WIKIPEDIA

White-box testing

White-box testing (also known as **clear box testing**, **glass box testing**, **transparent box testing**, and **structural testing**) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the expected outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

White-box test design techniques include the following code coverage criteria:

- Control flow testing
- Data flow testing
- Branch testing
- Statement coverage
- Decision coverage
- Modified condition/decision coverage
- Prime path testing
- Path testing

Contents

Overview

White-box testing is a method of testing the application at the level of the source code. These test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error-free environment by examining any fragile code. These white-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on.[1] These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.[1]

Levels

1. Unit testing. White-box testing is done during unit testing to ensure that the code is working as intended; before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.[1]
2. Integration testing. White-box testing at this level is written to test the interactions of interfaces with each other. The unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.[1]
3. Regression testing. White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.[1]

Basic procedure

White-box testing's basic procedures require the tester to have an in-depth knowledge of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for test cases to be created. The following are the three basic steps that white-box testing takes in order to create test cases:

1. Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code and security specifications.[2] This is the preparation stage of white-box testing to lay out all of the basic information.
2. Processing involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results.[2] This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.
3. Output involves preparing final report that encompasses all of the above preparations and results.[2]

Advantages

White-box testing is one of the two biggest testing methodologies used today. It has several major advantages:

1. Side effects of having the knowledge of the source code is beneficial to thorough testing.[3]
2. Optimization of code becomes easy as inconspicuous bottlenecks are exposed.[3]

3. Gives the programmer introspection because developers carefully describe any new implementation.[3]
4. Provides traceability of tests from the source, thereby allowing future changes to the source to be easily captured in the newly added or modified tests.[4]
5. Easy to automate.[5]
6. Provides clear, engineering-based rules for when to stop testing.[6][5]

Disadvantages

Although white-box testing has great advantages, it is not perfect and contains some disadvantages:

1. White-box testing brings complexity to testing because the tester must have knowledge of the program, including being a programmer. White-box testing requires a programmer with a high level of knowledge due to the complexity of the level of testing that needs to be done.[3]
2. On some occasions, it is not realistic to be able to test every single existing condition of the application and some conditions will be untested.[3]
3. The tests focus on the software as it exists, and missing functionality may not be discovered.
4. The resulting test can be fragile because they are tightly coupled to the specific implementation of the thing being tested. The code under test could be rewritten to implement the same functionality in a different way that invalidates the assumptions baked into the test. This could result in tests that fail unnecessarily or, in the worst case, tests that now give false positives and mask errors in the code.

Modern view

A more modern view is that the dichotomy between white-box testing and black-box testing has blurred and is becoming less relevant. Whereas "white-box" originally meant using the source code, and black-box meant using requirements, tests are now derived from many documents at various levels of abstraction. The real point is that tests are usually designed from an abstract structure such as the input space, a graph, or logical predicates, and the question is what level of abstraction we derive that abstract structure from.[5] That can be the source code, requirements, input space descriptions, or one of dozens of types of design models. Therefore, the "white-box / black-box" distinction is less important and the terms are less relevant.

Hacking

In penetration testing, white-box testing refers to a methodology where a white hat hacker has full knowledge of the system being attacked. The goal of a white-box penetration test is to simulate a malicious insider who has knowledge of and possibly basic credentials for the target system.

See also

- Black-box testing
- Gray-box testing
- White-box cryptography

References

1. Williams, Laurie. "White-Box Testing" (http://www.chaudhary.org/WhiteBox.pdf) (PDF): 60–61, 69. Retrieved 13 February 2013.
2. Ehmer Khan, Mohd (July 2011). "Different Approaches to White Box Testing Technique for Finding Errors" (http://www.sersc.org/journals/IJSEIA/vol5_no3_2011/1.pdf) (PDF). *International Journal of Software Engineering and Its Applications*. **5**: 1–6. Retrieved 12 February 2013.
3. Ehmer Khan, Mohd (May 2010). "Different Forms of Software Testing Techniques for Finding Errors" (http://ijcsi.org/papers/7-3-1-11-16.pdf) (PDF). *IJCSI International Journal of Computer Science Issues*. **7** (3): 12. Retrieved 12 February 2013.
4. Binder, Bob (2000). *Testing Object-oriented Systems*. Addison-Wesley Publishing Company Inc.
5. Ammann, Paul; Offutt, Jeff (2008). *Introduction to Software Testing* (http://cs.gmu.edu/~offutt/softwaretest/). Cambridge University Press. ISBN 9780521880381.
6. Myers, Glenford (1979). *The Art of Software Testing*. John Wiley and Sons.

External links

- BCS SIGIST (British Computer Society Specialist Interest Group in Software Testing): http://www.testingstandards.co.uk/Component%20Testing.pdf *Standard for Software Component Testing*], Working Draft 3.4, 27. April 2001.
- http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf has more information on control flow testing and data flow testing.
- http://research.microsoft.com/en-us/projects/pex/ Pex - Automated white-box testing for .NET