Testable Architecture

Matthew Renze
SOFTWARE CONSULTANT

@matthewrenze   www.matthewrenze.com

Overview

Test-Driven Development

Test Automation Pyramid

Pros and Cons

Demo

The Current State of Testing

Very little testing

Ineffective testing

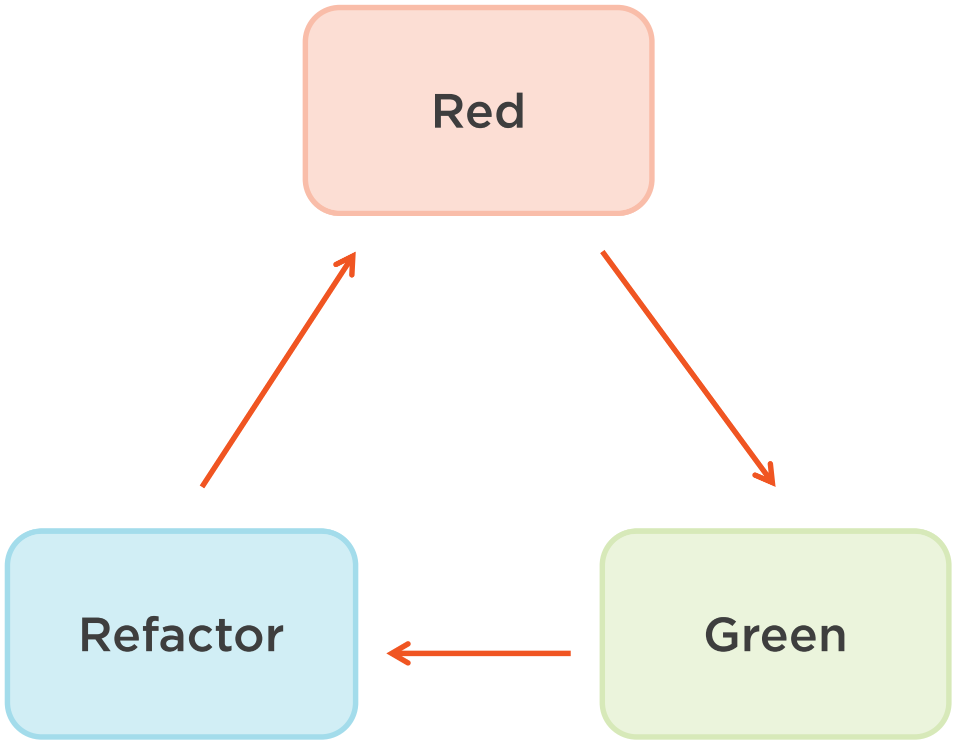
Inefficient testing

Not enough time

Not my job

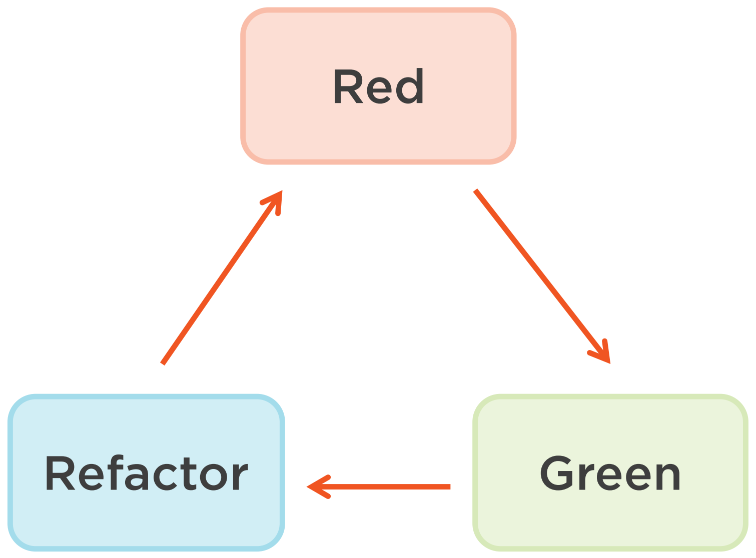
It’s too hard

Test-Driven Development



Test-Driven Development

1. Create a failing test
2. Get the test to pass
3. Improve the code



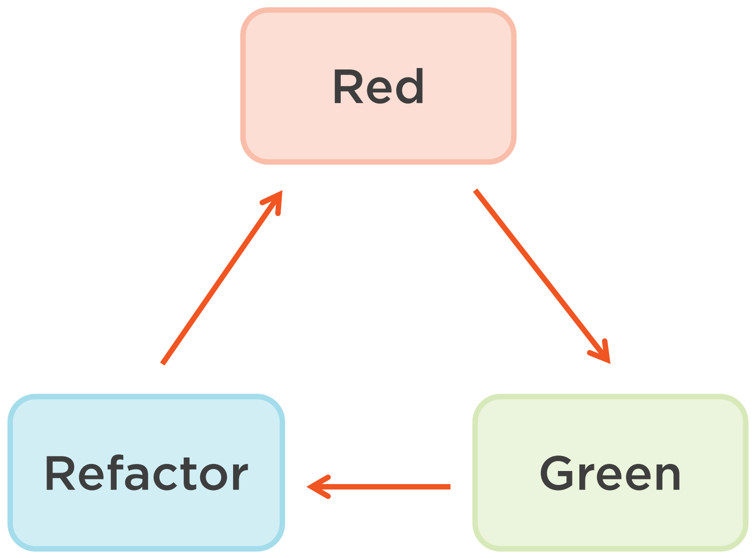
Test-Driven Development

Comprehensive suite of tests

Drives testable design

More maintainable

Eliminates fear



Types of Tests

Unit tests

Integration tests

Component tests

Service tests

UI tests

Functional tests

Acceptance tests

Smoke tests

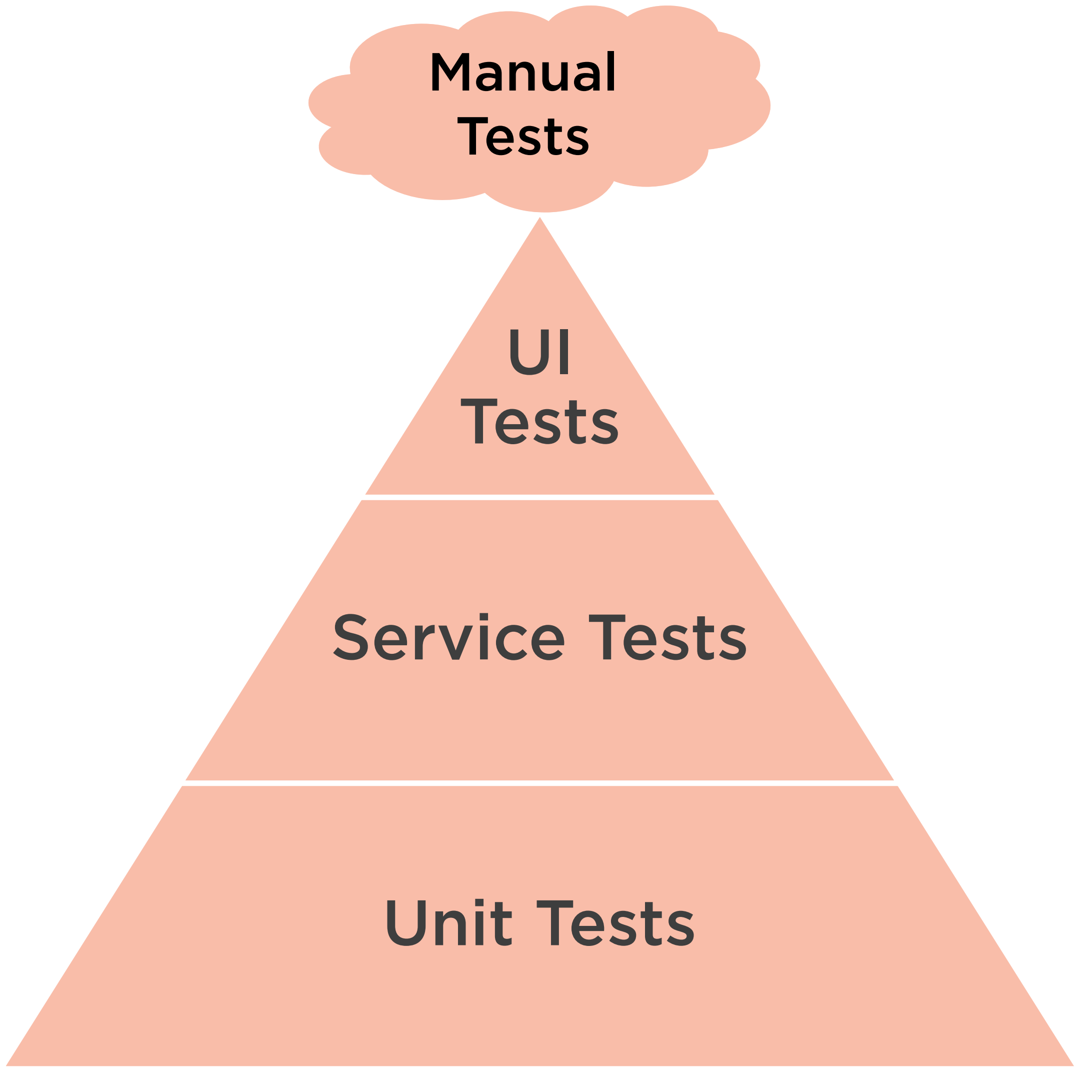
Exploratory tests

Automated tests

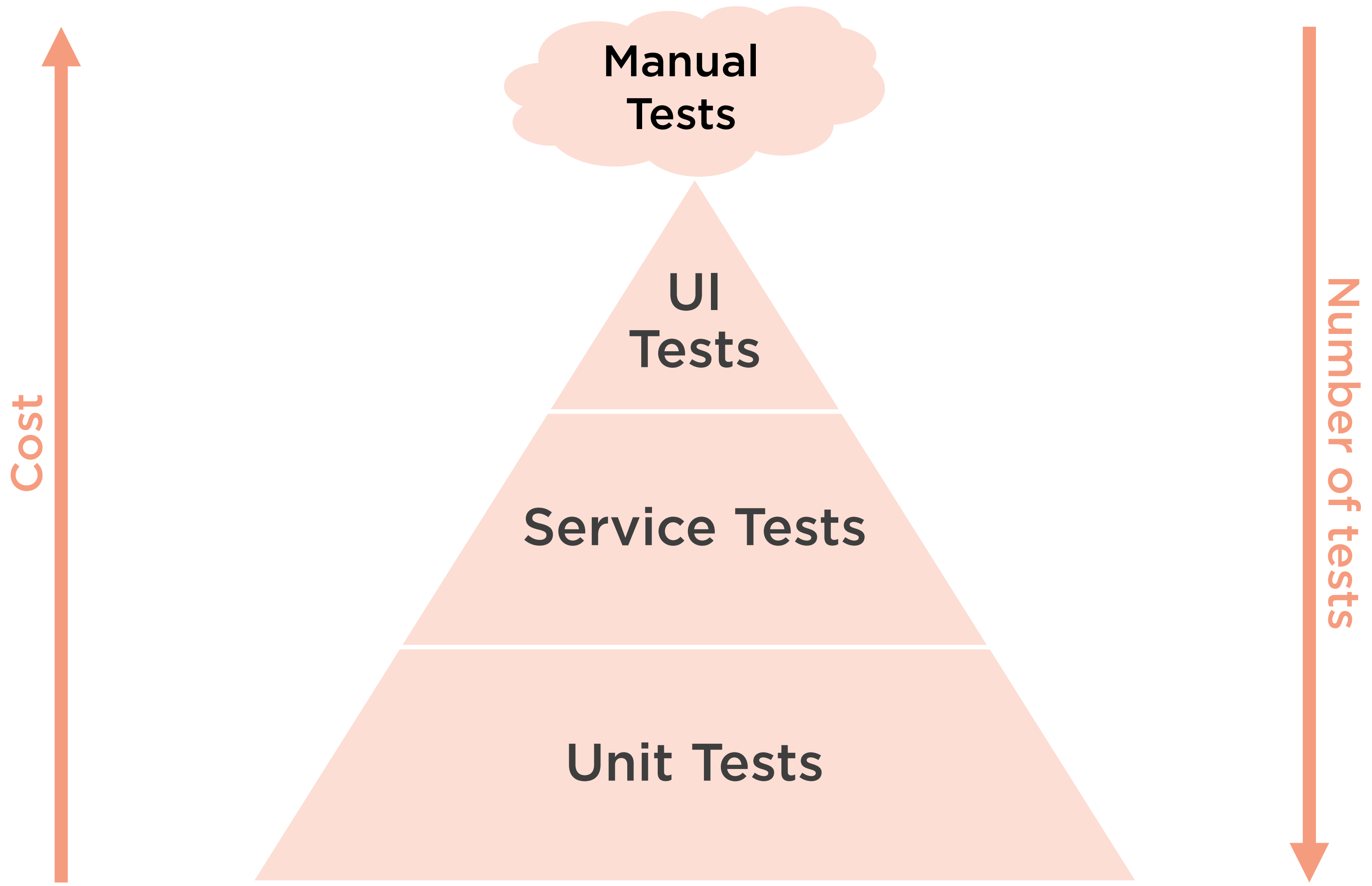
Semi-automated tests

Manual tests

Test Automation Pyramid



Test Automation Pyramid



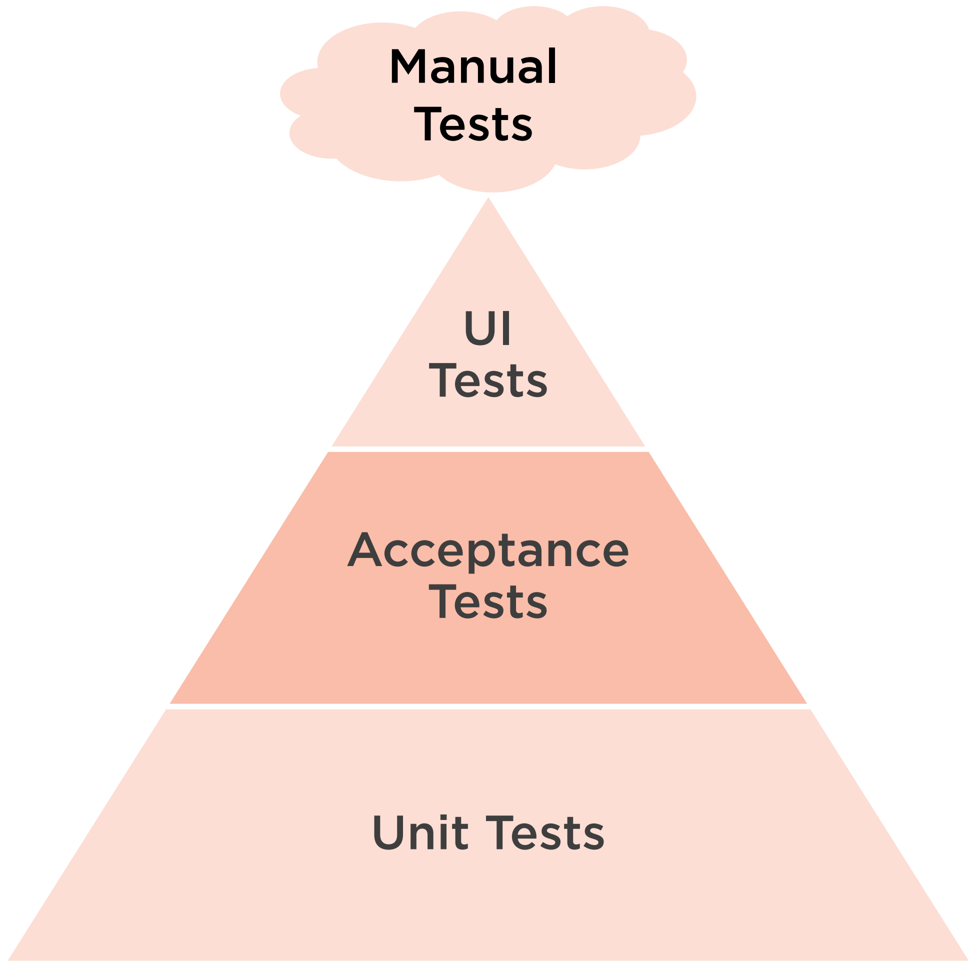
Acceptance Tests

Verify functionality

Language of the business

Criteria for completeness

Full tests are problematic

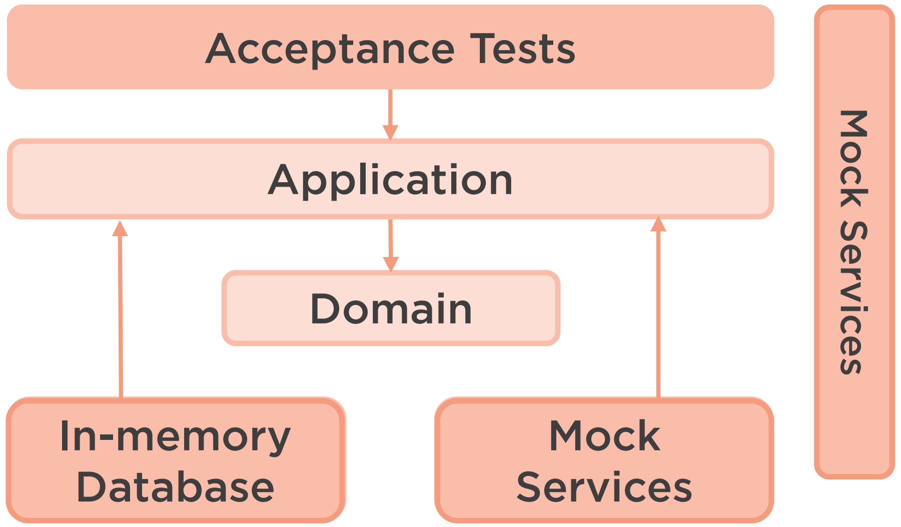


Acceptance Tests

Eliminate user interface

Eliminate database

Eliminate dependencies



Acceptance Tests

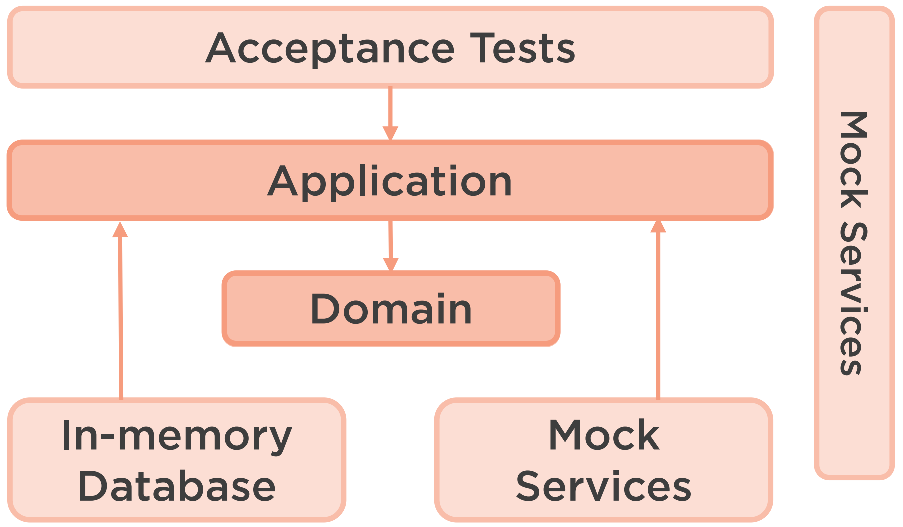
Focus on the essential

Minimize coded UI tests

Smoke test instead

Minimize manual tests

Exploratory test instead



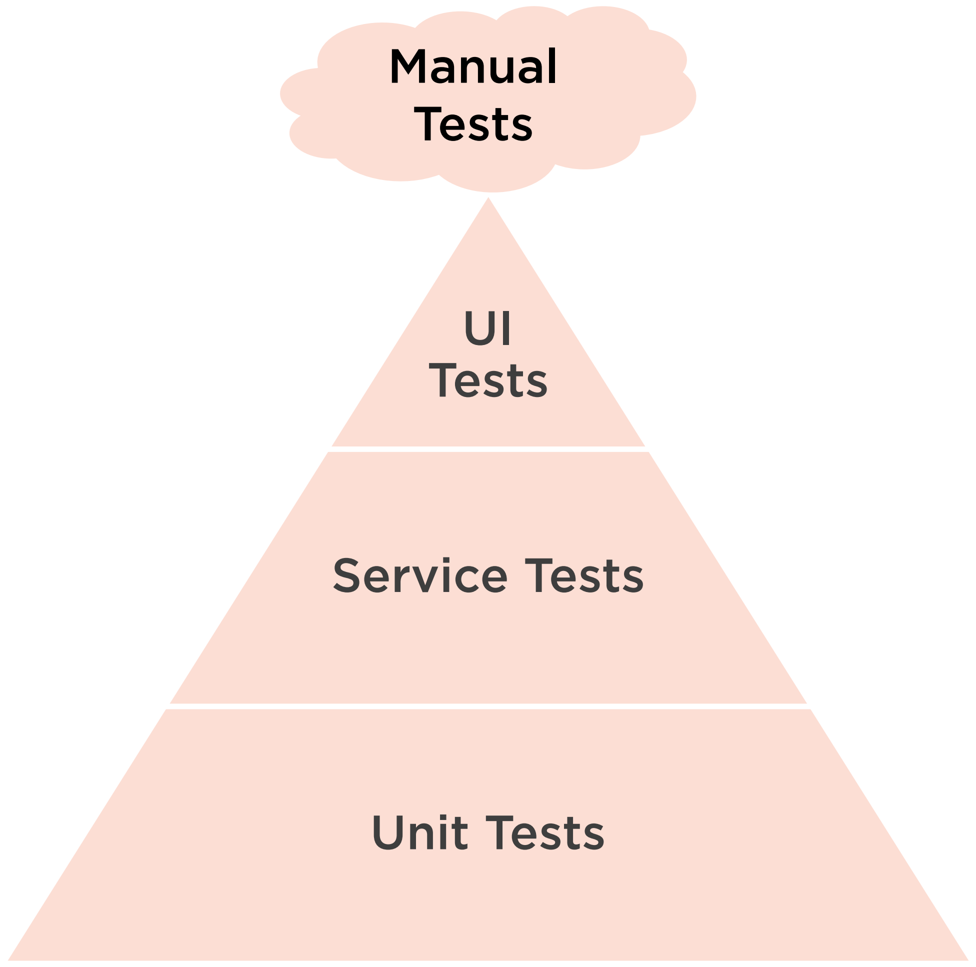
Why Create Testable Architecture?

Pros

Easier to test

Improves design

Eliminates fear



Why Create Testable Architecture?

Pros	Cons
Easier to test	Higher up-front cost
Improves design	TDD requires discipline
Eliminates fear	Requires team buy-in

Setup

Show SaleTests (Top)

Show SaleTests (Tests)

Show CreateSaleCommandTests (Top)

Show CreateSaleCommandTests (SetUp)

Show CreateSaleCommandTests (Add)

Show CreateSaleCommandTests (Save)

Show CreateSaleCommandTests (Notify)

Show CreateSaleCommandTests (End)

Show CreateASale Feature Tests

Show CreateASale Steps

Wrap Up

Summary

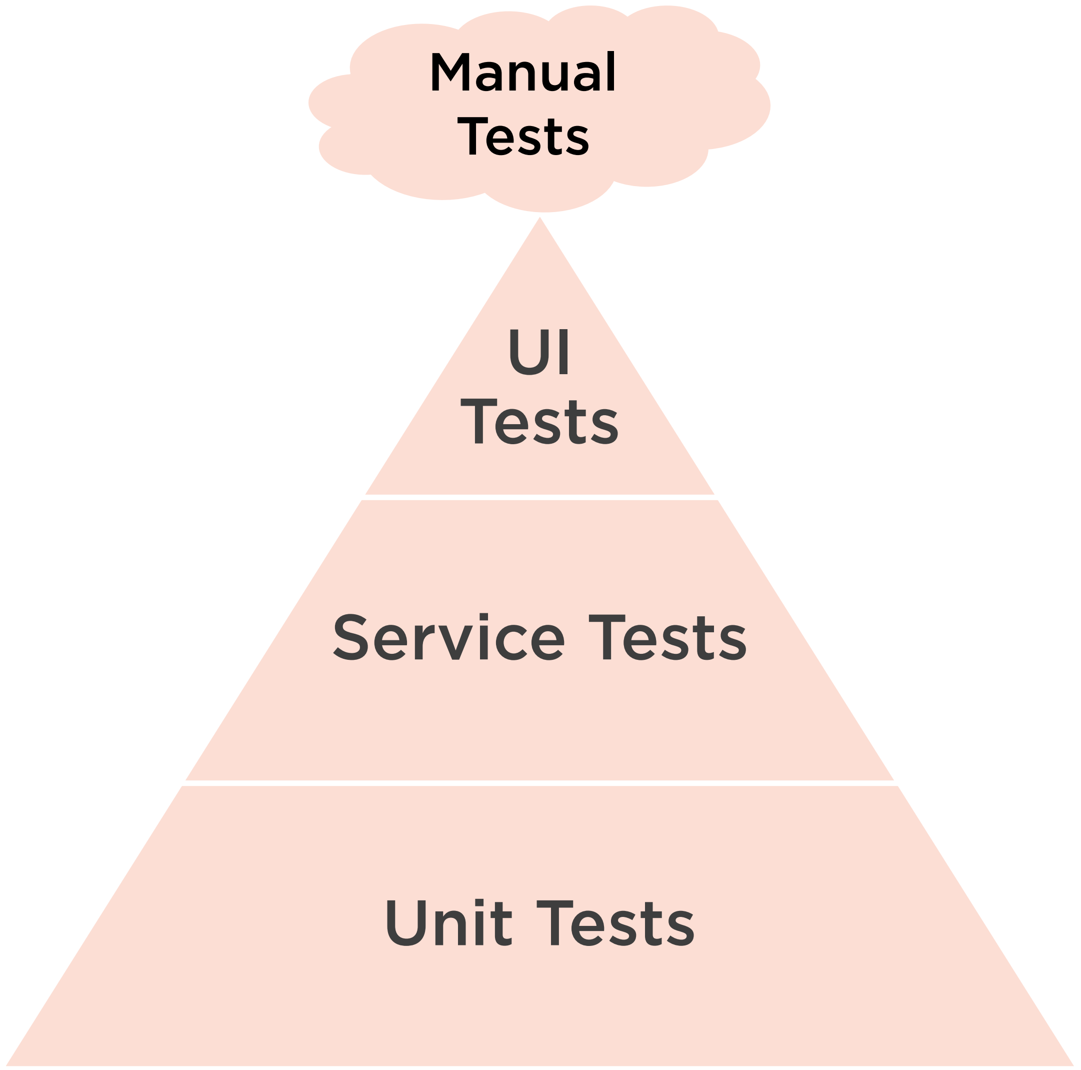
Test-Driven Development

Test Automation Pyramid

Pros and Cons

Demo

Test Automation Pyramid



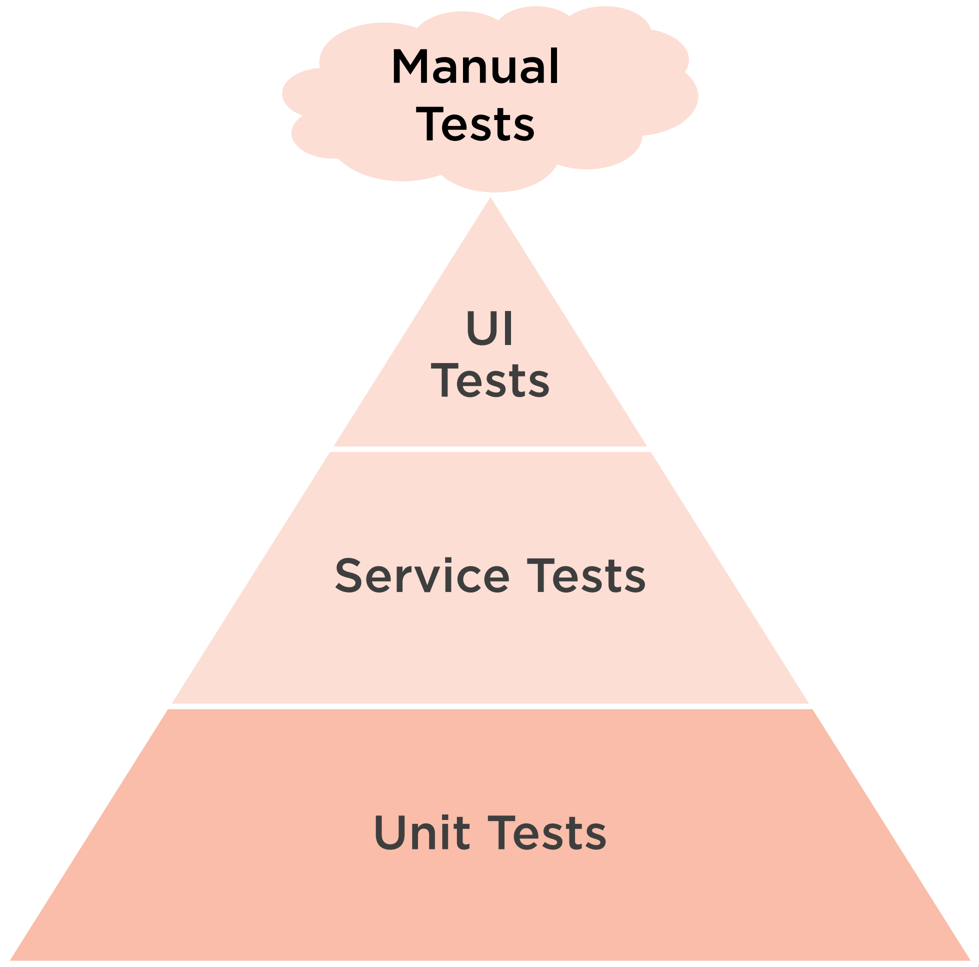
Unit Tests

Verify a unit of code

Creates seams in code

Mock out dependencies

Test in isolation



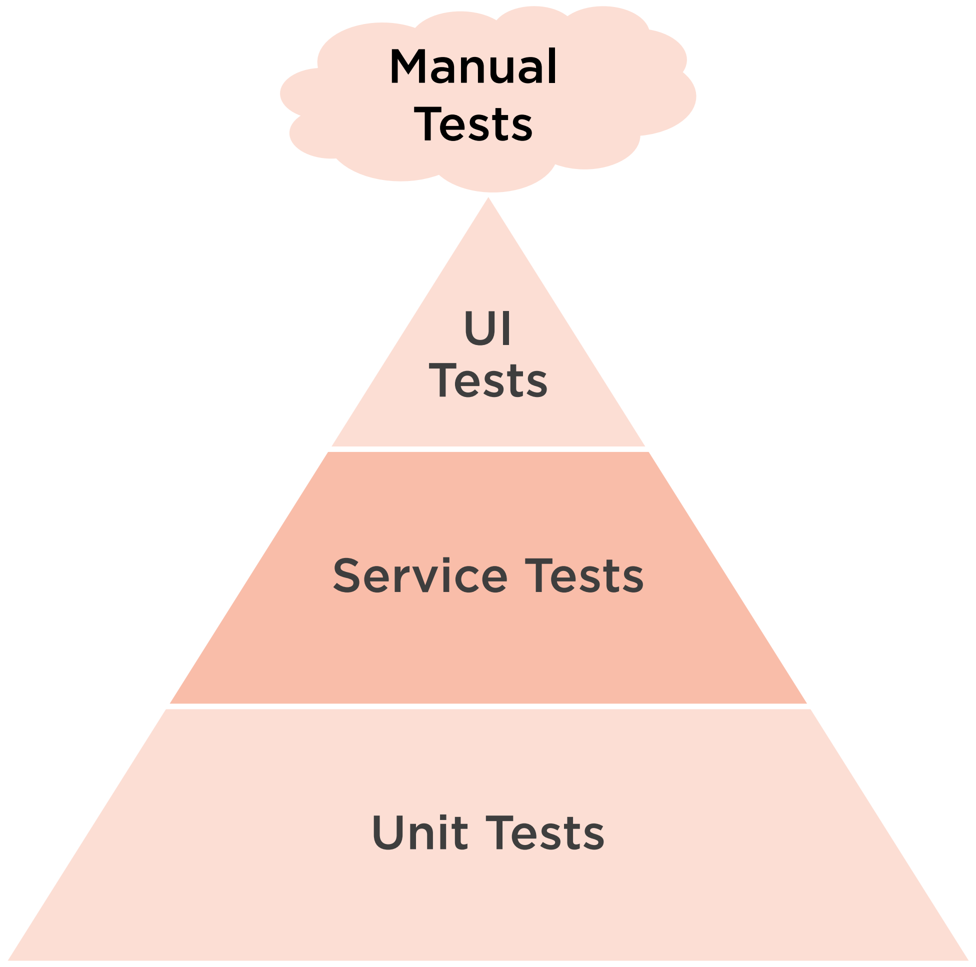
Service Tests

Verify functionality

Set of services

Covers service code

Tested in isolation



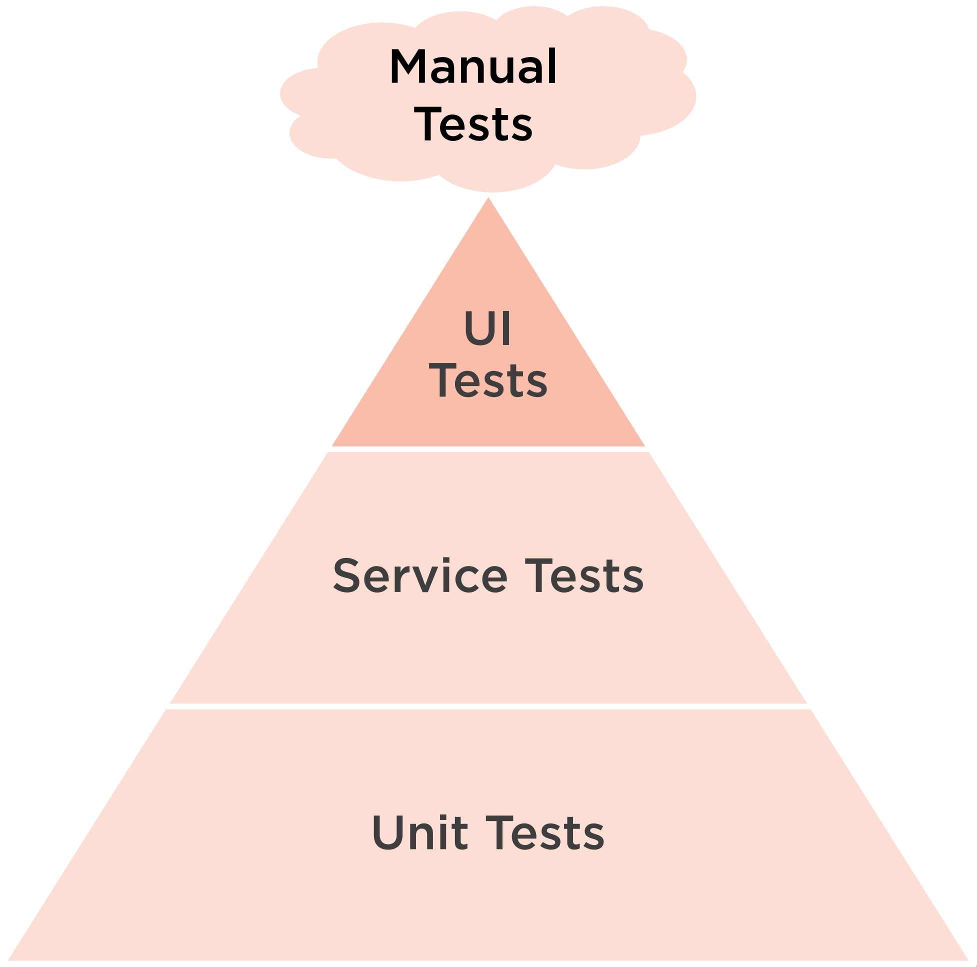
UI Tests

Verify full functionality

High cost

Very brittle

Should be minimal



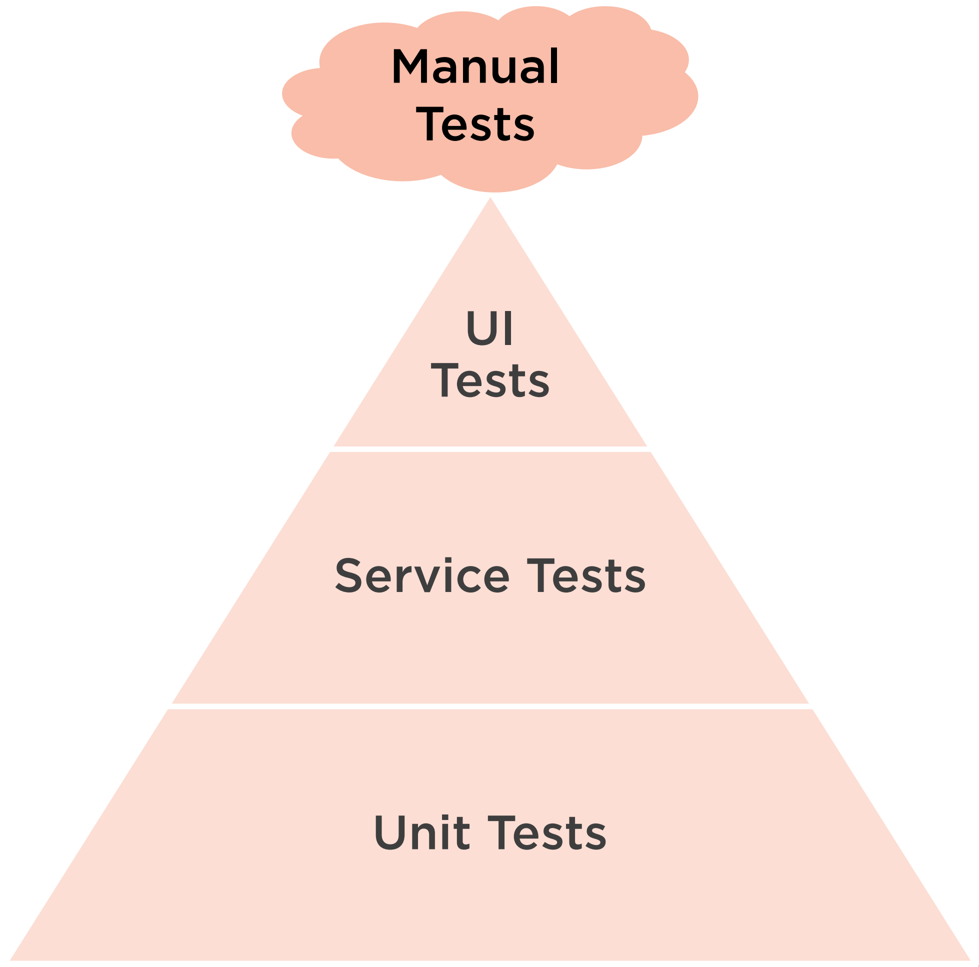
Manual Testing

Test by hand

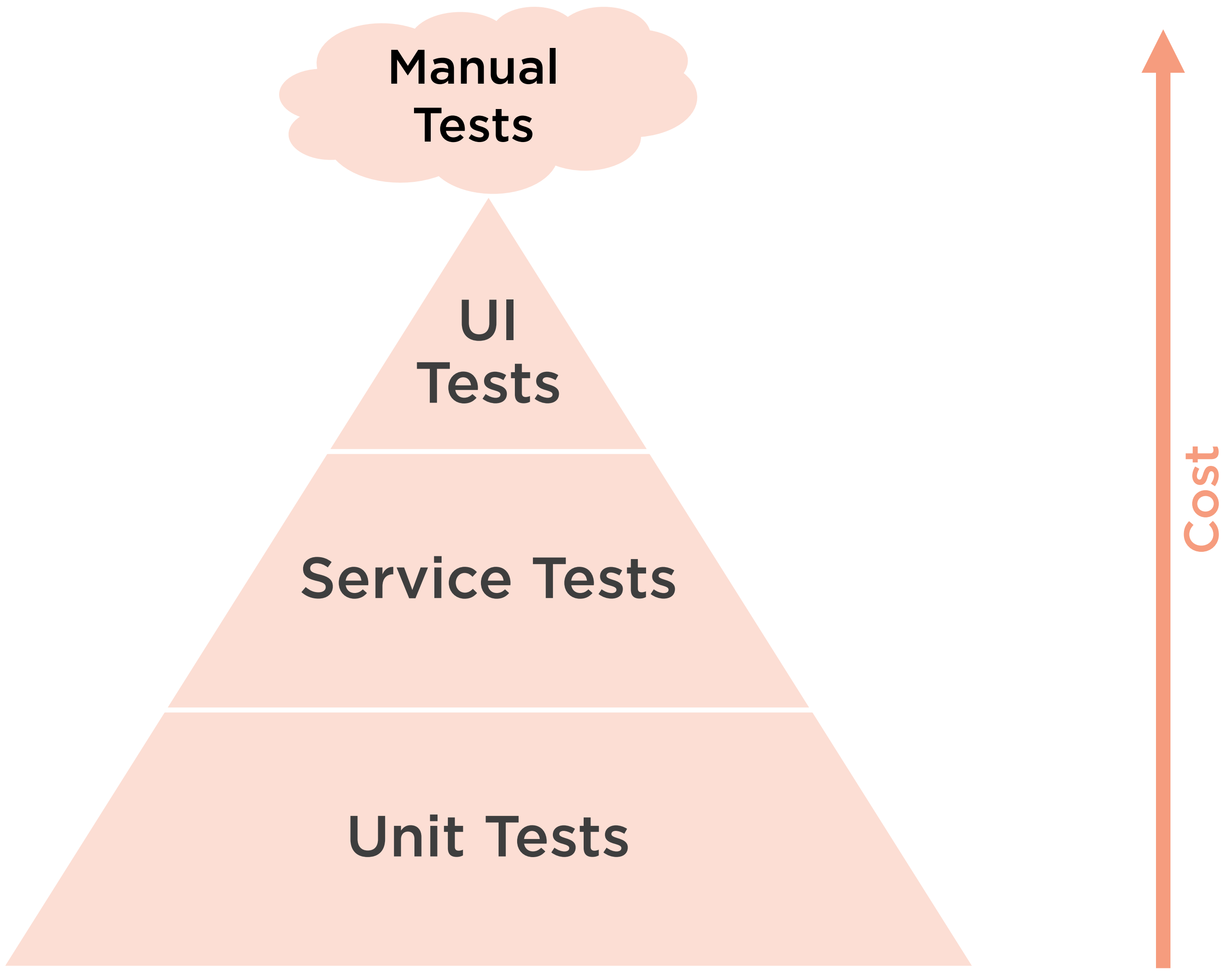
Most expensive

Use where appropriate

Automate to free up testers



Test Automation Pyramid



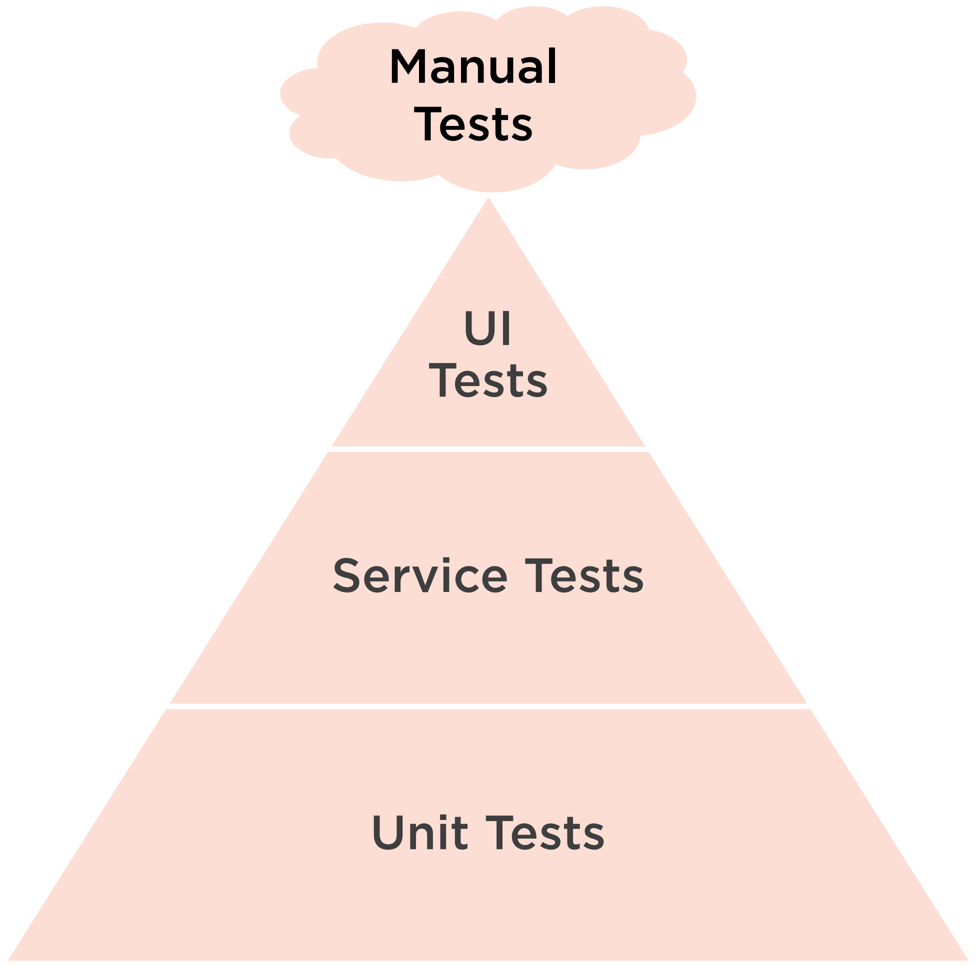
Acceptance Tests

Verify functionality

Language of the business

Criteria for completeness

Full tests are problematic



Acceptance Tests

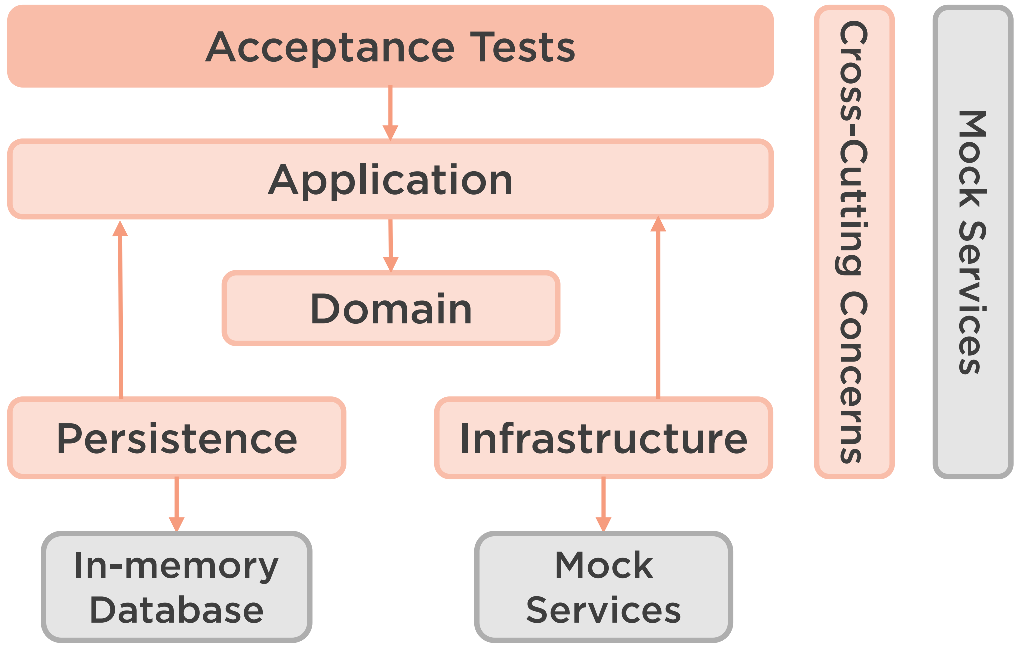
Eliminate user interface

Eliminate database

Eliminate dependencies

Minimize coded UI tests

Minimize manual tests


Acceptance Tests
Application
Domain
Persistence
Infrastructure
In-memory Database
Mock Services
Cross-Cutting Concerns
Mock Services