Reconciling Simulation and Distributed Deployment in a MAS Development Framework

Andrei Olaru Ionuț Hodoroagă

[andrei.olaru@upb.ro]

[ionut.hodoroaga@stud.acs.upb.ro]

AI-MAS Group

National University of Science and Technology POLITEHNICA Bucharest

25.05.2026

- Context
- Approach
- Implementation
- Future

Reconciling Simulation and Distributed Deployment in a MAS Development Framework

overview

Context

MAS platforms:

Network deployment

- multiple machines
- real-time
- fewer agents
- reduced environment support



Agent-based modeling and simulation

- single machine / computing cluster
- simulated / controlled time
- large number of agents
- complex environment support



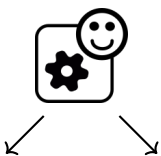
Context

MAS platforms:

Network deployment

- multiple machines
- real-time
- fewer agents
- reduced environment support

Jade SPADE



Agent-based modeling and simulation

- single machine / computing cluster
- simulated / controlled time
- large number of agents
- complex environment support

repast NetLogo

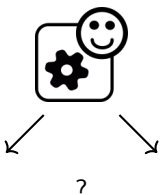
Context

MAS platforms:

Network deployment

- multiple machines
- real-time
- fewer agents
- reduced environment support

Jade SPADE



Agent-based modeling and simulation

- single machine / computing cluster
- simulated / controlled time
- large number of agents
- complex environment support

repast NetLogo

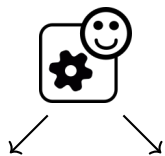
Context

MAS platforms:

Network deployment

- multiple machines
- real-time
- fewer agents
- reduced environment support

Jade SPADE



?

in:
· traffic
· drones
· smart spaces

Agent-based modeling and simulation

- single machine / computing cluster
- simulated / controlled time
- large number of agents
- complex environment support

repast NetLogo

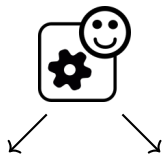
Context

MAS platforms:

Network deployment

- multiple machines
- real-time
- fewer agents
- reduced environment support

Jade SPADE



?

in:
· traffic
· drones
· smart spaces

Agent-based modeling and simulation

- single machine / computing cluster
- simulated / controlled time
- large number of agents
- complex environment support

repast NetLogo

Vision: Develop the means for the same agent code to be deployed in both *Sim*ulated and *IRL* (in real life) scenarios.

We see "*agent code*" as the code / behavior description / processing for the agent, minus all interactions with the environment of the agent or with other agents.

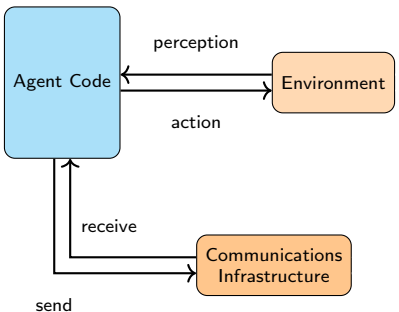
What is not "agent code":

- message delivery to other agents
- realizing actions in a physical environment
- obtaining perceptions from the environment
- management of timers and time-related triggers

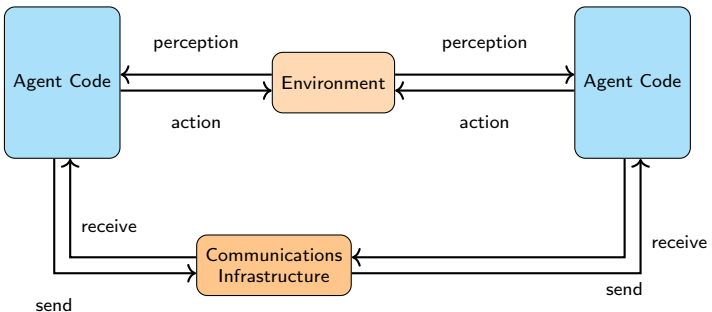
→ develop a set of principles to enable the same *agent code* to run both in network deployments and in agent-based simulation;

→ develop a set of building blocks that enable simulation of agent-based model applications and network-based deployment of agents, in the same framework.

IRL scenarios:



IRL scenarios:



- few agents per node
- concurrent, continuous execution
- real-time interactions

IRL:

- few agents per node
- concurrent, continuous execution
- real-time interactions

IRL:

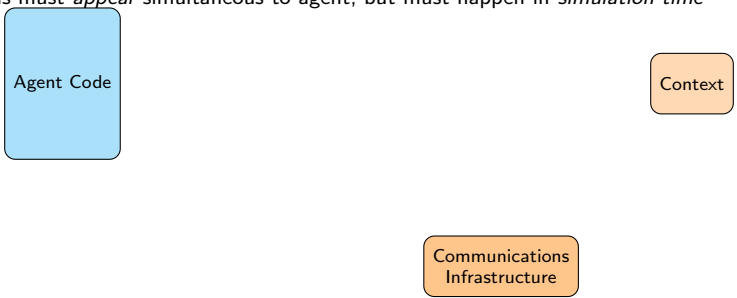
- few agents per node
- concurrent, continuous execution
- real-time interactions

Sim:

- all agents on one node
- execution *must* be sequential, and controllable
- interactions must *appear* simultaneous to agent, but must happen in *simulation time*

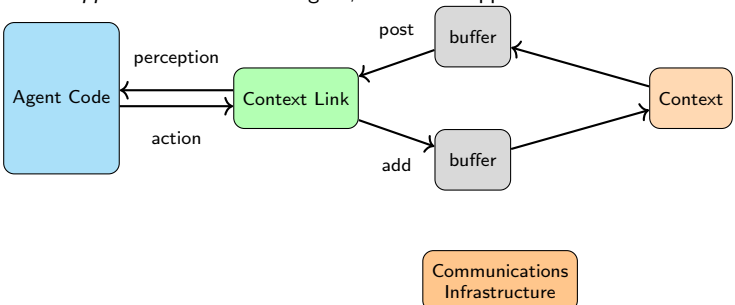
Sim scenarios:

- all agents on one node
- execution *must* be sequential, and controllable
- interactions must *appear* simultaneous to agent, but must happen in *simulation time*



Sim scenarios:

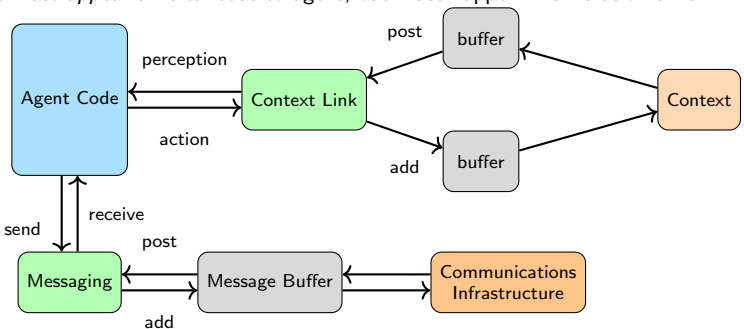
- all agents on one node
- execution *must* be sequential, and controllable
- interactions must *appear* simultaneous to agent, but must happen in *simulation time*



- Environment is represented by a set of *simulation contexts*.

Sim scenarios:

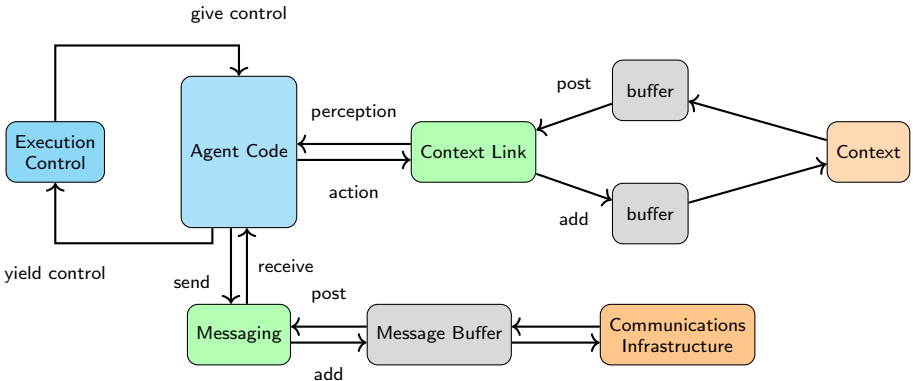
- all agents on one node
- execution *must* be sequential, and controllable
- interactions must *appear* simultaneous to agent, but must happen in *simulation time*



- Communications must go through a component external to the agent's decision process.

Sim scenarios:

- all agents on one node
- execution *must* be sequential, and controllable
- interactions must *appear* simultaneous to agent, but must happen in *simulation time*



- The execution model of agents must support step-wise execution or event-based control.

P1 Environment is represented by a set of *contexts* ← · simulation contexts · real contexts

P2 Communications must go through a component external to the agent's decision process.

P3 The execution model of agents must support either step-wise execution or event-based simulation control.

- the execution of the agent happens in small, self-contained steps – via a `step()` method or a `MAKE_STEP` event; or
- the execution of an agent can be interrupted and resumed at any time, with fine granularity, via events such as `suspend` and `resume`.

- Simulation object ← Any entity that is being simulated and is subject to execution control and separation from the environment.

Approach

- Simulation object ← Any entity that is being simulated and is subject to execution control and separation from the environment.
- Context ← Any entity which offers a link to / a representation of the environment. E.g. *space*, *time*, *communications*.

Approach

- Simulation object ← Any entity that is being simulated and is subject to execution control and separation from the environment.
- Context ← Any entity which offers a link to / a representation of the environment. E.g. *space*, *time*, *communications*.
- Executor ← An entity that manages the execution of simulation objects via:

Approach

- Simulation object ← Any entity that is being simulated and is subject to execution control and separation from the environment.
- Context ← Any entity which offers a link to / a representation of the environment. E.g. *space*, *time*, *communications*.
- Executor ← An entity that manages the execution of simulation objects via:
  - method calling → `step()` method.

Approach

- Simulation object ← Any entity that is being simulated and is subject to execution control and separation from the environment.
- Context ← Any entity which offers a link to / a representation of the environment. E.g. *space*, *time*, *communications*.
- Executor ← An entity that manages the execution of simulation objects via:
  - method calling → `step()` method.
  - starting the objects and issuing events to for simulation control.

The *agent code* must call context methods for

- access to the environment / information about other agents;
- sending messages to, or retrieving messages from other agents;
- time-related operations such as timers / waker behaviors

Agent code must be able to yield execution control after a given, a *limited time*, which must be below one *time step*.

→ This is already recommended practice in MAS frameworks.

The *agent code* must call context methods for

- access to the environment / information about other agents;
- sending messages to, or retrieving messages from other agents;
- time-related operations such as timers / waker behaviors

Agent code must be able to yield execution control after a given, a *limited time*, which must be below one *time step*.

· There is no requirement for internal agent structure, as long as the agent object implements

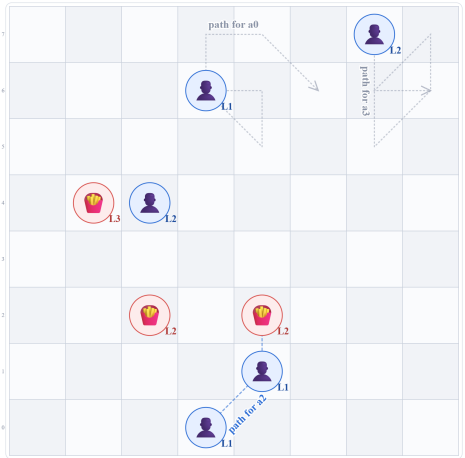
- a `step()` method, or
- a `postAgentEvent` method to receive execution control events.

We have started the validation of our approach by implementing medium-scale agent-based model applications in a framework that was built for network-based deployments, to evaluate performance and modeling choices.

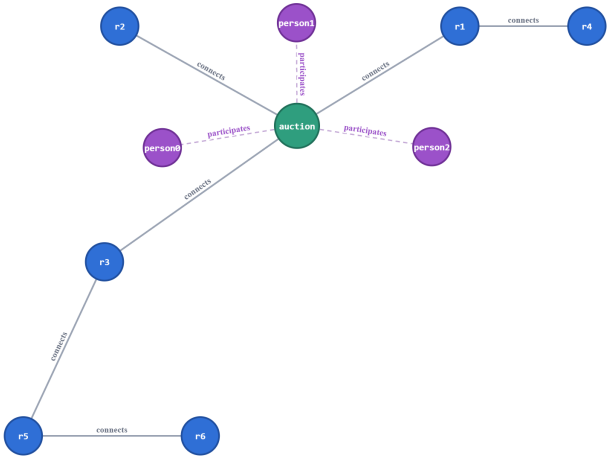
We have used FLASH-MAS – A Flexible, Lightweight Agent Shell 

Three scenarios:

wolf-sheep | level-based foraging | smart building conference rooms



level-based foraging

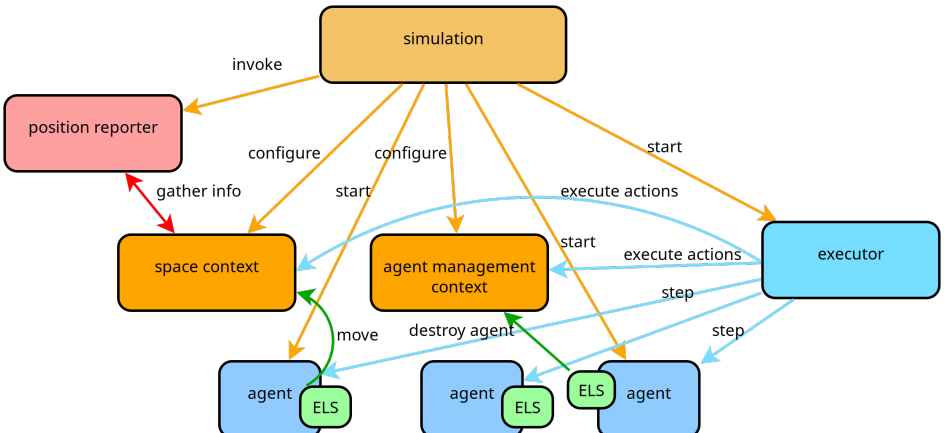


smart building conference room scheduling

Implementation

- Context support for:
  - temporal operations
  - space topologies (grid, graph, etc)
  - random numbers
- Environment Link Shard
- Execution support
  - stepwise execution (method-based or event-based)
  - free execution
- Agent groups
- Communications
  - network-based
  - local
  - topology-based

Implementation

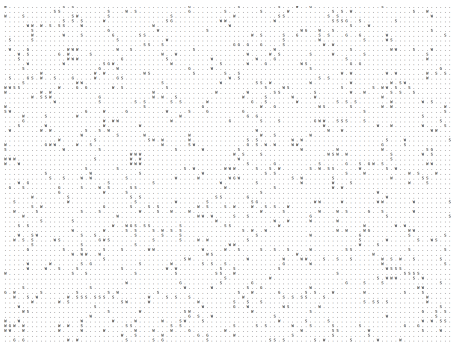


Implementation

```
-load_order simulation;executor;context;LBForagingGroup
-package net.xqhs.flash.abms abms.lbForaging
-loader LBForagingGroup
-simulation sim
     -executor StepWise:StepWise steps:100
     -context Random:random seed:42
     -context ProximityCommunication:comms
     -context Space:space width:8 height:8
     -context Foraging:foraging
     -LBForagingGroup g
          -patch Food n:3 level:2
          -agent Forager n:5 level:1 visionRange:2
```

Implementation

- successful deployment of ABM applications in a framework build for network-based deployments
- experiments with up to 5k agents without lag issues



Future

- We have shown that ABM simulations can be successfully deployed on a general purpose MAS framework, by integrated principles P1-P3 that we have devised
- We have developed a generic set of building blocks to support modeling of ABM applications in an entity-based framework.

Future work:

- flexible and uniform modeling of time as time units / simulated time / real time
- improved support for event-based execution control
- improved visualization

Future


Start application
Stop application
Agent (50)
serverPylon (1)
Monitoring&Control (1)
active Agents
Visualizations Dashboard
Clear Dashboard
Export YAML
Import YAML
+ Add Plo
test (agent)
33%
1x
state
tessst (agent)
66%
1x
teset (agent)
66%
1x
No Entities Selected

Thank You!

Questions are welcome!

This research is supported by the project "Romanian Hub for Artificial Intelligence – HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 334906.

[andrei.olaru@upb.ro]