The Future of Agent-Based Software Engineering: Goals and Verification & Validation are Key

Michael Winikoff*
RMIT University
winikoff@gmail.com

Marketing

The key question is this: how to "market" the results of work in our field to other research fields and to practitioners?

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

1. Clearly identifying (and quantifying? [1]) the "value-add" of agent technology, and telling a simpler and easier-to-understand "story". I believe that this story should focus on the words "complexity", "adaptability", and "goals"; and not on the words "agent" or "autonomous".
2. Working against the perception of agents as "esoteric AI" by continuing the excellent work of AgentLink in documenting case studies of agent-based solutions to real problems [5].
3. Providing agent-based solutions in other areas, such as service-oriented computing [3] and autonomic computing, and publishing in those areas' conferences and journals.

Finally, I believe it's important to produce useful tools, not just papers; and that it's essential to "close the loop" by using our tools and techniques "in anger" [6], thus gaining feedback to guide further work, including detecting unrealistic assumptions.

Agent-Oriented Software Engineering (AOSE)

I agree with Scott DeLoach that reducing differences and moving towards standardisation are important (although I believe that this alone is not sufficient).

I disagree with Mike Georgeff that we still need to "*Bring AO methodologies down to the practice level*": I believe strongly that this has already been achieved by recent methodologies such as Prometheus.

A key challenge is how to design more dynamic agent systems, including those where the structure of the system changes at runtime, and those that exhibit emergent behaviour. Current methodologies are mostly still limited to the design of relatively static and predictable agent systems. Another key area for future work is the "non-classical" phases of the software life-cycle: debugging, testing and software maintenance and evolution.

*This "type 2" position statement was written while on sabbatical at Otago University, Dunedin. I would like to thank Stephen Cranefield for comments on a draft.

Key Research Issues

Goals: 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". Goals, achieved persistently and flexibly, are what give agents their adaptability, and, if our agents are adaptable, then it arguably makes sense to investigate building on this to form a robust and adaptable society of agents. However, there is more research to be done: how to make the design process more goal-oriented? what goal types are useful? how to deal with interactions between goals? how to use flexible and robust goal achievement by single agents to enable multiple agents to achieve goals in a robust and flexible way? how does this compare with conventional approaches for choreography/orchestration? how does this compare with agent-based approaches (e.g. norms, protocols, social commitments, teamwork)? how to design goal-based agents in systems which exhibit emergent behaviour? how to ascribe goals to non-goal-based (and possibly emergent) behaviours?

Validation & Verification: A recurring issue in practice [5, 4] 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. An obvious answer to this problem is the use of some form of formal methods [2]. However, much more research is needed to make this practical. One key 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. For instance, what does it mean for a manufacturing management and optimisation system [5, section 3] to be functioning correctly? A second key challenge is dealing with systems that are dynamic, including those whose structure may change. One possible approach is to use an assume-guarantee style reasoning [2], another is to defer (some) reasoning and checking to run-time.

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] 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.

[3] 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.

[4] M. Luck and P. McBurney **Computing as interaction: agent and agreement technologies**. In: V. Marik (Editor): Proceedings of the 2008 IEEE International Conference on Distributed Human-Machine Systems. Athens, Greece, March 2008.

[5] 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.

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