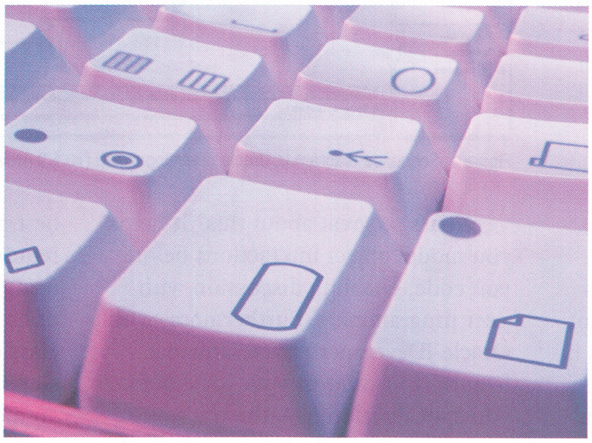
BY KELLI WISETH

Picturing Program Design

UML offers standard notation for modeling complex systems.

For millennia, people have been using graphical approaches—stick figures and the like—to communicate ideas. When the first programmers started creating applications for business users (and it quickly became clear that neither party understood the other's language), once again stick figures (along with boxes, lines, bubbles, and the occasional word) came to the rescue.

Over time, various graphically oriented techniques, such as producing the data definition diagrams used by structured programmers and the entity relationship diagrams used for relational database applications, became mainstream approaches to facilitating analysis and design as well as a means of recording and communicating system requirements. Today, in the world of object-oriented and distributed computing, Unified Modeling Language (UML) diagrams serve these purposes.



DIANCE FERNANDEZ/RON SELLERS

WHAT IS UML?

Managed by the Object Management Group (OMG, www.omg.org), UML offers an industry-standard means of modeling complex systems. It is a unification of several leading object-oriented modeling approaches from the mid-1990s—primarily Booch notation (named for its creator, Grady Booch), James Rumbaugh's Object Modeling Technique (OMT), and Ivar Jacobson's Object-Oriented Software Engineering (OOSE).

Modeling—creating simplified representations of various aspects of a software system (prior to building it) that convey information about the system from a variety of perspectives—is a basic concept of object-oriented development. Just as a blueprint provides the basis that gives the architect, the contractor, and ultimately the homeowner a sense of the design and features of a house before construction begins, UML "bridges the gap between the world that customers, users, and analysts understand and that of the developers, engineers, and architects implementing the system," according to Guus Ramackers, Oracle principal product manager for UML and cochair of the OMG's UML 2.0 task force.

One of the key benefits of modeling is that the development process is more productive and the systems you develop will more likely meet requirements, says Ramackers. "UML helps you capture your requirements. It lets you define what the system should do, through use cases and activities, and it lets you graphically represent the objects that will do it." The result is greater developer productivity and a greater likelihood that the resulting system will perform as expected.

DIFFERENT DIAGRAMS

The UML 1.5 specification comprises nine different types of diagrams, many of which are specifically relevant to environments other than those involving J2EE enterprise business-level applications, says Ramackers. "For example, state-machine diagrams are particularly relevant to real-time applications but are much less relevant to modeling data-focused enterprise applications." The UML diagrams commonly used by Java and J2EE developers include the following:

Use-case diagrams model the interactions between a system and external entities. The external entities are depicted by tried-and-true stick figures (*actors* in the vernacular of UML); these are typically human users of the system but may be other processes or other systems. The *use case*, represented as an oblong oval, describes a system capability or service in a few words. The focus is on *what* the system does, not on how it does it. Use-case diagrams are typically accompanied by detailed textual definitions. Depending on the intended audience, use cases can be drawn and discussed at various layers of abstraction.

Class diagrams model the *structure* (rather than the behavior) of the class that makes up the system (or, more likely, the classes that make it up). These are depicted as simple boxes broken up into three general areas that show the name of the class, the class attributes, and the operations (*methods*, in Java). Class diagrams are widely used to define class structure in any programming language (Java, C++, and so on). The class diagram is used to show relationships (inheritance, association, dependency) among classes and interfaces in a hierarchy.

Activity diagrams show the processing flow of a business or system process, such as a workflow or integration process. Much as in the flowchart of structured programming, activities describe the processing rules behind a use case or the sequencing of sets of use cases. Activity diagrams typically incorporate multiple objects, but they may also be restricted to a single object to provide a high-level view of the states of an object and the methods that cause it to change state.

Sequence diagrams let you design the interaction among the objects in a system in the context of time—*when* the interaction occurs (when one action occurs in relationship to another action), rather than based on *what* occurs. Says Ramackers, "You'll have any number of objects in the diagram, and you can specify how the method calls are being made between objects and how the messages are being sent between the different objects."

UML AND ORACLE JDEVELOPER

Oracle has been involved with UML since the beginning, as one of the participants in developing the original UML 1.0 specification (published in 1999) as well as an active contributor to ongoing working group activities in the OMG.

Oracle's focus for UML support in Oracle JDeveloper is to provide the UML diagrams "that are most relevant to the J2EE platform and SOA [service-oriented architecture]," says Ramackers. Oracle JDeveloper supports the use-case, class, activity, and sequence diagrams just described.

Figure 1 shows a sequence diagram in Oracle JDeveloper 10g (10.1.3 preview). The key notation in a sequence diagram is the object *lifeline*—denoted by a box (bearing the name of the object and its type) with a line extending downward, with one or more vertically oriented narrow rectangles of different sizes positioned along the line. Each rectangle typically models the invocation of a method on the object. Actions occur along an object's lifeline in chronological order, from top to bottom.

Figure 1, for example, shows several object lifelines and their objects' interactions at various points in time, initiating messages to other objects or responding to messages from them.

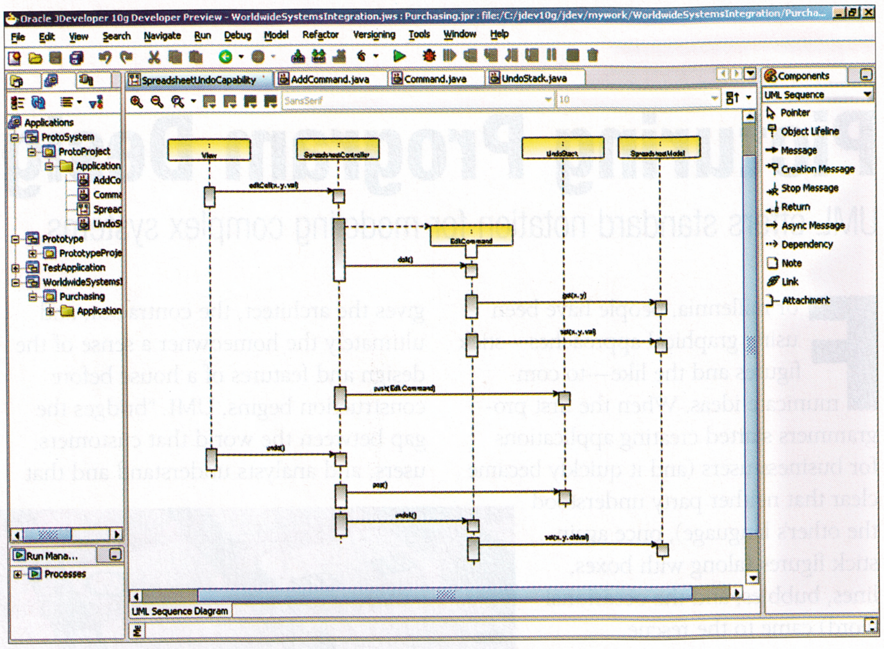


Figure 1: Sequence diagram in Oracle JDeveloper 10*g* (10.1.3 preview)

What's so great about this? It helps you model object interactions before you code, enabling discussion with peer programmers. Furthermore, with Oracle JDeveloper's trace sequence feature, you can test your logic before you start developing—the trace sequence automatically steps through the diagram, highlighting each element in the order in which messages will be sent, essentially enabling you to debug before you even begin coding. Once you start coding, you can tie the Sequence Modeler to the debugger to visualize debug traces.

SUPPORT FOR MORE UML MODELS

The online version of this article (at oracle.com/technology/oramag/oracle/05-may/o35industry.html) includes sections that describe Oracle JDeveloper's support for additional UML models and the new UML 2.0 standard, adopted in February 2005.

CONCLUSION

In the early days of enterprise software development, programmers often just plunged in and started coding—an approach that sometimes meant that millions of dollars were wasted on projects that couldn't be completed.

Today, it's a commonly accepted best practice to start talking about the requirements for any new system by using UML—for example, in terms of its various use cases and using class diagrams to express relationships among classes. UML provides an industry-standard way to model systems before building them and to talk about system features and capabilities by using an easy-to-understand, ageless graphical paradigm to effectively capture requirements and document system design. ■

Kelli Wiseth (kelli@alameda-tech-lab.com) is technology director at Alameda Tech Lab and Research Center (alameda-tech-lab.com).

nextSTEPS

READ online-only article content
oracle.com/technology/oramag/oracle/05-may/o35industry.html

more about UML
www.omg.org

DOWNLOAD Oracle JDeveloper 10*g*
oracle.com/technology/products/jdev

LEARN how to create a UML use case
oracle.com/technology/products/jdev/101/viewlets/101/UseCaseModeling_viewlet_swf.html

LOCATE UML and MDA partners
oracle.com/technology/products/jdev/htdocs/partners