

Foreword

Ho-hum, just what we need, another book about the UML. Doesn't *UML Distilled* cover everything a developer really needs to know about the UML? *UML Distilled* covers all of the UML diagram types; this book doesn't even cover all of the different diagrams.

Hang on a minute, this book covers all of the diagrams I use on real projects. It omits all of the diagrams I have to remind teams not to use. Maybe Robert Martin has written something that will surpass *UML Distilled*...

Uh-oh, first glances can be deceptive. This is a great book that deserves to become a classic. *UML for Java Programmers* is the first book I have ever read that treats the UML as a tool for programmers to help them make their day job easier. A welcome change from all of the UML books that assume you want to become a language lawyer.

I really like the way that this book focuses on the specification and implementation level use of the UML, to communicate precise, unambiguous descriptions of a proposed design and existing code. Uncle Bob promotes the idea that an occasional UML diagram can save time by prescreening ideas before going to the expense of writing the code and that sometimes a UML diagram can be a great way of explaining how a part of an existing application works.

This is a book that explains how to use the UML on real projects, one that focuses on getting practical value out of using the UML. Simple things that make a big difference, like using UML diagrams to choose between alternatives, explaining design ideas on a white board using the UML, and the need to erase diagrams once they have achieved their purpose.

Having seen all too many shelfware CASE tools, I appreciate Robert's warnings about CASE tools. Personally I'd rather see organizations invest in decent meeting rooms with photocopying white boards rather than waste money on CASE tools. Yes, CASE tools can be kind of cool, but there are many other investment opportunities that will give much better payback in terms of developer productivity.

This book challenges developers to understand the value that can derive from drawing UML diagrams, and encourages developers to push back against the language lawyers and the UML police who encourage the inappropriate use of precision in UML diagrams.

Uncle Bob has done a great job in not only showing how to use the UML effectively, but in also explaining how to recognize when the UML diagrams are depicting bad design ideas. After all you can have a beautiful diagram that would make a language lawyer happy, but if the design stinks, you need to fix it.

The design guidelines and heuristics in this book transform it from a simple guide to the UML to a great book on how to do OO design. This book demonstrates that there is much more to OO design than simply drawing UML diagrams. The diagrams do not really make much of a difference; what matters is the critical thinking about the consequences of each of the design decisions. Yes, the diagrams can make it easier to see these conse-

quences, but what really matters is that people know how to look for and deal with these consequences.

This book deserves to become a classic because it exposes the dirty little secret of software development: good design evolves out of many iterations of hard work on a problem.

This also explains why this book uses Java. Java has been through enough iterations that it is now useful. We have been through the hype and come out the other side. Uncle Bob does a great job of showing lots of straightforward Java code that is a massive step beyond the normal, toy code samples that many books show.

The combination of Java and the UML works well to show what good OO design looks like. Uncle Bob, many thanks for a great book.

Pete McBreen
author, *Software Craftsmanship*
April 2003