1. Record Nr. UNINA9910458107903321

| | |
|---|---|
| Autore | Robinson David |
| Titolo | Aspect-oriented programming with the e verification language [[electronic resource] ] : a pragmatic guide for testbench developers / / David Robinson |
| Pubbl/distr/stampa | Amsterdam ; ; Boston, : Elsevier/Morgan Kaufmann, c2007 |
| ISBN | 1-281-03832-6<br>9786611038328<br>0-08-055155-6 |
| Edizione | [1st edition] |
| Descrizione fisica | 1 online resource (265 p.) |
| Collana | The Morgan Kaufmann series in systems on silicon |
| Disciplina | 005.1/17 |
| Soggetti | Object-oriented programming (Computer science)<br>Electronic books. |
| Lingua di pubblicazione | Inglese |
| Formato | Materiale a stampa |
| Livello bibliografico | Monografia |
| Note generali | Description based upon print version of record. |
| Nota di bibliografia | Includes bibliographical references (p. [239]-240) and index. |
| Nota di contenuto | Front cover; Aspect-Oriented Programming with the e Verification Language; Copyright page; Acknowledgments; Table of Contents; Foreword; Preface; About Verilab; Chapter 1. Introduction to Aspect Oriented Programming (AOP); 1.1. What are aspects? - Part I; 1.2. Why do I need aspects? What's wrong with crosscutting concerns?; 1.3. Surely OOP doesn't have any problems?; 1.4. Why does AOP help?; 1.5. Theory vs real life - What else is AOP good for?; 1.6. What are aspects? - Part II; Chapter 2. AOP in e; 2.1. How do I extend a class?<br>2.2. How do I extend a class for multiple values of a determinant?2.3. How do I extend a type?; 2.4. How do I introduce a new noncoverage member to a class?; 2.5. How do I introduce a coverage group to a class?; 2.6. How do I extend a coverage group?; 2.7. How do I change the behavior of a method?; 2.8. How do I limit the scope of my extensions?; 2.9. Using return in method advice; 2.10. Controlling the order of method extension calls; Chapter 3. Using AOP to Organize Your Code; 3.1. A word about style; 3.2. What aspects do I want to use?; 3.3. Mapping aspects to files<br>Chapter 4. Creating Flexible CodeChapter 5. Creating Pluggable Code; 5.1. The extendable case statement; 5.2. The factory pattern; Chapter 6. Improving Your Productivity; 6.1. Shifting the power; 6.2. Dealing |

with broken code; 6.3. Handling workarounds; 6.4. Reducing and deferring class complexity; 6.5. Adding problem-specific functionality; 6.6. Reducing the OOP-induced overhead; Chapter 7. AOP in Action; 7.1. Creating a class with a selectable algorithm; 7.2. Creating a configuration interface for an eVC; 7.3. Using aspects to create a layered verification environment
7.4. Creating reusable layered sequences7.5. Testing your verification environment; 7.6. Debugging using AOP; 7.7. Encapsulating tests; Chapter 8. Analysing e Code; 8.1. The e toolkit; 8.2. Finding class declarations and extensions; 8.3. Finding the class inheritance hierarchy; 8.4. Finding the determinants used by a class; 8.5. Finding method declarations and extensions; 8.6. Finding field declarations; 8.7. Finding event declarations; 8.8. Finding enumerated type declarations and extensions; 8.9. How do I find where a value is added to a type?
8.10. Finding cover group declarations and extensions8.11. Finding the source of a message in the log file; 8.12. Finding aspects; Bibliography; Epilogue; Index; A; C; D; E; F; H; I; J; M; N; O; P; R; S; U; W

Sommario/riassunto

What's this AOP thing anyway, really-when you get right down to it-and can someone please explain what an aspect actually is?Aspect-Oriented Programming with the e Verification Language takes a pragmatic, example based, and fun approach to unraveling the mysteries of AOP. In this book, you'll learn how to: Use AOP to organize your code in a way that makes it easy to deal with the things you really care about in your verification environments. Forget about organizing by classes, and start organizing by functionality, layers, components, protocols, functional coverage, c