Up-to-date books concerning "Working Effectively with Legacy Code" [closed]

Asked 9 years, 9 months ago Modified 9 years, 9 months ago

Viewed 1k times

2

Closed. This question is seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. It does not meet Stack Overflow guidelines. It is not currently accepting answers.

We don't allow questions seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. You can edit the question so it can be answered with facts and citations.

Closed 7 years ago.

Improve this question

As a team we're trying to modernize the code of an existing project (medium sized, say 300K LoC) while trying to not fall into the trap of doing a full re-write

The existing code is tightly coupled and lacks unit tests but the newer parts are well written.

I'm looking for some literature (books, blogs, etc...) on how to proceed. I see that the book "Working Effectively with Legacy Code" is highly recommended however that book is now eleven years old. A lot has changed since then. I'm afraid that it might miss some crucial techniques. Especially since so much has happened in unit-testing and breaking tight coupling (for example I do believe dependency injection has only been a thing for the last eight years or so)

Is there any up-to-date book that I can read that covers breaking dependencies and writing unit test for an existing code base?

Edit: a quick note, the project is 99% C# with a bit of C++/CLI mixed in for the parts that handle video encoding.

unit-testing refactoring

Share Follow

edited May 23, 2017 at 11:46

Community Bot

1 ● 1

asked Mar 10, 2015 at 8:43

Roy T.

9,608 ● 2 ● 52 ● 74

2 Answers

Sorted by: Highest score (default)

2

The book might be a couple of years old, but as long as you are working with an object-oriented/procedural language, the techniques in this book are *timeless*. It's all about isolating hard to test components by introducing seams.

It seems you are working on a .NET codebase. In that case I can also recommend picking up a more recent book: The Art of Unit Testing with examples in C# (2nd edition) by Roy Osherove which also touches upon working with legacy code. Mind, he's taken a lot of inspiration from Feather's book. The book also touches on some advanced mocking framework features (fake out private methods etc) but for these techniques YMMV as it ties your tests very closely to the current implementation.

If you want to dig into some more recent stuff, I would recommend reading up on some legacy code guru blogs like Adrian Bolboaca's and J.B. Rainsberger's. Techniques like golden master testing etc. are not represented in the book either but are valuable techniques when you are reworking a legacy codebase.

Finally, if you can get your team together for a legacy code retreat that might be a nice introduction to the techniques as well.

edited Mar 10, 2015 at 12:09

answered Mar 10, 2015 at 8:59

prgmtc

750 ● 3 ● 11

I agree that a lot of the techniques covered are still relevant, but 11 years is a lot of time, especially in software development. For example: the application I'm working on is only 6 years old. And DI is definitely not the only 'young' technique useful for decoupling or unit testing. Another example, Visual Studio added Unit Testing support in 2005 (one year after this book was written). .Net only has generics since 2007. – Roy T. Mar 10, 2015 at 9:31

1 You're right about 11 years being a lot of time in our field :) I added an extra reference to my answer regarding a more recent book. – prgmtc Mar 10, 2015 at 10:11

0

I would implement functional tests with something like Selenium before dealing with Unit Tests if they are not there already.

I would worry with Unit Testing for each module being changed. You did not specified, but I am guessing the legacy software is a monolithic application, of not, doing system testing would also be a good strategy.

Share Follow

answered Mar 10, 2015 at 8:50

lnrdo

406 ● 2 ● 13