UnCodemy

Software Testing Levels and Types PDF

Testing levels describe where tests happen in the build-up of a product: unit, integration, system and acceptance. Testing types describe what is checked: functional behaviour or non-functional qualities such as performance, security and usability. Any type can happen at several levels, so think of levels and types as two separate axes.

Key takeaways

- Levels are unit, integration, system and acceptance.
- Types describe the objective: functional or non-functional.
- The same type can be applied at different levels.
- Choose levels and types by risk, not by habit.
- Static testing such as reviews finds defects without running code.

What is levels and types of software testing?

Levels of software testing are stages from small components to the full product, while types of testing are categories that describe the purpose of a test, such as functional, performance, security or usability testing.

What is the difference between levels and types?

A level answers "how big is the thing under test?". A type answers "what are we trying to learn?". For example, performance testing is a type; it can be done on a single function (unit level) or on the whole system. Mixing the two ideas is a common source of confusion in exams.

What are the four levels?

Level	What is tested	Example
Unit	A function or class in isolation	A discount calculator returns the right value
Integration	Modules working together	Cart service calls payment service correctly
System	The full product in a test environment	A user completes a purchase from start to end

Level	What is tested	Example
Acceptance	Readiness for business or users	Business users confirm the workflow
Acceptance testing includes user acceptance testing, alpha and beta testing, and contract or regulatory acceptance.		

Which integration approaches exist?

Big bang combines everything at once and makes faults hard to locate. Top-down starts with high-level modules using stubs for lower ones. Bottom-up starts with low-level modules using drivers. Sandwich mixes both. Incremental approaches make it easier to find where an interface fails.

What are functional types?

Functional testing checks behaviour against requirements. Within it you will meet smoke testing (is the build stable?), sanity testing (does a specific fix or feature work?), regression testing (did changes break existing features?), retesting (is the defect fixed?), exploratory testing (learning and testing at once) and user interface testing.

What are non-functional types?

Type	Question
Performance	How fast is it under normal load?
Load	Can it handle expected users?
Stress	What happens beyond limits?
Security	Can data and access be protected?
Usability	Is it easy to use?
Compatibility	Does it work across browsers, devices, systems?
Accessibility	Can people with disabilities use it?
Reliability	Does it keep working over time?

What is static versus dynamic testing?

Static testing examines work products such as requirements, designs and code without running them: reviews, walkthroughs and static analysis. Dynamic testing runs the software. Static testing finds defects early, when they are cheapest to fix.

How do confirmation and maintenance testing fit?

Confirmation testing, or retesting, checks that a reported defect is fixed. Regression testing checks that the fix did not break other areas. Maintenance testing happens after changes to a live system such as patches or migrations.

How do I choose what to run?

Start from risk: which failures would hurt users or the business most? Cover core flows at the system level, protect critical modules with unit and integration tests, add non-functional tests where they matter, and use acceptance tests to confirm business fit. Review the mix each release.

What mistakes appear in practice?

Testing only at the end, relying on the interface for everything, skipping integration tests, ignoring non-functional needs until late and confusing smoke with sanity. A balanced approach finds problems earlier.

How do these ideas show up in a real release?

During a sprint, developers run unit tests on every commit, the pipeline runs integration tests against a test database, testers run system tests on a staging build and the product owner performs acceptance checks. Before release the team may add performance and security checks. Seeing the same feature pass through several levels helps you explain why one kind of test is never enough, and why defects found early cost less to fix.

How do I answer level and type questions in interviews?

Define the term in one sentence, give a small example and say who usually performs it. For example, integration testing checks that modules work together, such as the cart and payment services, and developers or testers do it after unit tests. Then mention one risk it covers that other levels would miss. This three-part answer is short, clear and shows practical understanding instead of memorised definitions.

What is a short memory aid for levels?

Think of building a car: test the parts (unit), test how parts fit (integration), test the whole car on the road (system) and let the buyer test drive it (acceptance). The picture is simple and easy to recall in an exam.

What is a practical exercise for this topic?

Pick a food delivery app and list one example test for each level and for five types. Share your list with a friend and compare answers.

How do I avoid mixing up the terms?

Make flash cards with the term on one side and an example on the other, and shuffle them daily until you can answer without hesitation.

Frequently asked questions

What are the levels of software testing?

They are unit, integration, system and acceptance testing, which move from small components to full product readiness.

What is the difference between functional and non-functional testing?

Functional testing checks what the system does against requirements, while non-functional testing checks qualities such as speed, security and usability.

Can the same type of testing occur at different levels?

Yes. For example, performance testing can target a single function or the whole system, so level and type are independent.

What is static testing?

Static testing examines requirements, designs and code through reviews or analysis without running the software, and finds defects early.

What is the difference between retesting and regression testing?

Retesting confirms a specific fixed defect, while regression testing checks that changes did not break other existing features.

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