Exercise walk-through

12.5
Testing and continuous integration

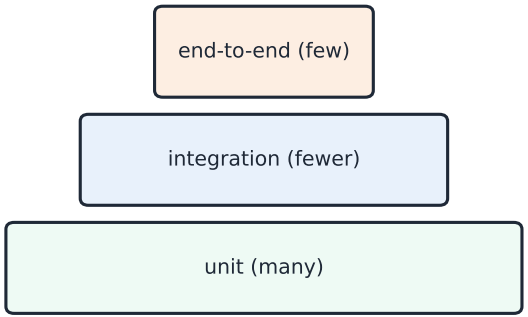
University Computer Science

Testing: the pyramid and the pipeline

1. **Coverage** 92% line, 81% branch: the gap is decisions never exercised - the branch list is the work queue behind an 85% gate.
2. **Levels** FR-2 at unit level: due = date + 21 with normal, leap February and year-boundary cases; integration reads the loan back; end-to-end scripts the member.
3. **Loop** Push the failing sixth-borrow test: red; implement the limit: green, coverage rises, merge unblocked.

Test coverage and CI flow

the pyramid



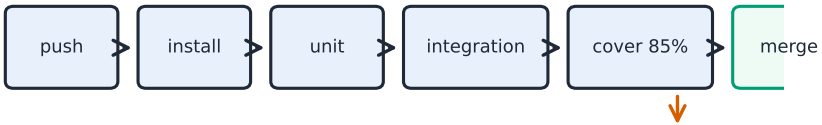
92% line, 81% branch — the branch list is the queue

FR-2 at all three levels:

unit: +21 days (leap Feb, year end)

integration: loan reads back; e2e: scripted member

the pipeline



red blocks the merge

red-green loop: failing test for the sixth borrow -> CI red -> implement the limit check -> green, coverage rises, merge

assert behaviour (the due date), not internals

a test without an assertion is coverage theatre

> Push triggers install, unit, integration, the 85% gate - green merges.

Exercises 2.1·2.3

2.1 Why do unit tests form the pyramid's base? [3]

2.2 Line 92% but branch 81% – what does the gap mean? [3]

2.3 What is a test without an assertion? [3]

Exercises 2.1·2.3

2.1 Why do unit tests form the pyramid's base? [3]

2.1

- Fast and isolated: they run constantly and pinpoint failures ✓.

2.2 Line 92% but branch 81% – what does the gap mean? [3]

2.2

- Decisions never exercised: each untested branch is an untested behaviour ✓.

2.3 What is a test without an assertion? [3]

2.3

- Coverage theatre – a number without a verdict ✓.

Exercise 3.1

3.1 The renewal feature (FR-4) lands next sprint.

Design its tests at all three levels: name the unit cases (dates included), the integration case, and the end-to-end case. State which level catches a leap-February bug fastest and why. State the CI order and where the coverage gate sits. [8]

Exercise 3.1

3.1 The renewal feature (FR-4) lands next sprint.

Design its tests at all three levels: name the unit cases (dates included), the integration case, and the end-to-end case. State which level catches a leap-February bug fastest and why. State the CI order and where the coverage gate sits. [8]

3.1

- Unit: due extension arithmetic – renew from D gives $D+14$, twice gives $D+28$, third refused; cases with leap February and year boundary dates ✓.
- Integration: the renewed due date persists and reads back ✓.
- End-to-end: a scripted member renews and the screen shows the new due date ✓.
- Leap-February: the unit level – pure date arithmetic, runs in milliseconds, no setup ✓.
- CI order: install → unit → integration → the 85% coverage gate → merge; red at any stage blocks ✓.

Exercise 3.2

3.2 A PR arrives with 93% line coverage on the new module and one test whose body is empty (no assertions).

Classify the coverage claim honestly. State what the empty test contributes and what it should contain. Give the red-green loop for the reservation-blocks-renewal rule from a failing test to merge. [7]

Exercise 3.2

3.2 A PR arrives with 93% line coverage on the new module and one test whose body is empty (no assertions).

Classify the coverage claim honestly. State what the empty test contributes and what it should contain. Give the red-green loop for the reservation-blocks-renewal rule from a failing test to merge. [7]

3.2

- 93% line with an empty test: line coverage counts executed lines, so the claim is true and worthless – the behaviour is unverified ✓.
- The empty test contributes nothing but coverage; it must assert the observable outcome (the due date, the refusal) ✓.
- Loop: write the failing test (reservation present, renew refused) → push, CI red → implement the reservation check → CI green, coverage rises → merge unblocked ✓.