Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

TEAMS

Create Team

Black box vs White box Testing

Ask Question

Which type of testing would you say should be the emphasis (for testers/QAs), and why?

A quick set of definitions from wikipedia:

Black box testing

- takes an external perspective of the test object to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid input and determines the correct output. There is no knowledge of the test object's internal structure.

White box testing

- uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. In electrical hardware testing, every node in a circuit may be probed and measured; an example is in-circuit testing (ICT).

Is internal knowledge important for the tester/QA? I've heard arguments that testing with this knowledge in mind enables them to better test for problems, but I've also heard arguments that this knowledge can distract from functional needs and promote "testing to to the code" rather than to the intended solution.

testing qa black-box white-box

edited Jun 7 '09 at 21:00

Orion Edwards
82k 48 188 269

asked Dec 31 '08 at 2:37

TM.
44.1k 26 107 124

1 I found the question and answers very helpful and stole them to answer my own question stackoverflow.com/questions/402161/… – Gennady Vanin Геннадий Ванин Dec 4 '10 at 16:56

15 Answers

- Black box testing should be the emphasis for testers/QA.
- White box testing should be the emphasis for developers (i.e. unit tests).
- The other folks who answered this question seemed to have interpreted the question as *Which is more important, white box testing or black box testing*. I, too, believe that they are both important but you might want to check out this

22 9

answered Dec 31 '08 at 4:02

Glenn

5,988 3 23 36

2 What is the name of the "IEEE article". The link provided is broken – Gennady Vanin Геннадий Ванин Dec 4 '10 at 16:25

2 The link still works. The article is called Frequently Forgotten Facts about Software Engineering. – Glenn Feb 20 '11 at 19:20

15 -1 Unit tests have nothing to do with white box testing vs blackbox testing. Unit tests can use both whitebox and blackbox styles. Well written unit tests, while they are testing "internal" modules/objects/functions, can and should test each *unit* as a black box. – B T Jun 19 '14 at 19:46

3 @WarrenParks Black box testing isn't about whether you have or do not have knowledge of it. Rather, black box testing is about writing tests that don't assume things about the internal non-public parts of the things your testing. – B T Nov 17 '15 at 7:27

1 Downvoted, because this answer does not answer why. And I disagree about whitebox testing. It is almost always more useful to do blackbox testing. If there is something that you can only test using whitebox testing, then it is either incidental and irrelevant or it is dead code. – Sammi Oct 10 '16 at 13:10

White Box Testing equals Software Unit Test. The developer or a

Black Box Testing equals Integration Testing. The tester ensures that the system works according to the requirements on a functional level.

Both test approaches are **equally important** in my opinion.

A thorough unit test will catch defects in the development stage and not after the software has been integrated into the system. A system level black box test will ensure all software modules behave correctly when integrated together. A unit test in the development stage would not catch these defects since modules are usually developed independent from each other.

edited Dec 31 '08 at 2:51

answered Dec 31 '08 at 2:45

cschol
7,774 10 51 75

4 This answer gives incorrect definitions of whitebox and blackbox testing. Both can be unit tests and both can be integration tests. That is really just beside the point. Whitebox tests DO NOT test if a behavior works according to specs. Whitebox testing is testing whether or not you can exercise all code branches in a given implementation without error. Blackbox testing however, is about whether or not a given implementation does the right thing under different circumstances. Whether or not it behaves according to specs. – Sammi Oct 10 '16 at 14:11

1 Concur with @Sammi. Also note, than when doing TDD writing a test cannot be white-box, as the implementation does not actually exist yet. High branch coverage however is still achieved as you are only (supposed to) writing minimum code to fulfill tests. However, this is not a guarantee for perfect test coverage (also note the red-green-*refactor* step) – BatteryBackupUnit Apr 13 at 6:59

called see-through and non-see-through boxes or glass and steel boxes or something. But whitebox and blackbox are the common terms used in computer science. – Sammi Jun 26 at 17:10

Black Box

1 Focuses on the functionality of the system Focuses on the structure (Program) of the system

2 Techniques used are :

· Equivalence partitioning

· Boundary-value analysis

· Error guessing

· Race conditions

· Cause-effect graphing

· Syntax testing

· State transition testing

· Graph matrix

Tester can be non technical

Helps to identify the vagueness and contradiction in functional specifications

White Box

Techniques used are:

· Control Structure Testing

1. Condition Testing
2. Data Flow testing

· Loop Testing

1. Simple Loops
2. Nested Loops
3. Concatenated Loops
4. Unstructured Loops

Tester should be technical

Helps to identify the logical and coding issues.

answered Jun 23 '10 at 16:09

kalidass
61 1 1

"Both" has been stated above, and is the obvious answer...but IMO, white box testing goes far beyond developer unit testing (althoughI suppose it could depend on where you draw the line between white and black). For example, code coverage analysis is a common white box approach - i.e. run some scenarios or tests, and examine the results looking for holes in testing. Even if unit tests have 100% cc, measuring cc on common user scenarios can reveal code that may potentially need even more testing.

Another place where white box testing helps is examining data types, constants and other information to look for boundaries

256 are treated one way, while larger values are treated another way...and perhaps the number 42 has yet another code path.

So, to answer the original question - both bb and wb are essential for good testing.

answered Dec 31 '08 at 6:54

Alan

1,010 7 11

In my experience most developers naturally migrate towards white box testing. Since we need to ensure that the underlying algorithm is "correct", we tend to focus more on the internals. But, as has been pointed out, both white and black box testing is important.

Therefore, I prefer to have testers focus more on the Black Box tests, to cover for the fact that most developers don't really do it, and frequently aren't very good at it.

That isn't to say that testers should be kept in the dark about how the system works, just that I prefer them to focus more on the problem domain and how actual users interact with the system, not whether the function SomeMethod(int x) will correctly throw an exception if x is equal to 5.

answered Dec 31 '08 at 3:48

Brian B.

1,759 11 10

It's a bit of an open door, but in the end both are about equally

1. software that does what it needs to do, but internally has problems?
2. software that is supposed to work if you look at the sources, but doesn't?

My answer: Neither is totally acceptable, but software cannot be proven to be 100% bugfree. So you're going to have to make some trade-offs. Option two is more directly noticable to clients, so you're going to get problems with that sooner. On the long run, option one is going to be problematic.

answered Dec 31 '08 at 2:53

Wouter van Nifterick
18.2k 2 62 113

5 Isn't #2 much much worse than #1? – Warpin Jan 26 '10 at 4:49

Black Box Testing: Black Box testing is just observation no need Internal Knowledge or structure of software product. just putting valid and Invalid data input and expecting the correct result. here tester find the defect but unable to Find the Location of defect.black box testing done in all testing level.

Black box testing tecniques are: 1. Equivalance partition 2. Boundary Value Analysis 3. Decision table 4. State Transition Diagram 4. Use case diagram

White Box Testing: White box is testing it requires the knowledge of internal logic and structure of software product. here we will check the loop, condition and branch. here we find not only the defect but also and location of defect.

21 1

- Usually the white-box testing is not possible for testers. Thus the only viable answer for testers is to emphasize black-box approach.
- However, with aspect-oriented-programming and design-by-contract methodology, when the testing goals are programmed into the target code as contracts (seen from the static view of a program), and/or when the testing temporal logic is programmed into the code as cross-cuts (dynamic view of the test logic), white-box testing would become not only possible but also a preferred take for testers. Given that said, it will need be an expertise-demanding take, the testers need to be not only good testers, but also good programmers or more than good programmers.

answered Mar 19 '11 at 7:15

minghua
2,041 1 18 39

QA should focus on **Black box testing**. The main goal of QA is to test what the system does (do features meet requirements ?), not how it does it.

Anyway it should be hard for QA to do white box testing as most of QA guys aren't tech guys, so they usually test features through the UI (like users).

A step further, I think **developpers** too should focus on **Black**

does, not how. Moreover White box testing implies you know how the method will fill its contract, that seems incompatile with TDD to me.

IMHO if your SUT is so complex that you need to do white box testing, it's usually time for refactoring.

answered Nov 3 '14 at 19:00

user1075613
381 5 9

What constitutes, "internal knowledge?" Does knowing that such-and-such algorithm was used to solve a problem qualify or does the tester have to see every line of code for it to be "internal?"

I think in any test case, there should be expected results given by the specification used and not determined by how the tester decides to interpret the specification as this can lead to issues where each thinks they are right and blaming the other for the problem.

answered Dec 31 '08 at 3:31

community wiki
JB King

---

1 That limits all your test cases to those where you already know the answer. That would have halved the number of showstopper issues my colleagues and I have found over the years. "Each thinks they are right" is an invaluable indication that something needs more

- *Black box testing: Is the test at system level to check the functionality of the system, to ensure that the system performs all the functions that it was designed for, Its mainly to uncover defects found at the user point. Its better to hire a professional tester to black box your system, 'coz the developer usually tests with a perspective that the codes he had written is good and meets the functional requirements of the clients so he could miss out a lot of things (I don't mean to offend anybody)
- Whitebox is the first test that is done in the SDLC.This is to uncover bugs like runtime errors and compilation errrors It can be done either by testers or by Developer himself, But I think its always better that the person who wrote the code tests it.He understands them more than another person.*

answered Jan 7 '09 at 14:38

community wiki
Lord Leadhead

---

*Black-Box testing: If the source code is not available then test data is based on the function of the software without regard to how it was implemented. -strong textExamples of black-box testing are: boundary value testing and equivalence partitioning.

*White-Box testing: If the source code of the system under test is available then the test data is based on the structure of this source code. -Examples of white-box testing are: path testing and data flow testing.

Simple...Blackbox testing is otherwise known as Integration testing or smoke-screen testing . This is mostly applied in a distributed environment which rely on event-driven architecture. You test a service based on another service to see all possible scenarios. Here you cannot completely forecast all possible output because each component of the SOA/Enterprise app are meant to function autonomously. This can be referred to as High-Level testing

while

White box testing refers to unit-testing. where all expected scenarios and output can be effectively forecasted. i.e Input and expected output.This can be referred to as Low-level testing

edited Apr 29 '13 at 16:05

answered Apr 25 '13 at 18:14

Kermit_ice_tea
262 1 5 13

I only partially agree with the top rated answer for this question. Which type of testing would you say should be the emphasis (for testers/QAs), and why?

1. I agree that: "Black box testing should be the emphasis for testers/QA."
2. I agree that White box testing should be the emphasis for developers, but I don't agree that White Box testing is just unit tests.

- Unit Testing: For testing paths within a unit
- Integration Testing:For testing paths between units
- System Testing: For testing paths between subsystems

answered Jul 24 '14 at 17:17

leroneb
43 4

Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester. White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

answered Sep 29 '16 at 12:16

Pratik 2011
1
