These slides are based on Wooldridge’s book. They can be used for free in any academic teaching scenario.
Five ongoing trends have marked the history of computing (Wooldridge, 09):

- ubiquity;
- interconnection;
- intelligence;
- delegation; and
- human-orientation
Five trends

Ubiquity

- The continual reduction in cost of computing capability has made it possible to introduce processing power into places and devices that would have once been uneconomic.
- As processing capability spreads, sophistication (and intelligence of a sort) becomes ubiquitous.

Interconnection

- Computer systems today no longer stand alone, but are networked into large distributed systems.
- Since distributed and concurrent systems have become the norm, some researchers are putting forward theoretical models that portray computing as primarily a process of interaction.
Five trends

Intelligence

- The complexity of tasks that we are capable of automating and delegating to computers has grown steadily to the limits than we can define as intelligent

Delegation

- Computers are doing more for us – without our intervention
- We are giving control to computers, even in safety critical tasks

Human orientation

- The movement away from machine-oriented views of programming toward concepts and metaphors that more closely reflect the way we ourselves understand the world
- Programmers conceptualize and implement software in terms of higher-level – more human-oriented – abstractions
Where does it bring us?

- *Delegation* and *Intelligence* imply the need to build computer systems that can act effectively on our behalf. This implies:
  - The ability of computer systems to act *independently*
  - The ability of computer systems to act in a way that *represents our best interests* while interacting with other humans or systems

- Interconnection and Distribution have become core motifs in Computer Science

- But *Interconnection* and *Distribution*, coupled with the need for systems to represent our best interests, implies:
  - systems that can *cooperate* and *reach agreements* (or even *compete*) with other systems that have different interests (much as we do with other people)
Computer Science progression

- These issues were not studied in Computer Science until recently

- All of these trends have led to the emergence of a new field in Computer Science: **multiagent systems**

- Programming has progressed through:
  - machine code;
  - assembly language;
  - machine-independent programming languages;
  - sub-routines;
  - procedures & functions;
  - abstract data types;
  - objects;
  to **agents**
Agents and Multiagent Systems, a 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).

- A *multiagent 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.
Agent technologies can be grouped into three categories:

- **Agent level** (micro)
  - Technologies and techniques concerned only with individual agents (e.g. procedures for agent reasoning and learning)

- **Interaction level**
  - Technologies and techniques concerned with the communications between agents (communication languages, interaction protocols, resource allocations mechanisms, etc.)

- **Organization level** (macro)
  - Technologies and techniques related to agent societies as a whole (structure, trust, norms, obligations, etc.)
Examples for Agent Scenarios I

**Space scenario:** Due to an unexpected system failure, an unmanned space ship near Saturn loses contact with its Earth-based ground crew and becomes disoriented. Rather than simply disappearing in the void, the probe detects aberration from its course, detects system-failure in flight control system, reboots flight control system, corrects flight trajectory.

**requires:**

- Autonomous behavior
- Appropriate sensors
- Failure detection
- Suitable set of actions
Remote Agent

Overview

1999 Co-Winner of NASA’s Software of the Year Award

It’s one small step in the history of space flight. But it was one giant leap for computer-kind, with a state of the art artificial intelligence system being given primary command of a spacecraft. Known as Remote Agent, the software operated NASA’s Deep Space 1 spacecraft and its fuel-efficient ion engine during two experiments that started on Monday, May 17, 1999. For two days Remote Agent ran on the on-board computer of Deep Space 1, more than 60 million miles (96.5 million kilometers) from Earth. The tests were a step toward robotic explorers of the 21st century that are less costly, more capable and more independent from ground control.

Features

Current Status
Notes on the progress of the experiment.

DS1 Mission
More information on DS1 can be found at their website.

Progress Display
A Java applet which shows the remote agent’s activities onboard the spacecraft.

DS1 Animation
A short computer graphics animation created for the news media.

A second remote agent experiment was conducted on Friday,
Air-traffic control scenario: Local air-traffic-control system (ATCS) suddenly fails, leaving flights in the vicinity of the airport with no air-traffic control support. Fortunately, autonomous air-traffic control systems of nearby airports detect failure, communicate and coordinate to assign each flight to the control of one of the ACTS.

requires:

- Taking the initiative when necessary
- Agents cooperating to solve problems beyond the capabilities of any individual agent

OASIS system, Sydney airport, Australia, 1992 (Ljunberg, Lucas)
Holiday scenario: Personal agent running on PDA is given the task of booking a holiday trip according to given preferences. PDA converses with several web sites which sell services such as flights, hotel rooms and hire cars. After hard negotiation on your behalf with a range of sites, it presents you with a package holiday.

requires:

- Formalism to specify (fuzzy) requirements / tasks / preferences
- Autonomous negotiation with providers with good balance between several conflicting resources / entities: e.g. negotiation time vs. optimality of result
- Ability to compare offers; matching offers with preferences
Multiagent Systems is Interdisciplinary

- The field of Multiagent Systems is influenced and inspired by many other fields:
  - Economics
  - Philosophy
  - Game Theory
  - Logic
  - Ecology
  - Social Sciences

- This can be both a strength (infusing well-founded methodologies into the field) and a weakness (there are many different views as to what the field is about)
Two Views of the Field

*Agents as a paradigm for software engineering:*

- Software engineers have derived a progressively better understanding of the characteristics of complexity in software. It is now widely recognized that *interaction* is probably the most important single characteristic of complex software.
- Over the last two decades, a major Computer Science research topic has been the development of tools and techniques to model, understand, and implement systems in which interaction is the norm.

*Agents as a tool for understanding societies:*

- Multiagent systems provide a novel new tool for simulating societies, which may help shed some light on various kinds of social processes.
- This has analogies with the interest in “theories of the mind” explored by some artificial intelligence researchers.

*EOS project, Palaeolithic culture at the last ice age, 1995, Doran, Palmer*
Objections to MAS I

MAS vs. Distributed/Concurrent Systems (DS/CS)

- MAS design should incorporate results from DS (dealing with shared resources, deadlock-avoidance, coordination over time, etc.)

- DS/CS: Cooperation and Coordination schemas hard-wired at design-time. MAS: Cooperation and Coordination is negotiated at run-time (i.e. more flexible)

- Agents in MAS: self-interested → „economic“ encounters. DS: components usually all work towards a common goal only
Objections to MAS II

MAS vs. Artificial Intelligence (AI)

- MAS ← [subfield-of] → AI?

- Al modern view (Russell, Norvig, 95): “AI: Research on building intelligent agents“ (planning, learning (classification, clustering, inductive logic programming, etc.), reasoning (expert systems, theorem provers, modal logics, temporal logics etc.), perceiving (OCR etc.), moving and acting in the real world (Robotics) etc.)

- Not all results from AI required to construct good agents. MAS research does not always require super-sophisticated agents.

- Until 1980s classical AI ignored social aspects of agency. These are important parts of intelligent activity in real-world settings.
Objections to MAS III

MAS vs. Game-Theory/Economics


- Game-theory describes properties of equilibria or optimal solutions etc. (e.g. Nash-eq.) but usually not the algorithms and their complexity required to build agents that enforce these solutions or equilibria.

- Some restrictive assumptions of game theory such as rationality may not apply to several MAS scenarios.

MAS vs. Social science

- MAS can draw insights from the study of human societies, but there is no particular reason to believe that artificial societies will be constructed in the same way.

- MAS take inspiration and cross-fertilization from social models, but hardly subsumption.