



Palteformes multi-agents

Syma – cursus SCIA

Julien Saunier - julien.saunier@ifsttar.fr



Plan

- Madkit/Magique: structure organisationnelle
- Jade: agents cognitifs communicants
- Netlogo, Cormas: simulation



Madkit

- Plateforme multi-agents générique
- Deux types d'agents:
 - Threadés
 - Turtles
- Gestion des messages et des concepts AGR
- Gestion du cycle de vie des agents
- Ecrite en Java, architecture modulaire par plug-ins
- Distribuée



Le modèle Agent / Groupe / Rôle

– Agent:

- *Une entité autonome et communicante*
 - Joue des *rôles* dans des *groupes*. Un agent peut avoir *plusieurs rôles* et être membre de *plusieurs groupes*.



– Rôle:

- *la représentation abstraite de la fonction d'un agent.*
 - Les rôles sont *locaux* aux groupes. Un rôle peut être joué par *plusieurs agents*

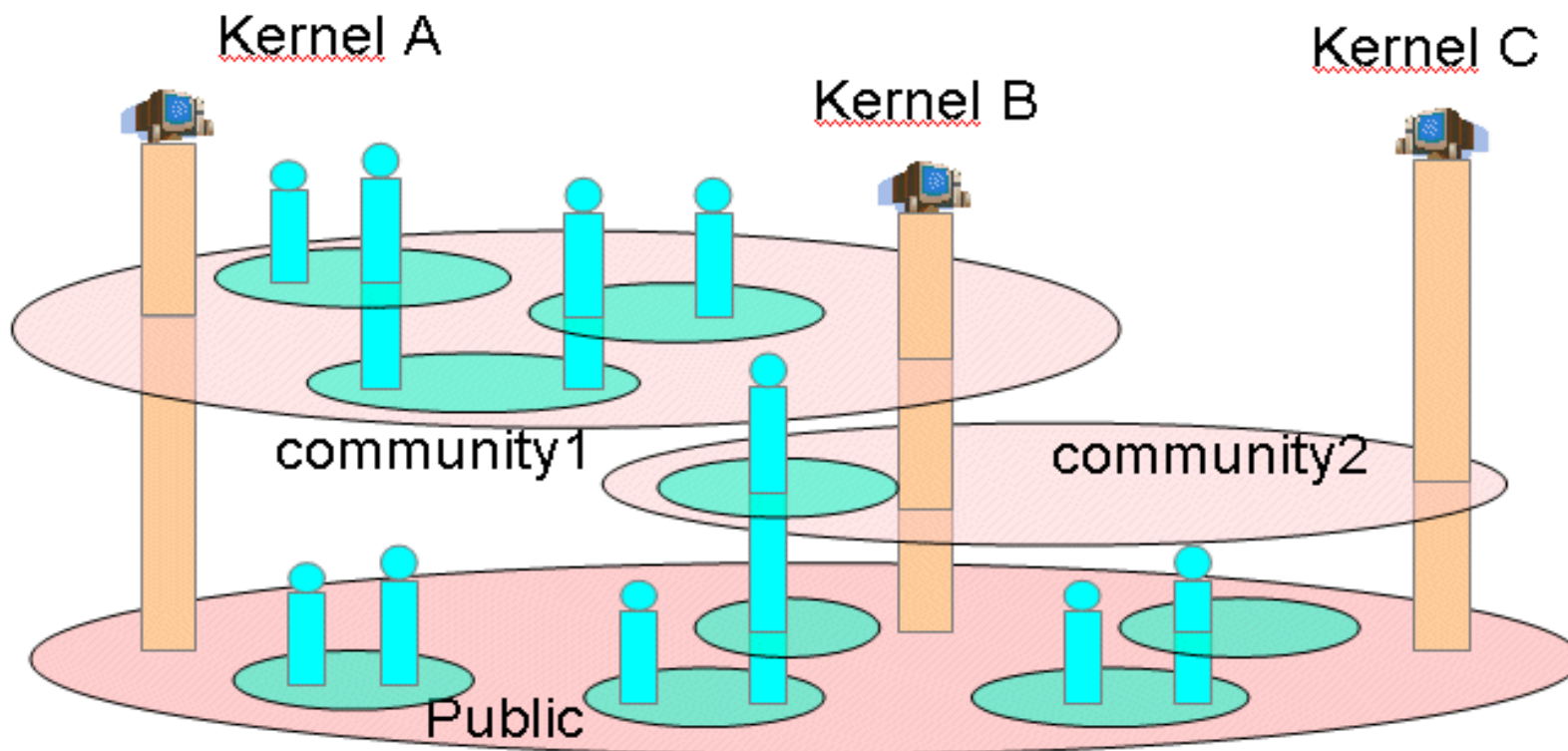
– Groupe:

- *un ensemble d'agents partageant une caractéristique*
 - Deux agents peuvent communiquer que s'ils sont membres du même groupe.



Madkit

Représentation du modèle AGR



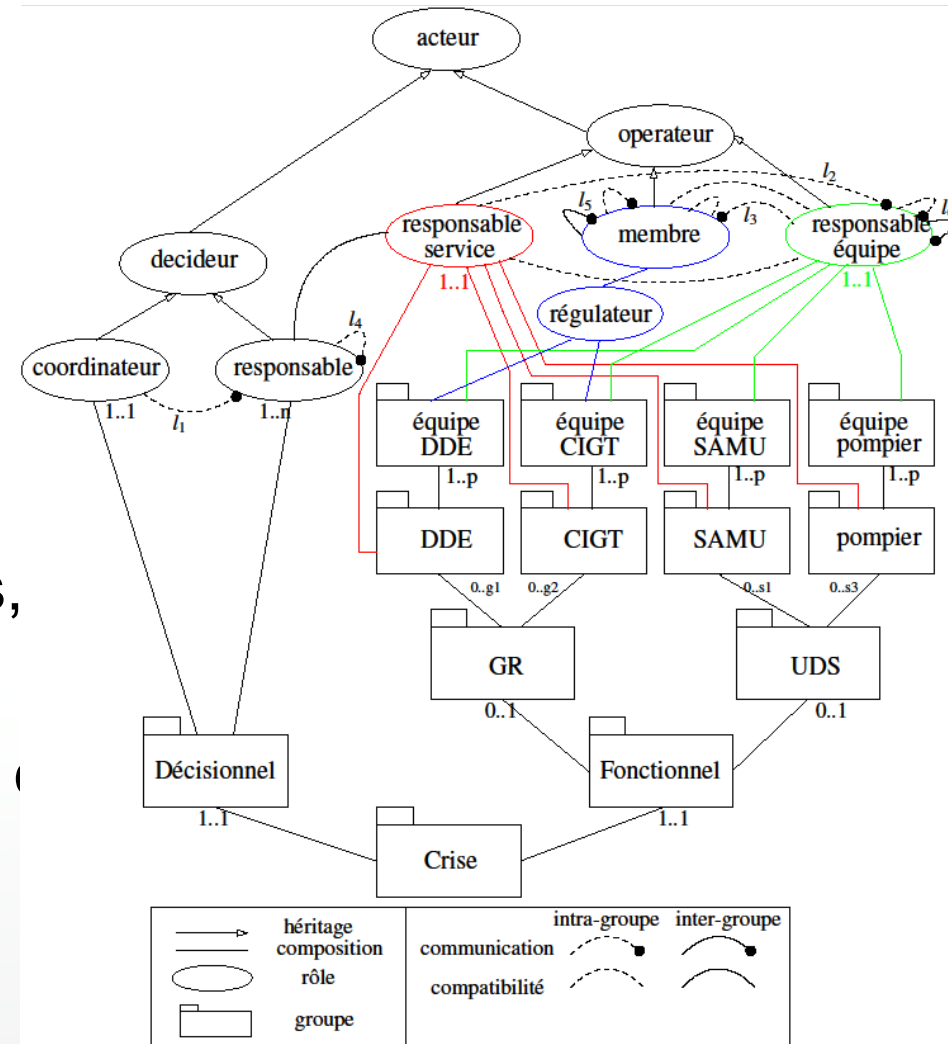
Modèle sous-jacent

■ Spécification Structurelle :

$\langle R, \subset, rg \rangle$

- R : ensemble des identifiants des rôles,
- \subset : relation d'héritage sur les rôles,
- rg : spécification du groupe racine (l'organisation).

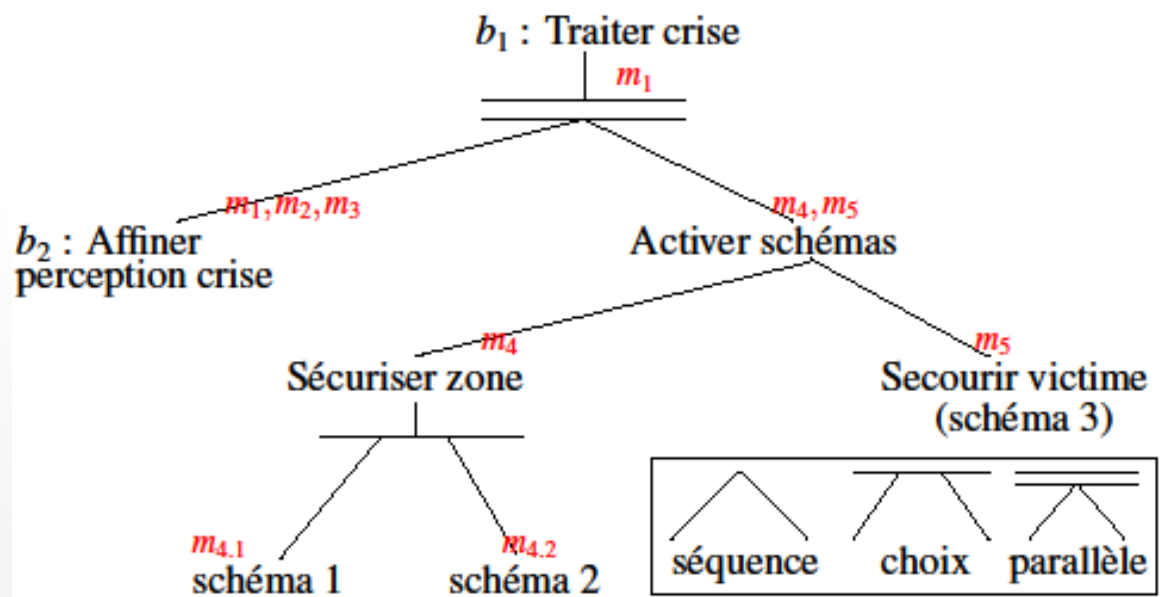
Moise



Modèle sous-jacent

Moise

- **Spécification Fonctionnelle : $\langle M, G, S \rangle$**
 - M : ensemble des missions,
 - G : ensemble des buts collectifs et individuels à satisfaire,
 - S : structurations arborescentes de ces buts en plans.





Exemple: PingPong

- 2 agents du groupe ping-pong ont le même rôle, joueur.
- Un des deux fonde le groupe
- Un agent appartient toujours à un groupe, même seul
- Les agents communique par messages



Exemple: PingPong

```
import madkit.kernel.*;
import madkit.lib.messages.*;

public class PingPong extends Agent
{
  AgentAddress other = null;
  boolean creator=false;
  ...
}
```



Exemple: PingPong

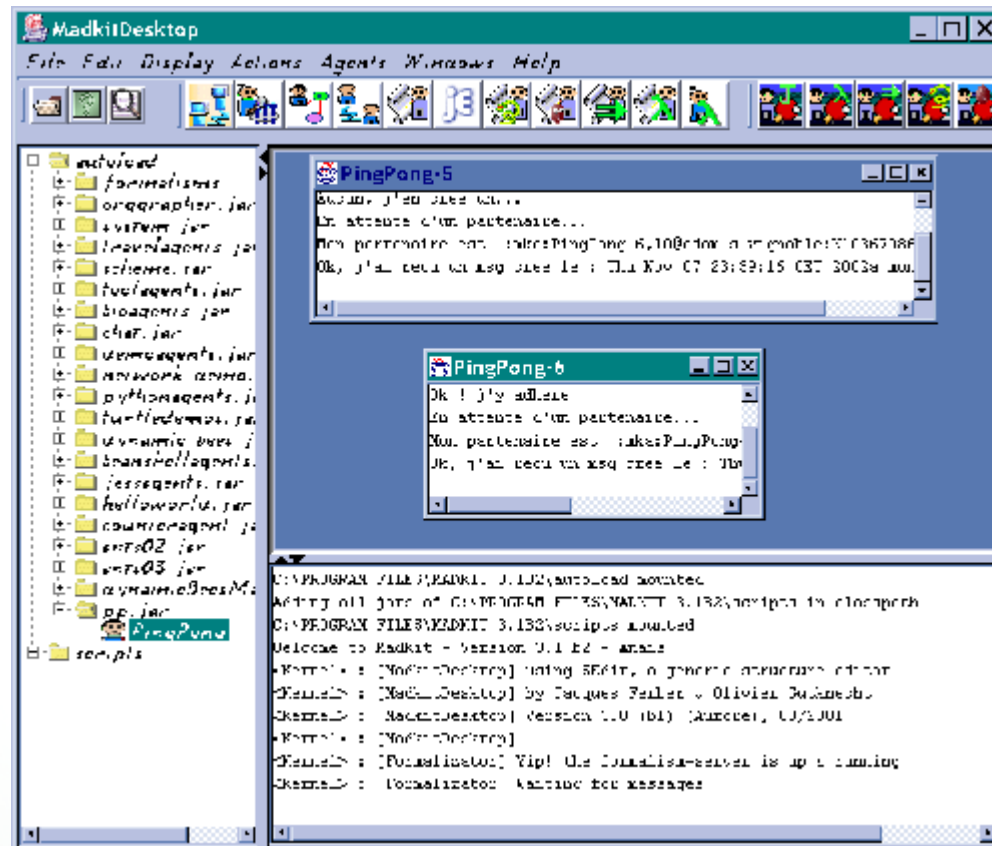
```
...
public void activate()
{
    println("PingPong agent active");
    println("recherche le groupe pingpong...");
    if (isGroup("pingpong"))
    {
        println ("Ok ! j'y adhere");
        creator=false;
    }
    else
    {
        println ("Aucun, j'en cree un...");
        createGroup(true,"pingpong",null,null);
        creator=true;
    }
    requestRole("pingpong", "player", null);
}
```



Exemple: PingPong

```
// Si je ne suis pas le createur du groupe, j'envoie la la balle en premier
if (! creator)
sendMessage(other, new StringMessage("Ball"));
for (int i = 5; i > 0; i--)
{
Message m = waitNextMessage();
StringMessage ans = (StringMessage) m;
println("Ok, j'ai reçu un msg cree le : " +
m.getCreationDate() + "a mon tour ...");
pause(1000);
sendMessage(other, new StringMessage(ans.getString()));
}
}
public void end()
{
println("Bye Bye !!");
println("Fin de agent PingPong
");
}
}
```

Exemple: PingPong



MAGIQUE

Multi-AGent hiérarchIQUE

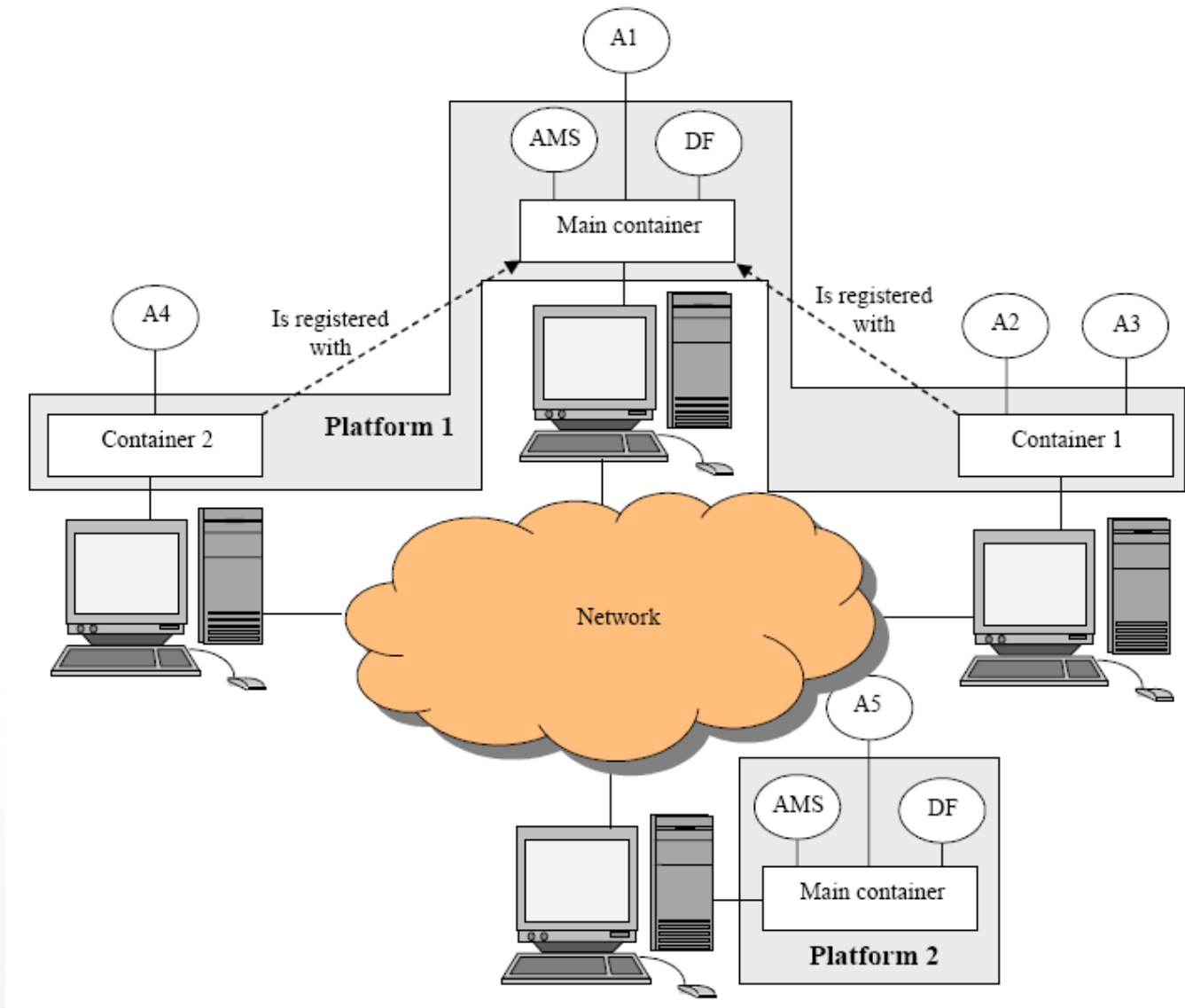
The screenshot displays the Multi-Agents System Throwing interface. The main window shows a hierarchical tree of agents:

- Super Agent (lambik.lif.fr) is the root.
- Collatz Agent (lambik.lif.fr) and Math Agent (lambik.lif.fr) are children of the Super Agent.
- Parity Agent (lambik.lif.fr), Divider Agent (lambik.lif.fr), Multiplier Agent (lambik.lif.fr), and Adder Agent (lambik.lif.fr) are children of the Math Agent.

On the left, the 'Computers' panel shows a file system with a folder named 'fr'. Below it, the 'Agents Properties' panel is visible, with tabs for 'Properties' and 'Skills', and buttons for 'add' and 'remove'. On the right, the 'Agents Classes' panel shows a file explorer view of the classpath, listing files like 'doc', 'src', and 'magique.jar' with their corresponding class names.

Agent Cognitif

Plateforme Jade





JADE

Description

- Services :
 - The Dummy Agent,
 - The Sniffer Agent,
 - The Introspector agent,
 - Remote Management Agent,
 - The DF GUI,
 - The LogManagerAgent,
 - The SocketProxyAgent agent

Jade

Service

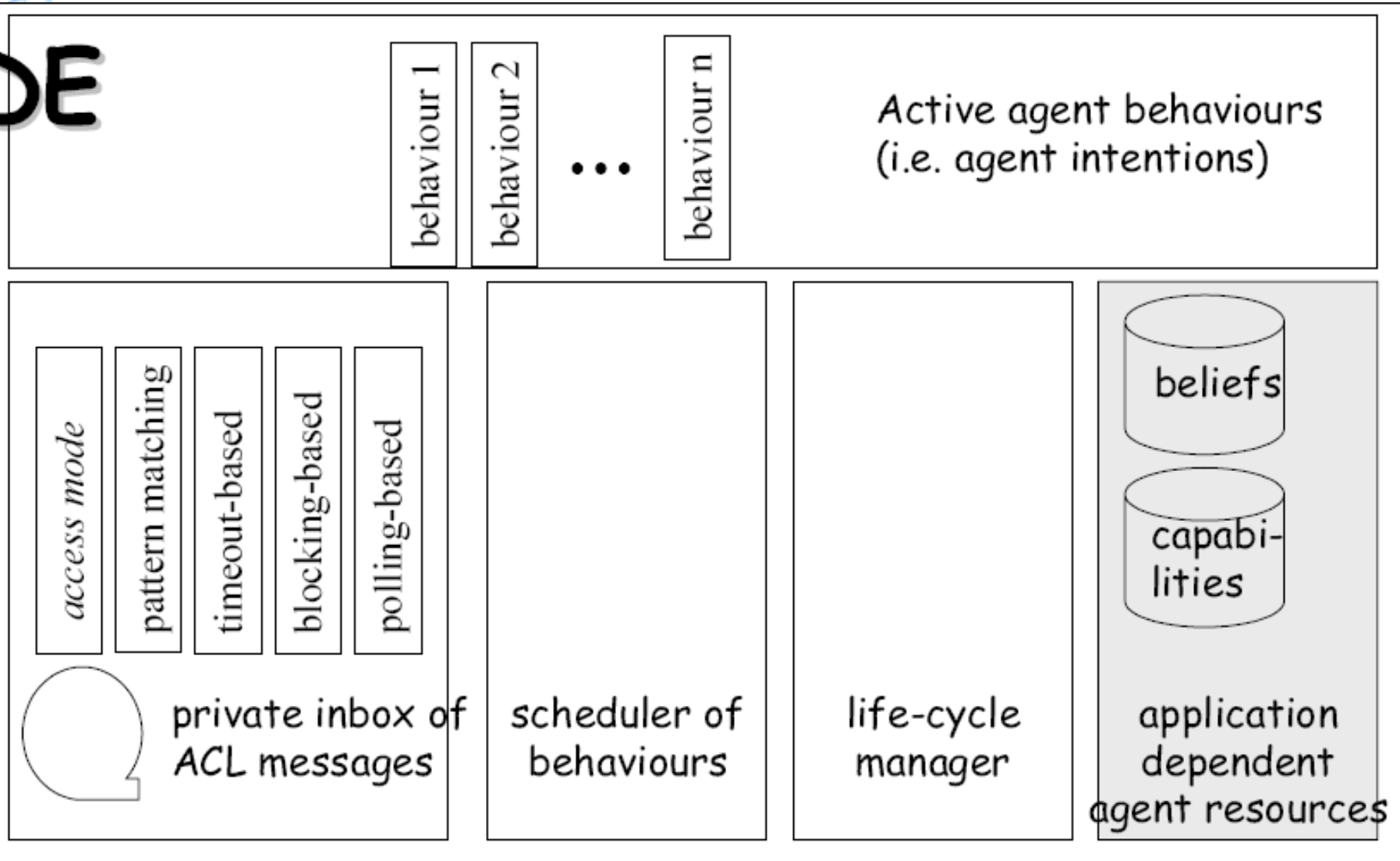
The screenshot shows the Jade GUI interface. The window title is "da0:DummyAgent". The interface is divided into several sections:

- General**: Contains tabs for "General", "Current message", and "Queued message".
- Toolbar**: Includes buttons for "Reset", "Send", "Open", "Save", "Open queue", "Save queue", "Set as current", "Reply", "View", and "Delete".
- Current message**: A detailed view of the selected message with the following fields:
 - Sender: da0
 - Receiver: Paula
 - Communication type: inform
 - Content: true
 - Language: SLO
 - Ontology: (empty)
 - Protocol: Null
 - Conversation: Conversation-1
 - In-reply-to: (empty)
 - Reply-with: Follow-Conv-1
 - Reply-by: Set 20000317T172351000
 - Envelope: (empty)
- Message Log**: A list of messages with columns for date, time, and type. The messages are:

Date	Time	Type	Direction	From	To
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:25	PM: INFORM	recv	Paula	Paula
2/16/00	5:25	PM: INFORM	sent	Paula	Paula
2/16/00	5:24	PM: INFORM	recv	Paula	Paula
2/16/00	5:24	PM: INFORM	sent	Paula	Paula

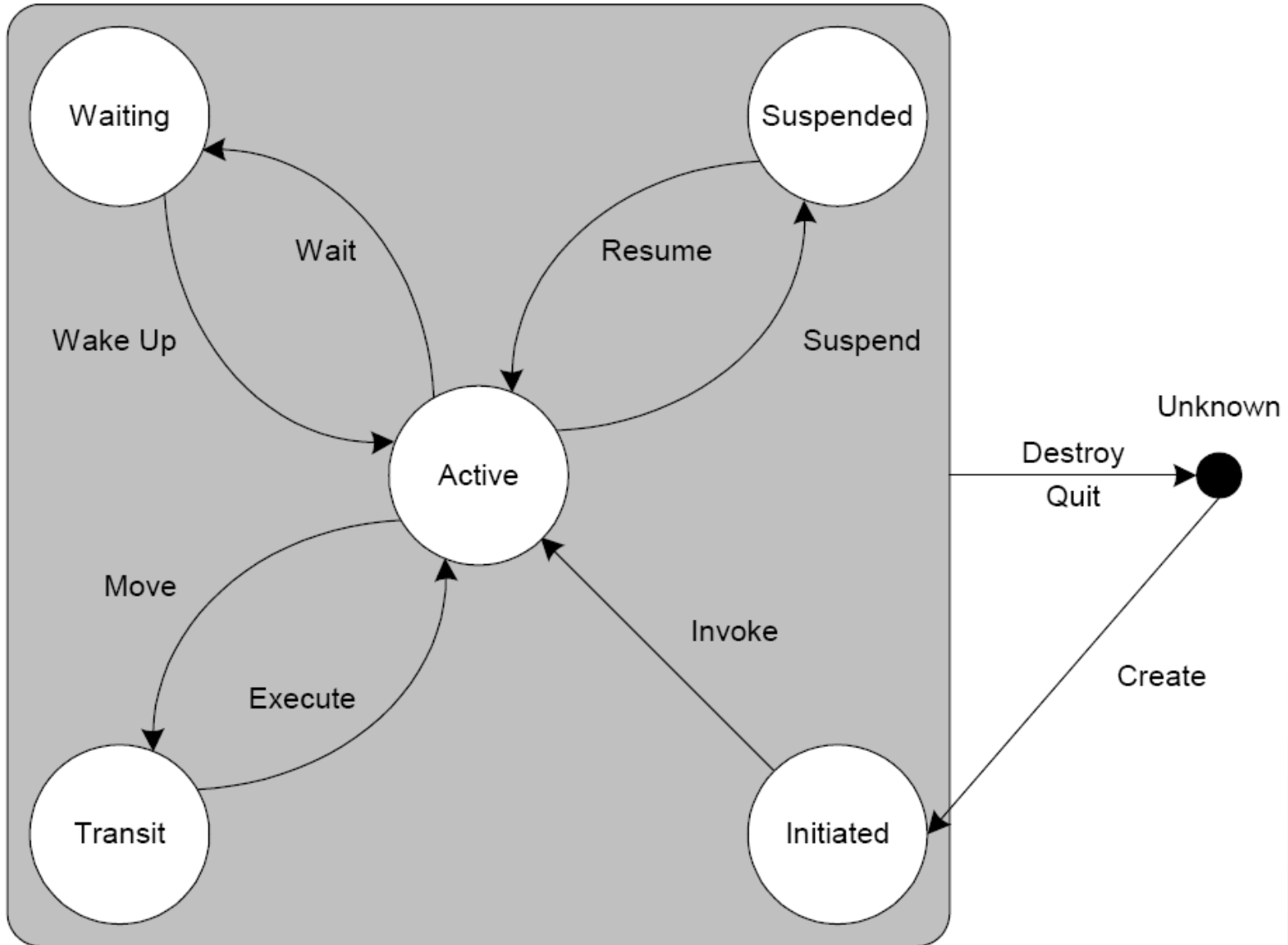
Agent architecture

DE

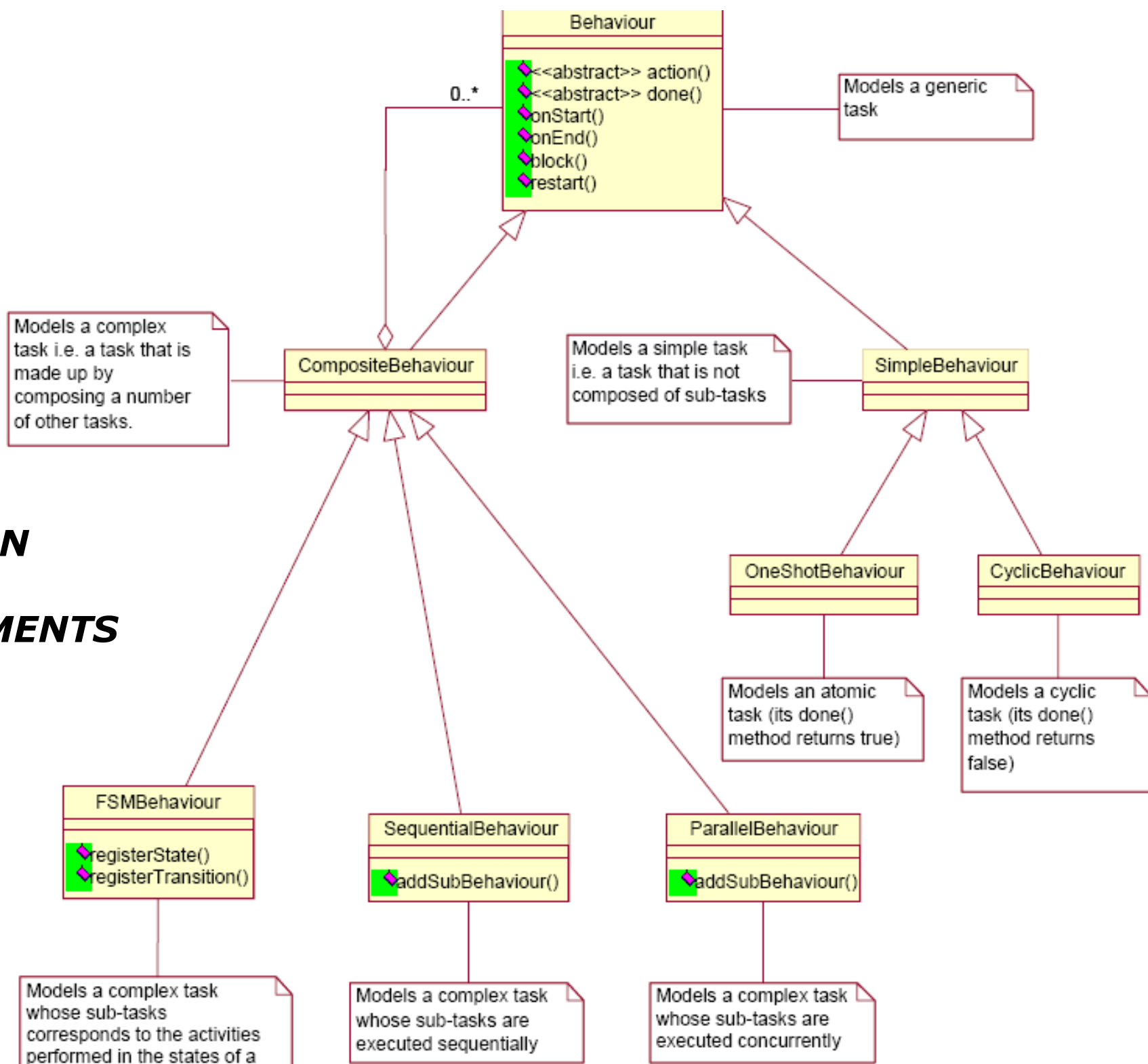


Agent

Cycle de vie



GESTION des COMPORTEMENTS





Simulation

- Plateformes Netlogo et variantes langagières
- Plateformes CORMAS (agents situés), GAMA...

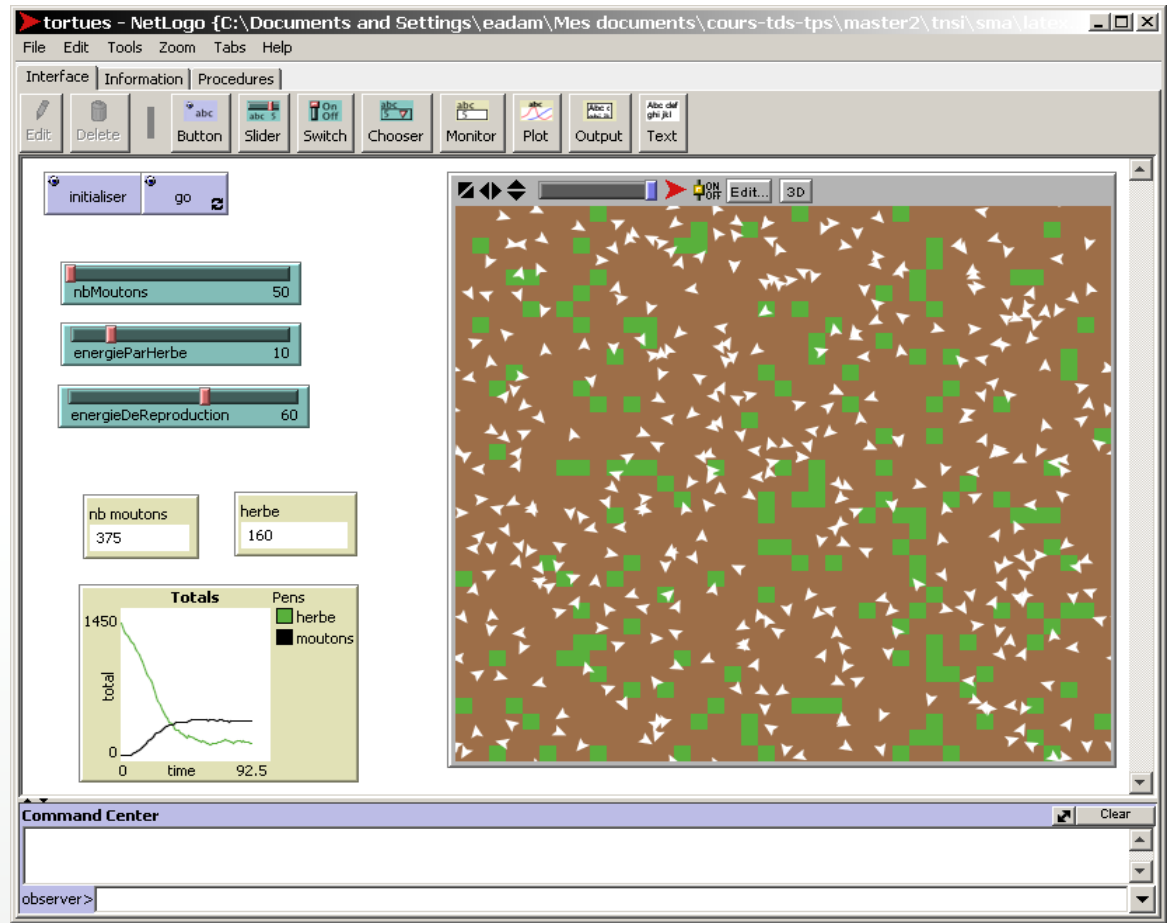


Netlogo

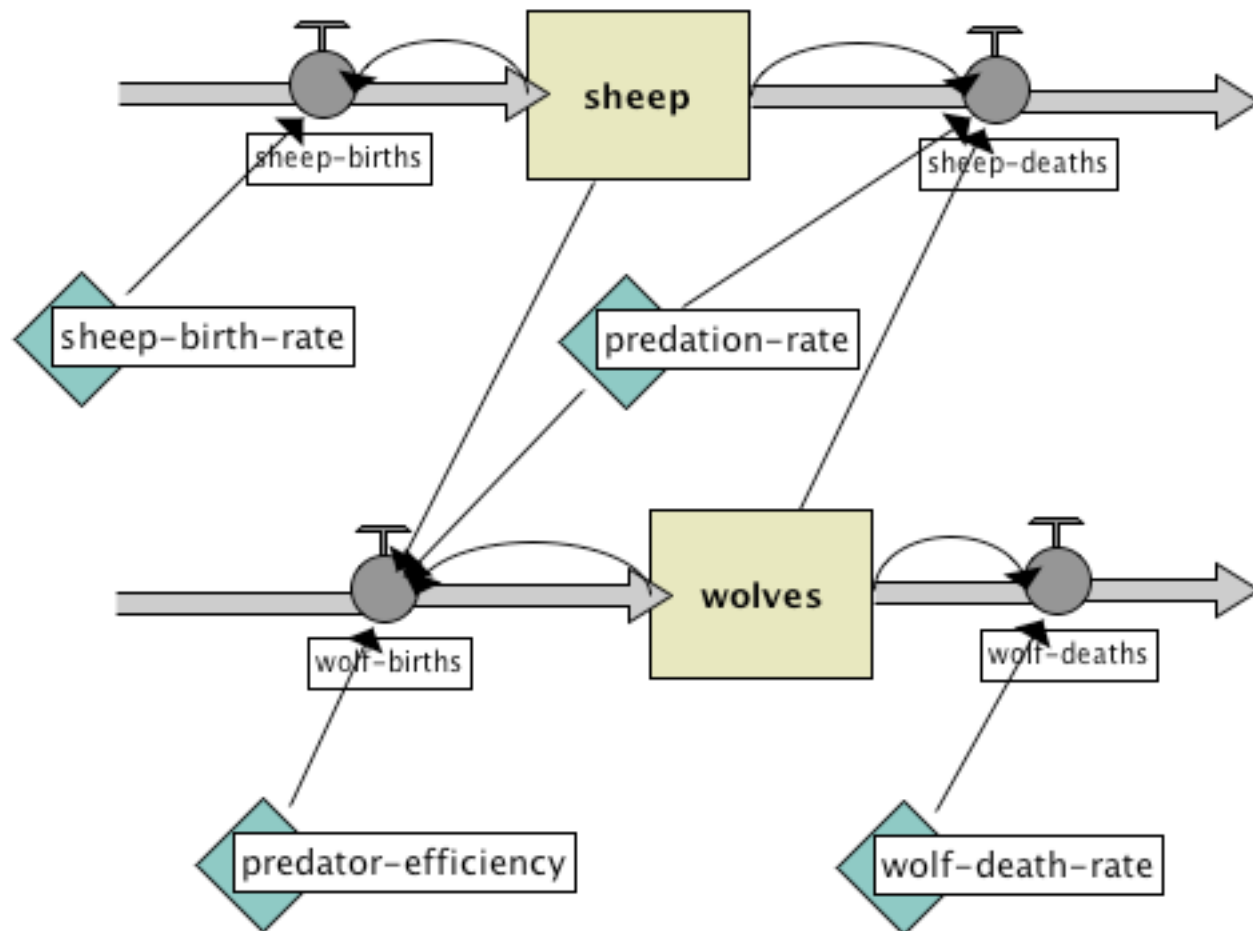
- Utilisation d'une plateforme multi-agent réactifs : NetLogo
- Utilisation de types cellules (patches), agents (turtles) et grille.
- Possibilité d'utiliser des indicateurs dédiés aux états des cellules,
- agents (compteurs, graphiques)
- Cas particulier de NetLogo : programmation "à la Logo" et affichage
- 3D possible. . .

Cas de l'éco-système

- Décomposition du problème: des agents représentent les moutons et des cellules représentent l'herbe.



Cas de l'éco-système



Cormas

The screenshot displays the Cormas software interface. At the top, a window titled 'Cell>>newState' contains a menu bar (Browser, Edit, Find, View, Category, Class, Protocol, Method, Tools, Help) and a toolbar with various icons. Below the toolbar, there are tabs for 'Category' and 'Hierarchy'. The 'Category' tab is active, showing a tree view with 'Cell' selected. To the right, there are tabs for 'Instance', 'Class', 'Shared Variable', and 'Instance Variable'. The 'Instance' tab is active, showing a list of instances including 'controle' and 'newState'. Below this, there are tabs for 'Source', 'Rewrite', and 'Code Critic'. The 'Source' tab is active, displaying the following code for the 'newState' method:

```
newState
| aliveNeighbours |

aliveNeighbours := (self neighbourhood select: [:a | a state = #alive]) size.

self state = #dead & (aliveNeighbours = 3)
  ifTrue: [^self bufferState: #alive].

(self state = #alive and: [aliveNeighbours = 2 or: [aliveNeighbours
  ifTrue: [^self bufferState: #alive].

^self bufferState: #dead
```

At the bottom of the window, there is a status bar showing 'Method: #newState (controle)' and 'Parcel: none'.

Below the main window, there is a 'Simulation' section with the following controls:

- A '><' button for navigation.
- An 'Initialize...' button with a dashed border.
- A 'Step' button.
- A 'Run' button.
- A numeric input field containing the value '100'.
- The text 'simulat' is partially visible on the right side.

Cormas

Observation of the space

Situated entities

- Fire_Man
- Fire_Cell

Symbols

- fireman

Methods

- pov (Fire_Man)

Color

Brightness :

Red : 0.67

Green : 0.00

Bleu : 0.67

0.67

Apply Clo

Torroidal 10 x 10 (4) Fire_Cell -> nil

Tesselation Topology Tools

The grid shows a 10x10 layout with green cells and purple triangles. The purple triangles are located at (row, column) coordinates: (1, 4), (1, 6), (2, 2), (2, 8), (3, 4), (3, 6), (4, 2), (4, 8), (5, 2), (5, 4), (5, 6), (5, 8), (6, 2), (6, 4), (6, 6), (6, 8), (7, 2), (7, 4), (7, 6), (7, 8), (8, 2), (8, 4), (8, 6), (8, 8), (9, 2), (9, 4), (9, 6), (9, 8), (10, 2), (10, 4), (10, 6), (10, 8).

Exercice

- Dans cet exemple, des agents vendent des livres et d'autres achètent des livres. . .
- Chaque agent acheteur reçoit le titre du livre à acheter (le livre cible) en argument et invite périodiquement tous les agents vendeur connus à fournir une offre.
- Dès qu'une offre est reçue, l'agent acheteur l'accepte et publie un ordre d'achat.
- Si plus d'un agent vendeur fournit une offre, l'agent acheteur accepte la meilleure (le plus bas prix).
- L'agent acheteur stoppe après avoir acheté le livre cible.
- Chaque agent vendeur a une GUI minimale permettant à l'utilisateur d'insérer de nouveaux titres (et les prix associés) dans le catalogue local des livres à vendre.
- Les agents vendeur attendent sans interruption des demandes des agents acheteur.
- Lorsque les agents acheteur sont invités à fournir une offre pour un livre, ils vérifient si le livre demandé est dans leur catalogue et répondent avec le prix.
Autrement ils refusent.
- Quand ils reçoivent un ordre d'achat, ils l'exécutent et enlèvent le livre demandé de leur liste