Agents & Multi-Agent Systems

Distributed Systems / Paradigms

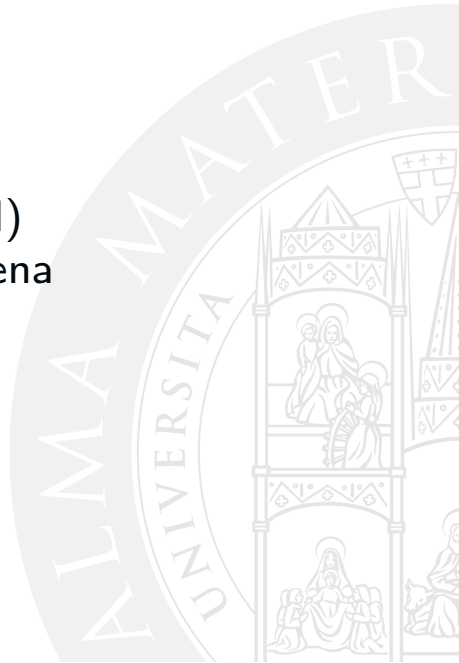
Sistemi Distribuiti / Paradigmi

Andrea Omicini

andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna a Cesena

Academic Year 2017/2018



Next in Line...

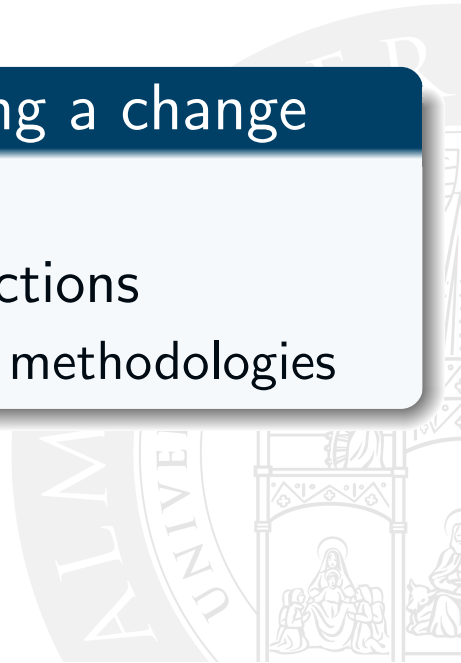
Focus on...

The Change is Widespread

- software systems (present and forthcoming ones) are essentially different from "traditional" ones
- the difference is widespread, and not limited to some application scenarios

Computer science & software engineering are undergoing a change

- dramatically
- complexity is too *huge* for traditional CS & SE abstractions
  - like object-oriented technologies, or component-based methodologies



The Next Crisis of Software

The scenario of the crisis

Computing systems

- are (going to be) anywere
  - are (going to be) embedded in every environment item/ object
- are (going to be) always connected
  - wireless technologies are making interconnection pervasive
- are (going to be) always active
  - to perform tasks on our behalf
  - more and more *autonomously*

Impact on Software Engineering

Which impact on the design & development of software systems?

quantitative

- in terms of computational units, software components, number & size of interconnections, people involved, time required, ...
- current processes, methods and technologies do *not scale* up

qualitative

- new software systems are *different* in kind
- *new features* never experimented before

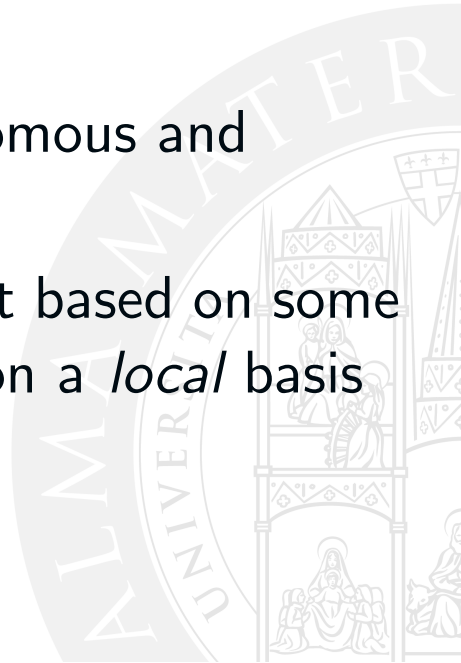
Novel Features of Complex Software Systems

situatedness
- computations occur as *immersed* within an environment
- computations and environment mutually affect each other, and cannot be understood separately

openness
- systems are permeable and subject to *change* in size and structure

locality in control
- components of a system are autonomous and proactive *loci* of control

locality in interaction
- components of a system interact based on some notion of spatio-temporal compresence on a *local* basis



This does not Hold for Agents & MAS only

Historically, fields like

- distributed artificial intelligence
- manufacturing and environmental control systems
- mobile computing
- pervasive / ubiquitous computing
- Internet computing
- peer-to-peer (P2P) systems

got the news early, and tried to face the issues in terms of new models, technologies, and methodologies

Situatedness: Examples

Control systems for physical domains

- manufacturing, traffic control, home care, health care systems
- explicitly aim at managing / capturing data from the environment through event-driven models / event-handling policies

Sensor networks, robot networks

- are typically meant to sense, explore, monitor and control partially known / unknown environments

Situatedness I

Situated action [Suchman, 1987]

- the notion of *situated action* stresses the relationship between an action and its context of performance
- actions are performed in a context: which affects the actions, and is affected by them
- the notion of *environment* is what is typically used here to denote the (computational) context

Environment as a first-class entity [Weyns et al., 2007]

- the notion of *environment* is explicit
- components / computations interact with, and are affected by the environment
- interaction with the environment is often explicit, too

Situatedness II

Is this new?

- every computation always occurred in some context
- however, the environment is *masked* behind some "wrapping" abstractions
- environment is not a *primary* abstraction

Situatedness III

Does masking / wrapping work?

- wrapping abstractions are often too simple to capture complexity of the environment
- when you need to sense / control the environment, masking it is not always a good choice
- environment dynamics is typically independent of system dynamics
  - the environment is often *unpredictable* and *non-formalisable* [Wegner, 1997]

Situatedness IV

Trend in CS and SE

- drawing a line around the system, explicitly representing
  - what is *inside* in terms of component's behaviour and interaction
  - what is *outside* in terms of environment, and system interaction with the environment
- *predictability* of components vs. *unpredictability* of the environment
  - this dichotomy is a key issue in the engineering of complex software systems

Openness: Examples

Critical control systems

- unstoppable systems, run forever
- they need to be adapted / updated anyway, in terms of either computational or physical components
- openness to change, and automatic reorganisation are essential features

Systems based on mobile devices

- the dynamics of mobile devices is out of the system / engineer's control
- system should work without assumptions on presence / activity of mobile devices
- the same holds for Internet-based / P2P systems

Openness

Permeable boundaries

- 'drawing lines' around systems does not make them isolated
- boundaries are often just conventional, and allow for mutual interaction and side-effects

The dynamics of change

- systems may change in structure, cardinality, organisation, ...
- technologies, methodologies, and (above all) abstractions should account for *modelling* (possibly *governing*) the dynamics of *change*

Openness: Further Issues

Where is the system?

- where do components *belong*?
- are system *boundaries* for real?

"Mummy, where am I"?

- how should components become *aware* of their environment...
- ... when they *enter* a system / are brought to existence?

How do we *control* open systems?

- ... where components come and go?
- ... where they can interact at their will?

Local Control: Examples

Cellular phone network

- each cell with its own activity / autonomous control flow
- autonomous (inter)acting in a world-wide network

World Wide Web

- each server with its own (reactive) independent control flow
- each browser client with its own (proactive) independent control flow

Local Control

Flow of control

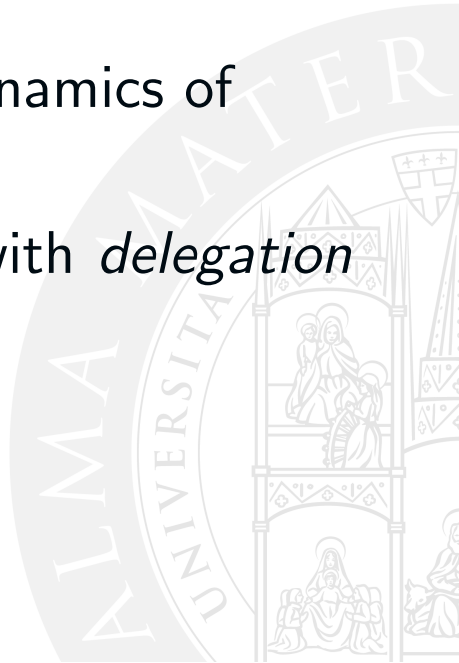
- key notion in traditional systems and in Computer Science
- *multiple flows* of control in concurrent / parallel computing
- however, not an immediate notion to deal with in complex software systems
  - a more general / abstract notion is required

Autonomy

- is the key notion here
- subsuming control flow
- motivating multiple, independent flows of control
- at a higher level of abstraction

Local Control: Issues of Autonomy

- autonomy of execution is an effective model for multiple independent computational entities, since it prevents the issues of *coupling*
- in an *open* world, autonomy of execution makes it easy for components to move across systems & environments
- autonomy of components more effectively matches dynamics of *environment*
- SE principles of locality and encapsulation cope well with *delegation of control* to autonomous components



Local Interactions: Examples

Control systems for physical domains

- each control component is delegated a portion of the environment to control
- interactions are typically limited to the neighbouring portions of the environment
- strict coordination with neighbouring components is typically enforced

Mobile applications

- local interaction of mobile devices is the basis for context-awareness
- interaction mostly occurs with the surrounding environment
- interoperation with neighbouring devices is typically enabled

Local Interactions

Local interactions in a global world

- autonomous components interact with the environment where they are located
  - interaction is limited in extension by either physical laws or logical constraints
- autonomous components interact openly with other systems
  - motion to and local interaction within the new system is the cheapest and most suitable model
- situatedness of autonomous components calls for context-awareness
  - a notion of locality is required to make context manageable

Summing Up

Complex software systems, then

- made of *autonomous* components
- *locally interacting* with each other
- *immersed* in an *environment*—both components and the system as a whole
- system / component *boundaries* are blurred—they are conceptual tools until they work

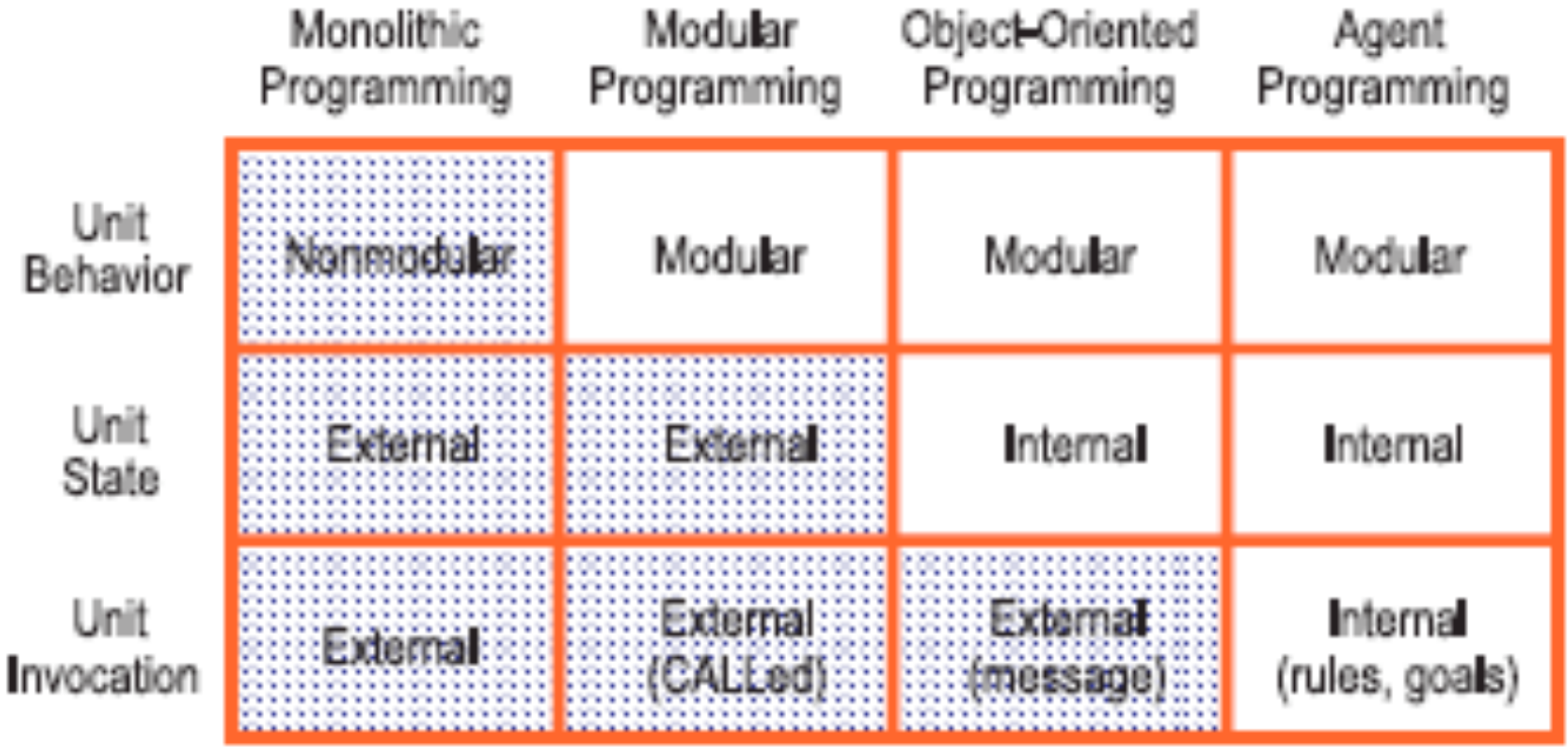
Change is ongoing

- computer science is changing
- software engineering is changing
- a (sort of) paradigm shift is occurring—a *revolution*, maybe

  [Zambonelli and Parunak, 2003]

Focus on...

Evolution of Programming Languages: The Picture



[Odell, 2002]

Evolution of Programming Languages: Dimensions

Historical evolution

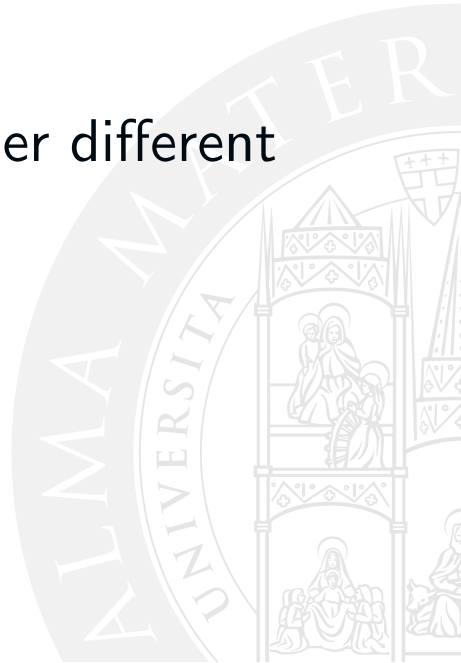
- monolithic programming
- modular programming
- object-oriented programming
- agent programming

Degree of modularity & encapsulation

- unit behaviour
- unit state
- unit invocation

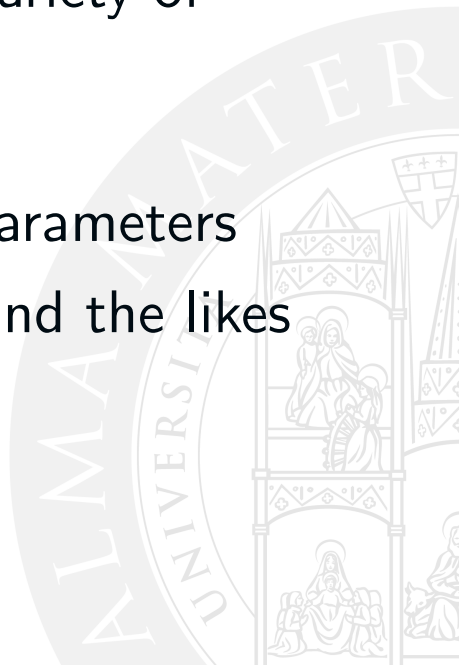
Monolithic Programming

- the basic unit of software is the whole program
- programmer has full control
- program's state is responsibility of the programmer
- program invocation determined by system's operator
- behaviour could not be invoked as a reusable unit under different circumstances
  - modularity does not apply to unit behaviour



Modular Programming

- the basic unit of software are structured loops / subroutines / *procedures* / ...
  - this is the era of procedures as the primary unit of decomposition
- small units of code could actually be *reused* under a variety of situations
  - modularity applies to subroutine's code
- program's *state* is *determined* by externally supplied parameters
- program invocation determined by CALL statements and the likes



Object-Oriented Programming

- the basic unit of software are *objects & classes*
- structured units of code could actually be *reused* under a variety of situations
- objects
  - have *local control* over variables manipulated by their own methods
    - variable state is *persistent* through subsequent invocations
    - object's state is *encapsulated*
  - are *passive*—methods are invoked by external entities
    - modularity does not apply to unit invocation
    - object's *control* is *not encapsulated*

Agent-Oriented Programming

- the basic unit of software are *agents*
  - encapsulating everything, in principle
    - by simply following the pattern of the evolution
  - whatever an agent is
    - we do not need to define them now, just to understand their desired features
- agents
  - could in principle be *reused* under a variety of situations
  - have *control* over their own *state*
  - are *active*
    - they cannot be invoked
    - agent's control is encapsulated

Features of Agents

Before we define agents, we already know that...

- ...agents are *autonomous* entities
  - encapsulating their thread of control
  - they can say "Go!"
- ...agents cannot be invoked
  - they can say "No!"
  - they do not have an interface, nor do they have methods
- ...agents need to encapsulate a *criterion* for their activity
  - to self-govern their own thread of control

Dimensions of Agent Autonomy

Dynamic autonomy

- agents are *dynamic* since they can exercise some degree of activity
  - they can say "Go!"
- from passive through reactive to active

Unpredictable / non-deterministic autonomy

- agents are *unpredictable* since they can exercise some degree of deliberation
  - they can say "Go!", they can say "No!"
  - and also because they are "opaque"—may be unpredictable to external observation, not necessarily to design
- from predictable to unpredictable through partially predictable

Objects vs. Agents: Interaction & Control

Message passing in object-oriented programming

- data flow along with control
  - data flow cannot be designed as separate from control flow
- a too-rigid constraint for complex distributed systems...

Message passing in agent-oriented programming

- data flow through agents, control does not
  - data flow can be designed independently of control
- with agents, complex distributed systems can be designed just based on the design of the information flow

Agents Communication

Agents communicate

- interaction between agents is a matter of exchanging information
  - toward agent communication languages (ACL) [Singh, 1998]
- agents can be involved in *conversations*
  - they can be involved in associations lasting longer than the single communication act
  - differently from objects, where one message just refer to one method

Philosophical Differences [Odell, 2002] I

Decentralisation

- object-based systems are completely pre-determined in control: control is essential centralised at design time
- agent-oriented systems are essentially decentralised in control

Emergence

- object-based systems are essentially predictable
- multi-agent systems are intrinsically unpredictable and non-formalisable and typically give raise to emergent phenomena

Philosophical Differences [Odell, 2002] II

Analogies from nature and society

- object-oriented systems have not an easy counterpart in nature
- multi-agent systems closely resembles existing natural and social systems

Towards the Coexistence of Agents and Objects

Final issues from [Odell, 2002]

- should we *wrap* objects to *agentify* them?
- could we really *extend objects* to make them agents?
- how are we going to *implement the paradigm shift*, under the heavy weight of legacy?
  - technologies, methodologies, tools, human knowledge, shared practises, ...

Next in Line...

Focus on...

Towards Seamless Agent Middleware

The first question

- how are we going to *implement the paradigm shift*, under the heavy weight of legacy?

Mainstreaming agent technologies [Omicini and Rimassa, 2004]

- observing the state of agent *technologies*
- focussing on agent *middleware*
- devising out a possible *scenario*

The Technology Life-Cycle

A successful technology from conception to abandonment

- first ideas from research
- premiere technology examples
- early adopters
- widespread adoption
- obsolescence
- dismissal

Often, however, this does not happen

- new technologies fail without even being tried for real
- which are the factors determining whether a technology will either succeed or fail?

Dimensions of a Technology Shift

Technology scenario has at least three dimensions

programming paradigm $\rightarrow$ new technologies change the way in which systems are conceived

development process $\rightarrow$ new technologies change the way in which systems are developed

economical environment $\rightarrow$ new technologies change market equilibrium, and their success is affected by market situations

3-D space for a success / failure story

- what is going to determine the success / failure of agent-based technologies?

The Programming Paradigm Dimension I

Pushing the paradigm shift

- evangelists gain space on media
- technological geeks follow soon
- drawbacks
  - too much hype may create unsupported expectations
  - perceived incompatibility with existing approaches
  - possible dangers for conceptual integrity

The Programming Paradigm Dimension II

Middleware for the paradigm shift

- technology support to avoid unsupported claims
- seamlessly situated agents vs. wrapper agents
  - communication actions towards agents
  - pragmatical actions towards objects
- this allows agents to be used in conjunction with sub-systems adopting different component models

The Development Process Dimension

Accounting for real-world software development

- availability of development methods & tools is critical
  - no technology is to be widely adopted without a suitable methodological support
- day-by-day developer's needs should be accounted, too

Agent-Oriented Software Engineering (AOSE) methodologies

- adopting agent-based metaphors and abstractions to formulate new practises in software engineering
- current state of AOSE methodologies [Cossentino et al., 2014]
  - early development phases are typically well-studied
  - later phases are not, neither the tools, nor the fine-print details

The Economical Environment Dimension I

Innovation has to be handled with care

- stakeholders of new technologies may enjoy advantages of early positioning
- however, they often focus too much on *novelty* and *product*, rather than on *benefits* and *service*
  - "we are different" alone does not help much
  - software is a quite peculiar product: nearly zero marginal cost, and almost infinite production capability

The Economical Environment Dimension II

Agent-oriented middleware & infrastructures

- promoting agent-oriented technologies through integration with existing object-oriented middleware & infrastructures
- creating a no-cost space for agent technologies
  - where (agent) technologies are no longer "sold" as whole packages
  - whose choice do not require any design commitment
  - where however agents represent the most effective choice for most components
- allow agent metaphors to add their value to existing systems with no assumption on the component model
- e.g. agent-based simulation frameworks, agent architectures for the integration of AI techniques, ACL-based messaging services, ...

Focus on...

Convergence Towards The Agent

Many areas contribute their own notion of agent

- artificial intelligence (AI)
- distributed artificial intelligence (DAI)
- parallel & distributed systems (P&D)
- mobile computing
- programming languages and paradigms (PL)
- software engineering (SE)
- robotics

On the Notion of Intelligence in AI

Reproducing intelligence

- AI is first of all concerned with reproducing *intelligent processes* and *behaviours*, where
  - intelligent processes roughly denote *internal* intelligence—like understanding, reasoning, representing knowledge, ...
  - intelligent behaviours roughly represent *external*, observable intelligence—like sensing, acting, communicating, ...

Symbolic intelligence

- classic AI promoted the so-called *symbolic* acceptation of (artificial) intelligence
  - based on *mental representation* of the external environment
  - where the environment is typically oversimplified
  - and the agent is the only source of disruption

On the Notion of Agent in AI

Encapsulating intelligence

- agents in AI have from the very beginning worked as the units encapsulating *intelligence*
  - *individual* intelligence
  - within the symbolic interpretation of intelligence

Cognitive agents

- AI agents are essentially cognitive agents
  - they are *first* cognitive entities
  - *then* active entities
  - in spite of their very name, coming from Latin *agens* [*agere*]—the one who acts

On the Notion of Agent in DAI [Wooldridge, 2002]

Overcoming the individual dimension

- no more a single unit encapsulating individual intelligence
- and acting *alone* within an oversimplified environment

Social acceptation of agency

- agents are individuals within a society of agents
  - agents are components of a *multi-agent system* (MAS)
- agents are distributed within a *distributed environment*

Agent Features in DAI [O'Hare and Jennings, 1996]

A DAI agent...

- ...has an explicit representation of the world
- ...is situated within its environment
- ...solves a problem that requires intelligence
- ...deliberates / plans its course of actions
- ...is flexible
- ...is adaptable
- ...learns

A DAI Agent Represents the World: What?

What should be represented?

- what is relevant, what is not relevant?
  - more precisely, which knowledge about the environment is relevant for an agent to effectively plan and act?
  - so, which portion of the environment should the agent explicitly represent somehow in order to have the chance to behave intelligently?

Representation is partial

- necessarily, an agent has a *partial* representation of the world
- its representation includes in general both the current *state* of the environment, and the *laws* regulating its dynamics

A DAI Agent Represents the World: How?

The issue of Knowledge Representation (KR)

- how should an agent represent knowledge about the world?
- representation is not neutral with respect to the agent's model and behaviour
  - and to the engineer's possibilities as well
- choosing the right KR language / formalism
  - according to the agent's (conceptual & computational) model
  - multisets of tuples, logic theories, description logics, ...?

A DAI Agent Represents the World: Consistency I

Perception vs. representation

- environment changes, either by agent actions, or by its own dynamics
- even supposing that an agent has the potential to observe all the relevant changes in the environment, it can not spend all of its activity monitoring the environment and updating its internal representation of the world
- so, in general, how could consistency of internal representation be maintained, and to what extent?
  - in other terms, how and to what extent can an agent be ensured that its knowledge about the environment is at any time consistent with its actual state

A DAI Agent Represents the World: Consistency II

Reactivity vs. proactivity

- an agent should be *reactive*, sensing environment changes and behaving accordingly
- an agent should be *proactive*, deliberating upon its own course of actions based on its mental representation of the world
- so, more generally, how should the duality between reactivity and proactivity be ruled / balanced?

A DAI Agent Solves Problems I

An agent has inferential capabilities

- new data representing a new solution to a given problem
- new knowledge inferred from old data
- new methods to solve a given problem
- new laws describing a portion of the world

An agent can change the world

- an agent is equipped with actuators that provide it with the ability to affect its environment
- the nature of actuators depends on the nature of the environment in which the agent is immersed / situated
- in any case, agent's ability to change the world is indeed limited

A DAI Agent Deliberates & Plans I

An agent has a goal to pursue

- a goal, typically, as a state of the world to be reached—something to achieve
- a task, sometimes, as an activity to be brought to an end—something to do

A DAI Agent Deliberates & Plans II

An agent understands its own capabilities

- its capabilities in terms of actions, pre-conditions on actions, effects of actions
- "understands" roughly means that its admissible actions and related notions are somehow represented inside an agent, and there suitably interpreted and handled by the agent
- perception should in some way interleave with action either to check action pre-conditions, or to verify action effects

A DAI Agent Deliberates & Plans III

An agent is able to build a plan of its actions

- it builds possible plans of action according to its goal/task, and to its knowledge of the environment
- it deliberates on the actual course of action to follow, then acts consequently

A DAI Agent is Flexible & Adaptable

Define *flexible*. Define *adaptable*.

- what do these words *exactly* mean?
- adaptable / flexible with respect to what?
- can an agent change its goal dynamically?
- or, can it solve different problems in different contexts, or in dynamics contexts?
- can an agent change its strategy dynamically?
- these properties are both important and potentially misleading, since they are apparently intuitive, and everybody thinks he/she understands them exactly

A DAI Agent Learns I

What is (*not*) learning?

- learning is not merely agent's change of state
- learning is not merely dynamic perception—even though this change the agent's state and knowledge

What could an agent learn?

- new knowledge
- new laws of the world
- new inferential rules?
  - new ways to learn?

A DAI Agent Learns II

A number of areas insisting on this topic

Machine learning, abductive / inductive reasoning, data mining, neural networks, ...

DAI Agents: Summing Up

In the overall, a DAI agent has a number of important features

- it has a (partial) representation of the world (state & laws)
- it has a limited but dynamic perception of the world
- it has inferential capabilities
- it has a limited but well-known ability to change the world
- it has a goal to pursue (or, a task to do)
- it is able to plan its course of actions, and to deliberate on what to do actually
- once understood what this means, it might also be flexible and adaptable
- it learns, regardless of how this term is understood

A PL Agent is Autonomous in Control

Complexity is in the control flow

- the need is to abstract away from control
- an agent encapsulates control flow
- an agent is an independent *locus* of control
- an agent is never invoked—it merely follows / drives its own control flow
- an agent is *autonomous* in control
  - it is never invoked—it cannot be invoked

A PL Agent is neither a Program, nor an Object

An agent is not merely a program

- a program represents the *only* flow of control
- an agent represents a single flow of control within a multiplicity

An agent is not merely a "grown-up" object

- an object is invoked, and simply responds
- an agent is never invoked, and can deliberate whether to respond or not to any stimulus

A P&D Agent is mobile [Fuggetta et al., 1998]

An agent is not bound to the Virtual Machine where it is born

- reversing the perspective
  - it is not that agents are mobile
  - it is that objects are not
- *mobility* is then another dimension of computing, just uncovered by agents

A new dimension requires new abstractions

- new models, technologies, methodologies
- to be used for reliability, limitations in bandwidth, fault-tolerance, ...

A Robotic Agent is Physical & Situated

A robot is a physical agent

- it has both a computational and a physical nature
  - complexity of physical world enters the agent boundaries, and cannot be confined within the environment

A robot is intrinsically situated

- its intelligent behaviour cannot be considered as such separately from the environment where the robot lives and acts
- some intelligent behaviour can be achieved even without any symbolic representation of the world
  - non-symbolic approach to intelligence, or *situated action* approach [Brooks, 1991]
- *reactive architectures* come from here

A SE Agent is an Abstraction

An agent is an abstraction for engineering systems

- it encapsulate complexity in terms of
  - information / knowledge
  - control
  - goal / task
  - intelligence
  - mobility
- agent-oriented software engineering (AOSE)
  - engineering computational systems using agents
  - agent-based methodologies & tools

A MAS Agent is a Melting Pot I

Putting everything together

- the area of multi-agent systems (MAS) draws from the results of the many different areas contributing a coherent agent notion
- the MAS area is today an independent research field & scientific community
- as obvious, MAS emphasise the *multiplicity* of the agents composing a system

A MAS Agent is a Melting Pot II

Summing up

- a MAS agent is an autonomous entity pursuing its goal / task by interacting with other agents as well as with its surrounding environment
- its main features are
  - autonomy / proactivity
  - interactivity / reactivity / situatedness

A MAS Agent is Autonomous I

A MAS agent is goal / task-oriented

- it encapsulates control
- control is finalised to task / goal achievement

A MAS agent pursues its goal / task...

- ...proactively
- ...not in response to an external stimulus

A MAS Agent is Autonomous II

So, what is new here?

- agents are goal / task oriented...
- ...but also MAS as wholes are
- *individual* vs. *global* goal / task
  - how to make them coexist fruitfully, without clashes?

A MAS Agent is Interactive I

Limited perception, limited capabilities

- it depends on other agents and external resources for the achievement of its goal / task
- it needs to interact with other agents and with the environment [Agre, 1995]
  - communication actions & pragmatical actions

A MAS agent lives not in isolation

- it lives within an agent *society*
- it lives immersed within an agent *environment*

A MAS Agent is Interactive II

Key-abstractions for MAS

- agents
- society
- environment

Summing Up I

The notion of agent is multi-faceted

- many reliable scientific sources
- many more or less convergent / divergent definitions
- a synthesis is currently ongoing in the MAS community

Finally, defining the agent notion

- it is now possible...
- ...but it is also insufficient, now
- to fully define MAS

Summing Up II

Meta-model is incomplete

- what about agent society?
- what about agent environment?

Next in Line...

Autonomy as the Core of Agents

Lex Parsimoniae: Autonomy

! *autonomy* as the essential feature of agents

? which other typical agent features may follow / descend from this—somehow?

Computational Autonomy

- agents are autonomous as they *encapsulate* (the thread of) *control*
- control does not cross agent boundaries
  - only data (knowledge, information) do
- agents have no interface, cannot be controlled / invoked
- looking at agents, MAS can be conceived as an aggregation of multiple distinct *loci* of control interacting with each other by exchanging information

(Autonomous) Agents (Pro-)Act

Action as the essence of agency

- the etimology of the word *agent* is from the Latin agens
- so, agent means "the one who acts"
- any coherent notion of agency should naturally come equipped with a model for agent actions

Autonomous agents are pro-active

- agents are literally *active*
- autonomous agents encapsulate control, and the rule to govern it

→ autonomous agents are *pro-active* by definition

- where *proactiveness* means "making something happen", rather than waiting for something to happen

Agents are Situated

The model of action depends on the context

- any "ground" model of action is strictly coupled with the context where the action takes place
- an agent comes with its own model of action
- any agent is then strictly coupled with the environment where it lives and (inter)acts
- agents are in this sense are intrinsically *situated* [Suchman, 1987]

Agents are Reactive

Situatedness and reactivity come hand in hand

- any model of action is strictly coupled with the context where the action takes place
- any action model requires an adequate representation of the world
- any effective representation of the world requires a suitable balance between *environment perception* and *representation*

→ Any *effective* action model requires a suitable *balance* between environment perception and representation

  - however, any non-trivial action model requires some form of perception of the environment—so as to check action pre-conditions, or to verify the effects of actions on the environment

- in this sense, agents are supposedly *reactive* to change

Are Autonomous Agents Reactive?

Reactivity as a (deliberate) reduction of proactivity

- an autonomous agent could be built / choose to merely react to external events
- it may just wait for something to happen, either as a permanent attitude, or as a temporary opportunistic choice
- in this sense, autonomous agents may also be reactive

Reactivity to change

- reactivity to (environment) change is a different notion
- this mainly comes from early AI failures, and from robotics

  [Brooks, 1986, Brooks, 1991]
- It stems from agency, rather than from autonomy—as discussed above

(Autonomous) Agents Change the World

Action, change & environment

- any model for action brings about a notion of *change*
  - an agent acts in order to change something in the MAS
- two admissible targets for change by agent action: an agent could act in order to change...

  (agent) ... the state of another agent
  - since agents are autonomous, and only data flow among them, the only way another agent can change their state is by providing them with some information
  - change to other agents essentially involves *communication actions*

  (environment) ... the state of the environment
  - change to the environment requires *pragmatical actions*
  - which could be either physical or virtual depending on the nature of the environment

Autonomous Agents are Social

From autonomy to society

- from a philosophical viewpoint, autonomy only makes sense when an individual is immersed in a society
  - autonomy does not make sense for an individual in isolation
  - no individual alone could be properly said to be autonomous
- this also straightforwardly explain why any program in any sequential programming language is not an autonomous agent *per se*

  [Graesser, 1996, Odell, 2002]

Autonomous agents live in a MAS

- single-agent systems do not exist in principle
- autonomous agents live and interact within agent societies & MAS
- roughly speaking, MAS are the only "legitimate containers" of autonomous agents

Autonomous Agents are Interactive

Interactivity follows, too

- since agents are subsystems of a MAS, they interact within the global system
  - by essence of systems in general, rather than of MAS
- since agents are autonomous, only data (knowledge, information) crosses agent boundaries
- information & knowledge is exchanged between agents
  - leading to more complex patterns than message passing between objects

Autonomous Agents Do not *Need* a Goal I

Agents govern MAS computation

- by encapsulating control, agents are the main forces governing and pushing computation, and determining behaviour in a MAS
- along with control, agent should then encapsulate the *criterion* for regulating the thread(s) of control

Autonomous Agents Do not *Need* a Goal II

Autonomy as self-regulation

- the term "autonomy", at its very roots, means self-government, self-regulation, self-determination
  - "internal unit invocation" [Odell, 2002]
- this does *not* imply in any way that agents *needs* to have a goal, or a task, to be such—to be an agent, then
- however, this *does* imply that autonomy captures the cases of goal-oriented and task-oriented agents
  - where goals / tasks / ... play the role of the criteria for governing control

Goal-/Task-Orientedness does not Define Agents

Example: finite-state automaton with encapsulated control

- an agent might be a finite-state automaton
- encapsulating control as an independent thread
- equipped with state transition rules
- the criteria for the govern of control would there be embodied in terms of (finite) states and state transition rules

Goal-orientedness and task-orientedness are just possible features for agents

- they are not defining features anyway

Focus on...

Agents as Autonomous Entities

Definition (Agent)

Agents are *autonomous computational entities* [Omicini et al., 2008]

genus agents are computational entities

differentia agents are autonomous, in that they encapsulate control along with a criterion to govern it

Agents are *autonomous*

- from autonomy, many other features stem
  - autonomous agents *are* interactive, social, proactive, and situated
  - they *might* have goals or tasks, or be reactive, intelligent, mobile
  - they live within MAS, and *interact* with other agents through *communication actions*, and with the environment with *pragmatical actions*

"Weak" Notion of Agent

Four key qualities [Wooldridge and Jennings, 1995]

Weak agents are

- autonomous
- proactive
- reactive (to change)
- social

Are Autonomous Agents Intelligent?

Intelligence helps autonomy

- autonomous agents have to self-determine, self-govern, ...
- intelligence makes it easy for an agent to govern itself
- while intelligence is not mandatory for an agent to be autonomous
  - however, *intelligent autonomous agents* clearly make sense

Are Autonomous Agents Mobile?

Mobility could be an expression of autonomy

- autonomous agents encapsulate control
- at the end of the story, control might be independent of the environment where an agent lives—say, the virtual machine on which it runs
- *mobile autonomous agents* clearly make sense
  - even though mobility is not required for an agent to be autonomous

Do Autonomous Agents Learn?

Learning may improve agent autonomy

- by learning, autonomous agents may acquire new skills, improve their practical reasoning, etc.
- in short, an autonomous agent could learn how to make a better use out of its autonomy
- *learning autonomous agents* clearly make sense
  - learning, however, is not needed for an agent to be autonomous

"Strong" Notion of Agent

Mentalistic notion [Wooldridge and Jennings, 1995]

Strong agents have mental components such as

- beliefs
- desires
- intentions
- goals
- plans
- knowledge about the world
- ...

Intelligent agents and mental components

Intelligent autonomous agents are naturally (and quite typically) conceived as strong agents

Summing Up

Agents are autonomous

- *autonomy* is their defining features
- all other features descend from, or, are related to that one
- other relevant features – such as intelligence, mobility, ability to learn – clearly improve the impact of autonomy, yet do not define agency

Weak vs. strong

- the classical notions of weak and strong agency can be easily mapped upon autonomy

References I

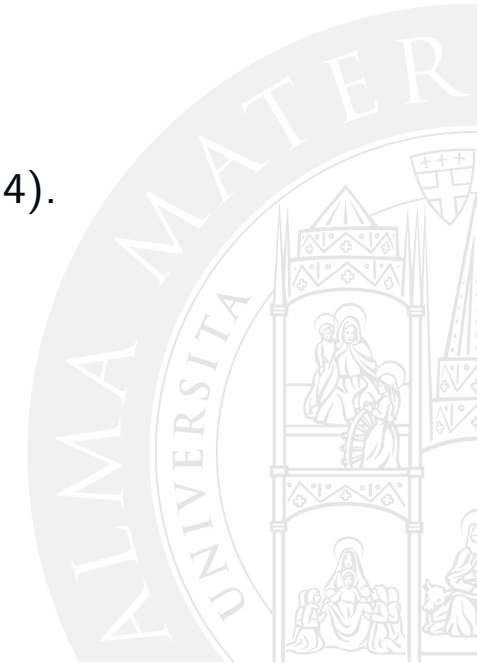
Agre, P. E. (1995).
Computational research on interaction and agency.
Artificial Intelligence, 72(1-2):1–52.
Special volume on computational research on interaction and agency, part 1.

Brooks, R. A. (1986).
Achieving artificial intelligence through building robots.
Technical report, Massachussets Institute of Technology (MIT).

Brooks, R. A. (1991).
Intelligence without representation.
Artificial Intelligence, 47:139–159.

Cossentino, M., Hilaire, V., Molesini, A., and Seidita, V., editors (2014).
Handbook on Agent-Oriented Design Processes.
Springer Berlin Heidelberg.

Fuggetta, A., Picco, G. P., and Vigna, G. (1998).
Understanding code mobility.
IEEE Transactions on Software Engineering, 24(5):342–361.



References II

Graesser, S. F. A. (1996).
Is it an agent, or just a program?: A taxonomy for autonomous agents.
In Jörg P. Müller, Michael J. Wooldridge, N. R. J., editor, *Intelligent Agents III Agent Theories, Architectures, and Languages: ECAI'96 Workshop (ATAL) Budapest, Hungary, August 12–13, 1996 Proceedings*, volume 1193 of *Lecture Notes In Computer Science*, pages 21 – 35. Springer.

Odell, J. (2002).
Objects and agents compared.
Journal of Object Technologies, 1(1):41–53.

O'Hare, G. M. and Jennings, N. R., editors (1996).
Foundations of Distributed Artificial Intelligence.
Sixth-Generation Computer Technology. John Wiley & Sons Ltd., hardcover edition.

Omicini, A., Ricci, A., and Viroli, M. (2008).
Artifacts in the A&A meta-model for multi-agent systems.
Autonomous Agents and Multi-Agent Systems, 17(3):432–456.
Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

References III

Omicini, A. and Rimassa, G. (2004).
Towards seamless agent middleware.
In *IEEE 13th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2004)*, pages 417–422, 2nd International Workshop "Theory and Practice of Open Computational Systems" (TAPOCS 2004), Modena, Italy. IEEE CS.
Proceedings.

Singh, M. P. (1998).
Agent communication languages: Rethinking the principles.
IEEE Computer, 31(12):40–47.

Suchman, L. A. (1987).
Plans and Situated Actions: The Problem of Human-Machine Communication.
Cambridge University Press, New York, NYU, USA.

Wegner, P. (1997).
Why interaction is more powerful than algorithms.
Communications of the ACM, 40(5):80–91.

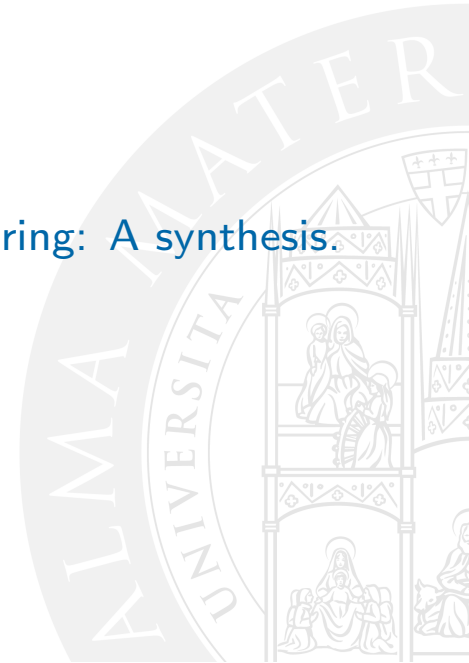
References IV

Weyns, D., Omicini, A., and Odell, J. (2007).
Environment as a first-class abstraction in multi-agent systems.
Autonomous Agents and Multi-Agent Systems, 14(1):5–30.
Special Issue on Environments for Multi-agent Systems.

Wooldridge, M. and Jennings, N. R. (1995).
Intelligent agents: Theory and practice.
Knowledge Engineering Review, 10(2):115–152.

Wooldridge, M. J. (2002).
An Introduction to MultiAgent Systems.
John Wiley & Sons Ltd., Chichester, UK.

Zambonelli, F. and Parunak, H. V. D. (2003).
Towards a paradigm change in computer science and software engineering: A synthesis.
The Knowledge Engineering Review, 18(4):329–342.



Agents & Multi-Agent Systems

Distributed Systems / Paradigms

Sistemi Distribuiti / Paradigmi

Andrea Omicini

andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna a Cesena

Academic Year 2017/2018