AI and Agents
State of the Art

Eduardo Alonso

■ This article is a reflection on agent-based AI. My contention is that AI research should focus on interactive, autonomous systems, that is, agents. Emergent technologies demand so. We see how recent developments in (multi-) agent-oriented research have taken us closer to the original AI goal, namely, to build intelligent systems of general competence. Agents are not the panacea though. I point out several areas such as design description, implementation, reusability, and security that must be developed before agents are universally accepted as the AI of the future.

Now that we have entered the new millennium, it is time to review where we are and where we are going in AI. Although substantial progress has been made in the development and application of AI technology, much remains to be done. As Nils J. Nilsson (1995, p. 11) stated:

> We have to distinguish between intelligent programs and the specialised systems, that is, the tools, that they use. No doubt, building the tools is important. But working on the tools alone has not moved us closer to AI's original goal.

The most persistent and troubling of all the criticisms, to paraphrase Michael Wooldridge (1998), is that AI has simply failed to deliver on its basic promise, that is, to build intelligent systems of general competence. Thus, there is a feeling that after more than 50 years of AI research,[1] it is time to go back to the Good Old-Fashioned Artificial Intelligence (GOFAI).

Times have changed though, and the new GOFAI cannot stick with so-called traditional software. Emergent Technologies such as the internet demand personal, continuously running, autonomous systems.

Like many AI authors (see Russell and Norvig [1995] and Huhns and Singh [1998]), I believe that for AI systems to perform successfully, they must be able to behave in an autonomous, flexible manner in unpredictable, dynamic, typically social domains. In other words, the new GOFAI should develop agents.

By *autonomy,* I mean the ability of the systems to make their own decisions and execute tasks on the designer's behalf. Delegating some responsibilities to the system and avoiding the tedious task of writing down the corresponding code is certainly attractive. Moreover, in unknown scenarios where it is difficult to control directly the behavior of our systems, the ability of acting autonomously is essential (for example, the National Aeronautics and Space Administration's [NASA] RAX spaceship controller).[2]

It is precisely their autonomy that defines agents. Even though it is possible to encapsulate some behaviors by specifying private methods in object-oriented programming, the system cannot deny access to a method that has been declared ''public,'' which means that the object must execute the method any time it is requested to do so. On the contrary, agents must decide by themselves whether to execute their methods according to their goals (agents must be proactive), preferences, and beliefs. As it is has been said, ''what objects do for free, agents do for money'' (Wooldridge 1999). Agents typically operate according to a remote programming paradigm, or computer-to-computer programming. *Remote programming* enables the client (user) to store on the server (host) not only the procedures but also any accompanying instructions and peripheral data. Each time the events specified in the instructions occur, the server calls the procedures and executes the instructions on site,

without any further intervention from the client computer.

Moreover, agents must be flexible. When designing agent systems, it is impossible to foresee all the potential situations an agent might encounter and specify an agent behavior optimally in advance. Agents therefore have to learn from, and adapt to, their environment. This task is even more complex when nature is not the only source of uncertainty, and the agent is situated in an environment that contains other agents with potentially different capabilities, goals, and beliefs. For example, the components of interaction in the internet (potentially adversarial agents, protocols, and languages) are not known a priori.

An agent must also show a social attitude. In an environment populated by heterogeneous entities, agents must have the ability to recognize their opponents and form groups when it is profitable to do so.

Of course, in restricted domains, there is no need to learn anything. However, in such cases, there is also no need to design agents. An object-oriented approach would be more appropriate. However, intelligence and learning are closely tied in uncertain domains where autonomous agents must postpone making decisions until relevant information has been acquired. Not surprisingly, learning has recently received increasing attention in connection with agents and multiagent systems (Alonso et al. 2001; Kudenko and Alonso 2001; Sen 1998; Weiss 1999, 1997). Alternatively, novel planning algorithms such as GRAPHPLAN (Blum and Furst 1997) have been extended for conditional planning (Anderson, Smith, and Weld 1998), contingent planning (Weld, Anderson, and Smith 1998), temporal planning (Smith and Weld 1999), and probabilistic planning (Blum and Langford 1999) in an attempt to handle uncertainty. It is too early, though, to say which planning algorithm will dominate.[3]

An agent must also show a social attitude. In an environment populated by heterogeneous entities, agents must have the ability to recognize their opponents and form groups when it is profitable to do so. Developing agent teams has been a topic of intensive research in the agent community (Grosz and Kraus 1999; Tambe et al. 1999). It is not a coincidence that most agent-based platforms (such as AGENTBUILDER,[4] JACK,[5] MADKIT,[6] and ZEUS[7]) incorporate multiagent tools. Some authors (for example, Zambonelli et al. [2000]) do state that agent-oriented software engineering needs to be developed precisely because there is no first class notion of organizational structure (the very essence of multiagent systems) in the object-oriented world.

The new systems must also be general. An agent must have the competence to display an action repertoire general enough to preserve its autonomy in dynamic environments. Again, it becomes clear that the ability to learn and adapt is one of, if not the most important feature of intelligence. An agent can hardly be called intelligent if it is not eventually able to perform well when you put it into an environment different from the one it was initially designed for.

Finally, the use of agent technology to build intelligent systems has the following additional advantages:

Agents are a natural metaphor to understand and use intelligent systems. It is intuitive to model our systems as we think of ourselves, that is, as human agents with beliefs, desires, and intentions (BDI [Rao and Georgeff 1991]). Agents thus fit perfectly in the ''user-friendly'' paradigm. Apart from executing actions on the user's behalf, agents can also assist users teaching or training them, helping different users collaborate, and monitoring events and procedures.

Artificial and virtual pets (for example, AIBO,[8] ASIMO,[9] CREATURES[10]) and affective and believable characters (such as STEVE [Johnson et al. 1998]) are examples of agent-based applications.

More importantly, agents are regarded as the technology in which the information technology and telecommunications sectors should converge. They have a role to play at the client side of such systems, providing customers with personalized, proactive interfaces to new services and products. They have a role to play as middleware, putting users in contact with the goods and services that best suit their needs. In addition, they have a role to play as servers, cooperating and negotiating on behalf of organizations and other end users. Moreover, as Nick R. Jennings put it, ''Electronic Commerce is the most important application for Agent Technologies because it is reality-based and constitutes a massive market'' (Noriega and Sierra 1999). Not surprisingly, agents are being extensively used to implement electronic markets and electronic auctions (for example, KASBAH [Chavez and Maes 1996], BAZAAR [Zeng and Sycara 1997], MAGMA [Tsvetovatyy et al. 1997], MAGNET,[11] FISHMARKET).[12] Moreover, the Trading Agent Competition (TAC)[13] provides a platform to test electronic-commerce applications.

Generally speaking, multiagent systems are an attractive platform for the convergence of various AI technologies. Such is the underlying philosophy of the RoboCup competitions (for both robots and simulators), where teams of agents must display their individual and collective skills in real time (Asada et al. 1999). Even though it is hard to find a practical application for artificial soccer other than pure entertainment, the idea of using such a popu-

lar spectacle as a demonstration of AI systems has been very successful.[14]

When the domain involves a number of distinct problem-solving entities (or data sources) that are physically or logically distributed (in terms of their data, expertise, or resources), an agent-based approach can often provide an effective solution. The (Computational) Grid Forum is an ideal scenario to prove the suitability of agents to solve inherently distributed problems.[15,16]

When the domain is so large, sophisticated, or unpredictable, the overall problem can indeed be partitioned into a number of smaller and simpler components that are easier to develop, and maintain, and that are specialists at solving the constituent problems. It has been shown that distributing problem solving over several levels of abstraction reduces the time complexity of the planning process (Korf 1987; Montgomery and Durfee 1993). Moreover, in a hierarchical multiagent system, the learning module can be applied within each abstract problem space. Because the abstract problem spaces define more general and, thus, simpler problems and theories, agents learn more efficient and appropriate rules (Knoblock, Minton, and Etzioni 1991).

No doubt, our defense of agent-based technology as the new revolution in AI might seem out of date. After all, agent research and technology groups have been established by such companies as Microsoft, IBM, Sun Microsystems, AT&T, and Netscape. Besides, agent-based applications have been reported in manufacturing, process control, telecommunications systems, air traffic control, traffic and transportation management, information filtering and gathering, electronic commerce, business process management, entertainment, and medical care (Jennings and Wooldridge 1998). Nonetheless, one of the key problems of recent years has been the divide between theoretical work in agent-based systems and its practical complement that have, to a large extent, developed along different paths. As a consequence, designers lack a systematic methodology for clearly specifying and structuring their applications as multiagent systems. Most agent-based applications have been designed in an ad hoc manner—"either by borrowing a methodology (typically an object-oriented one) and trying to shoehorn it to the multiagent context or by working without a methodology and designing the system based on intuition and past experience" (Jennings, Sycara, and Wooldridge 1998, p. 31).

In particular, the following areas must be developed:

Design description: Conceptual modeling and specification languages are used to describe a system design, that is, a unified agent modeling language (UAML). For example, there is no universally accepted definition of the term *agent.* Any control system of a UNIX demon can be viewed as an agent.

Properties and analysis: Methods and techniques are needed to specify, establish, and verify properties of an agent system.

Since the mid-1980s, problems with symbolic reasoning led to so-called reactive agents. As Michael Wooldridge (1999) put it, "A major point of such systems is that the overall behaviour emerges from the interaction of the component behaviours when the agent is placed in its environment. There are obvious advantages to reactive approaches: simplicity, economy, computational tractability, and robustness against failure. However, such systems make it very hard to engineer agents to fulfill specific tasks. Ultimately, there is no principled methodology for building such agents: One must use a laborious process of experimentation, trial and error to engineer an agent."

From a user point of view, the agent must exhibit somewhat predictable behavior and provide some sort of explanation for its actions. Designers then need a formal semantics to verify formally that a given behavior conforms to the specification.

From a user point of view, the agent must exhibit somewhat predictable behavior and provide some sort of explanation for its actions. Designers then need a formal semantics to verify formally that a given behavior conforms to the specification.

Implementation: Agent- and market-oriented programming languages, as well as standard coordination protocols and agent communication languages (ACLs), need to be implemented.

Although some object-oriented features such as abstraction, inheritance, and modularity make it easier to manage increasingly more complex systems, JAVA (or its distributed extensions JINI and RMI) and other object-oriented programming languages cannot provide a direct solution to agent development. As for ''real'' agent-oriented programming, AGENT-0 (Shoham 1993) and the like (PLACA [Thomas 1993], AGENT-K [Davies and Edwards 1994], ELEPHANT 2000 [McCarthy 1990]),[17] are still small and simple languages, used mainly to test ideas about agent-oriented programming rather than to develop any realistic systems.

Another issue to address in a multiagent scenario is interoperability. The debate should not be focused exclusively on the pros and cons of different languages and protocols (FIPA,[18] KQML,[19] CORBA[20]) but also on ontologies. Currently, ontologies are often specified informally or implicitly in the agent implementation. For true interoperation, agents will need explicitly encoded, sharable ontologies.

Reusability: Methods and techniques to specify and maintain reusable models and reusable software for multiagent systems, agents, and agent components. It was made abundantly clear in AgentLink2's last meeting (Amsterdam, December 2001) that users do not like new, expensive, difficult-to-learn systems.[21] They would rather reuse their old systems and add new technology to them.

Reusability also refers to mobility. Obviously, we want our agents to roam wide area networks such as the World Wide Web. That is, we want to (re)use them continuously in different scenarios. Once again, for this ideal to be realized, standard languages and protocols are needed.

Security: For individuals to be comfortable with the idea of delegating tasks to agents, they must first trust them. Issues that need to be addressed include authentication, privacy of communication and user's personal profile information, trust, auditing, accountability, and defense against malicious or incompetent agents (see Castelfranchi et al. [2000] for an overview on trust and agents).

I focused this article on software agents, or softbots. I would also like to mention the work that has been done in robots. The bad news is that (Menzel and D'Aluisio 2000) despite 25 years of intensive research aimed at the development of a robotics science, the statistics of actual robot use in industry have remained essentially constant: More than 90 percent of industrial robots are used for spot welding and spray painting.

Hopefully, the introduction of qualitative analyses in vision (active or purposive vision [Bianchi and Rillo 1996]), the use of fuzzy logics for navigation (THINKING CAP [Parsons et al. 1999]), and the development of behavior-based robotics and hybrid architectures (Arkin 1999) will change this situation in the near future.

Notes

1. For Americans, AI started off at the Massachusetts Institute of Technology in 1957 with Marvin Minsky's and John McCarthy's Artificial Intelligence Project. Europeans, however, cite Alan Turing's (1950) article in Mind as the origin of AI.

2. rax.arc.nasa.gov.

3. See AIPS-00 Planning Competition, www.cs.toronto.edu/aips2000/.

4. www.agentbuilder.com/.

5. www.agent-software.com.au/.

6. www.madkit.org/

7. www.labs.bt.com/projects/agents.htm/

8. www.jp.aibo.com.

9. world.honda.com/ASIMO/.

10. www.cyberlife.co.uk.

11. www-users.cs.umn.edu/~gini/magnet.html.

12. www.iiia.csic.es/ Projects/fishmarket/newindex.html.

13. auction2.eecs.umich.edu/

14. www.robocup.org

15. www.gridforum.org.

16. www.cs.cf.ac.uk/User/O.F.Rana/agent-grid-2002/

17. McCarthy, J. 1990. elephant 2000: A Programming Language Based on Speech Acts. Unpublished Manuscript.

18. www.fipa.org.

19. www.cs.umbc.edu/kqml.

20. www.corba.org.

21. www.agentlink.org.

References

Alonso, E.; D'inverno, M.; Kudenko, D.; Luck, M.; and Noble, J. 2001. Learning in Agents and Multi-Agent Systems. *Knowledge Engineering Review* 16(3): 277–284.

Anderson, C.; Smith, D.; and Weld, D. 1998. Conditional Effects in GRAPHPLAN. Paper presented at the Fourth Conference on AI Planning Systems, 7–10 June, Pittsburgh, Pennsylvania.

Arkin, R. C. 1999. *Behavior-Based Robotics*. Cambridge, Mass.: The MIT Press.

Asada, M.; Kitano, H.; Noda, I.; and Veloso, M. 1999. RoboCup: Today and Tomorrow—-What We Have Learned. *Artificial Intelligence* 110:193–214.

Bianchi, R., and Rillo, A. 1996. A Distributed Control Architecture for a Purposive Computer Vision System. In Proceedings of the Second Symposium on Intelligence in Automation and Robotics, 288–294. Washington, D.C.: IEEE Computer Society.

Blum, A. L., and Furst, M. L. 1997. Fast Planning through Planning Graph Analysis. *Artificial Intelligence* 90:281–300.

Blum, D., and Langford, J. 1999. Probabilistic Planning in the GRAPHPLAN Framework. Paper presented at the Fifth European Conference on Planning, 8–9 September, Durham, United Kingdom.

Castelfranchi, C.; Falcone, R.; Firozabadi, B.; and Tan, Y. H. 2000. Guest Editorial. *Applied Artificial Intelligence* 14(9): 863–865.

Chavez, A., and Maes, P. 1996. KASBAH: An Agent Marketplace for Buying and Selling Goods. Paper presented at the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, 22–23 April, London, United Kingdom.

Davies, W., and Edwards, P. 1994. AGENT-K: An Integration of AOP and KQML. Paper presented at the CIKM Workshop on Intelligent Information Agents, 2 December, Gaithersburg, Maryland.

Grosz, B. J., and Kraus, S. 1999. The Evolution of Shared Plans. In *Foundations of Rational Agency,* eds. M. Wooldridge and A. Rao, 227–262. Norwell, Mass.: Kluwer Academic.

Huhns, M. N., and Singh, M. P., eds. 1998. *Readings*

in Agents. San Francisco, Calif.: Morgan Kaufmann.

Jennings, N. R., and Wooldridge, M. J., eds. 1998. *Agent Technology: Foundations, Applications, and Markets*. Berlin: Springer.

Jennings, N. R.; Sycara, K.; and Wooldridge, M. 1998. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems* 1(1): 7–38.

Johnson, W.; Rickel, J.; Stiles, R.; and Munro, A. 1998. Integrating Pedagogical Agents into Virtual Environments. *Presence* (5)2: 523–546.

Knoblock, C. A.; Minton, S.; and Etzioni, O. 1991. Integrating Abstraction and Explanation-Based Learning in PRODIGY. In Proeedings of the Ninth National Conference on Artificial Intelligence, 93–102. Menlo Park, Calif.: American Association for Artificial Intelligence.

Korf, R. E. 1987. Planning as Search: A Qualitative Approach. *Artificial Intelligence* 33(1): 65–88.

Kudenko, D., and Alonso, E., eds. 2001. Proceedings of the AISB-01 Symposium on Adaptive Agents and Multi-Agent Systems. York, U.K.: Society for the Study of Artificial Intelligence and the Simulation of Behavior.

Menzel, P., and D'aluisio, F. 2000. *Robo Sapiens: Evolution of a New Species*. Cambridge, Mass.: MIT Press.

Montgomery, T. A., and Durfee, E. H. 1993. Search Reduction in Hierarchical Distributed Problem Solving. *Group Decision and Negotiation* 2:301–317.

Nilsson, N. 1995. Eye on the Prize. *AI Magazine* 16(2): 9–17.

Noriega, P., and Sierra, C., eds. 1999. *Proceedings of the First International Workshop on Agent-Mediated Electronic Trading, AMET-98*. Lecture Notes in Artificial Intelligence 1571. Berlin: Springer.

Parsons, S.; Pettersson, O.; Saffiotti, A.; and Wooldridge, M. 1999. Robots with the Best of Intentions. In *Artificial Intelligence Today: Recent Trends and Developments,* eds. Michael J. Wooldridge and Manuela Veloso, 329–338. Lecture Notes in Artificial Intelligence 1600. Berlin: Springer.

Rao, A. S., and Georgeff, M. P. 1991. Modeling Rational Agents within a BDI Architecture. Technical Report 14, Australian Artificial Intelligence Institute.

Russell, S., and Norvig, P. 1995. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, N.J.: Prentice Hall.

Sen, S., ed. 1998. *International Journal of Human-Computer Studies* (Special Issue on Evolution and Learning in Multi-Agent Systems) 48(1).

Shoham, Y. 1993. Agent-Oriented Programming. *Artificial Intelligence* 60(1): 51–92.

Smith, D., and Weld, D. 1999. Temporal Planning with Mutual Exclusion. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Tambe, M.; Adibi, J.; Al-Onaizan, Y.; Erdem, A.; Kaminka, G. A.; Marsella, S. C.; and Muslea, I. 1999. Building Agent Teams Using an Explicit Teamwork Model and Learning. *Artificial Intelligence* 110:215–240.

Thomas, S. R. 1993. PLACA, an Agent-Oriented Programming Language. Ph.D. dissertation, Computer Science Department, Stanford University.

Tsvetovatyy, M.; Gini, M.; Mobasher, B.; and Wieckowski, Z. 1997. MAGMA: An Agent-Based Virtual Market for Electronic Commerce. *International Journal of Applied Artificial Intelligence* 11(6): 501–523.

Turing, A. M. 1950. Computing Machinery and Intelligence. *Mind* 59:433–460.

Weiss, G., ed. 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, Mass.: MIT Press.

Weiss, G., ed. 1997. *Distributed Artificial Intelligence Meets Machine Learning*. Berlin: Springer.

Weld, D.; Anderson, C.; and Smith, D. 1998. Extending GRAPHPLAN to Handle Uncertainty and Sensing Actions. In Proceedings of the Sixteenth National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence.

Wooldridge, M. J. 1999. Intelligent Agents. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence,* ed. G. Weiss, 27–77. Cambridge, Mass.: MIT Press.

Wooldridge, M. J. 1998. Agent-Based Software Engineering. *IEEE Proceedings on Software Engineering* 144(1): 26–37.

Zambonelli, F.; Jennings, N. R.; Omicini, A.; and Wooldridge, M. J. 2000. Agent-Oriented Software Engineering for Internet Applications. In *Coordination of Internet Agents: Models, Technologies, and Applications,* eds. A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, 127–141. Berlin: Springer-Verlag.

Zeng, D., and Sycara, K. 1997. How Can an Agent Learn to Negotiate? In Intelligent Agents III. Agent Theories, Architectures, and Languages. In *Proceeings of the ECAI-96 Workshop (ATAL)*. Berlin: Springer-Verlag.

Eduardo Alonso is a lecturer in computing at City University, London. He is also a member of The Society for the Study of Artificial Intelligence and the Simulation of Behavior (SSAISB) Committee, the largest AI Society in the United Kingdom. Alonso's research has focused on the development of coordination protocols and the implementation of knowledge-based learning techniques in multiagent scenarios, particularly in electronic-commerce applications. His e-mail addresses are eduardo@soi.city.ac.uk and eduardo.alonso@aisb .org.uk.