UnCodemy

Shift-Left Testing, TDD and BDD PDF

Shift-left testing means starting quality activities earlier in the life cycle. Test-driven development writes a failing test first, makes it pass and then refactors. Behaviour-driven development describes behaviour in Given-When-Then examples agreed by business, developers and testers. Together they reduce late surprises and unclear requirements.

Key takeaways

- Defects found earlier are usually cheaper and easier to fix.
- TDD follows red, green, refactor in small steps.
- BDD uses plain-language examples in Given-When-Then format.
- The three amigos are business, development and testing perspectives.
- These practices improve communication; tools are optional.

What is shift-left testing, TDD and BDD?

Shift-left testing is the practice of moving testing activities earlier in development, and TDD and BDD are two methods that support it by driving code and requirements from tests and examples.

What does shift-left mean?

Imagine the development timeline running left to right, with testing on the far right. Shifting left means moving quality work towards the start: reviewing requirements, asking testers into planning, writing tests with code, running automated checks on every commit. The aim is to find misunderstandings and faults while they are small.

Which activities shift left?

Activity	How it moves earlier
Requirement review	Testers ask "how would we test this?" before coding
Acceptance criteria	Written with examples at story creation
Unit and component tests	Written with the code
Static analysis	Run in the editor and pipeline
Security and accessibility checks	Considered in design
Environment and data setup	Prepared in advance with scripts

What is TDD?

Test-driven development works in short cycles. **Red:** write a small test that fails because the feature does not exist. **Green:** write the simplest code that makes it pass. **Refactor:** clean the code while keeping tests green. Repeat. The tests form a safety net and shape the design towards small, testable units.

Can I see a TDD example?

```
test: discount(100, "STUDENT") returns 90      -> fails (red)
code: if code == "STUDENT": return amount * 0.9 -> passes (green)
test: discount(100, "NONE") returns 100         -> fails
code: add the default branch                     -> passes
refactor: move percentages into a table, tests still pass
```

Each step is minutes long. The tests document behaviour as examples.

What is BDD?

Behaviour-driven development focuses on shared understanding. Teams describe behaviour with concrete examples in the **Given-When-Then** form: Given a registered customer with items in the cart, When they apply a valid coupon, Then the total is reduced. These examples become acceptance criteria and may be automated. BDD is mainly a conversation technique; automation is a bonus.

What are the three amigos?

A short meeting with a business representative, a developer and a tester before work starts. They discuss a story, ask "what if" questions and agree on examples. This catches missing rules, unclear terms and edge cases when changing a sentence is cheaper than changing code.

How do TDD and BDD differ?

Aspect	TDD	BDD
Focus	Design and correctness of code units	Shared understanding of behaviour
Written by	Developers	Whole team
Language	Programming language	Plain business language
Level	Unit and component	Feature and acceptance
They complement each other: BDD examples guide features, and TDD builds the units inside them.		

How do I start shift-left in a team?

Invite testers to refinement sessions, add examples to each story, set a rule that code changes include tests, run a fast test suite on every commit and review test failures together. Start with one team and one practice, measure how many defects reach late stages and adjust.

What are common problems?

Writing huge scenarios that are slow and brittle, treating BDD as only a test tool, skipping the conversation, chasing coverage numbers in TDD, and expecting early testing to remove all later testing. Keep scenarios short and focused on business rules.

How do I measure whether shift-left works?

Track where defects are found, such as requirements review, development, system testing or production, and how long they take to fix. If more issues are found earlier and fewer escape to later stages, the practice is helping. Also watch the time developers wait for feedback from tests, because slow suites reduce the benefit. Share these observations with the team in a simple chart or table and decide on one improvement at a time.

How do testers contribute in a shift-left team?

Testers help by asking questions about requirements, suggesting examples and edge cases, reviewing unit tests for missing scenarios, building test data tools, maintaining the pipeline checks and exploring the product with fresh eyes. The role moves from only finding defects at the end to helping prevent them. This requires closer collaboration, more technical curiosity and good communication, all of which are valuable career skills.

What is a small practice exercise?

Take a simple rule, such as free delivery for orders above a set amount. Write three Given-When-Then examples with the team, then write unit tests in TDD style for the calculation. Notice how many questions arise before any code exists, and record them. That count shows the value of early conversation and gives you a story to tell in interviews.

What tools support these practices?

Unit test frameworks, Cucumber or similar tools for BDD, linters for static analysis and pipeline servers for fast feedback. Choose tools after agreeing the practice, not before.

Frequently asked questions

What is shift-left testing?

It means moving testing and quality activities earlier in the development life cycle to find and fix problems while they are small and cheap.

What are the steps of TDD?

Write a failing test (red), write the simplest code to pass it (green), then refactor while keeping the tests passing.

What is Given-When-Then?

It is a format for examples: Given a starting state, When an action occurs, Then an expected result follows, used in BDD scenarios.

Is BDD only about Cucumber?

No. BDD is mainly a collaboration practice for shared understanding, and tools like Cucumber are optional ways to automate the examples.

Do TDD and BDD remove the need for later testing?

No. They reduce defects early, but system, exploratory and non-functional testing are still needed.

Sources and further reading

- ISTQB glossary and resources
- ISTQB glossary
- Cucumber documentation

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