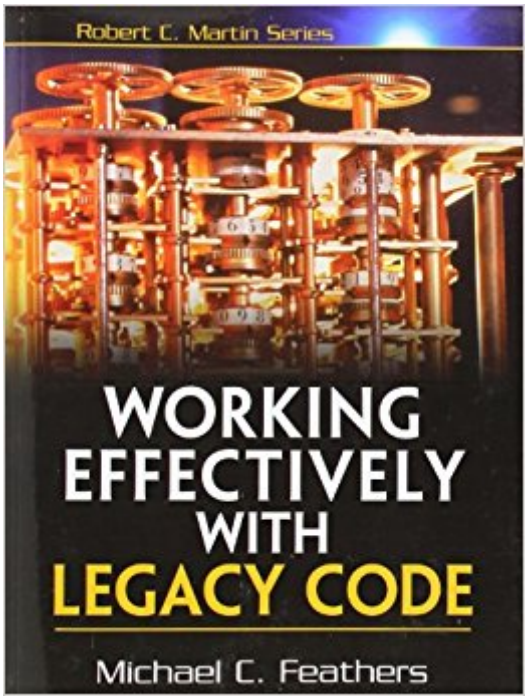
Working Effectively With Legacy Code



Synopsis

Get more out of your legacy systems: more performance, functionality, reliability, and manageability Is your code easy to change? Can you get nearly instantaneous feedback when you do change it? Do you understand it? If the answer to any of these questions is no, you have legacy code, and it is draining time and money away from your development efforts. In this book, Michael Feathers offers start-to-finish strategies for working more effectively with large, untested legacy code bases. This book draws on material Michael created for his renowned Object Mentor seminars: techniques Michael has used in mentoring to help hundreds of developers, technical managers, and testers bring their legacy systems under control. adding features, fixing bugs, improving design, optimizing performance Getting legacy code into a test harness Writing tests that protect you against introducing new problems Techniques that can be used with any language or platform--with examples in Java, C++, C, and C# Accurately identifying where code changes need to be made Coping with legacy systems that aren t object-oriented Handling applications that don t seem to have any structureThis book also includes a catalog of twenty-four dependency-breaking techniques that help you work with program elements in isolation and make safer changes.

Book Information

Paperback: 456 pages
Publisher: Prentice Hall; 1 edition (October 2, 2004)
Language: English
ISBN-10: 0131177052
ISBN-13: 978-0131177055
Product Dimensions: 7 x 1.1 x 9.1 inches
Shipping Weight: 1.5 pounds (View shipping rates and policies)
Average Customer Review: 4.5 out of 5 stars 112 customer reviews
Best Sellers Rank: #34,052 in Books (See Top 100 in Books) #10 inÃ Â Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Testing #31 inÃ Â Books > Textbooks > Computer Science > Software Design & Engineering #70 inÃ Â Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Software Development

Customer Reviews

Get more out of your legacy systems: more performance, functionality, reliability, and manageability

Is your code easy to change? Can you get nearly instantaneous feedback when you do change it? Do you understand it? If the answer to any of these questions is no, you have legacy code, and it is draining time and money away from your development efforts. In this book, Michael Feathers offers start-to-finish strategies for working more effectively with large, untested legacy code bases. This book draws on material Michael created for his renowned Object Mentor seminars: techniques Michael has used in mentoring to help hundreds of developers, technical managers, and testers bring their legacy systems under control. The topics covered include Understanding the mechanics of software change: adding features, fixing bugs, improving design, optimizing performance Getting legacy code into a test harness Writing tests that protect you against introducing new problems Techniques that can be used with any language or platform—with examples in Java, C++, C, and C# Accurately identifying where code changes need to be made Coping with legacy systems that aren't object-oriented Handling applications that don't seem to have any structure This book also includes a catalog of twenty-four dependency-breaking techniques that help you work with program elements in isolation and make safer changes. >

MICHAEL C. FEATHERS works for Object Mentor, Inc., one of the world's top providers of mentoring, skill development, knowledge transfer, and leadership services in software development. He currently provides worldwide training and mentoring in Test-Driven Development (TDD), Refactoring, OO Design, Java, C#, C++, and Extreme Programming (XP). Michael is the original author of CppUnit, a C++ port of the JUnit testing framework, and FitCpp, a C++ port of the FIT integrated-testing framework. A member of ACM and IEEE, he has chaired CodeFest at three OOPSLA conferences. >

I found Mr. Feathers to be a good author with a very down to earth writing style that made the early chapters easy to read. You may not want to read this book "cover to cover" as I did though. The last few chapters are very dry and technical. The latter half of the book is really more of a reference to whip out when you encounter a problem that you’re not sure how to solve. I highly recommend this to anyone who is working with, or about to work with, a legacy code base.

Great book. Provides step-by-step instructions on how to deal with code to get it better. Provide specific advice for every situation and rules that help you recognize the various situations and appropriate approach. Most of all, it helps you get over the desperation of having to deal with old

unwieldy code and shows you how, even in such situation you can make small improvements that over time would get you to a better place without having to stop what you are doing for a year to get there. Really useful.

Changed the way I think about software design. A must read.

I've been using many of these techniques ad hoc for years, but it's so much easier to explain when things that have names.The author explains that he considers "legacy code" to be any code without tests. This is, in my experience, almost entirely true.Much of the content of the book consists of specific techniques for incrementally getting code not designed for testing under test while adding features and fixes. Lots of gold here, and not buried deep.

This book covered everything that I expected for getting code that was previously not being automatically tested under test. In particular, the focus on making your designs more understandable by making them more testable is right on.The only shortcoming that I notice is in newer test technologies that can aid this process. In particular, mocking frameworks have come a long way. They can serve very effectively to break dependencies at test time, without removing them in production and without changing production code.

This is one of the most common books that I buy for teams that I'm helping manage and train. Not only does it help you work with legacy code but it helps instill type of culture that allows and promotes creating manageable' and testable code. I highly recommend every software engineer read this book. The code is in Java but the concepts easily translate to other languages.

One of the best unit testing books out there, especially when you sit with an untested code-base. You don't require a legacy code base to learn from this book, some of the "coolest" refactoring tricks, while keeping your tests running. Even if you're code base has zero tests, and your work environment gnarls at your attempts for change, this book lets you take courage that mountains can turns into little mole heaps.

A great reference for writing unit tests. I don't refer to it too often, but it is great to have as a reference next to my desk.

Working Effectively with Legacy Code Subaru Legacy & Forester: Legacy 2000 thru 2009 - Forester 2000 thru 2008 - Includes Legacy Outback and Baja (Haynes Repair Manual) Burn for Me: A Hidden Legacy Novel (Hidden Legacy series, Book 1) (Hidden Legacy Novels) 2012 International Plumbing Code (Includes International Private Sewage Disposal Code) (International Code Council Series) Building Code Basics: Commercial; Based on the International Building Code (International Code Council Series) Child-Centered Practices for the Courtroom and Community: A Guide to Working Effectively with Young Children and Their Families in the Child Welfare System The Sharing Knife, Vol. 2: Legacy (Legacy (Blackstone Audio)) The Virgin's Spy: A Tudor Legacy Novel (Tudor Legacy Trilogy Book 2) The Virgin's Daughter: A Tudor Legacy Novel (Tudor Legacy Trilogy Book 1) Deep in the Heart: Lone Star Legacy, Book 1 (LoneStar Legacy) Legacy of Dragonwand: Book 1 (Legacy of Dragonwand Trilogy) Legacy of Dragonwand: Book 2 (Legacy of Dragonwand Trilogy) Legacy of Dragonwand: Book 3 (Legacy of Dragonwand Trilogy) Girls Day Out: A Syrena Legacy Story (The Syrena Legacy) Legacy Lost: A Tor.Com Original (The Syrena Legacy) Learning to Labor: How Working Class Kids Get Working Class Jobs Working Hard, Working Poor: A Global Journey Working With Independent Contractors (Working with Independent Contractors: The Employer's Legal Guide) Working Length Determination: A Milestone in Endodontics: Comparative role of radiographs and electronic apex locator in working length determination Stop Physician Burnout: What to Do When Working Harder Isn't Working