Modeling with Agent-Space architecture

Andrej Lúčny

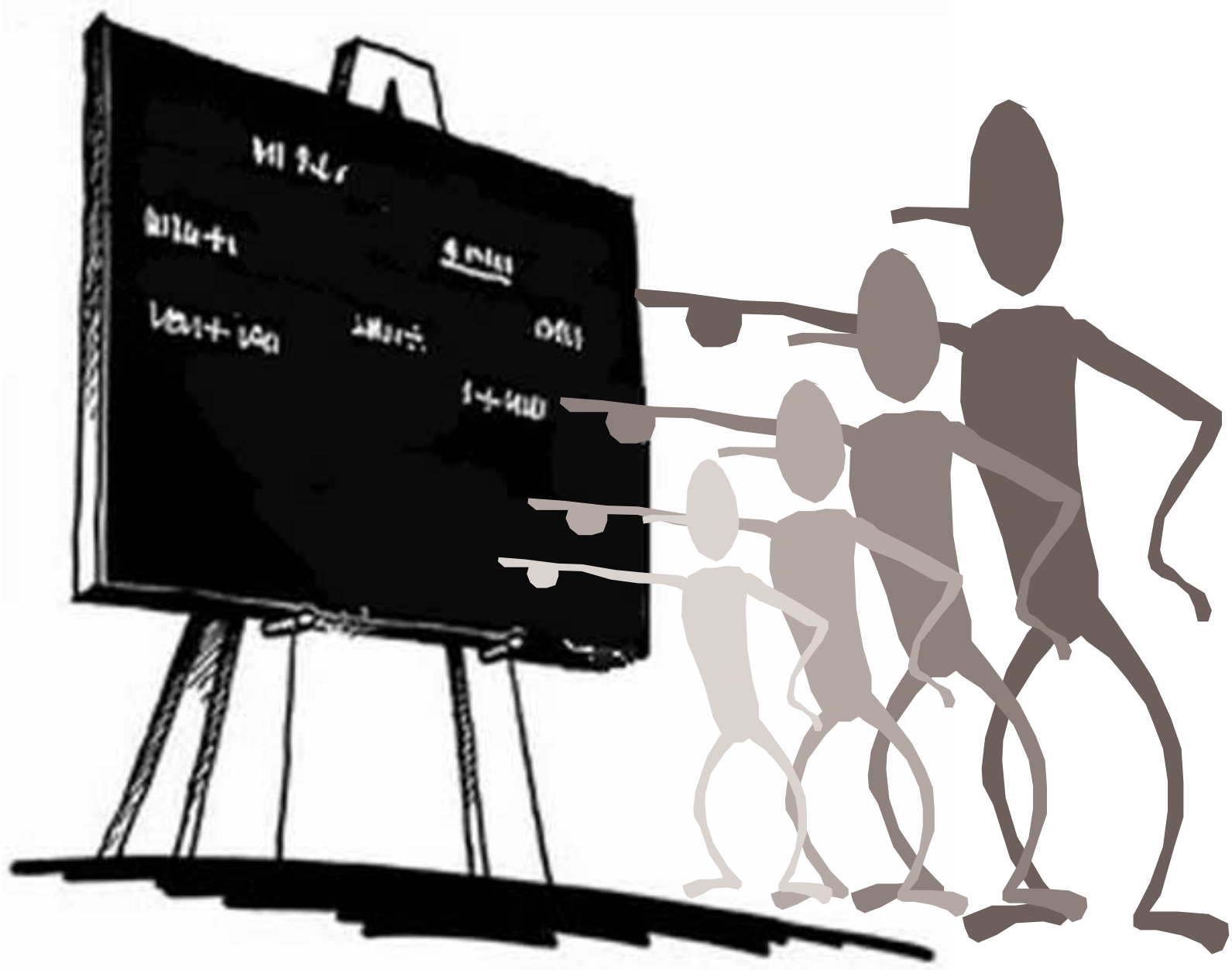
MicroStep-MIS & Comenius University Bratislava

andy@microstep-mis.com

http://www.microstep-mis.com/~andy

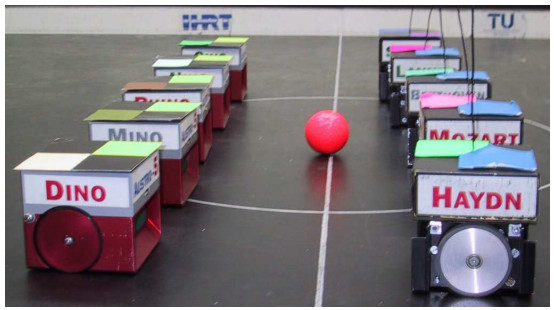
Agent-Space architecture

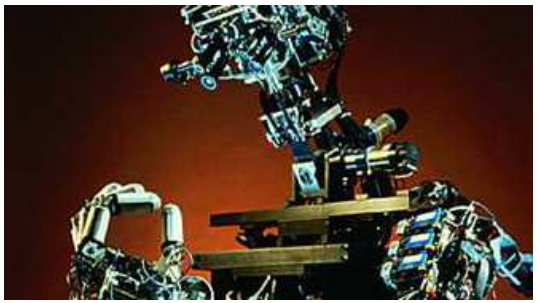
- is a conception of internal structure of interactive system models
- is suitable for decentralized models which global behavior emerges from interaction among modules with simpler but specific behavior
- has symbolic-computational nature, i.e. describes the modeled system by higher-level modules



- Agent-space is a blackboard architecture
- models consist of many agents equipped with space - a storage of named data blocks
- there is no direct communication among agents, they communicate only through space
- agents can read, write or delete particular blocks in space
- agent knows nothing about other agents, just knows names and structure of the blocks it manipulates with
- agents regularly perform quite a simple code within *sense-select-act* cycle which is blocked by timer or by trigger on change in space

- Agent-space is a kind of (generalized) agent-oriented programming, i.e. it employs similar modularity as typical for distributed systems, though the application is not necessarily distributed





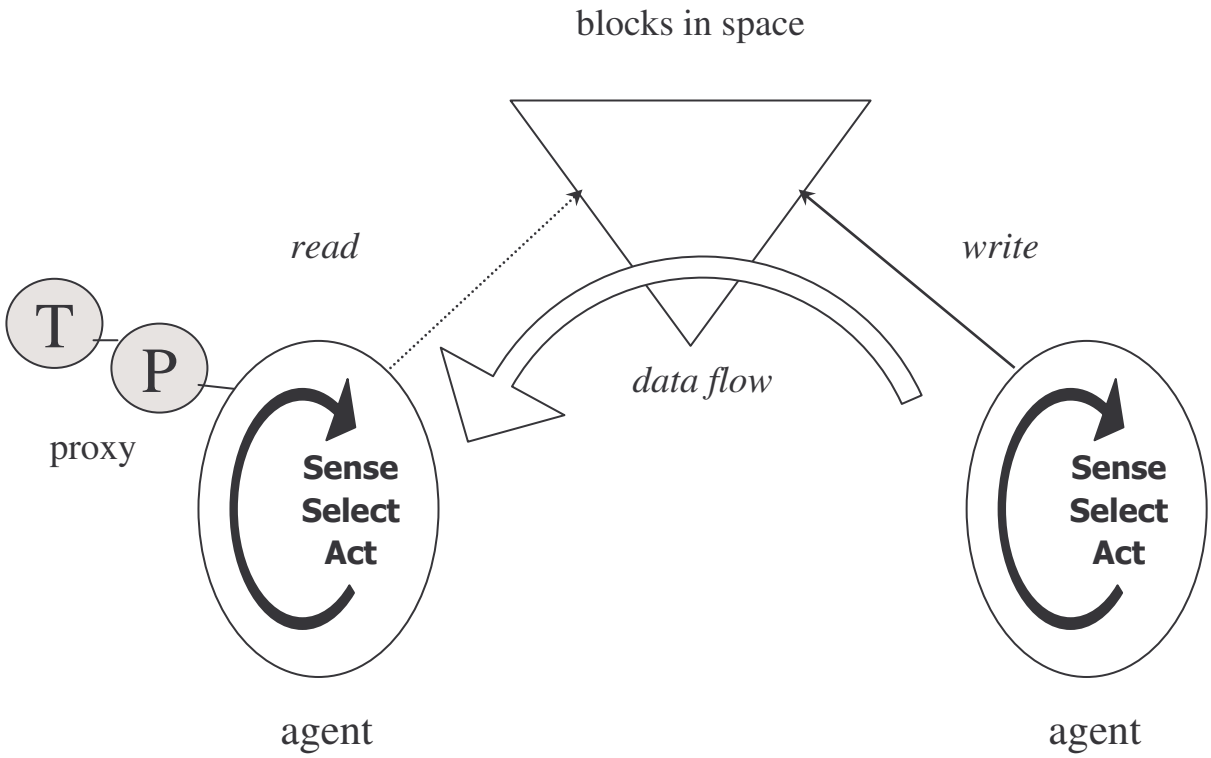
Network is a computer

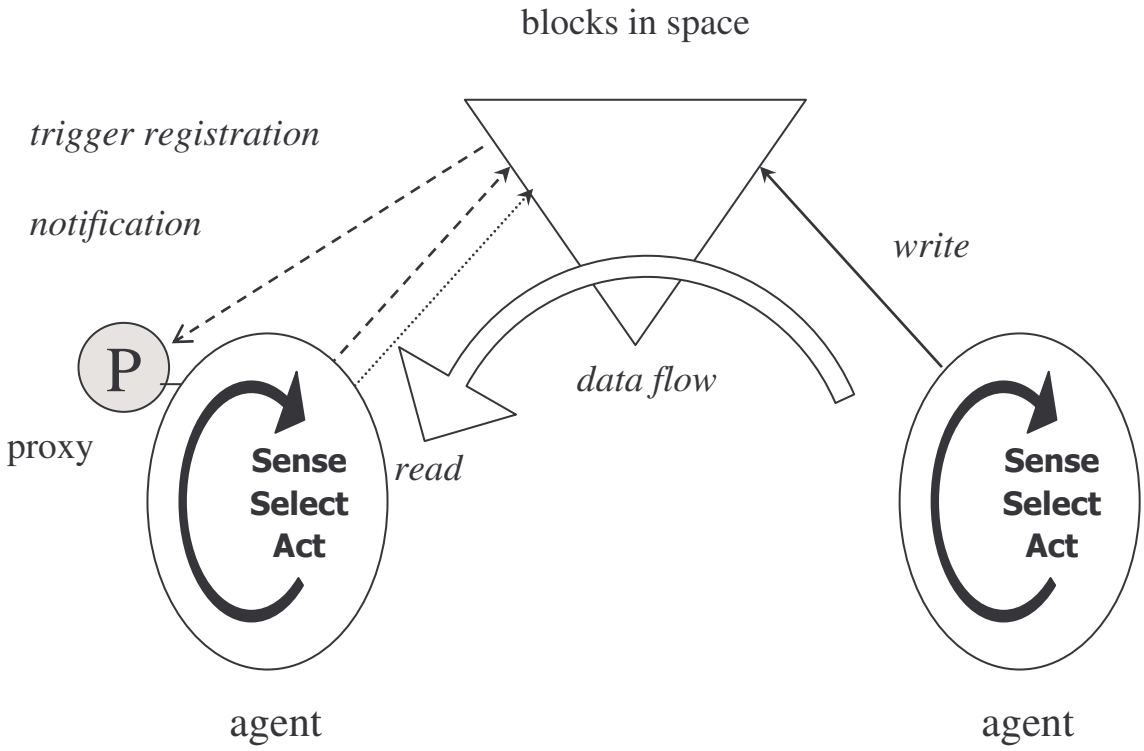
Computer is a network

- (we do not concern here that agent must have internal structures corresponding to mental states, rather its code is usual and as simple as possible)

Details of read, write and delete operations are set

- for operation in real-time (producers overwrite blocks regardless consumers have undertaken their values, moreover automatic expiration of the values is supported)
- to pay no attention to order of consumers and producers activation
- to enable a higher-level agent to influence communication between any two agents on lower level (to monitor, to stop or to emulate)


blocks in space
read
write
T
P
proxy
data flow
Sense
Select
Act
Sense
Select
Act
agent
agent


blocks in space
trigger registration
notification
write
P
proxy
data flow
read
Sense
Select
Act
Sense
Select
Act
agent
agent

Details of read, write and delete operations are:

- no method for block creation
- reading of non-existing blocks is handled by returning a default value specified by reader
- value stored in block can have a limited time validity specified by writer; after its expiration the block becomes automatically empty
- value stored in block can have a priority specified by writer; such value overwritten only by value with same or higher priority
- space has no knowledge about value meaning; the reader is responsible for correct interpretation

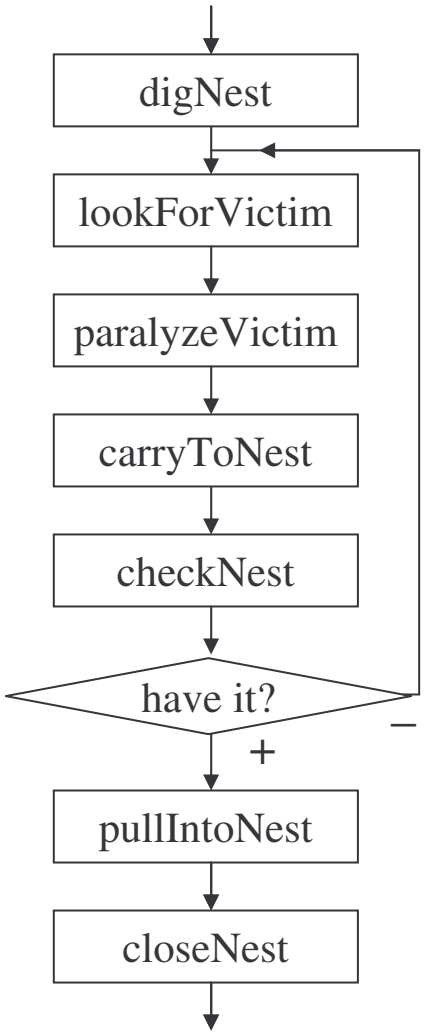
Resulted feature are:

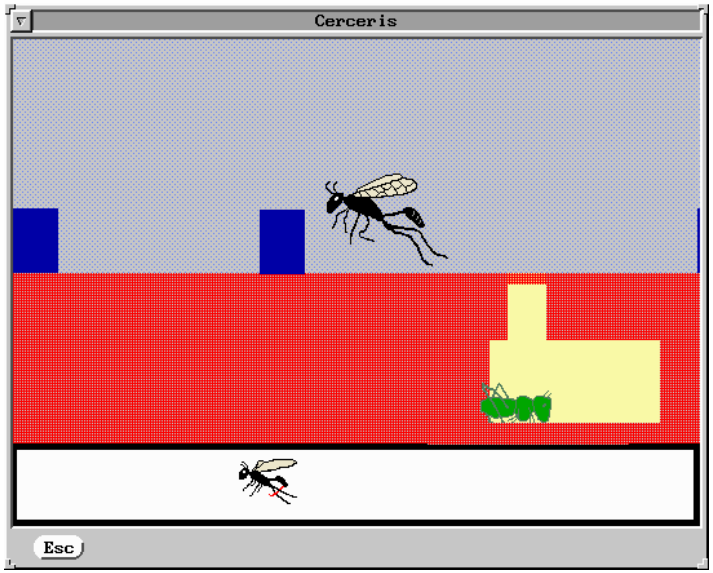
- each block can be written by many producers and read by many consumers
- consumers do not know how much producers generates the value or from whom the read value is coming
- if producers writes value with the same priority, it is almost random whose value will be undertaken by consumer
- if a consumer is too slow to undertaken all produced values, its input is automatically sampled at the highest possible speed
- each agent can be restarted without impact on system operation (recovery from errors)

This architecture is derived from (or has a relation to):

- Society of mind by Minsky (the fundamental nature of model)
- Fodor's model of mind (nature of modules)
- Subsumption architecture by Brooks (bottom-up incremental development)
- Blackboard architectures like Gelernter's LINDA (nature of communication among modules)
- Real-time systems (timers, triggers) as SRR

Example: modeling of mating behavior of digger wasp





space:

sensed blocks:
inseminated
victimDetected
paralyzedVictimDetected
nestDetected
insideNest
nestClosed

retained blocks:
nestReady
checkingPerformed
checking
victimInNest

behaviors:

Generates a small-step action at each call:

digNest()
lookForVictim()
paralyzeVictim()
carryToNest()
checkNest()
pullIntoNest()
closeNest()

agents:

```
for (each 1 second)
  if (inseminated && !nestReady) {
    if (insideNest) nestReady = true for 1 day;
    else digNest();
  }
```

```
for (each 1 second)
  if (nestReady && !victimDectected)
    lookForVictim();
```

```
for (each 1 second)
  if (victimDectected && !paralyzedVictimDectected)
    paralyzeVictim();
```

```
for (each 1 second)
  if (paralyzedVictimDectected && !nestDectected)
    carryToNest();
```

agents:

```
for (each 1 second)
  if (!checkingPerformed && ((paralyzedVictimDectected
     && nestDectected) || checking) {
    checking = true for 7 seconds;
    checkNest();
    if (insideNest)
      checkingPerformed = true for 40 seconds;
  }
```

```
for (each 1 second)
  if (checkingPerformed && paralyzedVictimDetected
      && !victimInNest)
    if (insideNest) victimInNest = true for 1 hour;
    else pullIntoNest();
```

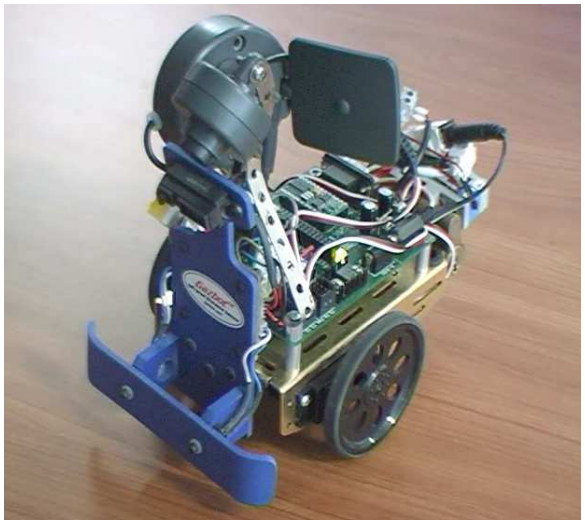
```
for (each 1 second)
  if (victimInNest & !nestClosed) closeNest();
```

Such models:

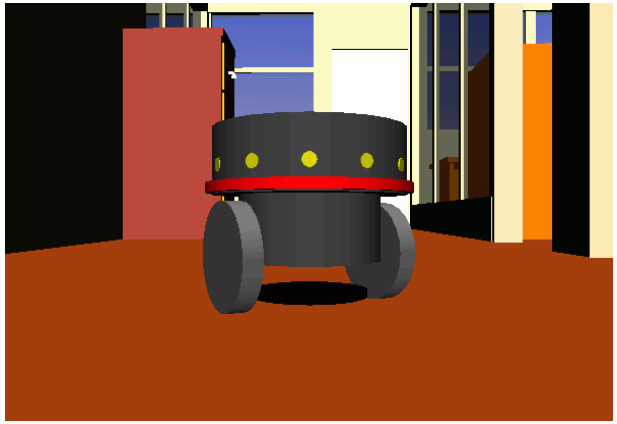
- do not contain specific handling of exception; handling exceptions is an inherent feature here
- are able to provide more information than those observed in nature and inbuilt into model
- can provide hypothesis which can be tested in nature and thus the models can be verified (e.g. from the model we estimate that there is an impact of distance of the moved victim – if small enough, no repetition of nest checking appears)

Other applications

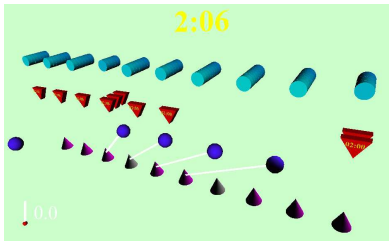
ball-following robot



3D model of robot ALLEN

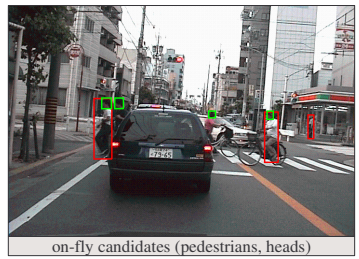


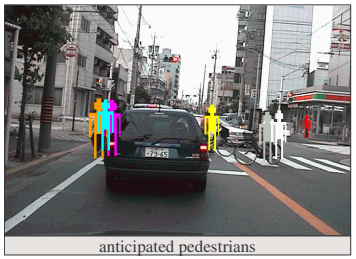
data collection in unreliable network (commercial project)

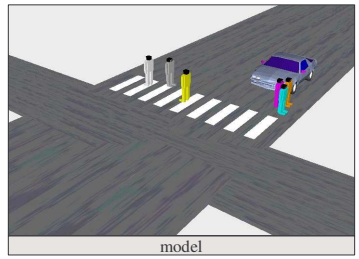


biomimetic pedestrian recognition


raw image


on-fly candidates (pedestrians, heads)


anticipated pedestrians


model