Component-Based Product Line Engineering with UML

Addison-Wesley, 2001
ISBN: 0-201-73791-4
Cover price: US$50.00
464 pages

First, a warning: The title of this book is misleading. If you're hoping for mainstream, tried-and-true process guidance, using standard notations supported by commercially available tools, then this is the wrong book. The following may help set your expectations for this book:

- The authors are researchers describing a theoretical approach (the KobrA methodology); they are not practitioners deriving best practices proven on real projects.

- The authors' proposed notation is not standard Unified Modeling Language, or UML,™ but rather a custom notation loosely based on UML. There is neither explicit support for this notation by commercially available tools nor any suggestion that this notation will ever be part of standard UML.

Don't get me wrong, though. This book is a worthwhile read, as it provides many interesting and important insights into component-based and product-line engineering concerns. The authors point out genuine issues and ambiguities in the current UML, and some of their proposed solutions are intriguing.

The KobrA (derived from Komponentenbasierte Anwendungsentwicklung, which is German for "component-based application development") methodology is based on a number of principles. Most of these -- such as parsimony, encapsulation, and locality -- are restatements of generally accepted component software engineering principles for keeping things simple, separating concerns, and minimizing coupling. Where KobrA is a bit more radical is in its application of what the authors call uniformity.

The uniformity principle reflects a major goal of KobrA: "to avoid the
feature overload found in many other methods." Every behavior-rich element of a system is a Komponent. Thus, the UML's subsystems, packages, behavior-rich classes, components, and even whole systems, all become Komponents. Since the term is so broad, KobrA uses qualifiers to distinguish between different kinds of Komponents:

- Instance vs. type
- Specification vs. realization

I find this uniformity quite attractive because it allows a system to be described as a Komponent and decomposed recursively into a hierarchy of Komponents. All behavior is described in terms of interacting Komponents, so you don't have to decide up front if the behavioral element is a class, a subsystem, or a package. The terminology is simple and consistent.

Nevertheless, I see great value in the software industry adhering to standardized notations to achieve effective communication between developers across projects and organizations. Although KobrA is a legitimate challenge to the UML that aims at greater simplicity and unity, the working group responsible for the UML is also driven by other important concerns, such as how to maintain backward compatibility and how to extend the UML's modeling capabilities. When they release the new UML 2.0, we will see how well they have balanced these concerns and whether they were able to address some of the concerns about complexity that led Atkinson et al. to develop KobrA.

The rest of their book is a broad mix of software engineering guidance for component-based development, which they illustrate using KobrA notation. It is weak in some areas, such as bridging requirements and design, and rich in others, such as component specification. Some of the guidance is interesting and instructive, such as the thirty-six pages of mathematically rigorous description of component configuration management. Much of the guidance is theoretical, without supporting tools or reports of commercial experience -- so proceed with care.

- Bruce MacIsaac
Senior Software Developer
Rational Software
IBM Software Group

For more information on the products or services discussed in this article, please click here and follow the instructions provided.
Thank you!