JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2003**

Vol. 2, No. 6, November-December 2003

Book Review

The Object Constraint Language Second Edition, Getting Your Models Ready for MDA

by Jos Warmer and Anneke Kleppe, Addison-Wesley, Boston, MA, 2003. 206 pp., $39.99(paper). ISBN 0-321-17936-6

Reviewed by Charles Ashbacher

As a mathematician, I have always been drawn to the more formal programming methods of the Object Constraint Language (OCL). With it, you can place precise mathematical descriptions of what must be true before the code executes and what is guaranteed to be true after the code has run. Given the concise nature of mathematical notation, the OCL expressions also can replace many times their text in comments. The combination of Unified Modeling Language (UML) and OCL is a powerful one, allowing you to precisely describe the actions that code is expected to perform. However, the OCL is unlike other formal mathematical languages in that it must change in response to new ways of creating code.

One of the newest ways to design software systems is to use Model Driven Architectures (MDAs). An MDA is a high level framework that describes how models can be translated from language to language. Since the purpose is to allow for the translation to be done by machine, it is necessary for all restrictions to be written in a clear, unambiguous manner. Currently, the MDA process is divided into three steps:

1. The Platform Independent Model (PIM), which is at the highest level of abstraction and is independent of any specific technology.
2. The PIM is then translated into one or more Platform Specific Models (PSMs), where platform specific features are incorporated into each PSM.
3. Each PSM is then converted into code to be run on the specific platform.

Cite this book review as follows: Charles Ashbacher: Book review of “The Object Constraint Language Second Edition, Getting Your Models Ready for MDA”, in *Journal of Object Technology*, vol. 2, no. 6, November-December 2003, pp.139-140.
http://www.jot.fm/books/review11

When I first encountered the fundamentals of MDAs, my skepticism meter rose to a high level. Even with the precision of UML, creating accurate models is very hard, and creating appropriate code from those models is probably harder. Sure, it is possible to create the constructors, setters and getters and other simple components of a class automatically, but code is so complex that it is difficult to look at a model and know precisely what it is supposed to do.

After reading this book, the level of my skepticism meter has dropped quite a bit. From the presentation, it is easy to see how important OCL is to the accurate rendition of code from models. In fact, it is hard to believe that MDA could possibly succeed without OCL being heavily incorporated into the models.

The opening chapter of the book is a brief explanation of MDAs and how UML and OCL are used in combination to create the models. A solid introduction to the MDA way of doing things, it also describes the foundations, but not the specifics of OCL. It is assumed that the reader is familiar with the UML. In chapter two, the OCL is described using an example of a customer loyalty program. This is an excellent example, as it is easy to understand and allows most of the basic expressions of the OCL to be used.

Chapter three is where the specifics of how the OCL is used in models are covered. While the examples are well done, one does not skim this chapter and understand it. The combination of diagrams and constraints must be read with a great deal of care in order to understand them. I was impressed with this approach, as the authors did not sacrifice accuracy of rendition for simplicity of understanding. The coverage of chapter four is in how the OCL is implemented and chapter five describes how the structures of the OCL are used in the creation of MDA models. This is where the UML and OCL metalanguages are merged to describe the functions of software. Once again, be prepared to spend some time in examining the diagrams in great detail. I found myself reading and rereading some of the examples before I achieved an acceptable level of understanding.

The final sections of the book form an OCL reference manual, where many of the expressions of the OCL are demonstrated in the appropriate context. In general, they are demonstrated in circumstances independent of an MDA model, which is an appropriate simplification.

There are reasons to believe that the MDA approach to software is not yet mature enough to be trusted. Nevertheless, the advantages of using it in any way are enormous, and it cannot be used without a way to accurately express the environment in which segments of code are to run. The OCL provides a way to describe the environment and from this book you can learn how the UML and the OCL can be combined to create models with the required degree of precision necessary for the automatic translation of an MDA into code.