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:

Problem Overview

Issues:

- There are many Agent Programming Languages (*APL*s)

Problem Overview

Issues:

- There are many Agent Programming Languages (*APLs*)
  - 2APL
  - GOAL
  - JADEX
  - *Jason*

Problem Overview

Issues:

- There are many Agent Programming Languages (*APLs*)
  - 2APL
  - GOAL
  - JADEX
  - *Jason*
- Multiple varied environments are used

Problem Overview

Issues:

- There are many Agent Programming Languages (*APLs*)
  - 2APL
  - Goal
  - Jadex
  - *Jason*
- Multiple varied environments are used
  - Trading Agent Competition
  - MASSim Agent Competitions
  - Unreal Tournament 2004

Problem Overview

Issues:

- There are many Agent Programming Languages (*APLs*)
  - 2APL
  - Goal
  - Jadex
  - *Jason*
- Multiple varied environments are used
  - Trading Agent Competition
  - MASSim Agent Competitions
  - Unreal Tournament 2004
- How can we connect / compare these components?

Problem Overview

Issues:

- There are many Agent Programming Languages (*APLs*)
  - 2APL
  - GOAL
  - JADEX
  - *Jason*
- Multiple varied environments are used
  - Trading Agent Competition
  - MASSim Agent Competitions
  - Unreal Tournament 2004
- How can we connect / compare these components?
  - Connections: TCP/IP, RMI, wrapping Java code, JNI

Problem Overview

Issues:

- There are many Agent Programming Languages (*APLs*)
  - 2APL
  - Goal
  - Jadex
  - *Jason*
- Multiple varied environments are used
  - Trading Agent Competition
  - MASSim 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:

Goals

- Implementing an environment interface standard would:
  - make already working environments widely available

Goals

- Implementing an environment interface standard would:
  - make already working environments widely available
  - facilitate distribution of current and future environments

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

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 development of a fully heterogeneous multi-agent system (*MAS*)

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 development of a fully heterogeneous multi-agent system (*MAS*)
- In order to be accepted as a standard, it should:

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 development 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*

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 development 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

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 development 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:

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 development 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.)

- Ideally, no assumptions should be made about the internal structure or behaviour of any of the environments or entities

Goals (contd.)

- Ideally, no assumptions should 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.

Goals (contd.)

- Ideally, no assumptions should 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:

Goals (contd.)

- Ideally, no assumptions should 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

Goals (contd.)

- Ideally, no assumptions should 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

Goals (contd.)

- Ideally, no assumptions should 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

Goals (contd.)

- Ideally, no assumptions should 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 principles arise from the requirements of the project, and observed best practices in existing research

Principles

1. *Portability*: .jar files are suggested but not required, for easy exchange of environments between platforms

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

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, agents *map to* controllable entities

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, agents *map to* controllable entities
4. *Unified connections*: The *EIS* should facilitate communications between agents and the environment(s)

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, agents *map to* controllable entities
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 communicating about concepts, without restricting any existing approach

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, agents *map to* controllable entities
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 communicating about concepts, without restricting any existing approach
6. *Support for heterogeneity*: The *EIS* 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:

Related Work

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

- A&A: a generic paradigm for modelling environments. Implemented in the distributed middleware CArtAgO

Related Work

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

- A&A: a generic paradigm for modelling environments. Implemented in the distributed middleware CArtAgO
- Soar is an architecture for knowledge-rich agents capable of intelligent behaviour in dynamic environments

Related Work

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

- A&A: a generic paradigm for modelling environments. Implemented in the distributed middleware CArtAgO
- Soar is an architecture for knowledge-rich agents capable of intelligent behaviour 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

Related Work

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

- A&A: a generic paradigm for modelling environments. Implemented in the distributed middleware CArtAgO
- Soar is an architecture for knowledge-rich agents capable of intelligent behaviour 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

Related Work

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

- A&A: a generic paradigm for modelling environments. Implemented in the distributed middleware CArtAgO
- Soar is an architecture for knowledge-rich agents capable of intelligent behaviour 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

Related Work

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

- A&A: a generic paradigm for modelling environments. Implemented in the distributed middleware CArtAgO
- Soar is an architecture for knowledge-rich agents capable of intelligent behaviour 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

Related Work

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

- A&A: a generic paradigm for modelling environments. Implemented in the distributed middleware CArtAgO
- Soar is an architecture for knowledge-rich agents capable of intelligent behaviour 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

Existing Agent Programming Languages

- A number of existing APLs indicate common and uncommon features that the meta-model must support

Existing Agent Programming Languages

- A number of existing APLs indicate common and uncommon features that the meta-model must support
  - 2APL
  - Goal
  - Jadex
  - *Jason*

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

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

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

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

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 sophisticated abstract environments

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 sophisticated abstract environments
- The different APLs have differing degrees of environment management functionality available

Existing Environments

- Elevator environment

  The elevator environment is a simulator of arbitrary multi-elevator environments where elevators are controllable entities that agents can manage

Existing Environments

- Elevator environment

  The elevator environment is a simulator of arbitrary multi-elevator environments where elevators are controllable entities that agents can manage

- The Trading Agent Competition

  The TAC is an annual competition run on a research testbed for autonomous bidding agents. The interface uses TCP/IP and is time-sensitive

Existing Environments

- Elevator environment

  The elevator environment is a simulator of arbitrary multi-elevator environments where elevators are controllable entities that agents can manage

- The Trading Agent Competition

  The TAC is an annual competition run on a research testbed for autonomous bidding agents. The interface uses TCP/IP and is time-sensitive

- MASSim

  The *AgentContest* environment is a discrete simulation that relies on co-operation and co-ordination between multiple agents. Communication uses XML wrappers

Existing Environments

- Elevator environment

  The elevator environment is a simulator of arbitrary multi-elevator environments where elevators are controllable entities that agents can manage

- The Trading Agent Competition

  The TAC is an annual competition run on a research testbed for autonomous bidding agents. The interface uses TCP/IP and is time-sensitive

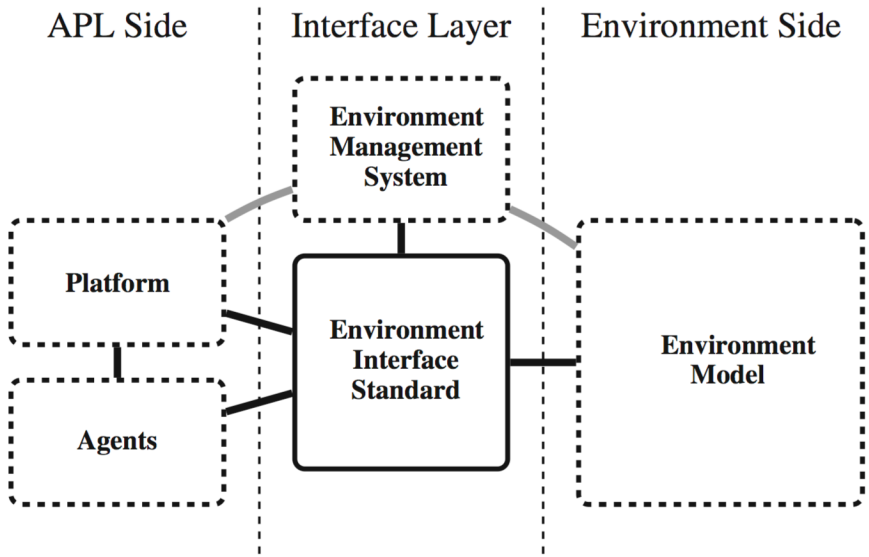
- MASSim

  The *AgentContest* environment is a discrete simulation that relies on co-operation and co-ordination between multiple agents. Communication uses XML wrappers

- Unreal Tournament 2004

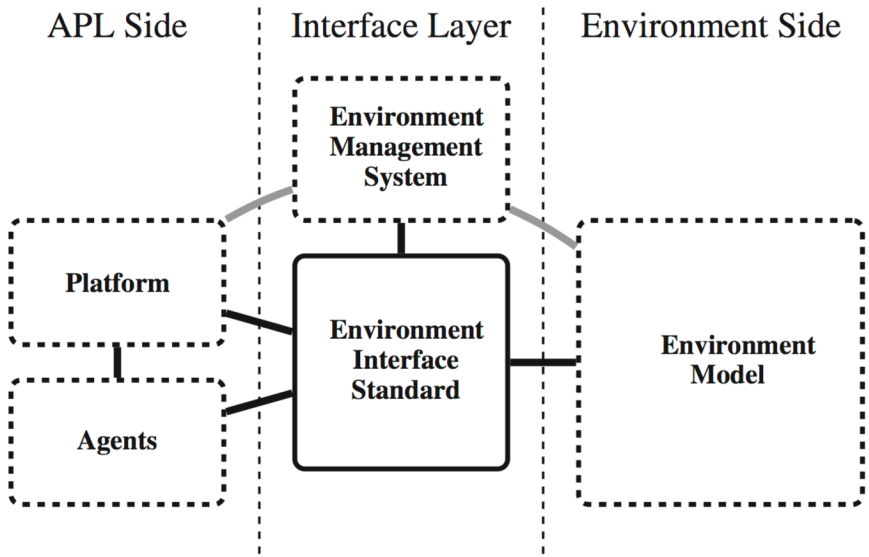
  UT2004 is a popular real-time testbed for continuous, dynamic multi-agent interaction environments. High- or low-level interaction is possible

Meta-model



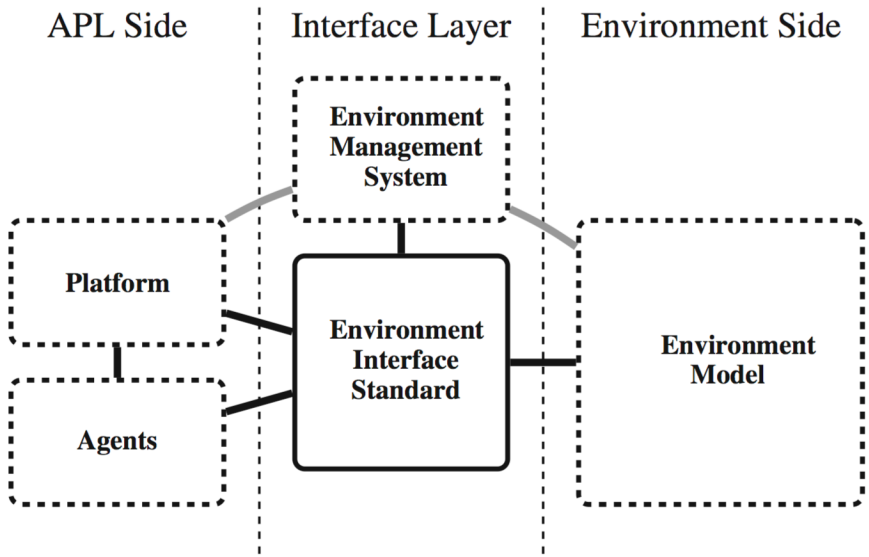
1. *Agent*: able to perceive its environment through sensors and act upon that environment through effectors.

Meta-model



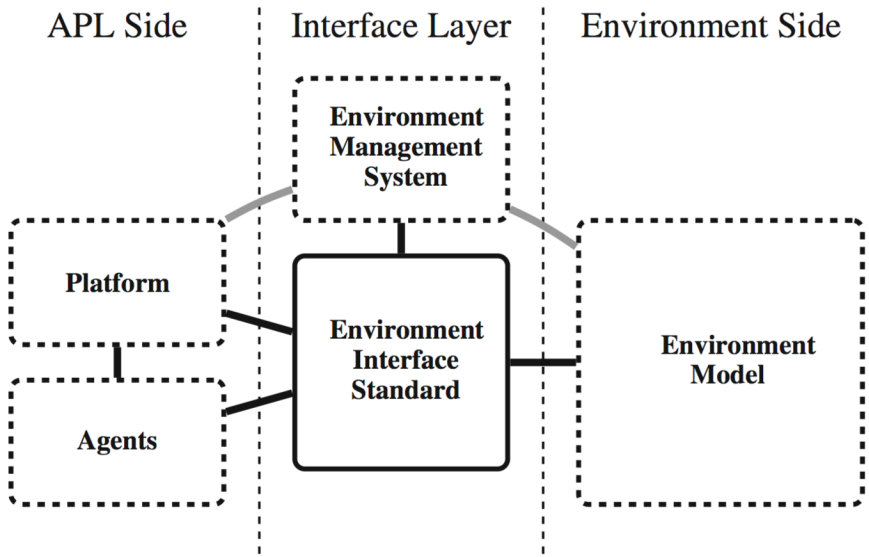
1. *Agent*
2. *Environment model*: contains *controllable entities* that give agents effective and sensory presence in the environment.

Meta-model



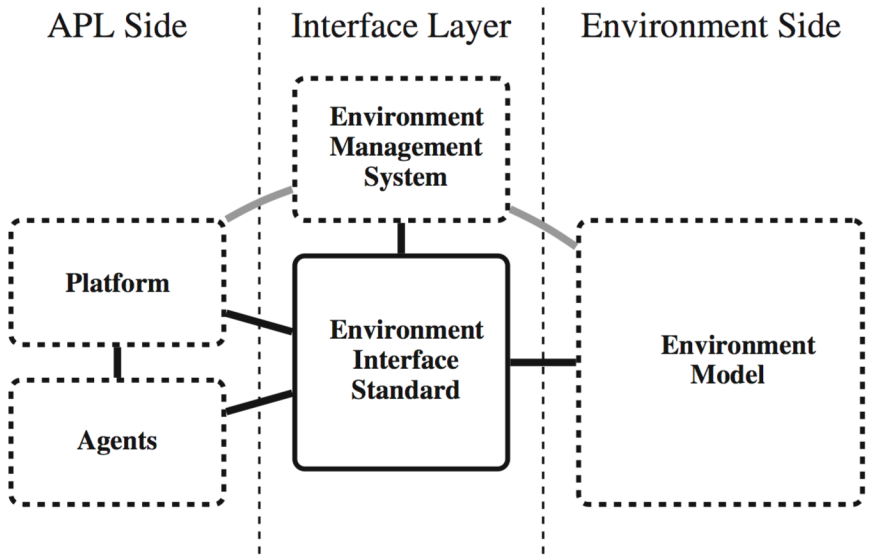
1. *Agent*
2. *Environment model*
3. *Platform*: instantiates and executes agents; connects agents to the environment and controllable entities.

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

- The interface immediate language (*IIL*) facilitates the exchange of data between different components

Interface Immediate Language

- The interface immediate language (*IIL*) facilitates the exchange of data between different components
- It provides an implementation-agnostic method for communicating about percepts, actions, and events

Interface Immediate Language

- The interface immediate language (*IIL*) facilitates the exchange of data between different components
- It provides an implementation-agnostic method for communicating about percepts, actions, and events
- The language consists of:

Interface Immediate Language

- The interface immediate language (*IIL*) facilitates the exchange of data between different components
- It provides an implementation-agnostic method for communicating about percepts, actions, and events
- The language consists of:
  - *Data containers*, which represent *actions* performed by agents, *results* of actions, and *percepts*. There are also *environment commands* that control the execution of the environment

Interface Immediate Language

- The interface immediate language (*IIL*) facilitates the exchange of data between different components
- It provides an implementation-agnostic method for communicating about percepts, actions, and events
- The language consists of:
  - *Data containers*, which represent *actions* performed by agents, *results* of actions, and *percepts*. There are also *environment commands* that control the execution of the environment
  - Data container *parameters*, which are either constant numerals and identifiers or functions over (lists of) parameters

Interface Immediate Language

- The interface immediate language (*IIL*) facilitates the exchange of data between different components
- It provides an implementation-agnostic method for communicating about percepts, actions, and events
- The language consists of:
  - *Data containers*, which represent *actions* performed by agents, *results* of actions, and *percepts*. There are also *environment commands* that control the execution of the environment
  - Data container *parameters*, which are either constant numerals and identifiers or functions over (lists of) parameters
- Example server connection code:

```
action(connect,agentsim1,team1,pa55w0rd)
```

Implementation

- Two-way connections are provided via interactions performed by the components and notifications performed by the *EIS*

Implementation

- Two-way connections are provided via interactions performed by the components and notifications performed by the *EIS*
  - Interactions are (possibly remote) function calls to the *EIS*, that can return a value

Implementation

- Two-way connections are provided via interactions performed by the components and notifications performed by the *EIS*
  - Interactions are (possibly remote) function calls to the *EIS*, that can return a value
  - Notifications follow the observer design pattern, components are *listeners* that have callback functions

Implementation

- Two-way connections are provided via interactions performed by the components and notifications performed by the *EIS*
  - Interactions are (possibly remote) function calls to the *EIS*, that can return a value
  - Notifications follow the observer design pattern, components are *listeners* that have callback functions
- For agents, the two methods correspond to the notions of *polling* and *interrupts*

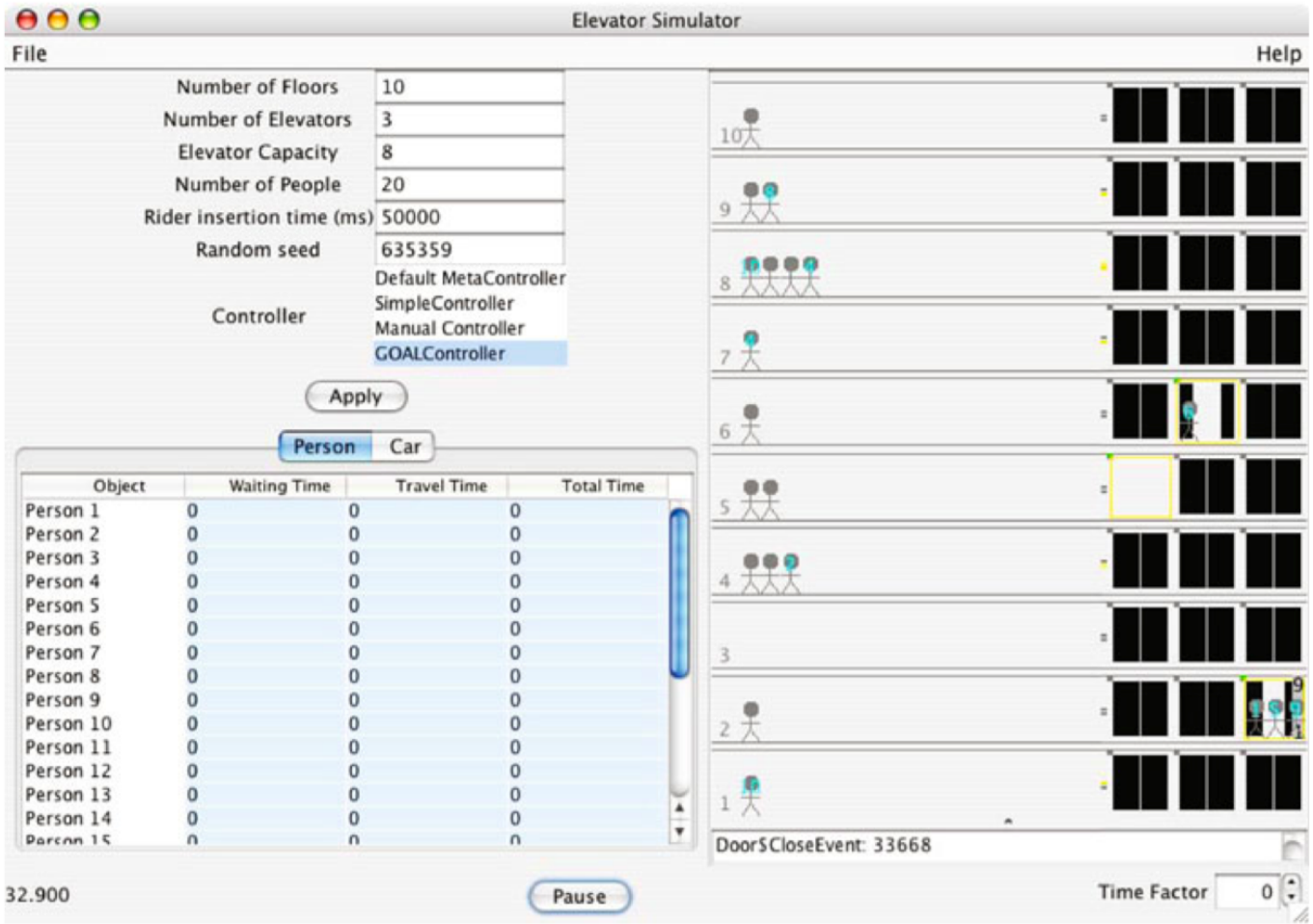
Implementation

- Two-way connections are provided via interactions performed by the components and notifications performed by the *EIS*
  - Interactions are (possibly remote) function calls to the *EIS*, that can return a value
  - Notifications follow the observer design pattern, components are *listeners* that have callback functions
- For agents, the two methods correspond to the notions of *polling* and *interrupts*
- Agents exist on the platform side, controllable entities exist in the environment, and agents may control entities through the *EIS* mapping between sets

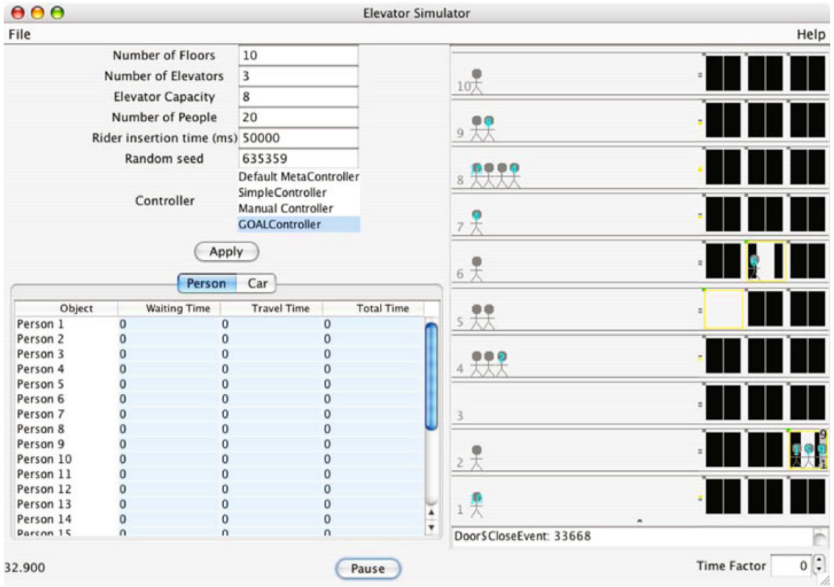
Implementation

- Two-way connections are provided via interactions performed by the components and notifications performed by the *EIS*
  - Interactions are (possibly remote) function calls to the *EIS*, that can return a value
  - Notifications follow the observer design pattern, components are *listeners* that have callback functions
- For agents, the two methods correspond to the notions of *polling* and *interrupts*
- Agents exist on the platform side, controllable entities exist in the environment, and agents may control entities through the *EIS* mapping between sets
- The *EIS* manages the sets of entities and agents, the mapping, observer notifications, and controlling manageable aspect of the environment

Case Study: Elevator

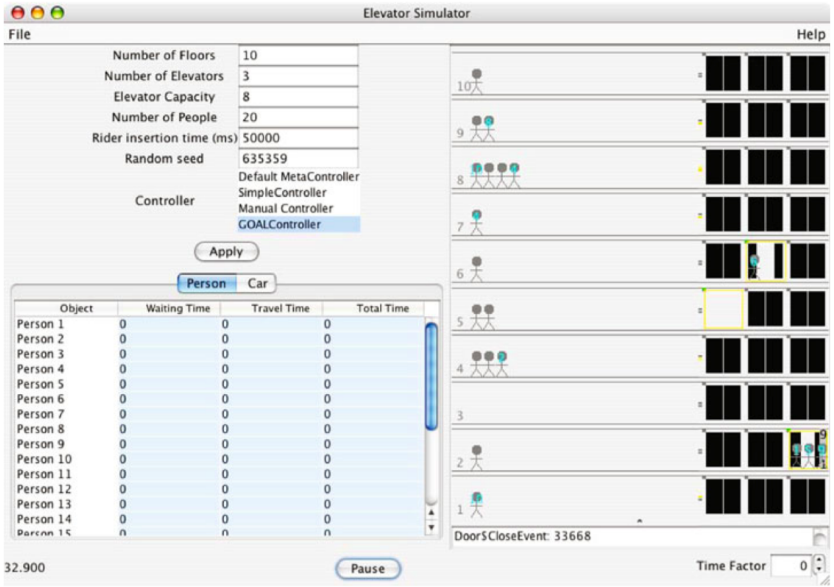


Case Study: Elevator



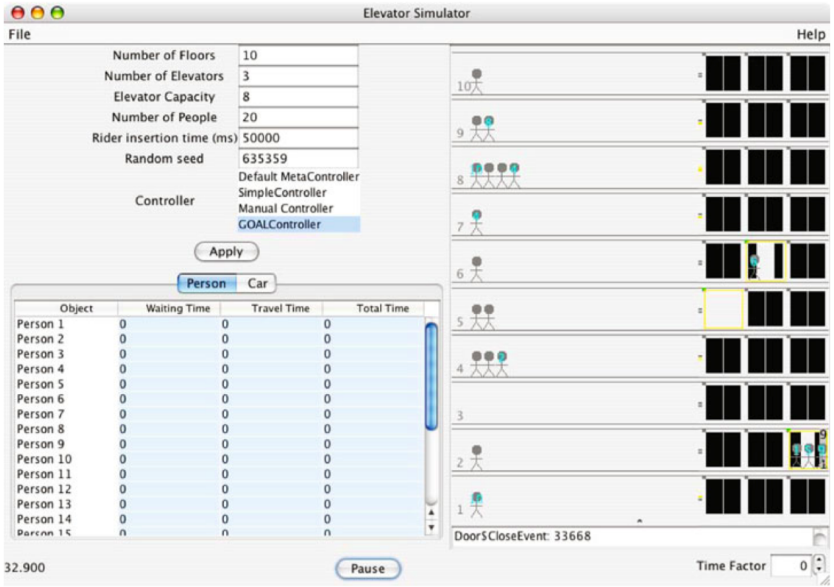
- Not designed for agents, originally adapted for GOAL

Case Study: Elevator



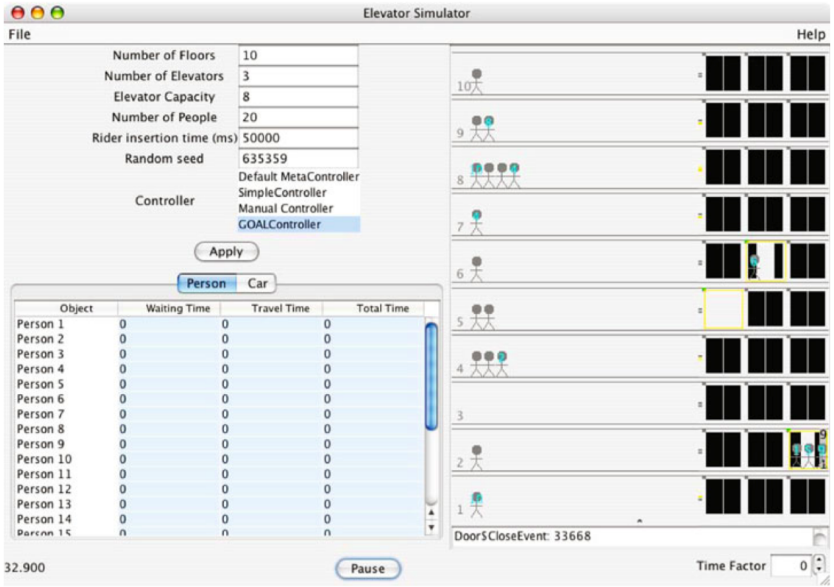
- Not designed for agents, originally adapted for GOAL
- *EIS* easily handles unique aspects of the environment

Case Study: Elevator



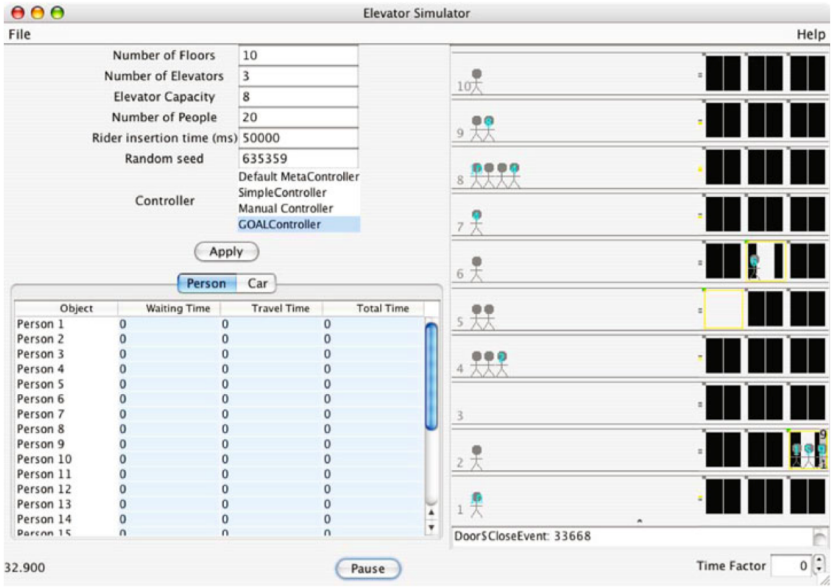
- Not designed for agents, originally adapted for GOAL
- *EIS* easily handles unique aspects of the environment
  - *Durative actions* take time to fulfil rather than being discrete single events

Case Study: Elevator



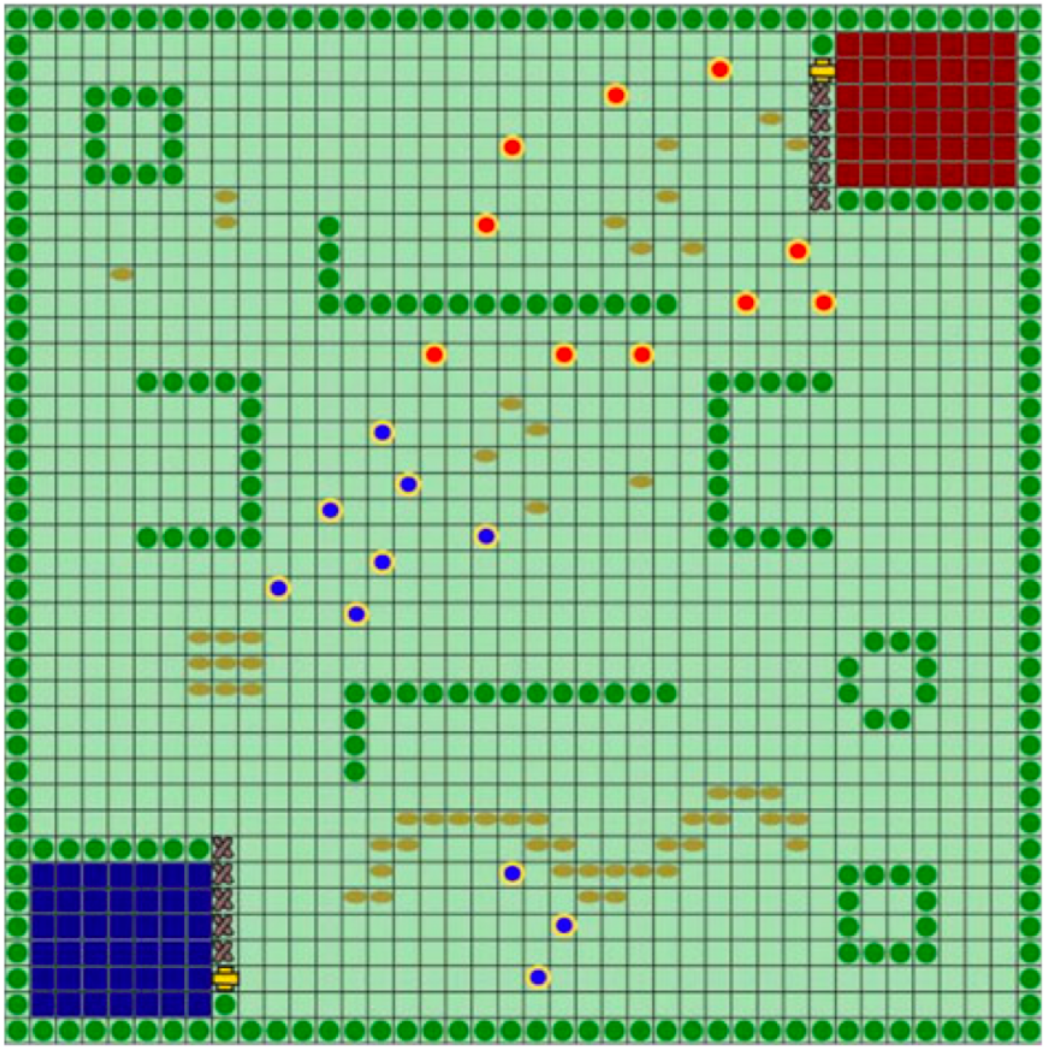
- Not designed for agents, originally adapted for GOAL
- *EIS* easily handles unique aspects of the environment
  - *Durative actions* take time to fulfil rather than being discrete single events
  - *EIS* supports the *partially observable* environment

Case Study: Elevator

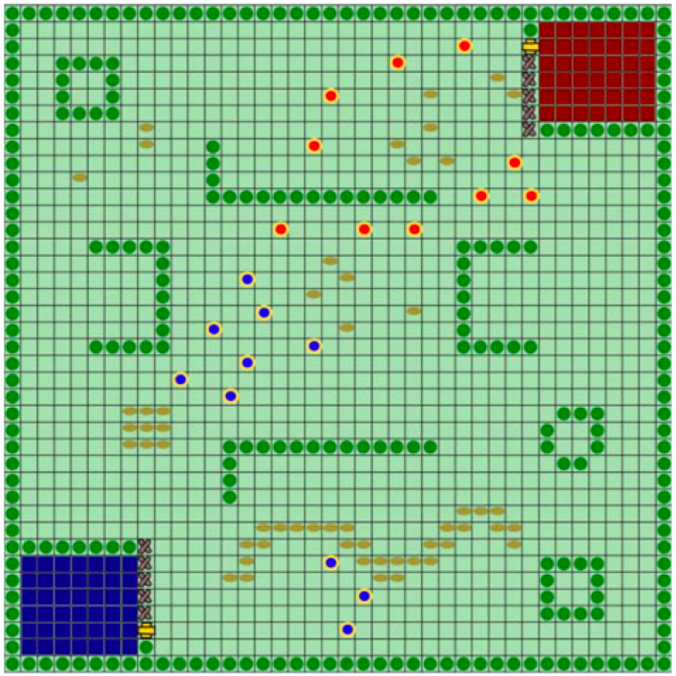


- Not designed for agents, originally adapted for GOAL
- *EIS* easily handles unique aspects of the environment
  - *Durative actions* take time to fulfil rather than being discrete single events
  - *EIS* supports the *partially observable* environment
- The elevator environment can now run with 2APL, GOAL-through-*EIS*, and *Jason*

Case Study: Agent Contest

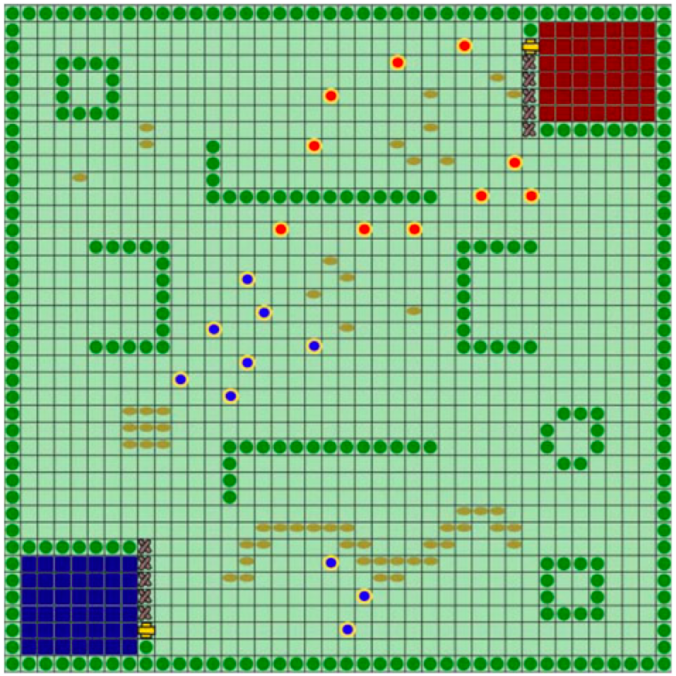


Case Study: Agent Contest



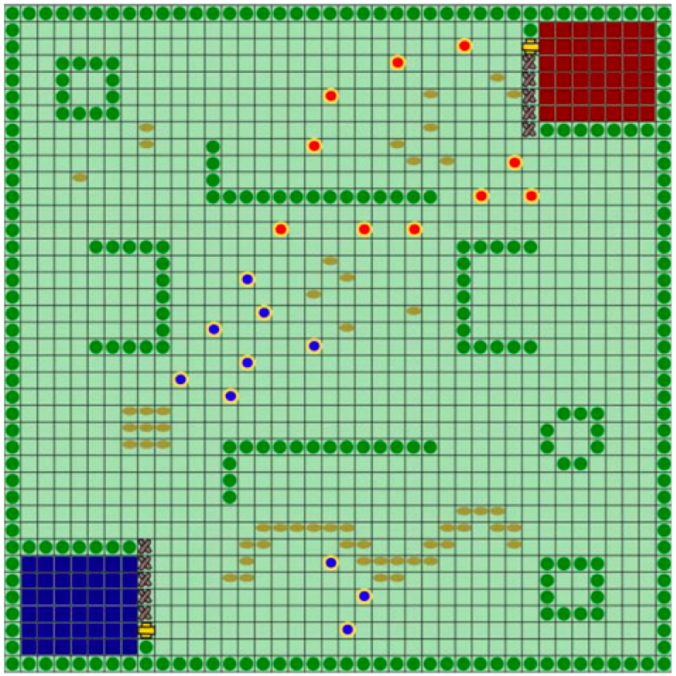
- Connection via TCP/IP is easily supported by the *EIS*

Case Study: Agent Contest



- Connection via TCP/IP is easily supported by the *EIS*
- Mappings from XML to the IIL work once completed

Case Study: Agent Contest



- Connection via TCP/IP is easily supported by the *EIS*
- Mappings from XML to the IIL work once completed
- The *EIS* does not impose restrictions on the connection between itself and environments

Case Study: Unreal Tournament

Case Study: Unreal Tournament

- More challenging due to the real-time low-level interface

Case Study: Unreal Tournament

- More challenging due to the real-time low-level interface
- Interface built on *Pogamut* and *GameBots*

Case Study: Unreal Tournament

- More challenging due to the real-time low-level interface
- Interface built on *Pogamut* and *GameBots*
- Provides abstraction for high-level actions

Case Study: Unreal Tournament

- More challenging due to the real-time low-level interface
- Interface built on *Pogamut* and *GameBots*
- Provides abstraction for high-level actions
- *Durative actions* and delegation implemented

Summary

- The environment interface standard as implemented conforms to the original goals and requirements

Summary

- The environment interface standard as implemented conforms to the original goals and requirements
- The ease with which it integrates with existing environments suggests that it supports a correct level of abstraction for agent-environment interaction

Summary

- The environment interface standard as implemented conforms to the original goals and requirements
- The ease with which it integrates with existing environments suggests that it supports a correct level of abstraction for agent-environment interaction
- It provides a number of benefits to future projects that support the interface:

Summary

- The environment interface standard as implemented conforms to the original goals and requirements
- The ease with which it integrates with existing environments suggests that it supports a correct level of abstraction for agent-environment interaction
- It provides a number of benefits to future projects that support the interface:
  - Standard functionality is provided by the interface implementation itself

Summary

- The environment interface standard as implemented conforms to the original goals and requirements
- The ease with which it integrates with existing environments suggests that it supports a correct level of abstraction for agent-environment interaction
- It provides a number of benefits to future projects that support the interface:
  - Standard functionality is provided by the interface implementation itself
  - Agent platforms that support the interface can connect to any environment that implements the interface

Summary

- The environment interface standard as implemented conforms to the original goals and requirements
- The ease with which it integrates with existing environments suggests that it supports a correct level of abstraction for agent-environment interaction
- It provides a number of benefits to future projects that support the interface:
  - Standard functionality is provided by the interface implementation itself
  - Agent platforms that support the interface can connect to any environment that implements the interface
- The *EIS* facilitates a new approach to comparison between APLs through a truly heterogeneous MAS