TEST DRIVEN DEVELOPMENT

Modern Test Pyramid - Illustrated

System Level Tests, Component Level Tests, Unit Level Tests

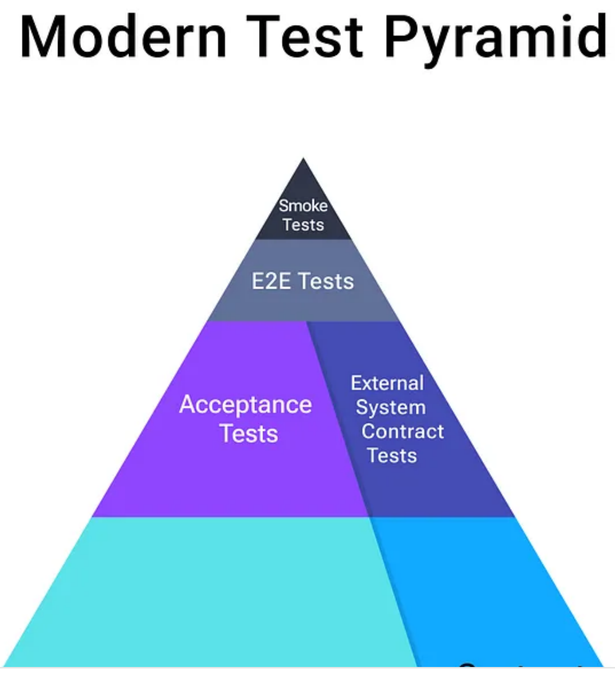
VALENTINA JEMUOVIĆ
APR 25, 2025 · PAID

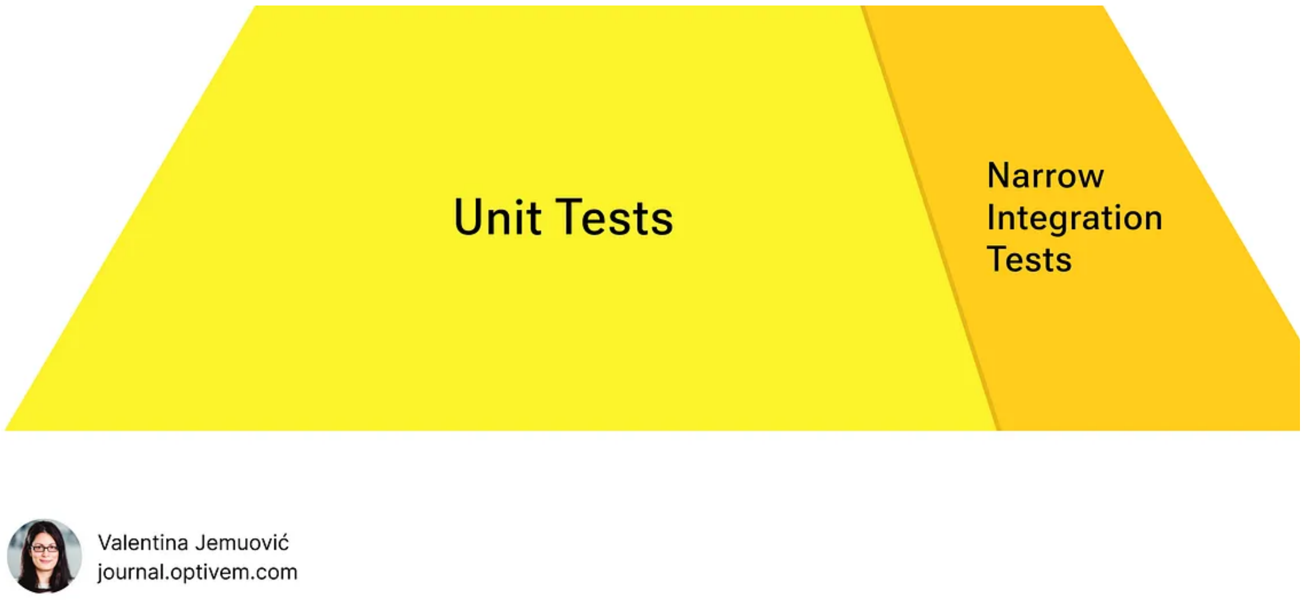
15 5 3 Share

📢 My **next Live Q&A Session** about **Unit Tests** is on Wed 30th April 17:00. If y
an Optivem Journal paid member, get your 100% discount ticket.

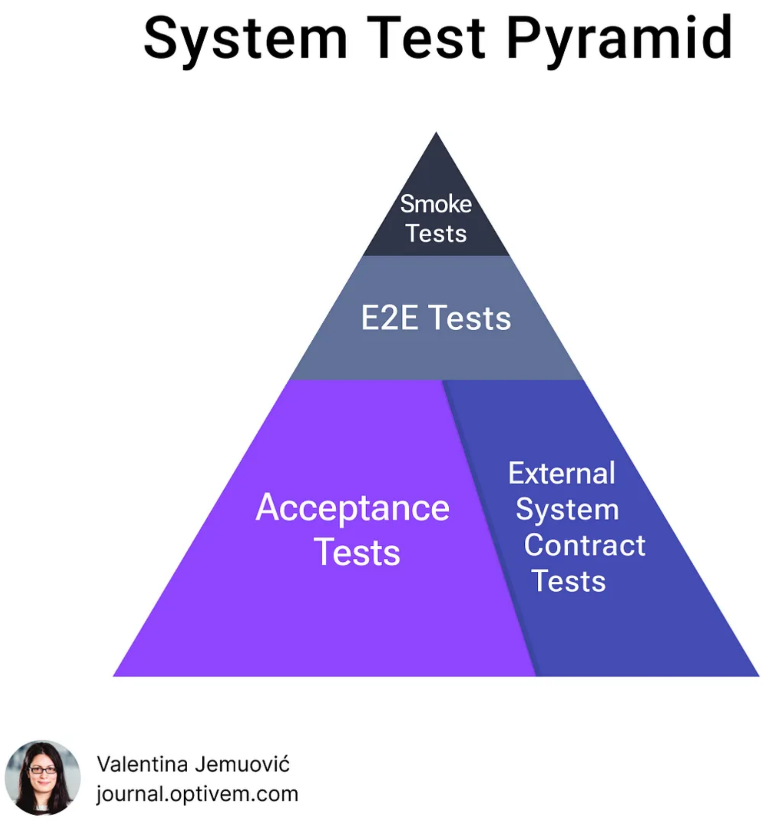
Upgrade to join the Live Q&A for free:

Previously, we introduced the Modern Test Pyramid:





Let’s break it down, firstly we have the System Test Pyramid, whereby we test t
whole System, as a black-box:



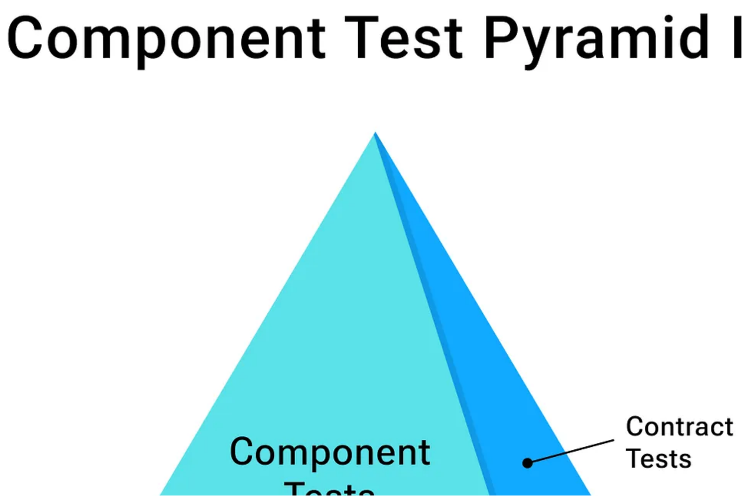
The System is composed of Component(s), for example:

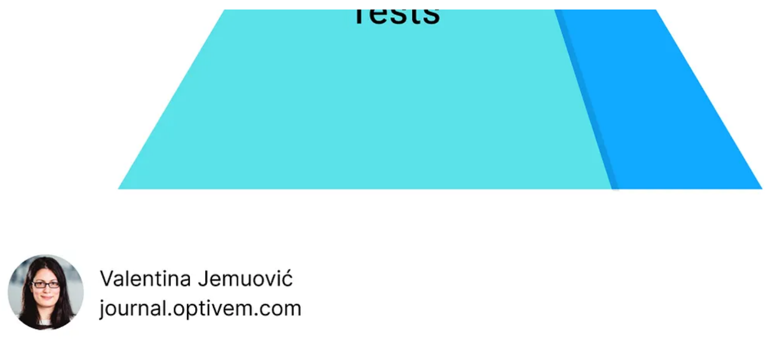
- Monolith:
  - Component #1: MVC App
- Frontend & Backend:
  - Component #1: Frontend
  - Component #2: Monolithic Backend
- Frontend & Microservices:
  - Component #1: Frontend
  - Component #2: Order Microservice
  - Component #3: Product Microservice
  - Component #4: Payment Microservice

For each Component, we make a decision whether to adopt:

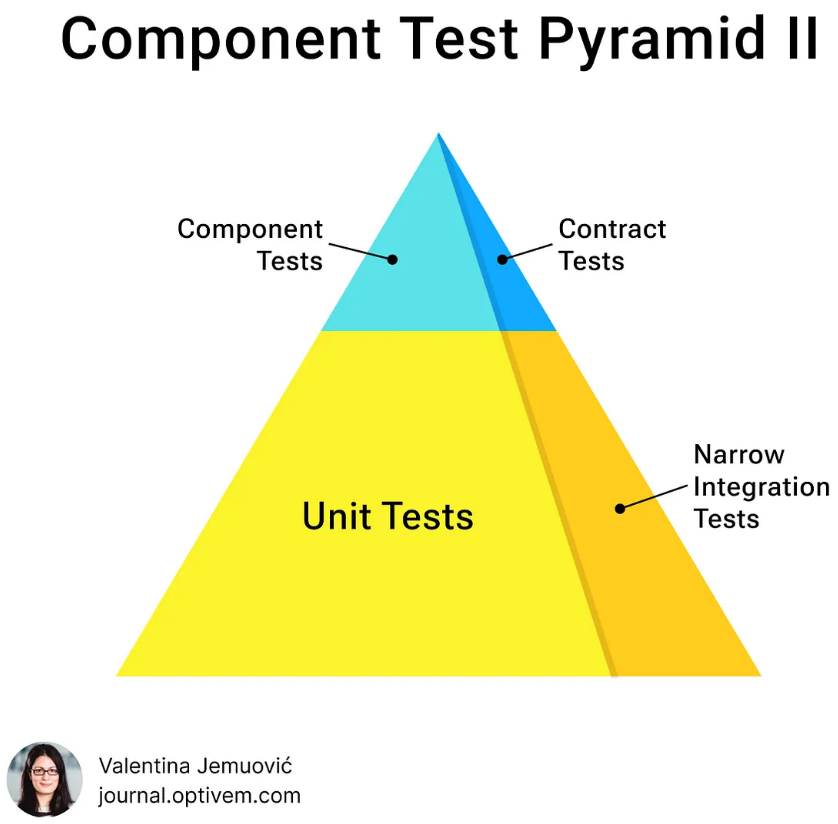
- Component Test Pyramid I; or
- Component Test Pyramid II

If the Component has relatively low business complexity, then it can make sens just adopt the following:

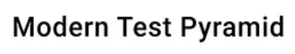


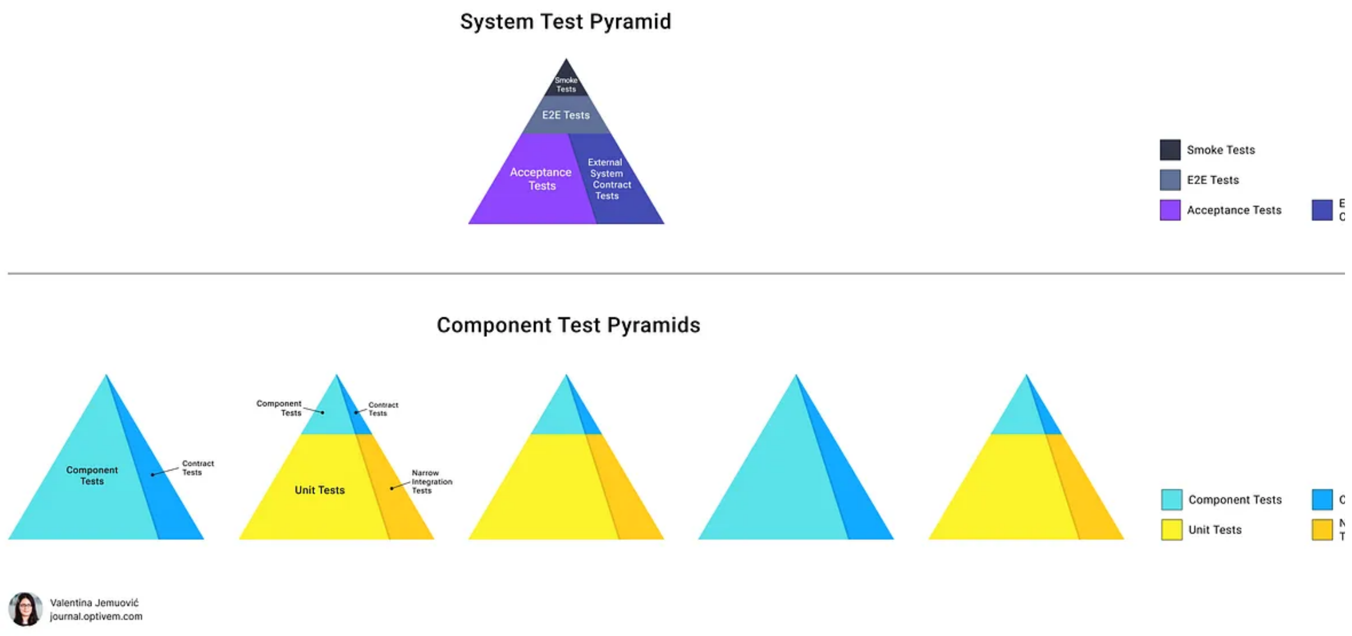


However, if the Component has higher business complexity, then the following would be more helpful in providing us with faster feedback:



Putting it all together, we have the single System Test Pyramid and we have Component Test Pyramid(s) which differ on per-Component basis:

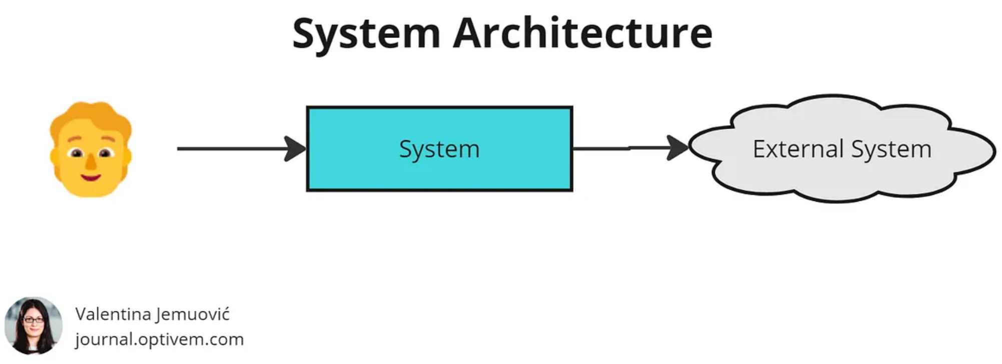




System Level

System Architecture

The User (e.g. Customer) uses our System (e.g. eShop) to satisfy some goals (e.
products online). Our System (eShop) connects to External Systems (PayPal, SA
ERP, Clock).



Note: Here we're showing only one External System, but there could be multiple External Systems.

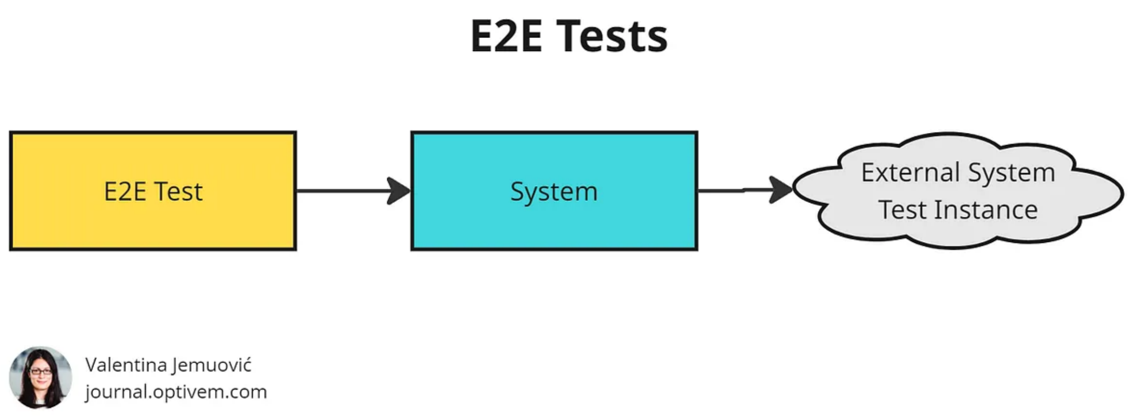
Smoke Test

A Smoke Test checks whether the System is up-and-running (e.g. does the eSho home page and other pages load).

E2E Test

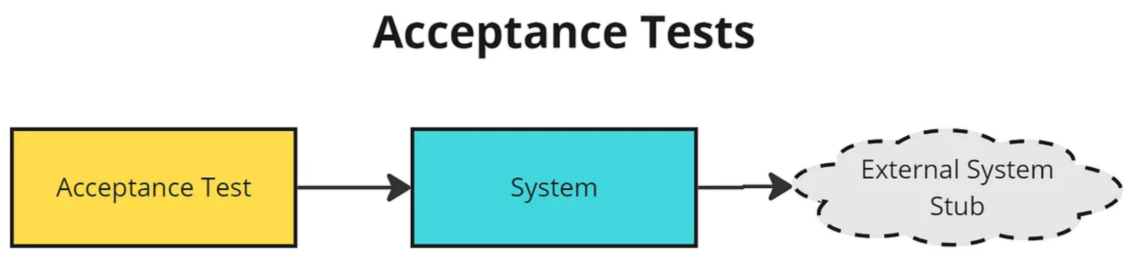
In the Production Environment, the End User uses the real System, connected t Production Instances of the External Systems.

An E2E Test is the same, except it runs in the UAT Environment (not Production our System is connected to Test Instances of External Systems.



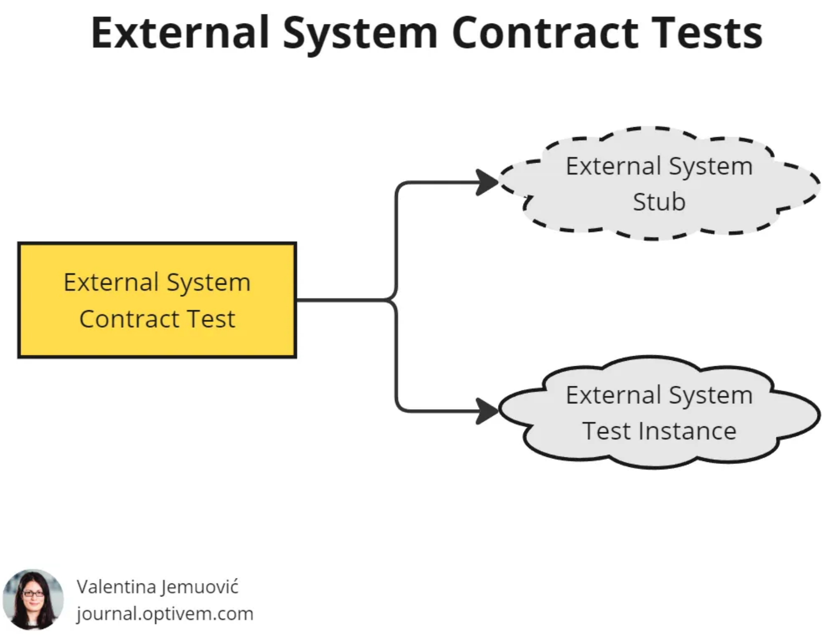
Acceptance Test

An Acceptance Test is like an E2E Test, except we stub out the External System (rather than using External System Test Instances). We do this so that we can simulate any scenario, thus we can test out any Acceptance Criteria.



External System Contract Test

Since the Acceptance Tests rely on External System Stubs, we need to be sure t
the stubs match the real External Systems.

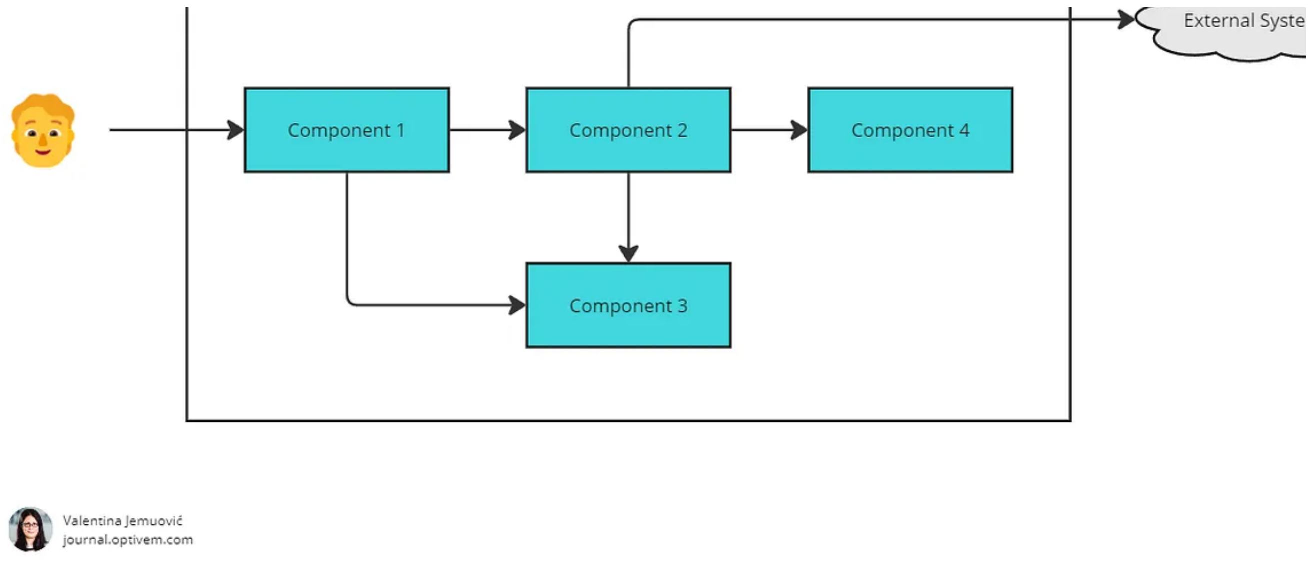

External System Contract Tests
External System Contract Test
External System Stub
External System Test Instance
Valentina Jemuović
journal.optivem.com

Component Level

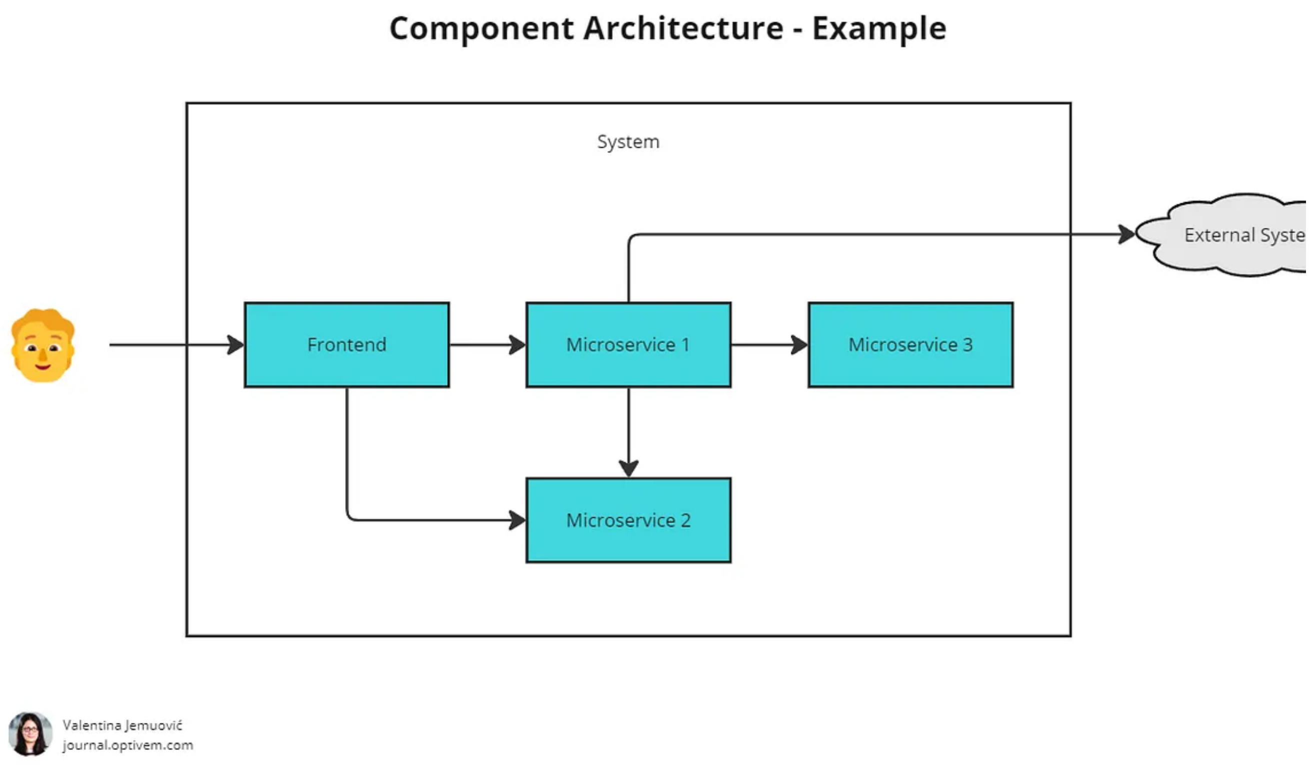
Component Architecture

Let’s zoom into the System. We can see that the System is composed of one or
Components.

Component Architecture
System

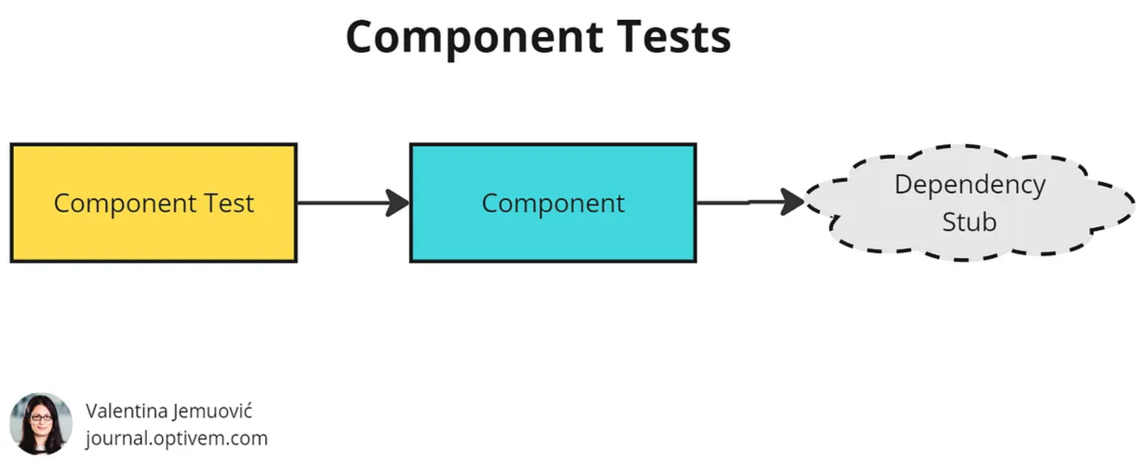


E.g. the System could be composed of Frontend & Monolithic Backend, or Front & Microservice Backend, etc.



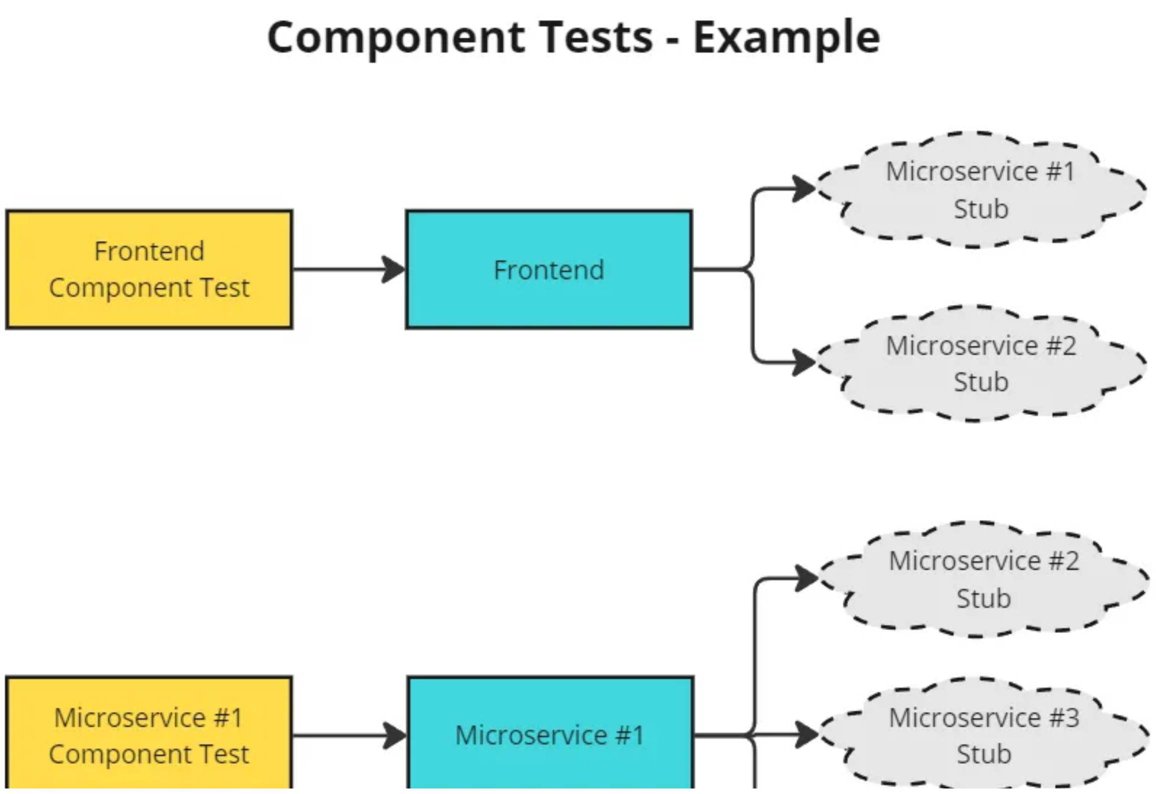
Component Tests

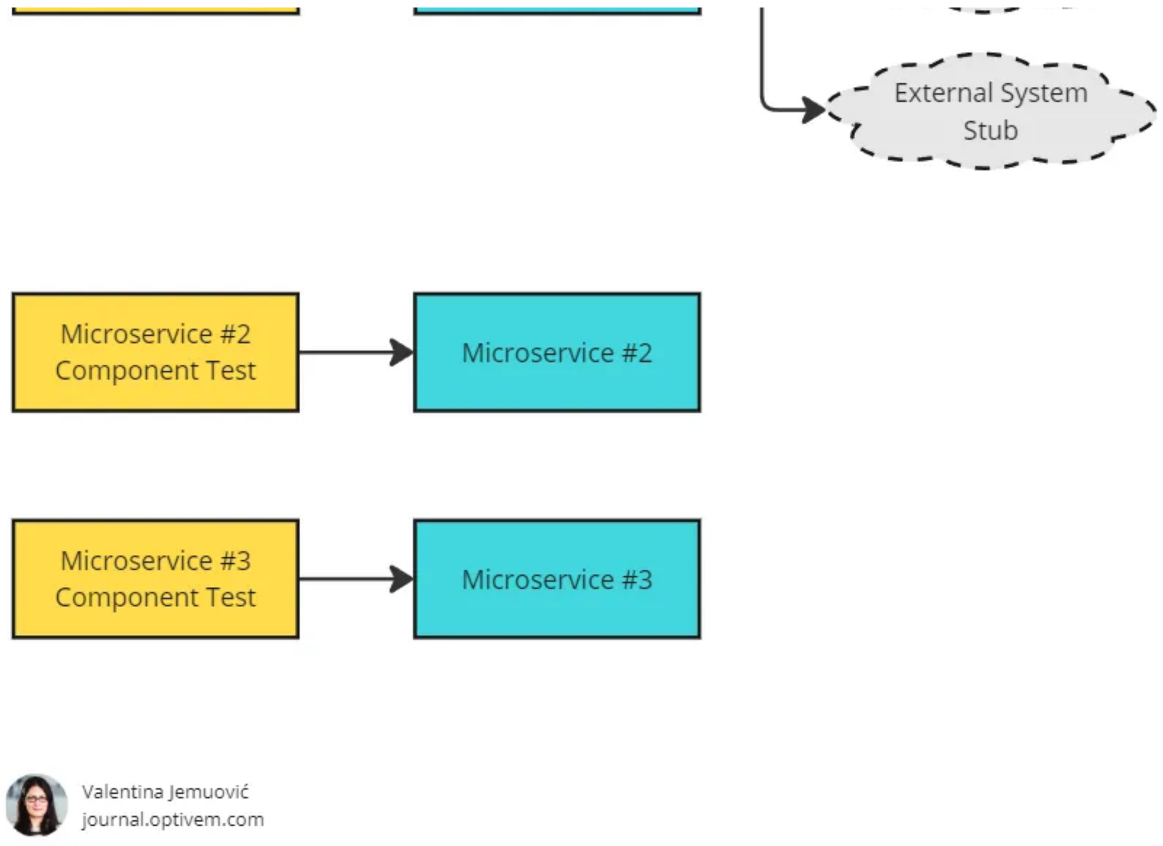
In order to test each Component in isolation, we need to stub out any depende (External Systems & other Components).



The following is an example, whereby we:

- Test Frontend in isolation - we target the UI components, and we mock out Backend.
- Test each Microservice in isolation - we target the Microservice via its API (REST API, SOAP Service, RabbitMQ consumer) and we stub out dependenc (other Microservices & External Systems)

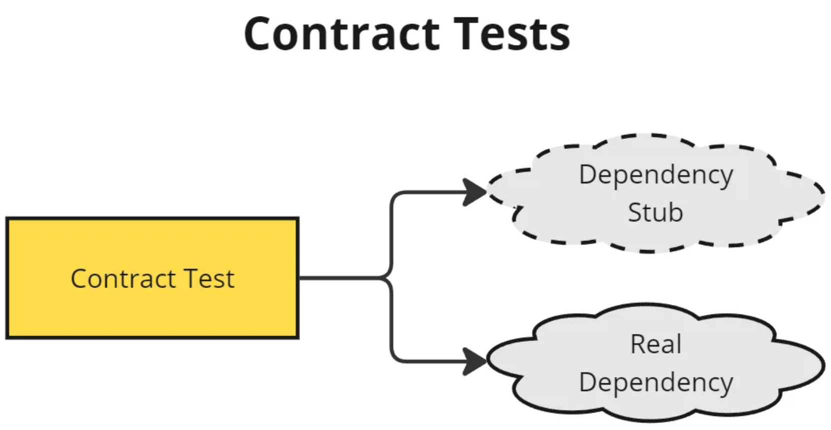




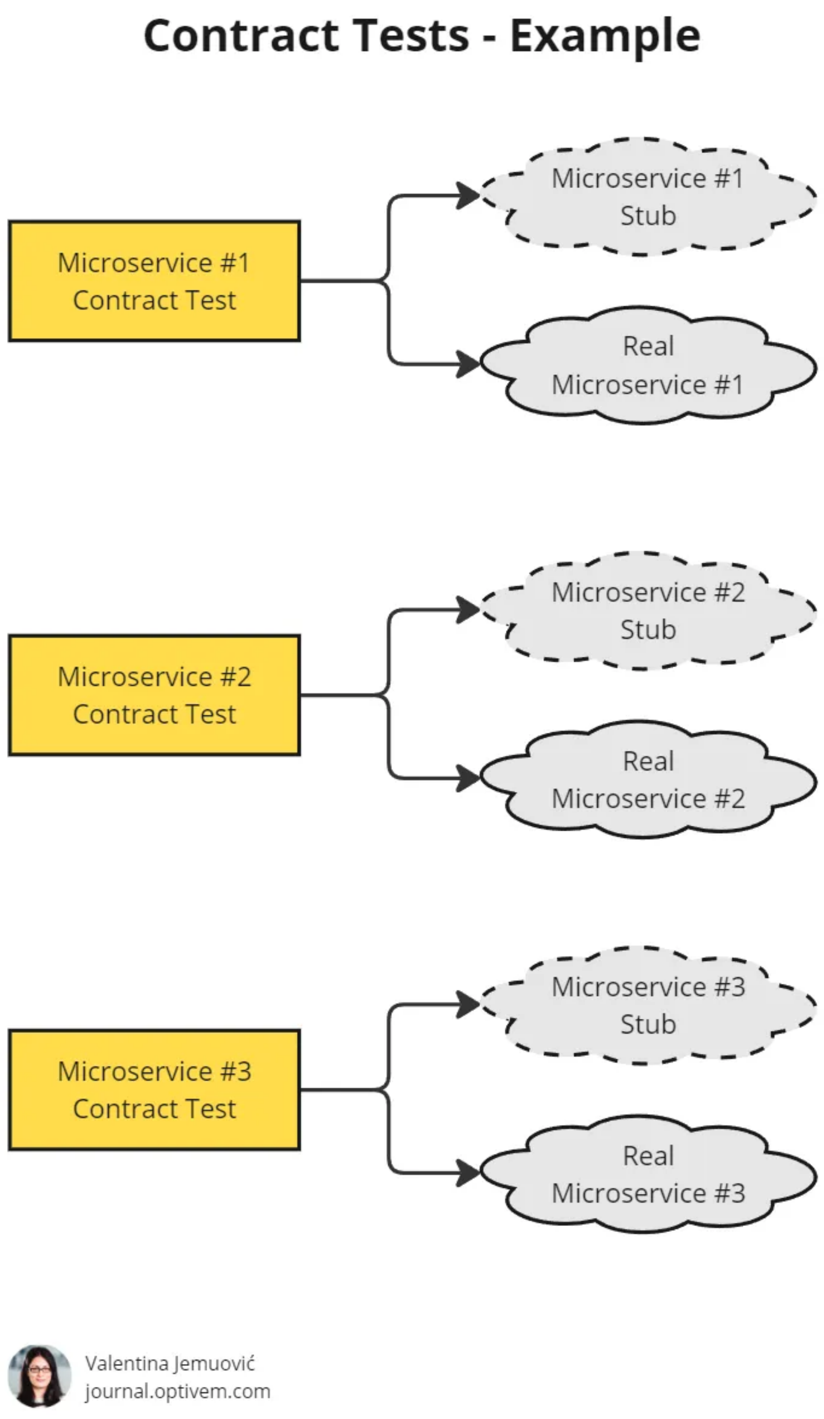
Note: We can use Dockerized instances of the database and message broker. Thus have the most realistic Component Tests.

Contract Tests

Since the Component Tests rely on Stubs, we need to have assurance that the s of the dependencies match the real instances of the dependencies.



The following is an example:

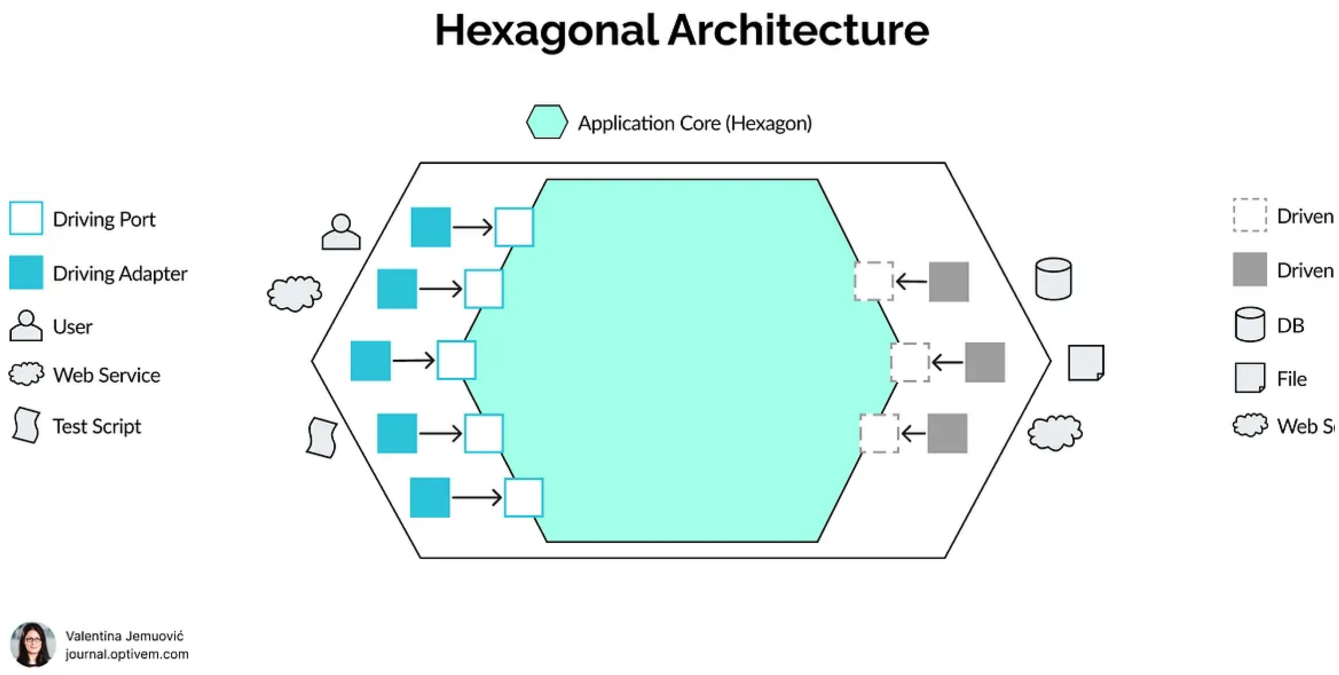


Note: For contract testing between Frontend & Backend, and within the Backend *between Microservices, it's very convenient to use a library (e.g. Pact).*

Unit Level

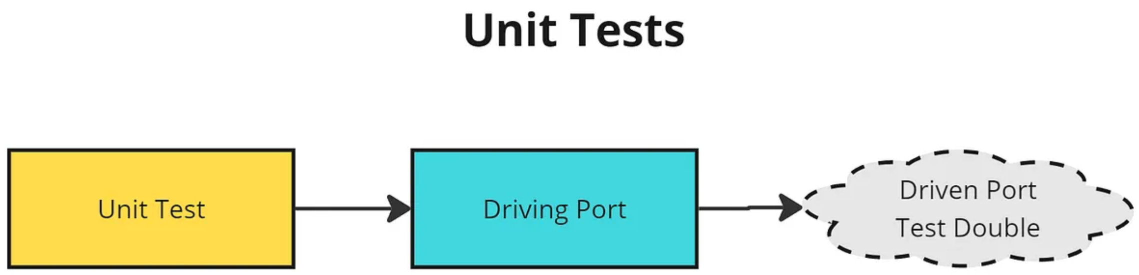
Hexagonal Architecture

Hexagonal Architecture is the fundamental architecture that enables unit testa
because it splits the business logic (Hexagon) away from I/O (adapters).

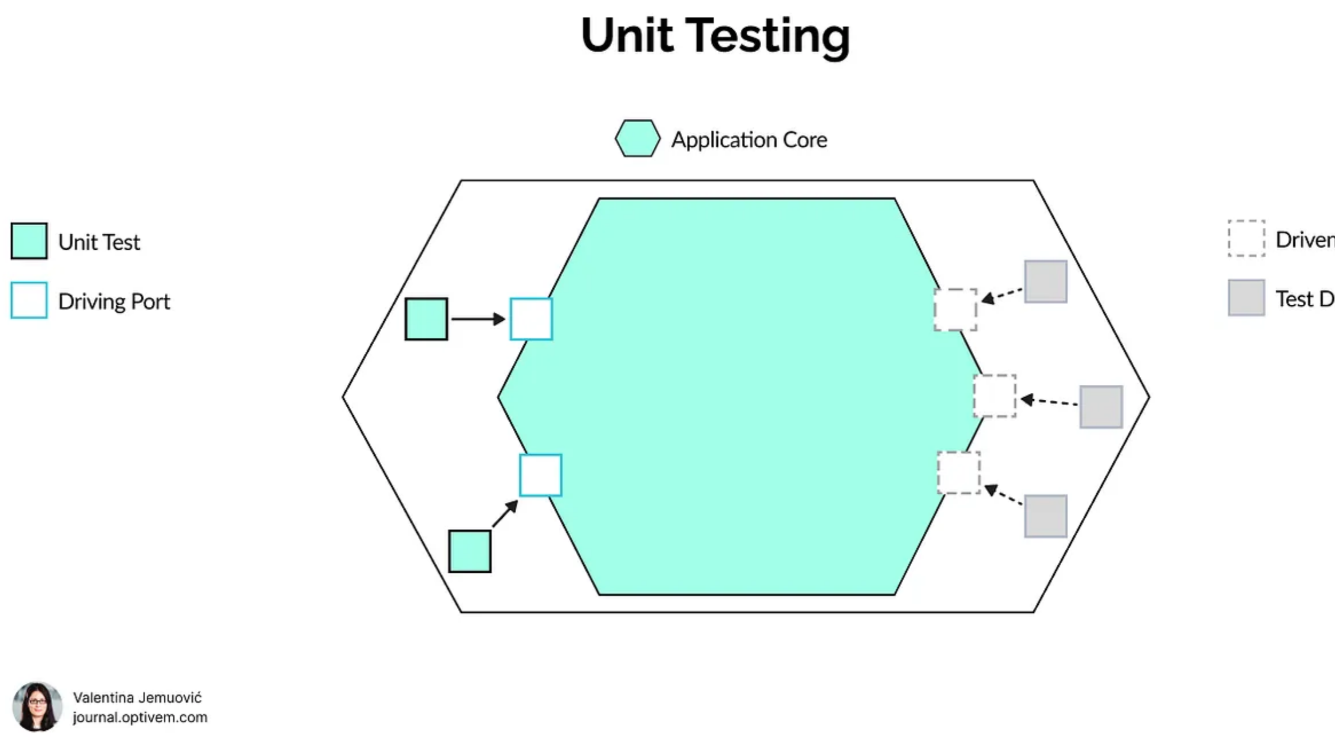


Unit Tests

Unit Tests tests the Business Logic (Hexagon). We write a Unit Test targeting a
Driving Port, and using Test Doubles for Driven Ports (so that we don't have an



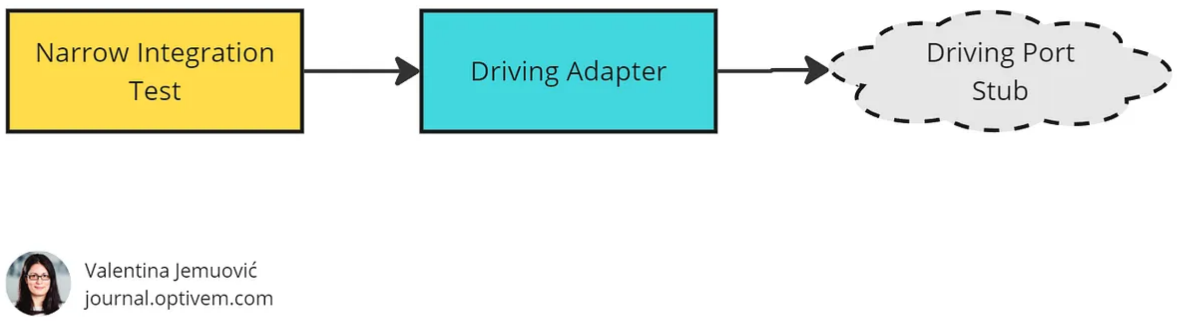
Valentina Jemuović
journal.optivem.com

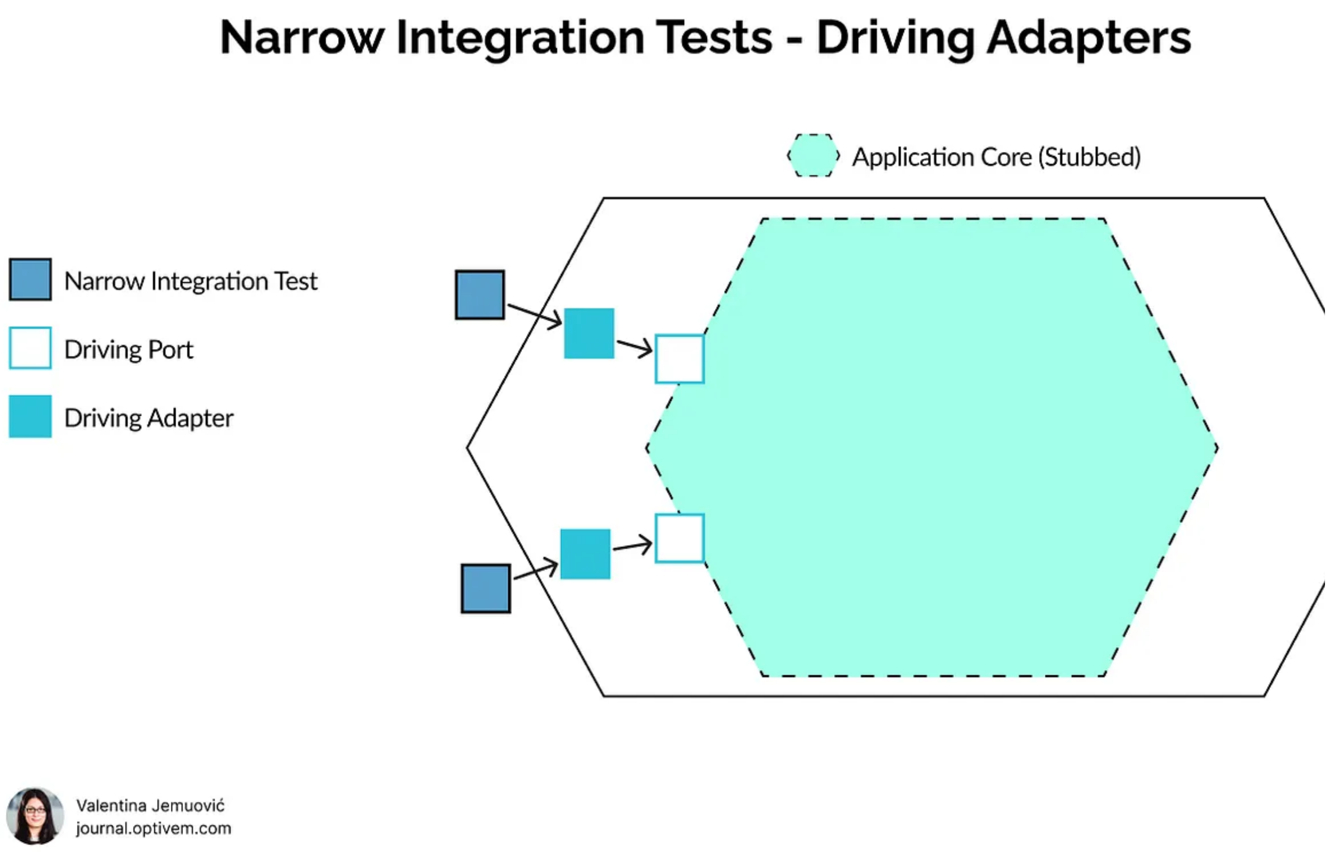


Note: Above, I've illustrated Sociable Unit Tests targeting the Driving Ports. If nee can write Unit Tests that target code inside the Hexagon (if there is a high combinatorial explosion or other mathematical/algorithmic complexity), though I this to be rarer.

Narrow Integration Tests - Driving Adapters

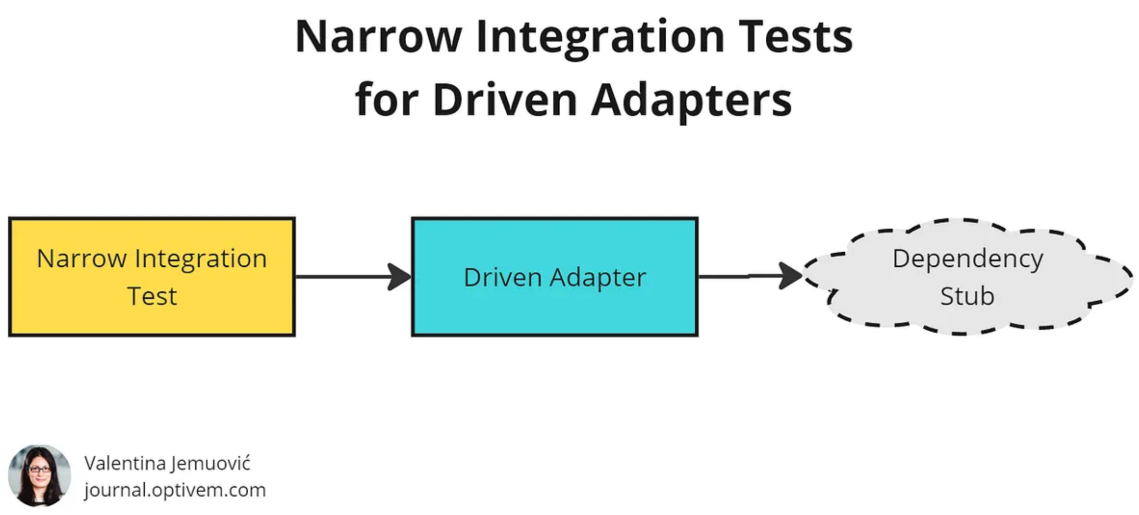
We can test Presentation Logic by testing Driving Adapters in isolation (hence stub out the Driving Ports, i.e., we stub out the Hexagon).

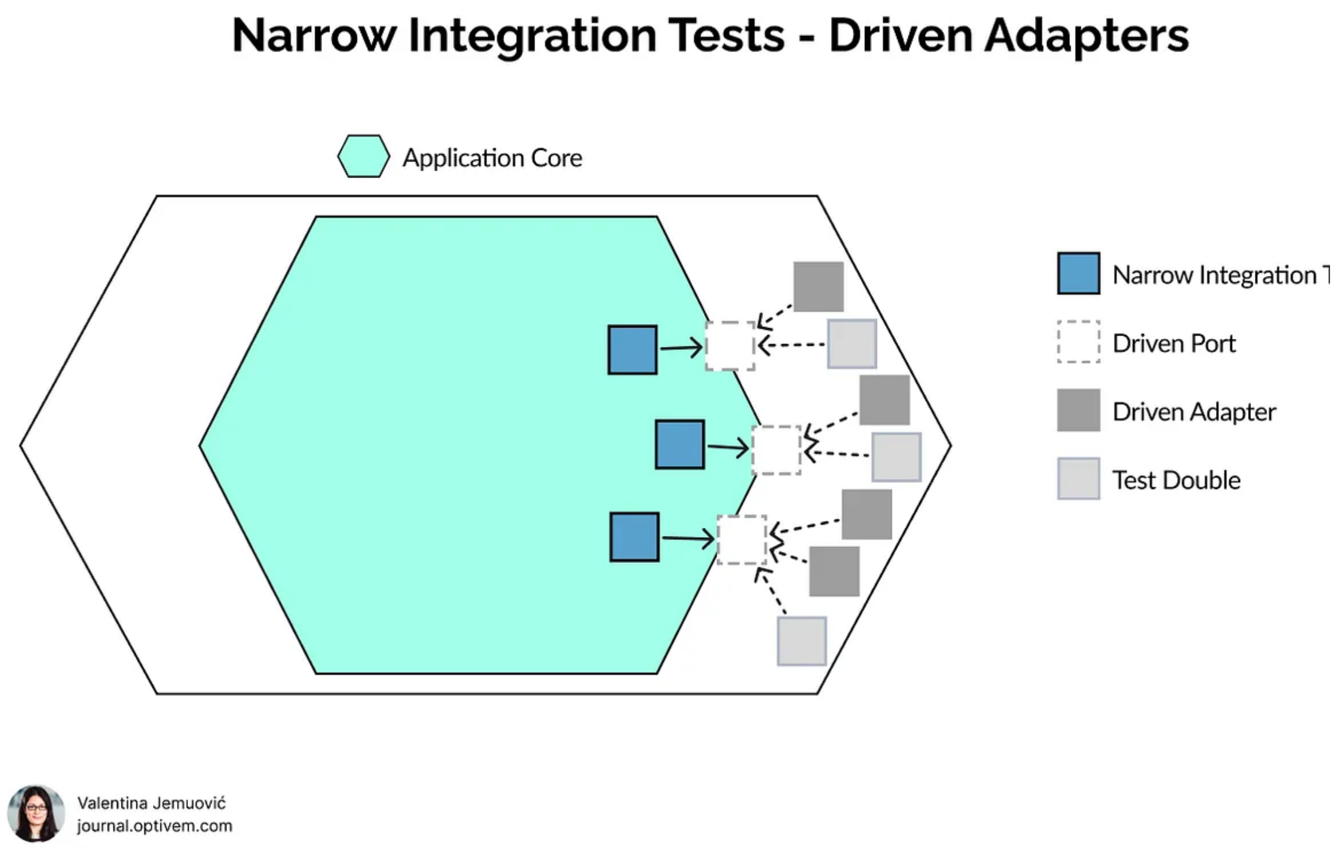




Narrow Integration Tests - Driven Adapters

We can test Infrastructure Logic by testing Driven Adapters (and stubbing out External Systems & other Components). By writing the test against the Driven I we can run the same test against all the Driven Adapters for that Driven Port (e the real & the fake, to assure that the fake matches the real).





📢 My **next Live Q&A Session** about **Unit Tests** is on Wed 30th April 17:00.

As a Paid Member of Optivem Journal, [get your 100% discount ticket]() for Live Q

Recommend Optivem Journal to your readers

TDD | Hexagonal Architecture | Clean Architecture

Recommend

15 Likes · 3 Restacks

Previous

Nex

Discussion about this post

Comments Restacks

Write a comment...

Jelena Cupac 25 Apr

Author

How do you balance between writing tests and development when teams are short on time

LIKE (4) REPLY

3 replies by Valentina Jemuović and others

Vinny Vinny's Substack 28 Apr

Liked by Jelena Cupac

Valen, this is my new favorite tech article ever. Incredible clear explanation.

LIKE (1) REPLY

3 more comments...