

Agent-Space Architecture

RNDr. Andrej Lúčný

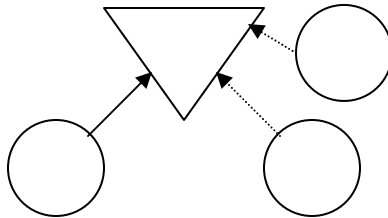
MicroStep-MIS & DAI FMFI UK

andy@microstep-mis.com

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

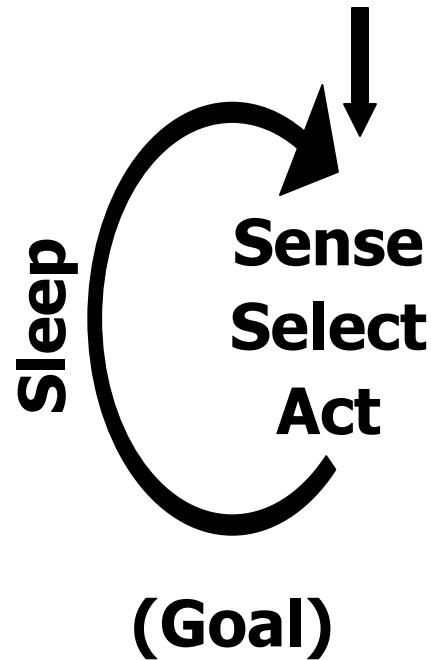
Agent-Space

- System consists of many *reactive agents* which communicate indirectly through an external entity called *space*



Reactive agent

- is a simple entity which constantly performs sense-select-act cycle



Reactive Agent

- Action Selection is based on ordinary single-thread code and does not require any special form or component
- Selected action is a reaction to sensed conditions and internal state of agent
- Internal state correspond to those variables which persist from the current course through loop to the next one

Purely reactivity

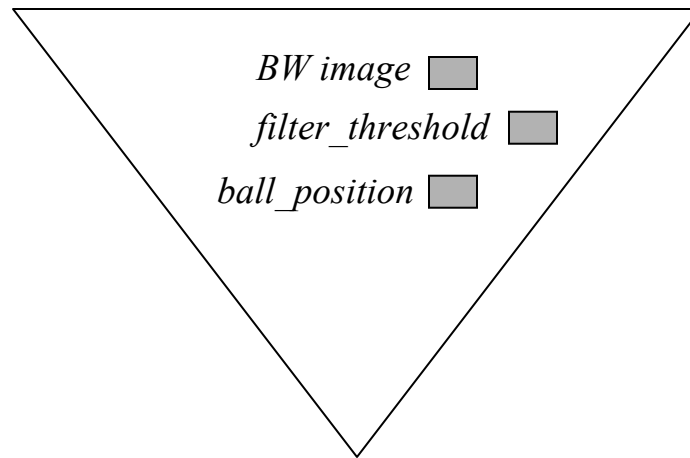
- Purely reactive agent has no internal state
- i.e. no global variables, no global constants,
...

Agent loop invocation

- Primarily agent has a *timer* with a given frequency which invokes it to perform one course through sense-select-act loop
- Additionally agent can exhibit interest to be informed when a change happens in its environment and then it is invoked by *trigger* mechanism

Space

- is a singleton entity which contains and manages named data blocks



Space services

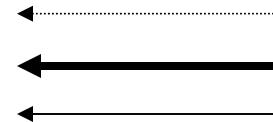
- Space provides to agents services which enable them manipulation with blocks
- The basic services are:

READ

WRITE

DELETE

FREEZE



Space services

- Further services:

TRIGGER REGISTRATION

Various multiply manipulation with blocks
(using wildcards)

- Synchronization: Space is able to perform several services without interruption

Blocks

- are named virtual places where data can be stored
- block can be empty, there is no service for its creation
- block is physically created at the first write, however it is dedicated be read also before this moment (recommended handling: defaults defined by agent)

Blocks

- block has time validity, after its expiration its content disappear automatically (from point of agent view)
- Space has no knowledge about meaning of blocks
- Agent which reads a block has to understand

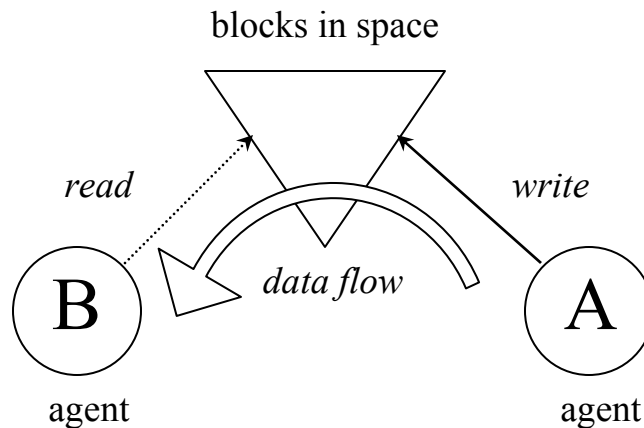
Stored data

Their nature depends on platform, the most relevant strategies are

- buffers of binary data
- Pointers to objects
- Clones of objects
- Marshaled objects
- XML documents
- XML documents mapping Simple Objects
- Mix

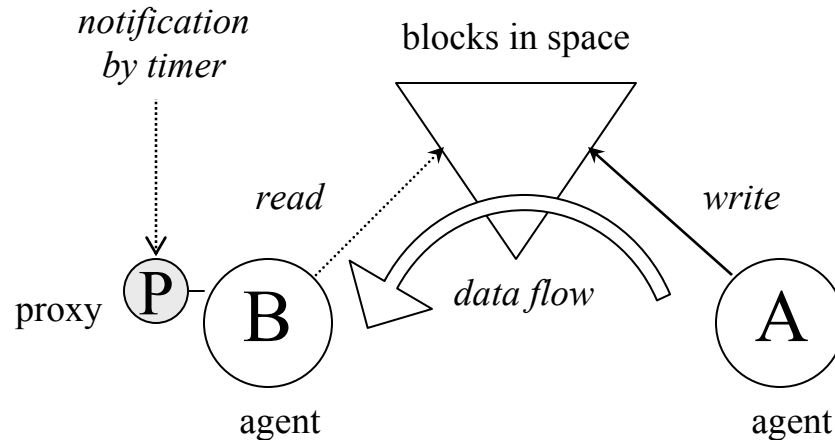
Data flow

Direct communication among agents is not allowed, only data flow from agent to agent can be realized through space – producer writes data, consumer has to read them



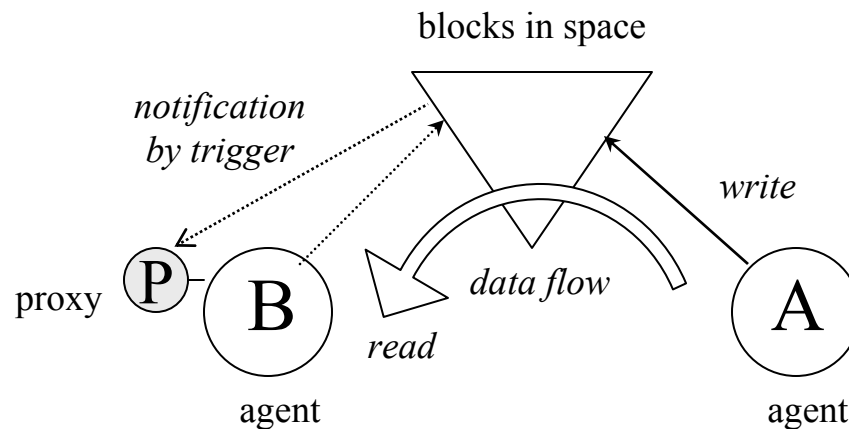
Data flow

Case 1: consumer is invoked by timer – it regularly checks whether there are a new data in the space (or do not deal with their novelty respectively)



Data flow

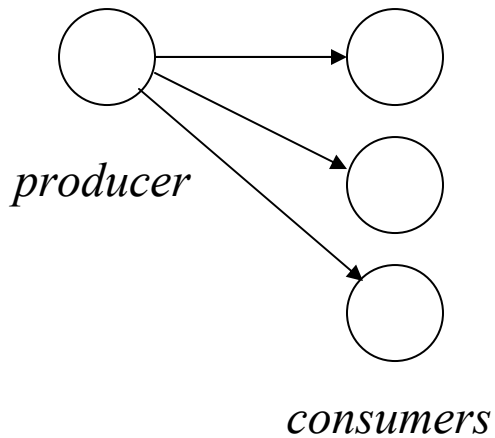
Case 2: agent is invoked by trigger, when a particular block is changed (but! Irregularity of the change is not a good reason to use this mechanism – just necessity to process the change immediately approve us to use it)



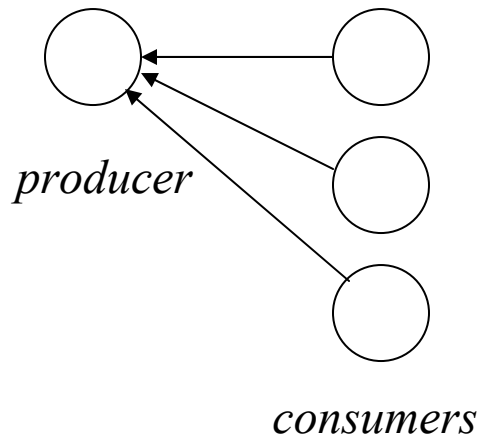
Data flow

advantage: data-flow many:many

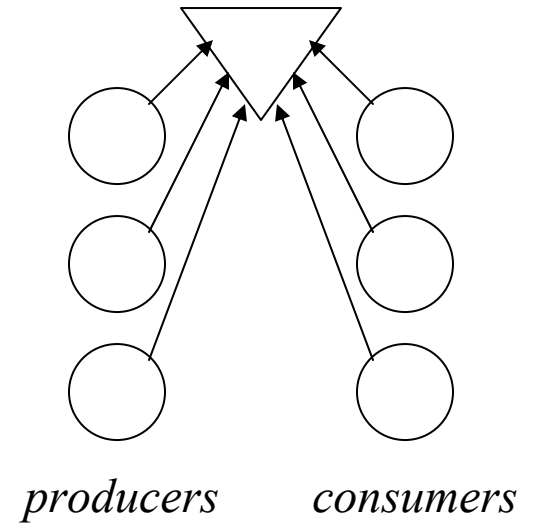
traditional



client-server

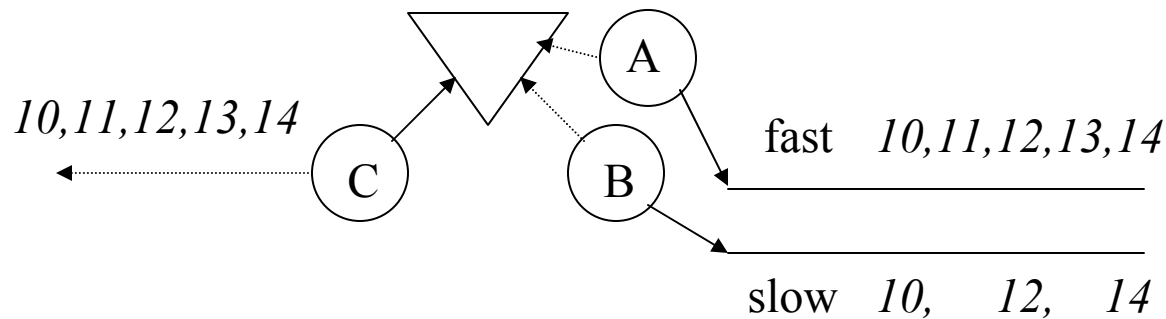


agent-space

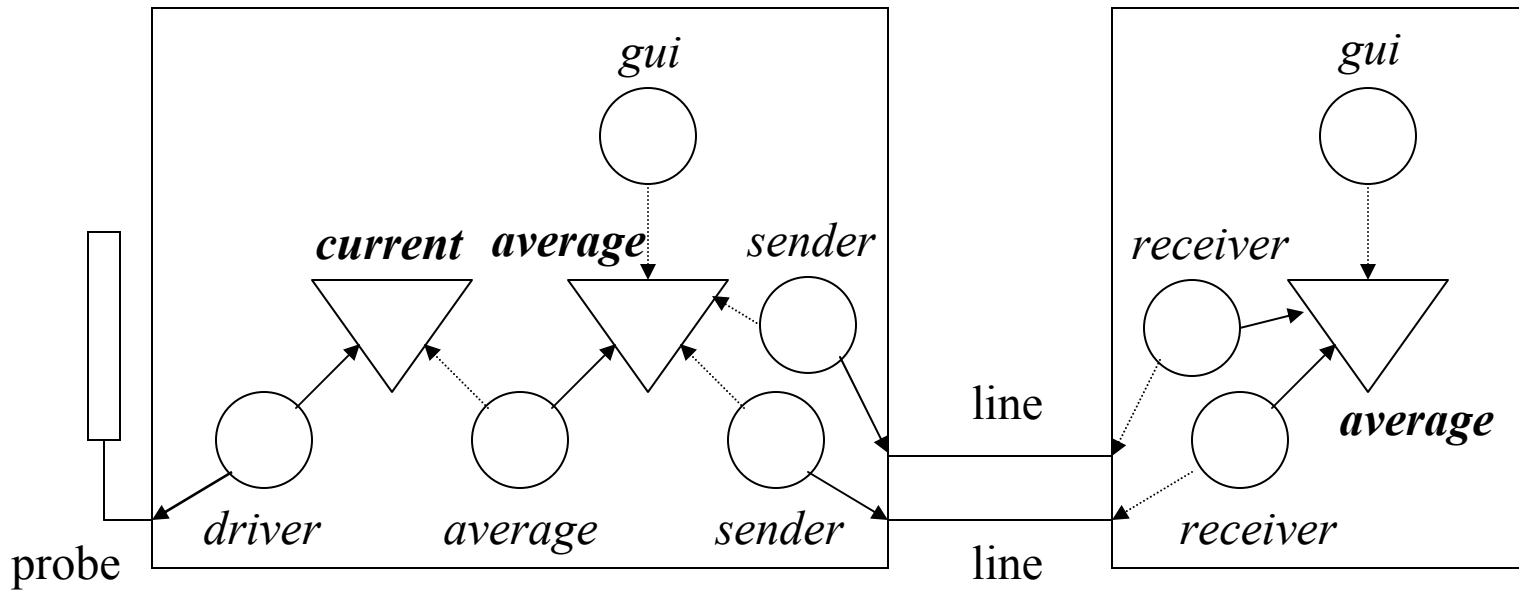


Implicit sampling

Since write operation overwrites data stored in a block regardless their consumers have undertaken them or not, any data flow is inherently (potentially) sampled.



An example



Features

All features are derived from

- Indirect communication
- Agent nature
- Data flow of many:many type
- Implicit sampling

Some of them are valid only for purely reactive agents

Features

- + No deadlocks
- + No management of data propagation
- transient states (inconsistency)

Features

- Data loss
- + Data sampling
- + supports real-time
- + supported by real-time

Features

- + Partial restart possible
- + Recovery from errors
- + Easy to start
- + Ability to configure

Features

- Space is a bottleneck
- + Space is independent from application domain (all code related to application domain is concentrated in reactive agents)
- + Space a re-useable component
- + Space is dedicated to be errorless and reliable enough

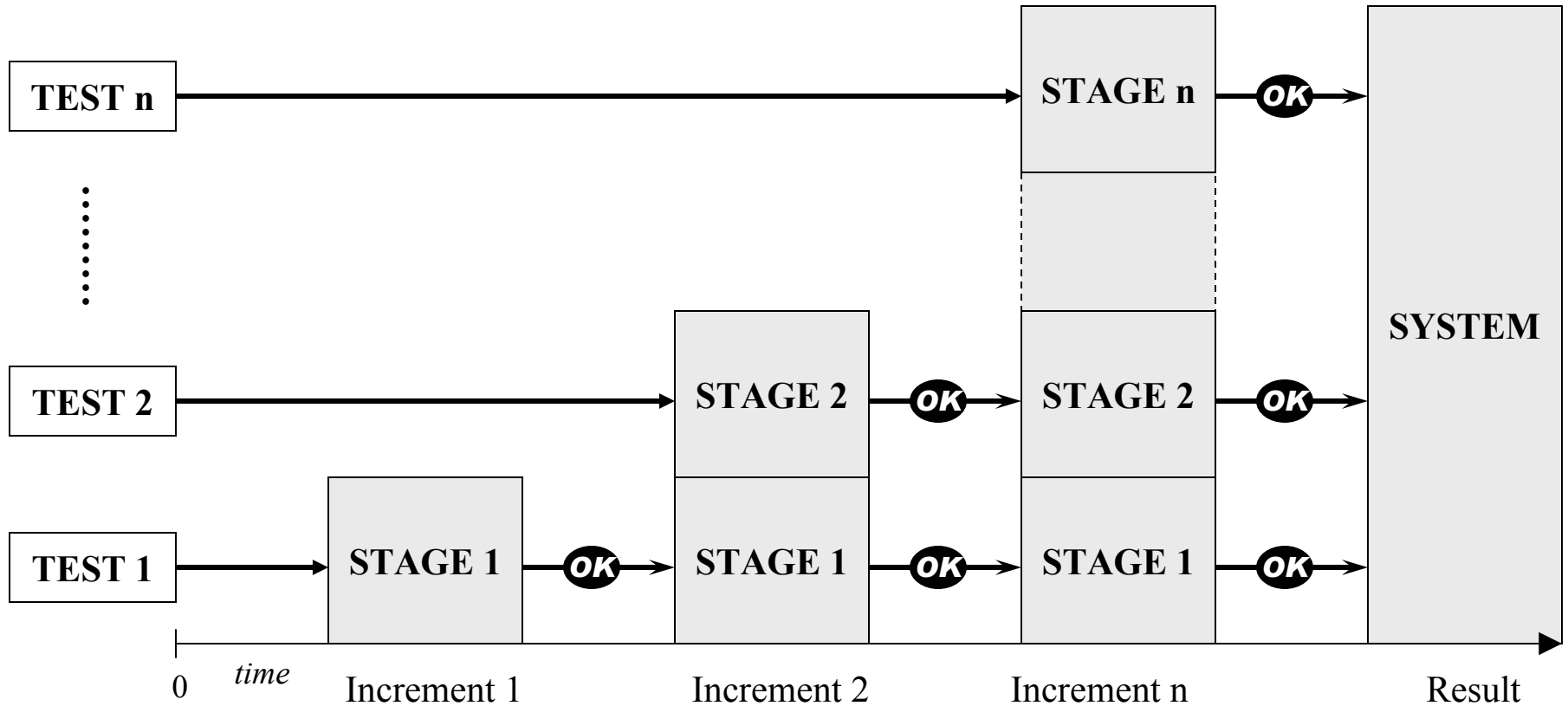
Features

- + system is open
- + a new agent can monitor what legacy agents do and employ results of their operation
- + a new agent can affect cooperation among legacy agents, it can inhibit it or intrude to it and change it completely
- + ability to modify

Development method

- Incremental development
- Bottom-up development
- Decomposition by activity
- Brooks's Subsumption method is applicable
- Some ideas from Fodor's and Minsky's models of mind can be expressed easily
- Modern platform for "New" AI

Development method



Thank you for your attention !

RNDr. Andrej Lúčný

MicroStep-MIS & DAI FMFI UK

andy@microstep-mis.com

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