

# In agile like development, who should write test cases? [closed]

Asked 16 years, 2 months ago   Modified 8 years, 6 months ago

Viewed 25k times



17



**Closed.** This question is [opinion-based](#). It is not currently accepting answers.



**Want to improve this question?** Update the question so it can be answered with facts and citations by [editing this post](#).

Closed 8 years ago.

[Improve this question](#)

Our team has a task system where we post small incremental tasks assigned to each developer.

Each task is developed in its own branch, and then each branch is tested before being merged to the trunk.

My question is: Once the task is done, who should define the **test cases** that should be done on this task?

Ideally I think the developer of the task himself is best suited for the job, but I have had a lot of resistance from

developers who think it's a waste of their time, or that they simply don't like doing it.

The reason I don't like having my QA people do it, is because I don't like the idea of them creating their own work. For example they might leave out things that are simply too much work to test, and they may not know the technical detail that is needed.

But likewise, the down part of developers doing the test cases, is that they may leave out things that they think will break. (even subconsciously maybe)

As the project manager, I ended up writing the test cases for each task myself, but my time is taxed and I want to change this.

Suggestions?

**EDIT: By test cases I mean the description of the individual QA tasks that should be done to the branch before it should be merged to the trunk. (Black Box)**

testing

project-management

agile

testcase

Share

Improve this question

Follow

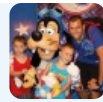
edited Jul 28, 2010 at 12:05



MvanGeest

9,661 ● 4 ● 42 ● 41

asked Oct 11, 2008 at 23:06



Brian R. Bondy

347k ● 126 ● 602 ● 640

- 
- 1 I'm voting to close this question as off-topic because it is about a developer environment rather than a programming problem. – [gunr2171](#) Jun 3, 2016 at 20:46
- 

16 Answers

Sorted by:

Highest score (default)



## The Team.

19

If a defect gets to a customer, it is ***the team's*** fault, therefore ***the team*** should be writing test cases to assure that defects don't reach the customer.



1. **The Project Manager (PM)** should understand the domain better than anyone on the team. Their domain knowledge is *vital* to having test cases that make sense with regard to the domain. They will need to provide example inputs and answer questions about expectations on invalid inputs. They need to provide at least the 'happy path' test case.
2. **The Developer(s)** will know the code. You suggest the developer may be best for the task, but that you are looking for black box test cases. Any tests that a developer comes up with are white box tests. That is the advantage of having developers create test cases – they know where the seams in the code are.  
  
Good developers will also be coming to the PM with questions "What should happen when...?" – each of

these is a test case. If the answer is complex "If a then x, but if b then y, except on Thursdays" – there are multiple test cases.

3. **The Testers (QA)** know how to test software. Testers are likely to come up with test cases that the PM and the developers would not think of – that is why you have testers.

Share Improve this answer

Follow

edited Jun 20, 2020 at 9:12



Community Bot

1 • 1

answered Oct 16, 2008 at 22:56



mandersn

325 • 2 • 10

- 
- 4 I think it's pretty unlikely that PMs "understand the domain better than anyone." In general PMs are not business specialists but expert in cost, schedule and risk management. More likely the PM would find someone who might be an expert. The BA is more likely to understand the domain.  
– [Chris McCauley](#) Jul 28, 2010 at 12:09

- 
- 1 I've been on three agile teams, and our PM and BA were generally the same person. – [mandersn](#) Aug 14, 2010 at 18:19

---

On the team, the PMs won't know the domain better than anyone else. Great if they do, but in general they do not. On a small project, I can see that being possible, but not for mid and large sizes. – [Roy Oliver](#) Jun 20, 2014 at 13:48

---

Testcases should come from the analysts who have access to the customers and are writing the requirements. Test



I think the Project Manager, or Business Analyst should write those test cases.

8

They should then hand them over to the QA person to flesh out and test.



That way you ensure no missing gaps between the spec, and what's actually tested and delivered.



The developer's should definately not do it, as they'll be testing their unit tests. So it's a waste of time.

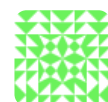
In addition these tests will find errors which the developer will never find as they are probably due to a misunderstanding in the spec, or a feature or route through the code not having been thought through and implemented correctly.

If you find you don't have enough time for this, hire someone else, or promote someone to this role, as it's key to delivering an excellent product.

Share Improve this answer

answered Oct 11, 2008 at 23:18

Follow



[Bravax](#)

10.5k ● 8 ● 43 ● 68



From past experience, we had pretty good luck defining tests at different levels to test slightly different things:

5



1st tier: At the code/class level, developers should be writing atomic unit tests. The purpose is to test individual classes and methods as much as possible. These tests should be run by developers as they code, presumably before archiving code into source control, and by a continuous-integration server (automated) if one is being used.

2nd tier: At the component integration level, again have developers creating unit tests, but that test the integration between components. The purpose is not to test individual classes and components, but to test how they interact with each other. These tests should be run manually by an integration engineer, or automated by a continuous-integration seerver, if one is in use.

3rd tier: At the application level, have the QA team running their system tests. These test cases should be based off the business assumptions or requirements documents provided by a product manager. Basically, test as if you were an end user, doing the things end users should be able to do, as documented int eh requirements. These test cases should be written by the QA team and the product managers who (presumably) know what the customer wants and how they are expected to use the application.

I feel this provides a pretty good level of coverage. Of course, tiers 1 and 2 above should ideally be run before sending a built application to the QA team. Of course, you can adapt this to whatever fits your business model, but

this worked pretty well at my last job. Our continuous-integration server would kick out an email to the development team if one of the unit tests failed during the build/integration process too, incase someone forgot to run their tests and committed broken code into the source archive.

Share Improve this answer

answered Oct 15, 2008 at 15:42

Follow



CodingWithSpike

43.7k ● 18 ● 105 ● 139



4



We experimented with a pairing of the developer with a QA person with pretty good results. They generally 'kept each other honest' and since the developer had unit tests to handle the code, s/he was quite intimate with the changes already. The QA person wasn't but came at it from the black box side. Both were held accountable for completeness. Part of the ongoing review process helped to catch unit test shortcomings and so there weren't too many incidents that I was aware of where anyone was purposely avoiding writing X test because it would likely prove there was a problem.

I like the pairing idea in some instances and think it worked pretty well. Might not always work, but having those players from different areas interact helped to avoid the 'throw it over the wall' mentality that often happens.

Anyhow, hope that is somehow helpful to you.

Share Improve this answer

answered Oct 11, 2008 at 23:17

Follow



itsmatt

31.4k ● 11 ● 102 ● 165



2



The reason I don't like having my QA people do it, is because I don't like the idea of them creating their own work. For example they might leave out things that are simply too much work to test, and they may not know the technical detail that is needed.

Yikes, you need to have more trust in your QA department, or a better one. I mean, imagine of you had said "I don't like having my developers develop software. I don't like the idea of them creating their own work."

As a developer, I know that there are risks involved in writing my own tests. That's not to say I don't do that (I do, especially if I am doing TDD) but I have no illusions about test coverage. Developers are going to write tests that show that their code does what they think it does. Not too many are going to write tests that apply to the actual business case at hand.

Testing is a skill, and hopefully your QA department, or at least, the leaders in that department, are well versed in that skill.

Share Improve this answer

answered Oct 12, 2008 at 0:23

Follow





Jonathan Arkell

10.8k ● 2 ● 26 ● 32

---

The analogy with the developers is not right, the correct one would be that I don't like them defining their own spec. Which I don't allow them to do either. I do have faith in my QA department. – [Brian R. Bondy](#) Oct 12, 2008 at 0:57

---



2



"developers who think it's a waste of their time, or that they simply don't like doing it" Then reward them for it.

What social engineering is necessary to get them to create test cases?

Can QA look over the code and test cases and pronounce "Not Enough Coverage -- Need More Cases". If so, then the programmer that has "enough" coverage right away will be the Big Kahuna.

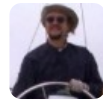
So, my question is: Once the task is done, who should define the goal of "enough" test cases for this task? Once you know "enough", you can make the programmers responsible for filling in "enough" and QA responsible for assuring that "enough" testing is done.

Too hard to define "enough"? Interesting. Probably this is the root cause of the conflict with the programmers in the first place. They might feel it's a waste of their time because they already did "enough" and now someone is saying it isn't "enough".

Share Improve this answer

answered Oct 12, 2008 at 23:27

Follow



S.Lott

391k ● 82 ● 517 ● 788



2



the QA people, in conjunction with the "customer", should *define* the test cases for each task [we're really mixing terminology here], and the developer should write them. first!

Share Improve this answer

answered Oct 15, 2008 at 16:00



Follow



Steven A. Lowe

61.1k ● 19 ● 135 ● 204



1



Select (not just pick randomly) one or two testers, and let them write the test cases. Review. It could also be useful if a developer working with a task looks at the test cases for the task. Encourage testers to suggest improvements and additions to test sets - sometimes people are afraid to fix what the boss did. This way you might find someone who is good at test design.



Let the testers know about the technical details - I think everyone in an agile team should have read access to code, and whatever documentation is available. Most testers I know can read (and write) code, so they might find unit tests useful, possibly even extend them. Make sure the test designers get useful answers from the developers, if they need to know something.

Share Improve this answer

answered Oct 14, 2008 at 18:36

Follow



jschultz  
146 ● 5



1




My suggestion would be to having someone else look over the test cases before the code is merged to ensure quality. Granted this may mean that a developer is overlooking another developer's work but that second set of eyes may catch something that wasn't initially caught. The initial test cases can be done by any developer, analyst or manager, not a tester.

QA shouldn't write the test cases as they may be situations where the expected result hasn't been defined and by this point, it may be hard to have someone referee between QA and development if each side thinks their interpretation is the right one. It is something I have seen many many times and wish it didn't happen as often as it does.

Share Improve this answer

answered Oct 15, 2008 at 15:52

Follow



JB King  
11.9k ● 4 ● 40 ● 49



1



I loosely break my tests down into "developer" tests and "customer" tests, the latter of which would be "acceptance tests". The former are the tests that developers write to verify that their code is performing correctly. The later are tests that someone *other* than developers write to ensure that behavior matches the spec. The developers must *never* write the acceptance



tests because their creation of the software they're testing assumes that they did the right thing. Thus, their acceptance tests are probably going to assert what the developer already knew to be true.

The acceptance tests should be driven by the spec and if they're written by the developer, they'll get driven by the code and thus by the current behavior, not the desired behavior.

Share Improve this answer

answered Oct 15, 2008 at 15:55

Follow



Ovid

11.7k ● 9 ● 49 ● 76



The **Agile canon** is that you should have (at least) two layers of tests: developer tests and customer tests.

1



**Developer tests** are written by the same people who write the production code, preferably using *test driven development*. They help coming up with a well decoupled design, and ensure that the code is doing what the developers think it is doing - even after a refactoring.



**Customer tests** are *specified* by the customer or customer proxy. They are, in fact, the specification of the system, and should be written in a way that they are both executable (fully automated) *and* understandable by the business people. Often enough, teams find ways for the customer to even *write* them, with the help of QA people. This should happen while - or even before - the functionality gets developed.

Ideally, the only tasks for QA to do just before the merge, is pressing a button to run all automated tests, and do some additional exploratory (=unscripted) testing. *You'll want to run those tests again after the merge, too, to make sure that integrating the changes didn't break something.*

Share Improve this answer

answered Oct 31, 2008 at 7:01

Follow



**Ilja Preuß**

2,421 ● 17 ● 15



### **A test case begins first in the story card.**

**1**

The purpose of testing is to drive defects to the left (earlier in the software development process when they are cheaper and faster to fix).



Each story card should include acceptance criteria. The Product Owner pairs with the Solution Analyst to define the acceptance criteria for each story. This criteria is used to determine if a story card's purpose has been met.

The story card acceptance criteria will determine what automated unit tests need to be coded by the developers as they do Test Driven Development. It will also drive the automated functional test implemented by the automated testers (and perhaps with developer support if using tools like FIT).

Just as importantly, the acceptance criteria will drive the automated performance tests and can be used when

analyzing the profiling of the application by the developers.

Finally, the user acceptance test will be determined by the acceptance criteria in the story cards and should be designed by the business partner and or users. Follow this process and you will likely release with zero defects.

Share Improve this answer

edited May 24, 2009 at 22:17

Follow

answered May 12, 2009 at 17:36



Cam Wolff

1,420 ● 15 ● 13



1



I've rarely have heard of or seen Project Managers write test cases except for in the smaller teams. In any large, complex software application have to have an analyst that *really* knows the application. I worked at a mortgage company as a PM - was I to understand sub-prime lending, interest rates, and the such? Maybe at a superficial level, but real experts needed to make sure those things worked. My job was to keep the team healthy, protect the agile principles, and look for new opportunities for work for my team.

Share Improve this answer

answered Apr 11, 2012 at 19:57

Follow



Joe Fecarotta

11 ● 1



0



The system analyst should review over all test-cases and its correct relation with the use-cases. Plus the Analyst should perform the final UAT, which could be based on test-cases also. So the analyst and the quality guy are making sort of peer-review.

The quality is reviewing the use-cases while he is building test-cases, and the analyst is reviewing the test-cases after they are written and while he is performing UAT.

Share Improve this answer

answered [Sep 18, 2014 at 15:54](#)

Follow

community wiki  
[Hossam](#)



0



Of course BA is the domain expert, not from technical point of view. BA understands the requirements and the test cases should be mapped to the requirements. Developers should not be the persons writing the test cases to test against their code. QA can write detail test steps per requirement. But the person who writes the requirement should dictate what needs to be tested. Who actually writes the test cases, I dont care too much as long as the test cases can be traced back to requirements. I would think it makes sense that BA guides the testing direction or scope, and QA writes the granular testing plans.

Share Improve this answer

answered Feb 11, 2015 at 15:48

Follow



Vinnie

1



0



We need to evolve from the "this is how it has been done or should be done mentality" it is failing and failing continuously. The best way to resolve the test plan/cases writing issue is that test cases should be written on the requirements doc in waterfall or the user story in agile as those reqs/user stories are being written. This way there is no question what needs to be tested and QA and UAT teams can execute the test case(s) and focus time on actual testing and defect resolution.

Share Improve this answer

answered Aug 19, 2015 at 21:22

Follow



UAT Guru

1