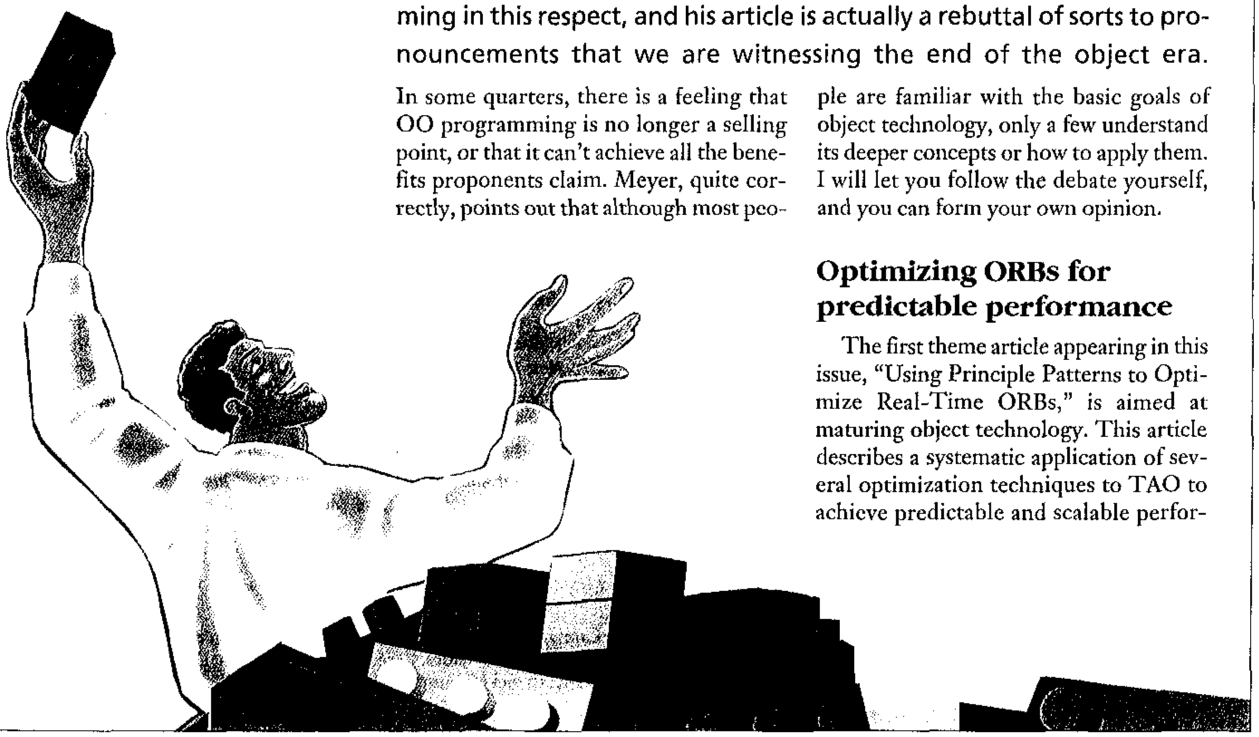
Guest Editor's Introduction

Murthy Devarakonda
IBM T.J. Watson Research Center

Current Developments in Object-Oriented Systems

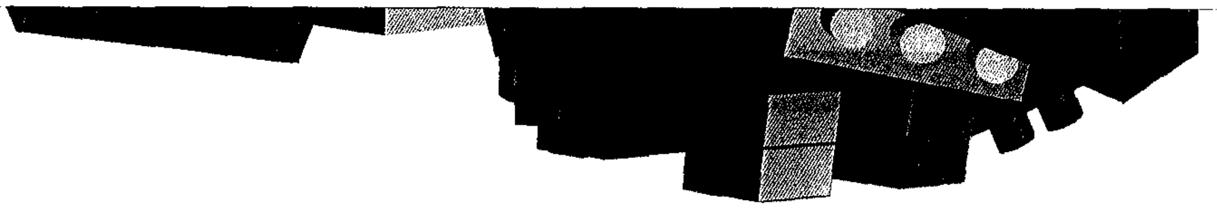
In the July–September 1998 issue of *IEEE Concurrency*, I introduced the Object-Oriented Systems track, stating that object-oriented programming was commonplace. I seem to have understated just how commonplace OO programming really is. According to Bertrand Meyer, OO programming is so obvious that many practitioners do not realize that the ideas were once new and controversial.[1] He finds an analogy between structured programming and OO programming in this respect, and his article is actually a rebuttal of sorts to pronouncements that we are witnessing the end of the object era.

In some quarters, there is a feeling that OO programming is no longer a selling point, or that it can't achieve all the benefits proponents claim. Meyer, quite correctly, points out that although most people are familiar with the basic goals of object technology, only a few understand its deeper concepts or how to apply them. I will let you follow the debate yourself, and you can form your own opinion.



Optimizing ORBs for predictable performance

The first theme article appearing in this issue, "Using Principle Patterns to Optimize Real-Time ORBs," is aimed at maturing object technology. This article describes a systematic application of several optimization techniques to TAO to achieve predictable and scalable perfor-



mance. First, the authors identify challenges in meeting soft and hard real-time requirements for CORBA middleware, and then they implement solutions for these challenges using a set of well-known optimization principles.

They attack typical performance issues in ORBs such as server-side thread-pool management, data copying, memory allocation, and request demultiplexing to server objects. Almost every large distributed-system design must address performance issues similar to these. However, the important exercise here is to understand each of these issues as they relate to CORBA middleware and choose those optimizations that result not only in performance improvements but also in performance predictability. Indeed, the experimental approach can be a good framework for optimizing any complex distributed system. I hope that, in time, most commercial vendors will incorporate the optimization techniques explored here into their own ORBs and that application developers requiring quality-of-service requirements won't have to choose between an ORB and building everything from scratch.

Component management

In the second article in this issue, Fabio Kon and Roy Campbell propose a framework for representing dependencies among components in a distributed system. Object-oriented programming naturally leads to component-based software construction; if it is possible to reason about dependencies among components, then programmers can build reliable and dynamically configurable systems using reusable components. The system can check dependencies statically (for example, at install time) or manage them dynamically (at runtime).

Dependency management is not a radically new idea. Any runtime environment that supports loadable components provides some degree of dependency checking, however rudimentary. COM and JavaBeans implicitly keep track of this information, and when component dependencies are not properly honored, the program or system crashes. Operating systems supporting dynamically loadable modules keep track of such information.

Kon and Campbell have taken component-dependency management to a new level by providing a framework that explicitly represents and manages such dependencies. This lets programmers offer fault tolerance, QoS, and on-the-fly upgrading of complex software through dynamic reconfiguration. A component configurator framework captures the dependency representations, and the configurator has been implemented in TAO—the freely available ORB described in the first article. CORBA provides a well-known distributed-object model to experiment with these ideas, and the TAO implementation offers flexibility. We don't know if all component-based systems need the full flexibility this dependency management offers, but the work provides a data point, and hopefully several researchers will experiment with it to evaluate its usefulness.

New features for CORBA 3

One final thought is how OMG is evolving CORBA. Jon Siegel categorizes coming attractions in CORBA 3 as Internet integration, QoS control, and the CORBAcomponent architecture (see CORBA 3 release info at www.omg.org/news/pr98/compnent.html and the news story in this issue on page 10). In the Internet integration, OMG has firewall specification including support for bidirectional GIOP (General Inter-ORB Protocol) connections. This feature gets around a nasty problem in going over the firewalls. Previously, GIOP connections carried invocations in one direction, so implementations supporting callbacks had to use two different connections between a client and an object, and the second connection operated in the reverse direction. For a typical firewall, connections coming in from the outside are not allowed; therefore, supporting callbacks over a firewall posed a serious problem. The bidirectional GIOP connections solve this problem.

Among the QoS control specifications, the notable ones are *messaging specification*, which allows a variety of invocation modes with the ability to control QoS, and the *specification of real-time CORBA*, which standardizes resource controls using priority models to achieve predictable behavior. The CORBAcomponent architecture provides a container environment, support for Enterprise JavaBeans, and a software distribution format. For an application programmer, the advantage is that the container provides persistence, transactionality, and security at a level higher than the CORBAservices.

Once again, what I see in CORBA evolution is object technology reaching a higher level of practical sophistication, letting us build large and complex systems with this technology.

Acknowledgments
The articles appearing in this issue were selected from the 5th Usenix Conference on Object Oriented Technologies and Systems (May 1999, San Diego), published after a rereview with the kind permission of USENIX.

References

1. B. Meyer, "A Really Good Idea," *Computer*, Vol. 32, No. 12, Dec. 1999, pp. 144-147.

Murthy Devarakonda is a research manager and technical strategist in personal systems software at the IBM T.J. Watson Research Center. His research interests include object-oriented systems, distributed systems, file systems, and end-user systems software. He received his PhD in computer science from the University of Illinois at Urbana-Champaign. He is a member of the ACM, IEEE Computer Society, and USENIX. Contact him at the IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598; mdev@us.ibm.com.