JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2006**

Vol. 5, No. 7, September-October 2006

Review of MagicDraw UML® 11.5 Professional Edition[1]

Reviewed by Dave Neuendorf

UML design tools are intended to make OO analysis and design easier. All too often, though, the tools present obstacles to getting the work done, tempting a user to return to the casual flexibility of a whiteboard. During three months of frequent use, MagicDraw UML has proven to be an exception. It is a tool that cooperates in modeling without imposing unnecessary constraints.

REVIEW APPROACH

In using MagicDraw on a real project, I created many class and sequence diagrams, one state machine diagram and one activity diagram. For this review, I also reproduced various diagrams found in Scott Ambler's *The Elements of UML 2.0 Style* (a book which I highly recommend). My intention was to test the difficulty of following Mr. Ambler's style guidelines using MagicDraw.

The sections below, through "State machine diagrams," generally follow the sequence of the chapters in Ambler. A section is included when there is something interesting to say about how MagicDraw handles UML features. If there is no section for a given diagram type, no problems were encountered in creating or editing diagrams of that type using the software. The final sections cover other features of MagicDraw.

GENERAL GUIDELINES AND COMMON UML ELEMENTS

Ambler recommends that designers avoid crossing lines in diagrams. The layout functionality in MagicDraw can make it easier to route lines efficiently, though automatically laying out a whole diagram seldom produced an eye-pleasing result. Fortunately, layouts can be applied to selected elements in a diagram. Such targeted automatic layout was much more useful. When lines must cross, Ambler suggests using circuit diagram crossing symbols. MagicDraw has no provision for this.

[1]Vendor: No Magic, Inc; www.nomagic.com
Tested under MS Windows XP Professional

Cite this column as follows: Dave Neuendorf: "Review of MagicDraw UML® 11.5 Professional Edition", in Journal of Object Technology, vol. 5, no. 7, September-October 2006, pp. 115-118
http://www.jot.fm/issues/issue_2006_09/review8

Another of Ambler's suggestions is to show only those details that are necessary to communicate the point of a diagram. MagicDraw allows the user to delete operations and attributes from a class in a diagram without removing them from the underlying model. If any such elements are not depicted, the software automatically displays an ellipsis.

Ambler recommends that less important details be presented below or to the right of more important elements. For example, stereotypes should be shown at the bottom of a class box, or at the right of an operation signature. MagicDraw allows a great deal of flexibility in positioning various text elements on diagrams, but this does not extend to stereotypes, which are always at the top of a class box. The name of an interface is always below the interface itself when depicted lollipop-style: the opposite of Ambler's preference. On the other hand, MagicDraw is unusual in even providing lollipop notation for interfaces.

CLASS DIAGRAMS

There is little in Ambler's style recommendations for class diagrams that could not be accomplished using MagicDraw. Control over showing visibility of attributes and operations is simple. The software would accept operation parameters with both name and type, or just the type. There is a toolbar for creating association classes. The "property strings" called for in Ambler can easily be simulated using constraints. Tree routing similar relationships to a common class is simple using the "tree style layout" option in the layout menu.

Ambler shows one way of indicating "power types" on shared generalizations using a dashed line separator. MagicDraw provides a horizontal separator as well as a rectangular shape, but there is no vertical separator.

One feature that can save a lot of work is the ability to convert one element to another. For example, it is common to refactor a class to an interface, or *vice versa*. Whenever there is a possible conversion for a given element, the element's context menu has a "convert to…" selection. If the target element type does not support a sub-element, the incompatible element is deleted. For example, converting from a class to an interface automatically removes any attributes.

PACKAGE DIAGRAMS

With one exception, a designer could use MagicDraw to create package diagrams following Ambler's guidelines. There is no way to draw a relationship between an actor and a package in a use-case package diagram.

SEQUENCE DIAGRAMS

There is no direct way to include an actor in a sequence diagram. A workaround is to create the actor in a use-case diagram and copy it to the sequence diagram, then use the actor element's context menu to set "suppress content" to true.

MagicDraw makes it very easy to enter operation signatures as messages. Once the user starts typing the operation name, a context menu appears containing all matching operations. It is also simple to add a new operation to a class while creating a message to an object. This facilitates the design practice of working back and forth on a class diagram and a sequence diagram, keeping the two in sync.

STATE MACHINE DIAGRAMS

Drawing state machine diagrams works as one would expect. A very helpful feature is the ability to insert an action into the control flow by simply drawing or dragging it onto the link where it should be inserted. The same technique works also in activity diagrams.

JAVA CODE GENERATION AND REVERSE ENGINEERING

Java code engineering was not extensively tested for this review, but in a few conversions between existing code and generated class diagrams the software worked as expected. Code generated from a class diagram was formatted as specified in the project settings. In a round trip test, MagicDraw did not disturb changes made in an external editor.

DOCUMENTATION

The User Manual for MagicDraw is provided in the form of a 488 page PDF file. Since search is limited to the simple facility provided by Adobe Acrobat Reader, it is not always easy to find needed information. The English in the document is awkward, though usually readable enough.

TECH SUPPORT

No Magic tech support gave very good service in support of this review. Of course, whenever a writer contacts tech support during the course of a review, it is possible that the vendor may be providing special treatment in order to make a good impression. A better indication of the actual quality of support is the response to a tech support issue which arose during a real-world project before this review. In the first week of using MagicDraw, we had all sorts of problems getting diagrams to plot correctly on a large HP plotter. The No Magic support engineer noted that MagicDraw was shipped with a Sun

Java 1.5 VM which has known problems with large format HP plotters, and suggested reverting to a Java 1.4 VM. This solved the problem.

Even more impressive than this fast response was the fact that it came from multiple sources at No Magic. It appeared that development engineers were monitoring the tech support threads and passing along their knowledge as well. That demonstrates an unusually strong commitment to customer support.

CONCLUSION

MagicDraw is a serious contender among UML design tools. While providing extensive UML 2.0 support, it defers to the user's judgment when stylistic issues arise.

MagicDraw UML is strongly recommended.