Toward Interaction-Oriented Programming

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA

singh@ncsu.edu

Abstract

Although much progress has been made in agent theory and practice, bottlenecks remain in the construction of complex multiagent systems. We introduce *interaction-oriented programming (IOP)* as an approach to orchestrate the interactions among agents. As envisioned, IOP is more tractable and practical than general agent programming, especially in settings where the internal details of autonomously developed agents are hidden. By enabling declarative specification and enactment of agent interactions, IOP can channel the intellectual energies of designers into the most amenable and effective design tasks. Our preliminary approach—implemented in an actor language—formally specifies certain interactions, and executes them in a distributed manner.

Interaction-oriented programming (IOP) is envisioned as a class of formalisms and techniques to develop multiagent systems. IOP is concerned with (i) the semantics of interactions, (ii) languages for expressing interactions, and (iii) techniques and tools for realizing multiagent systems.

IOP includes abstractions for (a) a rigorous understanding of events in a multiagent system, (b) message passing to implement control and data flow (Hewitt 1991), (c) patterns of interactions, (d) knowledge-level communication constraints (Singh 1994), and (e) social constructs (Gasser 1991). What is new here is the focus on programmability and common structure or "design-patterns" without, however, any loss of rigor. We have developed a formal approach for (a), (b), and (c). Aspects (d) and (e) are being studied.

Figure 1 shows our conceptual architecture. At the bottom is the assumed infrastructure. At the top are the application-specific multiagent systems to be built. In between are the IOP layers of the interaction specification and management. They include functionality to specify the *patterns of interaction*, translate them into low-level "events," and schedule them through passing appropriate messages among agents. The low-level events correspond to the agents' significant (external) transitions. Capturing the patterns explicitly enables us to flexibly take advantage of their commonalities, thereby maintaining the key properties of interactions across different situations by controlling low-level events appropriately. Thus a programmer can create a multiagent system by defining (or reusing) agents, and setting them up to interact in some desired way. The interactions require knowledge only of the agents' external events that feature in the interactions.

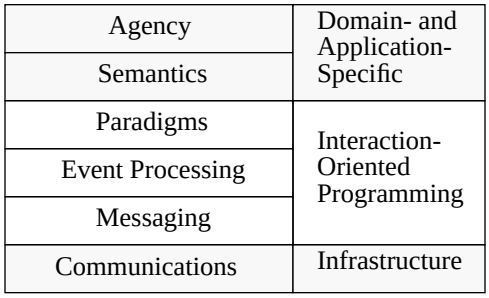

Figure 1: Layering of functionality and focus

The full paper (Singh 1996) introduces a formal notation, and uses it to specify and schedule patterns of interactions. These patterns can include data flow and control flow interactions, which underlie common protocols for coordination and negotiation. Future work includes the elaboration of IOP to accommodate more sophisticated kinds of interaction, based on communication constraints and social commitments.

References

Gasser, L. 1991. Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence* 47:107–138.

Hewitt, C. 1991. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence* 47:79–106.

Singh, M. P. 1994. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications*. Heidelberg, Germany: Springer Verlag.

Singh, M. P. 1996. Toward interaction-oriented programming. NCSU Computer Science TR-96-15. `www4.ncsu.edu/eos/info/ dblab/ www/mpsingh/ papers/agents+multiagents/ iop.ps`.