Future Directions for Agent-Based Software Engineering

Michael Winikoff

RMIT University, Australia and University of Otago, New Zealand
E-mail: michael.winikoff@otago.ac.nz

Abstract:

We briefly review the current state of play in the area of agent-based software engineering, and then consider "what next?". We discuss a range of marketing activities that will together help in making people from other communities aware of work in this area. We then outline a number of research topics that are seen as vital to the future of the field. Although (as always) more research is needed, recent progress in both research and industrial adoption has been most encouraging, and the future of agent-based software engineering looks bright.

Keywords: Agent-based software engineering, Position statement

Reference to this paper should be made as follows: M. Winikoff (2008) 'Future Directions for Agent-Based Software Engineering', Special section on Future of software engineering and multi-agent systems, International Journal of Agent-Oriented Software Engineering (IJAOSE).

Biographical Notes:

Michael Winikoff's research interests concern notations for specifying and constructing software. In particular, he is interested in agent oriented software engineering methodologies and is co-author of the book *Developing Intelligent Agent Systems: A Practical Guide*, published by John Wiley and sons in 2004.

1 INTRODUCTION

The world is a complex place where things change, sometimes in unexpected ways. For software to work effectively in the real world it must be able to adapt to changes, displaying characteristics such as flexibility, robustness, reactivity and proactiveness.

Agent-based computing aims to provide a means of conceptualising, designing, and realising software systems that exhibit these desired characteristics. Agent-based Software Engineering, more often referred to as Agent-Oriented Software Engineering (AOSE), is concerned with how to effectively engineer agent systems, that is, how to specify, design, implement, verify (including testing and debugging), and maintain agent systems.

This position statement briefly reviews progress over the past few years (section 2) and then considers the vital question: "what next?". In looking at where we should be focusing our efforts I consider separately so-called "marketing" or "out-reach" activities (section 3) and research activities (section 4).

2 THE PAST

Agent-based computing generally, and agent-based software engineering specifically, has come a long way in the past few years. After many years of work on agent-oriented software engineering methodologies, we are now in the position where there are a number of methodologies that are well-developed. These methodologies (e.g. MaSE (8) and its successor O-MaSE (7), Tropos (2), Gaia (41; 43), Prometheus (27) and Passi (5)) provide processes, notations and techniques that have been developed and refined over many years on the basis of experience. They also provide tool support, and have been used successfully by various people other than the methodology's creators.

Indeed, we are now at a point where analysis and design activities have been well studied, and there is emerging recognition that the field is approaching a level of maturity and convergence where standardisation, akin to UML, needs to be considered (28) (see also Scott DeLoach's paper in this issue).

Considering agent-based computing more broadly, the list of applications of agents is now quite impressive (e.g. (22; 23), and the AAMAS industry tracks), and the benefits of agents are starting to be measured (1), providing much-needed numbers that are much more convincing than the sorts of high-level and abstract arguments about near-decomposable systems (e.g. (19)) that were the best that we could do a few years ago.

3 THE FUTURE: MARKETING

The key question is this: how to make others aware of the work that we have been doing, so that they can use it and build on it where appropriate? By "others" here we mean both researchers in related fields, and software developers.

There is no easy answer, but a multi-pronged strategy needs to be used which includes:

1. Identifying the "value-add" of agent technology, and developing a clear and convincing "story".
2. Documenting case studies of agent-based solutions to real-problems.
3. Working to ensure that the work of the agent community is recognised in related fields.

It is important to develop a clear "story" that explains what agents are, and where and why they are useful (and also where are they not useful!). The diversity of the agent community is arguably a handicap here in that we do not present a coherent story to outsiders.

My belief is that for too long we have been trying to tell outsiders a story based on words such as "agents" and "autonomy" (e.g. (18; 25; 26)). This is, in my opinion, not a good approach because it begs the question "what is an agent?" (and results in a debate), and because "autonomy" can be a scary concept, evoking AI notions of rampant robots.

Instead, I think the story that we should be telling should focus on words such as "complexity", "adaptability", and "goals". Indeed, the argument presented by Georgeff (15) avoids the term "agent" completely, and focuses (in the context of service orchestration) on how goal-directed orchestration reduces coupling and increases robustness.

It is clearly more convincing if the story is supported by numerous case studies, showing how agent-based systems worked to solve real-world problems. Thus, collecting case studies, as has been done by AgentLink (23) and by the AAMAS industry track, is important in that it provides concrete examples of applications across a range of domains, and also because it helps to dispel the perception of agents as "esoteric AI". Perhaps even more convincing, but hard to obtain, are measurements that quantify the difference that agent-based solutions can make. For example, the analysis of Benfield *et al.* (1) which found significant benefits to developing software structured as BDI (Belief-Desire-Intention) agents.

There are a number of related fields, such as service-oriented computing (18), the grid (14) and autonomic computing (32), where the issues that have been tackled by researchers within the agents community are very relevant. Sadly, awareness in these communities of the existence of work in the agents community is sometimes lacking. Ghose's paper (in this issue) explores this issue in more detail, with particular focus on the service-oriented computing community. What we need to do here is to publish work in these areas' conferences and journals, and organise other means to link across the communities (e.g. discussion panels, joint events, invited talks).

Additionally, I believe it's important to continue to produce useful tools, not just papers (a point that is also made by Calisti in her paper in this issue). Suppose that ABC Software Company has decided that it makes sense for them to develop an agent-based solution. Having only published papers, and no tools, leaves them in the position of having to develop their own agent platform before they can even begin solving their problem (cf. the pitfalls of "you want your own agent architecture" and "You obsess on infrastructure" (42), where Wooldridge and Jennings noted that, back in 1999, "*... almost every multiagent system project that we have encountered has devoted significant resources to implementing this infrastructure from scratch.*"). Clearly, to be of use, tools need to be documented, supported, and reliable, not research prototypes.

Finally, it's essential to "close the loop" by using our tools and techniques "in anger" (38), i.e. in a real setting. Doing this not only gains credibility, but, more importantly, gives us feedback to guide further work. In particular, as researchers we often have to make simplifying assumptions in order to make progress. Some of these assumptions are reasonable, whereas others need to be relaxed in order for our work to be applicable in real settings. Only by applying our work "in anger" can we really find out (and often be surprised by!) which assumptions need to be relaxed.

4 THE FUTURE: RESEARCH DIRECTIONS

Broadly speaking, there are three aspects that have been foci for researchers in agent-based software engineering. The first is individual agents, and is characterised by questions around how to structure individual agents. A dominant approach is the so-called "cognitive" approach where agents are designed in terms of mentalistic notions such as beliefs, goals and plans. The argument for ascribing these notions to software agents (the "intentional stance" (9)) is that it provides a simpler way of understanding complex entities. The second aspect is interaction between agents, and is characterised by questions around how to obtain effective interaction and communication between agents. Topics of interest to agent interaction research include agent communication languages and interaction protocols. The third aspect is open societies of agents, and concerns how to ensure that an open society operates smoothly (e.g. using norms, institutions, and economic paradigms). Luck and McBurney (21) term these aspects "agent", "interaction" and "organization" (these concepts are similar, but not identical, to the "micro", "macro" and "meso" terminology of Zambonelli and Omicini (44)).

In this section we briefly outline three research areas that I believe are key for the future of the field. The first concerns research within the "agent" aspect, specifically, concerned with the foundational concept of goals. The second area is concerned with reconciling the different aspects, and integrating techniques, both between the three aspects, and between the cognitive and the emergent approaches to engineering agent systems. The third key area concerns obtaining assurance that an agent system will meet its requirements.

A fourth area for future work is extending the scope of AOSE methodologies beyond the so-called "classical" phases of the software life-cycle to encompass activities such as debugging (13; 29), testing (11; 16; 24; 45) and software maintenance and evolution (6).

4.1 Goals

Goals, achieved persistently and flexibly, are what give agents their adaptability. That is, goals are foundational to cognitive agents (they also turn out to be useful for requirements engineering (36)). Indeed, I believe that we will (eventually) look back and view the contribution of our work as being "goal oriented programming/design", rather than "autonomous agents".

However, perhaps surprisingly for a foundational concept, there is still scope for foundational work, including answering questions such as:

- What evidence is there that cognitive agents are a good approach, and for which types of problems are they suitable? (e.g. (1))
- What goal types are useful?
  Most work has focussed on so-called "achievement" goals, but many other goal types are both possible and useful (37).
- What (generic) mechanism can be added to agents to detect interactions between goals, in order to avoid conflict (e.g. resource contention) and exploit positive interactions ("synergy")? (e.g. (3; 17; 33; 34))
- How to make the process of designing agent systems more goal-oriented? (e.g. (20))

4.2 Integrating Approaches

There has been a significant amount of work on each of the three aspects discussed earlier (agent, interaction, and organization). However, much less work has been done on *integrating* these aspects and their associated approaches to developing agent systems. For example, although there has been much work on agent interaction (most of which has treated agents as black boxes), the resulting mechanisms for interaction are not always a good match with cognitive agents. Even where the match is reasonable, there is duplication of mechanism: (e.g.) goals and plans for (single/internal) agent design, and something else (e.g. social commitments) for designing agent interactions.

It can be argued that if we assume that our agents are cognitive and use goals, then it makes sense to try and build robust agent interactions on top of such agents. An example is teamwork (a specific for of interaction) which has been characterised in terms of cognitive agent concepts, specifically, joint-persistent goals (4). The challenge is to extend to forms of interaction other than teamwork. Some specific questions to be considered are:

- How to use flexible and robust goal achievement by single agents to enable multiple agents to achieve goals in a robust and flexible way? (e.g. (15))
- How does this compare with conventional approaches for choreography/orchestration, and with agent interaction approaches (e.g. protocols, social commitments)?
- How can we link agent interaction concepts with individual agent concepts? (e.g. (39)).

Going beyond interactions amongst a small number of agents, there has been much work on *societies* of agents, which has developed a range of concepts such as norms, institutions, organizations, and game theory. Again, there is duplication of mechanism with respect to both individual agents and agent interactions, and there is a lack of link with interaction and individual agents. Some initial work that looks at linking cognitive agents with agent organizations has been done by Tinnemeier *et al.* (35).

Another issue that needs to be resolved is reconciling the cognitive approach and the "emergent" approach (where a collection of simple agents exhibit useful, typically emergent, collective behaviour). These two approaches each have their own research communities, and interaction between them is somewhat limited. Bridging this gap involves looking at questions such as:

- How to ascribe goals to non-goal-based (and possibly emergent) behaviours? (e.g. (30))
- How to design and build hybrid systems that combine aspects of emergent and of cognitive agents? (e.g. (31))
- How to design goal-based agents in systems which exhibit emergent behaviour?

4.3 Validation & Verification

"*... systems exhibiting emergent behavior [are] ... the equivalent of a specification that says "surprise me"*." – Lou Mazzucchelli, in (26).

A recurring issue in practice (21; 23) is how to obtain confidence that an agent system will behave appropriately in a range of situations. Conventional testing is less effective for flexible adaptable software, since there are many more possible behaviours to be tested:

"*... validation through extensive tests was mandatory ... However, the task proved challenging for several reasons. First, agent-based systems explore realms of behaviour outside people's expectations and often yield surprises* ..." (23, Section 3.7.2).

Indeed, investigation of the behaviour space of BDI agents (40) has shown that:

- Even for relatively small goal-plan trees[a] the number of possible behaviours is very large.

[a]A goal-plan tree is a visualisation of a BDI agent's plans where each goal/event has as children the plan instances that are applicable to it, and each plan has as children its sub-goals.

- A significant contributor to the size of the "behaviour space" is the use of failure handling.

- There are more behaviours that result in failure than there are behaviours that result in the root-goal being successfully achieved. Furthermore, the probability of a failed behaviour occurring is low. Taken together, these two findings imply that testing failed executions is particular hard: there are many of them, and they are hard to "access".

An obvious answer to this problem is the use of some form of formal methods to systematically explore the execution space (10). However, much more research is needed to make this practical. In addition to all the usual challenges that arise when verifying complex concurrent systems, there are two additional challenges.

The first challenge is that typically we don't want to just check well-known properties such as liveness or safety, but also various forms of domain-specific correctness. Unfortunately, it is not always clear how to specify such properties. For instance, what does it mean for a manufacturing management and optimisation system (23, section 3) to be functioning correctly? It is easy to specify that all packages are (eventually) delivered. But how can we specify that the delivery schedule is "efficient"? There may also be situations where an efficient schedule cannot be obtained (e.g. if a key bridge is closed) — how can we specify that the system will do a reasonable job under the circumstances?

The second challenge is dealing with systems that are dynamic, including those whose structure may change. One possible approach is to use an assume-guarantee style of reasoning (10), another is to defer (some) reasoning and checking to runtime (12).

Just to reemphasize: what is needed is research that leads to practical and useful tools, and that (eventually) avoids making assumptions that renders it inapplicable to real systems. Also, what is needed is not the ability to prove a system correct, as much as the ability to find errors in the system, and to obtain assurance regarding the system's level of correctness, and the assumptions that need to hold for it to function correctly.

5 CONCLUSION

We briefly reviewed the past, and then discussed ideas for "marketing", and outlined key research directions, namely further work on the foundational concept of goals, work on integrating different aspects, techniques for verifying and validating agent systems, and extending the scope of AOSE methodologies. Although much work remains to be done, recent progress in both research and industrial adoption has been most encouraging, and the future of agent-based software engineering looks bright.

6 ACKNOWLEDGEMENT

I would like to thank Stephen Cranefield and Lin Padgham for their comments on drafts of this paper.

References

[1] Steve S. Benfield, Jim Hendrickson, and Daniel Galanti. Making a strong business case for multiagent technology. In Peter Stone and Gerhard Weiss, editors, *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 10–15. ACM Press, 2006.

[2] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8:203–236, May 2004.

[3] Bradley J. Clement and Edmund H. Durfee. Identifying and resolving conflicts among agents with hierarchical plans. In *AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities, Technical Report WS-99-12*, 1999.

[4] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[5] Massimo Cossentino. From requirements to code with the PASSI methodology. In Brian Henderson-Sellers and Paolo Giorgini, editors, *Agent-Oriented Methodologies*, pages 79–106. Idea Group Inc., 2005.

[6] Khanh Hoa Dam, Michael Winikoff, and Lin Padgham. An agent-oriented approach to change propagation in software evolution. In *Australian Software Engineering Conference*, pages 309–318. IEEE Computer Society, 2006.

[7] Scott A. DeLoach. Engineering organization-based multiagent systems. In Alessandro F. Garcia, Ricardo Choren, Carlos José Pereira de Lucena, Paolo Giorgini, Tom Holvoet, and Alexander B. Romanovsky, editors, *SELMAS*, volume 3914 of *Lecture Notes in Computer Science*, pages 109–125. Springer, 2005.

[8] Scott A. DeLoach, Mark F. Wood, and Clint H. Sparkman. Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):231–258, 2001.

[9] Daniel C. Dennett. *The Intentional Stance*. MIT Press, 1987.

[10] Matthew B. Dwyer, John Hatcliff, Corina Pasareanu, Robby, and Willem Visser. Formal software analysis: Emerging trends in software model checking. In *International Conference on Software Engineering: Future of Software Engineering*, pages 120–136, May 2007.

[11] Erdem Eser Ekinci, Ali Murat Tiryaki, and Övünç Çetin. Goal-oriented agent testing revisited. In Jorge J. Gomez-Sanz and Michael Luck, editors, *Ninth International Workshop on Agent-Oriented Software Engineering*, pages 85–96, 2008.

[12] M. Feather, S. Fickas, A. van Lamsweerde, and C. Ponsard. Reconciling system requirements and runtime behavior. In *Proceedings of IWSSD'98: 9th International Workshop on Software Specification and Design*. IEEE Computer Society Press, April 1998.

[13] David Flater. Debugging agent interactions: a case study. In *Proceedings of the 16th ACM Symposium on Applied Computing (SAC2001)*, pages 107–114. ACM Press, 2001.

[14] Ian T. Foster, Nicholas R. Jennings, and Carl Kesselman. Brain meets brawn: Why grid and agents need each other. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), 19-23 July 2004, New York, NY, USA*, pages 8–15. IEEE Computer Society, 2004.

[15] Michael Georgeff. Service orchestration: The next big challenge. *DM Review Special Report*, June 2006. `http://www.dmreview.com/specialreports/20060613/1056195-1.html`.

[16] Jorge J. Gomez-Sanz, Juan Botía, Emilio Serrano, and Juan Pavón. Testing and debugging of MAS interactions with INGENIAS. In Jorge J. Gomez-Sanz and Michael Luck, editors, *Ninth International Workshop on Agent-Oriented Software Engineering*, pages 133–144, 2008.

[17] John F. Horty and Martha E. Pollack. Evaluating new options in the context of existing plans. *Artificial Intelligence*, 127(2):199–220, 2001.

[18] Michael N. Huhns, Munindar P. Singh, Mark Burstein, Keith Decker, Edmund Durfee, Tim Finin, Les Gasser, Hrishikesh Goradia, Nick Jennings, Kiran Lakkaraju, Hideyaki Nakashima, H. Van Dyke Parunak, Jeffrey S. Rosenschein, Alicia Ruvinsky, Gita Sukthankar, Samarth Swarup, Katia Sycara, Milind Tambe, Tom Wagner, and Laura Zavala. Research directions for service-oriented multiagent systems. *IEEE Internet Computing*, 9(6):65–70, November-December 2005.

[19] Nicholas R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[20] Jason Khallouf and Michael Winikoff. Goal-oriented design of agent systems: A refinement of prometheus and its evaluation. *International Journal on Agent Oriented Software Engineering*, 2008. Accepted, to appear.

[21] Michael Luck and Peter McBurney. Computing as interaction: Agent and agreement technologies. In V. Marik, editor, *Proceedings of the 2008 IEEE International Conference on Distributed Human-Machine Systems*, March 2008.

[22] Peter McBurney and Michael Luck. The agents are all busy doing stuff! *IEEE Intelligent Systems*, 22(4):6–7, July/August 2007.

[23] S. Munroe, T. Miller, R.A. Belecheanu, M. Pechoucek, P. McBurney, and M. Luck. Crossing the agent technology chasm: Experiences and challenges in commercial applications of agents. *Knowledge Engineering Review*, 21(4):345–392, 2006.

[24] Cu D. Nguyen, Anna Perini, and Paolo Tonella. Experimental evaluation of ontology-based test generation for multi-agent systems. In Jorge J. Gomez-Sanz and Michael Luck, editors, *Ninth International Workshop on Agent-Oriented Software Engineering*, pages 165–176, 2008.

[25] James Odell. Agent technology: What is it and why do we care? *Enterprise Architecture, executive report*, 10(3):1–25, 2007.

[26] James Odell. Agents: A necessary ingredient in todays highly collaborative world. *Business Technology Trends & Impacts, Council Opinion*, 8(1), 2007.

[27] Lin Padgham and Michael Winikoff. *Developing Intelligent Agent Systems: A Practical Guide*. John Wiley and Sons, 2004. ISBN 0-470-86120-7.

[28] Lin Padgham, Michael Winikoff, Scott DeLoach, and Massimo Cossentino. A unified graphical notation for AOSE. In *Ninth International Workshop on Agent Oriented Software Engineering (AOSE)*, 2008.

[29] Lin Padgham, Michael Winikoff, and David Poutakidis. Adding debugging support to the prometheus methodology. *Journal of Engineering Applications in Artificial Intelligence*, 18/2, March 2005.

[30] H. Van Dyke Parunak and Sven Brueckner. Dynamic imputation of agent cognition. In Matthias Nickles, Michael Rovatsos, and Gerhard Weiß, editors, *Agents and Computational Autonomy*, volume 2969 of *Lecture Notes in Artificial Intelligence*, pages 209–226. Springer, 2004.

[31] H. Van Dyke Parunak, Paul Nielsen, Sven Brueckner, and Rafael Alonso. Hybrid multi-agent systems: Integrating swarming and BDI agents. In Sven Brueckner, Salima Hassas, Márk Jelasity, and Daniel Yamins, editors, *Engineering Self-Organising Systems, 4th International Workshop, ESOA 2006, Hakodate, Japan, May 9, 2006, Revised and Invited Papers*, volume 4335 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2007.

[32] Gerald Tesauro, David M. Chess, William E. Walsh, Rajarshi Das, Alla Segal, Ian Whalley, Jeffrey O. Kephart, and Steve R. White. A multi-agent systems approach to autonomic computing. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 464–471, Washington, DC, USA, 2004. IEEE Computer Society.

[33] John Thangarajah, Lin Padgham, and Michael Winikoff. Detecting and avoiding interference between goals in intelligent agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 721–726, 2003.

[34] John Thangarajah, Michael Winikoff, Lin Padgham, and Klaus Fischer. Avoiding resource conflicts in intelligent agents. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 18–22. IOS Press, 2002.

[35] Nick A.M. Tinnemeier, Mehdi Dastani, and John-Jules Meyer. Orwell's nightmare for agents? Programming multi-agent organisations. In *Proceedings of the workshop on Programming Multi-Agent Systems (ProMAS'08)*, 2008.

[36] A. van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01)*, pages 249–263, Toronto, 2001.

[37] M. Birna van Riemsdijk, Mehdi Dastani, and Michael Winikoff. Goals in agent systems: A unifying framework. In *Autonomous Agents and Multiagent Systems (AAMAS)*, pages 713–720, 2008.

[38] Philip Wadler. Functional programming: An angry half-dozen. *SIGPLAN Notices*, 33(2):25–30, February 1998.

[39] Michael Winikoff. Implementing commitment-based interactions. In Edmund H. Durfee, Makoto Yokoo, Michael N. Huhns, and Onn Shehory, editors, *6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007*, pages 868–875. IFAAMAS, 2007.

[40] Michael Winikoff and Stephen Cranefield. On the testability of BDI agent systems. Information Science Discussion Paper 2008/03, University of Otago, Dunedin, New Zealand, 2008. `http://www.business.otago.ac.nz/infosci/pubs/papers/dpsall.htm`.

[41] M. Wooldridge, N.R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3), 2000.

[42] Michael Wooldridge and Nicholas R. Jennings. Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, 3(3):20–27, 1999.

[43] F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: the Gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, July 2003.

[44] Franco Zambonelli and Andrea Omicini. Challenges and research directions in agent-oriented software engineering. *Autonomous Agents and Multi-Agent Systems*, 9(3):253–283, 2004.

[45] Zhiyong Zhang, John Thangarajah, and Lin Padgham. Automated unit testing for agent systems. In *Second International Working Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, pages 10–18, 2007.