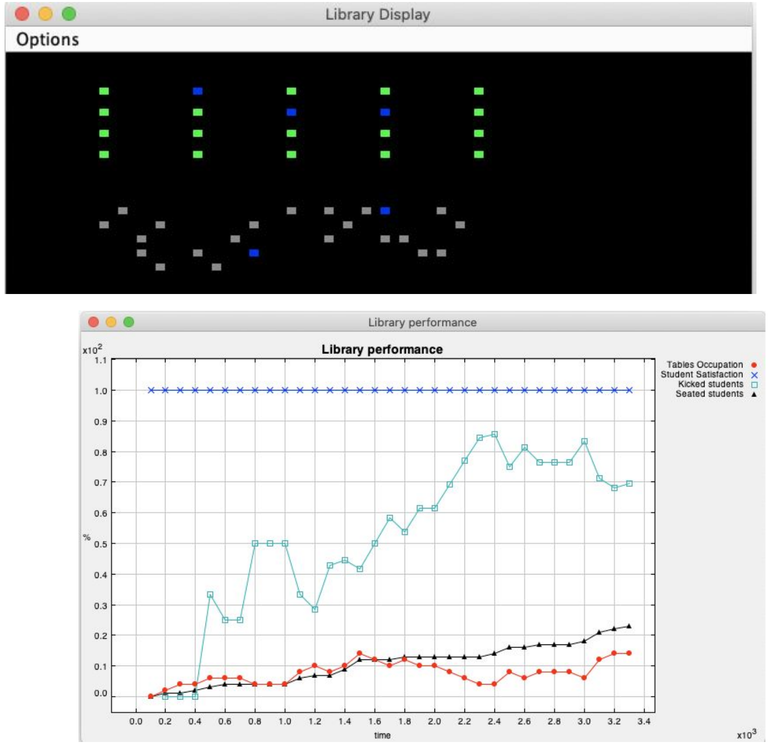


Fig 12: Graphical execution output

1. Detailed Example (3/3)

The simulation lasts the time stipulated in the WORKING_TIME variable in the input parameter window. Still, it can be paused (for data analysis), stopped, reset, and rerun using, at any time, the Repast interface commands.
In some cases, waiting for a full run of the simulation isn't necessary. For instance, if every student already visited the library and was eventually kicked, no more changes to the simulation can happen, as seen in the following figure.

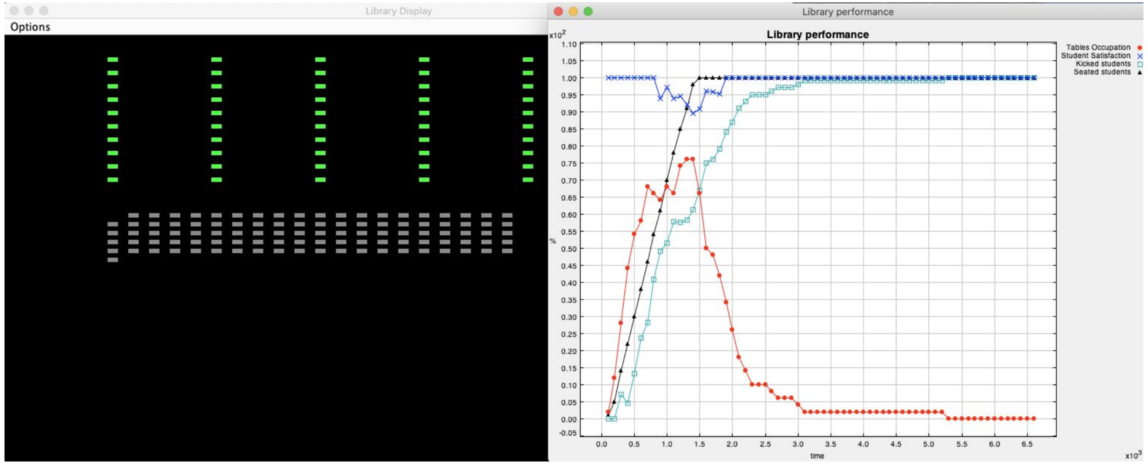


Fig 13: Simulated possible final state.