Agents: a technological opportunity

Luca Cernuzzi

DEI – Universidad Católica “Nuestra Señora de la Asunción” – Paraguay

Special thanks to: Franco Zambonelli
Università di Modena e Reggio Emilia - Italia

Agenda

- The Agent Paradigm
- Key Concepts for Agents and Multiagent Systems
- Agent Oriented Software Engineering
  - Abstractions for Agents and Multiagent Systems
  - Process Models and Methodologies
  - AOSE Open Issues
- Some Implementation Issues
- Conclusions

The Agent Paradigm

- Scientist and Engineers looking for methods, techniques, and notations to model the problems of the real world and to design the solutions for such problems
- In Software Engineering different paradigms have been proposed:
  - Structured
  - Object Orientation
  - Components
  - Agents
  - etc.
- From the Latin “agentis”: “those who act”.
  An “agent” is someone who act on behalf of other, with power to act derived from a delegation

Examples of Agents

- Real world: Secret Agents, Travel Agents, Sports/Showbiz Agents, Purchasing Agents, etc.
- Software: Filtering agents (antivirus, anti-spam), Shopbots/price comparison agents, etc.
- What do these jobs have in common?
  - They engage in tasks each with a specific goal (e.g., selling a house, looking for better prices in internet, etc.)
  - They are delegate by someone (who trust them)
  - They know how to do (have the power and the knowledge to do)

Software Agents

- In general, we can talk of “software agents” when
  - Referring to software that has a “goal” to pursue
  - Acting on our behalf to pursue that goal
  - Having the power and knowledge to pursue this goal in autonomy
- “Agent” is one of the more ubiquitous buzzwords in computer science today
  - It’s getting used for almost any piece of software
  - In several cases, inappropriately
- In any case, we need some more “technical” characterization and definition

What are Agents?

- There has been some debate
  - On what an agent is, and what could be appropriately called an agent
- Two main viewpoints (centered on different perspectives on autonomy):
  - The (strong) Artificial Intelligence viewpoint:
    - An agent must be, proactive, intelligent, and it must converse instead of doing client-server computing
  - The (weak) Software Engineering Viewpoint
    - An agent is a software component with internal (either reactive or proactive) threads of execution, and that can be engaged in complex interactions protocols

What are Multiagent Systems?

- Again….
  - The (strong) artificial intelligence viewpoint
    - A multiagent system is a society of individuals (AI software agents) that interact by exchanging knowledge and by negotiating with each other to achieve either their own interest or some global goal
  - The (weak) software engineering viewpoint
    - A multiagent system is a software systems made up of multiple independent and encapsulated loci of control (i.e., the agents) interacting with each other in the context of a specific application viewpoint….

The SE viewpoint on AO

- We commit to it because:
  - It focuses on the characteristics of agents that have impact on software development
    - Concurrency, interaction, multiple loci of control
    - Intelligence can be seen as a peculiar form of control independence; conversations as a peculiar form of interaction
  - It is much more general:
    - Does not exclude the strong AI viewpoint
    - Several software systems, even if never conceived as agents-based one, can be indeed characterized in terms of weak multi-agent systems
- Let's better characterize the SE perspective on agents…

Key Characteristics of Agents

- A software agent is a component that is:
  - **Goal-oriented**: designed and deployed to achieve a specific goal (or to perform a specific task)
  - **Autonomous**: capable of acting in autonomy towards the achievement of its specific goals, without being subject to a globally controlled thread of control
  - **Situated**: it execute in the context of a specific environment (computational or physical), and is able to act in that environment by sensing and affecting (via sensors and actuators)
- In addition, it can be:
  - **Proactive**. It can act opportunistically and in an unsolicited way towards the achievement of its goals (as opposed to Reactive agents, that acts only on reactions to events)
  - **Social**. Interact with other agents in a multiagent systems

The Concept of Goal-Orientedness

- How is a global application goal achieved?
- Division of labor (as in object-based applications)
  - Functions assigned to different components
  - Coordination is for composing functionalities to lead to global goal
  - As in pipe organizations
- Division of responsibilities (as in agent-based applications)
  - Sub-goals assigned to different components
  - Coordination is for orchestrating the achievement of a global goal
  - As in modern distributed organizations

The Concept of Autonomy

- Related to the “decision making”
- Centralized decision making, as in process-based and object-based applications
  - global goal achieved via a *global control scheme* for the application entities
  - design by *delegation of control*
- Distributed decision making, as in agent-based applications
  - sub-goals assigned to autonomous agents (integrating execution capabilities, i.e., threads) which try to achieve in autonomy their own goal
  - design by *delegation of responsibility*
  - *Agents can say “NO!”*

The Concept of Situatedness

- Context-awareness is important for adaptivity
  - And it is even more important when
    - Goal-orientedness
    - Distributed decision making
  - Are involved
- Unfortunately, there are also several agent systems that does not take situatedness into the proper account…
- Clearly, autonomy and situatedness make agent **adaptive entities**, suitable for the dynamics of modern software scenarios!!!

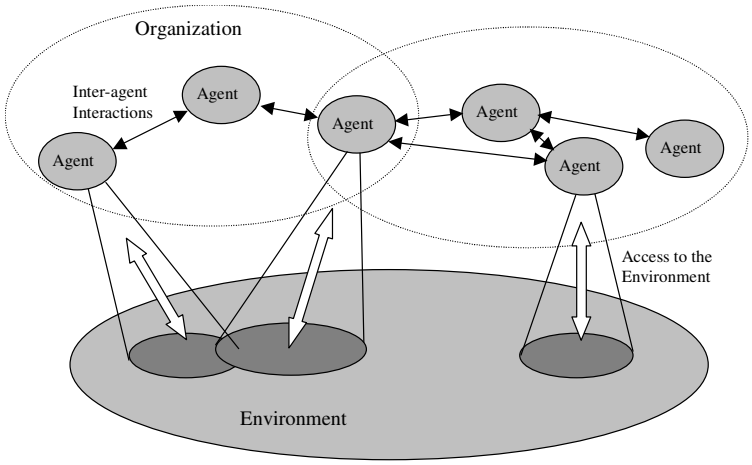
The Concept of Proactivity

- Not only agents have autonomous decision-making capabilities
  - They can also decide to autonomously activate towards the pursuing of the goal
  - They do not need any specific event or solicitation to do that
- Proactivity is a sort of extreme expression of autonomy
- Reactive agents are the less autonomous
- Proactive are the more autonomous

The Concept of Sociality

- Agents are rarely living in an isolated mono-agent world
  - They usually live in a multi-agent world
- Sociality refer to the fact that the typically interactions are more sophisticated than client-server ones
  - Exchange of knowledge
  - Delegation of tasks or responsibility
  - Open world, competitions in actions, negotiations
- Resembling more the interactions occurring in a society of humans…
- Clearly, the capability of acting in a social context is expression of adaptivity, and will make it possible to build, with agents, very **adaptive and complex systems**, able to deal with openness of the system and (together with situatedness) with environmental dynamic!!!

MAS Characterisation



Why Multiagent Systems? (1)

- A single agent has finite rationality
  - Limit on the amount of knowledge he can rationally handle in a given time
  - Require splitting a decision between more agents
  - E.g., an agent in charge of analyzing a very large search space
- Distribution
  - Several problems are intrinsically distributed
  - Knowledge can be acquired only in loco
  - Require more agents to be allocated where the knowledge can be obtained
  - E.g., agents devoted to the control of physical processes must monitor in loco

Why Multiagent Systems? (2)

- Interactions between personal agents
  - Many problems requires components of different stakeholders/organizations to interact
  - So, the problem is intrinsically composed of multiple agents
  - E.g., an E-commerce site require both the buyer and the seller, which will be represented by two different agents interacting with each other
- Multiagent organizations for real-world organizations
  - Software systems devoted to support the work of some human organization (e.g., a work group or a network enterprise)
  - Should somehow "mimic" the structure of the real-world organization to minimize conceptual mismatch between the real-world and the software organization

Why Multiagent Systems? (3)

- Multiagent systems are "paradigmatic" of modern distributed systems
  - Made up of decentralized autonomous components (sensors, peers, mobile devices, etc.)
  - Interacting with each other in complex way (P2P networks, pervasive computing environments, etc.)
  - Situated in some environment, computational or physical
- We live every day in a world which goes on by interactions of autonomous agents (humans), each with limited rationality and interacting with each other to achieve a common goal or a self-interest goal
  - This is how we live at work, with friends, when playing sports…

The Needs for Agent Oriented Software Engineering

- The need for specific abstractions for modeling and developing Agents and Multiagent Systems (MAS)
- The need for methods, methodologies, techniques, notations and tools for the design of Agents and MAS (Agent Oriented Software Engineering – AOSE)

Agent-Oriented Abstractions (1)

- The development of a multiagent system should fruitfully exploit **abstractions** coherent with the above characterization:
  - **Agents,** autonomous entities, independent loci of control, situated in an environment, interacting with each other
  - **Environment**, the world of resources agents perceive
  - **Interaction protocols**, as the acts of interactions between agents
- In addition, there may be the need of abstracting:
  - The **local context** where an agent lives (e.g., a sub-organization of agents) to handle mobility & opennes
- Such abstractions translates into concrete entities of the software system

Agent-Oriented Abstractions (2)

- Each approach may introduce further abstractions
  - Around which to model software and to organize the software process
    - E.g., roles, organizations, responsibilities, beliefs, desires and intentions…
  - Not directly translating into concrete entities of the software system
    - E.g. the concept of role is an aspect of an agent, not an agent

Common Abstractions

- agents
- roles
- activities with their corresponding goals
- interaction with the environment
- interaction between agents
- organizational aspects (structure and control)

Almost all methodologies and engineering approaches cover the mentioned abstractions but in different ways

AOSE – Process Models

- **Software Process Model:** prescribes around which phases a process should be organized (and possibly but not necessarily which activities should be executed in some of the phases), in which order such phases should be executed, and when iterations and coordination (if any) between the work of the different phases should occur
- Mainly Software Process Models:
  - Waterfall
  - Evolutionary or Incremental ("Agile" approaches)
  - Transformation
  - Spiral

AOSE – Methodologies

- **Method:** prescribes a way of performing some kind of activity within a process, to properly produce a specific output (i.e., an artifact or a document) starting from a specific input
- **Methodology:** is a collection of methods, covering and connecting different stages in a process, with the purpose of prescribe a certain coherent approach to solving a problem in the context of a software process

Phases → / Process Model and Methodology ↓	Requirements Elicitation	Requirements Analysis	Design	Coding and Implementation	Verification & Testing	Deployment
Waterfall Like						
Gaia		X	X			
Roadmap	X (partially)	X	X			
Prometheus	X (partially)	X	X	X	X	
MaSE	X (partially)	X	X	X	X(partially)	
AOR		X	X	X		
Evolutionary and Incremental						
OPM/MAS	X	X	X			X
MASSIVE		X	X	X	X	X
Ingenias		X	X	X		
Tropos	X	X	X	X		
PASSI and Agile PASSI		X	X	X	X	X
Transformation						
DESIRE	X	X	X	X	X(partially)	
Spiral						
MAS-CommonKADs	X	X	X	X	X(partially)	X

AOSE – Open Issues (1)

Process Models:

- Mainly centered on waterfall or evolutionary approaches
- Need for more Agile models to reach a greater acceptation of the industry
- Really relevant the Requirements Elicitation phase

AOSE – Open Issues (2)

- A lot of **AOSE methodologies**
  --> ¿Which Methodologies?
- The evaluation problem:
  - Qualitative
  - Not compared results
  - Partiality
- Possible ways:
  - Quantitative Evaluations
  - Better instruments to compare

AOSE – Open Issues (3)

- System organization influence the performance
- **Organizational abstractions** are relevant for MAS (society of agents which interact to reach a global goal)
  - Explicitly modeling the organizational structure
  - Modeling the control regime

AOSE – Open Issues (4)

Tools that support the Software Engineer

- We need different tools for the different phases
- For the design process (normally covering different phases of the process model) some CASE (Computer Aided Software Engineering) are available
  - They offer support for modeling activities (using notations for systems modeling)
  - Usually, they do not explicitly consider the process model
  - Limited Code Generation (usually just templates)

AOSE – Open Issues (5)

Design and construction of **adaptive MAS**

- A lot of current MAS undergo continuous changes/adaptations (new requirements, environments, real world organizations, etc.)
  - To the level of one or a couple of agents
  - The need to adapt the organizational structure and the control regime

Issues in Implementing Agents and MAS

- How can we move from agent-based design to concrete agent code?
- Methodologies should abstract from:
  - Internal agent architecture
  - Communication architecture
  - Implementation tools
- However, depending on tools the effort from design to implementation changes:
  - It depends on how much abstractions are close to the abstractions of agent-oriented design
  - The methodology could strongly invite to exploit a specific infrastructure

Intra-agent Issues: Implementing Agents

- Two main categories of tools to implement agents:
  - Object-oriented tools: are very much related to the object-oriented approach, e.g., Aglet;
  - BDI toolkits: are based on the BDI model (e.g., Jade).
- The choice of the tool to adopt is hard and there is no general answer:
  - Performances;
  - Maintenance;
  - … and many other issues

Inter-agent Issues: Implementing MAS

- Inter-agent implementation aspects are orthogonal to intra-agent ones
  - Given a set of agents with internal architecture and with specified interaction patterns
  - How can we glue them together?
    - Letting agents know each other
  - How to enable interactions?
    - Promoting spontaneous interoperability
  - How to rule interactions?
    - Preventing malicious or self-interested behaviors?

Conclusions

- In our humble opinion, agents is a very interesting paradigm in software engineering
  - AOSE abstractions and methodologies apply to a wide range of scenarios
- Several assessed research works already exist
  - Modeling work
  - Methodologies
  - Implementation Tools
- Still, there are a number of fascinating and largely unexplored open research directions…
  - Ubiquity, self-organization, performance….

Muchas Gracias!!!

- Thanks a lot!!!
- Grazie Mille!!!
- Muito Obrigado!!!

- Luca Cernuzzi (lcernuzz@uca.edu.py)