LECTURE 2

SUMMARY

1. Agent paradigm

- there is no clear distinction in the literature between *agents* and *intelligent agents*
- **Agents** are a paradigm of
  - **SE**
    1. **software agents**
       - a new class of software that act on behalf of the user
         - find and filter information
         - negotiate for services
         - easily automate complex tasks
         - collaborate with other agents to solve complex problems
       - a powerful abstraction for **visualizing** and **structuring** complex software
       - procedures, functions, methods, objects are familiar **software abstractions** for software developers
         - software agents are a (relatively) new paradigm unfamiliar to many software developers
       - **central idea**: *delegation*
         - the owner (user) of an agent delegates a task to the agent and the agent autonomously performs the task on behalf of the user
         - the agent must be able to communicate with the user to receive its instructions and provide the user with the results of its activities
         - an agent must be able to monitor the state of its own execution environment and make the decisions necessary to carry out its delegated task
       - ***examples*** of software agents:
         - computer viruses (destructive agents)
         - artificial players in computer games
         - trading and negotiation agents
           - the auction agent at eBay
         - Web crawlers (spiders)
    2. **agent-based system**
       - a system in which the key abstraction is that of an agent
         - autonomous

        - reactive
        - pro-active
        - social
    - **AI/DAI**
        - artificial agent – a model of a human agent
        - **Intelligent** agents - **learning**
        - MAS are a paradigm of DAI

- the discipline of *(intelligent) agents* has emerged largely from research in AI
    - though, the only *intelligence* requirement we generally make for the agents is that they can make an acceptable decision about what action to perform next in their environment, in time for this decision to be useful.
    - other requirements for *intelligence* will be determined by the domain in which the agent is applied: not all agents will need to be capable of *learning*
- a controversial problem in the literature:
    - *Is the (intelligent) agents domain more related to AI, or to SE?*
    - In 1999, Oren Etzioni noted that "*intelligent agents* are 99% **computer science** and 1% **AI**"

    !!! The application and exploitation of agent technology can be viewed, primarily, as a *computer science* problem. Agents are software components that must be designed and implemented in much the same way that other software components are.

- all the key problems of SE (*specification*, *design*, *implementation*, *testing*) should be considered for **agent-based systems (ABS)**
- a major challenge of AI, and of CS, generally, is to assure a correct development (both as *specification* and *implementation*) of ABSs

2. Application areas

- *agents* are indicated for domains where *flexible autonomous action* is required
- *intelligent agents* are useful in domains where *learning* is required.
- MAS are indicated for domains where:
    - control, data, expertise are distributed (e.g. distributed health care, industrial products design, etc)
        - process controllers – autonomous reactive systems
    - centralised control is impossible or impractical
    - processing nodes have competing or conflicting viewpoints
- main application **areas**:
    - distributed/concurrent systems
    - networks
    - human-computer interfaces
- agents are often implemented as:
    - UNIX processes
    - processes in C
    - Java threads

3. Application domains

- **Distributed Systems**
  - the idea of an agent is seen as a natural metaphor, and a development of the idea of **concurrent object programming**.
  - **Example domains**:
    - air traffic control (Sydney airport), spacecraft control (NASA Deep Space 1)
    - business process management (BPM)
    - manufacturing systems, supply chain management
    - distributed electricity management
    - factory process control
    - electricity management
    - relevant problems from the DAI domain
      - *Distributed Sensor Network Establishment* (DSNE)
      - *Distributed Vehicle Monitoring* (DVM)
      - *Distributed Delivery* (DD)
  - **Internet agents (Web agents, Softbots)**
    - systematic search on Internet is difficult
    - searching the Internet for the answer to a specific query ⇒ **agents**
    - an **Internet agent** is a kind of '*secretary*'
      - act as 'proxy', hiding information that we are not interested in, and bringing to our attention information that is of interest
      - agent - **personal digital assistant** (PDA)
    - a **scenario**
      - upon logging into our computer, we are presented with a list of email messages, sorted into order of importance by our PDA
      - the PDA presents us articles that are very close to our own
      - communicates with other PDAs
  - **Agents for e-business**
    - e-commerce, e-banking, e-travel, e-gambling, etc
    - **e-commerce**
      - Simple examples *first-generation* e-commerce agents:
        - BargainFinder from Andersen
        - Jango from NETBOT (now EXCITE) – shopping agent
      - *Second-generation*: negotiation, brokering, … market systems
    - *customer profiling*
      - groups of customers sharing similar goals and characteristics
    - *recommender systems*
    - **e-banking**
      - agent - **personal financial assistant** (PFA)
- **Networks**
  - there is currently a lot of interest in **mobile agents**, that can move themselves around a network (e.g., the Internet) operating on a user's behalf
    - TELESCRIPT language developed by General Magic for *remote programming*
- **Human computer interaction (HCI)**
  - the use of agent in interfaces

    - **agent** – **personal assistant** (PA) of the user
        - agents sit 'over' applications, watching, learning, and eventually doing things without being told — taking the initiative
    - **intelligent (adaptive) interfaces**
    - Pioneering work at **MIT Media Lab** (Pattie Maes):
        - news reader, news filtering
        - web browsers
        - mail readers
        - meeting scheduling
    - MAXIMS (Pattie Maes)
        - email assistant: '*learns to prioritize, delete, forward, sort, and archive mail messages on behalf of a user …* '
        - MAXIMS works by 'looking over the shoulder' of a user, and learning about how they deal with e-mail
        - implemented in Common Lisp
        - uses *memory based reasoning* (a kind of CBR)
            - the new events (e.g. an e-mail arrives) are recorded in the form (*situation*, *action*)
            - new situation – determines the NNs from the set of stored situations

4. Agents vs. existing paradigms

- **Agents and procedures**
    - several similarities
        - procedure: *preconditions*, *postconditions*
        - the effects of the procedure may be viewed as its *goals* (what the developer tends to achieve)
        - conceptually, the environment is given by the *preconditions*
    - main differences: **reactivity, proactiveness**
        - **procedure**
            - if the environment does change (the preconditions underlying the procedure become false while the procedure is executing $\Rightarrow$ the behavior of the procedure may not be defined
        - **agent**
            - the agent is **reactive -** if the environment is dynamic, the agent reacts to events that occur in its environment that affects its goals
- **Agents and objects**
    - *Are agents just objects by another name*?
    - **Object**: computational entity
        - encapsulates some *state*
        - able to perform *actions* (methods) on this state
        - communicates via message passing
    - **Main differences**
        - **the degree to which agents and objects are autonomous**
            - characteristic of OOP: principle of *encapsulation*
                - objects have control over their own internal state
                - an object exhibits **autonomy** over its **state** (it has control over it)
                - it **does not** exhibit *autonomy* over its *behavior*

          - if an object *O* makes a method m available for another object *O*’ to invoke (e.g. it is public), *O* does not control when/ how many times m is invoked
          - in OOP, the decision lies with the object that invokes the method
      - in MAS, an agent has *autonomy* both over its *internal state* and its *behavior*
        - if an agent *j* asks another agent *i* to execute an action *a*, *i* will execute *a* only if it is in its best interests
        - in the agent case, the decision lies with the agent that received the request

	Autonomy over the internal state	**Autonomy over the behavior**
Agent	Yes	Yes
Object	Yes	No

    - **the notion of flexible autonomous behavior**
      - the standard OOP model has nothing to do with *flexibility*
      - **agents** are **smart** (capable of *flexible* behavior)
    - **agents are active**
      - a MAS is multi-threaded, each agent is assumed to have at least one thread of control in the system
      - in the standard OOP model there is a single thread of control in the system
        - concurrency in OOP: **active objects**
          - an active object has its own thread of control
          - AOs do not necessarily have the ability to exhibit **flexible** autonomous behavior
          - are generally autonomous
            - able to exhibit some behavior without being operated upon by another object

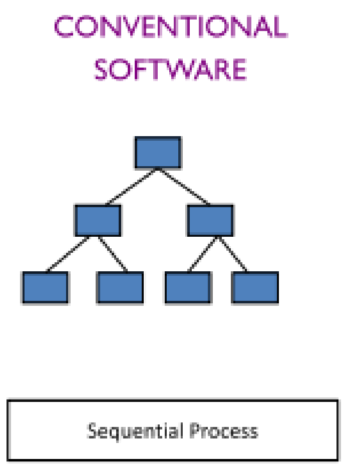
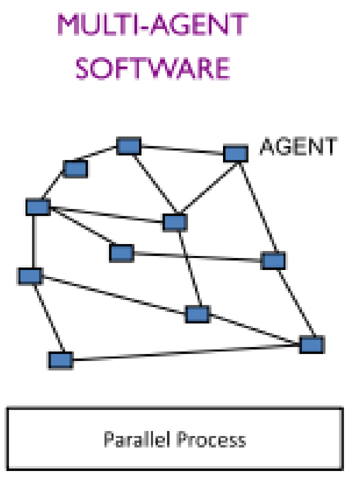
            ⇒ AOs have autonomy over their behavior

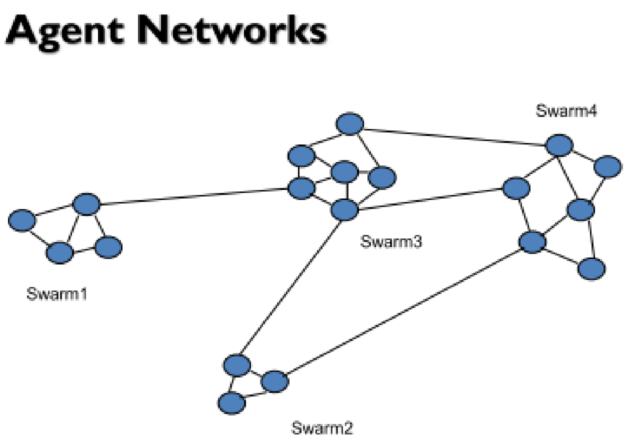
      ⇒ agents are *smart* active objects

- **Agents and Expert Systems (ES)**
  - *Are agents just expert systems by another name*?
  - **ESs**: Symbolic AI
    - rule based system
    - incorporate expert knowledge in a particular domain (e.g., blood diseases)
      - knowledge base
      - inference engine
    - various application domains: medicine, financial,
  - **MYCIN** knows about blood diseases in humans
    - it has a lot of knowledge about blood diseases, in the form of rules
    - a doctor can obtain expert advice about blood diseases by giving MYCIN facts, answering questions, and posing queries
  - **Main differences**:
    - agents situated in an environment
      - MYCIN is not aware of the environment

        - only information obtained is by asking the user questions
    - agents act, MYCIN does not act
    - MYCIN is not autonomous
        - usually needs the human intervention
- some real-time (typically process control) ESs are agents

5. Comparison of two software paradigms

CONVENTIONAL SOFTWARE	*MULTIAGENT SOFTWARE*
Hierarchies of programs/modules Centralized Data-driven Pre-programmed behavior	Large networks of small agents Distributed decisions Knowledge-driven Emergent behavior
	



6. Agent oriented software engineering (AOSE)

- **agents** are used to **model**, **design** and **build** *complex* software systems
    - e.g., agent based manufacturing control, supply chain management, BPM
    - make engineering of complex systems easier
    - tackling *complexity* through
        - **decomposition**
            - agents have their own persistent thread of control
            - agents control their own behavior
        - **abstraction**
            - abstraction for **visualizing** and **structuring** complex software
        - **organization**
            - MAS organizations / human organizations
- **FIPA** (Foundation of Intelligent Physical Agents)
    - standards for agent-based systems
- MAS researchers consider that designer have troubles when trying to capture MAS concepts using OO mechanisms
    - FIPA developed specific tools
        - AUML (Agent Unified Modeling Language)
        - design patterns for agents
        - methodologies for agent-based systems development
        - Agent Communication Language (ACL)

Complex system	Agent-based system
1. Subsystems	**1.** Agent organizations
2. Subsystems components	**2. Agents**
3. Interaction between subsystems and subsystem components	**3.** • **cooperating** to achieve common objectives • **coordinating** their actions • **negotiating** to resolve conflicts
4. Relationships between subsystems and subsystem components	**4.** Explicit mechanisms for representing and managing organizational relationships • association • conflict • subordination, etc

Bibliography

[1] Weiss, G. (Ed.): Multiagent Systems: *A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999 (available at www.cs.ubbcluj.ro/~gabis/weiss/Weiss.zip) [Ch. 1]

[2] Şerban Gabriela, *Sisteme Multiagent în Inteligenta Artificiala Distribuita. Arhitecturi si Aplicatii*, Editura RisoPrint, Cluj-Napoca, 2006

[3] Czibula Gabriela, *Sisteme inteligente. Instruire automată*, Editura RisoPrint, Cluj-Napoca, 2008