

UML in ACTION

GRADY BOOCH, GUEST EDITOR

THE UNIFIED MODELING LANGUAGE

(UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML was adopted as a standard by the Object Management Group (OMG) in November 1997 and now serves as the standard language of blueprints for software. As such, the UML has found widespread use: it has been applied successfully to build systems for tasks as diverse as e-commerce, command and control,

computer games, medical electronics, banking, insurance, telephony, robotics, and avionics. The UML transcends most traditional programming languages: there are mappings from the UML to Java, C++, Smalltalk, Visual Basic, Ada, and many 4GLs. The UML even applies to all of the commercial middleware architectures, including Enterprise Java Beans (EJB), the OMG's CORBA, and Microsoft's DNA. The UML has also found a place in codifying architectural frameworks: IBM's San Francisco frameworks and their Insurance Application Architecture both use the UML. The UML may not get the media attention of, for example, EJB, DNA, or XML, but in the past several years it has quietly proven itself to be an essential tool for building quality systems in an efficient and predictable fashion.

The UML had its beginnings in the late 1980s.¹ Faced with a new genre of object-oriented programming languages and increasingly complex applications, methodologists began to experiment with alternative approaches to analysis and design. The number of OO methods increased from fewer than 10 to more than 50 between 1989 and 1994. Many users of these methods had trouble finding a modeling language that met their needs completely, thus fueling the so-called method wars. As users learned from experience, new generations of these methods began to appear, a few clearly prominent, most notably Booch, Jacobson's Object-Oriented Software Engineering (OOSE), and

¹Booch, G., Rumbaugh, J., and Jacobson, I. *The Unified Modeling Language User Guide*. Addison Wesley, Reading, Mass., 1999.



Rumbaugh's Object Modeling Technique (OMT). Other important methods included Fusion, Shlaer-Mellor, and Coad-Yourdon. Each of these was a complete method, although each was recognized as having strengths and weaknesses. In simple terms, the Booch method was particularly expressive during the design and construction phases of projects, OOSE provided excellent support for use cases as a way to drive requirements capture, analysis, and high-level design, and OMT-2 was most useful for analysis and data-intensive information systems. The behavioral component of many OO methods, including the Booch method and OMT, was the language of statecharts, invented in 1983 by David Harel.

The UML effort started officially in October 1994, when Rumbaugh joined me at Rational Corporation. Our project's initial focus was the unification of the Booch and OMT methods. The version 0.8 draft of the Unified Method (as it was then called) was released in October 1995. Around the same time, Jacobson joined Rational and the scope of the UML project was expanded to incorporate OOSE. Our efforts resulted in the release of the UML version 0.9 documents in June 1996. Throughout 1996, we invited and received feedback from the general software engineering community. Notable contributions were made by Harel and Eran Gery from I-Logix, with whom we collaborated during that period (based on their 1995 work on an OO variant of statecharts). During this time, it also became clear that many software organizations saw the UML as strategic to their businesses. We established a UML consortium, with several organizations willing to dedicate resources to work toward a strong and complete UML definition. Partner organizations contributing to the UML 1.0 definition included Digital Equipment Corporation, Hewlett-Packard, I-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments, and Unisys. This collaboration resulted in the UML 1.0, a modeling language that was well-defined, expressive, powerful, and applicable to a wide spectrum of problem domains. UML 1.0 was offered to the OMG for standardization in January 1997 in response to their request for proposal for a standard modeling language.

Between January 1997 and July 1997, the original group of partners was expanded to include virtually all of the other submitters and contributors of the original OMG response, including Andersen Consulting, Ericsson, ObjecTime Limited, Platinum Technology, PTech, Reich Technologies, Softeam, Sterling Software, and Taskon. A semantics task force was formed, led by Cris Kobryn of MCI Systemhouse and administered by Ed Eykholt of Rational, to for-

malize the UML specification and to integrate the UML with other standardization efforts. A revised version of the UML (version 1.1) was offered to the OMG for standardization in July 1997. In September 1997, this version was accepted by the OMG Analysis and Design Task Force and the OMG Architecture Board and was put up for vote by the entire OMG membership. UML 1.1 was adopted by the OMG in November 1997. Maintenance of the UML was then taken over by the OMG Revision Task Force (RTF), led by Cris Kobryn. The RTF released an editorial revision, UML 1.2, in June 1998. In June 1999, the RTF released UML 1.3, which provided some technical cleanup.

IN THIS SECTION, CRIS KOBRYN DESCRIBES THE process and activities of the RTF and the changes made to the UML leading up to version 1.3. Kobryn goes on to offer a roadmap for the future of the UML.

A full study of the UML's use in industry could fill several volumes, so I've selected a representative set of application domains, which serve to demonstrate the range of the UML in action. Bran Selic examines the application of the UML for real-time systems, and offers some insight into the ongoing work to develop a UML profile for that domain. Alex Bell and Ryan Schmidt provide a case study for the UML in one very complex domain, the AWACS (Airborne Warning and Control System) modernization program. Grant Larsen next examines how the UML can be utilized to codify design and architectural patterns. Jim Conallen concludes this collection of articles by showing how the UML is applied to the development of Web-centric systems.

From hard real-time systems to e-business and virtually everything in between, the UML has become part of the mainstream of software development, enabling teams to reconcile and coordinate the needs of various stakeholders, to gain control of their systems' architecture, and to manage complexity. The articles appearing here provide a sampling of the UML as it is used in a variety of tasks, applications, and missions—it is hoped that *Communications'* readers will benefit from the authors' sharing of their experiences with the UML in action. ■

GRADY BOOCH (egb@rational.com) is Chief Scientist with Rational Software Corporation in Denver, CO; www.rational.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 1999 ACM 0002-0782/99/1000 \$5.00