Introduction to Agents and Multi-Agent Systems

Agent-Oriented Software Engineering

A.A. 2019-2020

Prof. Paolo Giorgini

UNIVERSITY OF TRENTO - Italy

Department of Information and Communication Technology

New perspective

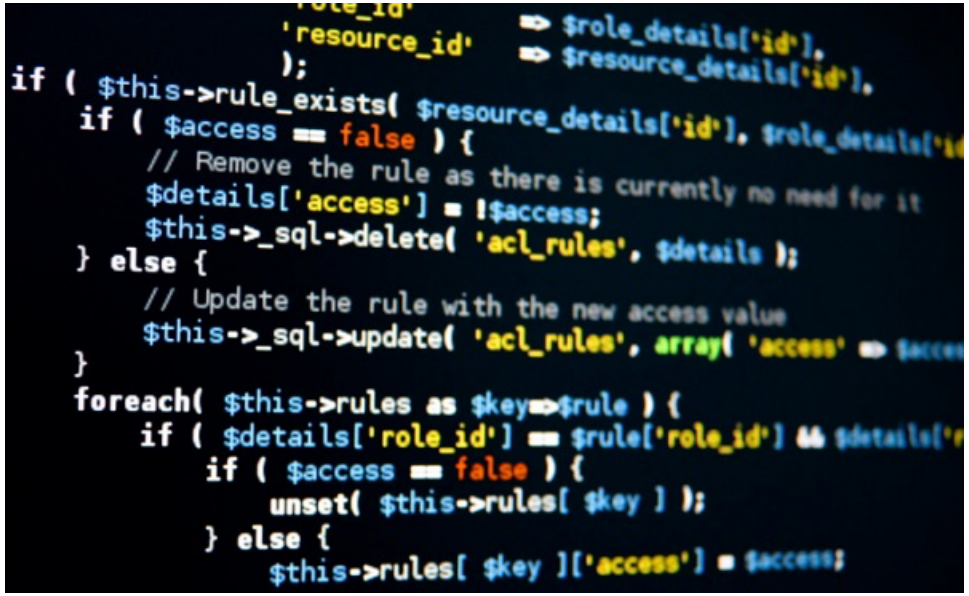
- Computers are not very good at knowing what to do
  - Every action a computer performs must be explicitly anticipated, planned for, and coded by a programmer
- They fail when they encounter a situation that their designers did not anticipate
  - they crash and may cause system crush and loss of life at worst
- Growing interest in developing software that may decide for themselves what they need to do in order to achieve their objectives
  - able to change their behavior in rapidly changing, unpredictable, or open environments

Examples

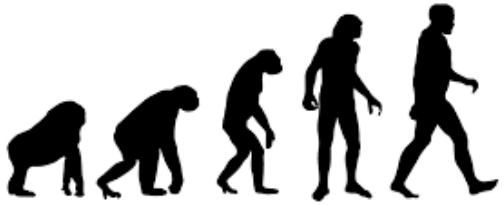
- Space probe making its long flight to outer planets
  - Ground crew is required to continually track its progress and decide to deal with unexpected eventualities
  - Very costly and if decisions are required quickly, it is simply not practicable
  - NASA and ESA are interested in the possibility of making probes more autonomous (richer onboard decision-making capabilities and responsibilities)
- Searching the Internet for the “best” way to spend a week-end with the fiance
  - can be a long and tedious process
  - Why not allow a computer program (an agent) do search for us?

Software product trends

- Past: centralized computing system; monolithic programming model
- Now: distributed computing system; heterogeneous, scalable, open and distributed programming model
  - Ubiquity
  - Interconnection
  - Task delegation
  - Social abilities
  - Context awareness
  - Intelligence
  - Goal-oriented…



Human Orientation



- Trend to represent software systems using concepts and metaphors closer to human models
  - Programmers tend to conceptualize and implement software in terms of higher-level – more human-oriented – abstractions
    - The object abstraction is not enough to model a piece of software
- Socio-technical systems
  - Technical components and human actors are part of the actual system and its design – no conceptual distinction between them
  - Strong interdependency between human actors and software components

Delegation

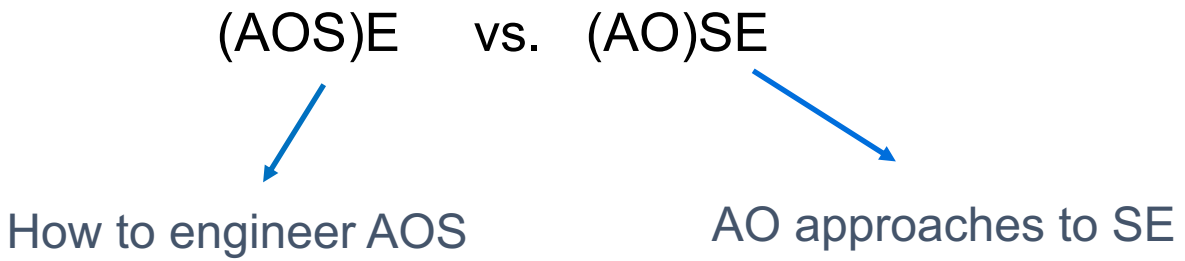
- Behave and act on human behalf
  - act independently
  - represents human best interests while interacting with other humans or systems
- Interconnection + Distribution + Delegation
  - ➔systems able to *cooperate* and *reach agreements* (or even *compete*) with other systems that have different interests (much as we do with other people)
- Delegation and control
  - Unpredictable behavior
  - Emergent and collective behavior

Development is getting harder

- Very ambitious requirements
  - E.g., Intelligent and human-like software
- Less certain requirements
  - Greater scope for change
  - More interdependent requirements
  - Socio and Organizational constraints
- More challenging environments
  - Greater dynamism
  - Greater openness
- Lack of appropriate design models

Software Process trend

- Past: waterfall development process model
- Now: incremental, agile, experimental development process model,…
  - **Agent-based software development:** new way of analysis and synthesis of software systems (supposed to be …)
  - **Agent-Oriented Software Engineering**
    - Based on the agent paradigm



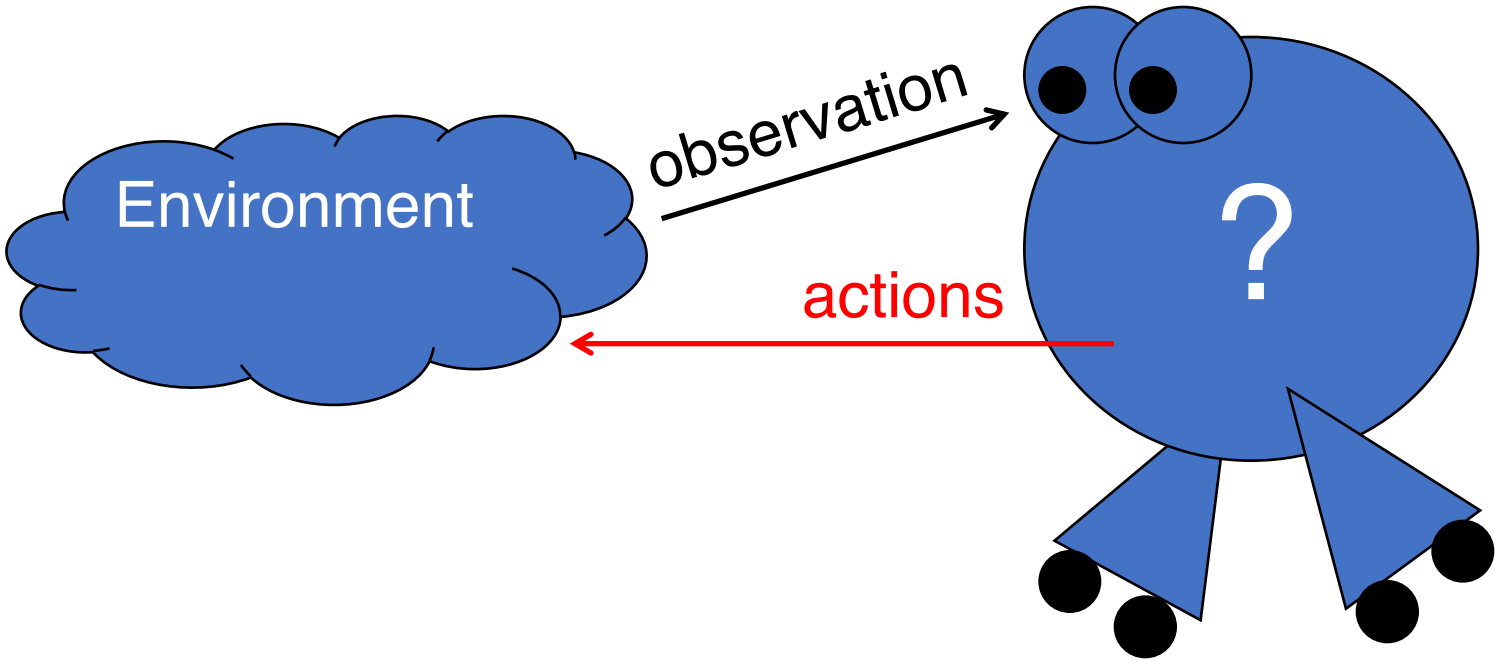
Agent? What is?

- Software agent, intelligent agent, autonomous agent, reactive agent, bla, bla …
- Very controversial definitions for the agent concept
  - *"…encapsulated computer system, situated in some environment, and capable of flexible autonomous action in that environment in order to meet its design objectives"* [Wooldridge]
    - autonomous: control over internal state and over own behaviour
    - situated: experiences environment through sensors and acts through effectors
    - flexible
      - reactive: respond in timely fashion to environmental change
      - proactive: act in anticipation of future goals

More Agent definitions

- More AI definition:
  - "*An agent is a computer system that is capable of independent action on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told)*" [Wooldridge&Jennigs]
  - "*An agent is a software entity that functions continuously and autonomously in a particular environment, often inhabited by other agents and processes*" [Shoham 97]
- More Software Engineering definition:
  - *"an agent is a software component with internal (either reactive or proactive) threads of execution, and that can be engaged in complex and stateful interaction protocols"*

An agent its environment



The environment

- An agent has not complete control over its environment
  - Partial control only when can influence its environment
  - The same action performed twice in apparently identical circumstances might appear to have entirely different effects (fail to have the desired effect)
- Classification of environment properties:
  - Accessible vs. inaccessible
  - Deterministic vs. non-deterministic
  - Episodic vs. non-episodic
  - Static vs. dynamic
  - Discrete vs. continuous

An example of agent

(neither autonomous or intelligent)

- Any control system can be viewed as an agent
  - Thermostat has a sensor for detecting the room temperature
  - Temperature is too low or OK (two possible values)
  - The thermostat can perform “heating_on” or “heating_off”
  - Of course heating_on action cannot guarantee to increase the temperature (partial control)

```
too cold → heating on
temperature OK → heating_off
```

Making it a bit more complex

- Thermostat + Automated Windows

```
too cold → heating on
temperature OK → heating_off
```

```
Stale air → open windows
Air is ok → close windows
```

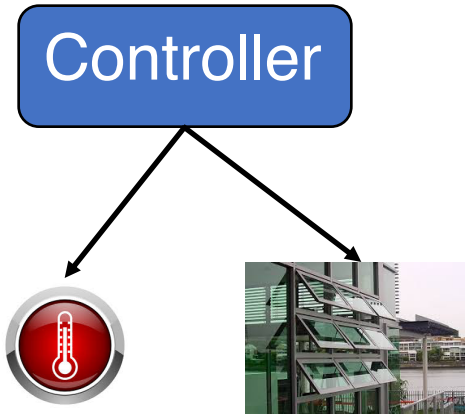
- Does it make sense to keep the heater on when windows are open?
  - Thermostat and Windows should be coordinated

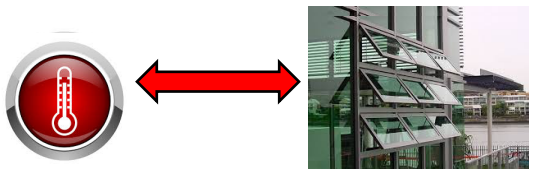
Solution?

`too cold ^ windows closed → heating on`

`temperature OK → heating_off`

Who is checking `(too cold ^ windows closed)?`





Consider also this

Can you close the window?

why she is asking me so?

```
window_closed → temp_ok
window_closed → noise_ok
```

may she want to increase the temperature or there is too much noise in the room?

```
feel_cold → NOT temp_ok
NOT noise_ok → NOT follow_the_lecture
dress_jumper -> NOT fell_cold
```

She is asking because she wants follow_the_lecture

Intelligent agents

- Proactiveness: ability to exhibit goal-directed behavior by **taking the initiative** in order to satisfy their delegated objectives
- Reactivity: ability to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their delegated objectives
- Social ability: ability of interacting with other agents (and possibly humans) in order to satisfy their design objectives

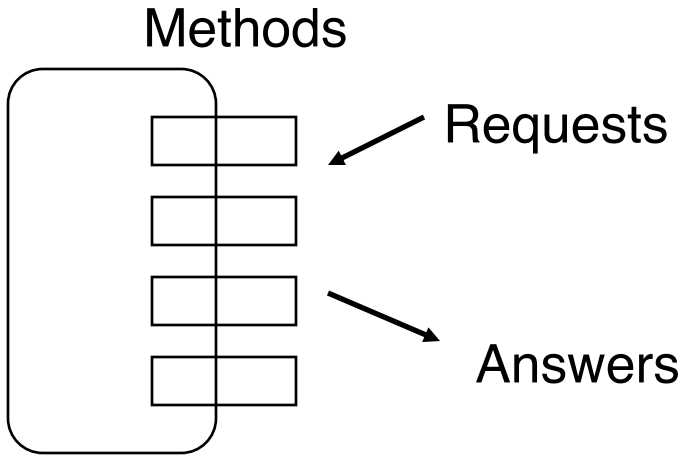
More Agent attributes

- Collaborative (i.e., work with other agents and entities to achieve a common goal)
- Knowledge-level communication ability (i.e., communicate with other entities in a language like speech-act, higher level than symbol level program to program protocols)
- Inferential capability (i.e., act on abstract task spec., using models of itself, situation, and/or other agents)
- Temporal continuity (i.e., persistence of state)
- Personality (i.e., manifesting attributes of a believable agent)
- Adaptability (i.e., learn and improve with experience)
- Mobility (i.e., migrate from one host to another in a self-directed way)

Object vs. Agent

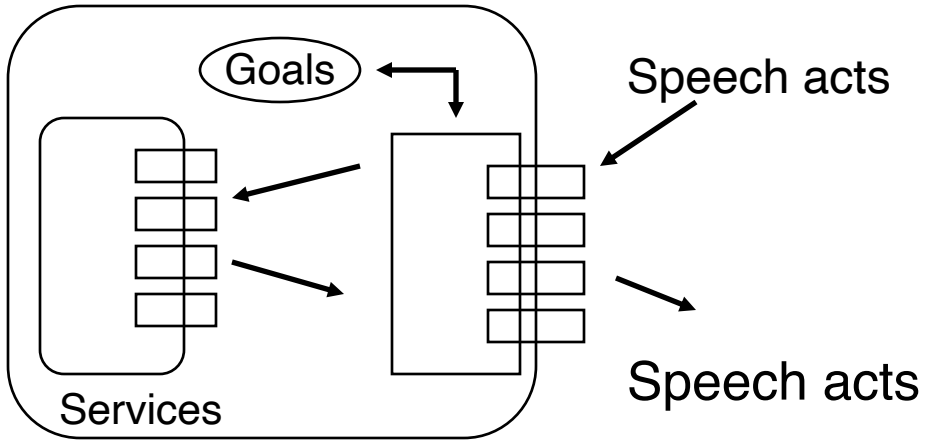
An **object**:

- Has methods (services)
- Message transmission
  - Method invocation



An **agent**:

- Has goals (goal-driven)
- Has skills (or services) to achieve goals
- Symbol level communication



[Ferber'99 revised]

Object vs. Agent (2)

- In objects methods can be public (not only private)
  - An object can exhibit autonomy over its state (encapsulation), but it does not exhibit control over its behaviors
  - An object activates the method of another object while an agent requests another agent to perform an action
- Agents can be implemented using objects-oriented techniques
- Standard object model has nothing whatsoever to say about reactiveness, proactiveness and autonomy
- In standard object model there is a single thread of control in the system while each agent has its own thread
  - Similar to active objects – they are agents that do not necessarily have the ability to exhibit flexible autonomous behavior

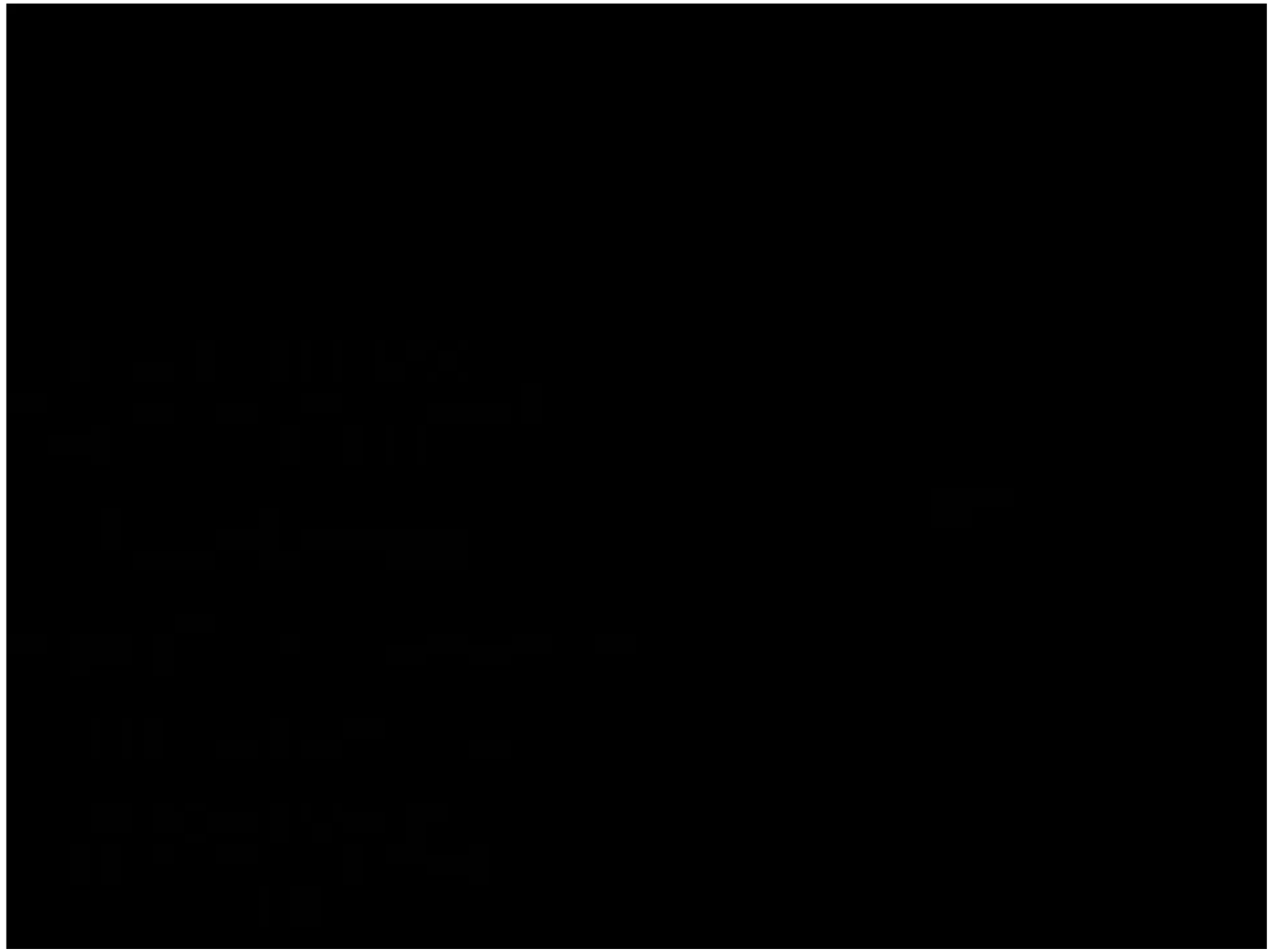
Role/Goal vs. Task

- We assign “tasks” (or services) to software
  - Example: “on-line registration software”
  - “What” and “how” are specified in advance
  - Changes in requirements are not tolerable
- We assign “roles/goals” to agents
  - Example: registration agent
  - “What” is specified in advance. “How” is determined dynamically
  - Changes in requirements can be tolerated
- An agent selects and executes tasks at runtime
  - Goal --> Tasks
- Higher level of conceptualization

More then a single agent

- In most cases, a single agent is not enough
  - No such thing as a single agent system (!?)
  - Multiple agents are the norm to represent
    - Decentralization
    - Multiple loci of control
    - Multiple perspectives
    - Competing interests
- A Multi-agent system definition
  - A multi-agent system is one that consists of a number of agents, which interact with one-another
  - In the most general case, agents will be acting on behalf of users with different goals and motivations
  - To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do

A coordination example



Another definition of MAS

- *"… If a problem domain is particularly complex, large, or unpredictable, then the only way it can reasonably be addressed is to develop a number of functionally specific and (nearly) modular components (agents) that are specialized at solving a particular problem aspect. When interdependent problems arise, the agents in the system must coordinate with one another to ensure that interdependencies are properly managed"* [Katia Sycara]

- Basics of a MAS:
  - Interaction
    - Cooperation, Coordination, Competition
    - Communication
  - Local and organizational interests
    - Agent objectives vs. organizational (social) objectives
  - Organizational structure
    - Overall structure, agents' roles, norms, rules, etc.

Agent Interactions

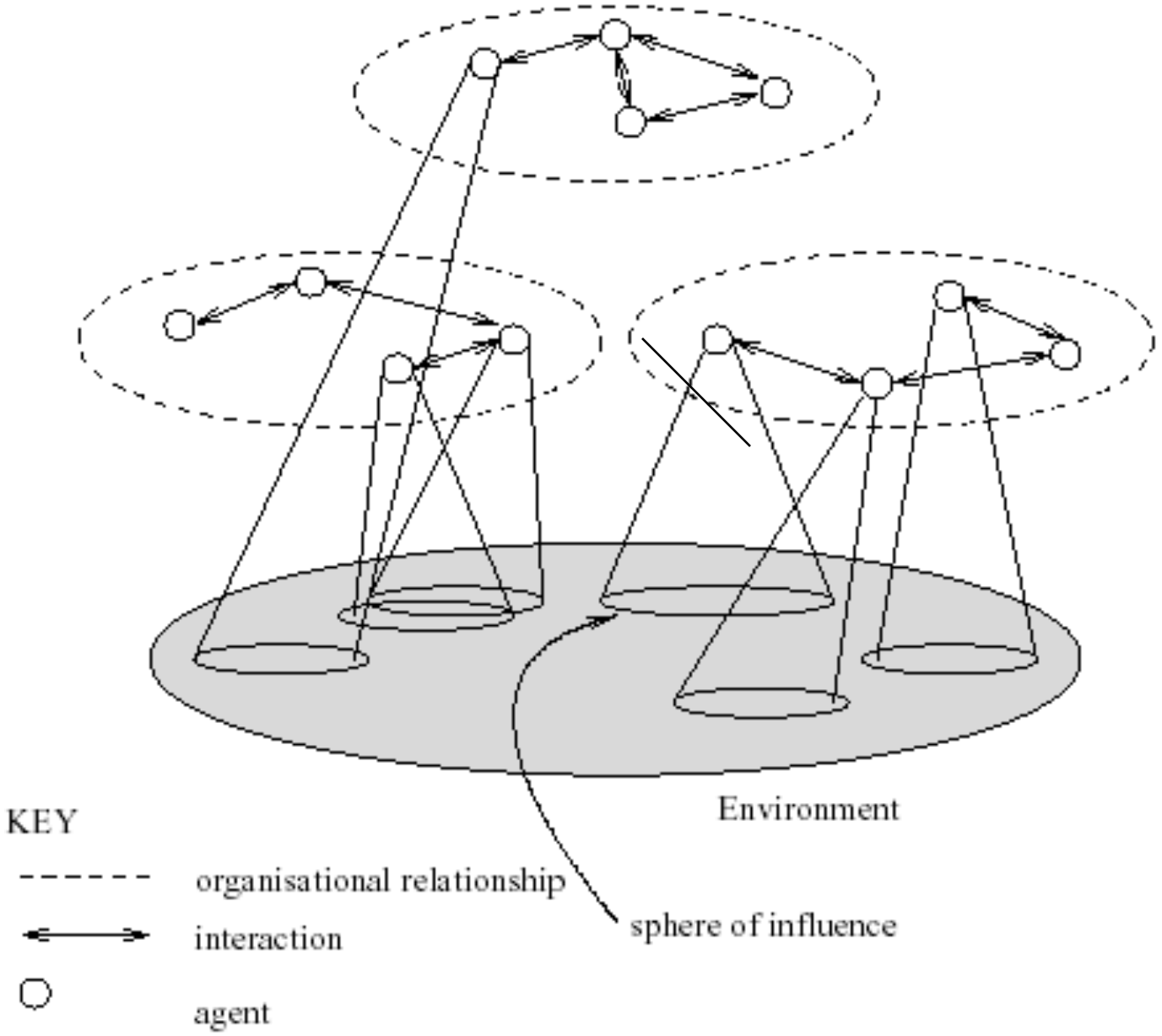
- Interaction between agents is inevitable
  - to achieve individual objectives
  - to manage inter-dependencies
- Conceptualised as taking place at knowledge-level
  - which goals, at what time, by whom, what for
- Agents act to achieve objectives
  - on behalf of individuals/companies
  - part of a wider problem solving initiative
- Agents interact in some organisational setting
  - organisational relationship between agents

Organisations

- The organisational context
  - Influences agents' behaviour
  - Relationships and rules need to be made explicit
    - Peers, teams, coalitions, institutional norms
- Agents act and interact in organisational institutions
  - They perform particular roles (which may change)
  - They obey particular norms and regulations (which are explicitly stated)
  - Different levels of an organisation
    - E.g., Strategic level, management level, operational level
  - Organisational processes

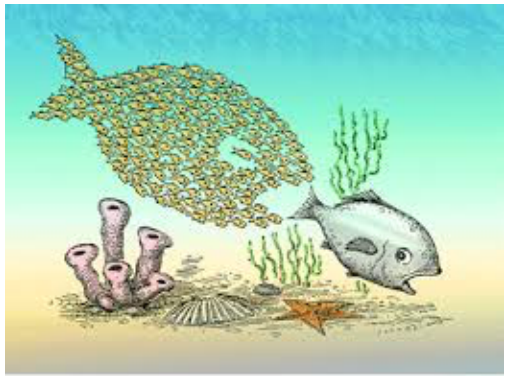
A canonical view

- A multi-agent system contains a number of agents…
  - …which interact through communication…
  - …are able to act in an environment…
  - …have different "spheres of influence" (which may coincide)…
  - …will be linked by other (organizational) relationships



Emergent behaviour

- Multi-agent system behaviour is not fully predefined
  - Behavior of a system that is not explicitly described by the behavior of the components of the system, and is therefore unexpected to a designer or observer.
- Result of dynamic interaction among the participants
  - flocking of birds cannot be described by the behavior of individual birds
  - market crashes cannot be explained by "summing up" the behavior of individual investors



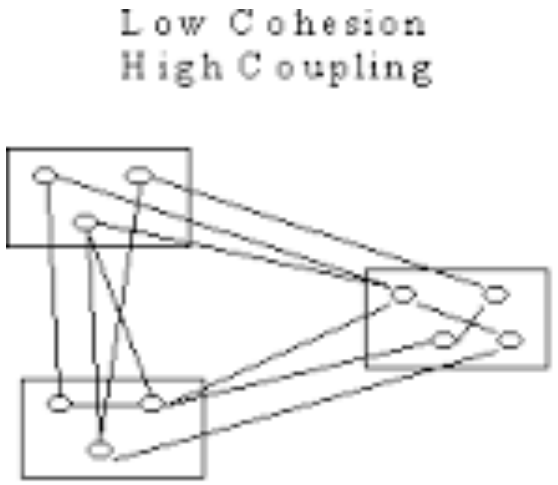
- Local objectives generate emergent behaviours
  - Depending on the social/organizational context
  - Depending on the dynamics
  - Openness of the system

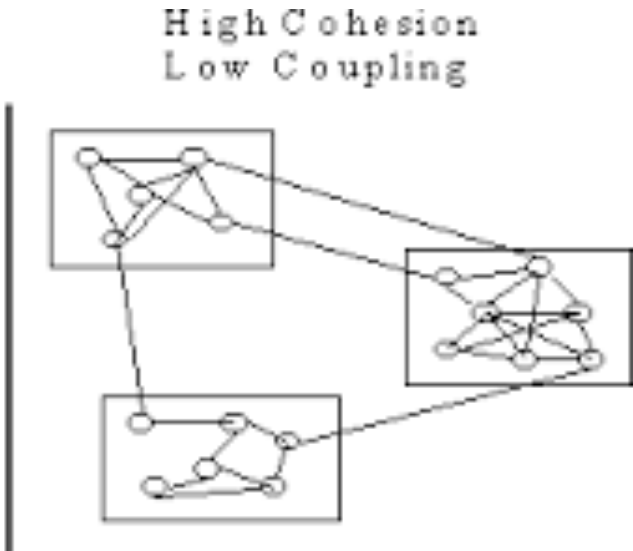
Interactions vs. Intra-actions

- In software systems, relationships among subsystems (or components, packages) are:
  - Intra-actions within a subsystem
  - Inter-actions among subsystems
- Intra-actions
  - conventional software engineering approaches handle mainly with intra-actions within each component
- Software trend goes more and more in the direction of Inter-actions
  - Service-oriented computing, P2P, B2B, etc.
  - predictable at design time (Distributed System Engineering)
  - not predictable at design time (Agent-based Engineering)

Software Engineering Principles

- **Cohesion**, the degree of interdependence between software modules; a measure of how closely connected two modules are; the strength of the relationships between modules
  - Maximize the cohesion
- **Coupling**, a measure of the strength of interconnections between program units
  - Minimize the coupling



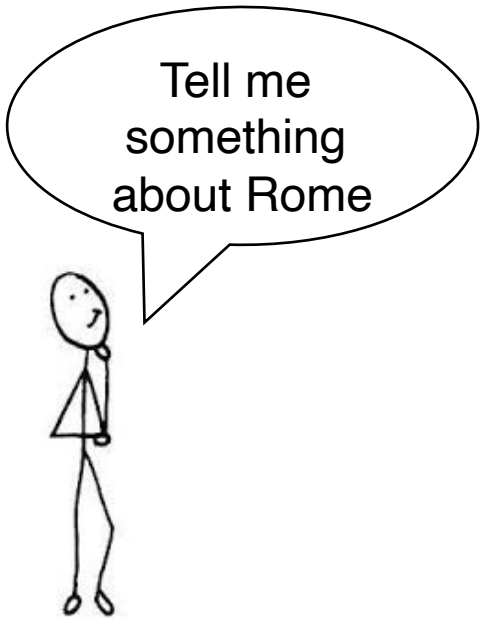


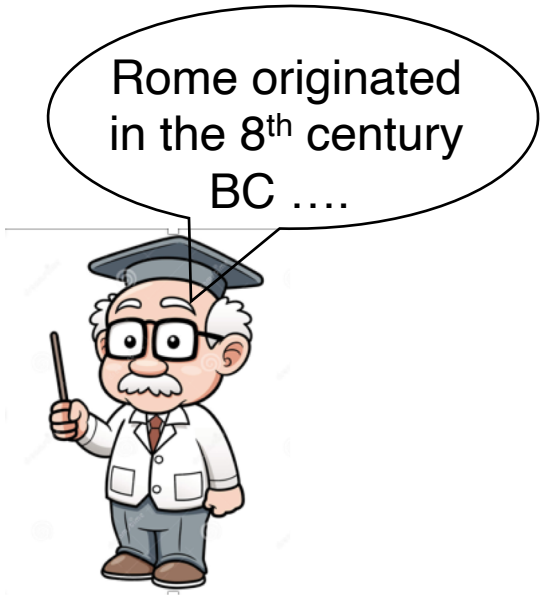
Agent communication (1)

- Basis of interaction and social organization in MAS
- Not just sending messages
  - Communication is carried out by signals between agents
- Significance
  - To be significant there must
    - be an agent that can perceive the message
    - be an interpretive system that can transform the message into meaning

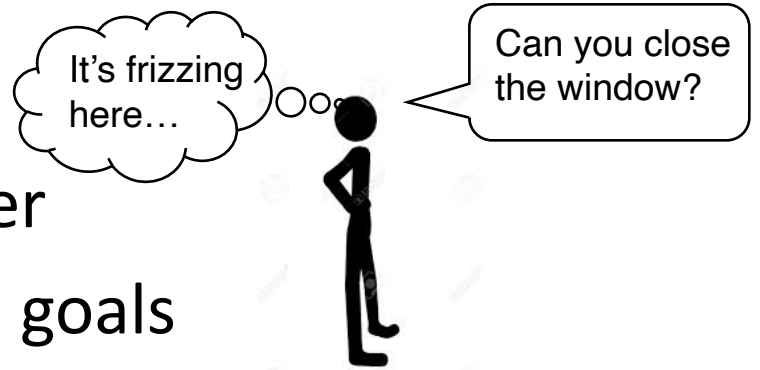
Agent communication (2)

- communication consists of
  - Sending of information from a sender to a receiver
  - Encoding/decoding of information in a language
  - Transmission of information on channel or medium
  - A context in which the interlocutors (sender and receiver) are place





Agent communication (3)

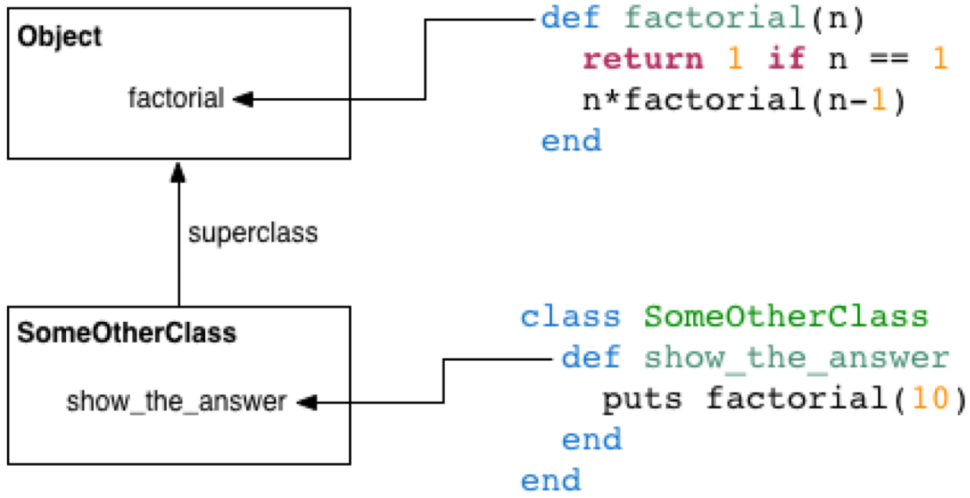


- Characterize the attitude of the sender
- Indicate agent's state, intentions, and goals

- Allow agents to coordinate and make common belief
- Allow order/request & acceptance/refusal
- Send data related to the state of the world
- Verify channel function via acknowledge functions, etc.
- Request for clarification/more detail
- Ensure messages are properly understood

Knowledge completeness

- Interacting (traditional) software systems must have complete knowledge of each other



- Interacting software agents may have complete knowledge about the other agents' goals, strategies (i.e., actions to select from) and utilities (i.e., the pay-offs of actions)
- Interactions with
  - complete knowledge --> cooperation and coordination
  - non complete knowledge --> competition

Basic questions

- In Multi-Agent Systems, we address questions such as:
  - How can cooperation emerge in societies of self-interested agents?
  - What kind of languages agents can use to communicate?
  - How can self-interested agents recognize conflict, and how they can (nevertheless) reach agreements?
  - How can autonomous agents coordinate their activities so as to cooperatively achieve goals?
- While these questions are all addressed in part by other disciplines (notably economics and social sciences), what makes the multi-agent systems field unique is that it emphasizes that agents in question are *computational, information processing* entities [Wooldridge]

Agents development

- Jennings said:
  - *“My guess is that agent-based computing will be what object-oriented programming was in the 1980s. Everybody will be in favour of it. Every manufacturer will promote his product as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And no one will know just what it is.”*
  - *“…agent system development is dominated by informal guidelines, heuristics and inspirations rather than formal principles and well-defined engineering techniques”*
- A more Software Engineering view:
  - Need of Agent-Oriented Software Engineering
    - Methodologies, modeling languages, systematic development processes, CASE tools, etc…

SE challenges

- Two different problems:
  - How do we build agents capable of independent, autonomous actions, so that they can successfully carry out goals and tasks we delegate to them?
  - How do we build agents that are capable of interacting (cooperating, coordinating, negotiating) with other agents in order to successfully carry out those delegated tasks, especially when the other agents cannot be assumed to share the same interests/goals?
- The first problem is *agent design*, the second is *society/organisational design*
  - micro vs. macro

OOSE vs. AOSE

- Object-Oriented Software Engineering:
  - engineering a computer program based on objects which are its building blocks and we use object-oriented methodologies during software analysis and design
- Agent-Oriented Software Engineering:
  - development of a computer program based on agents which are its building blocks and founded on concepts like knowledgeability, autonomy, proactiveness and interactivity. The agent concept is also used as foundation for an agent-oriented methodology used for analysis and design

Agent architectures

- Fundamental mechanism underlying the autonomous components that support behaviour in real-world, dynamic and open environments
  - Logic-based (symbolic)
  - Reactive
  - Layered (hybrid)
  - BDI (deliberative)
- Most used
  - Reactive -- hybrid -- deliberative

Logic-based architectures (AI)

- Intelligent behaviors can be generated by a symbolic representation of the environment and syntactically manipulating this representation
  - Logical formulae (symbolic representation)
  - Logical deduction or theorem proving (syntactic manipulation)
- A theory can explain how the agent should behave
  - how goals are generated so as to satisfy its delegated objectives,
  - how agent interleaves goal-directed and reactive behavior in order to achieve these goals
  - The theory (specification) is refined through a series of progressively more concrete stages until finally an implementation is reached

Example

```
Open(valve221)
Temperature(reactor4726,321)
Pressure(tank776,28)
```

Formulae are used to represent environment properties (info that agent has about the environment – beliefs not necessary this correspond to the truth, agents can be wrong in sensing the environment)

L a set of sentences in first-order logic, then the internal state of an agent is a subset of *L* (i.e., a set of formulae of first-order logic - Δ)

Decision-making process modeled as a set of deduction rules (ρ)

$\Delta \vdash_\rho \varphi$ (φ can be proved from Δ using only the deduction rules ρ)

Example

```
Open(v_221)
is_tank_val(t_7,v_221)
Temperature(reactor_1,321)
Pressure(t_7,28)
```

Internal state

Deduction rule

```
Pressure(T,X)^X>20 ^ is_tank_val(T,V) → Close(V)
```

```
Pressure(t_7,28) → Close(v_221)
```

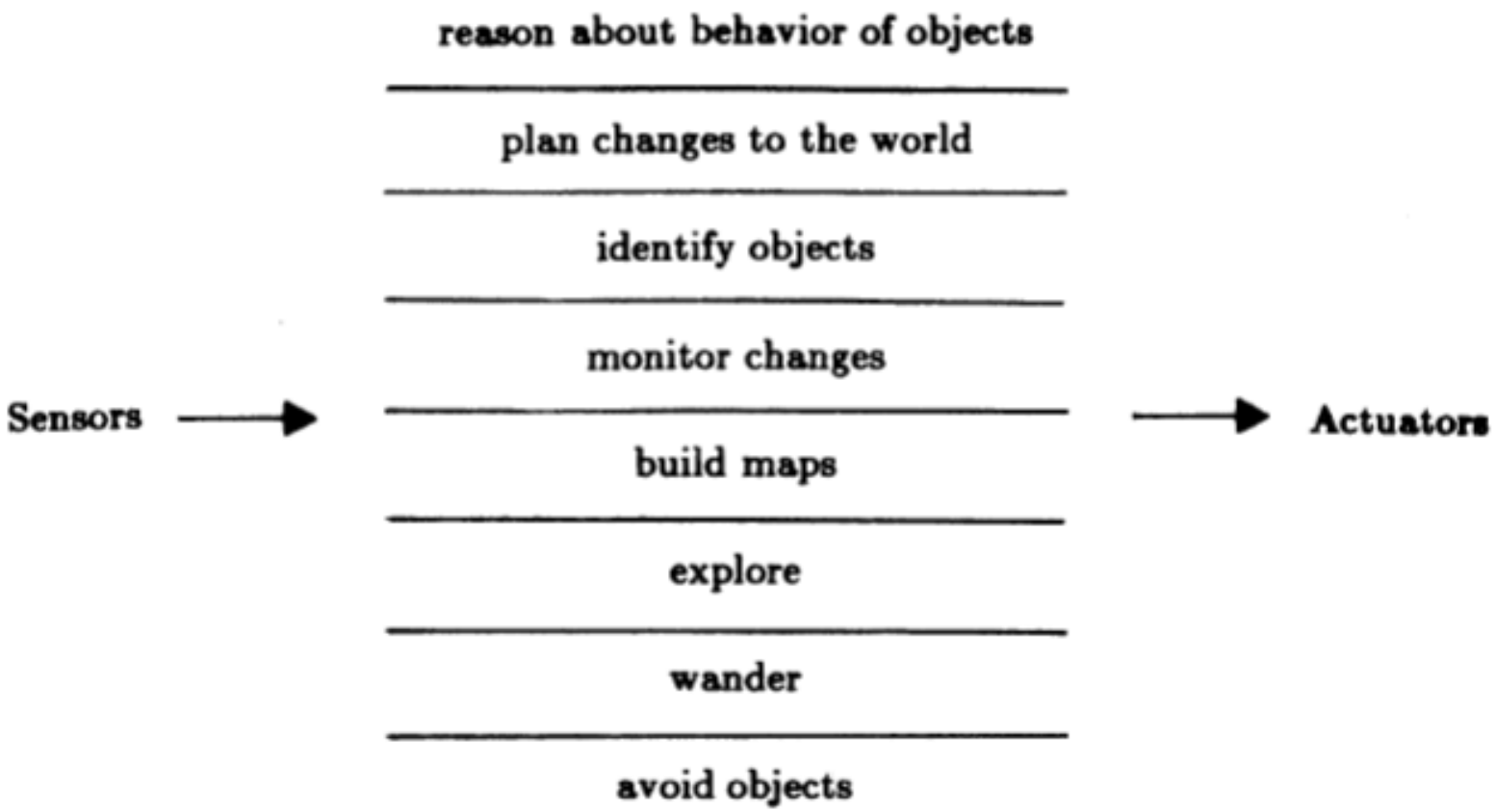
Action selection algorithm

function *action*(Δ,ρ) returns an action
begin
  for each $a \in Ac$ do
    if $\Delta \mid_{\rho} Do(a)$ then
      return a /*best action to perform */
    end-if
  end-for
  for each $a \in Ac$ do
    if $\Delta \not\mid_{\rho} \neg Do(a)$ then
      return a /* not explicitly forbidden action*/
    end-if
  end-for
return *null* /* not action found*/

Reactive Architecture - (Brooks, 1991)

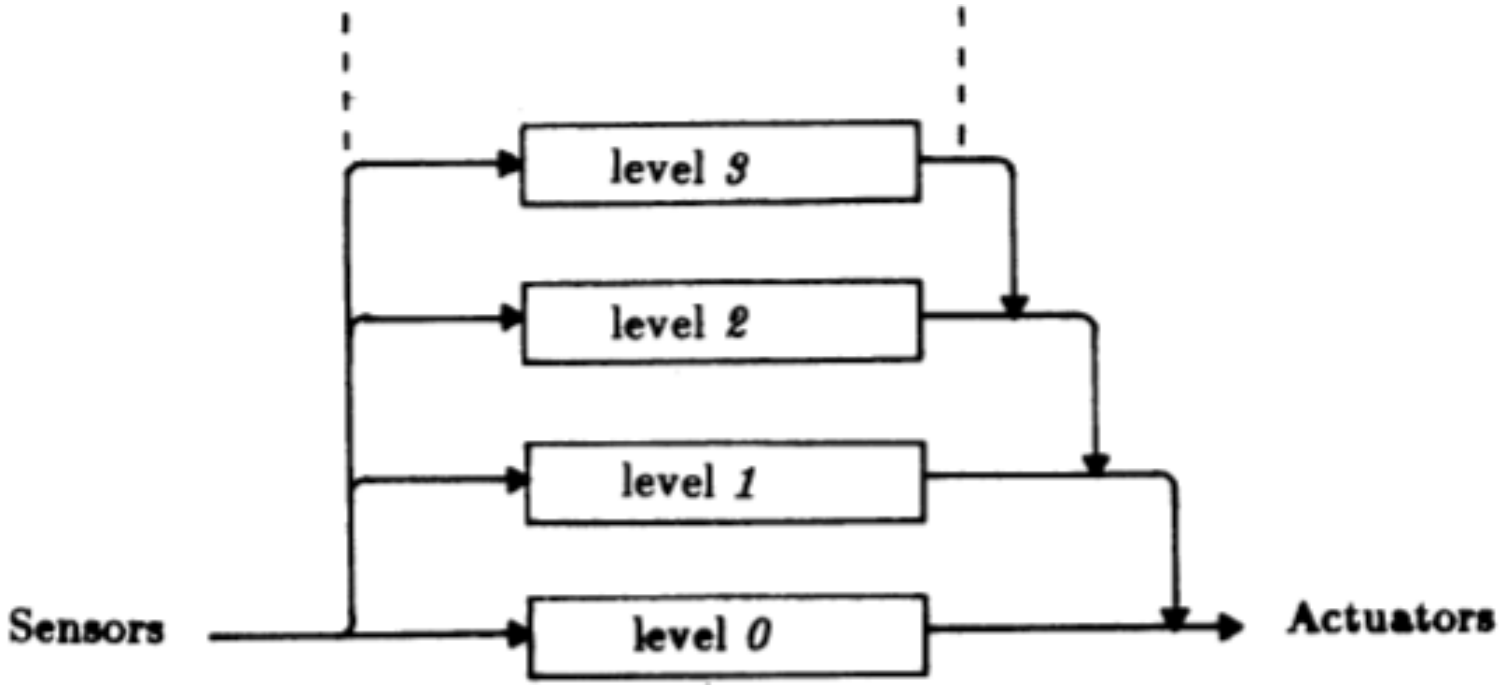
- Brooks has put forward three theses:
  - Intelligent behavior can be generated without explicit representations
  - Intelligent behavior can be generated without explicit abstract reasoning
  - Intelligence is an emergent property of certain complex systems
- Subsumption architecture
  - Hierarchy of task-accomplishing behaviors
  - Each behavior is a rather simple rule-like structure
  - Each behavior ‘competes’ with others to exercise control over the agent
  - Lower layers represent more primitive kinds of behavior (such as avoiding obstacles), and have precedence over layers further up the hierarchy
  - The resulting systems are extremely simple

Reactive Architecture (1)



From Brooks, "A Robust Layered Control System for a Mobile Robot", 1985

Reactive Architecture – subsumption



From Brooks, "A Robust Layered Control System for a Mobile Robot", 1985

Steels' Mars Explorer

- Steels' Mars explorer system, using the subsumption architecture, achieves near-optimal cooperative performance in simulated 'rock gathering on Mars' domain:
  - The objective is to explore a distant planet, and in particular, to collect sample of a precious rock. The location of the samples is not known in advance, but it is known that they tend to be clustered

Steels' Mars Explorer Rules

- The lowest-level behavior is obstacle avoidance:

  (1) **if detect an obstacle then change direction**

- Any samples carried by agents are dropped back at the mother-ship:

  (2) **if carrying samples and at the base then drop samples**

- Agents carrying samples will return to the mother-ship:

  (3) **if carrying samples and not at the base then travel up gradient**

- Agents will collect samples they find:

  (4) **if detect a sample then pick sample up**

- An agent with "nothing better to do" will explore randomly:

  (5) **if true then move randomly**

Advantages and Limitations of RAs

- Advantages
  - Simplicity, Economy, Computational tractability, Robustness against failure, Elegance
- Limitations
  - Agents without environment models must have sufficient information available from local environment
  - If decisions are based on local environment, how does it take into account non-local information (i.e., it has a "short-term" view)
  - Difficult to make reactive agents that learn
  - Since behavior emerges from component interactions plus environment, it is hard to see how to engineer specific agents (no principled methodology exists)
  - It is hard to engineer agents with large numbers of behaviors (dynamics of interactions become too complex to understand)

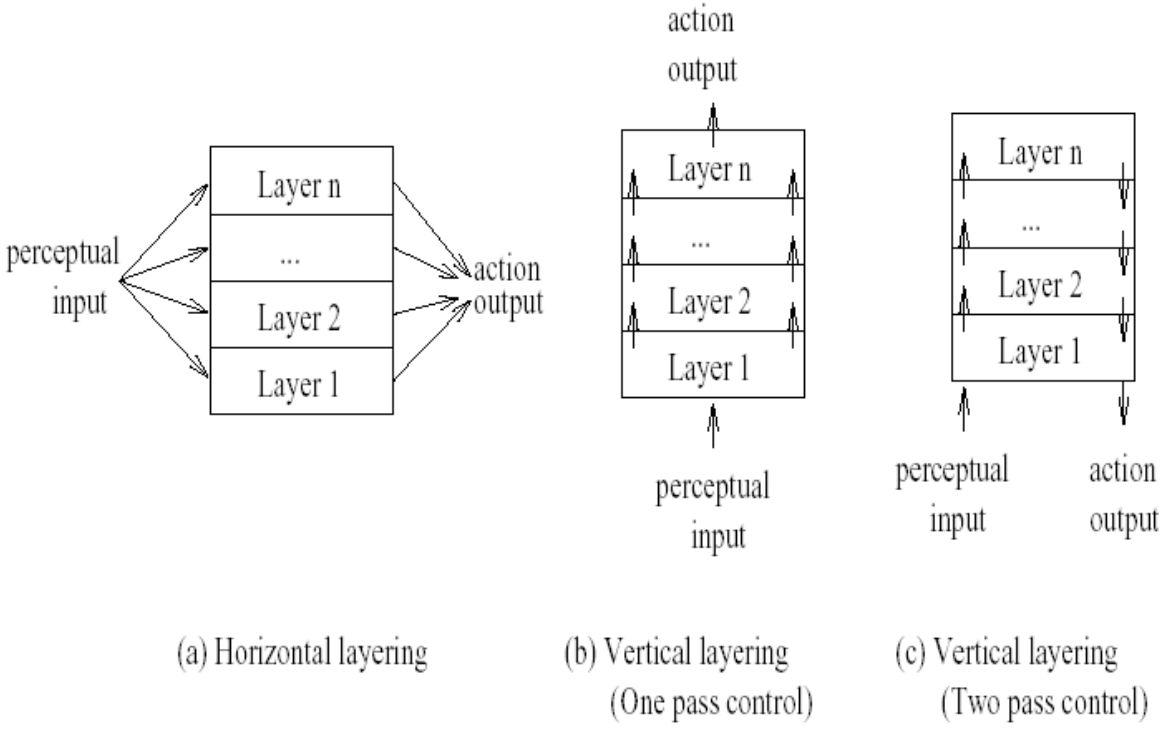
Hybrid Architectures

- An hybrid architecture is based on two (or more) parts:
  - deliberative, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI
  - reactive, which is capable of reacting to events without complex reasoning

    (often, the reactive component is given some kind of precedence over the deliberative one)
- This kind of structuring leads naturally to the idea of a layered architecture -- the agent's control subsystems are arranged into a hierarchy, with higher layers dealing with information at increasing levels of abstraction
  - Horizontal layering: each layer is directly connected to the sensory input and action output. Each layer itself acts like an agent, producing suggestions as to what action to perform.
  - Vertical layering: sensory input and action output are each dealt with by at most one layer each

Hybrid Architectures

m possible actions suggested by each layer, n layers



(a) Horizontal layering

(b) Vertical layering (One pass control)

(c) Vertical layering (Two pass control)

m^n interactions between layers

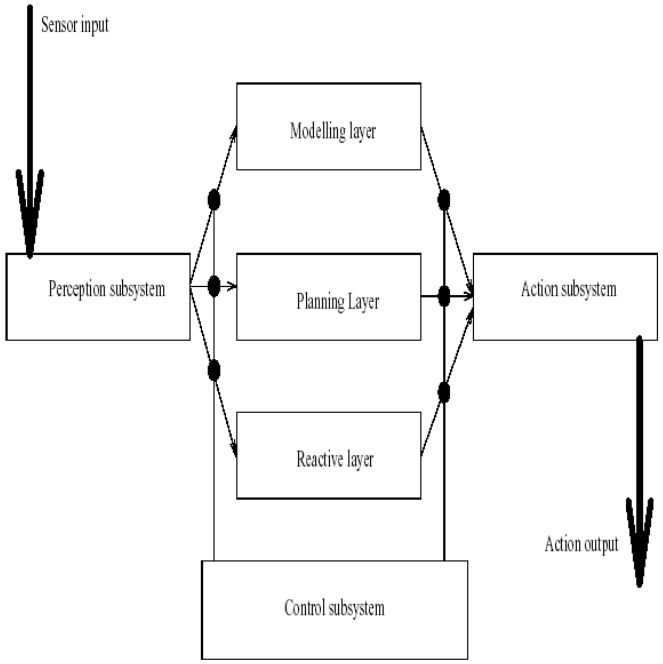
Introduces bottleneck in central control system

$m^2(n-1)$ interactions

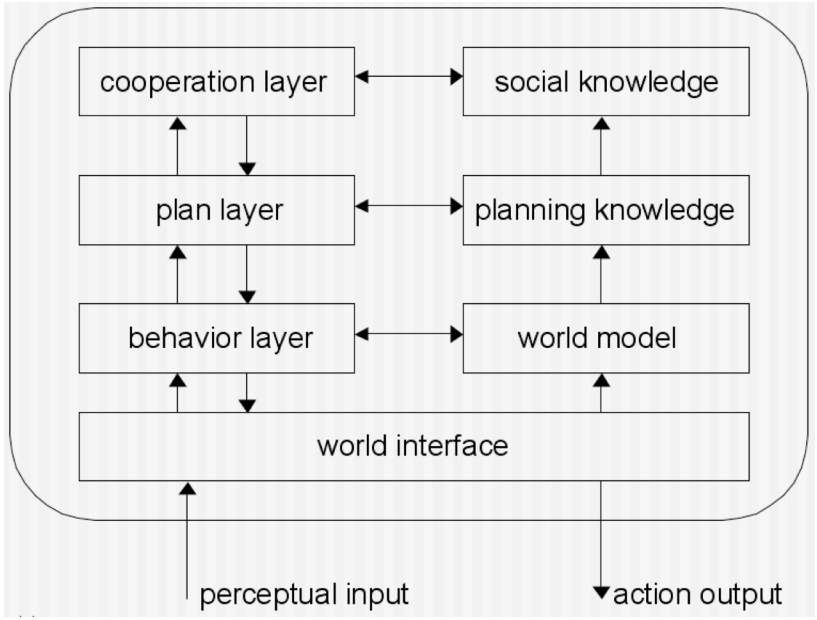
Not fault tolerant to layer failure

Two examples

Ferguson – TOURINGMACHINES



Müller –InteRRaP



Ferguson – TOURINGMACHINES

The *reactive layer* is implemented as a set of situation-action rules, *à la* subsumption architecture.

Example:

```
rule-1: kerb-avoidance
   if is-in-front(Kerb, Observer) and
speed(Observer) > 0 and
      separation(Kerb, Observer) < KerbThreshHold
   then change-orientation(KerbAvoidanceAngle)
```

The *planning layer* constructs plans and selects actions to execute in order to achieve the agent's goals

Ferguson – TOURINGMACHINES

- The *modelling layer* contains symbolic representations of the 'cognitive state' of other entities in the agent's environment.
- The three layers communicate with each other and are embedded in a control framework, which use *control rules*.
- Example:

```
censor-rule-1:
   if entity(obstacle-6) in perception-buffer
   then
      remove-sensory-record(layer-R,
entity(obstacle-6))
```

BDI architecture

- The most popular and used in agent community (Rao and Georgeff, 1995)
- Many implementations
  - PRS (1987), dMARS (1998), JAM (1999), Jack (2001), JADEX (2005)
- A model of practical reasoning (1980s) where:
  - Environment can evolve in many (unpredictable) directions
  - Many potential courses of actions available at any point
  - Many potential objectives at any point
  - Environment can only be sensed locally
  - Environment dynamic - implies resource bound in reasoning!
- Additional assumption: plans!
  - "Plan" used in the sense of a known recipe ("know-how") rather than a chosen course of action (i.e., not "here" is the plan")

BDI: Belief

- Information about the world, the past, …
- Why?
  - World is dynamic, so the agent needs to remember the past
  - Agent has local view so it needs to remember
  - Agent resource bounded, so store information rather than re-compute
- Since stored information is imperfect, semantics should be beliefs, not knowledge
  - Knowledge (in the context of logics of knowledge of belief) has the property that is agent knows P then P must be true
- Note: in addition to caching beliefs, also cache plans (recipes) for similar reasons

BDI: (Goal)Desire

- Desired end state (e.g., “you desire to graduate”)
- Captures why a particular piece of code is/was executing
  - … useful for recovering from failure (cf. execution cycle)
  - … (potentially) allows for reasoning about goal interaction(*)

- (*) not done in current systems but topic of current research.

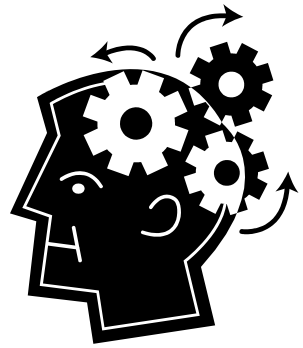
BDI: Intention

- Question: Are beliefs and desires enough?
- Answer: No, also need intentions
- Intention = selected course of action (plan instances)
- Why?
  - Resource bounded agents in dynamic environment need to plan bit-by-bit while acting
  - Intentions capture the partial plan (in the sense of “course of action”)
  - Intentions can also be used to coordinate agents
- Properties:
  - Intentions lead to action
  - Intentions are persistent by default
  - Intentions should be internally consistent
  - Intentions should be fleshed out by the time they need to be “executed”

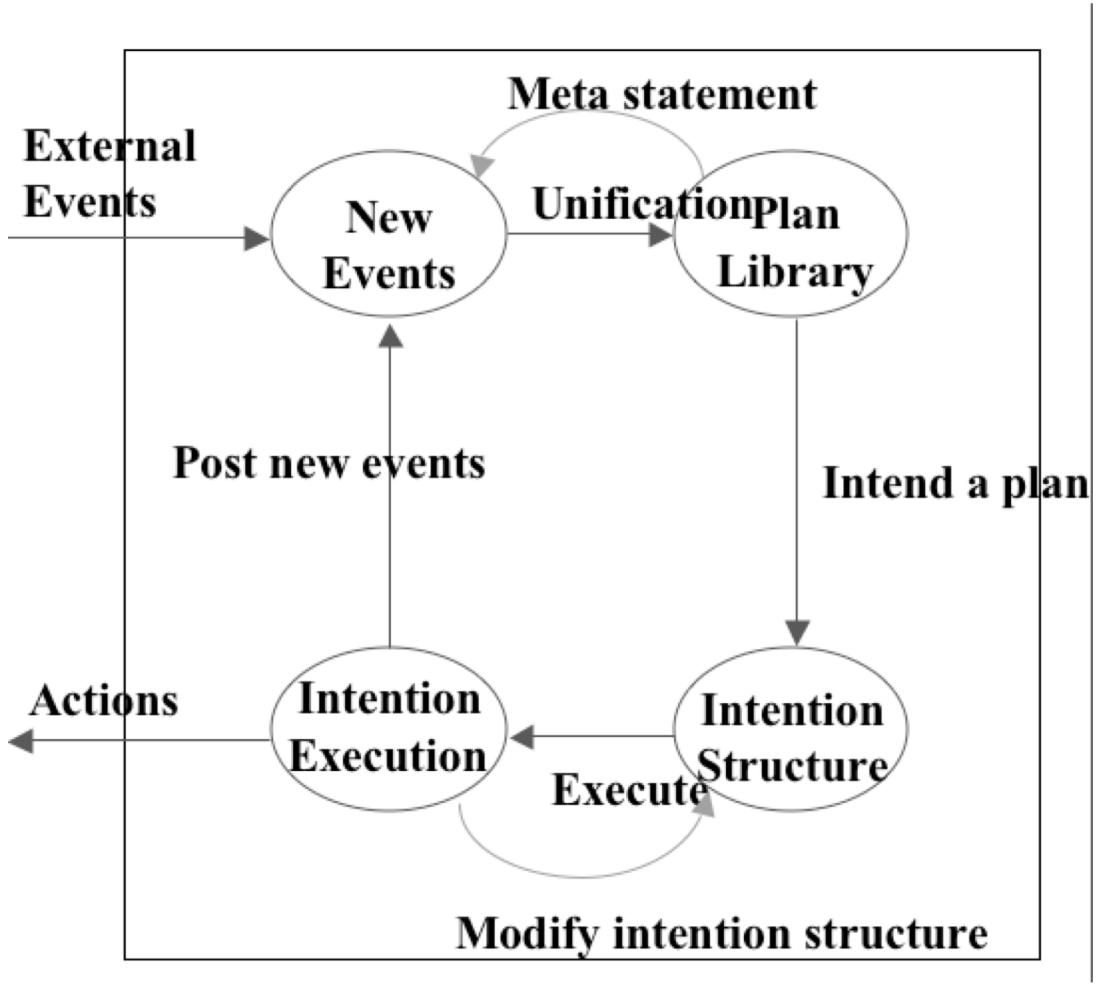
Commitment, re-planning, and options

- You are in the middle of following an intention, and the world changes, should you re-plan?
  - Classical decision theory: yes, always
  - Classical computing (no goals): no, never
- Neither is ideal - want mixture!
- A deliberating agent has many available options (too many to consider!), so want to reduce the options to consider
  - Don't consider options that conflict with selected intentions
  - Don't reconsider chosen options (unless there's a good reason)
- When doing further planning, assume that intentions will have been achieved
  - e.g., intend to return library book in the afternoon and go to movies in the evening, since library is at UNITN@Albere, plan to go to movies in the city

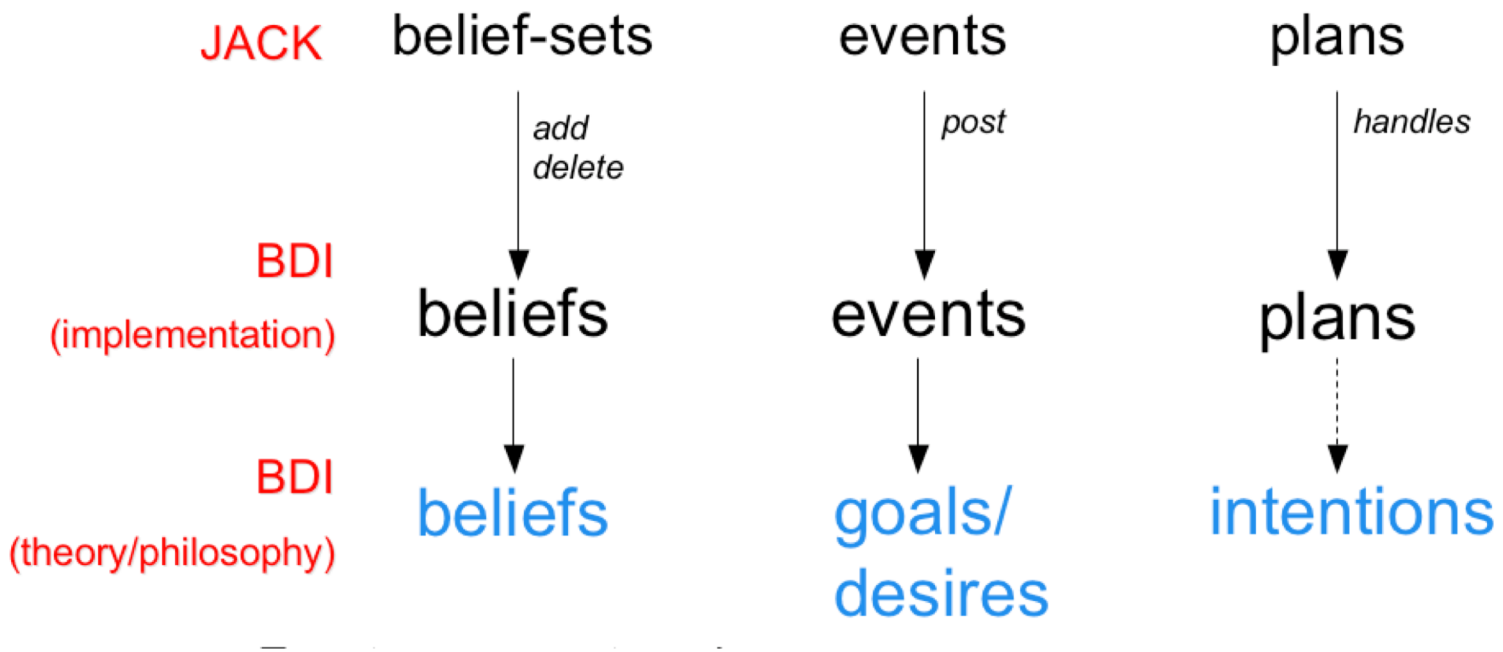
BDI: Implementation in Jack



- Architecture vs implementations
- **B**eliefs - local knowledge base (really database)
- **E**vents - used to model goals
- **P**lans - predetermined sequences of actions or sub-goals that can accomplish specified tasks
- **I**ntentions - currently running plans



BDI Theory and Jack implementation



MAS application areas

- Complex environments
  - Need of decomposing complex tasks and processes
  - Need of fully automatic and autonomous activities and processes
  - Safety-critical systems
  - Very dynamic and unpredictable domains
  - (Social) network
  - Distributed/concurrent systems
  - Human-computer interaction
- Some examples:
  - air traffic control (Sydney airport…sometime ago)
  - business process management
  - power systems management
  - distributed sensing
  - factory process control
  - hand-held devices with limited bandwidth
  - information gathering

MAS Opportunities (M. Luck et. al.- Roadmap)

1. Ambient Intelligence / Smart environments
2. Bioinformatics and Computational Biology
3. Grid Computing
4. Electronic Business
5. Simulation
   - Education and training
   - Scenario exploration
   - Entrainment

Ambient Intelligence / Smart environments

- An environment of potentially thousands of embedded and mobile devices (or software artefacts) interacting to support user-centred goals and activity.
  - autonomy
  - distribution
  - adaptation
  - responsiveness… as agents
- Artefacts need to interact with numerous other artefacts
  - pairs of artefacts (in one-to-one cooperation or competition),
  - groups of artefacts (in reaching consensus decisions),
  - artefacts and the infrastructure resources that comprise their environments
- Two particularly important points
  - scalability
  - heterogeneity of artefacts

Biological Science and Bioinformatics

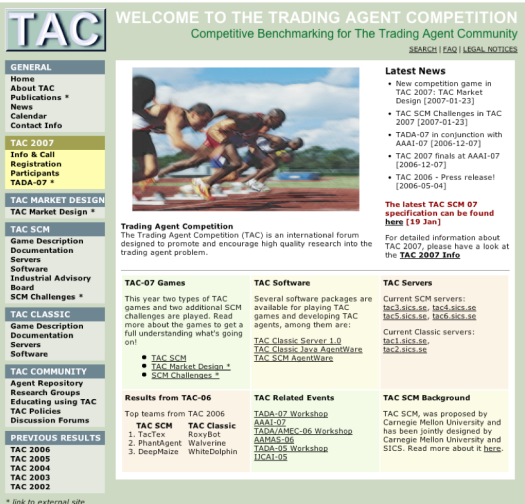
- Multi-agent systems for simulation and modeling of biological systems
  - similarly to the simulation of socio-economic and public policy domains
  - assessment of alternative assumptions and input parameters
  - development of a better understanding of the dynamics of complex interacting systems
- Bioinformatics:
  - information explosion caused by genomics and proteomics
  - great need for automated information-gathering and information-inference tools.
- Information-gathering agents may help in finding appropriate research literature or in conducting automated or semi-automated testing of data.
- Data mining agents may produce a set of potential hypotheses that can be induced from the data sources.
  - numerous databases and analysis tools independently administered in geographically distinct locations
- Some early work in this direction, using agents for genome analysis
  - GeneWeaver project in the UK
  - work using DECAF in the US

Grid computing

- High-performance computing infrastructure, known as *the Grid*, for supporting large-scale distributed scientific endeavor
- Grid computing is abstracted into several layers:
  - data-computation layer dealing with computational resource allocation, scheduling and execution;
  - information layer dealing with the representation, storage and access of information;
  - knowledge layer, which deals with the way knowledge is acquired, retrieved, published and maintained
- Agents are used to
  - engage interactions
  - Negotiate
  - make pro-active run-time decisions while responding to changing circumstances.
- Agents need to collaborate and to form coalitions of agents with different capabilities

Electronic Business

- Agents have been used in the first stages of eCommerce
  - product and merchant discovery and brokering
- Moving into real trading
  - negotiating deals and making purchases.
  - Examples: travel agencies and retailing
- The Trading Agents Competition (TAC) offered a sophisticated problem domain of multiple auctio for agents to compile travel packages for customers
  - Very interesting and promizing results
  - http://www.sics.se/tac



Simulation (1)

- Education and Training
  - defense simulations using multi-agent systems
    - enable military planners, strategic defense staff and even operational staff to gain experience of complex military operations through simulations and *war games*.
  - the decision-maker is allowed to learn through his or her mistakes, without these having real-world consequences.
- Scenario exploration
  - Traffic management (policies, measures, patterns),
  - water management,
  - privacy (legal principles as norms),
  - sustainable environment
  - logistics (dynamic planning and control),
  - health care (dynamic resource allocation),
  - education, physical systems (water motion, physical laws),
  - biological systems (genomics, neurology), economic models and business process automation.

Simulation (2)

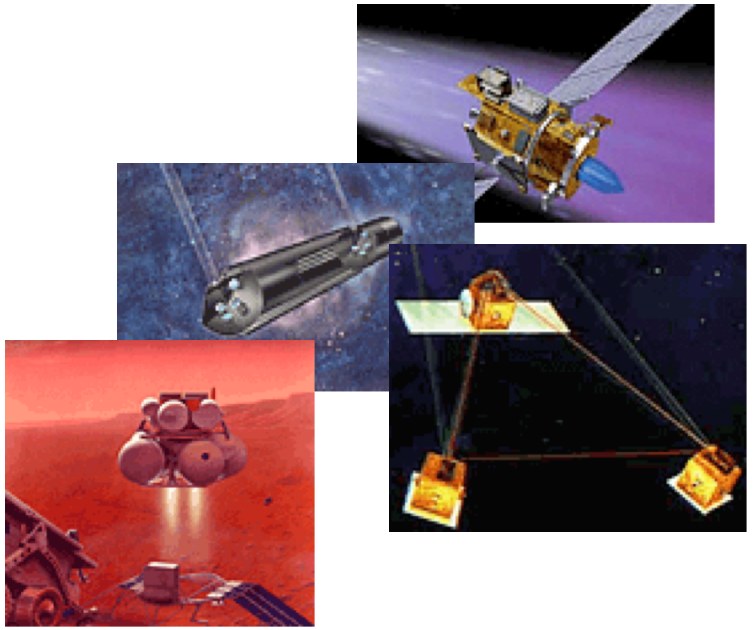
- Serious games
  - simulations of real-world events or processes designed for the purpose of solving a problem. Their main purpose is to train or educate users, though it may have other purposes.
- Entertainment
  - single-(human)-player computer games
  - multi-player games (humans or agents).
  - social simulation games, such as Maxis' *SimCity*
  - other interactive media,
    - interactive movies,
    - Interactive television
    - Interactive books

Simulating with PRESTO by Delta

iWD: Autonomous Response System by AOS

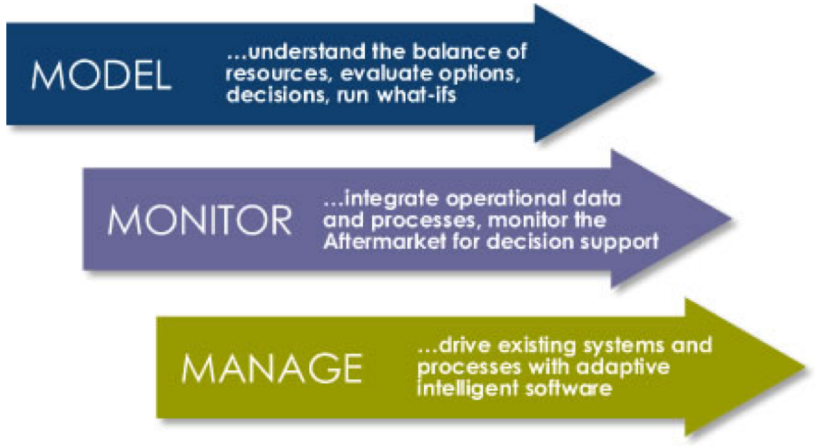
NASA: Remote Agent eXperiment

- In 1999, a software agent was given primary command of a spacecraft for the first time, as part of the Remote Agent eXperiment (RAX).The Remote Agent operated NASA's Deep Space 1 spacecraft during two experiments that started on Monday, May 17, 1999. For two days Remote Agent ran on Deep Space 1, more than 60,000,000 miles (96,500,000 kilometres) from Earth.
- Missions:
  - Deep Space One (DS1)
  - The Space Interferometry Mission (SIM)
  - Space Technology 3 (ST3)
  - In-situ Propellent (ISSP)



Aerogility

- Working with Rolls Royce, Lost Wax has developed Aerogility, a multi-agent system to help business managers better understand the complexities of the aerospace aftermarket.
  - Software agents represent the Aftermarket resources - people, assets and processes.
  - For each resource they capture their purpose, business goals and objectives.
  - The interactions between agents are determined by easily changed parameters covering overall strategies, management policies and organisation configurations, as well as business processes and rules.
- Aerogility is implemented in 3 phases:



Air Liquid America

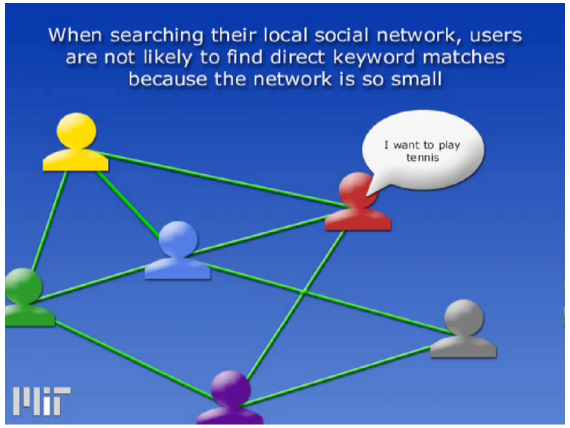
- Air Liquid America turned to agent technology to reduce production and distribution costs.
- Developed by NuTech Solutions, their system used a multi-agent ant system optimisation approach to discover efficient product distribution routes from the plant to the customer.
- As a result, Air Liquide managed to adapt production schedules to changing conditions and deliver products cost-effectively, where and when the customer demands, and in a manner that is responsive to unexpected events.

Some other experiences

- Agent Oriented Software has developed a system for the **UK Ministry of Defence** to model changes in human behaviour in military environments due to moderating influences, such as heat, tiredness, consumption of stimulants like caffeine, as well as battlefield experience and cultural factors. The objective was to offer a simulation environment for studying how people alter their decision making and interactions with other military personnel in these situations.
- An agent-based system developed by Acklin in The Netherlands for **international vehicle insurance claims** reduced human workload at one participating company by 3 people, and reduced the total time of identification of client and claim from an average of 6 months to under 2 minutes!
- Calico Jack has developed solutions tackling several key issues in primary **health care**. Among new services being offered as a result are the ability to co-ordinate repeat prescriptions using SMS (reducing load on the practice administrator, and simplifying the process for the patient), and to book appointments and handle reminders through a combination of SMS and email (with the aim of reducing the expense of wasteful missed appointments and smoothing the booking process for patients).
- Whitestein Technologies of Switzerland has developed an agent-based application designed to provide **automatic optimisation for large-scale transport companies**, taking into account the many constraints on their vehicle fleet, cargo, and drivers. Although the agent solution accounts for only 20% of the entire system, agent technology plays a central role in the optimisation.

An example from MIT
Real Time Searches on a Local Social Network

- "…*Cell phone contact lists represent a very large peer-to-peer network. We are creating a cell phone based application that allows users to perform real time searches on their local social network, against pieces of information that their contacts have provided about themselves. Our matchmaking agent uses Open Mind Common Sense to understand users' goals, and logically expand on their queries*".
- Play Demo



…from Trento
ANDIAMO Carpooling

- *"… based on the use of location and available car seats, ANDIAMO allows a substantial number of people to share car rides, using their cellular phones. ANDIAMO is an agent-based Carpooling system where an auction mechanism is used as a method of negotiation among autonomous and proactive agents"*
- Play Demo

… from CMU
NEO: Agent Crisis Response

- The NEO is a Multi-Agent System (MAS) demonstration of agent technology in a Non-combatant Evacuation Operation (NEO).
- NEO illustrates the interaction of agent teams in aiding human teams to cooperatively plan and execute a hypothetical evacuation of US civilians from a Middle Eastern city in an escalating terrorism crisis. In NEO, RETSINA and Open Agent Architecture (OAA) agent teams and systems cooperate with each other and their human counterparts to evaluate a crisis situation, form an evacuation plan, follow an evolving context, monitor activity, and dynamically re-plan. NEO demonstrates the interoperability and use of two disparate agent systems' teams for aiding humans (officers and Ambassador) to effectively monitor the scenario, retrieve and fuse information for immediate use, and to plan and re-plan an emergency evacuation.
- Video

… another one from CMU
Agent Storm

- Agent Storm is a RETSINA agent scenario where agents autonomously coordinate their team-oriented roles and actions while executing a mission in the ModSAF (Modular Semi-Automated Forces) simulation environment.
- The goal of Agent Storm is to increase the effectiveness of decision-making teams through the incorporation of agent technology in domains that are distributed, open and subject to time and other environmental contingencies.
- The impact of the project for decision-making teams is substantial and manifold. The project demonstrates that teams of agents can (1) reduce time for a team to arrive at a decision, (2) allow teams to consider a broader range of alternatives, (3) allow a team to flexibly manage contingencies by replanning, (4) reduce time for a team to form a shared situational model, (5) reduce individual and team errors, (6) support team cohesion and (7) increase overall team performance.

… one more from CMU
Joccasta

- In order to increase team decision making in the area of joint mission planning, we are incorporating intelligent software assistants into human teams.
- Software assistants can anticipate the information needs of their human team members, prepare and communicate task information, adapt to changes in situation and changes to the capabilities of other team members, and effectively support team member mobility.
- Video