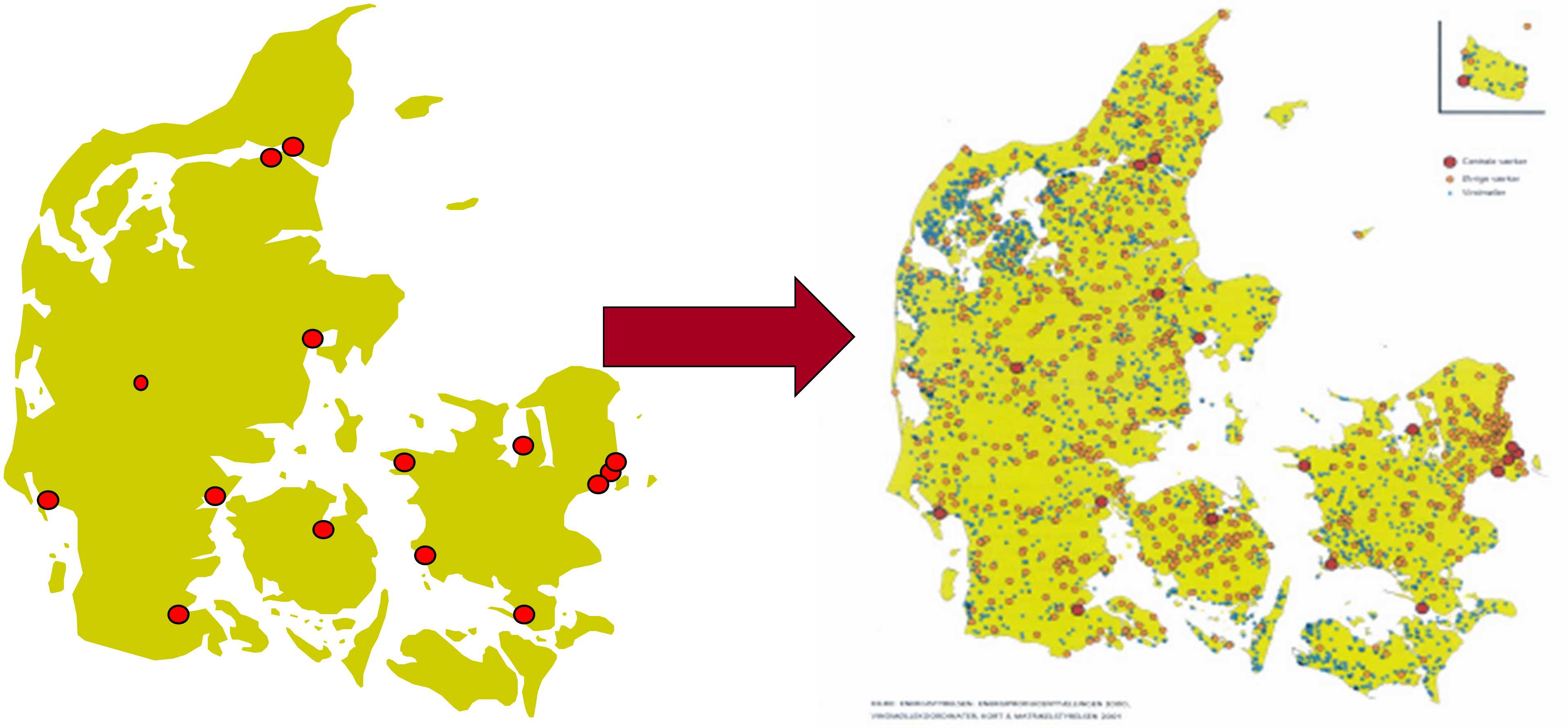
Introduction to Agent and Multiagent Systems

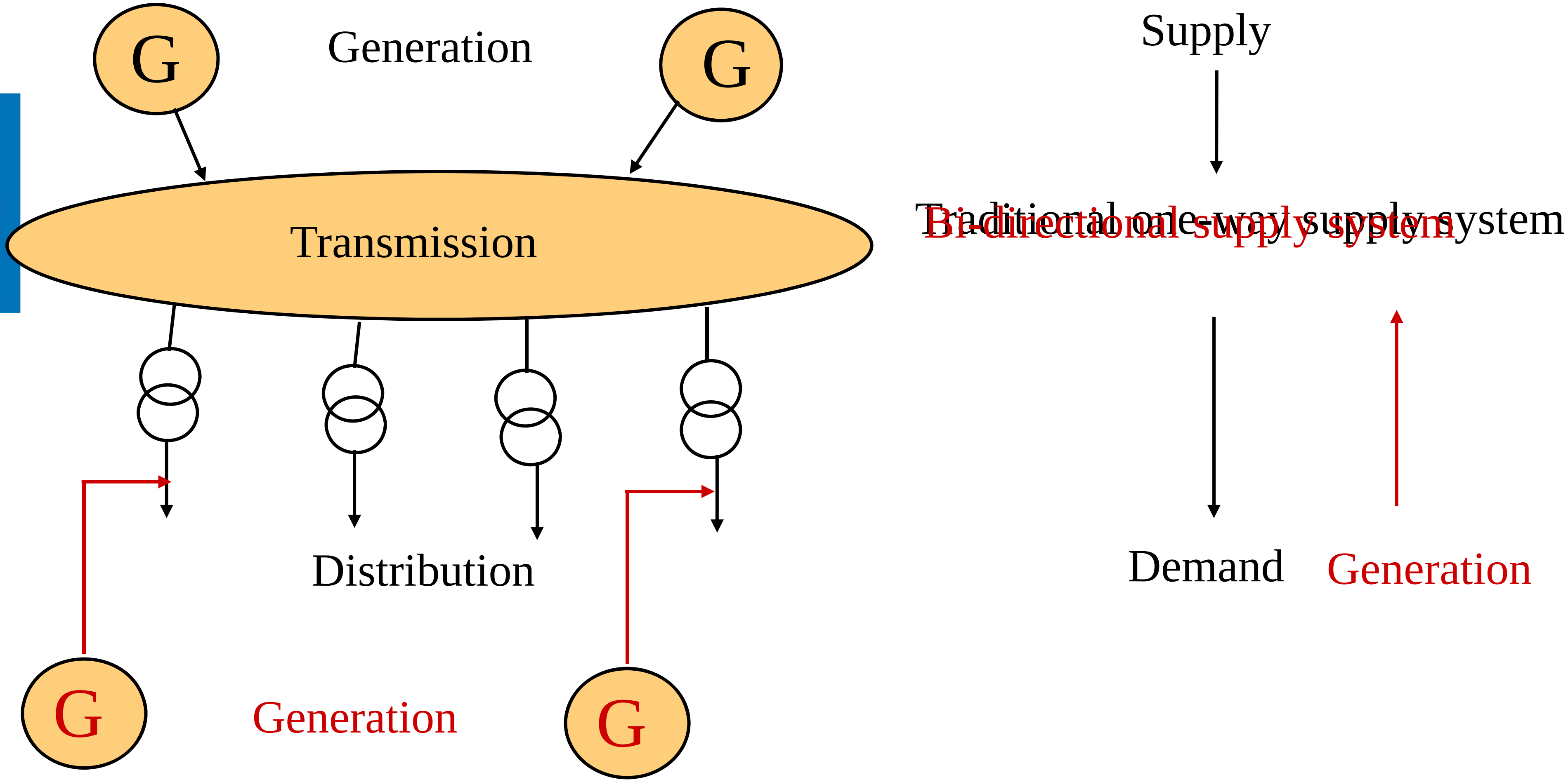
Computer Applications in
Power Systems – Advance course

EH2750

Change In Physical Structure of Power System (Danish example)



Change in Control Structure



New Trends in Power Systems

- **Distributed and local control**
- **A level of intelligence**
- **A shift from reactive to proactive mode**
- **Self-awareness and Intelligence**
    - **Intelligent Software Agents**

What is an agent?

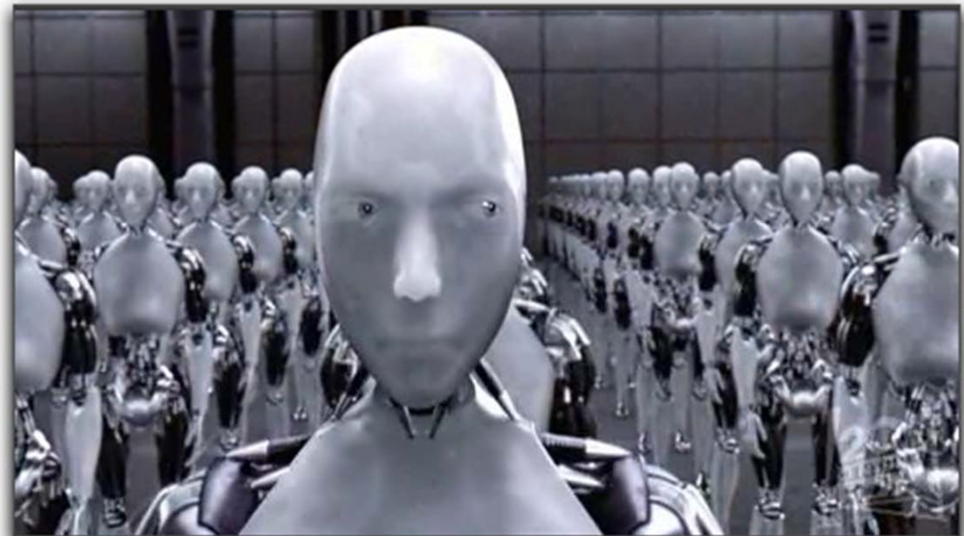
"An agent is an encapsulated computer system that is situated in some environment and can act flexibly and autonomously in that environment, to meet its design objectives"

- Encapsulated
  - Metaphor
  - BDI
- Situated
  - Physical and Software (Virtual)
- Autonomous
- Flexible
  - Proactive
  - Social

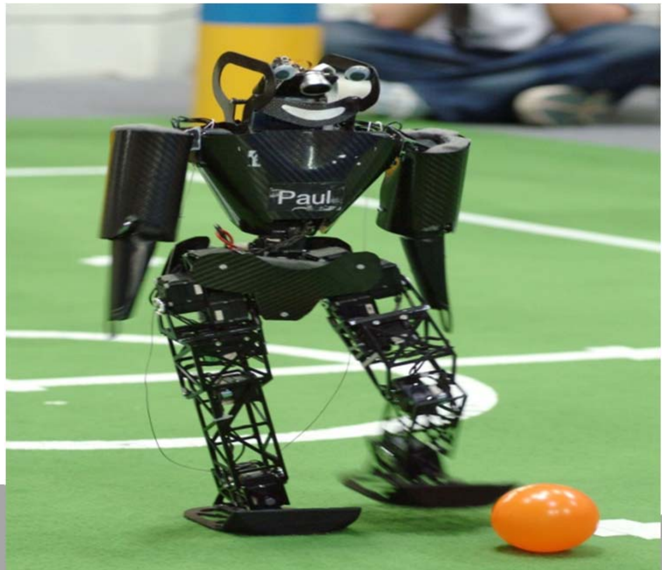
- Design Objective (Goals*)

What does it all means -- What really an agent could be?

Agent can be a Killing machine



And Agents Can Be Built For Very Practical Purposes also



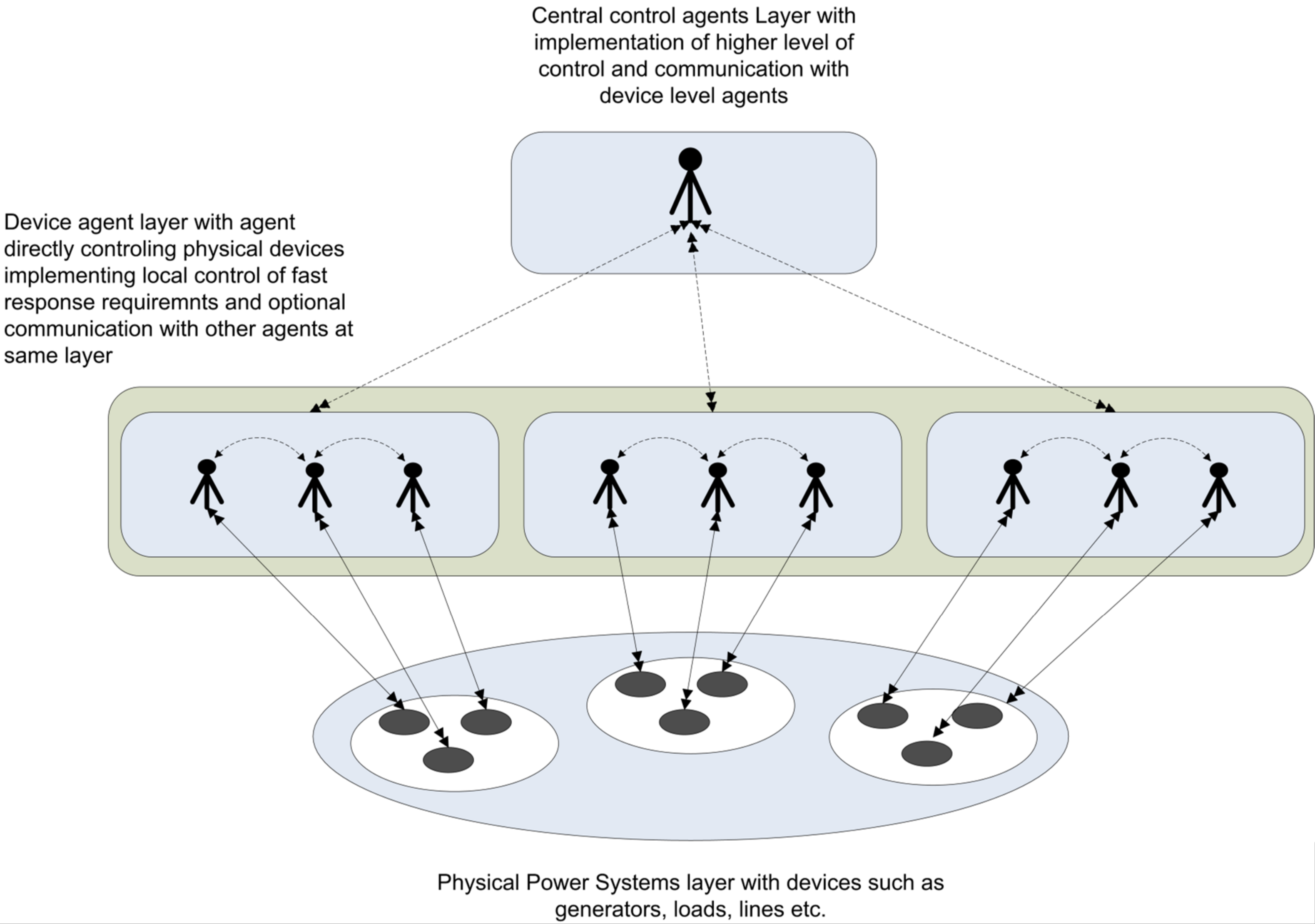
Intelligent Agents

- An agent may also be able to:
  - communicate with other software agents
  - learn from experience
  - adapt to changes in the environment
  - make plans
  - reason using, e.g., logic or game theory
  - move between different computers, and/or
  - negotiate with other agents
- Which of these abilities that are implemented in a particular agent depend on its tasks and purpose

Multi-agent System

"A Multi-Agent System (MAS) is a collection of agents co-operating or
competing with each other in order to fulfill common or individual goals"

Multiagent based distributed control

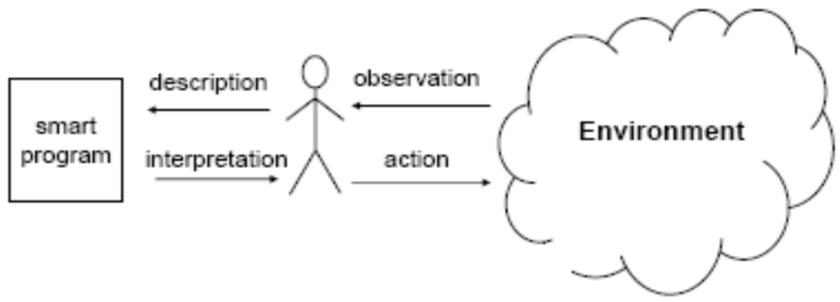


Software vs. Physical Agents

- Software agents
  – situated in a software environment, e.g., operating systems and networks (Internet)
  – *simple example:* software demons (e-mail)
  – *advanced example:* ”shopbots”
- Physical agents
  – situated in the physical reality
  – *simple example: energy saving devices*
  – *advanced example:* mobile robots

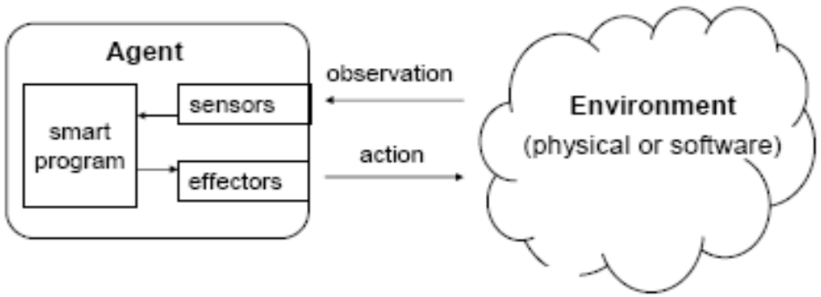
Intelligent Agents vs. Expert Systems

- In expert systems there is a human present between program environment



Intelligent Agents vs. Expert Systems

- Agent reside in the Environment and interacts with it directly (no interface is required)



Agents Vs Objects

- **Agents natural extension \ evolution of Objects BUT with some very fundamental differences**

  - **Level of Autonomy**
  - **Stronger design metaphor**
  - **High Level Interactions – Supporting Organizational structure**
  - **Proactively**
  - **Separate Thread of Execution – Independent life span**

Agents Vs Objects

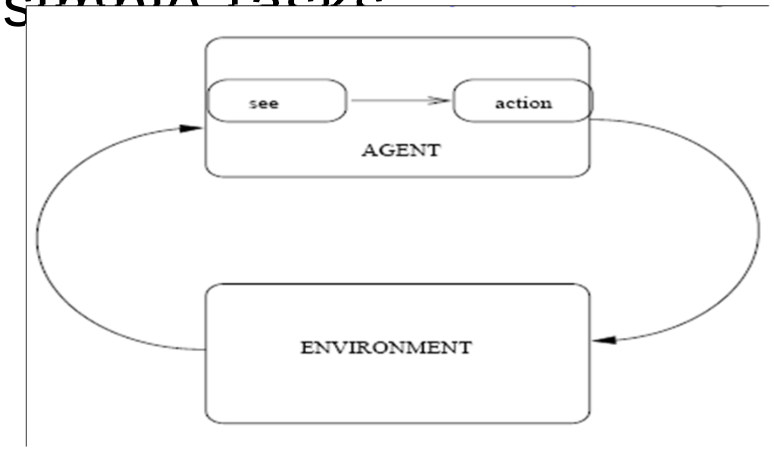
- It is about adding new abstraction entities:
  - OOP = structured programming + objects that have persistent local states
  - AOP = OOP + agents that have an independent execution thread + pro-activity + greater level of autonomy over itself
- An agent is able to act in a *goal-directed* fashion rather than just passively reacting to procedure calls
  - "An agent can say no!"

Agents as intentional systems

- Agents are sometimes modeled, using "mental states", e.g.
  - *Beliefs*, what the agent knows (or think it knows)
  - *Desires*, the goals of the agent
  - *Intentions*, what the agent has decided to do OR how agent is going to achieve desires/intentions

Purely reactive agents

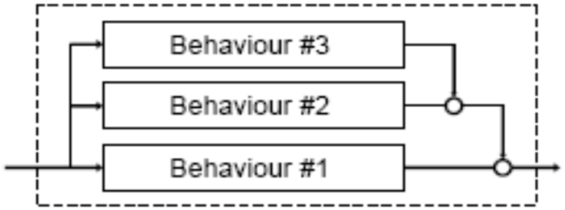
- The simplest kind of agent, appropriate for simple tasks



- Input: sensor data and received messages
- Output: effector signals and sent messages
- The most basic reactive agents are specified by a set of independent situation-action rules.

The subsumption architecture

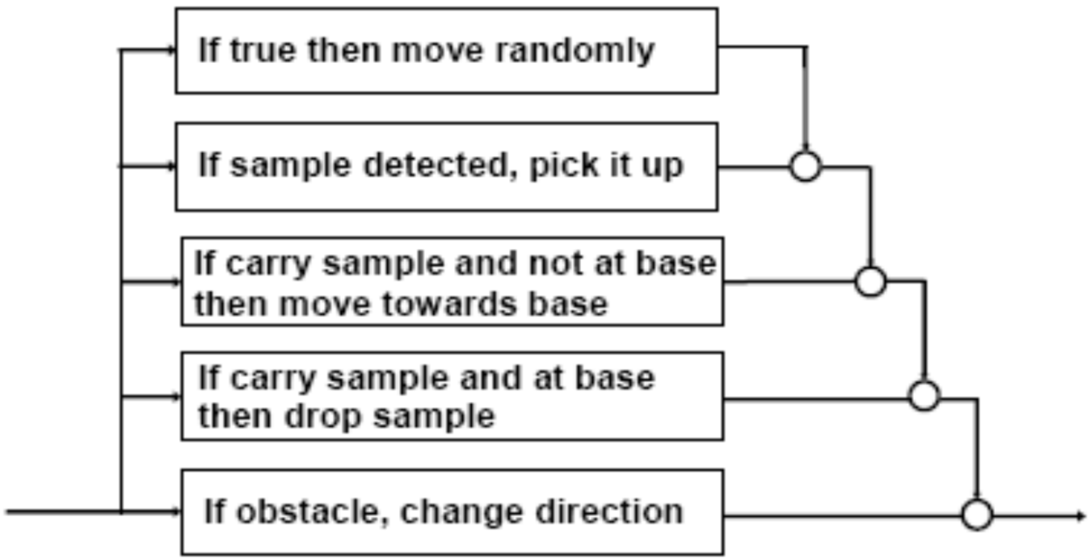
- A hierarchy of behaviors where each behavior is a simple rule-like structure



- The lower behaviors are the most basic and have precedence over higher behaviors
- Bottom-up methodology: start with the most basic behaviors and make them work first

Example: planet exploration

- Task: collect samples of precious rocks



Limitations of Reactive Agents

- Cannot perform tasks that require knowledge about the world that must be obtained by reasoning or from memory, e.g.
  - – response to events beyond the agent's current sensory limits
  - – learning from experience,
  - – some amount of problem solving
  - – prediction of other agents' behavior.
- Each behavior must be separately encoded, which may lead to complexity problems.

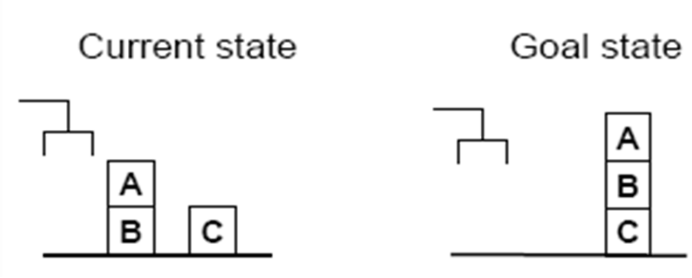
Deliberative agents

- Top-down methodology: begin with the overall architecture and then develop the different components separately.
- Ex. Planning (means-ends reasoning) agent:

Deliberative agents

- Top-down methodology: begin with the overall architecture and then develop the different components separately.
- Ex. Planning (means-ends reasoning) agent:

A simple planner



- Possible actions:
  - pickup(What) and putdown(Where)
- Resulting plan:
  - pickup(A), putdown(table), pickup(B), putdown(C), pickup(A), putdown(B)

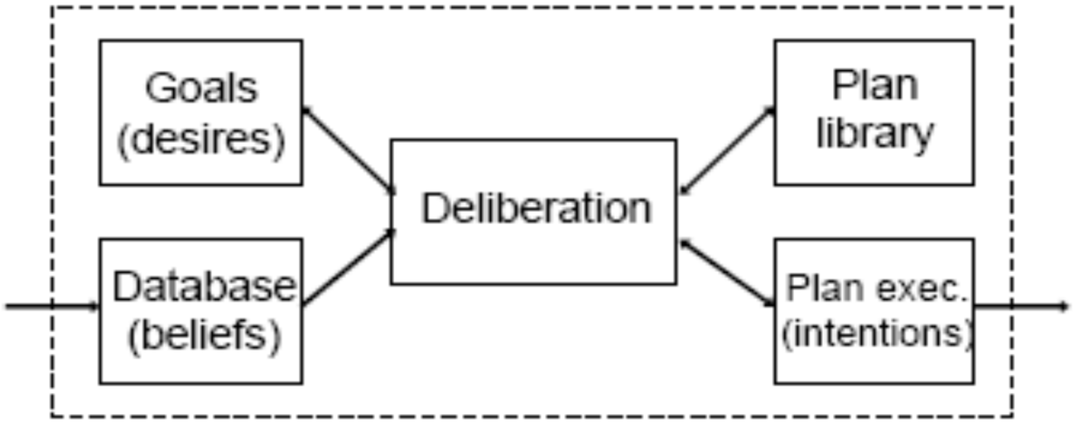
- Most of real world problems deals with finding a Plan(sequence of actions)

Limitations of planning agents

- Typically, a plan is constructed by searching through the space of possible action sequences until one is found that will transform the current state into the goal state
- But this is very time-consuming (NP-complete)
- Requiring search through potentially enormous problem spaces
- The situation may change while the agent is constructing its plan, making it useless
- May be suitable for abstract tasks, but have problems with "simpler" concrete tasks that require fast reaction (but no deliberation)

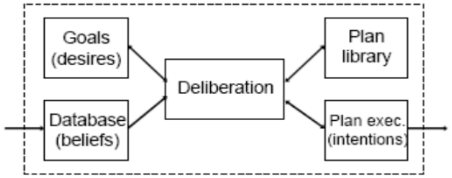
Procedural Reasoning System

- Use pre-complied plans!



- The plans are not just a sequence of actions but can also contain [sub] Goals

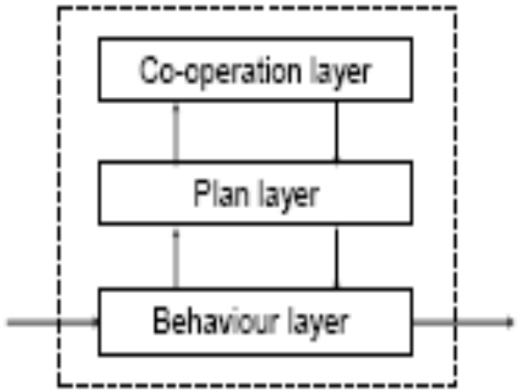
Basic functionality of PRS



- A plan has three componen
  – the goal: the post-conditi
  – the context: the pre-conditions
  – the body: a sequence of actions and/or goals
- Simplified algorithm:
- – select a goal from Desires and push it to the Intention stack
- – while not Intention stack empty
  - pop the Intention stack
  - if it is an action, perform the action
  - if it is a goal, (and it's not already fulfilled) find a plan that matches the goal and push the body to the Intention stack

Hybrid layered agents

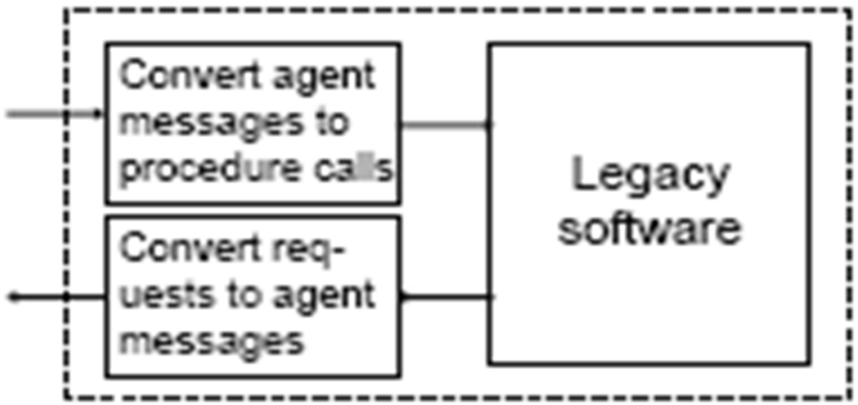
- **Combines reactive and deliberative behavior**



- A lower layer passes control to a higher layer when it is not competent to deal with the current situation.

Wrapper agents

- **An approach to "agentify" legacy software systems, such as, expert systems, database handlers, and control software**
- **The wrapping software makes the legacy code appear as a normal agent to other agents.**



Agent Communication Languages (ACLs)

- To interact effectively the agents need to be able to communicate with each other
- Most ACLs builds upon the *speech act theory*, which views language as *actions*.
- Examples: KQML and FIPA ACL
- They support a set of *performatives* that defines the permissible communicative operations that agents may attempt such as:
  – informing, asking questions, and commanding

FIPA ACL message

```
(inform
   :sender agent1
   :receiver auction-server
   :content
        (price (bid good02)
   150)
   :in-reply-to round-4
   :reply-with bid04
   :language sl
   :ontology auction
)
```

Sample KQML Message and Response

```
(ask-one
:sender student
:content
   'price(beer,Price)'
:receiver beer-server
:language Prolog
:ontology beer-bongs
)
```

```
(tell
:sender beer-server
:content 'price(beer,45)'
:receiver student
:language Prolog
:ontology beer-bongs
)
```

Main arguments for choosing an agent based approach

- Methodology - new level of abstraction enables the implementation of more complex control strategies
- Robustness - often no critical single point of failure
- Efficiency - less complex computations
- Flexibility - ACLs supports complex interaction
- Scalability - easy to add new agents to a MAS

When to use agents

- Modular - possible to identify distinct "entities"
- Decentralized - entities can perform useful tasks without continuous direction from some other entity
- Changeable - the structure of the system may change quickly and frequently
- Ill-structured - all information is not available when the system is being designed
- Complex - the system needs to exhibit a large number of different behaviors which may interact in sophisticated ways