

Foundations of Multi-Agent Systems

Andrea Omicini
andrea.omicini@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna a Cesena

18th European Agent Systems Summer School (EASSS 2016)
Catania, Italy, 25 July 2016

- 1 Complex Software Systems
- 2 Towards Agents
- 3 Autonomy
- 4 Agents
- 5 Building Agents & MAS
- 6 Hot Topics in MAS

Outline

- 1 Complex Software Systems
- 2 Towards Agents
- 3 Autonomy
- 4 Agents
- 5 Building Agents & MAS
- 6 Hot Topics in MAS



Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

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 changing

- dramatically
- complexity is too *huge* for traditional CS & SE abstractions
 - like object-oriented technologies, or component-based methodologies
- new computational *paradigm*? [Kuh62]

The Impending Crisis of Software

The Scenario of the (Ongoing) Crisis

Computing systems

- are (going to be) anywhere
 - 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 to hold for 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 [Suc87]

- 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

Situatedness II

Environment as a first-class entity [WOO07]

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

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* [Weg97]

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

- unstopable 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 [ZP03]

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - **Away from Objects**
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Evolution of Programming Languages: The Picture

	Monolithic Programming	Modular Programming	Object-Oriented Programming	Agent Programming
Unit Behavior	Nonmodular	Modular	Modular	Modular
Unit State	External	External	Internal	Internal
Unit Invocation	External	External (CALled)	External (message)	Internal (rules, goals)

[Ode02]

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) [Sin98]
- 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 [Ode02] I

Decentralisation

- Object-based systems are completely pre-determined in control: control is essentially 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 rise to emergent phenomena

Philosophical Differences [Ode02] 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 [Ode02]

- 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, ...

Outline

- 1 Complex Software Systems
- 2 Towards Agents**
- 3 Autonomy
- 4 Agents
- 5 Building Agents & MAS
- 6 Hot Topics in MAS



Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - **Moving Toward Agent Technologies**
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

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 [OR04]

- 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 → new technologies change the way in which systems are conceived

development process → new technologies change the way in which systems are developed

economical environment → 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 Methodologies

- adopting agent-based metaphors and abstractions to formulate new practises in software engineering
- current state of AOSE methodologies [CHMS14]
 - 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, ...

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - **The Many Agents Around**
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

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* acceptance 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*

AI & Agents: A Note

Reversing perspective [OP06]

- results from AI and MAS research are no longer easily distinguishable
- agents and MAS work as the introductory metaphors to most of the AI results
 - as exemplified by one of the most commonly used AI textbooks [RN02]
- classic AI results on planning, practical reasoning, knowledge representation, machine learning, and the like, have become the most obvious and fruitful starting points for MAS research and technologies
- it is quite rare nowadays that new findings or lines of research in AI might ignore the agent abstractions at all
- altogether, rather than a mere subfield of AI, agents and MAS could be seen as promoting a novel paradigm, providing a new and original perspective about computational intelligence and intelligent systems

On the Notion of Agent in DAI [Woo02]

Overcoming the individual dimension

No more agents

- as single units encapsulating individual intelligence
- acting *alone* within an oversimplified environment

Social acceptance of agency

Agents are

- *individuals* within a *society* of agents
 - components of a *multiagent system* (MAS)
- distributed within a *distributed environment*

Agent Features in DAI [OJ96]

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

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

An agent understands its own capabilities...

- ... in terms of actions, pre-conditions on actions, effects of actions
- ... where “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 II

An agent is able to build a plan of its actions

- An agent builds possible plans of action according to
 - its goal/task
 - its knowledge of the environment
- An agent
 - *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?
- The above properties are both important and potentially misleading, since they are apparently intuitive, and everybody thinks he/she understands them exactly

A DAI Agent Learns

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 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 [Gra96]

- 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 [Ode02]

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

A P&D Agent is Mobile [FPV98]

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 made evident 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 [Bro91]
- *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

Putting everything together

The area of Multiagent Systems (MAS)...

- ... is today an independent research field & scientific community
- ... building a coherent conceptual framework, by drawing from the results of the many contributing areas

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

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

So, what is new here?

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

A MAS Agent is Interactive

Limited perception, limited capabilities

- it depends on other agents and external resources for the achievement of its goal / task
- it should interact with other agents and the environment [Agr95]
 - communication actions & pragmatological actions

A MAS agent does not live in isolation

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

Key-abstractions for MAS

- agents
- society
- environment

Outline

- 1 Complex Software Systems
- 2 Towards Agents
- 3 Autonomy**
- 4 Agents
- 5 Building Agents & MAS
- 6 Hot Topics in MAS



Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - **Autonomy in Dictionaries**
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Oxford Dictionary of English (2nd Edition revised 2005)

Etimology

Early 17th cent.: from Greek *autonomia*, from *autonomos* 'having its own laws', from *autos* 'self' + *nomos* 'law'.

Dictionary

autonomy

- the right or condition of self-government
- a self-governing country or region
- freedom from external control or influence; independence.
- (in Kantian moral philosophy) the capacity of an agent to act in accordance with objective morality rather than under the influence of desires

Oxford Thesaurus of English (2nd Edition revised 2008)

Thesaurus

autonomy

- self-government, independence, self-rule, home rule, sovereignty, self-determination, freedom, autarchy;
- self-sufficiency, individualism.

Merriam-Webster I

Dictionary

autonomy

- 1 the quality or state of being self-governing; especially: the right of self-government
- 2 self-directing freedom and especially moral independence
- 3 a self-governing state

Synonyms accord, free will, choice, self-determination, volition, will

Antonyms dependence (also dependance), heteronomy, subjection, unfreedom

Merriam-Webster II

Thesaurus

autonomy

- 1 the act or power of making one's own choices or decisions:
accord, free will, choice, self-determination, volition, will
- 2 the state of being free from the control or power of another:
freedom, independence, independency, liberty,
self-determination, self-governance, self-government,
sovereignty

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - **Autonomy in Philosophy**
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Internet Encyclopedia of Philosophy I

Many acceptations of autonomy

- general** an individual's capacity for self-determination or self-governance
- folk** inchoate desire for freedom in some area of one's life
- personal** the capacity to decide for oneself and pursue a course of action in one's life
- moral** the capacity to deliberate and to give oneself the moral law, rather than merely heeding the injunctions of others
- political** the property of having one's decisions respected, honored, and heeded within a political context

Internet Encyclopedia of Philosophy II

Individual autonomy

- after Kant, autonomy is an essential trait of the *individual*, and strictly related with its *morality*, represented by some high-level ethical principles
- then, with the relation between its inner self and its individual actions
- that is, mind and behaviour

Internet Encyclopedia of Philosophy III

Independence from oneself

- a more demanding notion of autonomy requires not only self-determination, but also *independence from oneself*
- this conception is connected with notions of freedom and choice, and (maybe) non-determinism
- and requires the ability of reasoning on (and possibly changing) not just one own course of actions, but one own goals

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - **Autonomy in Military**
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Unmanned Systems Integrated Roadmap FY 2011-2036 I

Automatic vs. autonomous

Automatic systems are fully pre-programmed and act repeatedly and independently of external influence or control. An automatic system can be described as self-steering or self-regulating and is able to follow an externally given path while compensating for small deviations caused by external disturbances. However, the automatic system is not able to define the path according to some given goal or to choose the goal dictating its path.

Autonomous systems are self-directed toward a *goal* in that they do not require outside control, but rather are governed by laws and strategies that direct their behavior. Initially, these control algorithms are created and tested by teams of human operators and software developers. However, if *machine learning* is utilized, autonomous systems can develop modified strategies for themselves by which they select their behavior. An autonomous system is self-directed by choosing the behavior it follows to reach a human-directed goal.

Unmanned Systems Integrated Roadmap FY 2011-2036 II

Four Levels of Autonomy for Unmanned Systems (*Guy Edwards*)

Various levels of autonomy in any system guide how much and how often humans need to interact or intervene with the autonomous system:

- human operated
- human delegated
- human supervised
- fully autonomous

Unmanned Systems Integrated Roadmap FY 2011-2036 III

Level	Name	Description
1	Human Operated	A human operator makes all decisions. The system has no autonomous control of its environment although it may have information-only responses to sensed data.
2	Human Delegated	The vehicle can perform many functions independently of human control when delegated to do so. This level encompasses automatic controls, engine controls, and other low-level automation that must be activated or deactivated by human input and must act in mutual exclusion of human operation.
3	Human Supervised	The system can perform a wide variety of activities when given top-level permissions or direction by a human. Both the human and the system can initiate behaviors based on sensed data, but the system can do so only if within the scope of its currently directed tasks.
4	Fully Autonomous	The system receives goals from humans and translates them into tasks to be performed without human interaction. A human could still enter the loop in an emergency or change the goals, although in practice there may be significant time delays before human intervention occurs.

Unmanned Systems Integrated Roadmap FY 2011-2036 IV

Autonomy & Unpredictability

- The special feature of an autonomous system is its ability to be goal-directed in *unpredictable situations*.
- This ability is a significant improvement in capability compared to the capabilities of automatic systems.
- An autonomous system is able to *make a decision* based on a set of rules and/or limitations.
- It is able to determine *what information is important* in making a decision.
- It is capable of a *higher level of performance* compared to the performance of a system operating in a predetermined manner.

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - **Autonomy in Social Sciences & AI**
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Autonomy as a Relational Concept [Cas95] I

Autonomy as a social concept

- an agent is autonomous mostly in relation to other agents
- autonomy has no meaning for an agent in isolation

Autonomy from environment

- the Descartes' problem: (human, agent) behaviour is *affected by* the environment, but is *not depending on* the environment
- situatedness, reactivity, adaptiveness do not imply lack of autonomy

Outline

- 1 Complex Software Systems
- 2 Towards Agents
- 3 Autonomy
- 4 Agents**
- 5 Building Agents & MAS
- 6 Hot Topics in MAS

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 etymology 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* [Suc87]

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 [Bro86, Bro91]
- 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*
[Gra96, Ode02]

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” [Ode02]
- 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

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - **Defining Agents**
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Agents as Autonomous Entities [ORV08]

Definition (Agent)

Agents are *autonomous computational entities*

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 [WJ95]

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 [WJ95]

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

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - **Mentalistic Agents**
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Modelling Agents with Mental States I

Modelling agents based on **mental states**...

- ... eases the development of agents exhibiting complex behaviour
- ... provides us with a familiar, non-technical way of understanding and explaining agents
- ... allows the developer to build MAS by adopting the perspective of a cognitive entity engaged in complex tasks—e.g., what would **I** do in the same situation?
- ... simplifies the construction, maintenance, and verification of agent-based applications
- ... is useful when the agent has to communicate and interact with users or other system entities

Modelling Agents with Mental States II

[Den07]

The intentional stance is the strategy of interpreting the behaviour of an entity (person, animal, artifact, whatever) by treating it as if it were a rational agent who governed its 'choice' of 'action' by a 'consideration' of its 'beliefs' and 'desires'.

*The scare-quotes around all these terms draw attention to the fact that some of their standard connotations may be set aside in the interests of exploiting their central features: **their role in practical reasoning**, and hence in the prediction of the behaviour of practical reasoners.*

Modelling Agents with Mental States III

Agents with mental states

- agents governing their behaviour based on *internal states* that mimic cognitive (human) *mental states*
 - epistemic states* representing agents *knowledge*—their knowledge on the world
 - i.e., percepts, beliefs
 - motivational states* representing agents *objectives*—what they aim to achieve
 - i.e., goals, desires
- the process of selecting one action to execute among the many available based on the actual mental states is called *practical reasoning*

Practical vs. Epistemic Reasoning

Practical reasoning is reasoning directed towards actions—the process of figuring out what to do in order to achieve what is desired

[Bra87]

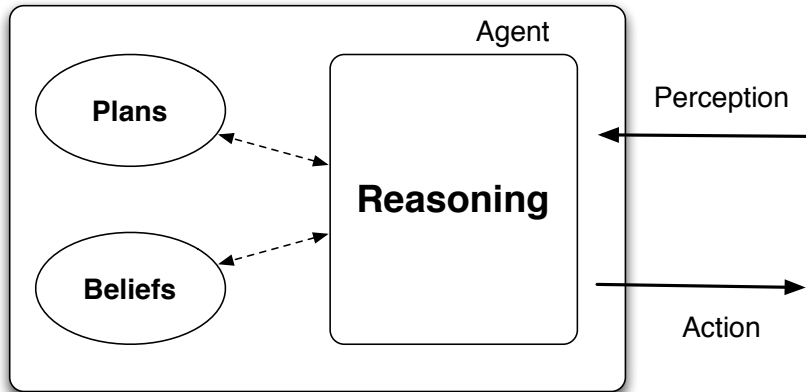
Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes.

Epistemic reasoning is reasoning directed towards knowledge—the process of updating information, replacing old information (no longer consistent with the world state) with new information

Practical Reasoning

- practical reasoning consists of two main cognitive activities
 - deliberation** when the agent makes decision on what state of affairs the agent desire to achieve
 - means-ends reasoning** when the agent makes decisions on how to achieve these state of affairs
- the outcome of the deliberation phase are the *intentions*
 - what agent desires to achieve, or what he desires to do
- the outcome of the means-ends reasoning phase is the selection a given course of actions
 - the workflow of the actions the agent intends to adopt in order to achieve its own goals expressed as intentions

Basic Architecture of a Mentalistic Agent



The BDI Framework I

According to [DG11]

- one of the most popular and successful framework for agent technology is defined by Rao and Georgeff [RG92]
- there, the notions of *belief*, *desire*, and *intention* are the core ones
- hence, agents in this framework are *mentalistic* ones, and are typically referred to as *BDI agents*

The BDI Framework II

beliefs represent at any time the agent's current *knowledge about the world*, including

- information about the current state of the environment inferred from perception devices
- messages from other agents
- internal information

desires represent a *state of the world* the agent is trying *to achieve*

intentions are the chosen means to achieve the agent's desires, and are generally implemented as plans and post-conditions

- as in general it may have *multiple desires*, an agent can have *a number of intentions active at any one time*
- these intentions may be thought of as *running concurrently*, with *one chosen intention active at any one time*

The BDI Framework III

Besides these components, the BDI model includes

- plan library** — namely, a set of “recipes” representing the *procedural knowledge* of the agent
 - event queue** — where
 - *events* — either perceived from the environment or generated by the agent itself to notify an update of its belief base
 - *internal subgoals* — generated by the agent itself while trying to achieve a desire
- are stored

The BDI Framework IV

Plans & plan library

- usually, BDI-style agents do not adopt first principles planning at all
- all plans must be *generated* by the agent programmer *at design time*, which are then *selected* for execution at run time
- pre-programmed plans are collected in the plan library
- the planning done by agents consists entirely of context-sensitive subgoal expansion, which is deferred until a subgoal is selected for execution

The BDI Abstract Architecture

Accordingly, the *abstract architecture* proposed by [RG92] comprise three dynamic and global structures representing agent's Beliefs, Desires, and Intentions (BDI), along with an input queue of events

- *update* (write) and *query* (read) operations are possible upon the three structures
 - update operation are subject to compatibility requirements
 - formalised constraints hold upon the mental attitudes
- the events that the system is able to recognise could be either external – i.e., coming from the environment – or internal ones—i.e., coming from some reflexive action
 - events are assumed to be *atomic*, and can be recognised after they have occurred

Summing Up

Agents are *autonomous computational entities*

- agents encapsulate control along with a criterion to govern it
- from autonomy, many other features (more or less) 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*

Intelligent agents are core components for complex systems

- *strong* vs. *weak* notion of agency
 - BDI architecture for mentalistic agents
- mobility
- learning capabilities

Outline

- 1 Complex Software Systems
- 2 Towards Agents
- 3 Autonomy
- 4 Agents
- 5 Building Agents & MAS**
- 6 Hot Topics in MAS

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - **Paradigm Shifts**
 - Programming Agents
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Paradigm Shifts in Software Engineering

New classes of programming languages

- new classes of programming languages come from paradigm shifts in Software Engineering
 - new meta-models / new ontologies for artificial systems build up new spaces
 - new spaces have to be “filled” by some suitably-shaped new (class of) programming languages, incorporating a suitable and coherent set of new abstractions
- the typical procedure
 - first, existing languages are “stretched” far beyond their own limits, and become cluttered with incoherent abstractions and mechanisms
 - then, academical languages covering only some of the issues are proposed
 - finally, new well-founded languages are defined, which cover new spaces adequately and coherently

The Problem of PL & SE Today

Things are running too fast

- new classes of programming languages emerge too fast from the needs of real-world software engineering
- however, technologies (like programming language frameworks) require a reasonable amount of time (and resources, in general) to be suitably developed and stabilised, before they are ready for SE practise
- most of the time, SE practitioners have to work with languages (and frameworks) they know well, but which do not support (or, incoherently / insufficiently support) required abstractions & mechanisms
- this makes methodologies more and more important with respect to technologies, since they can help covering the “abstraction gap” in technologies

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - **Programming Agents**
 - Programming Agent Interaction
 - Programming MAS
- 6 Hot Topics in MAS

Agents Without Agent Languages

What if we do not have an agent language available?

- for either theoretical or practical reasons, it may happen
 - we may need an essential Prolog feature, or be required to use Java
- what we do need to do: *(1) define*
 - adopt an agent definition, along with the agent's required / desired features
 - choose agent architecture accordingly, and according to the MAS needs
 - define a model and the languages for agent actions, both communicative and pragmatical
- what we do need to do: *(2) map*
 - map agent features, architecture, and action model / languages upon the existing abstractions, mechanisms & constructs of the language chosen
 - thus building an *agent abstraction layer* over our non-agent language foundation

The Agent Abstraction

AOP languages have *agent* as a basic abstraction

- an agent-oriented programming (AOP) language should support one agent definition (at least)
 - so, straightforwardly supporting mobility in case of mobile agents, intelligence somehow in case of intelligent agents, . . . , by means of well-defined language constructs
- required agent features play a fundamental role in defining language constructs

Agent Architectures

AOP languages support agent *architectures* [Woo02]

- agents have (essential) features, but they are built around an *agent architecture*, which defines both its internal structure, and its functioning
- an agent programming language should support one (or more) agent architecture(s),
- the agent architecture influences possible agent features

Agent Oriented Programming Languages (AOP) I

Programming languages for autonomous agents

- weak agents
- architecture to implement autonomous behaviour for distributed systems
- focus on conceptual integrity, interoperability, distribution, mobility, ...
 - agent-oriented middleware

Agent Oriented Programming Languages (AOP) II

Example

JADE a Java-based framework to develop agent-based applications in compliance with the **FIPA specifications** for interoperable, intelligent, multi-agent systems

where

- FIPA stands for *Foundation for Intelligent Physical Agents*
<http://www.fipa.org/>
- FIPA is the IEEE Computer Society standards organisation that promotes agent-based technology and the interoperability of its standards with other technologies

Agent Oriented Programming Languages (AOP) III

Programming languages for cognitive agents

- mentalistic agents
 - either BDI [RG91] or other cognitive architectures
- facilities and structures to represent internal knowledge, goals, ...
- architecture for implementing practical reasoning

Agent Oriented Programming Languages (AOP) IV

Examples

- Jason** (Brasil) <http://jason.sourceforge.net/> Agent platform and language for BDI agents based on AgentSpeak(L)
- JADEX** (Germany) <http://www.activecomponents.org/> Agent platform for BDI and Goal-Directed Agents
- 2APL** (Netherlands) <http://www.cs.uu.nl/2apl/> Agent platform providing programming constructs to implement cognitive agents based on the BDI architecture
- 3APL** (Netherlands) <http://www.cs.uu.nl/3apl/> A programming language for implementing cognitive agents
- ASTRA** (Ireland) <http://astralanguage.com/> A distributed / concurrent programming language based on AgentSpeak(TR).

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - **Programming Agent Interaction**
 - Programming MAS
- 6 Hot Topics in MAS

Intra- vs. Inter-Agent Languages I

Agent computation vs. agent interaction

- agents have both
 - an *inner behaviour*, and
 - an external, *observable behaviour*
- this reproduce the “computation vs. interaction / coordination” dichotomy of standard programming languages [Weg97, GC92]
 - computation** the inner functioning of a computational component
 - interaction** the actions determining the observable behaviour of a computational component

Intra- vs. Inter-Agent Languages II

What is new here?

- agent autonomy is new
 - the observable behaviour of an agent as a computational component is *driven / governed* by the agent itself
 - e.g., intelligent agents do practical reasoning—reasoning about actions—so that computation “computes” over the interaction space—in short, agent *coordination*

Languages for computation, languages for interaction

- agent programming languages should be either / both
 - intra-agent languages to define (agent) *computational behaviour*
 - inter-agent languages to define (agent) *interactive behaviour*

Intra- vs. Inter-Agent Languages III

Example: Agent Communication Languages (ACL)

- ACL are the easiest example of inter-agent language [Sin98]
 - they just define how agents speak with each other
 - however, these languages may have some requirements on internal architecture / functioning of agents

Agent Communication Languages (ACL) I

Speech acts

- inspired by the study of human communication
- communication is based on direct exchange of messages between agents
 - specifying agent communicative actions
- speaking agent acts to change the world around
 - in particular, to change the belief of another agent
- every message has three fundamental parts
 - performative** the pragmatics of the communicative action
 - content** the syntax of the communicative action
 - ontology** the semantics of the communicative action

Agent Communication Languages (ACL) II

Examples

- examples, working as standard protocols for information exchange between agents

KQML Knowledge Query Manipulation Language

<http://www.cs.umbc.edu/kqml/> [LF97]

FIPA ACL FIPA Agent Communication Language

<http://www.fipa.org/repository/aclspecs.html>
[FIP02]

Agent Observable Behaviour

AOP languages support agent *model of action*

- agent observable behaviour is not just communication, agent actions are not just communication actions
- agents act through either *communication* or *pragmatical actions*
- altogether, these two sorts of action define the admissible space for agent's observable behaviour
 - a *communication language* [Sin98] defines how agents speak to each others
 - an *action language* [GL93] should define how an agent can act over its environment
- AOP languages should account for *both* sorts of languages

Outline

- 1 Complex Software Systems
 - Toward a Paradigm Change
 - Away from Objects
- 2 Towards Agents
 - Moving Toward Agent Technologies
 - The Many Agents Around
- 3 Autonomy
 - Autonomy in Dictionaries
 - Autonomy in Philosophy
 - Autonomy in Military
 - Autonomy in Social Sciences & AI
- 4 Agents
 - Defining Agents
 - Mentalistic Agents
- 5 Building Agents & MAS
 - Paradigm Shifts
 - Programming Agents
 - Programming Agent Interaction
 - **Programming MAS**
- 6 Hot Topics in MAS

Programming the Interaction Space I

The space of MAS interaction

- inter-agent language roughly define the space of (admissible) MAS interaction
- inter-agent language should not be merely seen from the viewpoint of the individual agent (*subjective viewpoint*)
- the overall view on the space of (admissible) MAS interaction is the MAS engineer's viewpoint (*objective viewpoint*)
 - *subjective* vs. *objective* viewpoint over interaction [Sch01, OO03]

Programming the Interaction Space II

Enabling / governing / constraining the space of MAS interaction

- a number of inter-disciplinary fields of study insist on the space of (system) interaction
 - communication
 - dialogue / argumentation
 - coordination
 - organisation
 - security
 - workflow management
 - e-institutions / normative systems
 - ...

Coordination I

Coordination

- many different definitions around [OZKT01]
- in short, coordination is managing / governing interaction in any possible way, from any viewpoint [Weg96]
- Coordination in MAS is a *middleware* issue [OOR04]

Coordination II

Coordination models

- coordination models for distributed systems [GC92] perfectly fit MAS as action languages [COZ00], according to the objective coordination viewpoint [Sch01, OO03]
 - e.g., tuple-based language [Gel85]
- as software integrators [Cia96], coordination models get together agents within a MAS
- coordination models *govern MAS interaction* by mediating agent interaction through coordination primitives [COZ00]
- agent *societies* can be built around coordination *media*
 - e.g., tuple spaces [Gel85]

Coordination III

Why coordination models in MAS?

micro-level individual goals

- individual agents are built around some individual goal/task, which represent the micro-level of MAS

macro-level social/global goals

- agent societies / MAS are built around some global goal/task, which represent the macro-level of MAS
- agent societies can be built around coordination media

micro-level vs. macro-level

- making everything work smoothly is typically the main issue in governing the interaction space in MAS
- thus, the main issue of MAS coordination

Coordination IV

Example

TuCSoN Tuple Centers Spread over the Network
<http://tucson.unibo.it/> [LF97], a tuple-based
middleware for the agent coordination [OZ99].

In TuCSoN, the action language is basically represented by LINDA-based
primitives—rd, in, out.

Outline

- 1 Complex Software Systems
- 2 Towards Agents
- 3 Autonomy
- 4 Agents
- 5 Building Agents & MAS
- 6 Hot Topics in MAS**



Technologies, Methodologies & Standards I

Developing usable technologies

- a lot of experimental work, a number of nice agent technologies developed
- reliable agent technologies are under development
- integration with common technology standards

Providing usable methodologies and tools

- industry-level AOSE methodologies are needed [BGZ04, ZO04]
- AO development *process* should be addressed, too [CHMS14]
- the issue of *tools* is a key one

Technologies, Methodologies & Standards II

Building a shared AO technology environment worldwide

- FIPA started a decade ago [Fou05]
- OMG and IEEE (FIPA committee) are taking the lead today

Applications, of course

- some papers, already [PM08]

Autonomous Systems

MAS for the design and implementation of autonomous systems

- agents as autonomous components [HCF03]
- social action [Cas98]
- agent societies as aggregates featuring autonomy as an overall properties
- architecture for *distributed autonomy* [SO16]

Autonomous robots

- physical autonomy [Ros14]
- cognitive architectures for robot intelligence
 - EASSS 2016 tutorial on “Programming Autonomous Robots using PROFETA and the BDI model” by Corrado Santoro
- agents to model individual robots, MAS to model robot teams
 - workshop ARMS @ AAMAS 2016

Simulation

Using MAS for simulating complex systems

- real-world systems [Ode02]
- simulation of complex software systems
 - workshop MABS @ AAMAS 2016
- simulation for SE [MCOV13]

Knowledge-Intensive Systems (KIS)

Agents for KIS

- developing over *distributed cognition* [Kir99]
- within *knowledge-intensive environments* (KIE) [MO15]
- both intelligence and mobility are features, along with *self-* techniques*
- from *ontologies* to *semantic coordination* [NOV13]

Multi-Paradigm Integration

Agents as a meta-level paradigm

- subsuming OOP in AOP: wrapping objects [Ode02], using objects [ORV08]
- many language paradigms altogether [JBN⁺15]
- middleware integration for complex systems: MAS & EBS (event-based systems) [OM15]
 - EASSS 2016 tutorial on “Complex Event Recognition in Multi-Agent Systems” by Alexander Artikis

Complex Systems as MAS

Self-organising MAS

- self-* techniques for MAS [DMSGK06]
- mapping natural & social systems [Ode02]
- self-* techniques for SE

Intelligent Complex Systems

- encapsulating intelligence within individual agents
- as well as within social abstractions [Cas98]
- also adopting self-adaptation techniques

So What?

Motivations, basics & issues for agents and MAS

- ... time for questions...
- ... thanks for listening...
- ... and enjoy EASSS 2016!

References I



Philip E. Agre.

Computational research on interaction and agency.

Artificial Intelligence, 72(1-2):1–52, January 1995.

Special volume on computational research on interaction and agency, part 1.



Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors.

Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*.

Kluwer Academic Publishers, June 2004.



Michael E. Bratman.

Intention, Plans, and Practical Reason.

Harvard University Press, November 1987.



Rodney A. Brooks.

Achieving artificial intelligence through building robots.

Technical report, Massachusetts Institute of Technology (MIT), May 1986.



Rodney A. Brooks.

Intelligence without representation.

Artificial Intelligence, 47:139–159, 1991.

References II



Cristiano Castelfranchi.

Guarantees for autonomy in cognitive agent architecture.

In Michael J. Wooldridge and Nicholas R. Jennings, editors, *Intelligent Agents*, volume 890 of *Lecture Notes in Computer Science*, pages 56–70. Springer Berlin Heidelberg, 1995.
ECAI-94 Workshop on Agent Theories, Architectures, and Languages (ATAL) Amsterdam, The Netherlands 8–9 August 1994. Proceedings.



Cristiano Castelfranchi.

Modelling social action for ai agents.

Artificial Intelligence, 103(1):157–182, August 1998.



Massimo Cossentino, Vincent Hilaire, Ambra Molesini, and Valeria Seidita, editors.

Handbook on Agent-Oriented Design Processes.

Springer Berlin Heidelberg, 2014.



Paolo Ciancarini.

Coordination models and languages as software integrators.

ACM Computing Surveys, 28(2):300–302, June 1996.

References III



Paolo Ciancarini, Andrea Omicini, and Franco Zambonelli.

Multiagent system engineering: The coordination viewpoint.

In Nicholas R. Jennings and Yves Lespérance, editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *LNAI*, pages 250–259.

Springer-Verlag, 2000.

6th International Workshop (ATAL'99), Orlando, FL, USA, 15–17 July 1999. Proceedings.



Daniel Dennett.

Intentional systems theory.

In Ansgar Beckermann and Sven Walter, editors, *Oxford Handbook of the Philosophy of Mind*, chapter 19. Oxford University Press, 2007.



Aniruddha Dasgupta and Aditya K. Ghose.

BDI agents with objectives and preferences.

In Andrea Omicini, Sebastian Sardina, and Wamberto Vasconcelos, editors, *Declarative Agent Languages and Technologies VIII*, volume 6619 of *Lecture Notes in Computer Science*, pages 22–39. Springer Berlin Heidelberg, 2011.



Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos.

Self-organization in multi-agent systems.

The Knowledge Engineering Review, 20(02):165–189, January 2006.

References IV



Foundation for Intelligent Physical Agents (FIPA).
Agent Communication Language Specifications, 2002.



Foundation for Intelligent Physical Agents.
FIPA home page, 2005.



Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna.
Understanding code mobility.
IEEE Transactions on Software Engineering, 24(5):342–361, May 1998.



David Gelernter and Nicholas Carriero.
Coordination languages and their significance.
Communications of the ACM, 35(2):97–107, February 1992.



David Gelernter.
Generative communication in Linda.
ACM Transactions on Programming Languages and Systems, 7(1):80–112, January 1985.



Michael Gelfond and Vladimir Lifschitz.
Representing action and change by logic programs.
The Journal of Logic Programming, 17(2-4):301–321, November 1993.

References V



Stan Franklin Art Graesser.

Is it an agent, or just a program?: A taxonomy for autonomous agents.

In Nicholas R. Jennings Jörg P. Müller, Michael J. Wooldridge, 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, 1996.



Henry Hexmoor, Cristiano Castelfranchi, and Rino Falcone, editors.

Agent Autonomy.

Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer US, 2003.



Howell Jordan, Goetz Botterweck, John Noll, Andrew Butterfield, and Rem Collier.

A feature model of actor, agent, functional, object, and procedural programming languages.

Science of Computer Programming, 98:120–139, February 2015.



David Kirsh.

Distributed cognition, coordination and environment design.

In *Proceedings of the 3rd European Conference on Cognitive Science (ECCS 1999)*, pages 1–11, 1999.

References VI



Thomas S. Kuhn.

The Structure of Scientific Revolutions.

University of Chicago Press, Chicago, IL, USA, 1962.



Yannis Labrou and Tim Finin.

Semantics and conversations for an agent communication language.

In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 235–242.

Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.



Ambra Molesini, Matteo Casadei, Andrea Omicini, and Mirko Viroli.

Simulation in Agent-Oriented Software Engineering: The SODA case study.

Science of Computer Programming, 78(6):705—714, June 2013.

Special Section on Agent-oriented Design methods and Programming Techniques for Distributed Computing in Dynamic and Complex Environments.

References VII



Stefano Mariani and Andrea Omicini.

Anticipatory coordination in socio-technical knowledge-intensive environments:
Behavioural implicit communication in MoK.

In Marco Gavanelli, Evelina Lamma, and Fabrizio Riguzzi, editors, *AI*IA 2015, Advances in Artificial Intelligence*, volume 9336 of *Lecture Notes in Computer Science*, chapter 8, pages 102–115. Springer International Publishing, 23–25 September 2015.

XIVth International Conference of the Italian Association for Artificial Intelligence, Ferrara, Italy, September 23–25, 2015, Proceedings.



Elena Nardini, Andrea Omicini, and Mirko Viroli.

Semantic tuple centres.

Science of Computer Programming, 78(5):569–582, May 2013.

Special section: Self-Organizing Coordination.



James Odell.

Objects and agents compared.

Journal of Object Technologies, 1(1):41–53, May–June 2002.

References VIII

 Gregory M.P. O'Hare and Nicholas R. Jennings, editors.

Foundations of Distributed Artificial Intelligence.

Sixth-Generation Computer Technology. John Wiley & Sons Ltd., hardcover edition, April 1996.

 Andrea Omicini and Stefano Mariani.

Reconciling event- and agent-based paradigms in the engineering of complex systems: The role of environment abstractions.

In Danny Weyns and Fabien Michel, editors, *Agent Environments for Multi-Agent Systems IV*, volume 9068 of *Lecture Notes in Computer Science*, pages 117–130. Springer International Publishing, November 2015.

4th International Workshop, E4MAS 2014 – 10 Years Later, Paris, France, May 6, 2014, Revised Selected and Invited Papers.

 Andrea Omicini and Sascha Ossowski.

Objective versus subjective coordination in the engineering of agent systems.

In Matthias Klusch, Sonia Bergamaschi, Peter Edwards, and Paolo Petta, editors, *Intelligent Information Agents: An AgentLink Perspective*, volume 2586 of *LNAI: State-of-the-Art Survey*, pages 179–202. Springer, 2003.

References IX



Andrea Omicini, Sascha Ossowski, and Alessandro Ricci.

Coordination infrastructures in the engineering of multiagent systems.

In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter 14, pages 273–296. Kluwer Academic Publishers, June 2004.



Andrea Omicini and Agostino Poggi.

Multiagent systems.

Intelligenza Artificiale, III(1-2):76–83, March-June 2006.

Special Issue: The First 50 Years of Artificial Intelligence.



Andrea Omicini and Giovanni Rimassa.

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, 14–16 June 2004. IEEE CS. Proceedings.

References X



Andrea Omicini, Alessandro Ricci, and Mirko Viroli.

Artifacts in the A&A meta-model for multi-agent systems.

Autonomous Agents and Multi-Agent Systems, 17(3):432–456, December 2008.

Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.



Andrea Omicini and Franco Zambonelli.

Coordination for Internet application development.

Autonomous Agents and Multi-Agent Systems, 2(3):251–269, September 1999.

Special Issue: Coordination Mechanisms for Web Agents.



Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors.

Coordination of Internet Agents: Models, Technologies, and Applications.

Springer Berlin Heidelberg, March 2001.



Michal Pěchouček and Vladimír Mařík.

Industrial deployment of multi-agent technologies: review and selected case studies.

Autonomous Agents and Multi-Agent Systems, 17(3):397–431, December 2008.

Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

References XI



Anand S. Rao and Michael P. Georgeff.

Modeling rational agents within a BDI architecture.

In James F. Allen, Richard Fikes, and Erik Sandewall, editors, *2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, San Mateo, CA, 1991. Morgan Kaufmann Publishers.



Anand S. Rao and Michael P. Georgeff.

An abstract architecture for rational agents.

In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *3rd International Conference on Principles of Knowledge Representation and Reasoning (KR '92)*, pages 439–449, Cambridge, MA, USA, 25-29 October 1992. Morgan Kaufmann. Proceedings.



Stuart J. Russell and Peter Norvig.

Artificial Intelligence: A Modern Approach.

Prentice Hall / Pearson Education International, Englewood Cliffs, NJ, USA, 2nd edition, 20 February 2002.

References XII



Bernd Rosslenbroich.

On the Origin of Autonomy. A New Look at the Major Transitions in Evolution. History, Philosophy and Theory of the Life Sciences. Springer International Publishing, 2014.



Michael Schumacher.

Objective Coordination in Multi-Agent System Engineering. Design and Implementation, volume 2039 of LNCS. Springer, April 2001.



Munindar P. Singh.

Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, December 1998.



Giovanni Sartor and Andrea Omicini.

The autonomy of technological systems and responsibilities for their use. In Nehal Bhuta, Susanne Beck, Hin-Yan Liu, Robin Geiß, and Claus Kreß, editors, *Autonomous Weapon Systems. Law, Ethics, Policy*, chapter 3, pages 39–74. Cambridge University Press, July 2016.

References XIII



Lucy A. Suchman.

Situated actions.

In *Plans and Situated Actions: The Problem of Human-Machine Communication*, chapter 4, pages 49–67. Cambridge University Press, New York, NYU, USA, 1987.



Peter Wegner.

Coordination as constrained interaction.

In Paolo Ciancarini and Chris Hankin, editors, *Coordination Languages and Models. First International Conference, COORDINATION '96 Cesena, Italy, April 15–17, 1996.*

Proceedings, Lecture Notes in Computer Science, pages 28–33. Springer Berlin Heidelberg, April 1996.



Peter Wegner.

Why interaction is more powerful than algorithms.

Communications of the ACM, 40(5):80–91, May 1997.



Michael Wooldridge and Nicholas R. Jennings.

Intelligent agents: Theory and practice.

Knowledge Engineering Review, 10(2):115–152, June 1995.

References XIV



Michael J. Wooldridge.

An Introduction to MultiAgent Systems.

John Wiley & Sons Ltd., Chichester, UK, March 2002.



Danny Weyns, Andrea Omicini, and James Odell.

Environment as a first-class abstraction in multi-agent systems.

Autonomous Agents and Multi-Agent Systems, 14(1):5–30, February 2007.

Special Issue on Environments for Multi-agent Systems.



Franco Zambonelli and Andrea Omicini.

Challenges and research directions in agent-oriented software engineering.

Autonomous Agents and Multi-Agent Systems, 9(3):253–283, November 2004.

Special Issue: Challenges for Agent-Based Computing.



Franco Zambonelli and H. Van Dyke Parunak.

Towards a paradigm change in computer science and software engineering: A synthesis.

The Knowledge Engineering Review, 18(4):329–342, December 2003.

URL

Slides

- on APICe

→ <http://apice.unibo.it/xwiki/bin/view/Talks/MasfoundationsEasss2016>

- on SlideShare

→ <http://www.slideshare.net/andreaomicini/foundations-of-multiagent-systems>

Foundations of Multi-Agent Systems

Andrea Omicini
`andrea.omicini@unibo.it`

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna a Cesena

18th European Agent Systems Summer School (EASSS 2016)
Catania, Italy, 25 July 2016