Agents for Serious gaming: Challenges and Opportunities

Frank Dignum
Utrecht University

Universiteit Utrecht

Contents

- Agents for games?
- Connecting agent technology and game technology
- Challenges
- Infrastructural stance
- Conceptual stance
- Design stance
- CIGA: middleware solution
- Conclusions

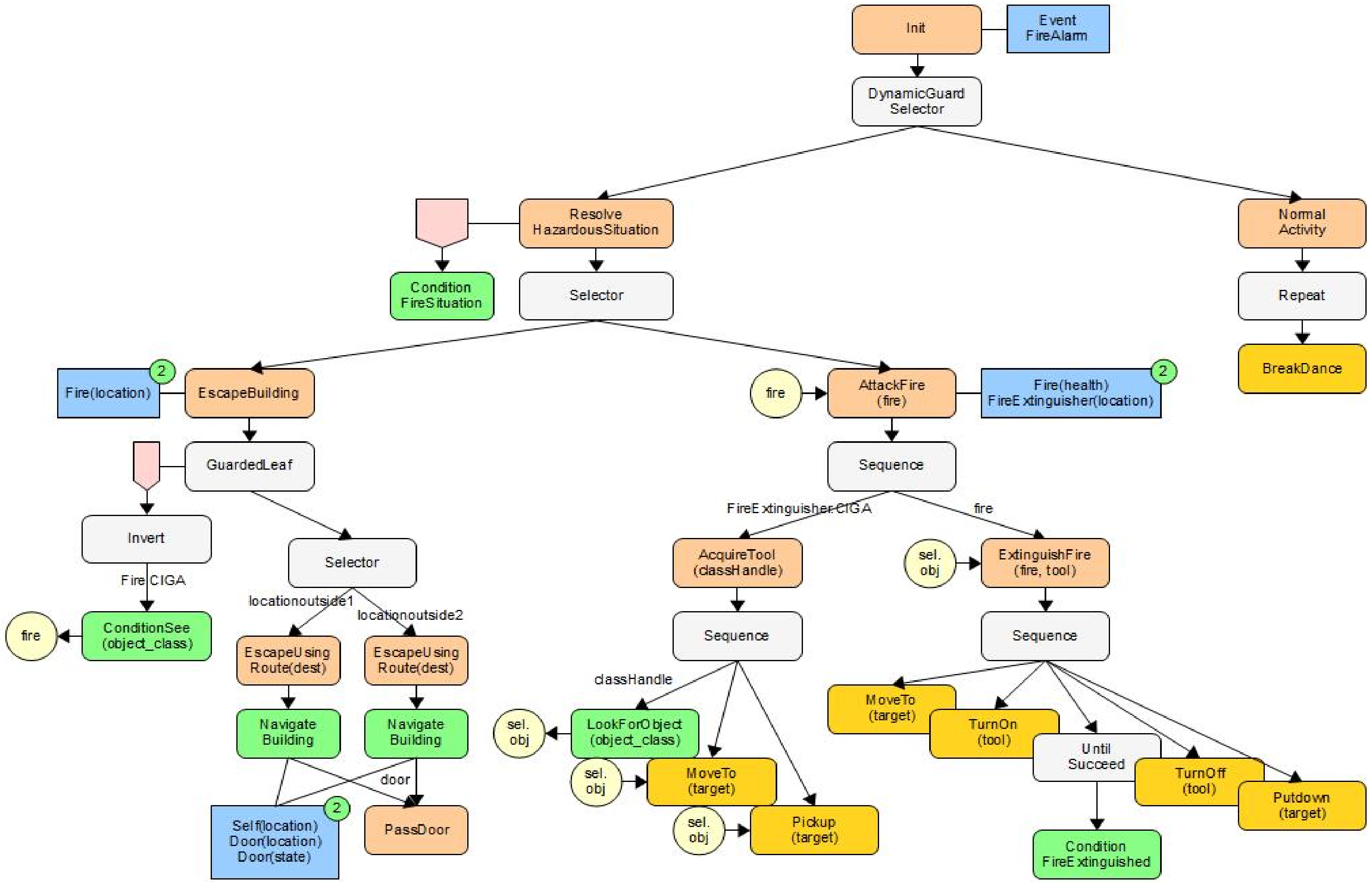
Do characters need to be intelligent?

Do we need agents for more serious games?

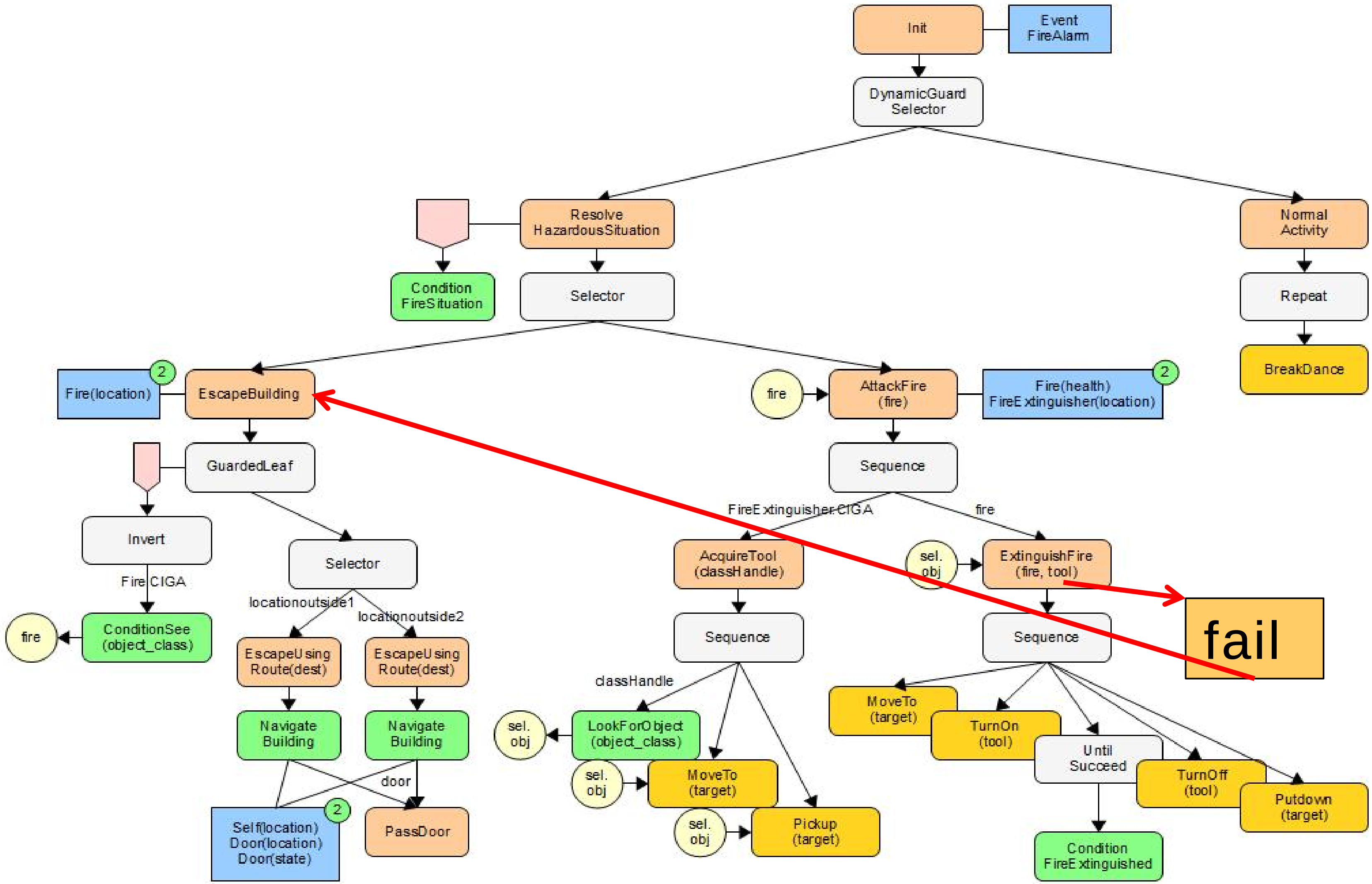
Agent features (claimed)

1. Goal directed
   - Agents find ways to reach a goal rather than execute a fixed procedure
   - In case of failure of a plan they can replan
2. Reactive behavior
   - Agents react to events in their environment (while keeping their goal in mind)
3. Social abilities
   - Agents know how to communicate in a high and flexible way (ACL is based on speech act theory)

GOAP vs. Agents



GOAP vs. Agents (failing actions)



Goal tree vs. rule based planning

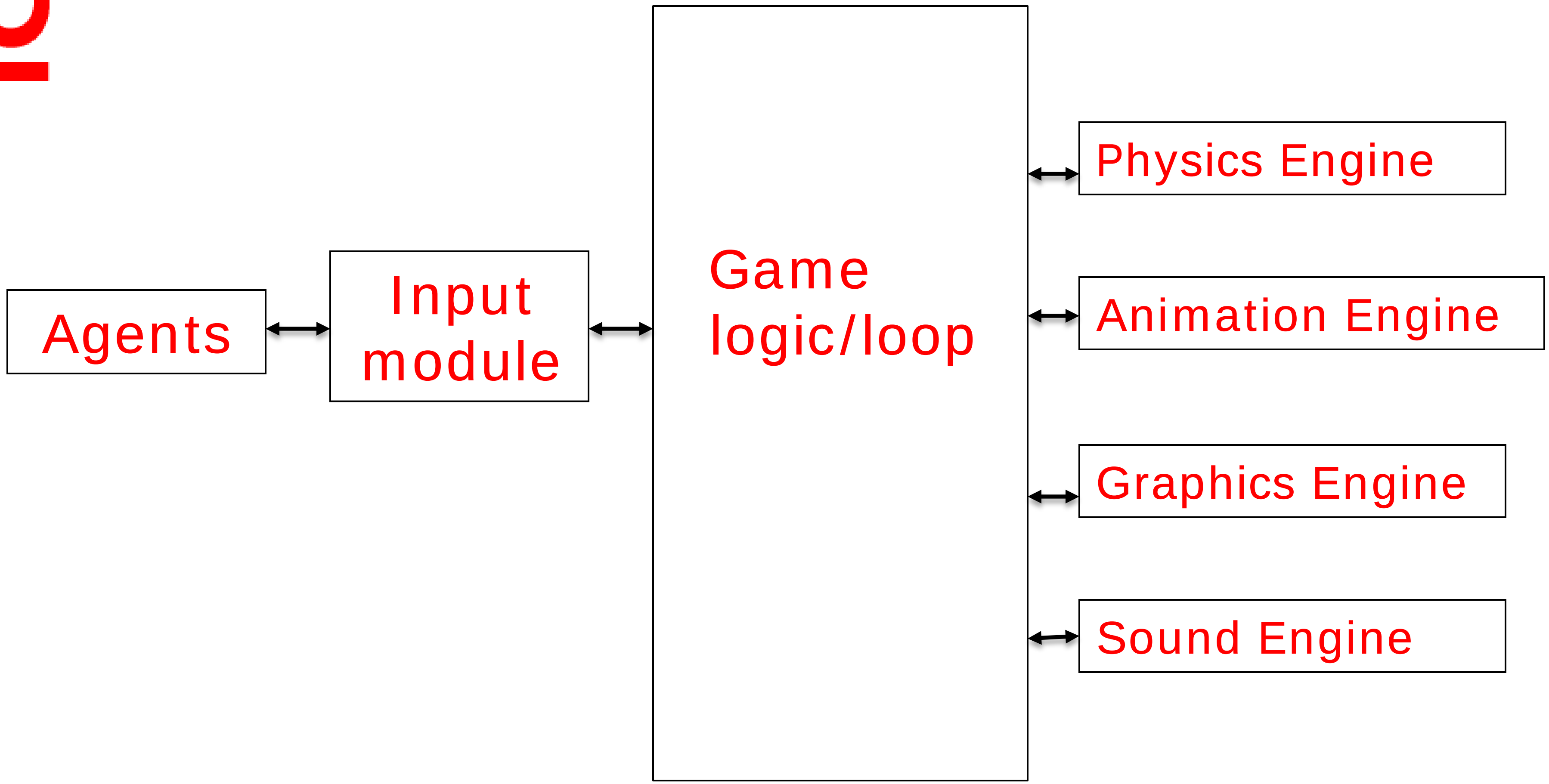
- Goal trees work well to describe default possibilities
- Trees get really messy when incorporating unexpected events and/or failures
- Rules are more suited to cope with these situations
- Divide rules in normal operation rules (default plans) and exception handling rules

- Flexibility comes at the cost of extra specification of general exception handling knowledge (based on domain)

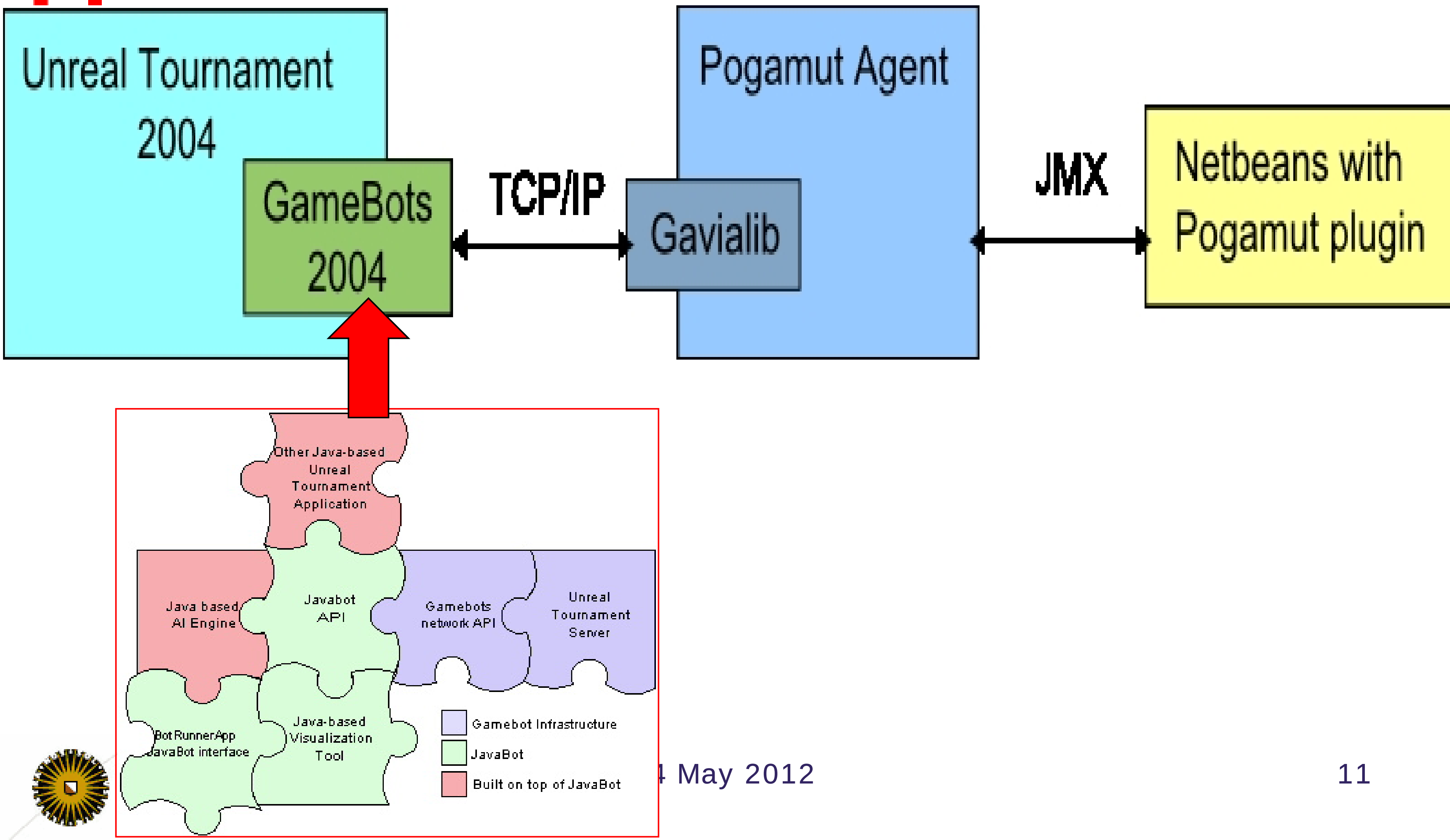
Agents for Games?

- Assume that we want to use agents for creating “intelligent” characters in games.
- Can we use Agent Technology to implement those agents in the games?
- I.e. can we make use of all the tools, techniques and platforms that are developed to implement intelligent agents for the incorporation of agents in games?
- If so, what do we need to do to couple the agent and game technologies?
- Or do we have to start from scratch and develop everything again specially for the game environment?

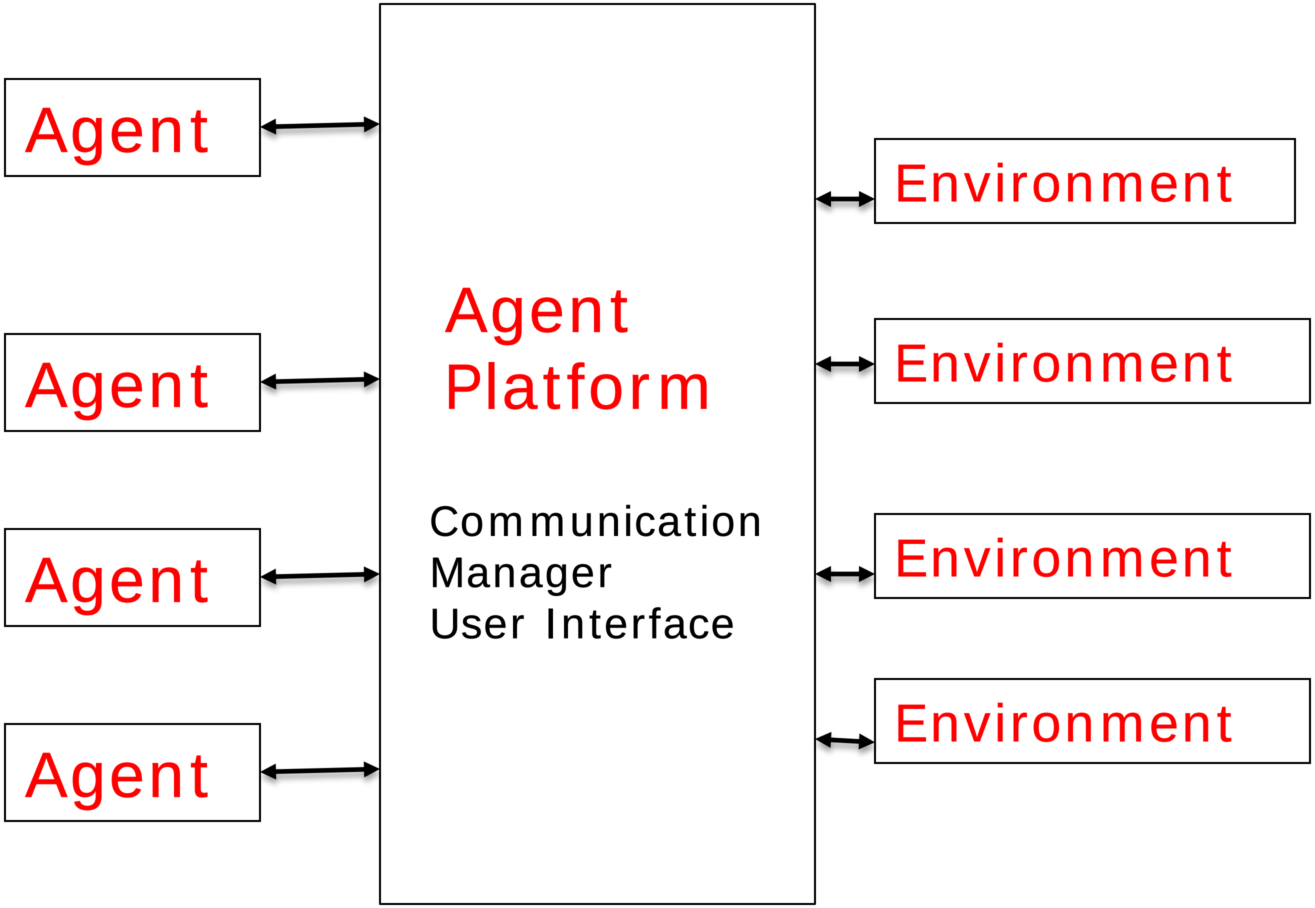
Game Engines and Agents
Client side approach



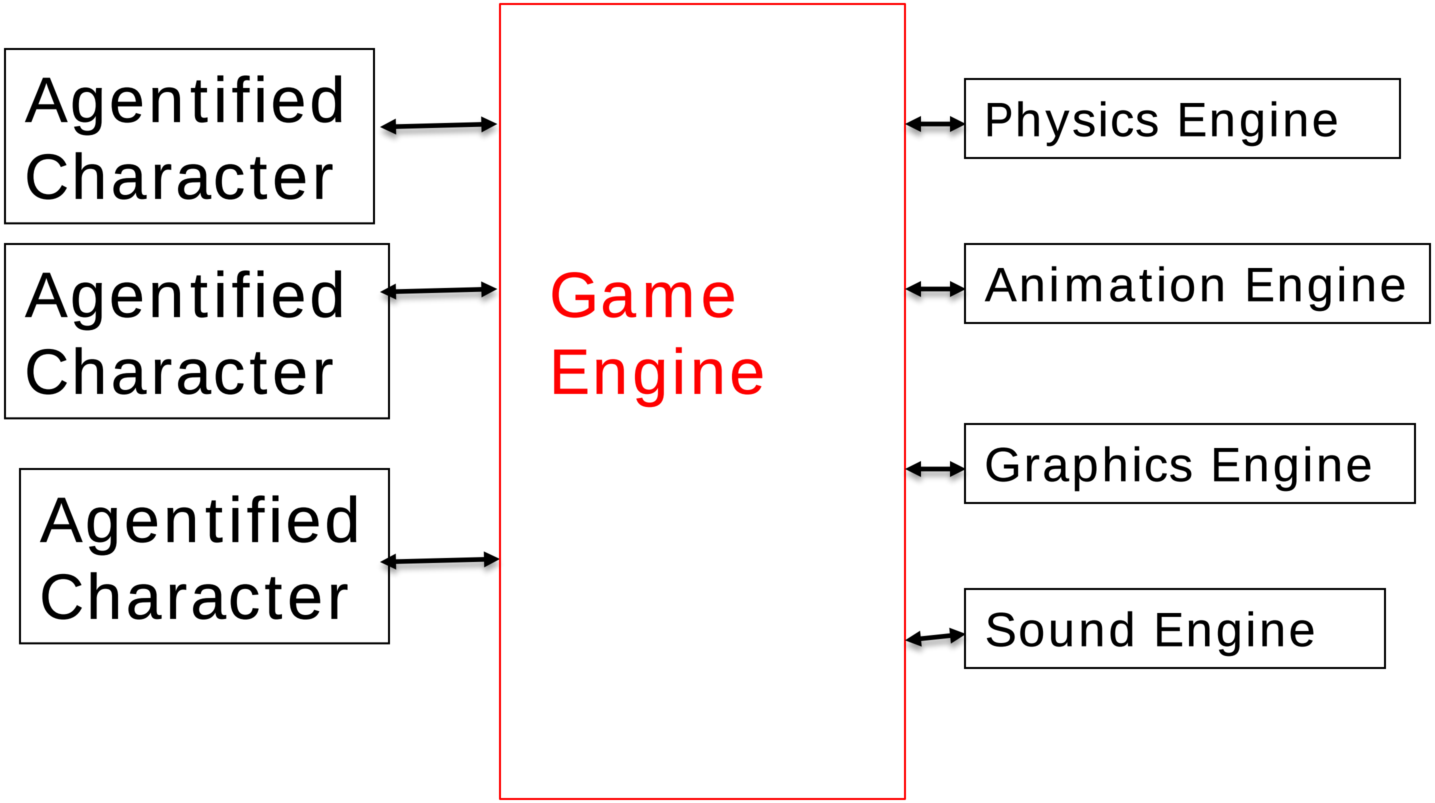
Example: Pogamut



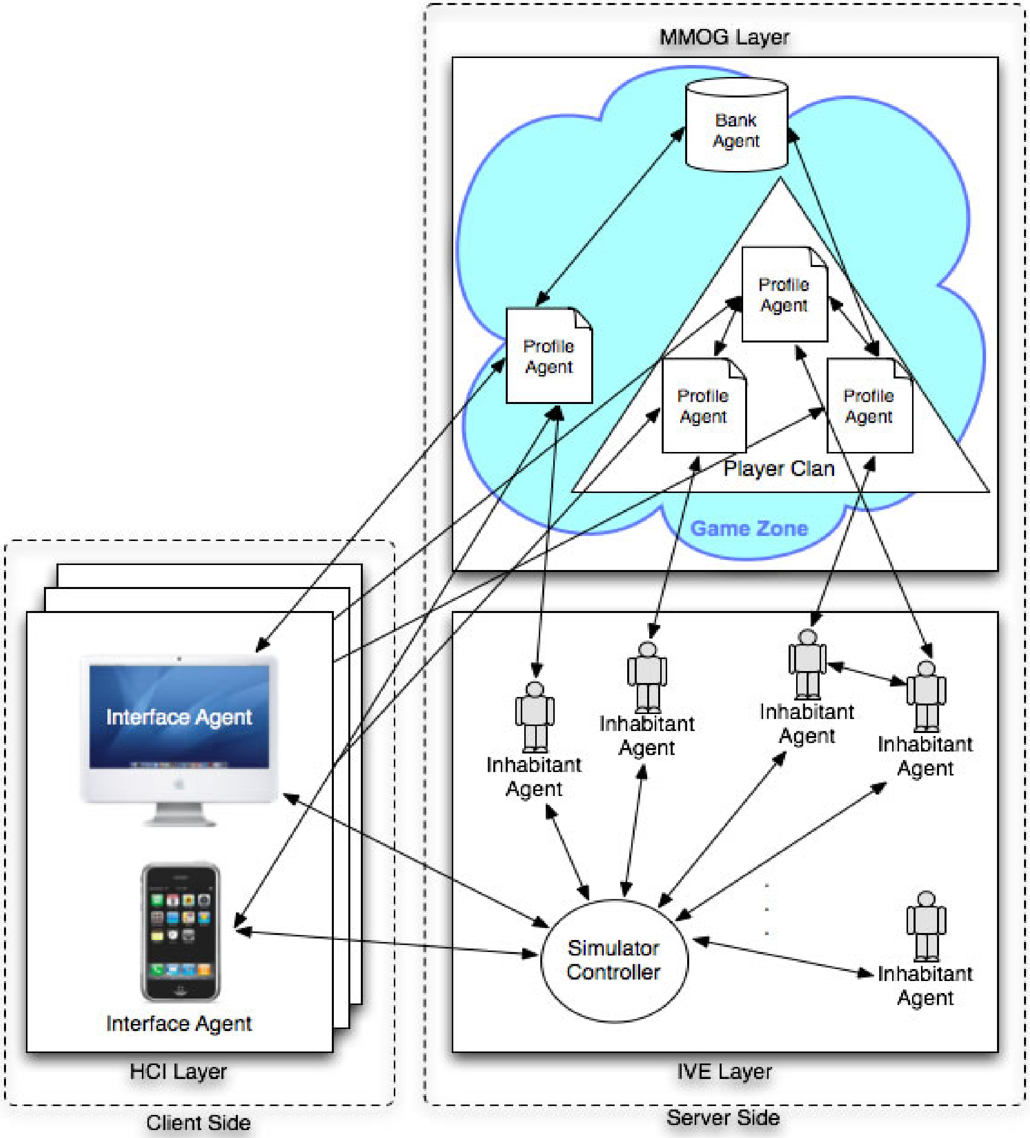
Multi Agent Systems



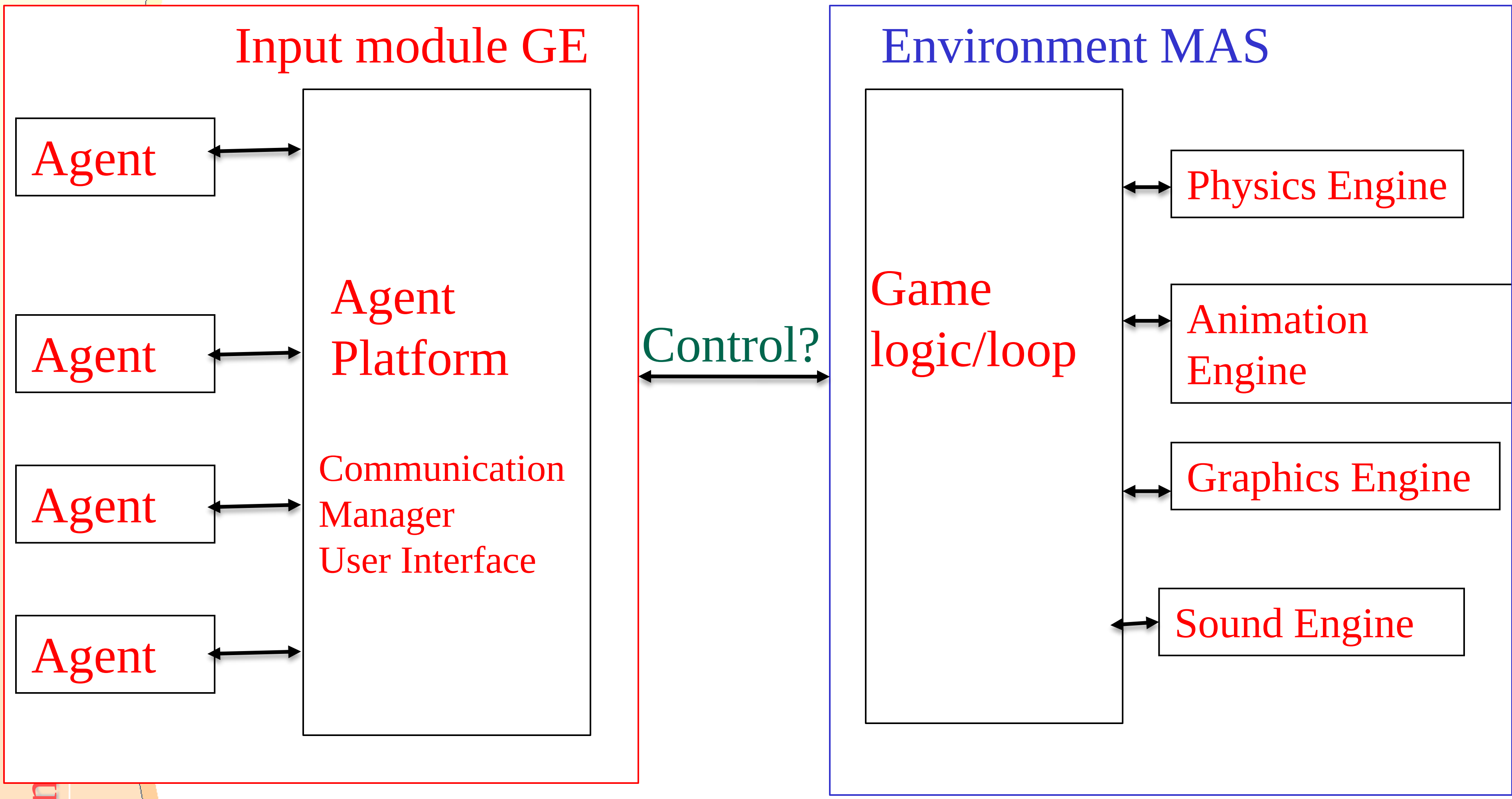
Game Engines and Agents
Server side approach



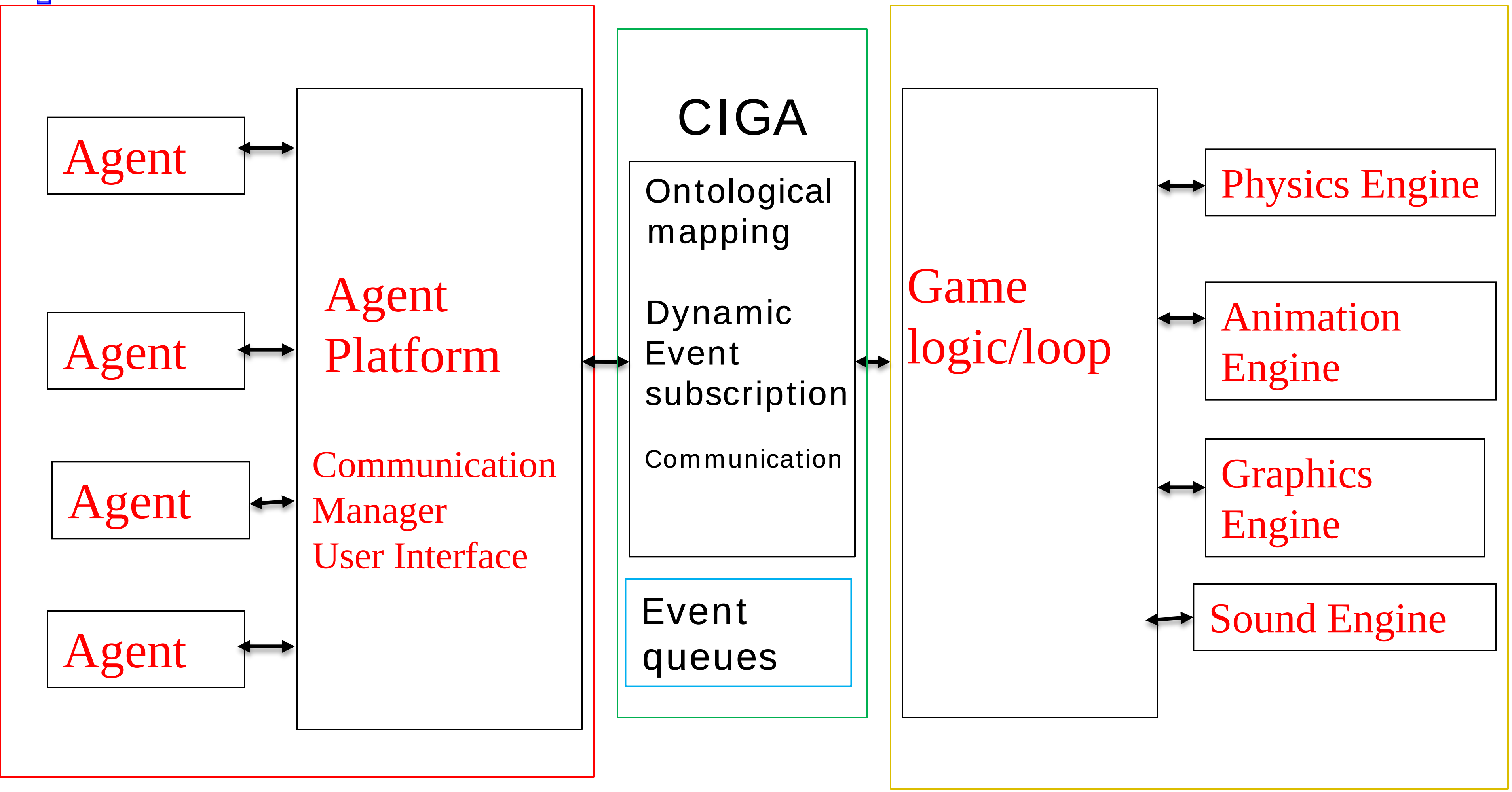
Example (THOMAS, Aranda et.al.)



Games plus Agents



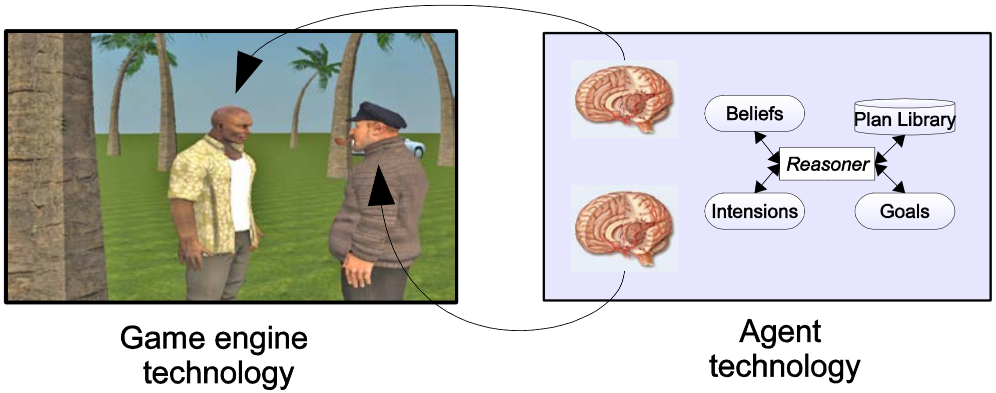
Games plus Agents



Department of Information & Computing Sciences

ICS

Intelligent Virtual Agent Design Issues

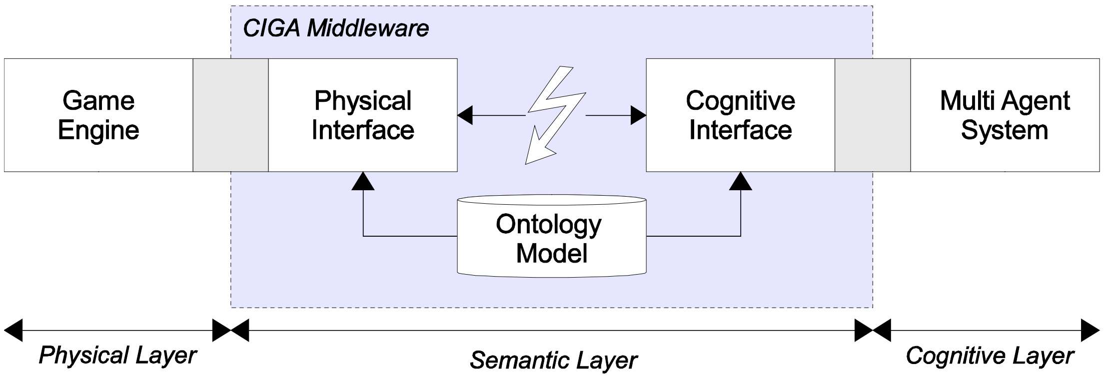


- IVA-design is distributed
  - Physical-layer + Cognitive-layer
  - Physical aspects vs. Cognitive aspects
- Cannot design these layers independently

Middleware Approach

- Bridge conceptual gap using a middleware
  - Design problems not responsibility of GE or MAS
- Middleware to provide technical facilities:
  - Translate data representations
  - Perception/action/communication mechanisms
- Don’t restrict designers in their IVA design, but offer technical solutions to help them realizing their design
- Performance determined by how the facilities are used
- Middleware itself is not part of the IVA design!
- CIGA Framework developed to follow this design approach

CIGA Framework



- Physical Interface: Connect to simulation environments
  - *E.g. CORE, (UT, CryEngine, Ogre, Delta3D, etc)*
- Cognitive Interface: Connect to agent systems
  - *E.g. Jadex, 2APL, BT-based MAS, etc*
- Connection Mechanism: Internal message-passing system
  - Introduced for flexibility and portability
  - *E.g. TCP/IP, Java/C++ bridge*
- Ontology Model: contract between GE and MAS
  - *E.g. Specify ontology using: Protégé, custom ontology editors*

Connecting the Game engine

- *Physical Interface* integrated into game engine as external component included in the update loop
- Motivation: become less dependent on the (limited) features provided by a particular game engine.
- Offers:
  - Monitoring entity creation
  - Time synchronization
  - Translation world state data to ontological sensory information
  - Perceptual attention: full control (what and when/how often)
  - Behavior realization: framework to implement actions

Department of Information & Computing Sciences

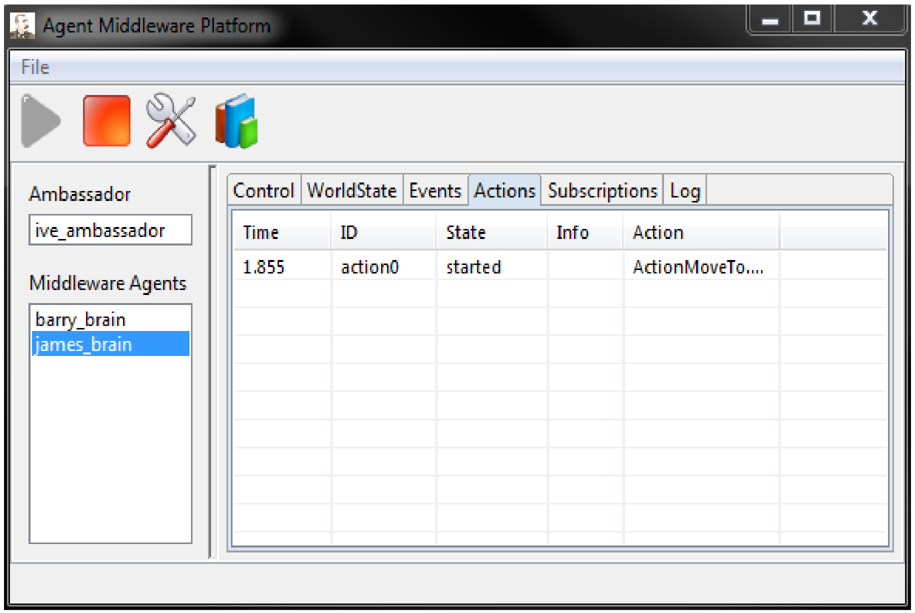
ICS

Connecting the MAS

- *Cognitive Interface*: integrated into MAS as event-based component (no synchronized update)
- Motivation: Provide simple interface for easy integration of wide range of MASs.
- Offers:
  - Notify MAS about possible entities to embody
  - Agent's sense-act interface where data are instances of ontology concepts
  - Access to ontology model from within the MAS

Department of Information & Computing Sciences
ICS

CIGA Platform + Tools



Run-time Platform GUI

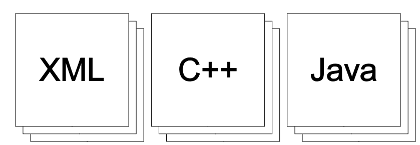
Features

- Monitor agents
  - Events, actions
  - Subscriptions, logs
- Test actions
- Profile agents
- Inspect ontology model

Middleware Configuration

protégé

Ontology-editor import scripts



Code Generation Tools

Aspects that make agents work in games

1. Ontology
   - reason on the right abstraction level
2. Perception
   - Get enough and not to much information
3. Action
   - Perform physical actions and react adequately on failure
4. Communication
   - Multi-modal communication

Data representation: Ontology

- Problem: Different data concepts in GE and MAS
  - World state vs. strategic abstraction level
- Solution: Translation-step during agent sensing on GE-side
- Design issue: Suitable abstraction level (not too low, not too high)

Conceptual Aspects
- interpretability

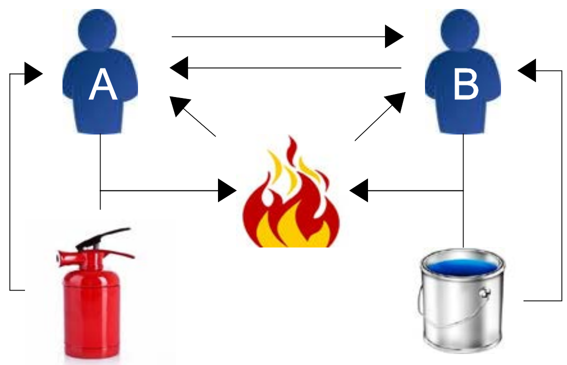
Technical Aspects
- efficiency - communication-costs

Ontology Model

- Contract on concepts communicated between GE and MAS

- Designers specify level of abstraction for sensory information and actions based on requirements for specific domain

Objects	Properties
PhysicalObject	location,size
– Human	gender,age
– Fire	type,heat
– FireExtinguisher	type
– Bucket	content,amount
Actions	**Parameters**
AttackFire	fire,equipment
Pickup	target
Communicate	target,message



ICS Department of Information & Computing Sciences

Ontology: Object Perception Model

- The Object Perception Model defines the ontology into which both the AT and the GE have to map.

Example:

```
<class name=” Character “>
  <property>    <name>ID</name>
                <type>number</type>
  </property>
  <property>    <name>Distance</name>
                <type>meters</type>
  </property>
  <property>    <name>Direction</name>
                <type>Orientation</type>
  </property>
  <property>    <name>Tool</name>
                <type>Tool</type>
  </property>
</class>
```

Ontology: Interaction model

```
<Agent name="Door-opener">
  <general>   <property>
      <name>HoldsOpeningTool</name> <type>Tools</type>
    </property>
<\general>

  <physical>    <property>
      <name>height</name> <type>meters</type>
    </property>
  </physical>

  <sensor name="eyes">  <property>
    <name>Range</name> <type>meters</type>
    </property>
  </sensor>

  <capability name="Open door"> <property>
      <name>target</name>  <type>Door</type>
    </property>
  </capability >
</Agent>
```

Ontology: Interaction model

- PRECONDITION "OpenDoor":
  Poss(OpenDoor(Agent,Door)) $\Leftrightarrow$
  Closed(Door) $\wedge$ Distance(Agent,Door)<1 $\wedge$
  Holds(Agent,Axe)

- POSTCONDITION "OpenDoor":
  Done(OpenDoor(Agent,Door)) $\Rightarrow$
  Open(Door) $\wedge$ Poss(Backdraft(Door))

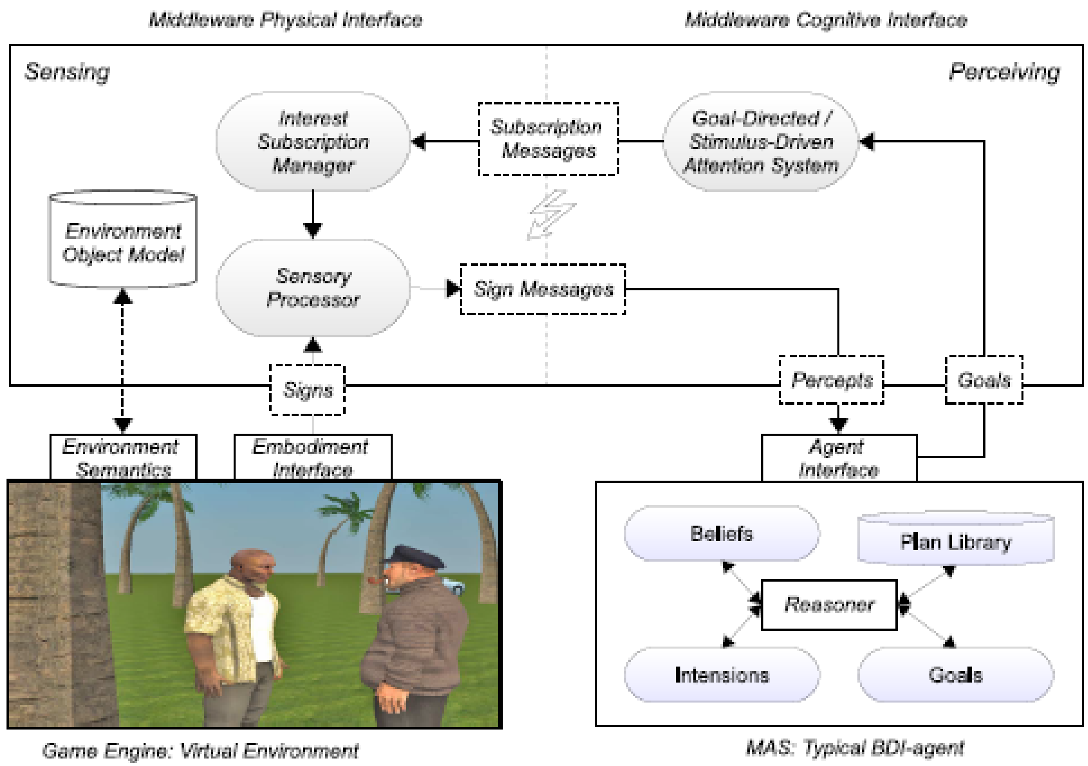
Control over Perception

- Problem: Perceptual attention for agents
  - Cannot attend to all information from the environment
  - Filtering cannot be performed by GE or MAS alone
- Solution: Subscription-based filtering mechanism
  - Agent controls sensing: what and when to sense
- Design issue: Balance flow of sensory information (not too much, not too little)

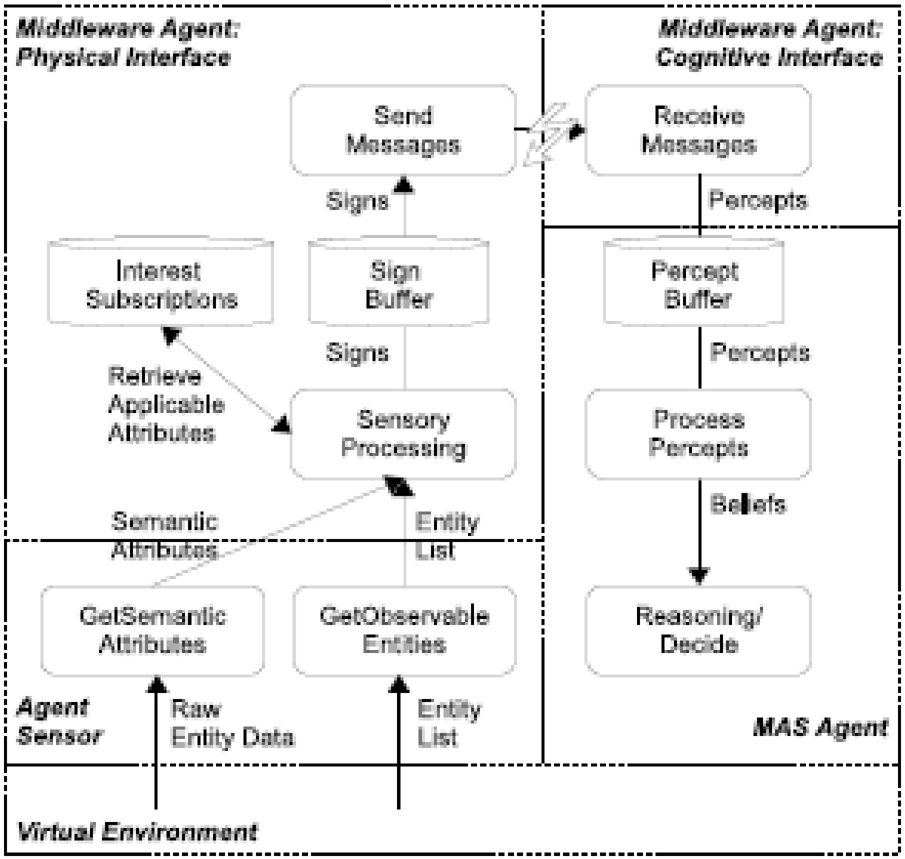
Conceptual Aspects
- goal-directed/ stimulus-driven

Technical Aspects
- performance MAS - performance GE - communication-costs

Perception framework



Implementation

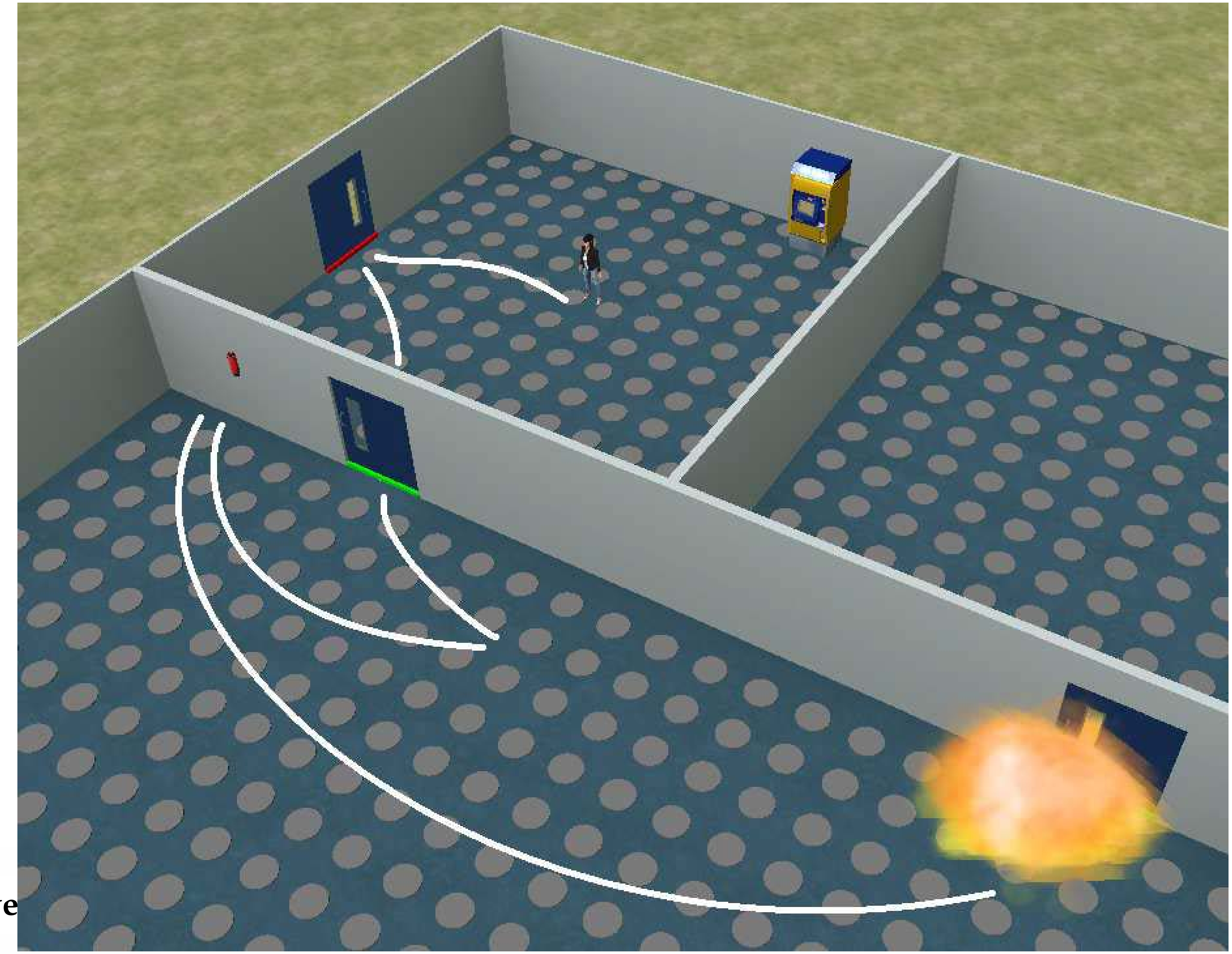


Subscription rules

Example:

$$Poss(Perceive(Character, ID)) \Leftrightarrow (Dist(Character, ID) < 150 \wedge LineofSight(Character, ID) \wedge Direction(Character, ID, towards))$$

Perception scenario



Control over Action Realization

- Problem: Different nature of actions in typical GE and MAS environments
  - Modality + Duration
- Solution: Action mechanism for body control + feedback channel
  - Dispatch, abort, feedback about status
  - Define actions at functional level
- Design issue: Suitable abstraction-level (not too low, not too high)

Conceptual Aspects
- control - individuality

Technical Aspects
- efficiency - communication-costs

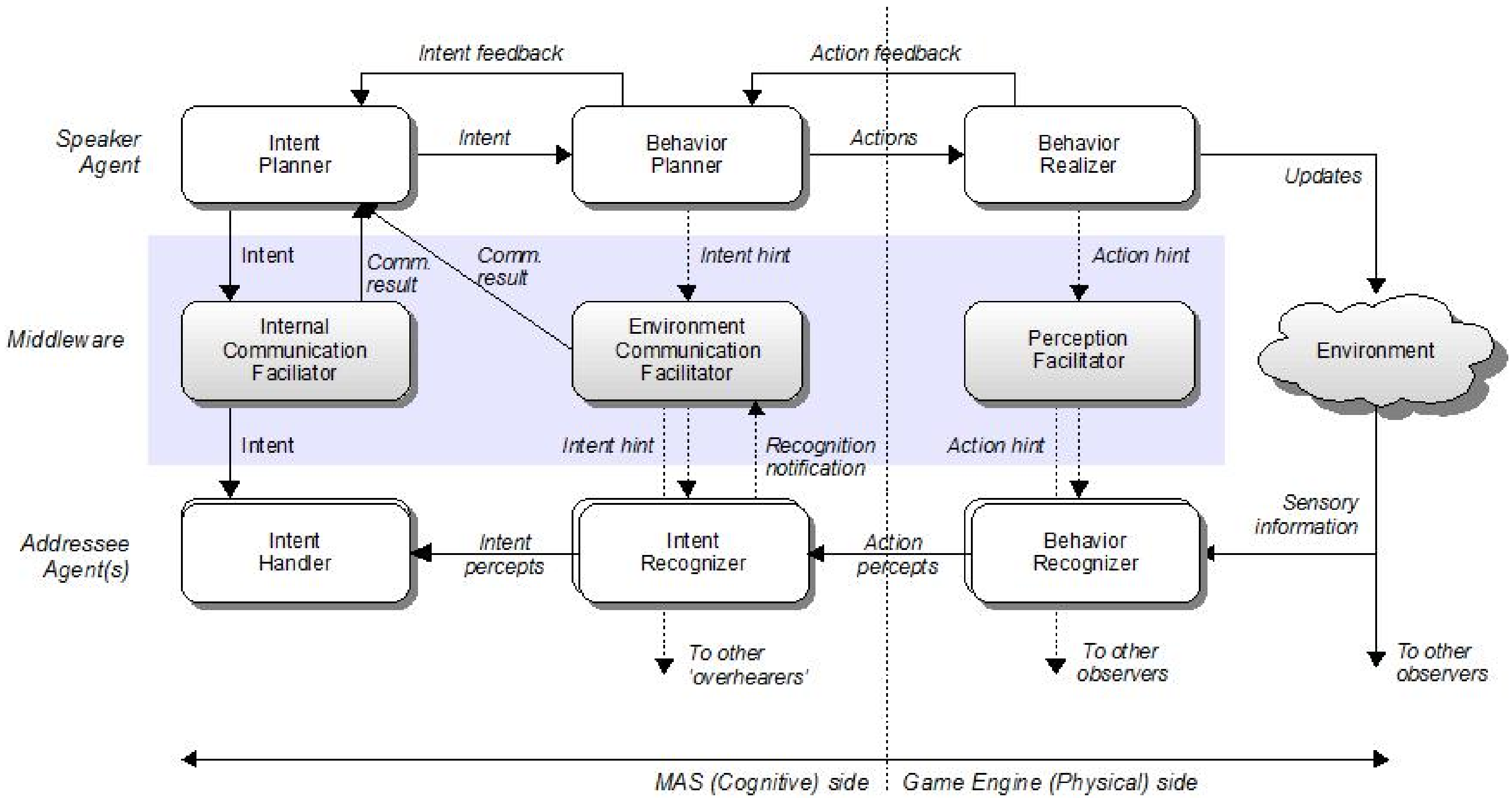
Communication

- Problem: Different communication in MAS and GE
  - Method: communicative intent (direct) vs. verbal and nonverbal communicative behavior (indirect)
  - Communication channel: reliable vs. unreliable
- Solution: Communication mechanism.
  - Allow MAS-communication through simulation environment
- Design issue: Choose method: behavior or intent

Conceptual Aspects	*Technical Aspects*
- interpretability	- complexity - efficiency

Communication is multi-modal

Multi-modal communication



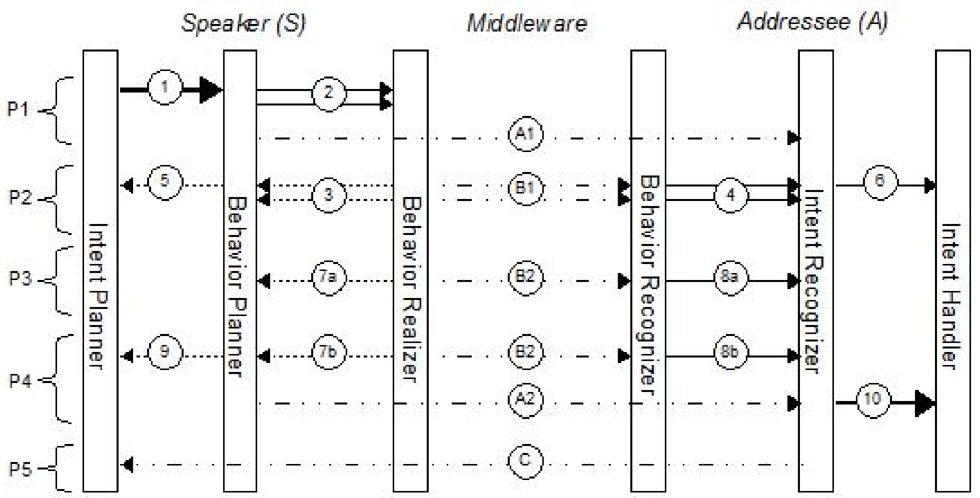
Example rules in modules:

- PRECONDITION:
  Poss(Send(Propose(Action,Agent))) $\Leftrightarrow$ Dist(Agent)<5

- POSTCONDITION:
  Done(Send(Propose(Action,Agent))) $\wedge$ Dist(Agent')<5
  $\Rightarrow$ Poss(Receive(Propose(Action,Agent)))

Can be used to describe physical constraints on communication and side effects of communication

Communicating agents



#	Agent	Activity	Message
1	S	schedule intent	*communicate(id=i1,content=inform_child_in_house)*
2	S	schedule action	*speech(id=a1,resource=child_in_house.mp3)*
3	S	receive action feedback	*action_feedback(action=a1,state=started)*
4	A	perceive action	*action_percept(action=a1,state=started)*
5	S	receive intent feedback	*intent_feedback(id=i1,state=started)*
6	A	perceive intent (no content)	*intent_percept(state=started)*
7	S	receive action feedback	*action_feedback(action=a1,state=finsihed)*
8	A	perceive action	*action_percept(action=a1,state=finished)*
9	S	receive intent feedback	*intent_percept(id=i1state=finished)*
10	A	perceive intent	*intent_percept(id=i1,state=ended)*

#	Middleware	Activity	Message
A1	Comm. Facilitator	send intent hint	*intent_hint(intent=i1, actions=a1;a2)(state=started)*
B1	Perception Facilitator	send action hint	*action_percept(action=a1,state=started)*
B2	Perception Facilitator	send action hint	*action_percept(action=a1,state=finished)*
A2	Comm. Facilitator	send intent hint	*intent_hint(intent=i1)(state=finished)*
C	Comm. Facilitator	send communication result	*communication_result(id=i1,observed_by=A)*

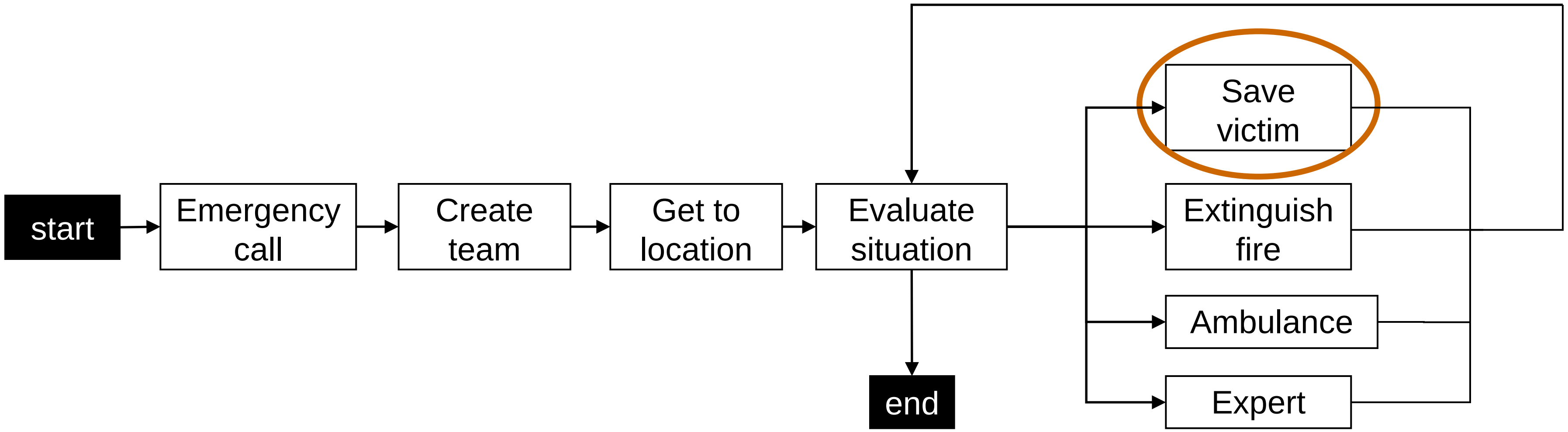
Designing games with agents: issues

- How intelligent can an agent behave (boundaries):
  - Story line
  - Game rules (including communication)
  - Environment (UI and look and feel)
  - Roles

Design games using OperA

- OperA specifies the boundaries of the behavior of the roles in the game
- OperA indicates landmarks that should be reached that can be used to specify the learning goals
- Agents can fill in the roles in different ways:
  - Scripted character
  - BDI agent
  - ...

OperA example: storyline



Department of Information & Computing Sciences

ICS

OperA example: Scene

Interaction Scene: save victim	
Roles	Leading_firefighter(1), door_opener(1), fire_extinguisher(1), ambulance(2), victim(3),
Trigger	∃H∈ people, ∃T∈ victim perceive(H,T)
Results	r1 = ∀ T ∈ victim, safe(T)
Interaction Patterns	PATTERN(r1) = { DONE(T, at(H,T)) BEFORE DONE(B, secure_area), DONE(B, secure_area) BEFORE DeadlineH), DONE(M, stabilise(H) BEFORE Dead(H)) DONE(T, transport_to_ambulance(H)) }
Norms	PERMITTED(E, blow_obstacles) OBLIGED(M,stabilise(T) BEFORE Dead(T)) OBLIGED (B, extinguish_fire BEFORE transport(H))

OperA example: Roles in a game

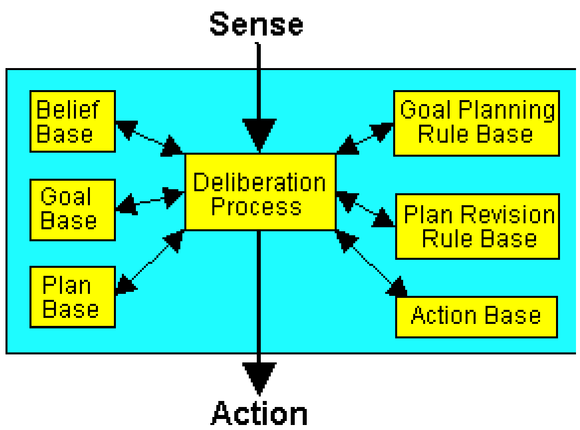
Role: leading firefighter	
Objectives	Fire_under_control, victims_save
Sub-objectives	{get_to_disaster_location, situation_assessment, plan_of_attack, extinguish_fire, rescue_victims}
Rights	Command_team_members, order_ambulance, get_experts
Norms	OBLIGED inform(headquarters,plan_of_attack) BEFORE NOW+10 IF DO safe(victim) or DO extinguish(fire) THEN PERMITTED damage(building) OBLIGED ensure_safety(team) OBLIGED safe(victims) BEFORE extinguish(fire)

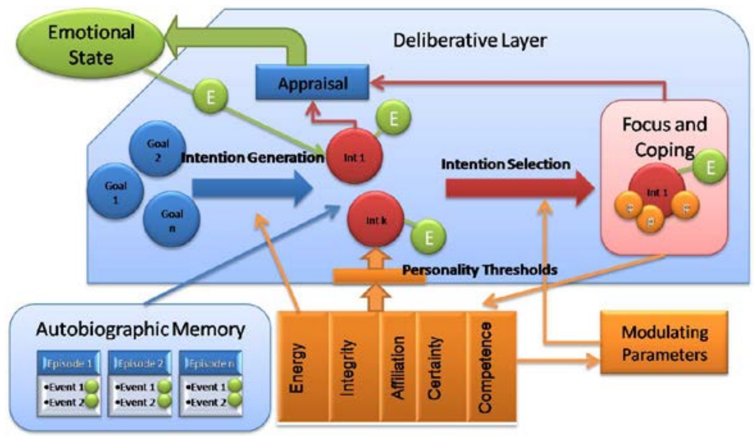
Conclusions

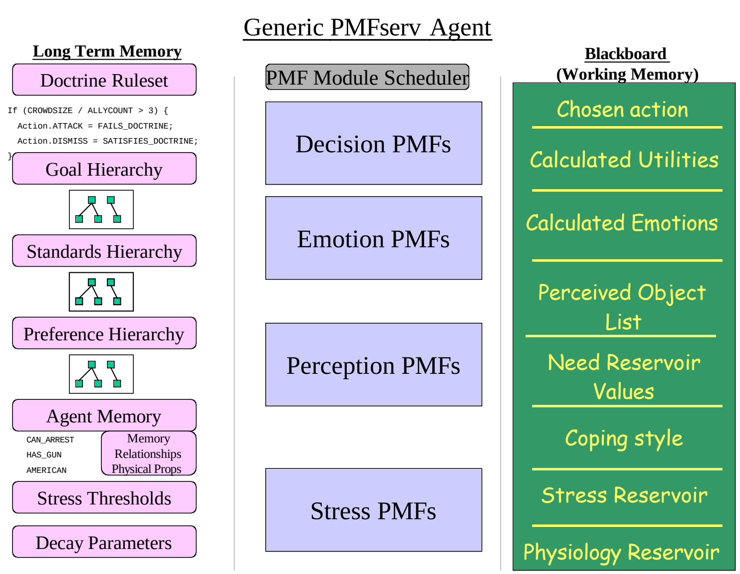
- Intelligence by design only
- Several stances needed to cover the connection between games and agents
- Need for a middleware between AT and GE
- CIGA is a principled approach that seems promising
- Infrastructure “easy”
- Conceptual connection is domain dependent
- Design using an OperA like methodology seems promising

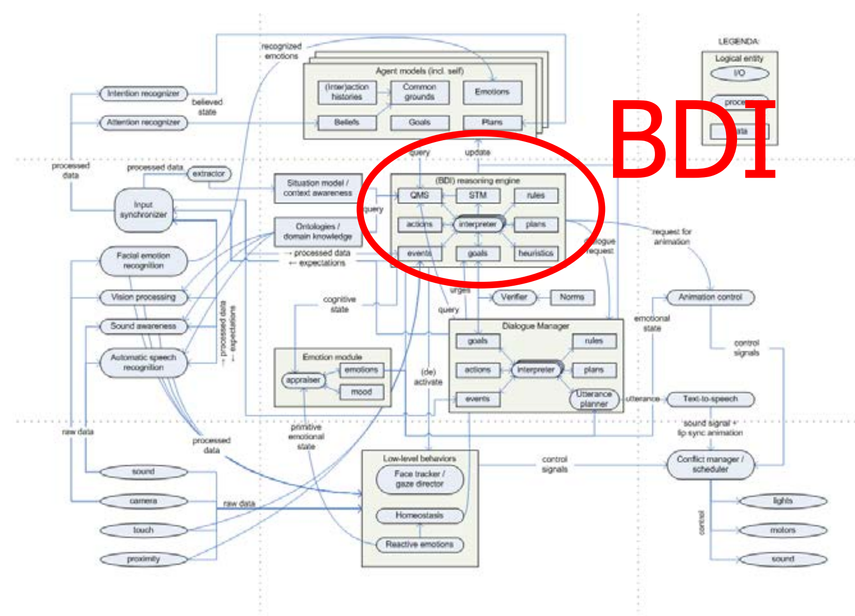
- What should be done by the agent and what by the game engine?
- Programming agents?
- What should be intelligent? (pathplanning vs. conversations)
- What agent technology/architecture to use?
    - Existing agent technology is not sufficient or very ad hoc

Agent architectures









Department of Information & Computing Sciences ICS

QUESTIONS?