How to test and compare multiagent systems?

Jürgen Dix
(joint work with
T. Behrens, K. Hindriks, J. Hübner, M. Köster, F. Schlesinger
M. Dastani, P. Novak)

Department of Computer Science
Clausthal University of Technology
16th July, IJCAI-NRAC, Barcelona, Spain

1 Multi-Agent Contest

- The Idea
- First Scenario (2005-2007)
- Second Scenario (2008-2010)

2 Environment Interface Standard (EIS)

- MASSIM
- Other environments
- EIS
- EISified APL's and Environments

3 New Scenario

1. Multi-Agent Contest

Overview

The idea

Toni/Torroni (2004): Need for a competition event for

1. **modelling** (problems using logic-based agents)
2. **specifying** (logic-based multi-agent systems, given a problem)
3. **programming/implementing** (logic-based multi-agent solutions)

Dastani/Dix/Novak were lured into it: Agent Contests 2005–2010.

Simulation Server: Suitable for problems like simulated mobile/cognitive robotics. **To provide a dynamic environment for those simulated players.**

Aim

- Stimulate research in the area of **multi-agent systems programming**
- Identify **key problems**
- Collect **suitable benchmarks** that can serve as milestones for evaluating new tools, models, and techniques
- Gathering test cases which require and enforce **coordinated actions**

Aim (2)

- Focus on Deliberation based on formal approaches and computational logics (CLIMA WS series)
- Scenarios should encourage cooperative problem solving not based on centralized approaches: agents should operate on their own, not as slaves.
- This is difficult to achieve: communication is done on each agent platform not in the server.
- All approaches are welcome, even non agent approaches (not based on a APL).

Challenge:

Solve a cooperative task in a dynamically changing environment.

First Contest in 2005

- No server platform available.
- We provided the participants with a precisely defined scenario.
- Participants had to implement the scenario, solve the problem (based on randomly generated maps).
- The whole system was sent to us for inspection.

Running these systems . . .

First Contest in 2005

- No server platform available.
- We provided the participants with a precisely defined scenario.
- Participants had to implement the scenario, solve the problem (based on randomly generated maps).
- The whole system was sent to us for inspection.

Running these systems . . . was a nightmare.

Details

Technical Infrastructure:

- TCP/IP based client/server-architecture
- simple processing of XML documents (message exchange)
- the organizers provide the server
- the participants connect

Discrete Simulation: in each step do

- send perceptions to agents
- wait for agents' actions or timeout
- let agents act and let the world evolve

Tournament

- Game between two players
- Step duration: 4 seconds
  (compare to 8 updates per second in a real-time game)
- different maps, 1000-1400 simulation steps
- 3 simulations = 1 match
- each team against all others, 1 match per pair

Overview

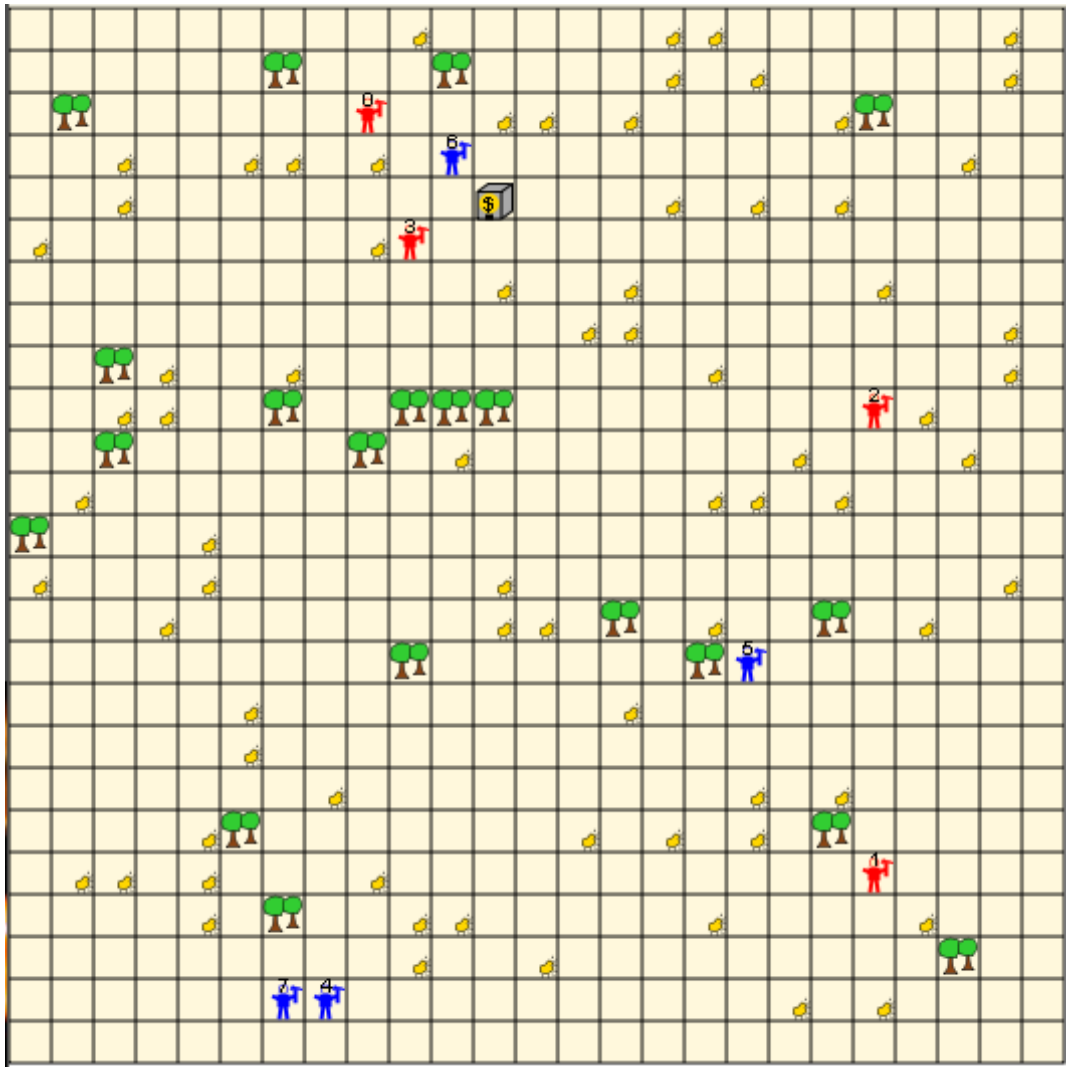
The Scenario

Simple idea:

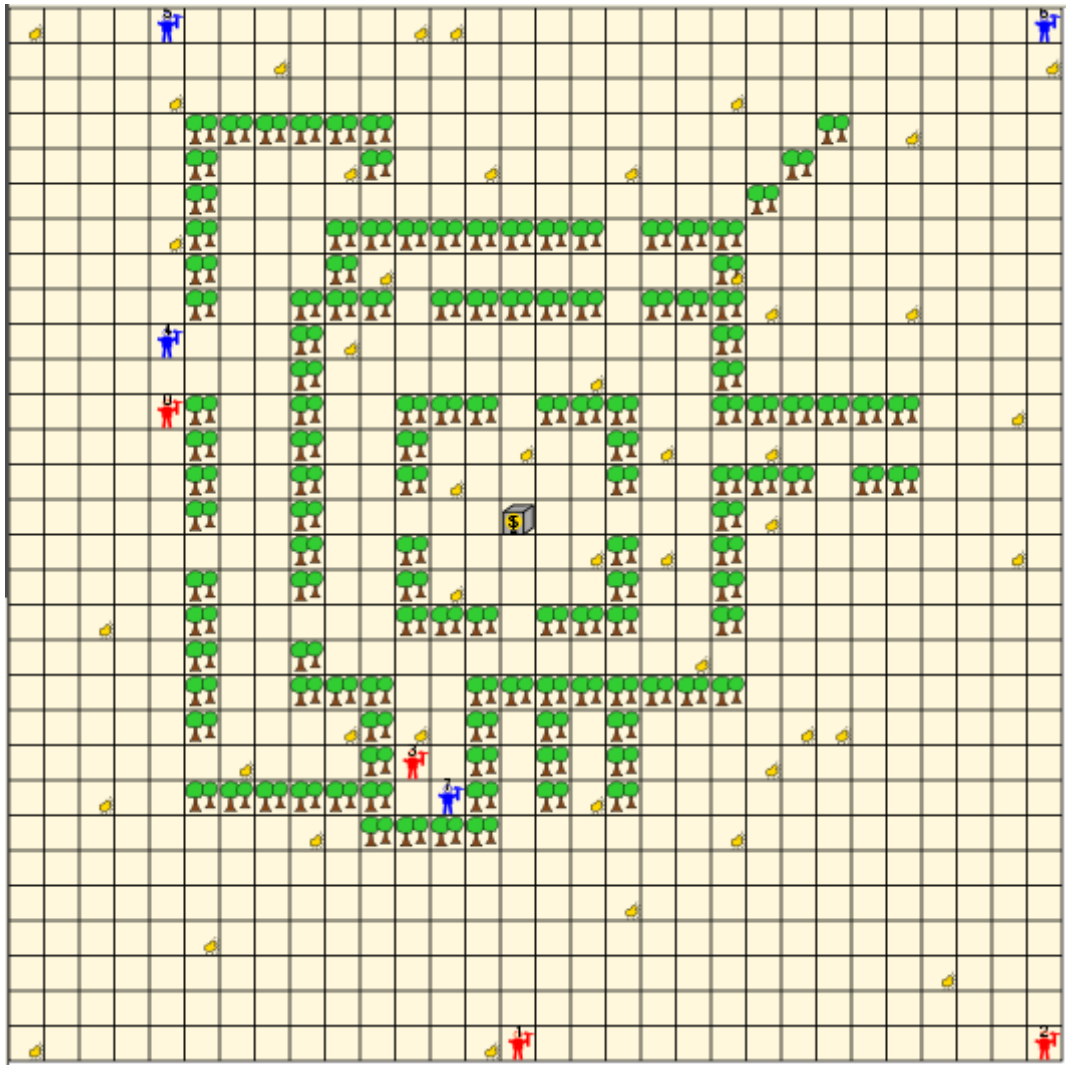
- Grid-like world where agents can move from one slot to a neighbouring slot;
- Food can appear dynamically at random,
- No lookahead,
- Depot for storing food,
- Agents can have different roles, communicate, cooperate,
- 4 agents starting from the corners.

Who can collect most of the food?

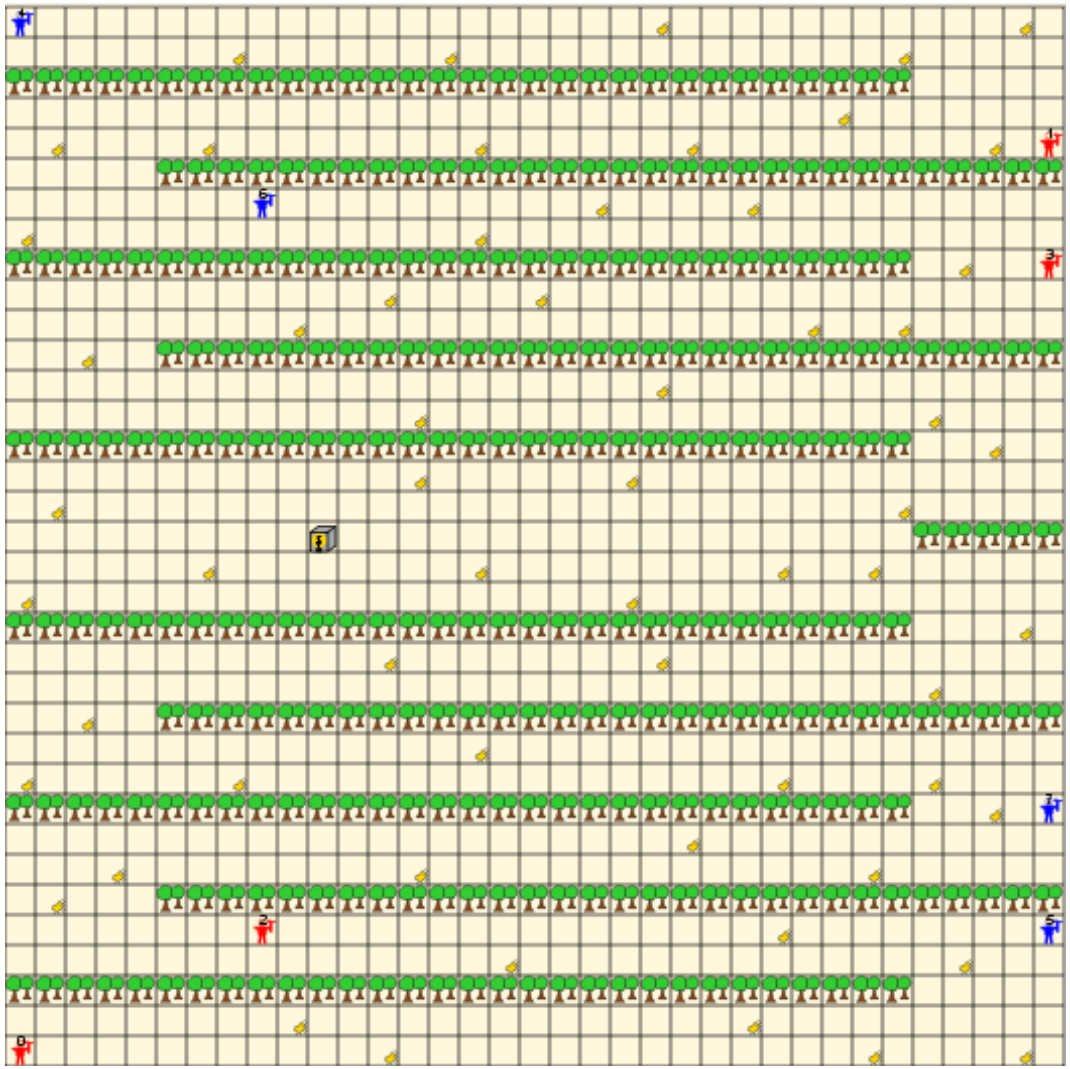
Simple maze (random)



More difficult maze



Most difficult maze



3 Contestants in 2006

3 Contestants in 2006

Bordini/Hübner/Tralamazza: (UK/Brazil/Switzerland) Jason: agent platform based on Agentspeak. Use algorithms like A* or DCOP (available in Java and thus easily useable).

3 Contestants in 2006

Bordini/Hübner/Tralamazza: (UK/Brazil/Switzerland) Jason: agent platform based on Agentspeak. Use algorithms like A* or DCOP (available in Java and thus easily useable).

Cares/Franch/Mayol: (Spain/Chile) Tropos: agent-oriented software development methodology. Using it, a Prolog implementation has been generated (built on last years contest team).

3 Contestants in 2006

Bordini/Hübner/Tralamazza: (UK/Brazil/Switzerland) Jason: agent platform based on Agentspeak. Use algorithms like A* or DCOP (available in Java and thus easily useable).

Cares/Franch/Mayol: (Spain/Chile) Tropos: agent-oriented software development methodology. Using it, a Prolog implementation has been generated (built on last years contest team).

Schiffel/Thielscher: (Germany) FLUX: CLP-based programming method, fluent calculus (solving the frame problem), based on situation calculus

3 Contestants in 2006 + trainings team (TUC)

Dummy team: contestants could play against it in the weeks before the contest (trainings phase). We made the team a bit stronger for the actual contest.

Bordini/Hübner/Tralamazza: (UK/Brazil/Switzerland) Jason: agent platform based on Agentspeak. Use algorithms like A* or DCOP (available in Java and thus easily useable).

Cares/Franch/Mayol: (Spain/Chile) Tropos: agent-oriented software development methodology. Using it, a Prolog implementation has been generated (built on last years contest team).

Schiffel/Thielscher: (Germany) FLUX: CLP-based programming method, fluent calculus (solving the frame problem), based on situation calculus

Figure : Gold Miners 2006: CLIMABot (blue) vs. brazil (red)

Winner 2006

From the three regular contestants in 2006, the

brazilian team

won by quite a margin.

Winner 2006

From the three regular contestants in 2006, the

brazilian team

won by quite a margin.

However, had the dummy team be allowed to enter the competition, it would have won it!

Winner 2006

From the three regular contestants in 2006, the

brazilian team

won by quite a margin.

However, had the dummy team be allowed to enter the competition, it would have won it!

Systems were not yet very stable, not playing strongly. Contest was used as a good debugging tool.

Some Notes

Depot: position is known, **could be blocked, but not easily**;

Teleporting: could be used as a feature (and has been)

Position: own position is known, agents are positioned without any advantage (after the maze has been generated);

Markers: allowed, but not used (to the contrary ...);

Fog: Perception of neighbouring cells is incomplete (1-10 %), **but it is known**;

Skill: actions could fail with a probability (2%);

Winner 2007

1	JiacIVteam	63
2	microJiacteam	54
3	Jasonteam	49
4	FLUXteam	43
5	APLteam	12
6	JACKteam	3

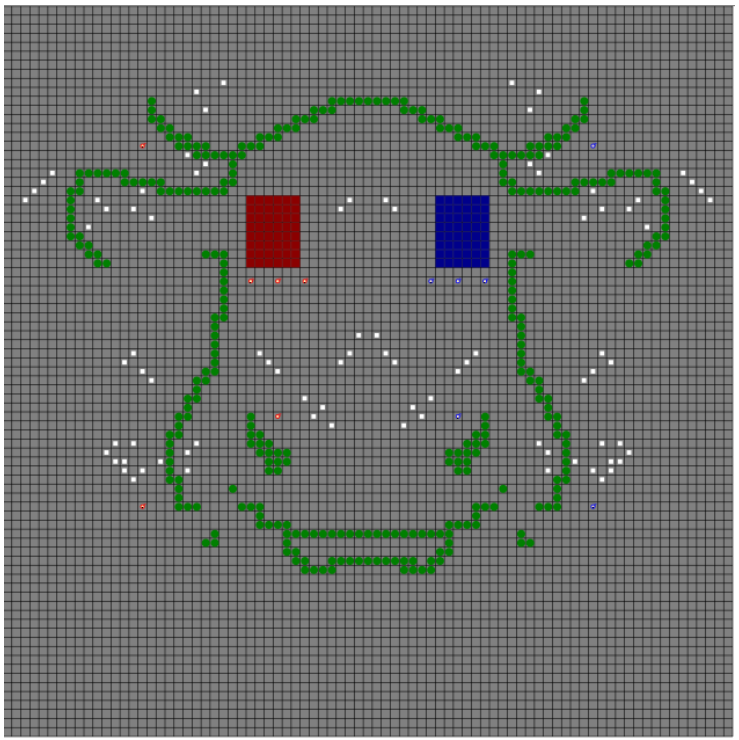
- **Problem**: Not the MAS but its programmed strategy is evaluated.
- Path finding is most important: efficient implementations of A* paid off.
- Self interested agents without any cooperation are sufficient.
- Find a scenario that favours true collaboration between agents.

Overview

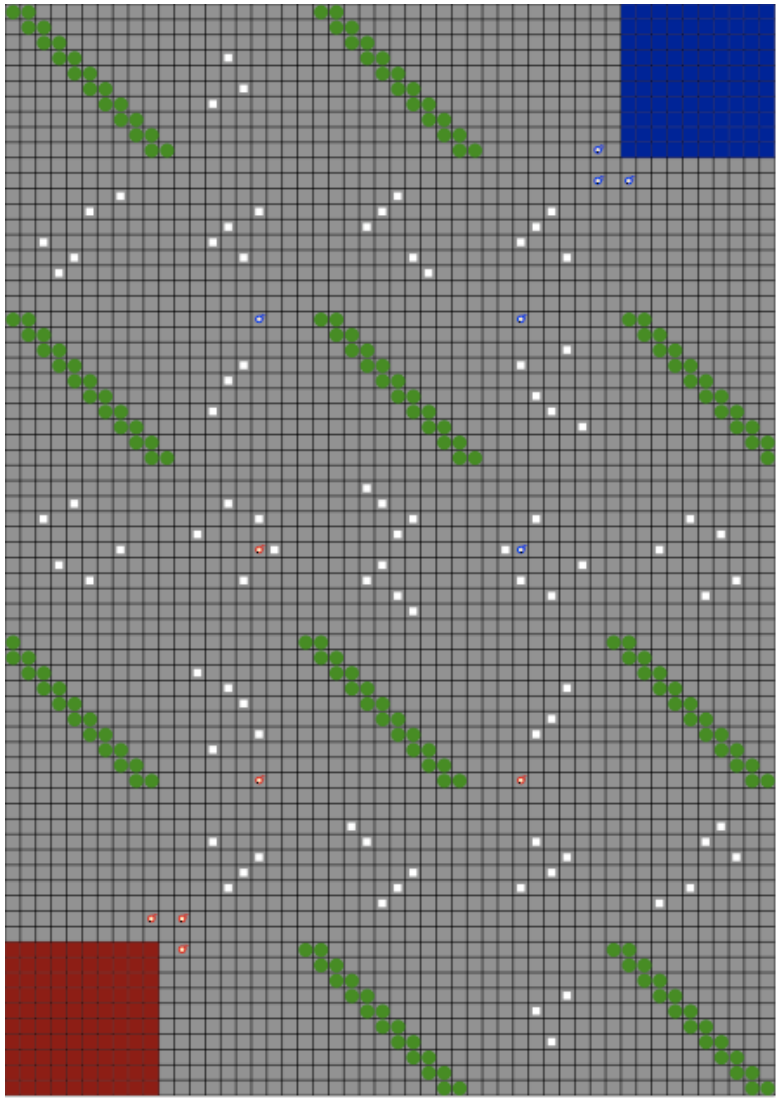
Scenario: Cows and Cowboys

Task: implement a team of agents that collects more cows than the opponent

Aim: agents have to cooperate and coordinate their actions



Environment



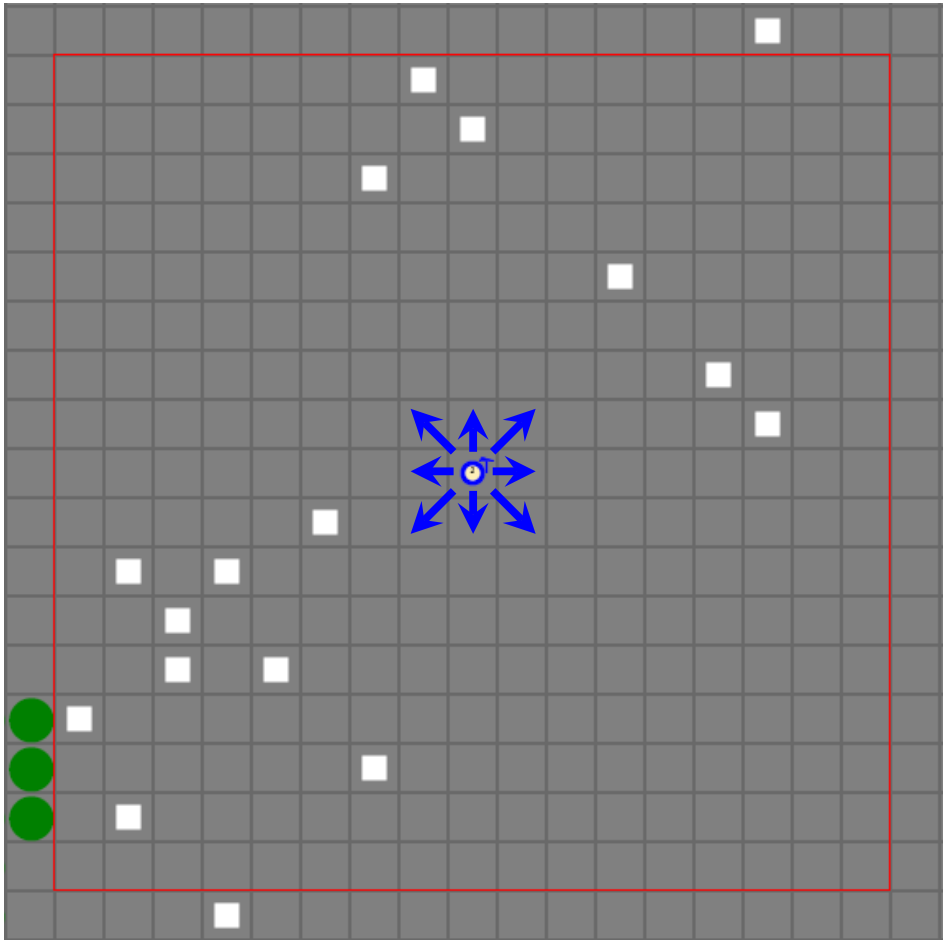
- Cows
- Cowboys
- Corrals
- Obstacles

What is the optimal solution?

What is the optimal solution?

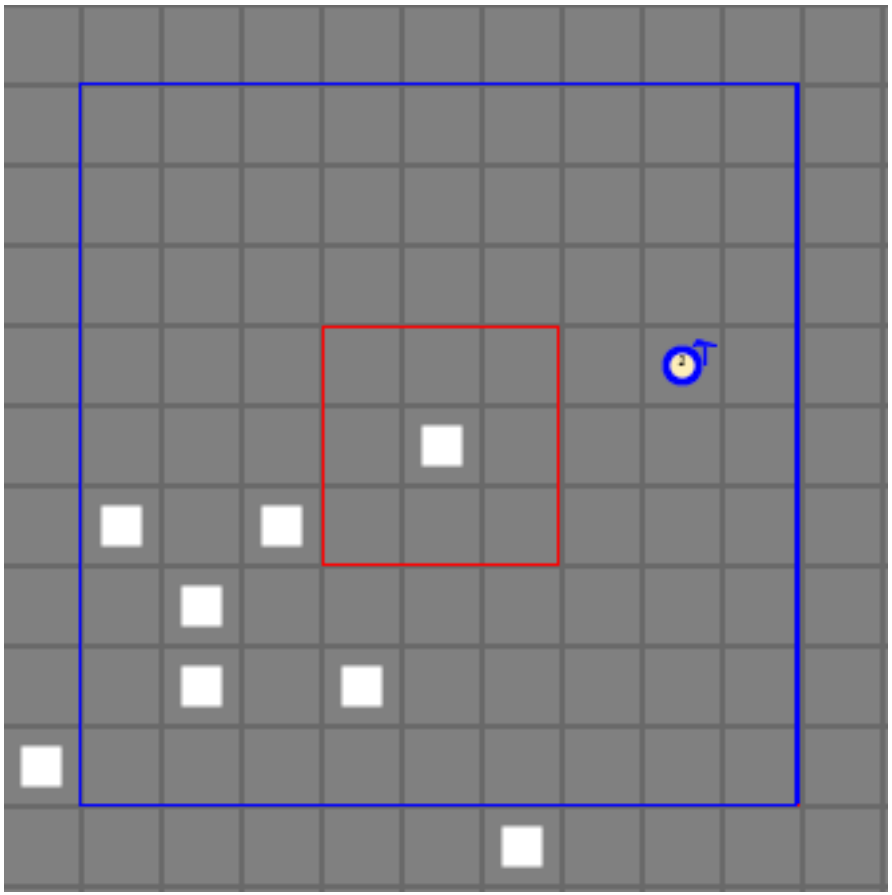
We have no idea!

Agents



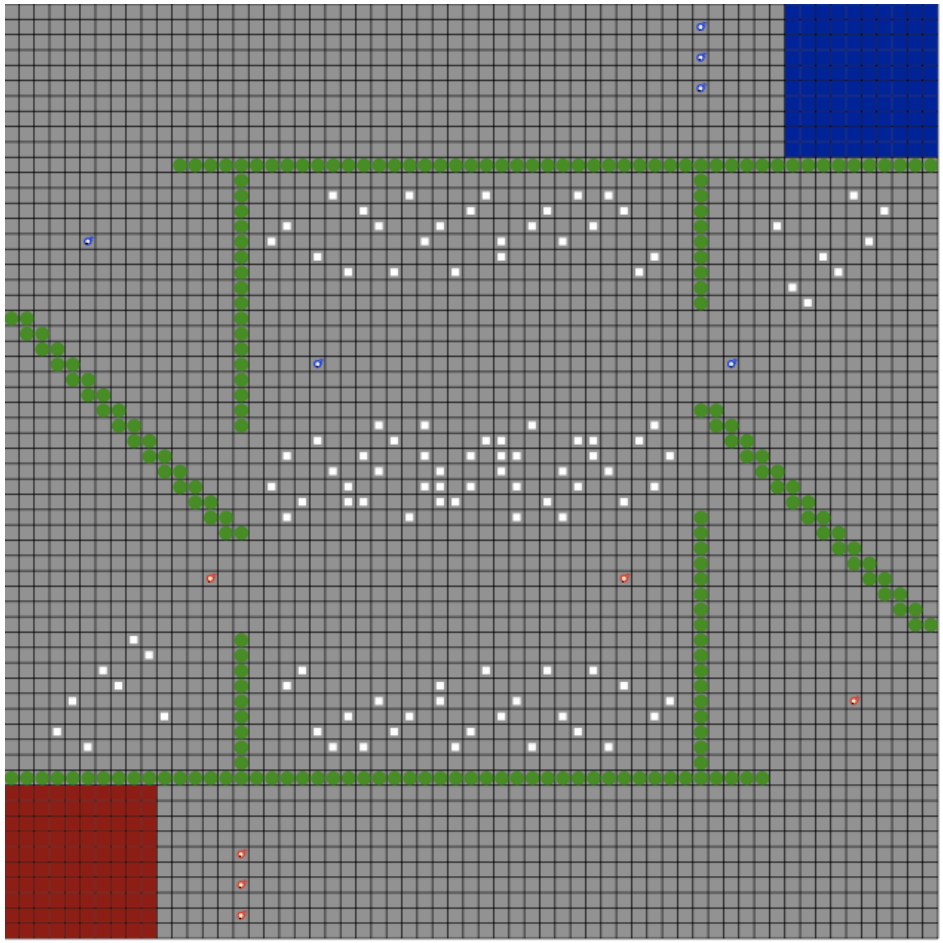
- fixed visibility range (square)
- actions: move to one of eight directions

Cows



- visibility range (square)
- afraid of: agents, obstacles
- feel good: near other cows and empty spaces
- actions: move to one of eight directions
- slower than agents

Map: Razoredge



Map: Cowskullmountain

Results in 2008

Rank	Team	CowScore	Points
1.	JIAC-TNG team	643	64
2.	Jadex	542	42
3.	SHABaN	373	37
4.	krzaczory	379	26
5.	Jason	393	21
6.	bogtrotters	305	13
7.	KANGAL	32	1

- Order of the games: Does it matter?
- Counting cows: Only at the end?
- Changing the strategy in the tournament by hand?
- Stealing cows: Foul play?

Game: JIAC vs Smaper

Game: Jadex vs JIAC. Introduction of fences.

Results in 2010

1. Brainbug, 57 points, (student course at DAI Lab, Berlin)
2. Cow Raiders, 48 points, (student course at DAI Lab, Berlin)
3. UCD Bogtrotters, 46 points, AgentFactory (Dublin)
4. Galoan, 36 points, (Iran, pure Java)
5. Argonauts, 29 points, Jason plus DLV (student course, Dortmund)
6. Jason DTU, 20 points, (Denmark)
7. PauLo, 11 points, pure Java, (Switzerland)
8. USPFarmers, 1 point, Cartago-Moise-Jason, (Brazil)

2. Environment Interface Standard (EIS)

Overview

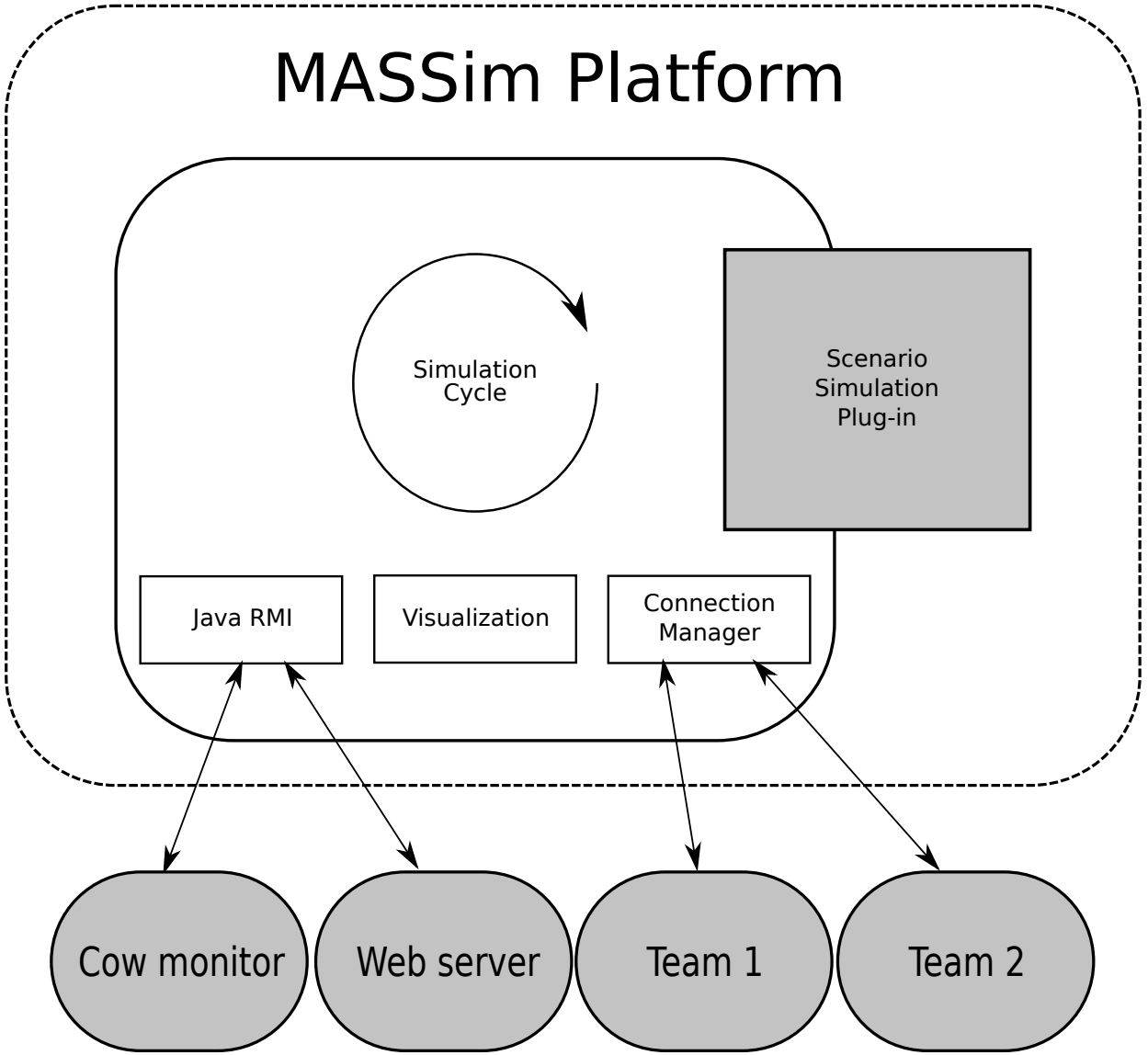


Figure : *MASSim* platform overview.

What exactly is an environment?

Are agents part of it? Is communication?

Environments: Many interesting environments are out there.
But for each APL they need to be re-programmed.

Standard: With a standard one single implementation would be enough.

Heterogeneity: A standard would also allow to connect agents from **different** agent platforms to participate in the same environment.

Overview

Domestic Robot

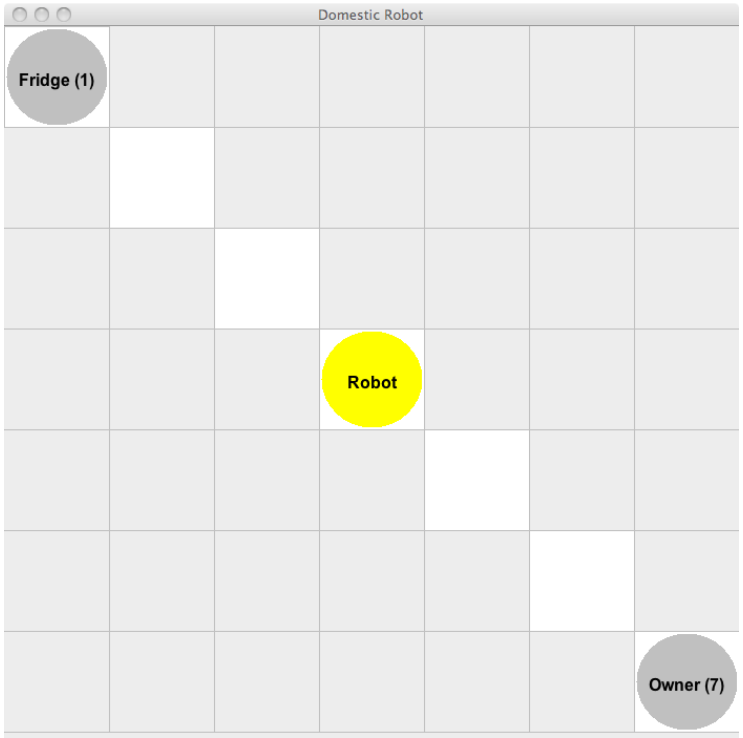


Figure : Domestic Robot

(EIS'ified by Jason developers)

Elevator

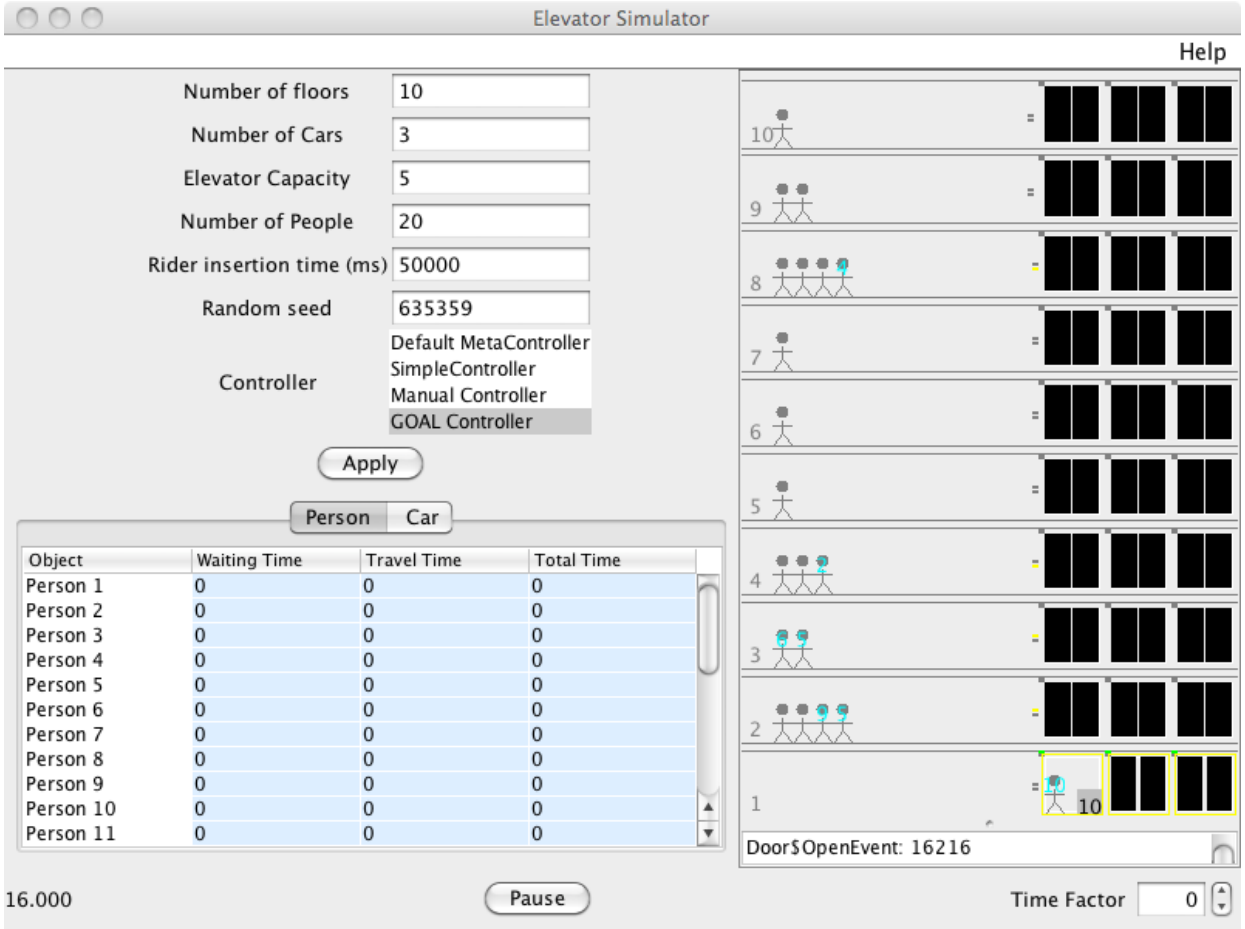


Figure : Elevator

(EIS'ified by Goal developers)

Vacuum World

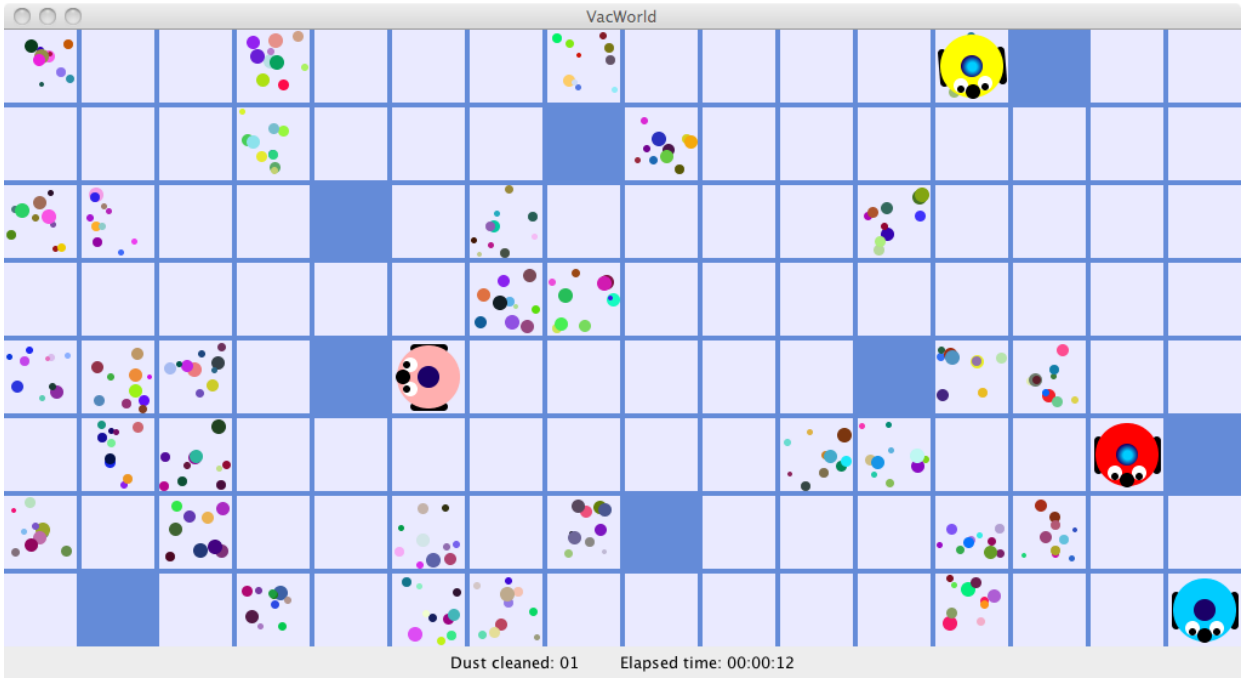


Figure : Vacuum World

(EIS'ified by Agent Factory developers)

Blocksworld

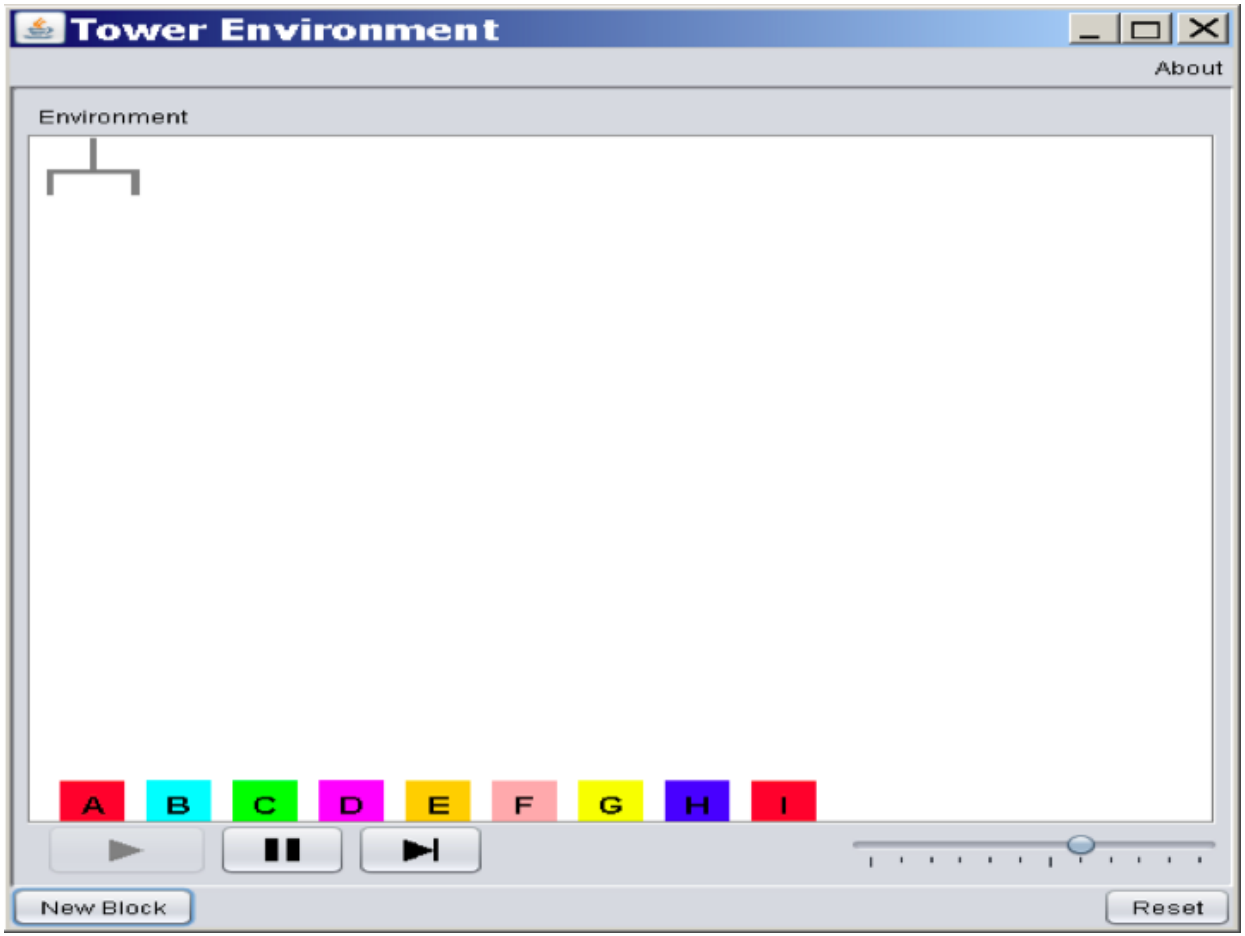


Figure : Blocksworld

(EIS'ified by Goal developers)

MASSim

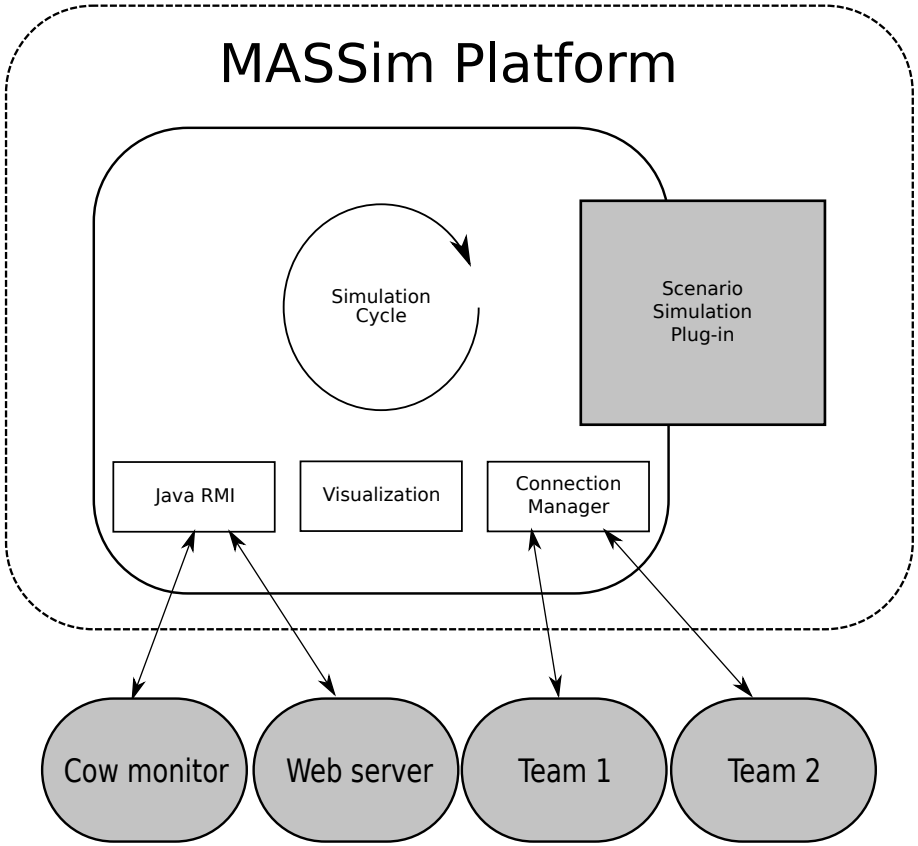


Figure : MASSim

(EIS'ified by Agent Contest developers)

Thus all scenarios supported by MASSim are available.

Unreal

Figure : Unreal Tournament

(EIS'ified by Goal developers)

Overview

Case Study UT

First person shooter game, fast pace

- right level of abstraction for agents to control bots
- low level details should be abstracted away
- Interface takes into account that decisions are made at a **knowledge or cognitive** level

Many other interfaces from agent platforms to UT exist. But the are all independent, no reuse, not well documented.

Principles

- **Portability**: Exchange of env. between platforms (eg jar files)
- **Generality**: minimal restrictions on platform or env.
- **Separation of concerns**: agents **are not** objects in the env, entities **are not** objects in the APL (EI just stores identifiers and relation)
- **Standards** for actions, percepts, events etc: define a language that represents each item as an abstract syntax tree.
- **Heterogeneity**: (1) run a central application containing the env. (2) provide a jar file based on EIS connecting the platforms to the env.

Ultimate Goal

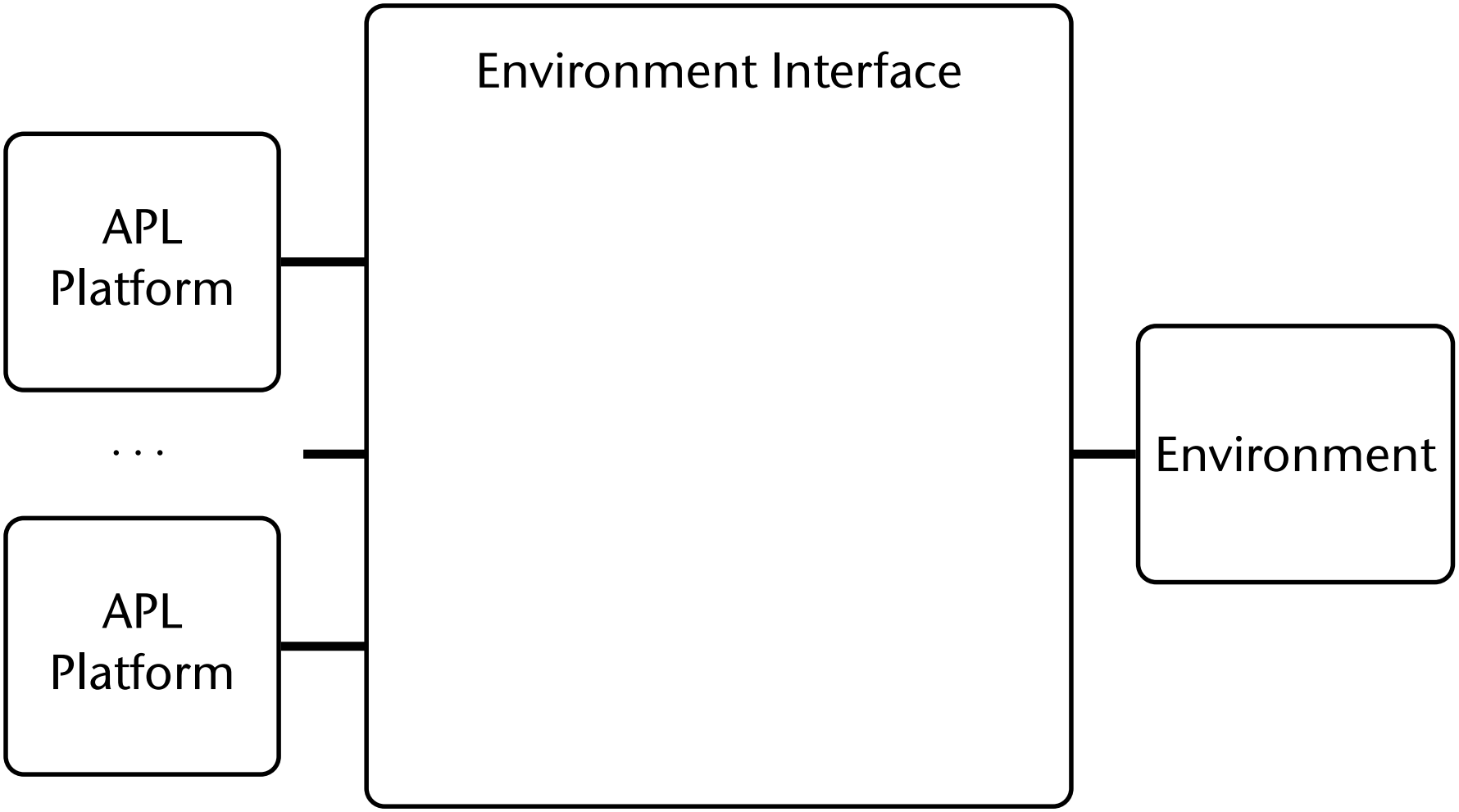


Figure : Agents from several platforms connected.

Comparison of several APL's

Criterion	2APL	GOAL	JADEX	JASON
Portability	jar-files	jar-files	everything	jar-files
Perceiving	sense-actions and external events	getting all percepts via a provided method	accessing env-objects or requesting percepts from an env-agent	getting all percepts via a provided method
Acting	invoking methods	invoking a method	manipulating an env object or sending a message to an env agent	invoking a method
Abstract Env Functionality	mapping from agent-names to agent-objects	no special functionality	no abstract env defined	logging and action-scheduling
Formats	logical terms and atom encoded as Java-objects	strings	java-objects	logical literals and structures encoded as Java-objects
Java accessibility	jar-files	jar-files	everything that is in the class-path	jar-files

Table : Comparison-matrix to give an overview.

Meta Model

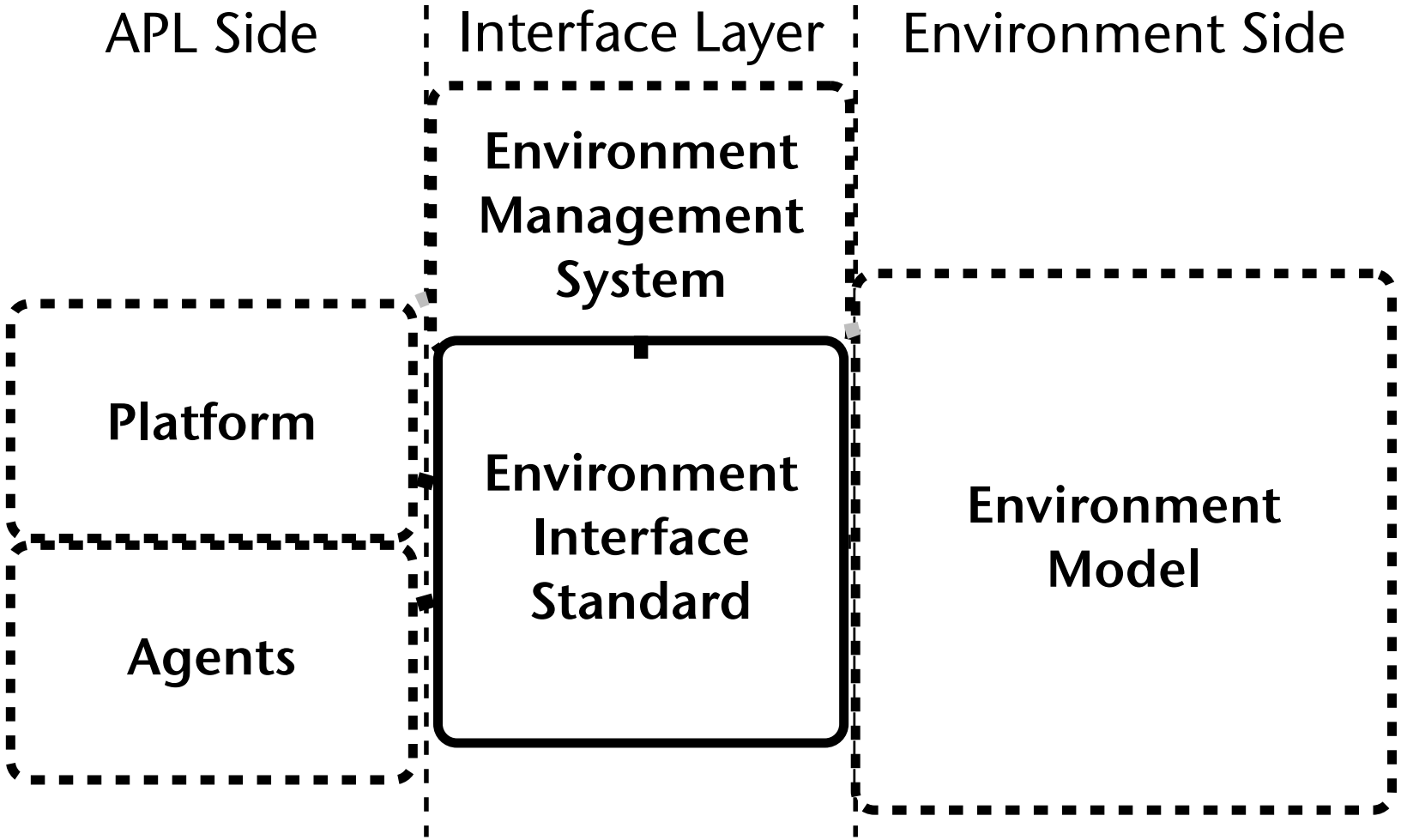


Figure : The interface layer defined by EIS acts as a kind of glue layer that facilitates the interaction of the components.

Meta Model

Agent: anything that perceives and acts

Env model: contains **controllable entities**, can be created or removed

Contr. entities: may be linked to concrete Java objects

EMS: actions to manage the env.: initialize, pause (important for debugging), unpause

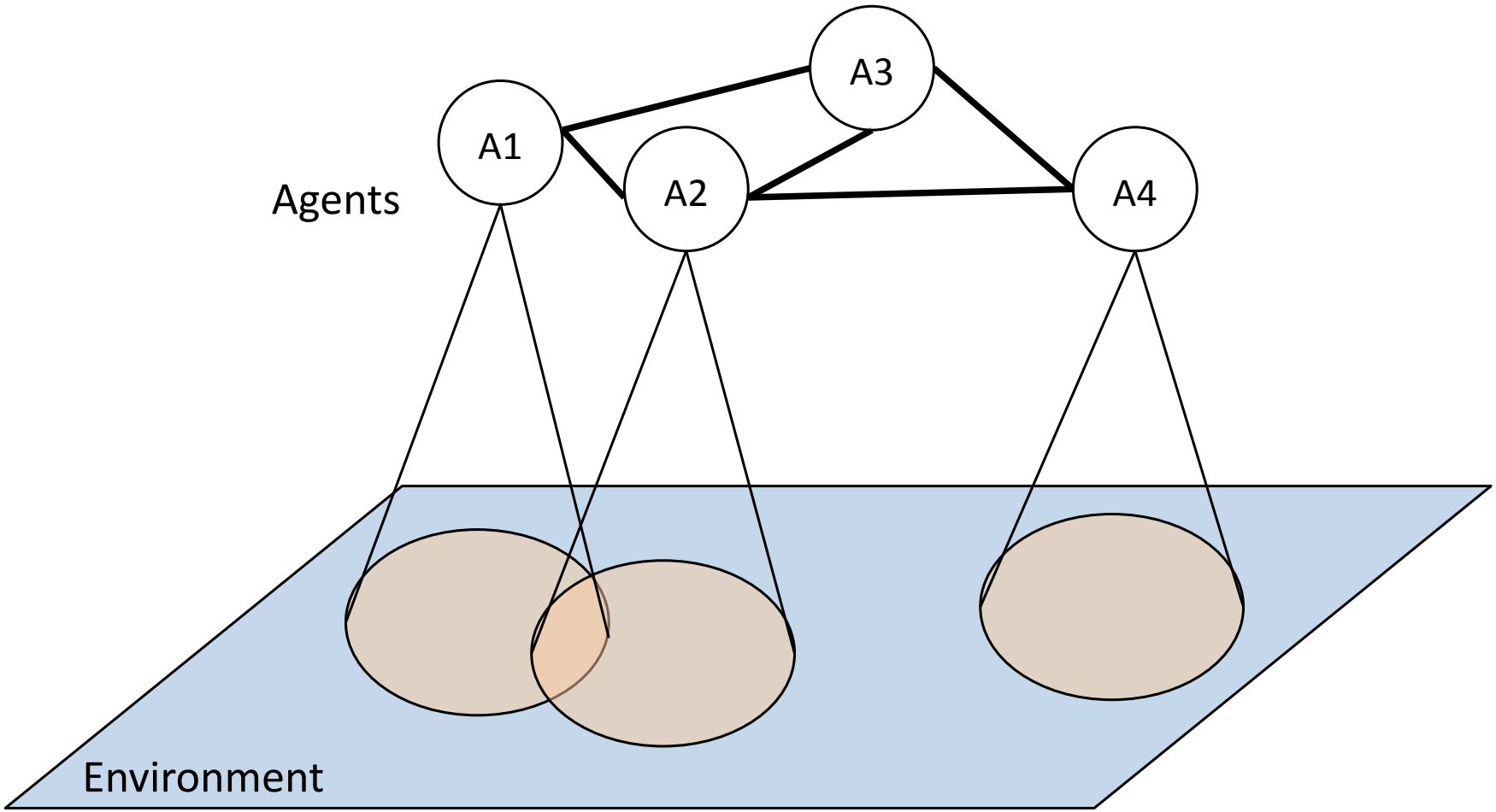


Figure : Environment-MAS Model.

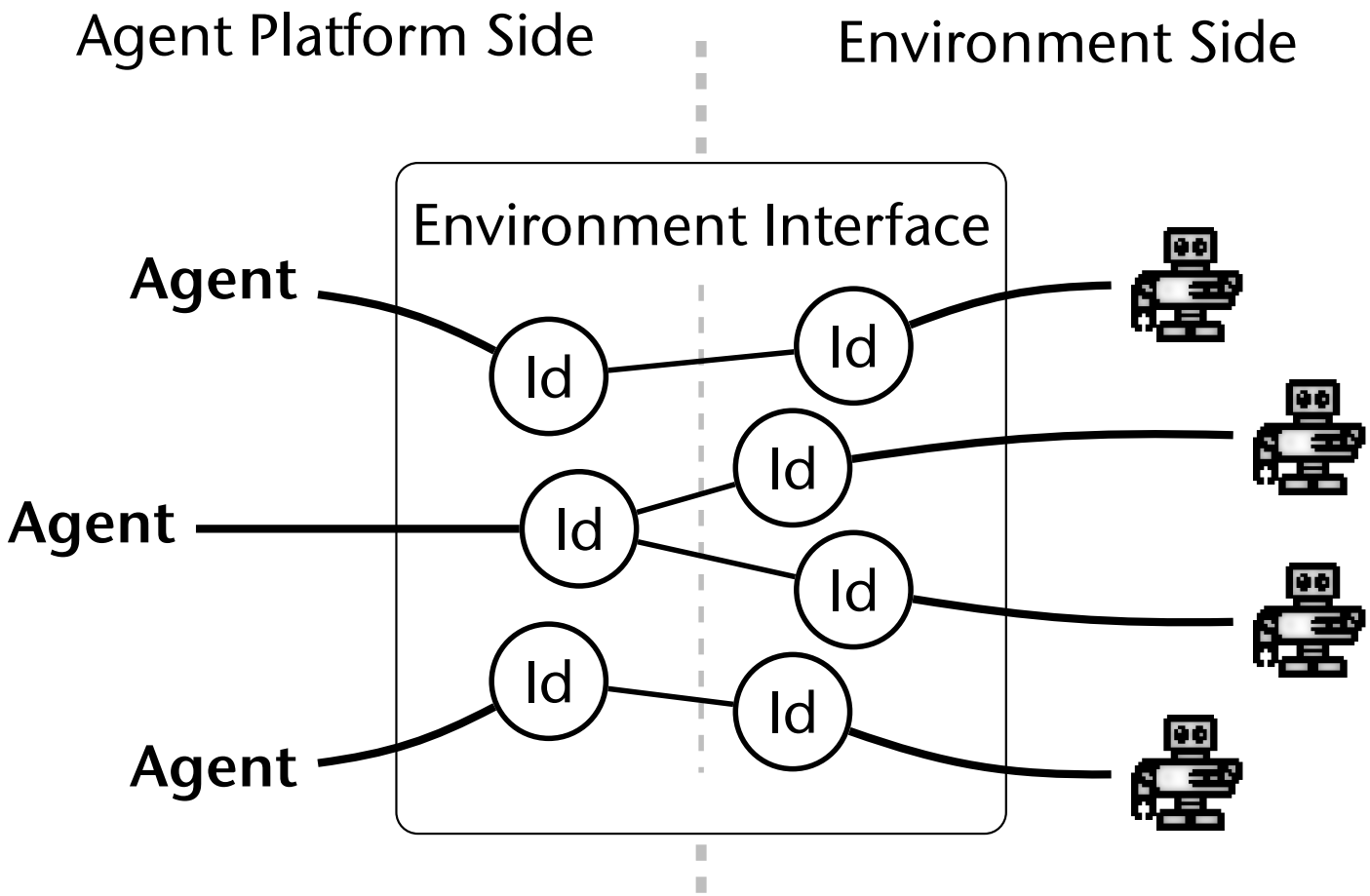


Figure : The agents-entities-relation.

All the links in the Meta Model (EMS to EIS, EIS to APL, EMS to ENV , etc) have to be defined in a generic way.

- Percepts: **active sensing** (part of the agent program), **passive sensing** (control cycle of the agent), perceptions sent automatically by the environment,
- interface intermediate language: convention of how to represent actions, percepts and events.

Interface should be agnostic to any implementation details of APL or ENV.

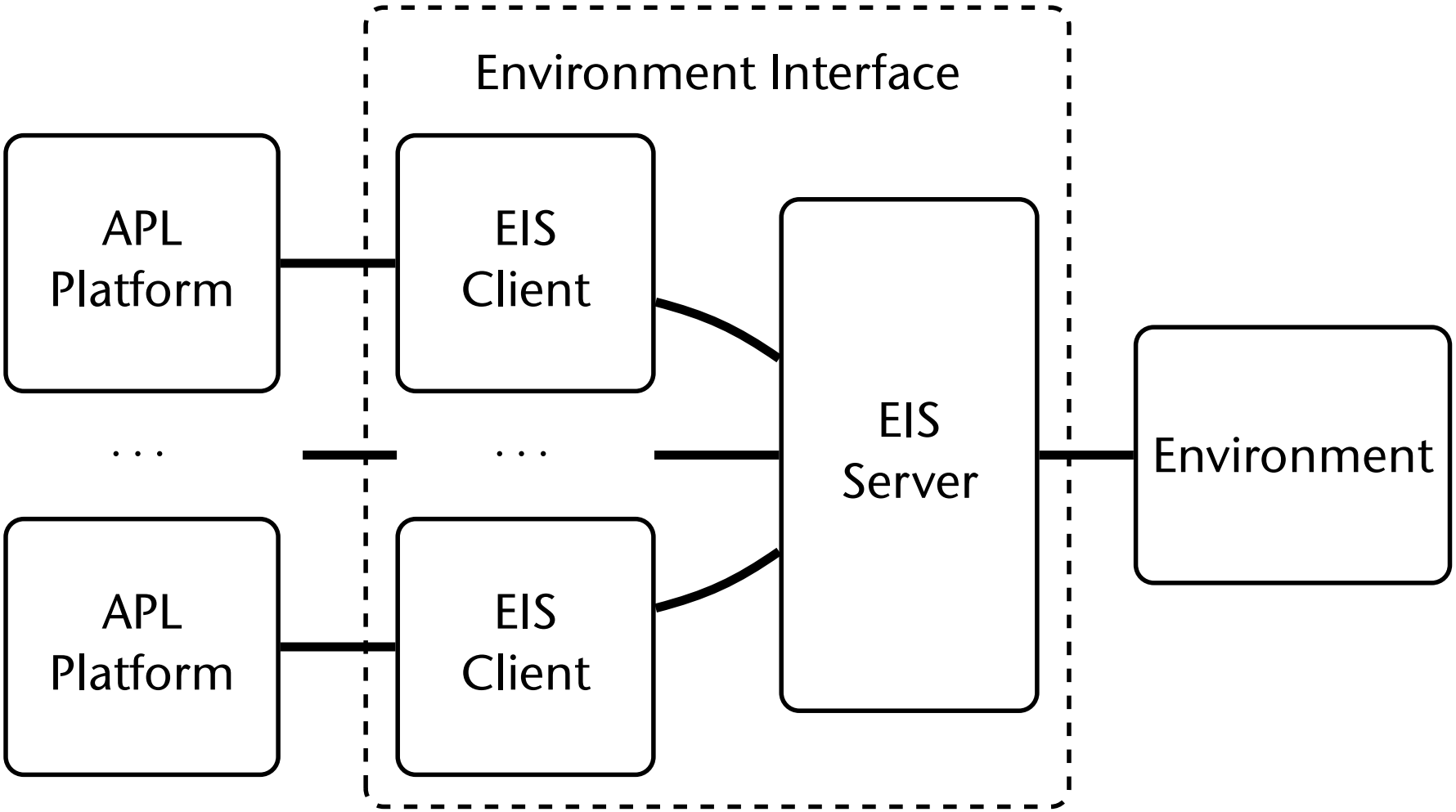


Figure : Distributed EIS with several processes.

Overview

More environments (soon to come)

LEGO Mindstorm: controlling robots in the physical world
(EIS'ified by Agent Factory developers)

OpenSim: virtual world similar to Second Life
(EIS'ified by Agent Factory developers)

RoboTennis: robots playing tennis
(EIS'ified by Agent Factory developers)

So far, the following APL's are connected to EIS:
GOAL, Jason, Jadex, Agent Factory, 2APL.

The A&A (Agents and Artifacts) model

Generic paradigm for modeling environments

Agents and artifacts: Environment is a **first-class abstraction for MAS engineering**, artifacts are **abstractions** to define env.-functionalities, as well as **entities** that are perceived, used and instantiated by agents.

CArtAgO: **implementation** of the A&A model, distributed middleware infrastructure open source

3. New Scenario

3 New Scenario

Agents on Mars

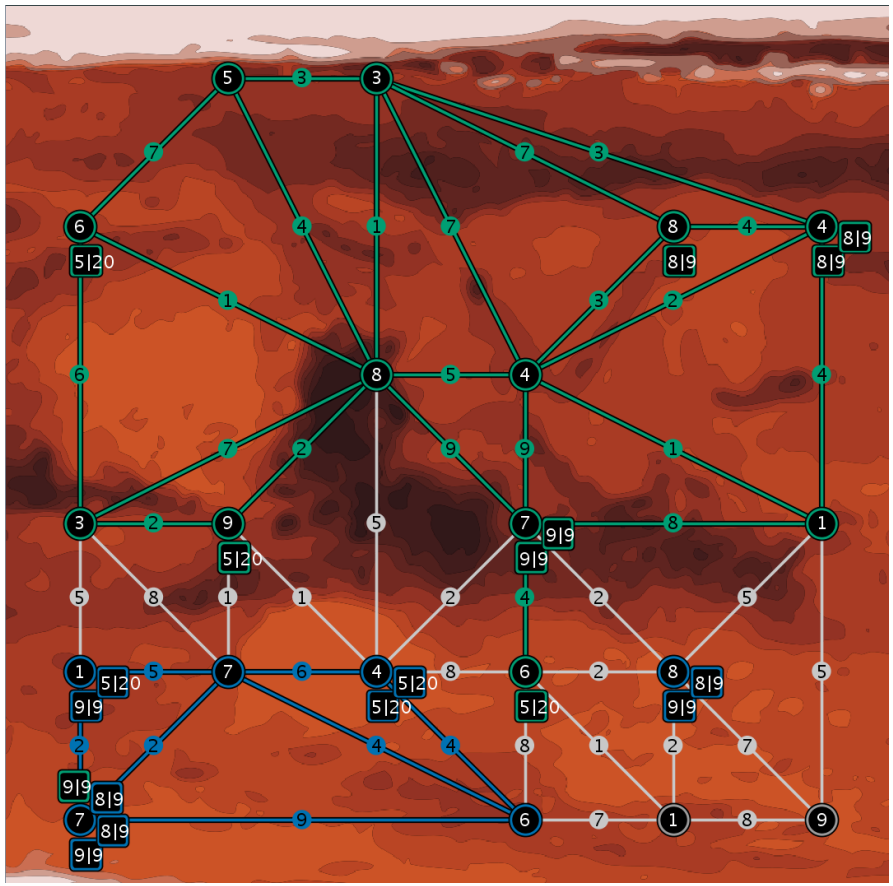
Focus on:

- **agent cooperation** and **agent coordination**
- **team decentralization**

Challenge

Occupy the biggest zones and earn a lot of money!

$$\texttt{score} = \sum_{s=1}^{\texttt{steps}} (\texttt{zones}_s + \texttt{money}_s)$$



Teams & All Terrain Planetary Vehicles

- Explorer: `skip`, `goto`, `probe`, `survey`, `buy`, `recharge`
  Energy: 12 Health: 4 Strength: 0 Visibility range: 2

Teams & All Terrain Planetary Vehicles

- Explorer: `skip`, `goto`, `probe`, `survey`, `buy`, `recharge`
  Energy: 12 Health: 4 Strength: 0 Visibility range: 2
- Repairer: `skip`, `goto`, `parry`, `survey`, `buy`, `repair`, `recharge`
  Energy: 8 Health: 6 Strength: 0 Visibility range: 1

Teams & All Terrain Planetary Vehicles

- Explorer: `skip`, `goto`, `probe`, `survey`, `buy`, `recharge`
  Energy: 12 Health: 4 Strength: 0 Visibility range: 2
- Repairer: `skip`, `goto`, `parry`, `survey`, `buy`, `repair`, `recharge`
  Energy: 8 Health: 6 Strength: 0 Visibility range: 1
- Saboteur: `skip`, `goto`, `parry`, `survey`, `buy`, `attack`, `recharge`
  Energy: 7 Health: 3 Strength: 4 Visibility range: 1

Teams & All Terrain Planetary Vehicles

- Explorer: `skip`, `goto`, `probe`, `survey`, `buy`, `recharge`
  Energy: 12 Health: 4 Strength: 0 Visibility range: 2
- Repairer: `skip`, `goto`, `parry`, `survey`, `buy`, `repair`, `recharge`
  Energy: 8 Health: 6 Strength: 0 Visibility range: 1
- Saboteur: `skip`, `goto`, `parry`, `survey`, `buy`, `attack`, `recharge`
  Energy: 7 Health: 3 Strength: 4 Visibility range: 1
- Sentinel: `skip`, `goto`, `parry`, `survey`, `buy`, `recharge`
  Energy: 10 Health: 1 Strength: 0 Visibility range: 3

Teams & All Terrain Planetary Vehicles

- Explorer: `skip`, `goto`, `probe`, `survey`, `buy`, `recharge`
  Energy: 12 Health: 4 Strength: 0 Visibility range: 2
- Repairer: `skip`, `goto`, `parry`, `survey`, `buy`, `repair`, `recharge`
  Energy: 8 Health: 6 Strength: 0 Visibility range: 1
- Saboteur: `skip`, `goto`, `parry`, `survey`, `buy`, `attack`, `recharge`
  Energy: 7 Health: 3 Strength: 4 Visibility range: 1
- Sentinel: `skip`, `goto`, `parry`, `survey`, `buy`, `recharge`
  Energy: 10 Health: 1 Strength: 0 Visibility range: 3
- Inspector: `skip`, `goto`, `inspect`, `survey`, `buy`, `recharge`
  Energy: 8 Health: 6 Strength: 0 Visibility range: 1

Money

Achievements:

- Having zones with fixed values, e.g. 10 or 20,
- Fixed numbers of probed vertices, e.g. 5 or 10,
- Fixed numbers of surveyed edges, e.g. 10 or 20,
- Fixed numbers of inspected vehicles, e.g. 5 or 10,
- Fixed numbers of successful attacks, e.g. 5 or 10, or
- Fixed numbers of successful parries, e.g. 5 or 10.

Percepts

In each step, the agents get these percepts:

- Current step,
- Current scores and money,
- Agents internals,
- Visible vertices,
- Visible edges,
- Visible vehicles,
- Probed vertices,
- Surveyed edges,
- Inspected vehicles.

Disabled Agents

Agents with health zero are disabled:

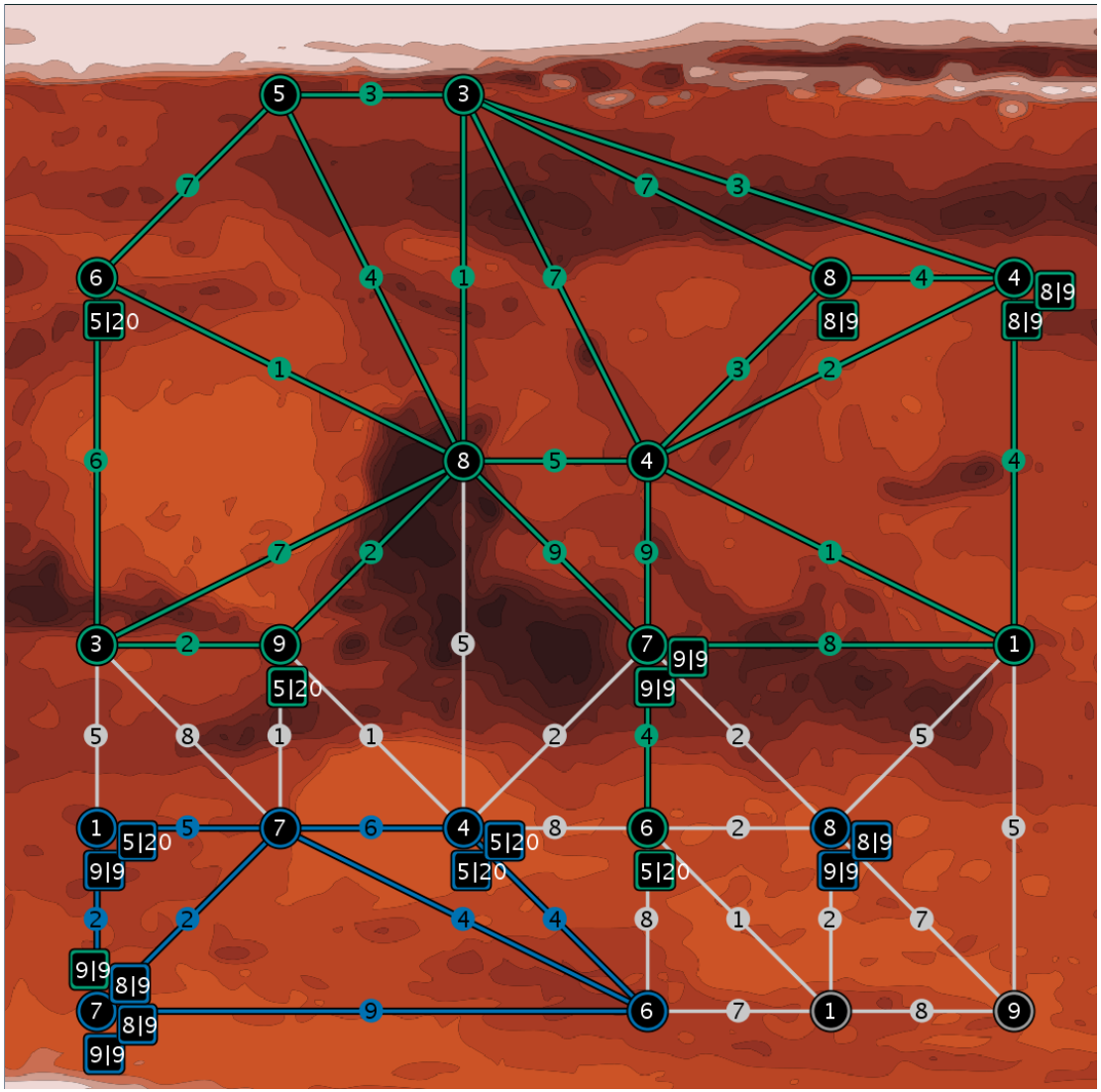
- Only the action `goto`, `repair`, `skip` are executable
- The `recharge` rate is set to 10 percent.

Simulation State Transition

The simulation state transition is as follows:

- Collect all actions from the agents,
- Let each action fail with a specific probability,
- Execute all remaining `attack` and `parry` actions,
- Determine disabled agents,
- Execute all remaining actions,
- Compute new percepts,
- Send the percepts out to the agents.

New Scenario



Consider entering the contest!!

Software Package: `http://multiagentcontest.org/2011`

- MASSim-Server including the new Agents-on-Mars-scenario,
- Monitor for inspecting and visualizing the environment,
- Java-based environment-interface that facilitates connecting to the server,
- Set of simple dummy-agents for testing purposes, and
- Detailed documentation on all components of the package.

Use EIS and MASSim for your lectures!

Download EIS and our MASSIM Server for the class room.

Ideally suited for a practical course on MAS: dummy agents in several agent languages available, environments for free using EIS.

Students develop agents in teams and play against each other at the end of the course.

Thank you for your attention!

Agentcontest: **Annals of Math and AI**, Special Issue on Agentcontest, Volume 59, issue 3/4, 2010

EIS: http://sourceforge.net/projects/apleis/, http://cig.in.tu-clausthal.de/eis
Annals of Math and AI, **59(4)**, 2011