Editor: Warren Keuffel ■ wkeuffel@computer.org

On Notations and Their Use

Fernando Berzal

Design Methods for Reactive Systems *by Roel J. Wieringa, Morgan Kaufmann, 2003, ISBN 1-55860-755-2, 500 pp., US$64.95.*

According to Roel Wieringa, "any creative decision about a product is a design decision," so decisions made during the design process should be documented in the product specification. *Design Methods for Reactive Systems*, therefore, is mainly about techniques for specifying software systems and their design decisions, while taking into account the systems environment. In fact, as Robert Glass has pointed out (*Facts and Fallacies of Software Engineering*, Addison-Wesley, 2002), missing requirements, usually in the form of unstated assumptions about the environment, are the hardest requirements errors to correct.

Specification techniques

Wieringa devotes much of the book to analyzing specification techniques, describing their notations, explaining their semantic details, and offering guidelines for their correct use. Although technically sound, some of his opinionated views could be considered controversial. Most of Wieringa's guidelines are sagely devised. However, heuristic as they might be, they should always be taken with a grain of salt.

The book arranges specification techniques into four groups. Written in laymen's terms, the first group, Function Notations, is useful for reaching an agreement with the customer. This group includes the mission statement, the function refinement tree, and service descriptions. The mission statement is the highest-level description of the system under design (SuD) as related to business goals. This is then refined into a function refinement tree or indented list ("a shopping list of things the system must do"). This tree or list includes items the user considers valuable: the use cases described by service descriptions.

The second group, Entity Notations, describes the subject domain. In describing these, Wieringa employs entity-relationship diagrams, complemented by a dictionary of key terms. The book's E/R diagrams are drawn using a notation similar to the Unified Modeling Language (UML) to describe system configurations, so you can't consider them data models in the usual sense. In fact, some of the diagrams might even be irksome to database-oriented readers, if they try to interpret them as database conceptual designs.

The third group, Behavior Notations, focuses on transforming stimuli into responses, with accompanying state changes in the SuD. The book analyzes event lists (describing events and their effects), decision tables, state transition tables, and state transition diagrams. It covers both standard Mealy diagrams and statecharts, which include state reactions as

ONLINE REVIEWS

- **"Programming Programs with C++ Templates"** by Ed Harcourt

  A review of *C++ Templates: The Complete Guide* by David Vandevoorde and Nicolai M. Josuttis.

- **"Measuring What's Measurable"** by Harekrishna Misra

  A review of *IT Measurement: Practical Advice from Experts* by the International Function Point Users Group.

www.computer.org/software/bookshelf

0740-7459/04/$20.00 © 2004 IEEE

Moore diagrams, state hierarchies, and parallelism. Once Wieringa explains syntactic details, he thoroughly examines the semantic differences between Statemate and UML; it's possibly the book's most elaborate and outstanding part.

Finally, the last group, Communication Notations, concentrates on the information exchange between systems and complements the view offered by Behavior Notations. Wieringa presents his own variant of dataflow diagrams and then generalizes it into a statechart-like notation that he calls "communication diagrams." Their only contribution is the ability to include types in standard DFDs. These diagrams can be accompanied by allocation and flowdown tables, or traceability tables, which help improve requirements traceability.

With respect to this last group, one thing I find particularly inappropriate in the use of hierarchical diagrams is repeating diagram elements at different levels of the diagram hierarchy. Although the author defends the value of channel addressing, he unnecessarily restricts hierarchical DFDs to destination addressing, therefore breaking diagram encapsulation.

Methodologies: Putting it all together

The book's final part describes three specification methodologies. First is postmodern structured analysis as a Yourdon-style variant of structured analysis that uses "a more elaborate representation of the context." Second is Statemate, along with its hierarchical activity diagrams (one more variant of DFDs) and its well-defined execution algorithm, whose formalization level makes it suitable for automation. Finally, Wieringa introduces a UML subset with a singular variant—that is, using communication diagrams to ensure coherence between "static structure diagrams" (object and class diagrams) and "behavior descriptions" (statecharts). Wieringa ends his methodology discussion with NYAM (Not Yet Another Method), where he acknowledges that the proper notation choice is situation dependent. In fact, he recommends notation subsets, using boxing categories as a metaphor—from flyweight to heavyweight—interpreted as giving more or less emphasis to the project front end.

I must acknowledge that *Design Methods for Reactive Systems* covers a wide range of specification techniques and offers valuable insight, although it deviates somewhat from terminology and notation standards. For instance, Wieringa dismisses some common terms because of their possible connotations. Moreover, he introduces his own share of new notations and variants. As Karl Wiegers said, we don't need more new models and notations; we need more practitioners to effectively apply known techniques ("Read My Lips: No New Models!" *IEEE Software*, Sept./Oct. 1998).

Nevertheless, Wieringa's book can be useful for software developers. If you're looking for ways to improve your analysis skills, you might consider classics such as *Rethinking Systems Analysis and Design* by Gerald Weinberg (Dorset House, 1988). But, if you want to fully understand the underpinnings of existing specification notations, this book can be a good place to start.

Fernando Berzal is an assistant professor in the Department of Computer Science and Artificial Intelligence at the University of Granada and cofounder of iKor Consulting (www.ikor.org). He is currently a visiting scholar at the University of Illinois at Urbana-Champaign. Contact him at berzal@acm.org.

Clichés Can Be Both Tiring and Helpful

Diomidis Spinellis

More Secrets of Consulting: The Consultant's Tool Kit *by Gerald M. Weinberg, Dorset House, 2002, ISBN 0-932633-52-8, 202 pp., US$33.95.*

I enthusiastically devoured a library copy of Gerald Weinberg's *The Psychology of Computer Programming* (Van Nostrand Reinhold, 1971) more than 15 years ago. The book was out of print for a long time, so I patiently waited 10 years for its silver anniversary edition to appear (Dorset House, 1998) to secure a copy for my bookshelf. I reread parts of *PoCP* while preparing this review and still recommend it highly: a landmark work when it appeared, *PoCP* still remains a must-read for every software professional.

Sadly, *More Secrets of Consulting* isn't in the same league. Where *PoCP* offers original insights, out-of-the-box thinking, and perceptive observations, *MSoC* offers many clichés formulated as "laws" and an extremely tiring tendency to baptize every idea with a proper name. Thus, a single page bombards us with references to the Helpful Model, the Mirror, Carl's Constructive Corollary, the Big Picture, Master of the Mirror, and Kenny's Law of Auto Repair. Furthermore, the valuable annotated bibliography appearing after each *PoCP* chapter has given its place to tens of footnotes promoting Weinberg's other books and seminars. The bibliography at the end of *MSoC* contains more books authored or coauthored by Weinberg (14) than all the other references combined (12).

I am, however, probably judging *MSoC* against an impossibly high standard. The economic downturn made many professionals turn to consulting for a living, and consultants, typically working in isolated environments, need all the help they can get. *MSoC* employs the metaphor of a toolkit to present qualities a consultant should have or strive to cultivate. So, for example, we read how we can use the Wisdom Box to select the right assignments, the Golden Key to open up new areas of learning and practicing, the Courage Stick to try new things, the Mirror to see ourselves, and the Oxygen Mask to lead a balanced life. The contrived names aside, Weinberg still has a lot to teach us, and the advice he dispenses is certainly worth the book's price.

Diomidis Spinellis is an associate professor in the Department of Management Science and Technology at the Athens University of Economics and Business. Contact him at dds@aueb.gr.