Moving multi-agent systems from research to practice

Scott A. DeLoach

Department of Computing and Information Sciences
Multiagent & Cooperative Robotics Laboratory
Kansas State University, USA
E-mail: sdeloach@ksu.edu

Abstract: The state of the art in multi-agent research and engineering is insufficiently reflected in the state of the practice in complex distributed systems because the community has yet to demonstrate the significant benefits of using agent-oriented approaches to solve complex problems. The practitioner's view that multi-agent approaches are not technically superior to traditional approaches is understandable; for every successful multi-agent system, it is possible to envision a non-agent approach that is equally suited for the task. Agent-oriented software engineering lies directly at the heart of this problem. In order to be accepted, the agent community needs to demonstrate that they can build reliable complex, distributed systems using agent-oriented approaches that are repeatable and sound. This paper identifies three obstacles that hamper progress towards such a demonstration: the lack of a common understanding of key multi-agent concepts, the lack of a common set of notations and models, and the lack of flexible, industrial strength methods and techniques for developing multi-agent systems.

Keywords: modelling language; metamodelling; method engineering.

Reference to this paper should be made as follows: DeLoach, S.A. (2009) 'Moving multi-agent systems from research to practice', *Int. J. Agent-Oriented Software Engineering*, Vol. 3, No. 4, pp.378–382.

Biographical notes: Scott A. DeLoach is an Associate Professor in the Computing and Information Sciences Department at Kansas State University, USA. His research focuses on methods and techniques for the analysis, design and implementation of complex adaptive systems, which have been applied to both multi-agent and cooperative robotic systems. Dr. DeLoach is best known for his work in agent-oriented software engineering. He is the creator of the Multiagent Systems Engineering (MaSE) methodology, its follow-on Organization-based Multiagent Systems Engineering (O-MaSE) methodology, and the associated agentTool analysis and design tool. He has more than 50 refereed publications and has advised over 25 graduate students. Dr. DeLoach came to Kansas State University after a 20-year career in the US Air Force.

1 Introduction

The state of the art in multi-agent research and engineering is insufficiently reflected in state of the practice in complex distributed systems for the basic reason that we have yet to demonstrate, or at least publicise, the significant benefits of using true agent-oriented approaches to solve complex problems. That, coupled with the perception that agent technology carries with it a steep learning curve due to its reliance on conceptually sophisticated reasoning schemes such as Belief-Desires and Intentions (BDI) type architectures, has scared many companies towards simpler, easy to justify technologies such as service-oriented approaches.

Practitioner's view that multi-agent approaches are not technically superior to traditional approaches is relatively easy to understand. For every successful multi-agent system, it is possible to envision a non-agent approach that is equally suited for the task. After all, almost all agent systems are programmed in the same programming languages as non-agent systems. What we have failed to demonstrate is that the agent approach can yield technically competitive (or better) solutions with a *real* benefit, most likely in terms of reduced costs, greater reliability, greater flexibility, or a greater chance of repeatable success.

While the perception of agent-oriented approaches as sophisticated and hard to understand is as much a public relations problem as it is a technical problem, the failure to produce examples of agent systems with significant benefits is not. It is the bottom line. I believe that a major part of the solution to both these problems lies in creating and documenting better approaches, tools, and techniques for developing agent systems. Thus, agent-oriented software engineering lies directly at the heart of this problem. Agent-oriented software engineering provides the tools and techniques to use in designing complex, adaptive systems. What we need to demonstrate is that we can build reliable complex, distributed systems using agent-oriented approaches that are repeatable and sound.

2 Making progress

While agent-oriented research has been around for over 15 years, it is still a relatively young area. With object-oriented approaches, Simula in 1967 and Smalltalk in 1972 were the first object-oriented programming languages (Dahl, 1968; Goldberg and Robson, 1983), although object-oriented programming did not become mainstream until the 1990s. As a parallel, research into agents and multi-agent systems did not begin to gain widespread attention until the early 1990s, so we may be a bit premature in expecting a large integration of agent oriented approaches in industry today. However, I believe agent oriented approaches do hold promise for large, complex, and adaptive systems and thus we are justified at looking at current obstacles to its use and acceptance.

I believe that there are currently several obstacles that hamper progress towards being able to use multi-agent systems and agent-oriented software engineering in mainstream applications. These include:

- the lack of a common understanding of key multi-agent concepts
- the lack of a common set of notations and models
- the lack of flexible, industrial strength methods and techniques for developing multi-agent systems.

The lack of an agreement on the key multi-agent concepts and their definitions is the first obstacle to be breached in the battle towards making multi-agent systems a mainstream paradigm. For instance, the vast majority of computer science students and practicing professionals would easily be able to define and generally agree upon the basic definitions of the object-oriented notions of objects, classes, generalisation, specialisation, and aggregation. Yet, at the same time, most experienced multi-agent researchers would have a difficult time trying to reach agreement on the commonly used notions of agents, roles, conversations, plans, organisations, or capabilities. The closest thing we have to agreement is on the definition of an intelligent agent as a computational system that senses and acts autonomously in a dynamic environment in order to realise a set of goals (Russell and Norvig, 2003). Although many researchers and practitioners use the names to represent similar concepts, the real problem lies in the relationships between the concepts.

Within this area, I believe is one of the key impediments to the acceptance of agent technology is its perception as complex and difficult to learn and use. This, unfortunately, is a problem perpetuated by the agent community. Many agent researchers still consider true agents to require mentalistic notions such as a BDI architecture. While there are many benefits of such approaches, they are certainly more complex than typical objects and thus provide an additional barrier to adoption. What we as a community should be stressing is the usefulness of an agent-oriented approach to *modelling and developing* distributed, complex, and adaptive systems, and focus less on agents and requirements for agenthood.

A second major obstacle I see is the lack of a common notation and models for multi-agent concepts. Of course, given that we have not decided on the definition of the concepts and their relationships themselves, finding a common representation may seem like an insignificant problem. However, a lack of a common notation makes it hard for practitioners to investigate different methods and techniques since they have to relearn notation for each different approach. Also, a common notation makes the similarities between approaches and models much easier to spot. In recent work with Padgham and Winikoff (Padgham *et al*., 2008), we found that after putting our respective set of models (O-MaSE (Garcia-Ojeda *et al*., 2008) and Prometheus (Padgham and Winikoff, 2004)) into a common notation, the similarities between the two methodologies and the concepts we used was much more readily apparent. However, I urge extreme caution when talking about formal standardisation efforts. While standardisation can be helpful in moving technology towards industrial adoption, early standardisation tends to stifle research. Given that we do not currently agree on all the main concepts of agent based approaches, standardisation, if at all successful, will likely choke off many promising avenues of research in favour of *standard* approaches.

The third obstacle is the lack of strong industry acceptance for any current agent-oriented methodologies. Reasons for this lack of acceptance include the variety of concepts and approaches upon which these methodologies are based along with a lack of tools to support them. However, I believe that one of the major reasons for this lack

of acceptance is that the current set of methodologies tends to be inflexible and hard to extend for a variety of applications. One solution is to allow users to customise methodologies to the different types of applications being developed. There have been some suggestions for increasing industrial acceptance. For instance, Odell *et al*. (2001) suggest presenting new techniques as an incremental extension of known and trusted methods, Bernon *et al*. (2004) suggest the integration of existing agent-oriented methodologies into one highly defined methodology, and Henderson-Sellers and Giorgini (2005) suggests the use of method engineering.

Among these approaches, method engineering seems to be the best choice for the current state of agent-oriented software engineering. While presenting agents as an incremental extension of object-orientation has it advantages from marketing perspective, much of the power of the agent-oriented approach is lost. Autonomous or pro-active objects are simply written off as a repackaging of existing object-oriented concepts and thus provide little additional advantage. Integration of existing agent-oriented methodologies into one highly defined methodology also sounds promising until one examines the variety of approaches and concepts currently being used. Adopting this strategy would require a solution to the first two obstacles and the creation of a homogenised process where all approaches and concepts must be able to be used together. At this point in time, solving the first two obstacles is a large enough challenge; creating a single process that includes all approaches and concepts is simply unfeasible. That leaves method engineering. While it too requires at least a partial solution to the first two obstacles, method engineering allows the integration of techniques and approaches that may not be completely compatible. For example, some approaches to modelling interactions between agents provide strong guarantees if you assume interactions between only two agents (Lacey and DeLoach, 2000); however, these approaches are not compatible with other techniques that allow modelling of interactions between several agents. Method engineering would allow the developers to pick the approach they need for their application without requiring all approaches to be compatible.

3 Conclusion

Based on these observations, agent-oriented software engineering and multi-agent researchers must seriously address the obstacles presented above if multi-agent approaches are to become viable for the development of complex distributed systems. As a first step, a core set of concepts that are well understood and accepted amongst multi-agent practitioners must be developed. Essentially, this boils down to defining a core metamodel for multi-agent systems. While not all concepts and relationships need be represented, the core concepts and their relationships must be defined. While there has been work towards defining a common metamodel (Bernon *et al*., 2004), unfortunately, the proposed metamodels tend to be overly complex and of limited practical use. Other work on metamodels does exist (*e.g.*, Ferber and Gutknecht, 1998; Beydoun *et al*., 2005; Juan and Sterling, 2003), however, there still remains insufficient convergence on common concepts and semantics. Once this first step is in place, the next two steps, creating a common notation and creating industrial strength methods and techniques can be vigorously pursued. Then, the existence of industrial strength methods and techniques will enable the ultimate goal of demonstrating the usefulness of multi-agent approaches in the development of sound and repeatable complex, distributed systems.

Acknowledgement

The arguments presented in this paper were based on work supported by grants from the US National Science Foundation (0347545) and by the US Air Force Office of Scientific Research (FA9550-06-1-0058).

References

Bernon, C., Cossentino, M., Pierre Gleizes, M., Turci, P. and Zambonelli, F. (2004) 'A study of some multi-agent meta-models', *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004*, Revised Selected Papers, Springer, pp.62–77.

Beydoun, G., Gonzalez-Perez, C., Low, G. and Henderson-Sellers, B. (2005) 'Synthesis of a generic MAS metamodel', *SELMAS '05: Proceedings of the Fourth International Workshop on Software Engineering for Large-Scale Multi-Agent Systems*, New York, NY: ACM, pp.1–5.

Dahl, O. (1968) *SIMULA 67 Common Base Language*, Norwegian Computing Center Publication, ISBN: B0007JZ9J6.

Ferber, J. and Gutknecht, O. (1998) 'A meta-model for the analysis and design of organizations in multi-agent systems', *ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems*, IEEE Computer Society, Washington, DC, USA, p.128.

Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H. and Valenzuela, J. (2008) 'O-MaSE: a customizable approach to developing multiagent development processes', in M. Luck and L. Padgham (Eds.) *Agent-Oriented Software Engineering VIII: The 8th International Workshop on Agent Oriented Software Engineering (AOSE 2007)*, Springer-Verlag Berlin, Inc., pp.1–15.

Goldberg, A. and Robson, D. (1983) *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley Longman Publishing Co., Inc.

Henderson-Sellers, B. and Giorgini, P. (2005) *Agent-Oriented Methodologies*, Idea Group Inc.

Juan, T. and Sterling, L. (2003) 'A meta-model for intelligent adaptive multi-agent systems in open environments', *AAMAS '03: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY: ACM, pp.1024–1025.

Lacey, T. and DeLoach, S.A. (2000) 'Verification of agent behavioral models', in H. Arabnia (Ed.) *Proceedings of the International Conference on Artificial Intelligence (IC-AI'2000)*, CSREA Press, Vol. II, pp.557–563.

Odell, J., Parunak, H.V.D. and Bauer, B. (2001) 'Representing agent interaction protocols in UML', *First International Workshop, AOSE 2000 on Agent-Oriented Software Engineering*, Secaucus, NJ: Springer-Verlag New York, Inc., pp.121–140.

Padgham, L. and Winikoff, M. (2004) *Developing Intelligent Agent Systems: A Practical Guide*, John Wiley and Sons.

Padgham, L., Winikoff, M., DeLoach, S. and Cossentino, M. (2008) 'A unified graphical notation for AOSE', in M. Luck and L. Padgham (Eds.) *Proceedings of the 8th International Workshop on Agent Oriented Software Engineering*, Berlin: Springer-Verlag, Inc., in press.

Russell, S.J. and Norvig, P. (2003) *Artificial Intelligence: A Modern Approach*, Pearson Education.