



Guest Editors' Introduction

Object-Oriented Programming in AI

Sherman R. Alpert, IBM T.J. Watson Research Center
Scott W. Woyak, EDS Research and Development
Howard J. Shrobe, Symbolics
Lloyd F. Arrowood, Martin Marietta

In recent years, the computer science community has shown increasing interest in the object-oriented programming paradigm. Many researchers have pointed out OOP's software engineering benefits, such as ease of reuse, modularity, and extensibility.¹ The artificial intelligence community has also shown interest,^{2,3} as evidenced by many object-oriented extensions to conventional AI programming languages such as Lisp and Prolog. The most prominent recent example is the Common Lisp Object System,^{4,5} a hardware-independent standard that demonstrates a desire to put object-oriented tools in the hands of large numbers of AI system builders.

Such activity reflects a belief that OOP can benefit AI application developers. To explore the nature of such benefits, *IEEE Expert* is initiating a special track: "Object-Oriented Programming in AI." This track will feature case studies demonstrating OOP's utility in constructing intelligent systems. Our goal is to illustrate the types of AI problems and applications for which OOP facilitates a computational solution. We begin the special track with two articles in this issue; several more articles will appear in subsequent issues.

OOP techniques have been used in many practical AI systems, including system development environments and domain-specific knowledge-based applications. The experience of building these systems has revealed a number of inherent benefits of the object-oriented approach. For example, objects encapsulate both data and processing capabilities in a

single entity, facilitating the representation of both knowledge and the entities of the domain being modeled. Since we can view objects as computer-based, executable instances of corresponding entities in the problem domain,⁶ the object-oriented approach lets us focus on those entities and construct computational entities that map closely to them and have similar abstracted capabilities.⁷ Systems can then be "inhabited" by the same entities as those in the problem domain. This feature has been exploited in intelligent simulation research.^{8,9} Moreover, these ideas apply equally well to conceptual entities, such as goals and rules in an expert reasoning system, as to concrete objects.

OOP also offers the advantages of modularity, which eases the task of the AI system developer who aims to "carve nature at the joints." Modularity combines with the additional object-oriented features of data hiding and polymorphism to allow the seamless integration of heterogeneous knowledge structures and inferencing mechanisms. Although different objects might employ disparate representations internally or behave differently, they might all respond to an identical set of messages, or protocol. For example, in Joshua,¹⁰ a generic set of messages defines a "Protocol of Inference" for modifying and querying a database of assertions. Although different types of assertions could have different ways of responding to messages, all are externally represented and accessed in a uniform way. In another example, the EDS/OWL environment¹¹ defines behaviors (meth-

ods, slot access, etc.) as objects adhering to a common protocol. Specialized behavior objects enable the uniform integration of rule-based, logic, and access-oriented programming.

Similarly, OOP can be the enabling technology for integrating AI components with conventional applications. Our first article, by David Franke, describes an object-oriented protocol that allows the integration of an inferencing mechanism with CAD tools. The protocol defines a set of messages for performing both forward and backward chaining inferencing. In particular, the protocol lets the rules operate on the tool's data structures as if the structures were assertions in the rule base. This provides a tighter coupling between the existing tool and the inferencing component, and it avoids translations between the different representations.

OOP also facilitates other strategies for the construction of intelligent systems. For example, one approach asserts that a knowledge-based task can be accomplished by decomposing it into several agents, each containing and representing a chunk of the complete knowledge needed to accomplish the overall task. By applying the object-oriented metaphor of intelligent communicating agents, we can view the computational system in terms of individual experts that accomplish goals via interaction. A variety of applications have employed these notions, including a natural-language parser in which objects representing parsing experts for individual sentence constituents interact to parse

entire sentences.¹² OOP's fundamental characteristics facilitate the construction of a system with such an architecture: because objects encapsulate both state and behavior, and because they possess inherent communication capabilities, they are natural candidates for the implementation of distributed, multiple-agent, intelligent systems.¹³

Our second paper, by Naser Barghouti and Gail Kaiser, further explores the use of object orientation in multiple-agent environments. In their rule-based software development environment, objects model both software modules and the agents that act on them (either human users or components of the environment). Rules capture the requirements of the software development methodology being modeled, for example, a requirement that certain test suites must be run after a particular module is modified. The central issue explored by Barghouti and Kaiser is how to support cooperation among a team of software developers working on a common project. Their solution involves an object-oriented database with a novel concurrency-control mechanism rooted in the semantics of the object-oriented representation.

Over the course of this special track we will examine other issues of interest to AI application developers, such as articles that

- compare the use of objects to represent knowledge with other approaches, such as frames;
- compare the OOP notion of inheritance to other models of representation sharing;
- examine the relationship between the use of objects to implement cooperating intelligent agents and the AI subfield of distributed artificial intelligence;
- discuss the adaptability of objects for automated knowledge acquisition or machine learning systems;
- explore the integration and management of objects in production environments that emphasize persistence, security, and performance; and
- describe the use of object-oriented programming in other specific knowledge-based application domains.

We thank Editorial Board member Sanjay Mittal, Editor-In-Chief B. Chandrasekaran, and former Managing Editor Henry Ayling for their support and guidance. We also thank our volunteer referees, without whom this track would not have been possible, and the researchers and application developers who submitted manuscripts. Lastly, we thank our respective employers, who have provided support in the form of time for this special track.

References

1. B.J. Cox, *Object-Oriented Programming: An Evolutionary Approach*, Addison-Wesley, Reading, Mass., 1986.
2. M. Stefik and D.G. Bobrow, "Object-Oriented Programming: Themes and Variations," *AI Magazine*, Vol. 6, No. 1, Winter 1986, pp. 40-62.
3. D.G. Bobrow and M.J. Stefik, "Perspectives on Artificial Intelligence Programming," *Science*, Vol. 231, 1986, pp. 951-957, reprinted in *Readings in Artificial Intelligence and Software Engineering*, C. Rich and R.C. Waters, eds., Morgan Kaufmann, San Mateo, Calif., 1986, pp. 581-587.
4. L.G. DeMichiel and R.P. Gabriel, "The Common Lisp Object System," in *Proc. European Conf. Object-Oriented Programming (ECOOP 87)*, J. Bezevin et al., eds., Springer-Verlag, Berlin, 1987, pp. 151-170.
5. S.E. Keene, *Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS*, Addison-Wesley, Reading, Mass., 1989.
6. B.J. Cox, "Message/Object Programming: An Evolutionary Change in Programming Technology," *IEEE Software*, Vol. 1, No. 1, Jan. 1984, pp. 50-61.
7. M.B. Rosson and S.R. Alpert, "The Cognitive Consequences of Object-Oriented Design," *Human-Computer Interaction*, Vol. 5, 1990, pp. 345-379.
8. H.H. Adelsberger et al., "Rule-Based Object-Oriented Simulation Systems," in *Intelligent Simulation Environments*, P.A. Luker and H.H. Adelsberger, eds., Society for Computer Simulation, San Diego, Calif., 1986, pp. 107-112.
9. H.R. Myler, "Object-Oriented Training Simulation," in *AI Papers 1988: Proc. Conf. AI and Simulation*, R.J. Uttamsingh, ed., Society for Computer Simulation, San Diego, Calif., 1988, pp. 156-160.
10. S. Rowley et al., "Joshua: Uniform Access to Heterogeneous Knowledge Structures, or Why Joking is Better than Conniving or Planning," in *Proc. National Conf. Artificial Intelligence (AAAI 87)*, MIT Press, Cambridge, Mass., 1987, pp. 48-52.
11. M.H. Ibrahim and S.W. Woyak, "An Object-Oriented Environment for Multiple AI Paradigms," in *Proc. IEEE Int'l Conf. Tools for Artificial Intelligence*, 1990, pp. 77-83.
12. S.R. Alpert and M.B. Rosson, "Object-Oriented Programming for AI Application Development," IBM research report, available from author at address below, 1990.
13. H. Lieberman, "Languages, Object Oriented," in *Encyclopedia of Artificial Intelligence, Volume 1*, S.C. Shapiro, ed., John Wiley & Sons, New York, 1987, pp. 452-456.

Shorman R. Alpert is on the research staff of the User Interface Institute at IBM's T.J. Watson Research Center. For several years prior, he was a software consultant. He is a doctoral student in the Computing in Education program at Columbia University's Teachers College, where he received his MA in 1987. He received a BS in computer science from the State University of New York at Stony Brook in 1974. His current research interests include educational issues in object-oriented programming, and his current projects includes a computer-based curriculum and an intelligent tutoring system for Smalltalk. He is a member of the IEEE Computer Society and the ACM.

Scott W. Woyak is a senior research engineer at EDS Research and Development. He was a principal developer of the EDS/OWL object-oriented AI programming environment, and he is now developing an object-oriented framework for graphical user interfaces. Woyak chaired a workshop on object-oriented programming in AI at IJCAI-89 and AAAI-90. He received a BS in computer science from Northwestern University, and he is a member of the IEEE Computer Society and the ACM.

Howard J. Shrobe is vice president of technology at Symbolics, where he was one of the architects of the Ivory microprocessor and the NS CAD system used to design it. He has since led the effort to develop Joshua, an artificial-intelligence programming language that introduced the notion of a Protocol of Inference. Shrobe also is a member of the Hardware Troubleshooting project and a lecturer in artificial intelligence at the Massachusetts Institute of Technology. His interests have included VLSI design, computer architecture, and artificial intelligence.

Shrobe is coauthor of *Interactive Programming Environments* and was editor of the AAAI book, *Exploring Artificial Intelligence: Surveys from the National Conferences on Artificial Intelligence*. He received his MS and PhD degrees from MIT's Artificial Intelligence Laboratory, where he was a cofounder of the Programmer's Apprentice project.

Lloyd F. Arrowood is a computing specialist in the Computing and Telecommunications Division of Martin Marietta Energy Systems, which manages Oak Ridge National Laboratory for the US Department of Energy. For the past six years, Arrowood has developed knowledge-based systems for a wide variety of applications. While at Oak Ridge, he has been one of the principal developers of ARK, a knowledge acquisition tool. Arrowood's current research interests include knowledge acquisition, machine learning, and truth maintenance. He holds a BA and an MS degree in computer science from the University of Tennessee.

Readers can reach the guest editors through Alpert at IBM's T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598.