EGI CF 2015, BARI, Exploiting the EGI Federated clouds, 11 Nov. 2015

Scalable Agent Platforms with friendly interaction for modeling practical problems

Presented by Luis Cabellos
Co-authors: Jesús Marco, Hector Rodríguez
Instituto de Física de Cantabria (IFCA)
UC, University of Cantabria
CSIC, NATIONAL RESEARCH COUNCIL
SANTANDER, SPAIN

Scope

- Introduction: Agents based models
- The “Machanguitos” platform
- Addressing a practical problem
- Ongoing work
- Interest of ABM for Federated Clouds

Introduction

- Elements of Computational Models for Concurrent Computing:
  - Agents
  - Actors
  - Entities*
- Agent based computing: systems are composed of multiple structures (agents) interacting over an environment. Components:
  - Agents
    - Internal State
    - Update Function
  - Environment
- Actor model: the system is composed of a single structure: the Actor. An Actor can:
  - Send/Receive messages
  - Have a Internal State/Logic
  - Spawn other Actors

ABM vs MAS

- ABM = Agents-Based Models
- MAS = Multi-Agent Systems
- They have different goals:
  - ABM: search for explanatory insight into the collective behavior of agents obeying **simple rules**
  - MAS: computerized system composed of multiple interacting intelligent agents within an environment.
  - Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve
- There is a considerable overlap: as we will see, the proposed platform, Machanguitos, can be seen as

The Easiest Simplest Multi-Agent System

Introducing Machanguitos : Features

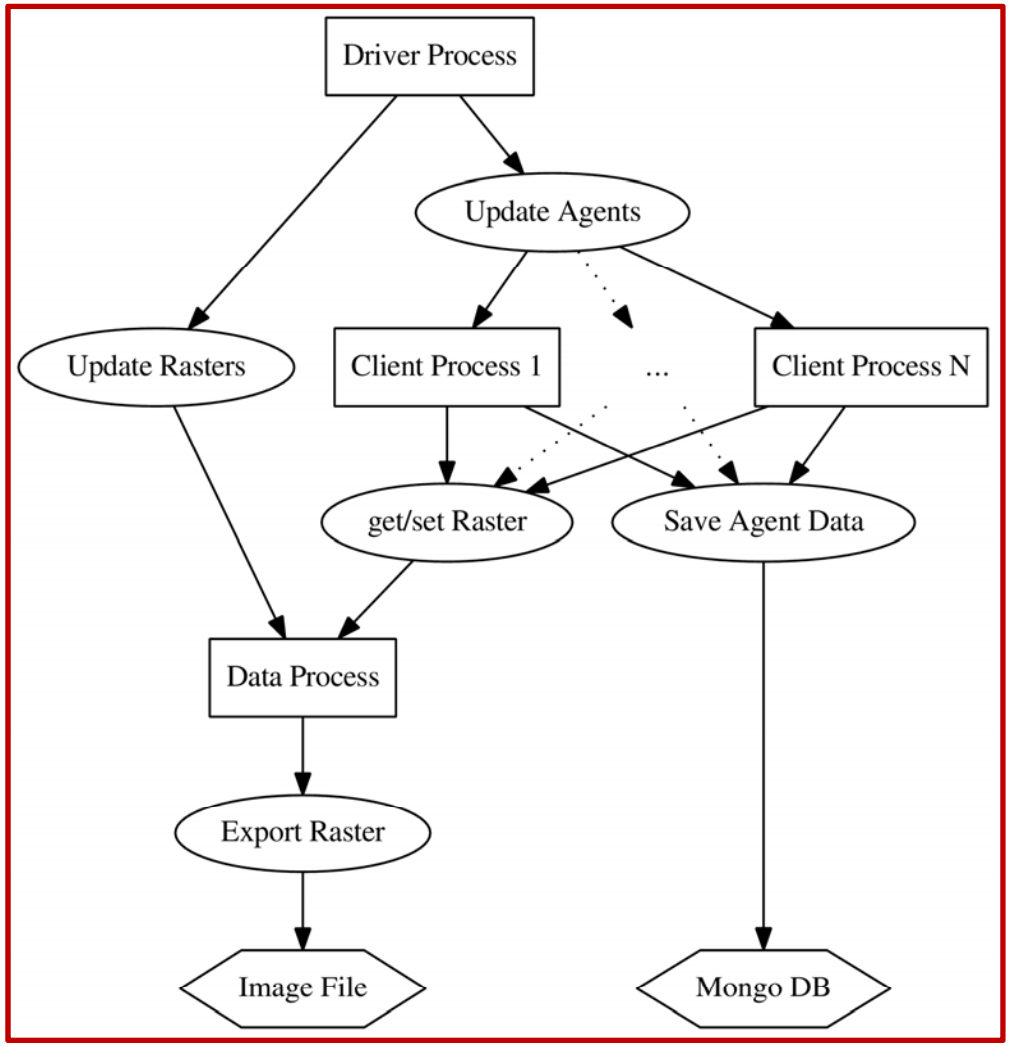
- Agent-Granularity: agents at various scale
  - Only 1 scale
- Decision-making heuristics
  - **Scripting Language for definition of Agent behavior**
- Learning rules or adaptive processes
  - Agents with internal state
- An interaction topology
  - Stand-alone Agents
- An (non-agent) environment
  - Raster 2D

Introducing Machanguitos : scripting

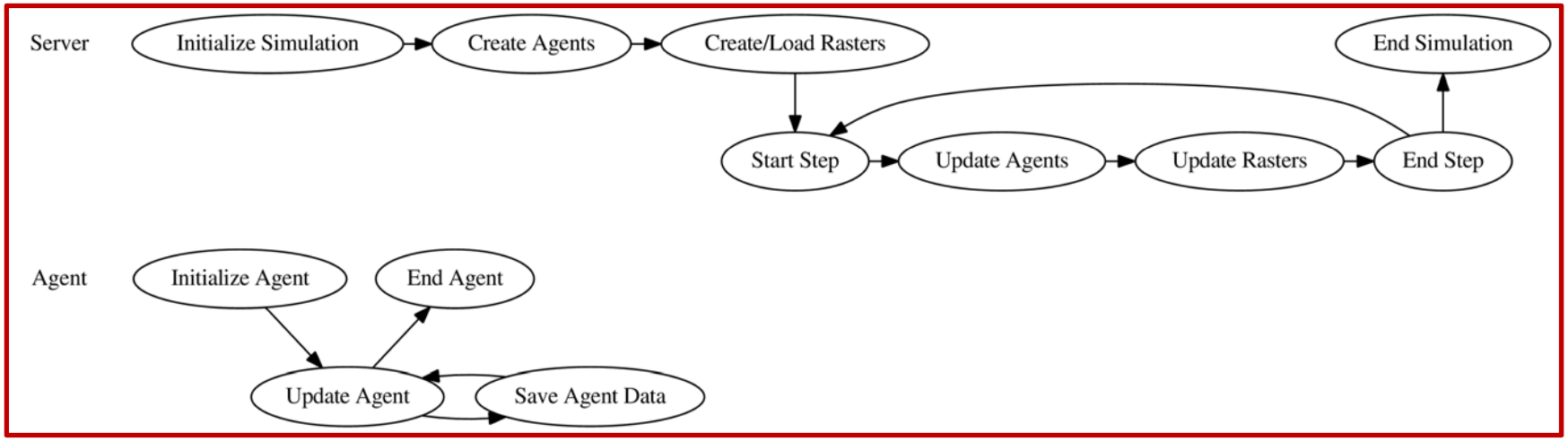
```
function Agent:checkHill(delta)
   self.dx = ((-20 + math.random(40))/ 100.0)*delta);
   self.dy = ((-20 + math.random(40))/ 100.0)*delta);
   tempX = self.x + self.dx;
   tempY = self.y + self.dy;
   local area = raster.area:get( 0, tempX, tempY);
   if area > 0 then
      self.x = tempX;
      self.y = tempY;
      raster.position:increment( 0, self.x, self.y, 200 );
   end
end

function Agent:eatAndPoop(delta)
   local grass = raster.grass:get( 0, self.x, self.y);
   if (grass > 0) then
      raster.grass:increment( 0, self.x, self.y, self.grassEated*delta );
   end
   local inc = self.grassToManure * delta;
   raster.manure:increment( 0, self.x, self.y, inc );
end
```

Machanguitos Run Model



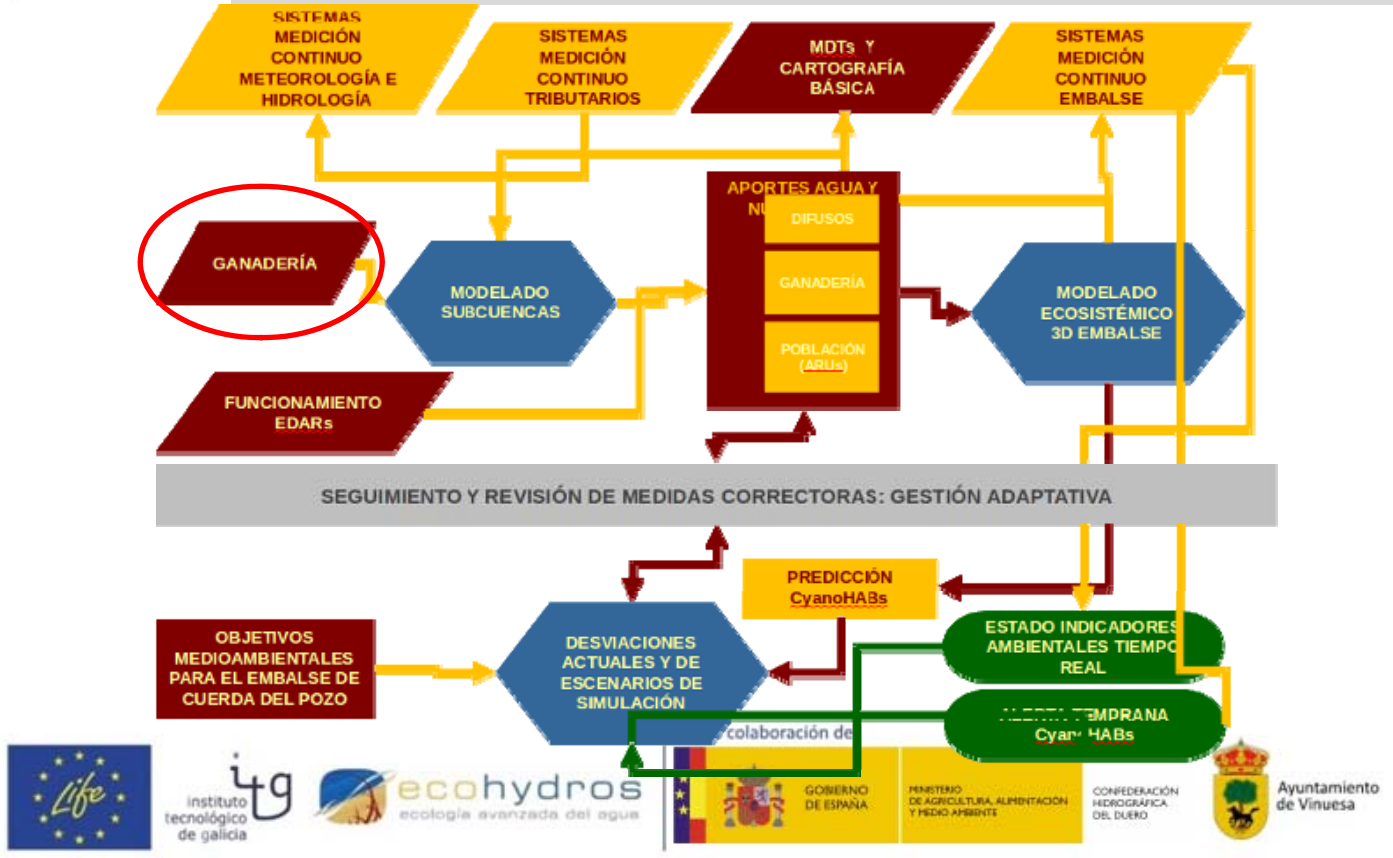
Machanguitos Run States



Addressing a practical problem

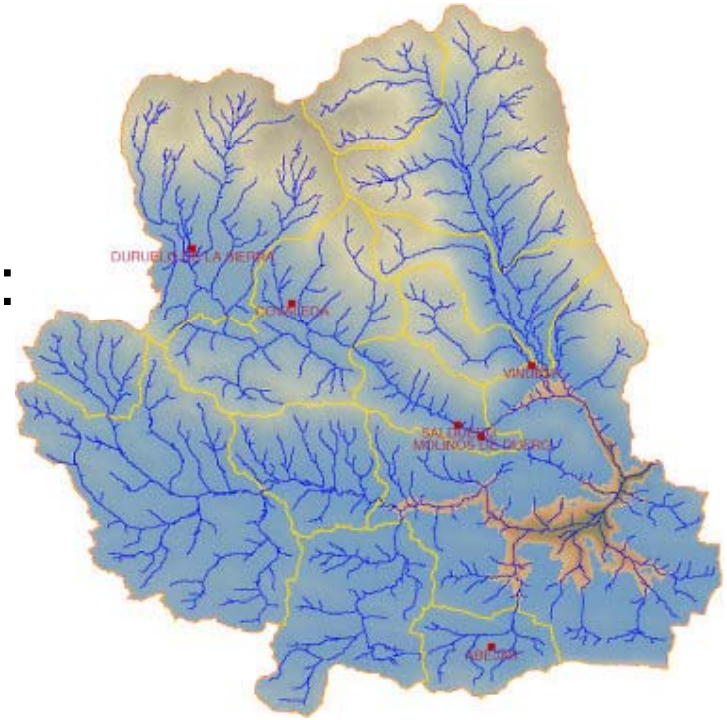
ROEM+

Advanced management of eutrophication problem in a water reservoir (LIFE+ project)



How to "assign" uncertainties to key but complex processes?

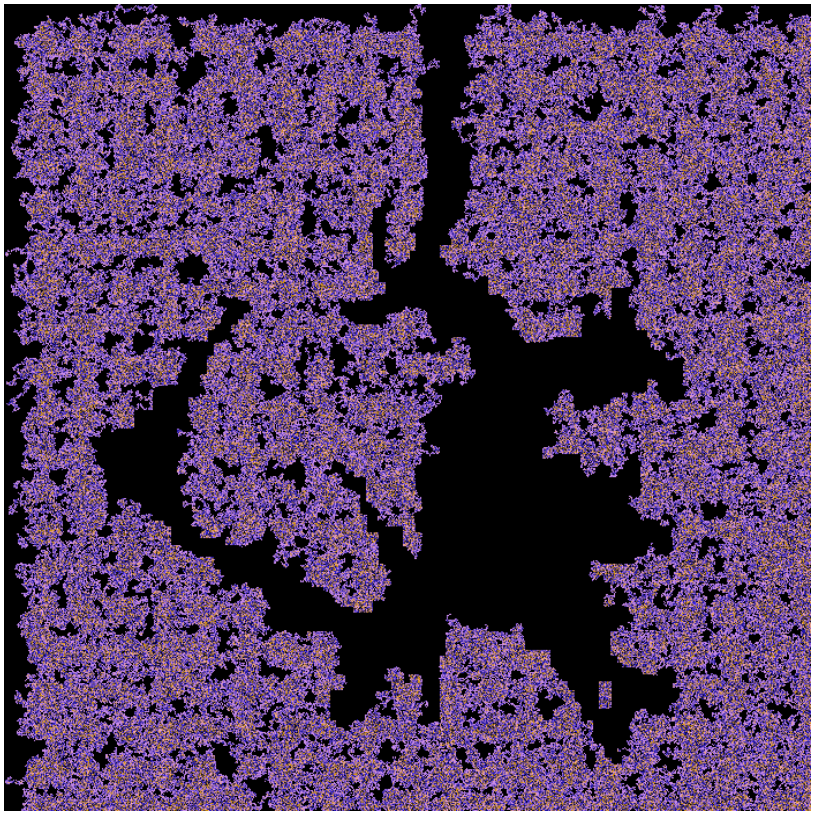
- **Practical Problem: impact of cattle management**
  - Extensive or semi-intensive
  - >6000 cows
  - >10000 sheeps
- Parameterization applied based on:
  - P and N deposits/animal
  - Run-off (7% if 30mm rain…)
- But real life is much more complex



ABM simulation of cows impact

- 1200 iterations
- 10K – 1M cows
- 5K - 500K sheeps

	15K agents	150K agents	1.5M agents
Cores	seconds	seconds	seconds
4	179	1086	8787
8	142	651	5645
~~16~~	~~141~~	~~722~~	~~6597~~
~~32~~	~~142~~	~~799~~	~~8149~~
~~64~~	~~167~~	~~1072~~	~~8289~~
~~128~~	~~206~~	~~1196~~	~~10759~~



On going work

Evolution of the Platform

- **Better concurrent access to the data**
- Add Actors properties
- Better definition of environment
- Other environments (Dynamic GIS)

Future of the Model

- Realistic scripts for cows and ships
- Realistic mineralization processes?
- Validation

Interest of ABM for Federated Clouds

Implementation as SaaS ?

- **Service orientation**
- Collect scenarios, scripts
- Well suited to distributed execution
- Multilayer approach

Many areas of application

- Socio-economic systems
- Smart cities

Questions?