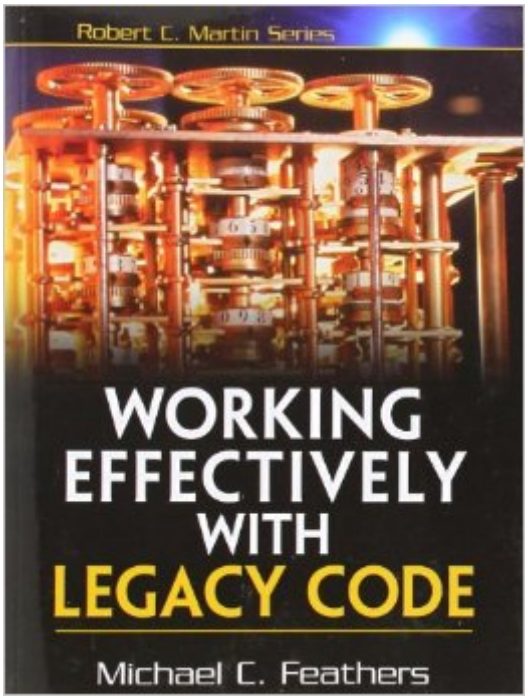
Working Effectively With Legacy Code



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: 6.9 x 1.1 x 9 inches
Shipping Weight: 1.5 pounds (View shipping rates and policies)
Average Customer Review: 4.6 out of 5 starsÂ Â See all reviewsÂ (104 customer reviews)
Best Sellers Rank: #52,427 in Books (See Top 100 in Books) #22 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Testing #59 inÂ Books > Textbooks > Computer Science > Software Design & Engineering #133 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Software Development

Customer Reviews

The average book on Agile software development describes a fairyland of greenfield projects, with wall-to-wall tests that run after every few edits, and clean & simple source code.The average

software project, in our industry, was written under some aspect of code-and-fix, and without automated unit tests. And we can’t just throw this code away; it represents a significant effort debugging and maintaining. It contains many latent requirements decisions. Just as Agile processes are incremental, Agile adoption must be incremental too. No more throwing away code just because it looked at us funny.Mike begins his book with a very diplomatic definition of "Legacy". I’l skip ahead to the undiplomatic version: Legacy code is code without unit tests.Before cleaning that code up, and before adding new features and removing bugs, such code must be de-legacified. It needs unit tests.To add unit tests, you must change the code. To change the code, you need unit tests to show how safe your change was.The core of the book is a cookbook of recipes to conduct various careful attacks. Each presents a particular problem, and a relatively safe way to migrate the code towards tests.Code undergoing this migration will begin to experience the benefits of unit tests, and these benefits will incrementally make new tests easier to write. These efforts will make aspects of a legacy codebase easy to change.It’s an unfortunate commentary on the state of our programming industry how much we need this book.

Martin Fowler’s book on Refactoring showed us how to improve the design of our code. We learned to make changes safely, by taking small, rote steps, and by ensuring that we ran our tests after each small change. But what if we’re working on the typical ugly system with no tests? In Working Effectively With Legacy Code, Michael Feathers tackles the problem that most of us end up dealing with.Feathers does an excellent job of articulating the problems and scenarios, using clear examples from C, C++, Java, and C#. Many of the code examples look a lot like real examples I come across all the time--they don’t appear to be fabricated.Working Effectively With Legacy Code contains a catalog that provides a wealth of solutions. The catalog shows how to resolve concerns like, "I’m changing the same code all over the place" and "how do I safely change procedural code?"The book is highly entertaining and comes across as a conversation with a really sharp, really patient guru developer. Often, it’s a chore to slog through code-heavy books. But Feathers manages to keep my attention with interesting stories, loads of examples, and well-written text.I think that Working Effectively With Legacy Code is an important book. The vast majority of existing code presents the classic catch-22: you can’t change it safely because it doesn’t have tests, and you can’t write tests without changing it to easily support testing. It’s not an easy problem, and most people will give you high-level ideas for solving it. Feathers is the first person to dig deep and present a wealth of knowledge and insight on the problem, all in one place. I’ll be pulling this book from my shelf for years to come.

"Working Effectively with Legacy Code" is a very valuable resource. The author defines "legacy code" as "code without tests." It doesn't matter whether the code was written last week or ten years ago. There is more emphasis on old code that nobody understands, mainly because it is messier and harder to work with.The examples in the book are mainly in C, C++ and Java, but there are a couple in C# and Ruby. While it is essential to know one of these languages, the author provides enough information to understand the others. When a technique only applies to a certain language, it is clearly indicated.The author shows how different diagrams can help you learn how to understand code. In addition to UML, there are dependency and effect sketches. The author uses these to show how to think about understanding and refactoring. Other tools, such as refactoring browsers and mocks are explained.Speaking of refactoring, there are "dependency breaking techniques" (aka refactorings) with step-by-step instructions (Martin Fowler style) throughout the book. There are also explanations of why patterns and design rules exist. Most importantly, there are lots and lots of cross-references and an excellent index.Working with legacy code isn't fun, but this book helps make it as painless as possible. With the split emphasis between psychological/understanding/techniques and refactoring, this book is both a great read and an excellent reference.

I work at a decent sized telecommunications company. We have legacy code written in C that is over 1 million lines of code. Some of the code was written as far back as 1988. Needless to say, we didn't follow TDD and there are not a lot of unit tests. We have recently increase the number developers to add features to this code base and I was hoping that this book would help.We've been doing a "techincal book club" for a while as part of continuous training. I've had about 20 engineers reading this book a few chapters a week and discussing them. Most of the reviews from the group have been negative. Hard to read, annoying editorial errors (duplicate text on following pages), and not really getting a lot out of it. The main problem is that our system is not using an object oriented language so a lot (most) of the techniques are not relevant.At first I thought it was just me, but as I asked the other engineers, there was a lot of concensus, even from engineers that have worked on Java/C++ projects in the past.I picked this book because of the following taglines on the back of the book:* Techniques that can be used with any language or platform-with examples in Java, C++, C, and C#* Coping with legacy systems that aren't object-orientedThere is one small section on non-object oriented code. It basically says that you should slowly migrate to an object oriented language.Anyway - we've stopped reading the book. If you're code is already object

oriented, this is probably a great book. If it's not, I wouldn't bother. Instead pick up a differnt book on how to migrate the code to an object oriented language.

Working Effectively with Legacy Code 2012 International Plumbing Code (Includes International Private Sewage Disposal Code) (International Code Council Series) The Sharing Knife, Vol. 2: Legacy (Legacy (Blackstone Audio)) Fighting for Total Person Unionism: Harold Gibbons, Ernest Calloway, and Working-Class Citizenship (Working Class in American History) Learning to Labor: How Working Class Kids Get Working Class Jobs Working With Independent Contractors (Working with Independent Contractors: The Employer's Legal Guide) The 5 Love Languages of Children: The Secret to Loving Children Effectively The 5 Love Languages of Teenagers: The Secret to Loving Teens Effectively Internal Cleansing : Rid Your Body of Toxins to Naturally and Effectively Fight Heart Disease, Chronic Pain, Fatigue, PMS and Menopause Symptoms, and More (Revised 2nd Edition) Writing That Works; How to Communicate Effectively In Business Official TOEIC Vocabulary 3000: Become a True Master of TOEIC Vocabulary...Quickly and Effectively! Effectively Managing and Leading Human Service Organizations (SAGE Sourcebooks for the Human Services) People Tactics: Become the Ultimate People Person - Strategies to Navigate Delicate Situations, Communicate Effectively, and Win Anyone Over (People Skills) People Tactics: Strategies to Navigate Delicate Situations, Communicate Effectively, and Win Anyone Over The SAP Green Book: A Business Guide for Effectively Managing the SAP Lifecycle REAL ESTATE: A Guide for First Time Agents to Effectively Grow Your Business From Nothing to a Sustainable Growing Career (Beginner's Guide, Career Management, Lead Generation, Real Estate Investors) Drum Class Method, Vol 1: Effectively Presenting the Rudiments of Drumming and the Reading of Music Drum Class Method, Vol 2: Effectively Presenting the Rudiments of Drumming and the Reading of Music The Crowdfunding Myth: Legally and Effectively Raising Money for your Business Communication at the Workplace: How to Interact More Effectively with Your Coworkers, Your Key to Success