Simulation and Development of Multi-Agent Systems

Henrique Lopes Cardoso

FEUP/LIACC

hlc@fe.up.pt

November 2016
Outline

• Agents: what are they?
• Agent-Based Modeling and Simulation (ABMS)
• Multi-Agent Systems (MAS)
• Blending approaches (ABMS+MAS)
• SAJaaS
• MASSim2Dev
• Summary and Extensions
What are Agents?

• Agent-based computing
  – Software agent
  – Intelligent agent
  – Active unit in a complex system
  – Interacting social component in a multi-agent environment
  – Modeling, design and programming paradigm (AOP)

• Different perspectives depending on your background
  – Multi-agent systems (MAS), a subfield of AI
    • Computer science, software engineering
  – Agent-based modeling and simulation (ABMS)
    • Social sciences (e.g. sociology, psychology, economics, demography), biological sciences, environmental modeling, ...
ABMS Perspective

• Agent-based model
  – computational model for simulating the actions and interactions of autonomous agents
  – assess overall properties or evolution of the system as a whole

• Goal
  – understand global or emergent phenomena associated with complex adaptive systems

• Agents are mostly homogeneous, simple, reactive...
  – ... but there are exceptions to this
MAS Perspective

• An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

• Intelligent agent
  – Reactivity: respond in timely fashion
  – Pro-activeness: goal-directed behavior
  – Social ability: interaction with other agents

• Multi-agent system
  – multiple interacting intelligent agents within an environment
  – solve problems that are difficult to model or impossible to solve using a monolithic system

• Focus on modeling actors in a system, and their interaction/coordination
  – agents are heterogeneous (architecturally, functionally), complex, adaptive
  – engineering perspective: validation of the future operation of actual agents
AGENT-BASED MODELING AND SIMULATION
Elements of an ABMS Tool

• Simulation concepts
  – Agents
  – Environment (space)
  – Model

• Scheduler
  – discrete-event simulator
  – time/tick stepped
  – agent-based
    • dynamic processes of agent behavior and interaction are simulated repeatedly over time

• Data collection and visualization

• Environment displays
ABMS Tools

- Three of the most widely used ABMS tools:
  
  **repast**
  
  - a family of advanced agent-based modeling and simulation platforms
    - Repast Simphony [North et al., 2005]

  **NetLogo**
  
  - NetLogo [Tissue and Wilensky, 2004]
    - a multi-agent programmable modeling environment

  **MASON**
  
  - MASON [Luke et al., 2005]
    - a fast discrete-event multiagent simulation library core in Java
RepastS Model Constructs

- **Context**: a group of agents
  - **Projections**: impose structure on agents and the space where they are situated
    - Apply to all agents in the Context
    - Continuous Space, GIS, Grid, Network
    - Projections provide an API for moving, neighboring, connecting, ...
  - **Sub-contexts**
    - Agents in a sub-context also exist in the parent context, but the reverse is not necessarily true

- **Agent**
  - POJOs

- **ContextBuilder** interface *(Data Loader)*
  - Context build(Context<Object> context);
  - Add agents, create projections, define sub-contexts
RepastS Scheduler

• Schedule **agent actions**

  – Methods of each class of objects that are in a Context

  – Three ways to work with the scheduler
    • Directly schedule a method invocation via the API (ScheduleParameters, Schedule)
    • Using **Java annotations**
    • Using **Watchers** (notifications of state changes in other agents)

  ❖ Note: In Repast3, model actions (e.g. display updates, data recording, snapshots) had to be scheduled through Java code as well. In Repast Simphony this is done via the **GUI**, and these actions do not feature in the code at all (they are managed by Repast’s runtime infrastructure).
RepastS Data Collection/Vis.

• **Data Sets**
  – Tabular data where each column represents a data source
  – **Data sources**
    • Standard (e.g. tick count)
    • **Method** invocations on objects that are inside a context
    • Custom
  – Aggregate / Non-aggregate
  – **Schedule parameters:** start time, priority, interval, ...

• **Writing data:** **Text Sinks**
  – A sink is associated with a data set
  – File / Console
  – Line / Tabular

• **Visualizing data:** **Charts**
  – A chart is associated with a data set
  – Time Series
  – Histogram (non-aggregate data)
RepastS Environment Displays

- Displays
  - Associated with projections
  - Chose which and how agents will be visualized
    - Class name
    - Style
  - Schedule parameters: start time, priority, interval, ...
  - 2D, 3D
RepastS GUI
MULTI-AGENT SYSTEMS
MAS Software Engineering

• **AOSE** (Agent-Oriented Software Engineering)
  – Abstractions: agent, environment, interaction protocol, context, roles, organizations, BDI
  – Methodologies: Gaia, MaSE, Prometheus, Tropos, ...

• **MAS programming constructs**
  – Agents (internal architecture and building blocks)
  – Infrastructure
    • Environment
    • Interaction artifacts/protocols (communication)
    • Distribution, mobility

• **Development tools**
  – IDE plugins, debugging
  – Agent and MAS visualization
MAS Development

- Some examples of platforms...
  - JADE
  - Jadex
  - Cougaar
  - Brahms

- ...and languages...
  - Jason (AgentSpeak)
  - 2APL
  - Concurrent MetateM

- ...and organizational/environment modeling and programming
  - Moise
  - CArtAgO
Standardization: FIPA

- **FIPA Abstract Architecture** Specification

  ![Diagram of Abstract Architecture]

  - Concrete realization: CORBA Elements
  - Message Transport, Agent Directory, Service Directory, ACL

- **Agent Communication Language (ACL) Specifications**
  - Message Structure, Communicative Act Library, Content Languages, Interaction Protocols

  ![Diagram of ACL Specifications]

- **Agent Management**

- **Agent Message Transport**

  ![Diagram of Agent Management and Message Transport]
JADE

• An Java framework for developing multi-agent systems

• FIPA-compliant
  – Agent Platform
    • Agent Management System (AMS)
    • Directory Facilitator (DF)
    • Message Transport System (MTS)

  – Agent Communication Language (ACL)
  – Interaction Protocols
JADE Architecture

Agent Management System

Directory Facilitator

Agents
JADE Programming

• Agents
  – 1 Agent = 1 Thread
  – Main construct: behaviour
    • Tasks, executed concurrently
  – Behaviour scheduling
    • Not preemptive, but “cooperative” (thread sharing)
    • Conceptually, behaviours should be seen as parallel
  – Communication using ACLMessages
  – Mobility and cloning
    • Agents can migrate throughout containers

• JADE API includes, among other things:
  – Several behaviour classes
  – ACL: messages, interaction protocols, ontologies
  – DFSERVICE / AMSService

• Deployment: distributed MAS
  – Agents execute within containers (JVMs)
JADE Agent Execution

1. `setup()`
   - Initializations
   - Addition of initial behaviours

2. Agent has been killed (delete() method called)?
   - Highlighted in red: the methods that programmers have to implement
   - Agent “life” (execution of behaviours)
   - Clean-up operations

3. Get the next behaviour from the pool of active behaviours
   - b.action()

4. b.done()?
   - NO
   - YES: Remove current behaviour from the pool of active behaviours

5. TakeDown()
JADE Tools

- Remote Monitoring Agent (JADE’s “GUI”)
- Dummy Agent
- Sniffer Agent
- Directory Facilitator GUI
- Introspector Agent
BLENDING APPROACHES
Rationale

• MAS provide powerful abstractions and mechanisms for effectively modelling real-world applications that are highly complex and dynamic
  – manufacturing, e-commerce, network management, distributed sensing and control, information retrieval, ...

• MAS need to be validated before being deployed and executed in real operating environments
  – scale and complexity of systems are too demanding to be managed in real execution testing scenarios

• Methodologies that support system validation through simulation are required
  – discrete-event simulation, agent-based simulation
Approaches

• **ABMS for agent-based software development**
  – SeSAm [Klügl et al., 2003]

• **Multi-agent based simulation (MABS)**
  – Domain-specific
    • MATSim [Balmer et al., 2008]
    • PlaSMA [Warden et al., 2010]
    • MASeRaTi [Ahlbrecht et al., 2014]
  – General purpose
    • Jadex [Braubach et al., 2012]

• **Extensions**
  – NetLogo+BDI+ACL agent programming [Sakellariou et al., 2008]
  – JADE+simulation
    • MISIA [García et al., 2011]
    • JRep [Gormer et al., 2011]
    • PlaSMA [Warden et al., 2010]
SeSAm

- Shell for Simulated Agent Systems
  - Relating agent-based simulation and software development
  - Virtual environments for agent based software

(a) Real world scenario
(b) Simulated testbed scenario
Jadex

- Active components: unified execution infrastructure for agents and workflows
- BDI agents
- Applications executable as simulations as well as real time
MISIA

- Middleware Infrastructure to Simulate Intelligent Agents
JRep

- Integration of JADE and Repast Simphony
PlaSMA

- Platform for Simulations with Multiple Agents (logistics domain)
  - JADE extension: simulation control (synchronization) and world model (ontology)
Simple API for JADE-based Simulations

SAJaaS
SAJaaS: why?

• **JADE:**
  – Multi-agent systems development
  – Not suited for multi-agent based simulation (MABS): scalability

• **Repast:**
  – Agent-based simulation
  – Lack support for agent programming and multi-agent features (communication, infrastructure, ...)

• **However:**
  – Need to *simulate while developing* a full-featured MAS, for testing purposes
Architecture
SAJaaS Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>JADE</th>
<th>Repast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Simulation Tools</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Scalability</td>
<td>Limited</td>
<td>High</td>
</tr>
<tr>
<td>Open Source</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Agent Execution</td>
<td>Behaviours Multi-thread Event-driven Asynchronous</td>
<td>Scheduler Single-thread Tick-driven Synchronous</td>
</tr>
<tr>
<td>Interaction</td>
<td>FIPA ACL</td>
<td>Method calls Shared resources</td>
</tr>
<tr>
<td>Ontologies</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

from the simulation scheduler point of view
Usage
SAJJaS API Overview
Benchmark Scenario

- **Service Consumer/Provider**
  - $5 \times n$ providers; $2 \times n$ consumers (AllProv, ProvSel)
  - DF, behaviours, FIPA-protocols, ACL, ontologies

---

Benchmark Scenario

- **Service Consumer/Provider**
  - $5 \times n$ providers; $2 \times n$ consumers (AllProv, ProvSel)
  - DF, behaviours, FIPA-protocols, ACL, ontologies

Main Advantages

- **MABS programmer**
  - has a rich set of **multi-agent programming features** offered by JADE
  - may explore **simulation-related features** offered by the simulation infrastructure (e.g. Repast)

- **Same implementation** can be used both for **simulation and deployment purposes**
  - checkout *MASSim2Dev*

- **Simulation performance gains** in certain scenarios: high communication-to-computation ratio
MAS Simulation to Development
https://web.fe.up.pt/~hlc/doku.php?id=massim2dev

MASSim2Dev
SAJaS and MASSim2Dev

- **SAJaS** re-implements some core JADE classes
  - Core classes
    - Agent, AID
  - Runtime infrastructure
    - Runtime, PlatformController, ContainerController, AgentController
  - FIPA services
    - FIPAService, AMSService, DFService, DFAgent
  - Agent dependencies
    - Behaviours, protocols

- **MASSim2Dev** Eclipse plugin
  - Goal: automate project conversion: JADE ↔ SAJaS
  - Currently supports three types of conversion:
    - SAJaS → JADE
    - JADE → SAJaS+Repast3
    - JADE → SAJaS+RepastS
Steps

1. Clone Java project

2. Refactor source files
   - Change project dependencies on JADE/SAJaS to the equivalent classes in SAJaS/JADE
   - JADE-SAJaS dictionary contains a mapping of classes

3. Create empty launcher
   - JADE, Repast3 or RepastS

4. Set build path
   - If JADE → SAJaS, add SAJaS library
GUI
SUMMARY
MAS Simulation vs Development

• Multi-Agent based Simulation (MABS)
  – Computer simulation where entities are modeled and implemented as agents
  – Agent-based simulation tools
    • Discrete-events, focus on performance, large scale, interaction environment
    • Lack of support for agent programming and MAS infrastructures

• Multi-Agent System
  – System composed of autonomous, intelligent and interacting agents
  – Development tools
    • Support for communication, distribution, standards (FIPA)
    • Multi-threaded, limited scalability, not appropriate for simulation

• MABS can be useful while developing MAS applications
SAJaS+MASSim2Dev

• Bridge simulation and development of MAS
  – Develop high-performance simulations using MAS development features: “MAS-like MABS”
  – Convert MABS into MAS automatically (write-once, simulate and deploy)

• Approach
  – Simple API for JADE-based Simulations (SAJaS)
    • Light implementation of most JADE features
    • Integration with simulation framework (e.g. Repast)
  – MAS Simulation to Development (MASSim2Dev)
    • Eclipse plugin conversion tool: JADE ↔ SAJaS
Extensions

• Enhancement of facilities included in SAJaS/MASSim2Dev
  – portable data collection and visualization tools
    • between Repast and JADE, through MASSim2Dev
  – additional simulation and conversion options

• Large-scale JADE-based BDI Simulation
  – enrich SAJaS with BDI reasoning agents
    • Jason
  – BDI simulation scalability
    • distributed vs shared BDI reasoning engine