UnCodemy

Black Box and White Box Testing PDF

Black box testing checks behaviour from the outside using requirements, without looking at code. White box testing uses knowledge of the code structure to design tests and measure coverage such as statements and branches. Grey box testing combines both. Most real projects use all three at different levels.

Key takeaways

- Black box focuses on inputs and outputs and suits testers and acceptance work.
- White box focuses on code paths and suits developers and unit testing.
- Grey box uses partial knowledge such as the database schema or API design.
- Coverage measures how much code tests exercise but not whether tests are good.
- Use both views for stronger results.

What is black box and white box testing?

Black box testing designs tests from requirements and observable behaviour without considering internal code, while white box testing designs tests from the internal structure of the software.

What is black box testing?

In black box testing the tester treats the system as a box whose inside cannot be seen. Tests come from requirements, user stories and specifications. The tester gives inputs, observes outputs and compares them with expected behaviour. This view matches how users experience the product, and it works at every level from component to acceptance.

Which black box techniques are common?

Equivalence partitioning groups inputs that should behave the same, so one value represents each group. **Boundary value analysis** checks values at the edges of ranges. **Decision tables** list combinations of conditions and results. **State transition testing** checks how the system moves between states. **Use case testing** follows user scenarios. **Error guessing** relies on experience of typical faults.

What is white box testing?

White box testing uses the internal structure: code, branches, loops and data flow. Developers usually perform it at unit level. The goal is to make sure that each part of the code is exercised and that logic works for different paths. It can find hidden errors that behaviour-based tests miss.

What coverage measures are used?

Coverage	Meaning
Statement	Share of statements executed
Branch or decision	Share of decision outcomes (true and false) executed
Condition	Each condition in a decision tried both ways
Path	Share of independent paths executed
Higher coverage is useful but is not proof of quality. A test can run a line without checking its result.	

Can I see a small example?

```
function grade(score):
    if score >= 90: return "A"
    elif score >= 60: return "B"
    else: return "C"
```

For branch coverage you need at least one test per outcome: 95 for A, 70 for B and 40 for C. A black box tester would add boundary values 89, 90, 59 and 60 from the requirement, which also reveals if the code uses the wrong comparison.

What is grey box testing?

Grey box testing uses partial knowledge of internals, for example knowing the database tables to verify stored data after a UI action, or knowing API contracts to design tests. It combines the user viewpoint with some structural insight, and is common in integration and API testing.

How do the approaches compare?

Aspect	Black box	White box
Knowledge of code	Not needed	Required
Based on	Requirements	Code structure
Typical testers	Testers, users	Developers
Typical level	System, acceptance	Unit, integration
Finds	Missing or wrong behaviour	Logic and path errors

When should I use each?

Use white box methods to protect logic-heavy code and measure unit test coverage. Use black box methods to confirm that requirements are met and to find usability and workflow problems.

Use grey box methods when you test services and databases. A mix gives fewer blind spots.

What mistakes appear most often?

Chasing a coverage number without meaningful assertions, ignoring requirements when reading code, and assuming testers need no technical knowledge. Review tests as carefully as code.

How do testers and developers share this work?

In many teams developers write unit tests with white box thinking and testers review them for missing cases, while testers design black box tests from requirements and developers review them for technical feasibility. This shared review finds gaps that either side alone would miss. Pair on a tricky feature once in a while: the developer explains the branches and the tester proposes inputs and boundaries, and together you decide which cases are worth automating.

How do I practise both approaches?

Take a small function with several branches, such as a shipping fee calculator. First, write black box tests from a written rule list using partitions and boundaries. Then read the code, draw its branches and add tests for any path that is not yet covered. Compare the two sets and note which defects each style would have found. This exercise teaches why interviewers expect testers to understand both views and gives you a concrete example to describe in an interview.

What is a quick checklist for choosing an approach?

Ask whether you have access to code, whether the goal is to confirm requirements or to protect logic, which level you are working at and what risk you want to reduce. Then pick techniques accordingly and record the reason in your test notes.

What is a sample interview answer?

"Black box testing checks behaviour from requirements, white box testing uses code structure, and I combine them: boundaries from the spec and coverage from the code."

Frequently asked questions

What is black box testing?

It is testing from the outside using requirements and behaviour, without looking at the internal code of the software.

What is white box testing?

It is testing designed from the internal structure of the code, often by developers, with coverage measures such as statements and branches.

What is grey box testing?

Grey box testing uses partial knowledge of internals, such as database design or API contracts, combined with an external user view.

Does high code coverage mean good quality?

No. Coverage shows which code ran, not whether tests check the right results, so use it as one signal among many.

Which approach should testers use?

Most testers use black box techniques, while developers use white box ones, and teams combine both for fewer blind spots.

Sources and further reading

- ISTQB glossary and resources
- ISTQB glossary

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550