CM30174 + CM50206 Intelligent Agents

Thomas Smith

East Building

December 8, 2013

Paper Overview

"Towards an environment interface standard for agent platforms"

Tristan M. Behrens, Koen V. Hindriks, Jürgen Dix

Annals of Mathematics and Artificial Intelligence (2011), 61:261–295.

Problem Overview

Issues:

- There are many Agent Programming Languages (*APL*s)
  - 2APL
  - GOAL
  - JADEX
  - *Jason*
- Multiple Environments
  - Agent Competitions
  - Unreal Tournament 2004
- How can we connect / compare these components?
  - Connections: TCP/IP, RMI, wrapping Java code, JNI
  - Comparisons: we can't.

Goals

- Implementing an environment interface standard would:
  - make already working environments widely available
  - facilitate distribution of current and future environments
  - allow direct comparison of APL platforms
  - enable the develpment of a fully heterogeneous multi-agent system (*MAS*)
- In order to be accepted as a standard, it should:
  - Provide an interface that is as *generic as possible*
  - *Reuse as much as possible* from existing interfaces
- Therefore the objective is to:

  > "*Design and develop an environment interface standard (EIS) that facilitates connecting agents programmed in various agent platforms to environments.*"

Goals (contd.)

- No assumptions need to be made about the internal structure or behaviour of any of the environments or entities
- However, the agent platform needs to support a minimal agent-based abstraction:
  Actions and percepts are treated as first-class entities.
- The standard is based on:
  - A meta-model for agent-environment interaction
  - A set of principles that encode useful constraints for implementing the standard
- The meta-model arises from a study of existing APLs, and avoids restricting existing approaches
- The princples are based on the requirements of the project, and observed best practices among existing work

Principles

1. *Portability*: .jar files are suggested but not required, for easy exchange of environments between platforms
2. *Generality*: The *EIS* should impose minimal restrictions on the platform or environments. Assumptions about the agents or the environments should be avoided
3. *Separation of concerns*: Agents are *separated* from the environment
4. *Unified connections*: The *EIS* should facilitate communications between agents and the environment(s)
5. *Standards for actions/percepts/events/etc.*: The *EIS* should provide a convention for communicataing about concepts, without restricting any existing approach
6. *Support for heterogeneity*: The interface standard needs to facilitate heterogeneity - connections between an environment and agents of multiple types

Related Work

There are a number of other projects that support generic connections between agents and an environment:

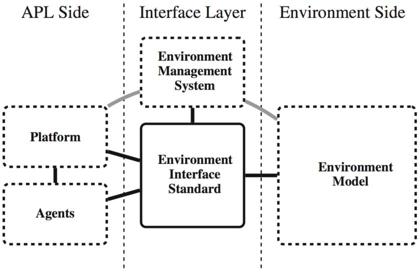
- A&A: a generic paradigm for modeling environments. Implemented in the distributed middleware CArtAgO
- Soar is an architecture for knowledge-rich agents capable of intelligent behavior in dynamic environments
- GameBots and Pogamut are a pair of projects designed to allow agents to control bots in UT2004. A number of other projects use them as an initial base
    - pyPOSH aims to use Behaviour Oriented Design agents
    - the ACT-R cognitive architecture uses GameBots
- The high level architecture (HLA) is a federated architecture for distributed simulations
- The *UtJackInterface* defines another UT2004 interface from scratch - none of these approaches facilitate reuse

Existing Agent Programming Languages

- A number of existing APLs indicate common and uncommon features that the meta-model must support
  - 2APL
  - GOAL
  - JADEX
  - *Jason*
- Though each APL is designed to fulfil similar purposes, they vary in implementation details
  - 2APL provides a common format for exchanging data between agents and the environment
  - GOAL uses a scheduler to manage execution of agents
  - JADEX store environments as beliefs of the agents
  - *Jason* provides spohisticated abstract environments
- The different APLs have differing degrees of environment management functionality available

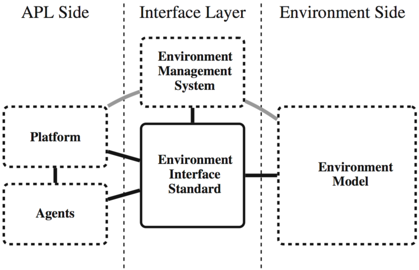
Existing Environments

Meta-model



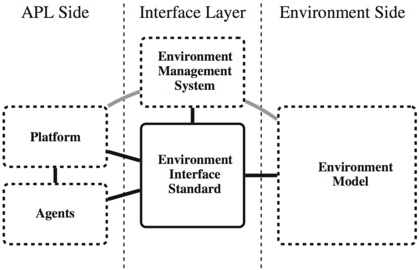
1. *Agent*: able to perceive its environment through sensors and act upon that environment through effectors.

Meta-model



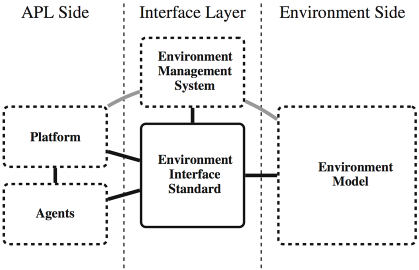
1. *Agent*
2. *Environment model*: contains *controllable entities* that give agents effective and sensory presence in the environment.

Meta-model



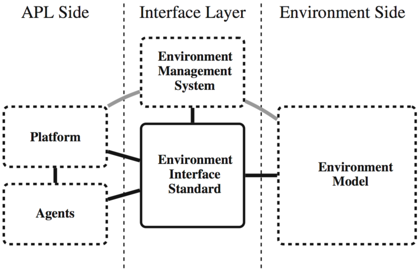
1. *Agent*
2. *Environment model*
3. *Platform*: instantiates and executes agents; connects agents to the environment and controllable entities.

Meta-model



1. *Agent*
2. *Environment model*
3. *Platform*
4. *Environment management system (EMS)*: provides actions for managing an environment, such as setup, pause and reset.

Meta-model

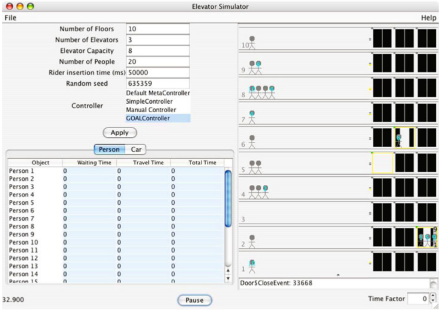


1. *Agent*
2. *Environment model*
3. *Platform*
4. *Environment management system (EMS)*
5. *Environment interface standard (EIS)*: the layer that connects the platform, the EMS, and the agents to the environment(s).

Interface Immediate Language

Implementation

Case Study: Elevator



Number of Floors: 10
Number of Elevators: 3
Elevator Capacity: 8
Number of People: 20
Rider insertion time (ms): 50000
Random seed: 635359
Controller: Default MetaController / SimpleController / Manual Controller / GOALController (selected)

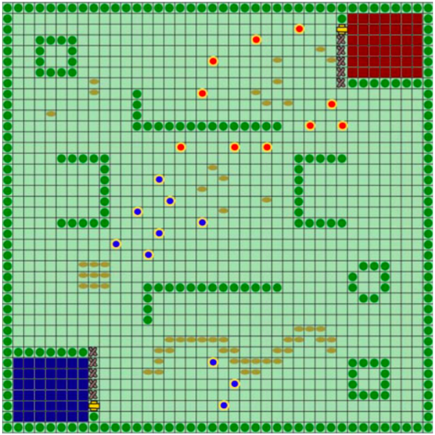
Object	Waiting Time	Travel Time	Total Time
Person 1	0	0	0
Person 2	0	0	0
Person 3	0	0	0
Person 4	0	0	0
Person 5	0	0	0
Person 6	0	0	0
Person 7	0	0	0
Person 8	0	0	0
Person 9	0	0	0
Person 10	0	0	0
Person 11	0	0	0
Person 12	0	0	0
Person 13	0	0	0
Person 14	0	0	0
Person 15	0	0	0

DoorSCloseEvent: 33668

32.900

Time Factor 0

Case Study: Agent Contest



Case Study: Unreal Tournament

Summary

standard functionality is provided by the interface implementation itself agent platforms that support the interface can connect to any environment that implements the interface