Simulation
Outline

- Introduction

- Multiagent-based Simulation approach

- Multiagent simulation platform

- Turtlekit
Introduction
Introduction

Simulation

- “The process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behaviour of the system or of evaluating various strategies (within the limits imposed by a criterion or a set of criteria) for the operation of the system.” (Shannon 1976)

- “Simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output” (Fishwick 1994)

Model

- “To an observer B, an object A is a model of an object A to the extent that B can use A to answer questions that interest him about A. “ [Minsky, 1965]

Simulation

Simulation supports

- Understanding, Exploration, Clarification
  - to understand the behavior of the reference system thanks to a model that is considered as a miniature reproduction of the reference system.

- Validation, Assessment, Verification
  - to test an hypothesis of the reference system, to validate or to certify the underlying theory.

- Control, action, control
  - to support a decision process or a control that will influence the state of the real reference system.

- Forecast, Prediction, Anticipation
  - to predict the possible evolutions of the reference system following evolutions or disturbances.

- Communication, Formation, Visualization
  - to show and share the model of the dynamic of the reference system.
Multiagent-based Simulation approach
Simulation

Methodologies

Fishwick

• The *model design* associates the real system with a representation of this system (*the model*).
  — This model is built from real observations (objective) or knowledge (subjective).
  — Data are usually formalized using formal semantics or mathematical logic to reduce ambiguities as much as possible.
  — It is then converted to algorithms,
• The *model execution* phase is the processing of the algorithm to produce numerical outputs.
• The *execution analysis phase*, deals with the analysis and confrontation of the results of the program with the behaviors observed in the model.


Simulation

Gilbert and Troitzsch

- Refine the Fishwick proposal with the addition of the model building phase
- The initial model is written into a computer program: the operational model
  - Adaptation to the simulator
  - There are both operational models and Simulators.
  - The differences between models introduces bias.


Multiagent-based Simulation approach

*bottom-up modeling approach*

- **Microscopic level**: simulation of the behavior of the components of the real system.
  - The components: the agents
  - Their relation: interaction and organization at a micro level.

- **Macroscopic level**: Observation, Analyze of properties of the multiagent system.

- **Example: Ants**
  - Micro level: ants are agents which put pheromones in the environment
  - Macro level: the shortest path
Multiagent-based Simulation approach overview

- A Multiagent-based Simulation (MABS) is a microscopic simulation model
  - A Multi-agent system: the multiagent model of an actual or theoretical physical system
  - Simulation: controls of the evolution of the model in time.

- Advantages
  - MABS supports
    - Multi-level modeling:
      - Different models of “individuals”: from simple entities to more complex ones.
      - Different levels of representation: “individuals” and “groups” within an unified conceptual framework.
    - The simulation of complex systems:
      - Structure preserving modeling of the simulated reality,
      - simulation of proactive behaviors,
      - Parallel computations,
      - Dynamic simulation scenarios

- Limits
  - Computation costs
Multiagent-based Simulation approach

**Contributors**

- **The Thematician (expert of the domain)**
  - *Role:* Defines the intention of the simulation process.
  - *Result:* the *domain model* which describes the multiagent model of the reality. The agents are informally associated to the components of the system and their relations are identified (interaction, organization).

- **The modeler**
  - *Role:* He translates the knowledge of the thematician.
  - *Result:* the *design model* where the agents are a refinement of the agents in the *domain model*. Their properties are expressed using concepts taken from multiagent domain (behavioral model, communications, …)

- **The Computer Scientist**
  - Role: He designs the operational model and writes the computer program.
  - Result: the computational system where agents are computational agents.

Multiagent simulation platform
The operational, simulated model can be executed on a

- Generic multiagent platforms
  - Advantage: the computer scientist knows his environment, i.e. the platform and the related multiagent model.
  - Limit: The platform must be adapted (or not) to support the simulation,
  - Example: JASON, JADE (Tapas, PlaSMA), MASH, MADKIT (Turtlekit)

- Generic simulation platforms
  - Advantage: the computer scientist can use the same environment for different design models.
  - Limit: a new operational model has to be built for each new simulations.
  - Example: MASON, SWARM, GAMA, CORMAS, TURTLEKIT, REPAST, NETLOGO, …

- specialized simulation platforms
  - Advantage: some parts of the operational model can be already available.
  - Limit: adaptation to a new platform.
  - Example (traffic simulation platform): Archisim, MATSim, MITSIMlab, …
Components of a multiagent platform

A scheduler
- A temporal model: discrete, continuous, event
- A scheduling policy
  - Synchronization of agent evolution
  - Simulation of the simultaneity
Scheduler

Temporal models

- **Discrete time model**
  - Time advances in discrete step, which are integer multiples of some basic period such as 1 second, 1 day or …
  - If the state at time $t$ is $q$ and the input time $t$ is $x$, then the state at time $t+1$ will be $\delta(q,x)$ and the output $y$ at time $t$ will be $\lambda(q,x)$
    - $\delta$ is called the state transition function
    - $\lambda$ is called the output function

- **Discrete Time Simulation**

```plaintext
T_i = t_i, T_f = t_f
x(0) = v_0, ..., x(9) = v_9
q(0) = q_0
t = T_i
while (t <= T_f) {
    y(t) = \lambda(q(t),x(t))
    q(t+1) = \delta(q(t),x(t))
    t = t + 1
}
```

Scheduler

Temporal models

- **Discrete event models**
  - appropriate for those systems for which changes in system state occur only at discrete points in time.
  - A discrete point in time is called an event.

- **Discrete Event Simulation**
  1. Initialize the state variables
  2. Initialize the ‘collection of pending events’
  3. Initialize the simulation clock
  4. while (there are pending events to be handled){
     Remove the pending event (E) with the smallest timestamp (t)
     Set simulation clock to that time t
     Execute the event handler for event E
  }
Scheduler

Multiagent scheduler

scheduling algorithm based on a continuous temporal model

T duration of the simulation
time = System.time();
T = time + T
Agents= {agents of the simulation};
For (a: agent)
   activate(a)
while (time < T)
   time = System.time();

scheduling algorithm based on a discrete temporal model

T duration of the simulation
time = 0;
Agents= {agents of the simulation};
while (time < T){
   For (a: Agents){
      activate(a)
      a.ContextComputation()
      a.DecisionProcess()
      a.actionProcessing()
   }
   time++
}
Simultaneity problem

**Scheduler**

- **Discrete simulation**
  - Let \( t \) be the simulation time value and \( a_i(t,q(t)) \) the action of the \( i^{th} \) agent following the current state of the simulated system \( q(t) \).
  - How to ensure that \( q(t) \) will be the same for \( a_i \) and \( a_{i+1} \) since \( a_i(t,q(t)) \) modifies the current state.

Dealing with Multi-Agent Coordination by Anticipation: Application to the Traffic Simulation at Junctions.
A Doniec, S Espié, R Mandiau, S Piechowiak - EUMAS, 2005
Simultaneity problem

**Solution**

- **No solution**
  - The most current solution,
  - The action of an agent should not change the world in an important way. The micro coordination problems resulting of the scheduling process are not taken into account.
  - The consequences of this choice have to be taken into account

- **The scheduling policy**
  - The activation order of the agents is randomized
  - If the number of agents and simulation steps are important then no agent should be advantaged.
  - If the simulation must be replayed, the random process has to be taken into account by the simulation model.

- **A dedicated mechanism**
  - The agents are activated in the same simulation state and the antagonism between their action is resolved by a decision process.
  - Influence / reaction model: The agents do not directly act in the simulation but emit influences that are validated by the decision process.

Turtlekit
Illustrative Example

Prey and Predators

• A multiagent model
  – Environment: a grid
  – Prey: reactive agents who avoid the predators
  – Predator: communicative agents who coordinate to catch the preys
  – When three predators are around a prey, this last one die

• Simulation
  – A scheduling process
    • Temporal model
    • Activation process
Turtlekit overview

- Plugin of the Madkit platform dedicated to the simulation
  - Supports the Madkit organizational model: the AGR model
    - Interaction are regulated by the organizational model
      - The communications are regulated following the organizational model
      - The perception can be implemented following the organizational model
Supports the simulation of heterogeneous multiagent model

- The superclass *AbstractAgent* contains the methods for the
  - Management of the life cycle:
    - activate(); end(); launchAgent(...); killAgent(...)
  - Communication management:
    - broadcastMessage(...); sendMessage(...); nextMessage();
    - isMessageBoxEmpty(); receiveMessage(Message m);
  - Organization management
    - createGroup(); leaveGroup(); requestRole(); getRoles(); isGroup(...);
    - getAgentsWithRole()

**Example**

```java
def setup():
    playRole("predator");
    ACLMessage m = new ACLMessage("INFORM","I’m a new predator");
    broadcastMessage("Turtlekit","HUNT","predator",m);
```

Class Agent

- Inherits of the superclass AbstractAgent
- Implements the Runnable Interface
  - Methods “to control” the its thread
    - exitImmediatelyOnKill() ; live() ; pause(int t) ; run().
  - Additional methods for communications
    - waitNextMessage() ; waitNextMessage(long timeout)

Example

```java
public void live() {
    while (true) {
        Message m = waitNextMessage();
        if (m instanceof ACLMessage)
            handleMessage((ACLMessage)m);
    }
}
```
private boolean caught(){
    int cpt=0;
    for(int i=-1;i<=1;i++)
        for(int j=-1;j<=1;j++)
            if (! (i==0 && j==0) ){
                Turtle[] tur = turtlesAt(i,j);
                if (tur!= null && tur.length>0 &&
                    tur[0].isPlayingRole("predator")) cpt++;
            }
    if (cpt>3) return true;
    return false;
}

public void setup(){
    playRole("predator");
    randomHeading();
    setColor(Color.red);
    if (countTurtlesHere()>0)
        fd(1);
}

### Turtlekit overview

#### Class Turtle
- Inherits of the superclass AbstractAgent
- Do not implements the Runnable interface
- Additional methods
  - Related to the simulation process
    - setup, activate, end
  - A turtle is a situated agent, he has methods to
    - To be located in the environment
      - setX, xcor, dx, distance, towards, getHeading,
    - To move in the environment
      - moveto, fd, home, turnLeft, turnRight
    - To perceive the environment
      - countTurtlesAt, countTurtleHere, turtlesAt, turtlesHere

#### Example

private boolean caught(){
    int cpt=0;
    for(int i=-1;i<=1;i++)
        for(int j=-1;j<=1;j++)
            if (! (i==0 && j==0) ){
                Turtle[] tur = turtlesAt(i,j);
                if (tur!= null && tur.length>0 &&
                    tur[0].isPlayingRole("predator")) cpt++;
            }
    if (cpt>3) return true;
    return false;
}

```java
public void setup(){
    playRole("predator");
    randomHeading();
    setColor(Color.red);
    if (countTurtlesHere()>0)
        fd(1);
}
```
Pseudo activation algorithm

time = 0; Turtle = {turtles of the simulation}; T duration of the simulation
While (time < T)
   For (t: Turtle)
      currentAction = scheduler.getCurrentActionTurtle(t)
      nextAction = activate(t, currentAction)
      scheduler.setCurrentActionTurtle(t, nextAction)
      time++
Message
- getCreationDate; getReceiver; getSender

ActMessage
- getAction; getContent; getFieldValue;
  getInReplyTo; getObject; setContent; setField;
  setInReplyTo; setObject

ACLMessage
- getAct, getPerformative; setPerformative;
  getReceivers; removeReceiver; clearAllReceiver.