Intelligent Systems Engineering Project

Simulation for the Management of an Autonomous Warehouse

Daniele Commodaro
daniele.commodaro@studio.unibo.it

Department of Computer Science - Science and Engineering (DISI)
Alma Mater Studiorum - University of Bologna

Academic Year 2023/2024

- Intro:
  - Brief introduction to the project
  - Purpose and main objectives
- Project overview:
  - Brief description of the main features
- System architecture:
  - Main components
  - High-level diagrams
  - Interactions between agents

- Detail design:
  - Behavior of the agents
  - Implementation of main features
- Demo:
  - Short demo of the simulation in action
- Results and conclusions:
  - Main results obtained
  - Reflections on effectiveness and possible future improvements
  - Conclusions

Brief introduction to the project

- Multi-agent system based on Jason
- Dynamic and realistic simulation for the management of an autonomous warehouse
- Agent operations: cargo management, navigation system, battery charging, random breakdowns
- GUI for real-time viewing

Purpose and main objectives

- System development
- Definition of autonomous behaviors
- Creation of a realistic navigation system:
- Goods Management
- Warehouse modeling and implementation of a GUI

Brief description of the main features

- Load transportation
- Fault management
- Charging needs
- Dynamic navigation
- Warehouse dynamics

Main components

- Infrastructure
- Environment
  - Update agents' perceptions
  - Handle agents' actions
  - Provides auxiliary methods
- Warehouse model
  - Grid representation
  - State variables and constants

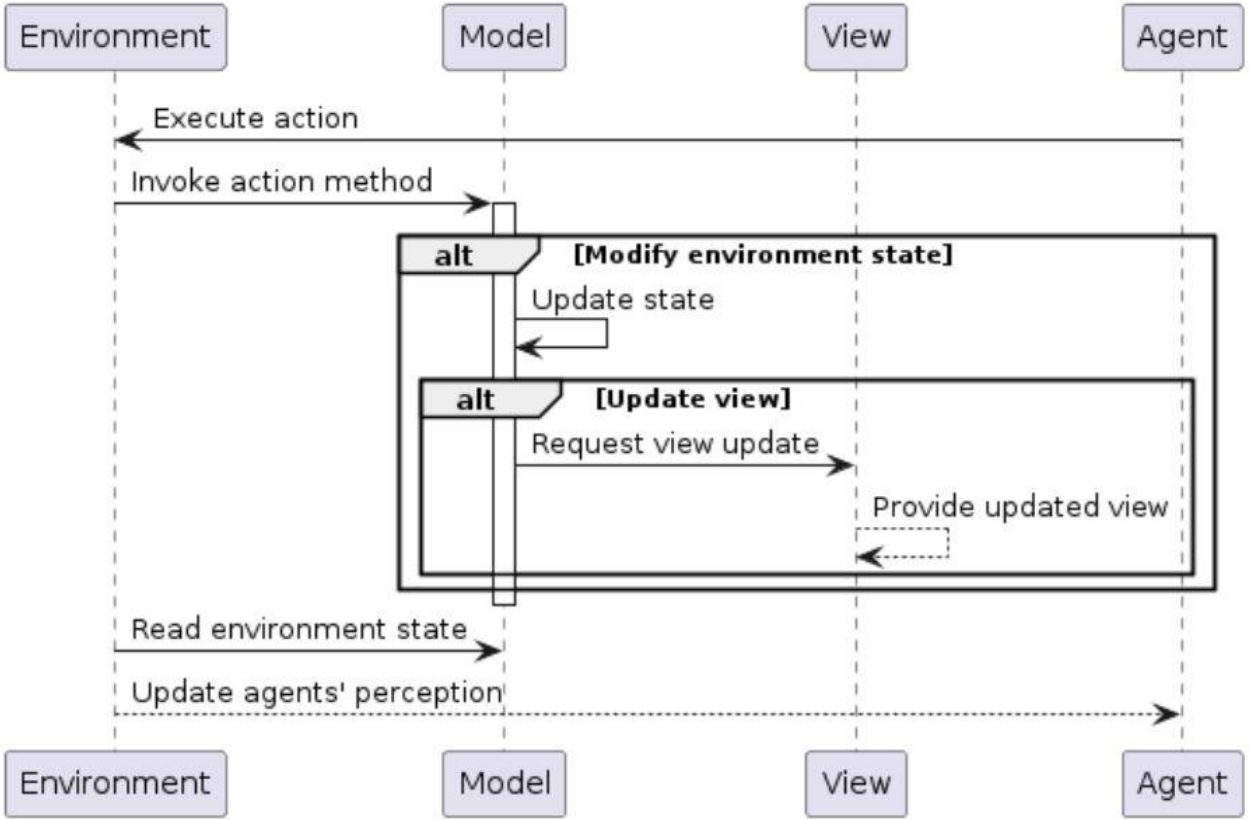
Main components

- Warehouse model
  - Initial placement of agents and environment objects
  - Methods for managing agent actions
  - Provide support methods
- GUI
  - Represents the environment
  - Sets the view attributes
  - Draws environment objects
  - Draws the agents

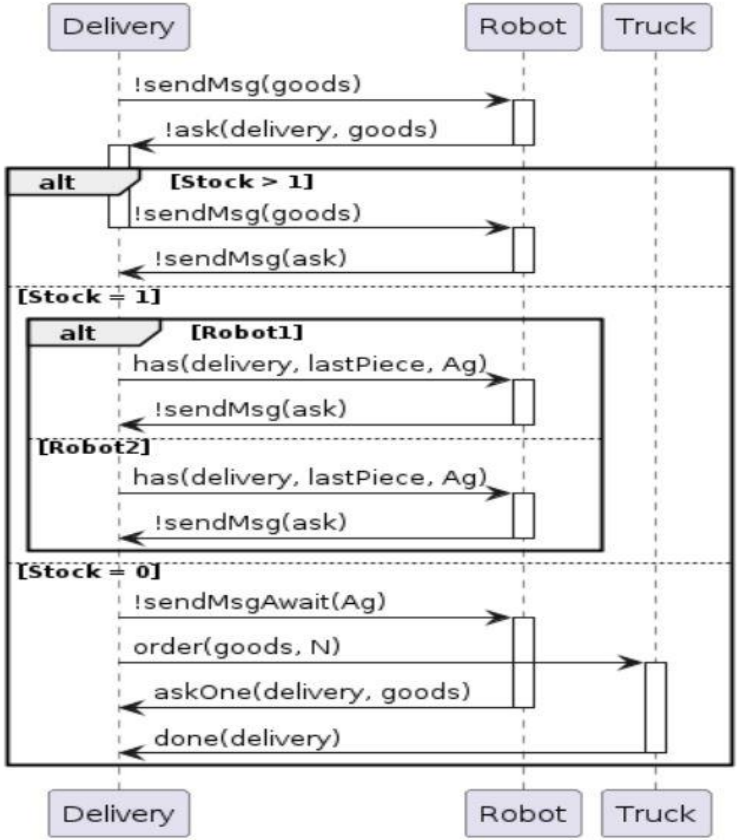
Main components

- Types of agents
  - Robot
  - Delivery
  - Truck
- Communication through messages
  - Use of semaphores

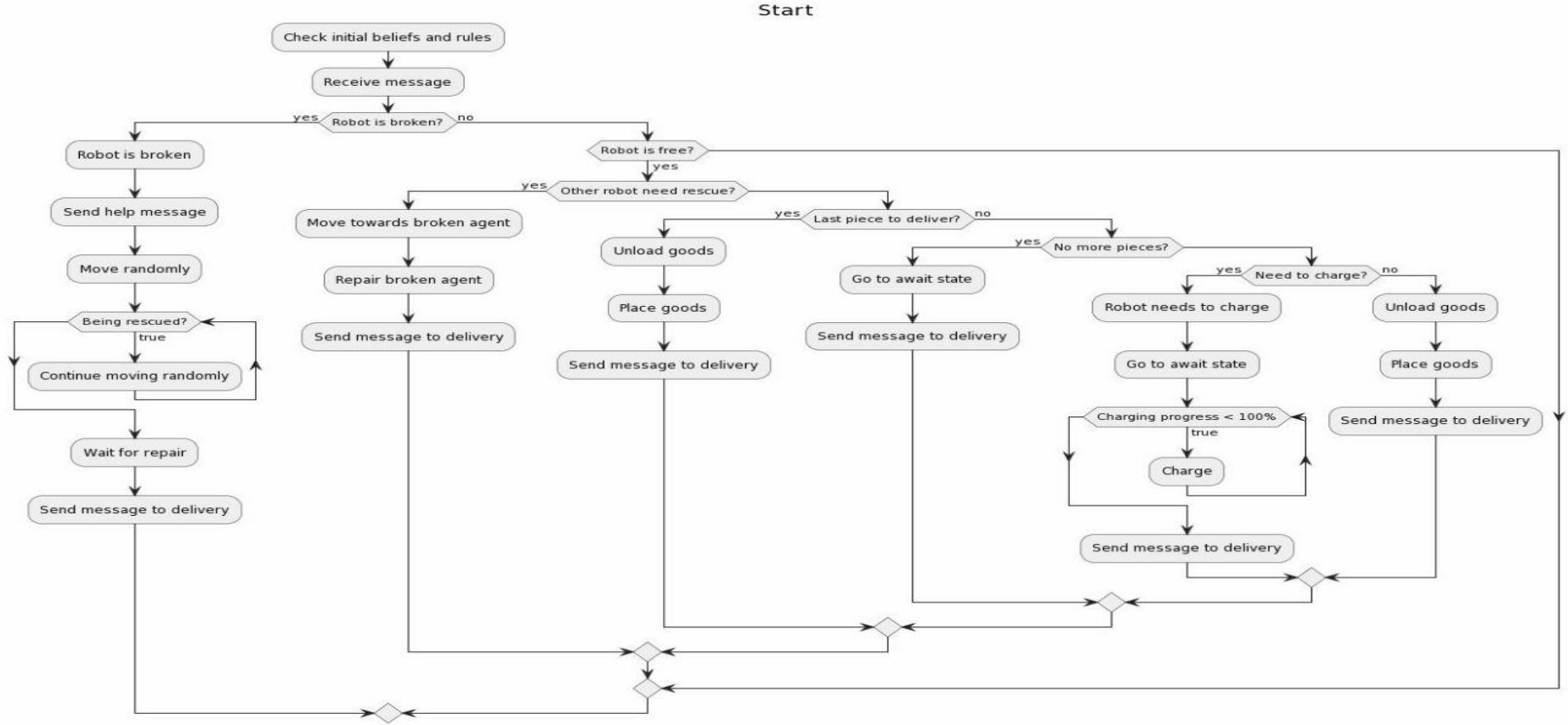
High-level diagrams



Interactions between agents



Behavior of the agents



Implementation of main features: cargo management

```
+!sendMsg(ask)
        : true
        <- .send(delivery, achieve, ask(delivery, goods)).

+!has(delivery, goods, Agent)
        : semaphore(free) & not need_charge(Agent) & not broken(_) & not rescuing(_)
    <- .print("Goal recived"); -semaphore(free); !at(Agent, delivery);
        unload(goods);
        utils.rand_int(R, 1, 4);
        !at(Agent, R);
        access(rack); place(R); free(rack); +semaphore(free); !sendMsg(ask);
        .date(YY, MM, DD);
        .time(HH, NN, SS);
        +consumed(YY, MM, DD, HH, NN, SS, Agent).
```

Implementation of main features: wait and movement

```
+!go(await, Agent)
   : semaphore(free) & not at(waitingZone) & not broken(_) & not rescuing(_)
   <- .print("I'm going to wait"); -rackType(0); -semaphore(free); !at(Agent, waitingZone).

+!at(Agent, P)
        : at(Agent, P)
        <- .wait(200).

+!at(Agent, P)
        : not at(Agent, P)
        <- move_towards(P); !at(Agent, P).
```

Implementation of main features: battery charging

```
limit(goods, 5).

need_charge(Ag) :-
        .date(YY, MM, DD) &
        .count(consumed(YY, MM, DD, _, _, _, B), QtdB) & limit(B, Limit) & QtdB > Limit.

+!has(delivery, goods, Agent)
   : semaphore(free) & need_charge(Agent) & not broken(_) & not rescuing(_)
   <- .print("I need to recharge");
      -consumed(_, _, _, _, _, _, Agent);
      -consumed(_, _, _, _, _, _, Agent);
      -consumed(_, _, _, _, _, _, Agent);
      -consumed(_, _, _, _, _, _, Agent);
      -consumed(_, _, _, _, _, _, Agent);
```

Implementation of main features: battery charging

```
!go(await, Agent); -semaphore(free);
charge(0); .wait(800);
charge(20); .wait(800);
charge(40); .wait(800);
charge(60); .wait(800);
charge(80); .wait(800);
charge(100); .wait(400);
charge(-1);
+semaphore(free);
!sendMsg(ask).
```

Implementation of main features: breakdown and rescue

```
+!has(delivery, goods, Agent)
    : semaphore(free) & broken(Rescuer)
    <- -semaphore(free); .print("BREAKDOWN"); show_broken(Agent);
       .send(Rescuer, achieve, provide_help(Agent)); !fault_resolved(Rescuer).

+!go(await, Agent)
    : semaphore(free) & broken(Rescuer)
    <- -rackType(0); -semaphore(free); .print("BREAKDOWN") show_broken(Agent);
       .send(Rescuer, achieve, provide_help(Agent)); !fault_resolved(Rescuer).

+!has(delivery, lastPiece, Agent)
    : semaphore(free) & broken(Rescuer)
    <- -semaphore(free); !unmanaged(last_piece); .print("BREAKDOWN"); show_broken(Agent);
       .send(Rescuer, achieve, provide_help(Agent)); !fault_resolved(Rescuer).
```

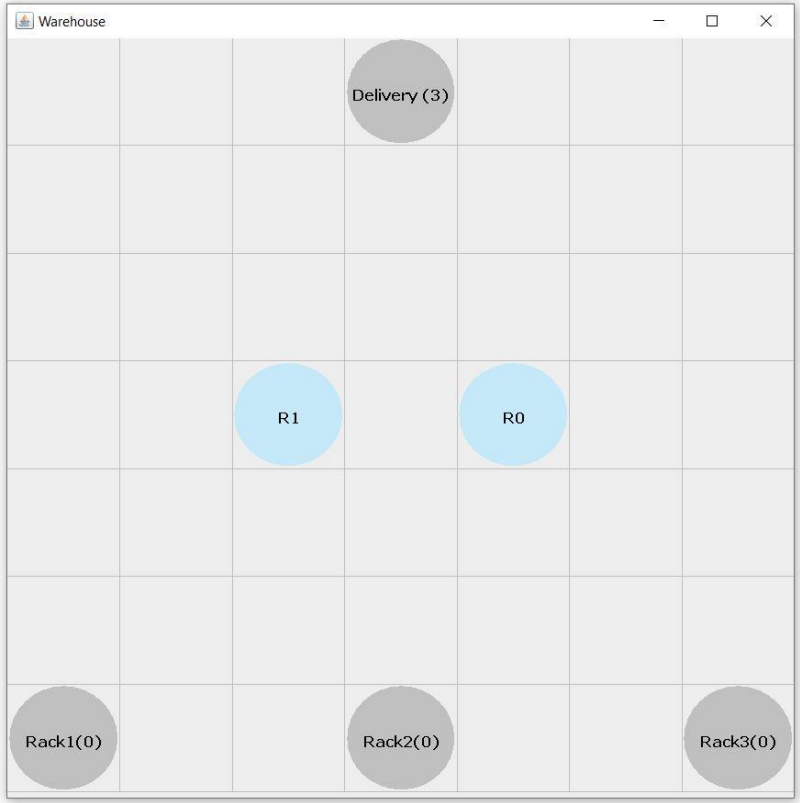
Implementation of main features: breakdown and rescue

```
+!has(delivery, goods, Agent)
    : semaphore(free) & need_charge(Agent) & broken(Rescuer)
    <- -semaphore(free); .print("BREAKDOWN"); show_broken(Agent); .send(Rescuer, achieve,
provide_help(Agent)); !fault_resolved(Rescuer).

+!fault_resolved(Rescuer)
        : not fault_resolved(Rescuer)
        <- move_towards(broken); .wait(500); !fault_resolved(Rescuer).

+!fault_resolved(Rescuer)
        : fault_resolved(Rescuer)
        <- .print("I have been reached for rescue"); .wait(4500); restore(env);
            +semaphore(free); !sendMsg(ask).
```

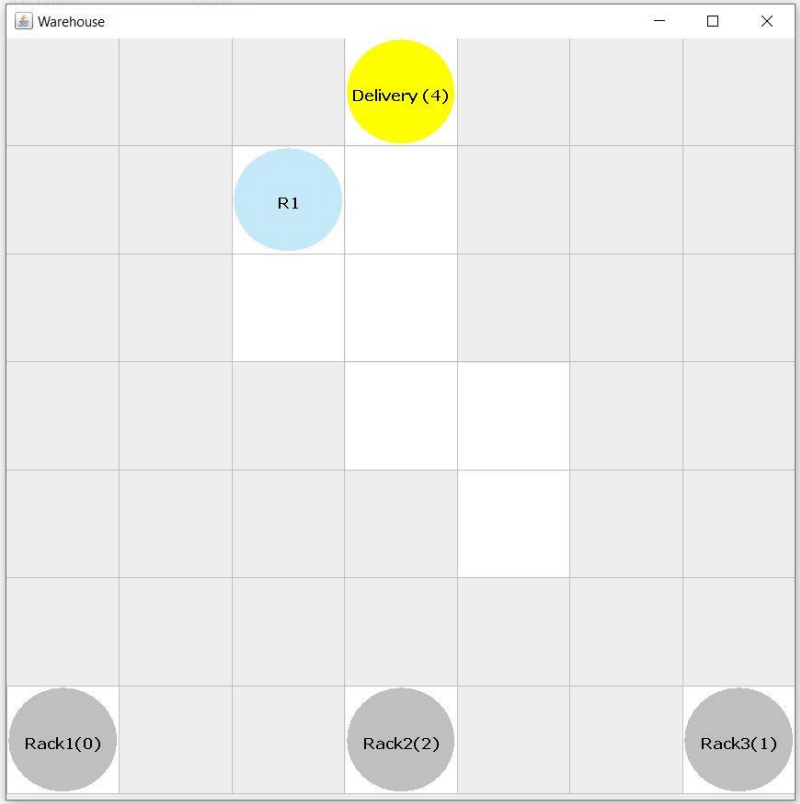
Implementation of main features: breakdown and rescue

```
+!has(delivery, goods, Ag)
    : semaphore(free) & rescuing(Agent)
    <- -semaphore(free); .print("I go to help"); !reach_robot(Agent).

. . .

+!reach_robot(Agent)
        : not reach_robot(Agent)
        <- catch(Agent); move_towards(Agent); !reach_robot(Agent).

+!reach_robot(Agent)
        : reach_robot(Agent)
        <- .print("Broken agent reached"); show_recovery(Agent); .wait(4500);
            -rescuing(Agent); restore(env); +semaphore(free); !sendMsg(ask).
```

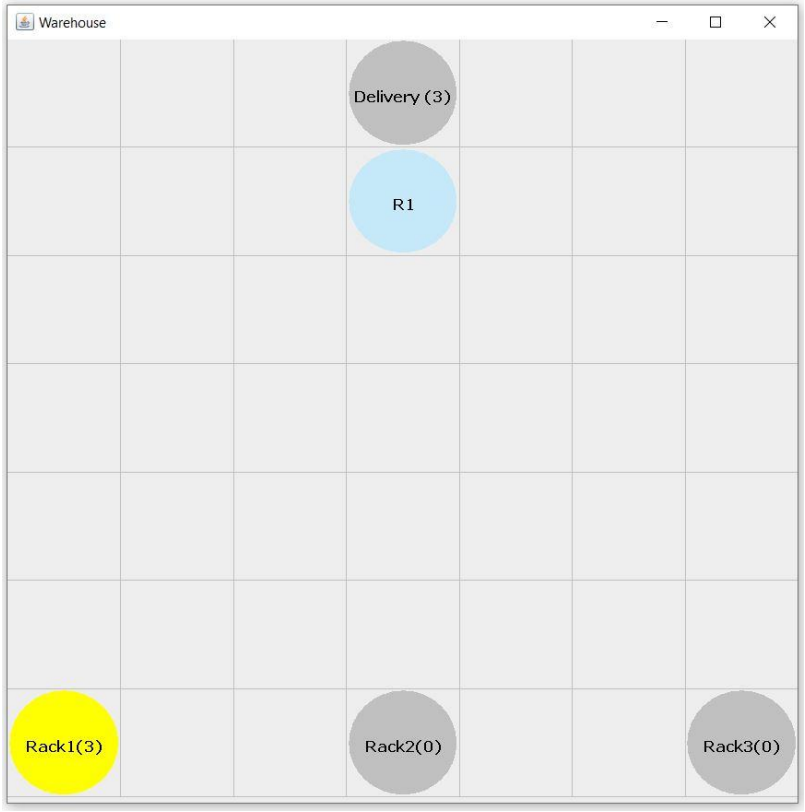
Short demo of the simulation in action



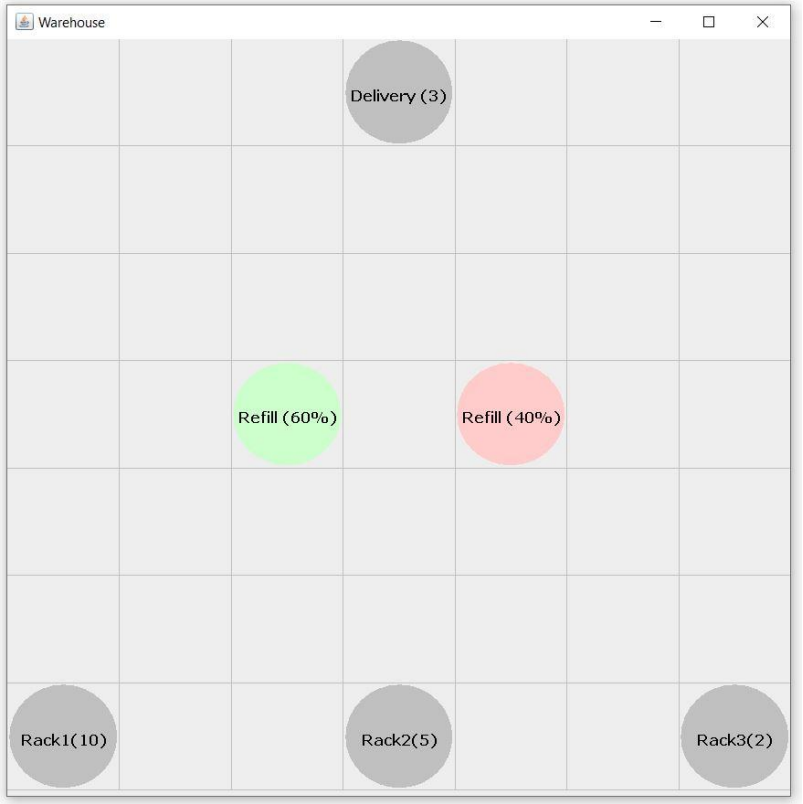
Short demo of the simulation in action



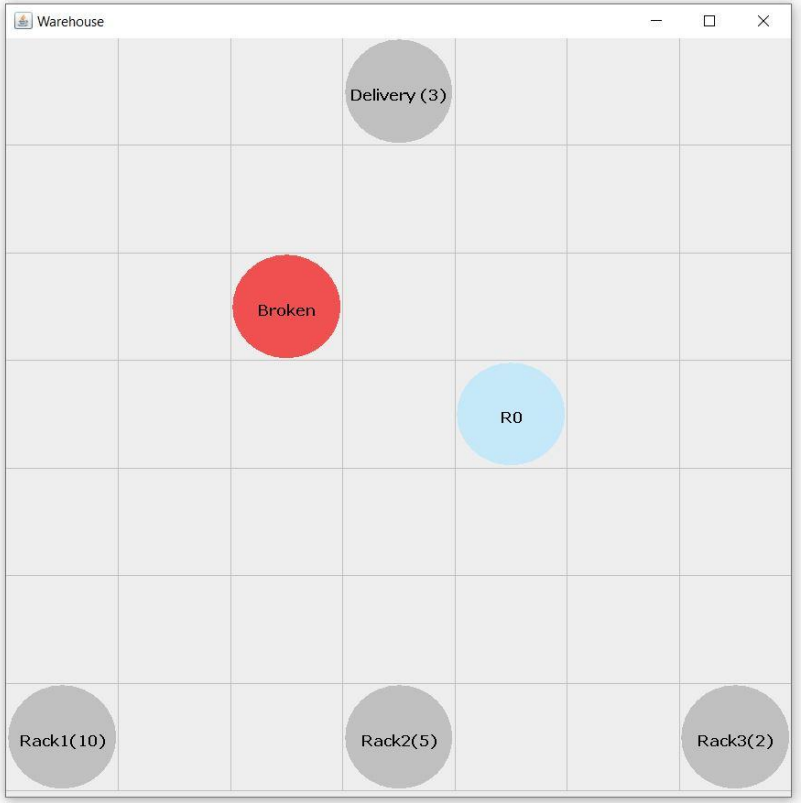
Short demo of the simulation in action



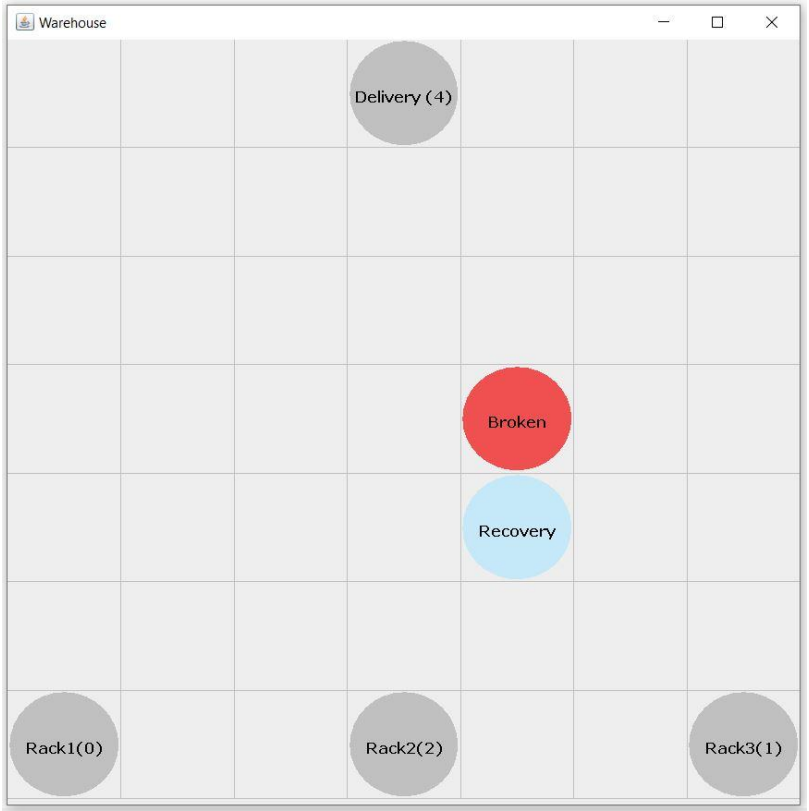
Short demo of the simulation in action



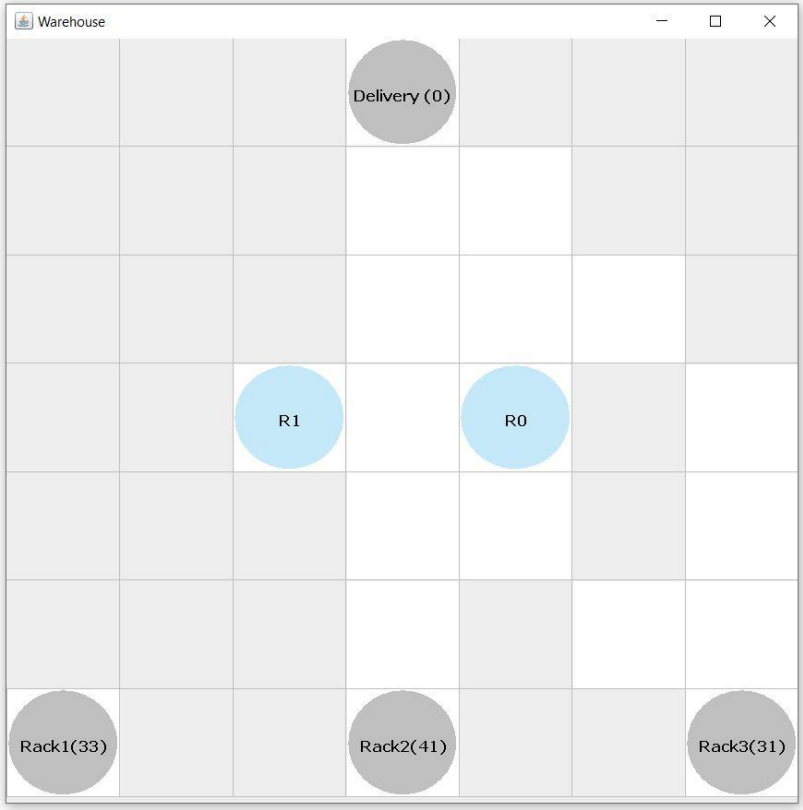
Short demo of the simulation in action



Short demo of the simulation in action



Short demo of the simulation in action



Main results, reflections and conclusions

- Main results:
  Analysis on the achievement of objectives
- Reflections
  - Difficulty detecting and correcting some problems
  - Some things could have been done better
- Conclusions
  - New skills were learned
  - Some features simple but effective
  - Learning took time and dedication

Thank you!