Refactoring for Testability on an existing system

Asked 16 years, 4 months ago Modified 15 years, 11 months ago

Viewed 791 times

5

I've joined a team that works on a product. This product has been around for ~5 years or so, and uses ASP.NET WebForms. Its original architecture has faded over time, and things have become relatively disorganized throughout the solution. It's by no means terrible, but definitely can use some work; you all know what I mean.

I've been performing some refactorings since coming on to the project team about 6 months ago. Some of those refactorings are simple, Extract Method, Pull Method Up, etc. Some of the refactorings are more structural. The latter changes make me nervous as there isn't a comprehensive suite of unit tests to accompany every component.

The whole team is on board for the need to make structural changes through refactoring, but our Project Manager has expressed some concerns that we don't have adequate tests to make refactorings with the confidence that we aren't introducing regression bugs into the system. He would like us to write more tests first (against the existing architecture), then perform the refactorings. My argument is that the system's class

structure is too tightly coupled to write adequate tests, and that using a more Test Driven approach while we perform our refactorings may be better. What I mean by this is not writing tests against the existing components, but writing tests for specific functional requirements, then refactoring existing code to meet those requirements. This will allow us to write tests that will probably have more longevity in the system, rather than writing a bunch of 'throw away' tests.

Does anyone have any experience as to what the best course of action is? I have my own thoughts, but would like to hear some input from the community.

asp.net refactoring automated-tests

Share

Improve this question

Follow

asked Aug 21, 2008 at 15:34

steve_c

6,255 ● 4 ● 33 ● 43

5 Answers

Sorted by: Highest score (default)

5

Your PM's concerns are valid - make sure you get your system under test before making any major refactorings.

I would strongly recommend getting a copy of Michael Feather's book Working Effectively With Legacy Code (by "Legacy Code" Feathers means any system that isn't adequately covered by unit tests). This is chock full of

good ideas for how to break down those couplings and dependencies you speak of, in a safe manner that won't risk introducing regression bugs.

Good luck with the refactoring programme; in my experience it's an enjoyable and cathartic process from which you can learn a lot.

Share Improve this answer

Follow

edited Aug 23, 2008 at 10:25

answered Aug 21, 2008 at 15:38

Ian Nelson

58.6k ● 20 ● 78 ● 104

2

Can you re-factor in parallel? What I mean is re-write the pieces you want to refactor using TDD, but leave the existing code base in place. Then phase out the existing code when your new tests meet the needs for your PM?

Share Improve this answer

Follow

answered Aug 21, 2008 at 15:37

Jay Mooney

2,256 ● 1 ● 19 ● 23

1

I would also like to throw in a suggestion to visit the Refactoring website by Martin Fowler. He literally wrote the book on this stuff.

As far as introducing unit tests into the equation the best method I have found is to find a top level component and identify all the external dependencies it has on concrete objects and replace them with interfaces. Once you've done that it will be a lot easier to write unit tests against your code base and you can do it one component at a time. Even better, you won't have to throw away any unit tests.

Unit testing ASP.Net can be tricky, but there are plenty of frameworks that make it easier to do. ASP.Net MVC, and WCSF to name a few.

Share Improve this answer

Follow

answered Sep 17, 2008 at 4:04

Josh

44.9k ● 7 ● 104 ● 124

0

Just tossing out a second recommendation for Working Effectively with Legacy Code, an excellent book that really opened my eyes to the fact that almost any old / crappy / untestable code can be wrangled!

Share Improve this answer

Follow

answered Aug 23, 2008 at 11:43

jonezy

1,923 ● 13 ● 21

Totally agree with the answer from Ian Nelson.
Additionally I would start to get some "high level" tests (functional or component tests) in place to preserve the behaviour from the view point of the user. This point might be the most important concern for your PM.

0

Share Improve this answer

Follow

edited May 23, 2017 at 11:55

Community Bot

1 ● 1

answered Jan 17, 2009 at 23:29

Theo Lenndorff

4,596 ● 3 ● 29 ● 43