Multi-Agent Systems: A new paradigm for Systems of Systems

Eduardo Alonso
Department of Computer Science
City University London
London, United Kingdom
e-mail: E.Alonso@city.ac.uk

Nicos Karcanias
Systems & Control Engineering Centre
City University London
London, United Kingdom
e-mail: N.Karcanias@city.ac.uk

Ali G. Hessami
Vega Systems Ltd.
London, United Kingdom
e-mail: hessami@vegaglobalsystems.com

Abstract **–We present the notion of Systems of Systems, its drivers, and the challenges we face in conceptualizing, designing, implementing and validating them. In this work in progress we propose Multi-Agent Systems as a new paradigm, taken from Artificial Intelligence, which seems to fit the purpose.**

Keywords-multi-agent systems; system of systems; autonomous agents; dynamic adaptive systems

I. SYSTEMS OF SYSTEMS

Systems of Systems (SoS) have been defined as systems that describe the large-scale integration of many independent self-contained systems to satisfy global needs or multi-system requests. The main drivers behind the notion of SoS are various yet inter-related, namely,

- The increasing number of interacting systems with strong connectivity in society and in industry –which underlies the so-called "embedded world meets the Internet world" view.
- Emergent behavior with the need to balance cooperation and autonomy.
- Growing overall complexity of systems.

Such drivers are shown in the dimensions that define SoS, typically, the geographic distribution of the overall system, their operational and managerial independence, and their evolutionary development –a SoS evolves over time as the constituent systems are changed, added or removed— and emergent behavior –a SoS is not restricted to the capabilities of the constituent systems.

Such dimensions can be recognized in several types of large systems as identified in Table 1:

TABLE 1. TYPES OF SoS

Type	System	System of Systems
ICT powered	Car, road	Integrated Traffic Network
	Wind turbine, fossil	Smart Grid
	Computer, routers	Distributed IT System
Biological	Animal, plant	Herd, forest
Sociological	Family, school, church	Town, education, religion
Environmental	Weather, river	Eco-system
Organizational	Company	SCM, stock market, economy
Political	Town council	Parliament, EU, UN

Other examples include water management, emergency response, smart grid, railways, satellites, distributed control systems, supply chain management, and inter-court law relationships –to name a few.

One fundamental aspect in the analysis of SoS is to distinguish them from Composite Systems (CoS). [1][2][3][4] identified the following commonalities and differences:

a) Both CoS and SoS are compositions of simpler objects, or systems.

b) Both CoS and SoS are embedded in the environment of a larger system.

c) The objects, or sub-systems in CoS do not have their independent goal, they are not autonomous and their behavior is subject to the rules of the interconnection topology.

d) The interconnection rule in CoS is expressed as a graph topology.

e) The subsystems in SoS may have their own goals and some of them may be autonomous, semi-autonomous, or organized as autonomous groupings of composite systems

f) There may be a connection rule expressed as a graph topology for the information structures of the subsystems in a SoS.

g) The SoS has associated with it a *global game* where every subsystem enters as an agent with their individual Operational Set, Goals.

The comparison between SoS and traditional CoS illustrates the need for a paradigm shift in studying increasingly complex systems, and that such paradigm must focus on two main requirements

1. Rather than controlling systems the aim is to find means of influencing systems towards agreed common goals.
2. Develop approaches with incomplete models and dynamically evolving/changing requirements.

It is worth noting that in a similar manner to composite systems, SoS can be the outcome of natural evolution or entirely man made or a hybrid of these. In this context, agreed common goals may be more related to the total emergent properties arising from the constituents and natural ecology as opposed to deterministic requirements imposed on the system. This is particularly relevant when dealing with SoS that need to show context/situational awareness and that must be dependable to guarantee security and safety.

Table 2 summarizes the change of paradigm from old-classic approaches in systems theory to new-SoS according to various characteristics.

 ISBN: 978-1-61208-246-2

TABLE 2. PARADIGM SHIFT

Characteristics	Old-classic	New-SoS
Scope of system	Fixed (known)	Not known
Specification	Fixed	Changing
Control	Central	Distributed
Evolution	Version controlled	Uncoordinated
Testing	Test phases	Continuous
Faults	Exceptional	Normal
Technology	Given and fixed	Uncertain
Emergence	Controller	Accidental
System development	Process model	Undefined

Apart from economic, societal and educational factors, in this transition the following technological challenges have been highlighted:

- Multidisciplinary approach (common language).
- System modeling, simulation (and verification).
- Emergent behavior.
- Methods, architectures, platforms and theory.
- Standards and requirements.

Our contention is that Multi-Agent Systems (MAS), a computing paradigm for developing intelligent systems, can be a credible instrument in addressing some of these challenges. In the rest of the paper we describe the main characteristics of software agents and of MAS and rationalize why MAS technology can be used to develop SoS. It must be noticed that the multi-agent approach we propose is understood as a means of realizing the notions of intelligent object and of system play, deviations from standard notions of system composition introduced in [1][2].

II. SOFTWARE AGENTS

Artificial Intelligence (AI) has evolved from the ideal of perfect automatic reasoning, typically in the form of axiomatic systems, which from a set of premises prove theorems by applying deductive rules, to building systems that display acceptable behavior.

From Aristotle's syllogisms, through Leibniz's universal language, Babbage's difference engine and Boole's "laws of thought" to Turing's notion of computability, an important effort has been invested in formalizing "intelligence" and in building (abstract and physical) machines to mechanize it. Early AI inherited the goal and the methods from this trend of research, and thus was devoted to develop theorem provers for mathematical reasoning.

It became soon clear however that if we were to implement systems that exhibited intelligence in real-life situations a shift of paradigm was needed. Logical systems typically assume perfect knowledge of an unchanging set of truths. As a consequence, notwithstanding the success of some expert systems, the original AI promise –to develop systems that showed general intelligence– was not fulfilled, resulting in the so-called AI Winter. Partly as a response to this situation, attention swung to "weak" AI. This new theory pivoted around the idea of "embeddedness". Intelligence was not considered as thinking logically in closed domains, rather it was an emergent property of systems situated in open environments, environments that imposed constraints on the system. The concept of intelligence moved from thinking to acting, from perfect rationality to bounded rationally, from heavy-weight logical systems to networks of light-weight reactive systems.

This alternative didn't survive either. However interesting their results may be (for instance, in the simulation of swarm intelligence) relying exclusively on the emergent behavior of loosely coupled simple systems poses serious methodological problems.

For the last couple of decades researchers have experienced the advent of new technologies such as the Internet. These demand personal, continuously running systems for which older notions of action –those resulting from either cumbersome symbolic reasoning or ever-adaptive reflexes– may be insufficient. Indeed, many researchers believe that in the XXI century for AI systems to perform "intelligently" they must be able to behave in an *autonomous*, *flexible* manner in unpredictable, dynamic, typically *social* domains. In other words, they believe that the "new" AI should develop *agents* [5] [6].

The concept of agent serves to represent the idea of a autonomous system that perceives the environment and acts on it. The agent has an internal state that represents their knowledge and their goal –typically, the maximization of their own utility function. Decisions on which actions to execute depend on the agent's internal state and on the characteristics of the environment in which they are embedded. In the next section we explore the three main characteristics of agency.

A. Autonomy

By autonomy we mean the ability of the system to make their own decisions and execute tasks on the designer's behalf. The idea of delegating some responsibility to the system is essential in scenarios where it is difficult to control directly the behavior of our systems. For example, space missions increasingly depend on their unmanned spaceships and robots to make decisions on their own.

It is precisely their autonomy the characteristic that uniquely defines agents. Traditionally, software systems execute actions (so-called methods) automatically: imagine that the Web application in your computer, the user or client, requests to access the contents of a webpage that is stored in another software system elsewhere, the server or host. The server cannot deny access to the content of the webpage; it must execute the "send" method whenever it is requested to do so. On the contrary, agents decide by themselves whether to execute their methods according to their own goals. Paraphrasing [7], "what traditional software systems do for free, agents do for money".

B. Adaptive behavior

Secondly, agents must be flexible. When designing agent systems, it is impossible to foresee all the potential situations they may encounter and specify their behavior optimally in advance. For example, the components of interaction in the Internet (agents, protocols, languages) are not known *a priori*. Agents therefore have to learn from and adapt to their environment. This task is even

 ISBN: 978-1-61208-246-2

more complex when Nature is not the only source of uncertainty, when the agent is situated in a multi-agent system (MAS) that contains other agents with potentially different capabilities, goals, and beliefs.

In addition, an agent must have the competence to display an action repertoire general enough to preserve its autonomy in dynamic environments. Certainly, an agent can hardly be called intelligent if it is not able to perform well when situated in an environment different from (yet in some way similar to) the one it was originally designed for.

Indeed, there is no need to learn anything in static, deterministic, fully observable domains where agents have perfect knowledge of state-action transitions. Nonetheless, intelligence and learning are tightly tied in environments where systems must make decisions with partial or uncertain information, that is, in domains where they must learn without supervision and without the luxury of having a complete model of the world –when facing the so-called *reinforcement learning problem* [8], in which the learner must discover which actions yield the most reward by exploiting and exploring their relationship with the environment.

C. MAS co-ordination

Agents also show social attitudes. In an environment populated by heterogeneous entities, agents would need the ability to recognize their opponents, and to form alliances when it is profitable to do so. It is not a coincidence that most agent-based platforms incorporate multi-agent tools [9]. Indeed, it is claimed that agent-oriented software engineering needs to be developed precisely because there is no notion of organizational structure in traditional software systems.

Generally speaking, the design and implementation of MAS is an attractive platform for the convergence of various AI technologies. That is the underlying philosophy of competitions such as RoboCup (http://www.robocup.org/) where teams of soccer agents must display their individual and collective skills in real-time. More importantly, multi-agent systems play several roles in IT and telecoms: for clients, they provide personalized, user-friendly interfaces; as middleware, they have been used extensively to implement electronic markets and electronic auctions.

The reasons for this happy marriage between MAS and new technologies are various. When the domain involves a number of distinct software systems that are physically or logically distributed (in terms of their data, expertise or resources), a multi-agent approach can often provide an effective solution. Relatedly, when the domain is large, sophisticated, or unpredictable, the overall problem can indeed be partitioned into a number of smaller and simpler components, which are easier to develop and maintain, and which are specialized at solving the constituent problems. That is, in most real-life applications (single) agents can grow "too big" to work well, and a *divide and conquer* strategy, where qualified agents work in parallel, seems more sensible. Examples include the geographical distribution of cameras in a traffic network or the integrated approach required to solve complex tasks, for instance collaboration between experts (surgeons, anesthetists, nurses) in an operating room.

To sum it up, it is widely accepted within the AI community that the "new" AI is about designing and implementing MAS capable of acting and learning in a quick and efficient manner. This affects MAS co-ordination and MAS learning.

Approaches to multi-agent behavior differ mainly in regards to the degree of control that the designer should have over individual agents and over the social environment, *i.e.*, over the interaction mechanisms [10]. In Distributed Problem Solving systems (DPS) a single designer is able to control (or even explicitly design) each individual agent in the domain –the task of solving a problem is distributed among different agents, hence the name. In MAS on the other hand, there are multiple designers and each is able to design only its agent and has no control over the internal design of other agents [11][12].

The design of interaction protocols is also tightly coupled to the issue of agents' incentives. When agents are centrally designed they are assumed to have a common general goal. As long as agents have to co-exist and cooperate in a single system, there is some notion of global utility that each agent is trying to maximize. Agents form teams that jointly contribute towards the overall goal. By contrast, in MAS each agent will be individually motivated to achieve its own goal and to maximize its own utility. As a result, no assumptions can be made about agents working together cooperatively. On the contrary, agents will collaborate only when they can benefit from that cooperation.

Research in DPS considers how work involved in solving a problem can be divided among several nodes so as to enhance the system's performance, that is, the aim is to make independent nodes solve a global problem by working together coherently, while maintaining low levels of communication. MAS researchers are also concerned with the coherence of interaction, but must build agents without knowing how their opponents have been designed. The central research issue in MAS is how to have these autonomous agents identify common ground for cooperation, and choose and perform coherent actions.

In MAS systems, agents typically make pair-wise agreements through negotiation about how they will co-ordinate, and there is no global control nor consistent knowledge nor shared goals or success criteria. So, the main purpose of this *incentive contracting* mechanism is to "convince" agents to reach reasonable agreements and do something in exchange for something else. In this case, AI researchers have followed the studies on bargaining with incomplete information developed in economics and game theory.

Using such approach agents are considered players that execute moves following a strategy. At the end of the game each agent receives a pay-off or return. The strategies the agents follow are typically modulated by their attitude towards risk, that is, whether they are risk-

 ISBN: 978-1-61208-246-2

averse (in which case following a minimax strategy where the agent tries to minimize the other's profit may work) or they tolerate risk. The strategies also vary according to the nature of the interaction (one-shot or continuous; simultaneous or sequential) and to the type of game itself, for instance, if the agents engage in a zero-sum game (where what one gains the other loses) or in a cooperative game (where there may exist win-win solutions). In any case, the interaction takes the form of a negotiation protocol. Negotiation is defined as a process through which in each temporal point one agent proposes an agreement from the negotiation set and the other agent either accepts the offer or does not. If the offer is accepted, then the negotiation ends with the implementation of the agreement. Otherwise, the second agent has to make a counteroffer, or reject its opponent's offer and abandon the process. So, the protocol specifies when and how to exchange offers (*i.e.*, which actions the agents will execute or abstain from executing and when) – for example, an Offer(x, y, δ_i, t_1) means that the negotiation process will start at time t_1 with agent x offering agent y a deal δ_i from the set of potential deals ($\delta_i \in \Delta$), typically of the from "I will do action 1 in exchange for action 2" or {Do(x, a_1), Do(y, a_2)}; then, in the next negotiation step, agent y will counteroffer with Accept(y, δ_i, t_2), in which case the negotiation episode ends with the implementation of the agreement, δ_i; or with Reject(y, δ_i, t_2), so that negotiation fails; or, alternatively, agent y can send a counteroffer, Offer(y, x, δ_j, t_2), with say δ_j = {Do(x, a_3), Do(y, a_2)}, "I would prefer you to execute action a_3 rather than a_1", so that negotiation progresses to the next stage in which the same routine applies.

As discussed above, intelligence implies a certain degree of autonomy in decision-making that in turn requires the ability to learn to make independent decisions in dynamic, unpredictable domains such as those in which agents co-exist.

The simplest way to extend single-agent learning algorithms to multi-agent problems is just to make each agent learn independently. Agents learn "as if they were alone". Communication or explicit co-ordination is not an issue therefore –co-operation and competition are not tasks to be solved but properties of the environment. Likewise, agents do not have models of other agents' mental states or try to build models of other agents' behaviors. However simple this approach to multi-agent learning may be, the assumption that agents can learn efficient policies in a MAS setting independently of the actions selected by other agents is implausible. Intuitively, the most appealing alternative is to have the agents learn Nash-equilibrium strategies [13][14][15].

IV. MAS TECHNOLOGY FOR SoS

We can conclude from the conceptual analysis presented above that the paradigm shift demanded by SoS requirements fits new trends in engineering computing systems. In particular, the drivers and characteristics of SoS as specified in section I are consistent with the notions of autonomous agents and of multi-agent systems as opposed to traditional objects and classes. Although MAS development still relies on object-oriented tools and techniques, it is a fact that agent-oriented engineering is the way forward in the era of large, complex, loosely connected software systems, that is, in the era of the Internet –indeed, the Open Systems Interconnection (OSI) protocol itself can be understood as a SoS. More specifically, it can be argued that the difference between CoS and SoS lies in the fact that the former are collections of objects that coordinate their behavior via DPS, whereas the latter are collections of agents that interact through incentive mechanisms such as negotiation and argumentation. In Table 3 we enumerate ontologies, architectures, methodologies, languages, platforms, infrastructures and validation tools that SoS can borrow from MAS.

TABLE 3. AGENT-ORIENTED SOFTWARE ENGINEERING (AOSE)

AOSE	**Standards, techniques and tools**
Ontologies	RDF Schema, OIL, DAML, OWL, SHOE
Architectures	BDI, InteRRAP, Touring Machines
Methodologies	Tropos, MAS-CommonKADS, PASSI, Prometheus, Gaia, ADELFE, MESSAGE
Design languages	Agent UML
Programming languages	AOP: AGENT0, PLACA, Agent-K, MetaM, April, MAIL, VIVA, GO!
	BDI: AgentSpeak, Jason, AF-APL, JACK, JADEX, 3APL
	GOAL, Golog, FLUX, CLAIM
Communication languages	KQML, FIPA, ARCOL, KIF, COOL
Coordination mechanisms	MAP, Negotiation, Argumentation, Auctions, Institutions
Tools and platforms	ZEUS, JADE, agentTool, RETSINA, JATLite, MADKIT, JAFMAS, Cougaar
Infrastructures	Jini, Ontolingua ReTAX++, OilEd
Validation	Deductive verification, model checking

In particular, perhaps the most defining characteristic of MAS is the fact that communication is understood and formalized in terms of the internal states of the agents involved. In fact, communication is understood as (speech) acts, as actions that agents execute in order to achieve their goals not as mere message passing. In addition, since the agent at the other end is also autonomous, the sender needs to consider the receiver's internal state (not just its "position" in a mailing queue) and its own intentions. For instance, unlike in object-oriented approaches, when sending a request, the sender x holds the goal of a receiver y achieving a particular proposition P, that is, of making P true. Moreover, as ix wants the receiver to really try to achieve P the preconditions also require that y intends along a run that P be eventually true. Finally, the rational effect to be achieved is that there is a run in which P eventually holds. Formally,

[x, request(y, P)]
FP: G(x, (I(y, FP)))
RE: FP

Typically, Agent Communication Languages (ACLs) come with a complete set of speech acts, including

"inform", "confirm" "agree", "promise", "disconfirm", "refuse", "declare", and of course, "request".

Of course, agents and MAS are a particular type of SoS. Agents and MAS are *software* SoS. A general theory of SoS must be more comprehensive and accommodate the characteristics of other types of systems, characteristics that cannot be reduced to conglomerates of computing devices and the way they inter-operate. Our contention is nevertheless that MAS can play the same role in developing SoS as objects and classes played in conceptualizing, formalizing and implementing CoS. In addition, and more importantly, the individual systems MASs consist of are autonomous and adaptive –the two defining properties of the systems in a SoS. In fact, MAS emerge recursively and hierarchically as a result of the free interaction of such systems and multi-agent systems – as SoS do.

REFERENCES

[1] N. Karcanias and A.G. Hessami, 2010. Complexity and the notion of Systems of Systems: Part (I) General Systems and Complexity. Proc. of the 2010 World Automation Congress International Symposium on Intelligent Automation and Control (ISIAC) 19-23 September 2010, Kobe Japan.

[2] N. Karcanias and A.G. Hessami, 2010. Complexity and the notion of Systems of Systems: Part (II) Defining the notion of Systems of Systems. Proc. of the 2010 World Automation Congress International Symposium on Intelligent Automation and Control (ISIAC) 19-23 September 2010, Kobe Japan.

[3] N. Karcanias and A.G. Hessami, 2011. System of Systems Emergence: Part (I) Principles and Framework. Proc. of the ICETET 2011, 4th Internationa Cnference on Emerging Trends in Engineering and Technology, SV129, November 18-20, Port Louis, Mauritius.

[4] N. Karcanias and A.G. Hessami, 2011. System of Systems Emergence: Part (II) Synergetics Effects and Emergence. Proc. of the ICETET 2011, 4th Internationa Cnference on Emerging Trends in Engineering and Technology, SV129, November 18-20, Port Louis, Mauritius.

[5] E. Alonso, 2002. AI and Agents: State of the Art, AI Magazine, 23 (3), 529–551.

[6] E. Alonso, 2012. Actions and Agents. In K. Frankish and W. Ramsey (Eds.), The Cambridge Handbook of Artificial Intelligence, Chapter 5. Cambridge, England: Cambridge University Press.

[7] N. Jennings and M. Wooldridge, (eds.), 1998. Agent Technology: Foundations, Applications, and Markets. Berlin: Springer-Verlag.

[8] E. Alonso, M. d'Inverno, D. Kudenko, M. Luck and J. Noble, 2001. Learning in Multi-Agent Systems, Knowledge Engineering Review 16 (3), 277-284.

[9] M. Luck, P. McBurney, O. Shehory and S. Willmott (eds.), 2005. Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). AgentLink III.

[10] M. Wooldridge, 2009. An Introduction to MultiAgent Systems, 2nd edition. Chichester, England: John Wiley & Sons.

[11] A. H. Bond and L. Gasser, Less (eds.), 1988. Readings in Distributed Artificial Intelligence. San Mateo, CA: Morgan Kaufmann Publishers.

[12] E. H. Durfee, 1988. Coordination for Distributed Problem Solvers. Boston: MA: Kluwer Academic.

[13] L. P. Kaelbling, M. Littman and A. Moore, 1996. Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research 4: 237-285.

[14] G. Weiss, (ed.), 1999. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA: The MIT Press.

[15] G. Weiss and P. Dillenbourg, 1999. What is 'multi' in multiagent learning?, in Pierre Dillenbourg (ed.), Collaborative learning. Cognitive and computational approaches (pp. 64–80), Oxford: Pergamon Press.

 ISBN: 978-1-61208-246-2