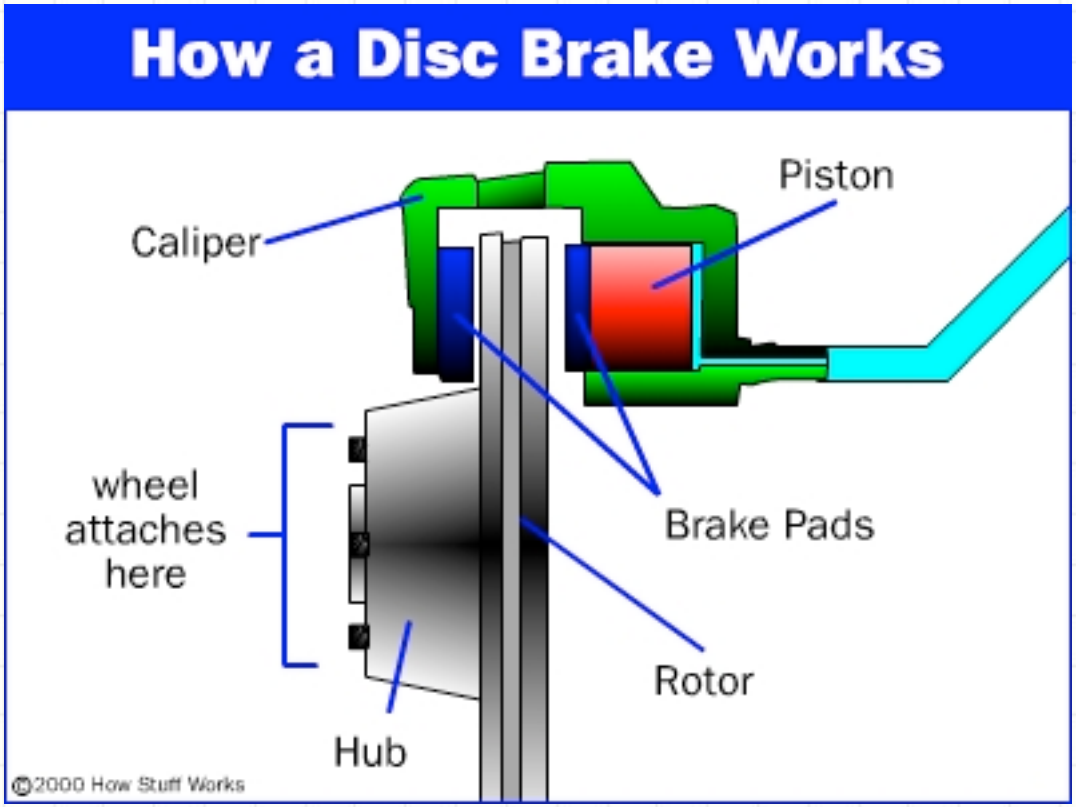
Functional & Integration Testing

G. Molines

2017-2018

Unit-, Functional- & Integration tests

Unit-Test


How a Disc Brake Works
Piston
Caliper
wheel attaches here
Brake Pads
Rotor
Hub
©2000 How Stuff Works

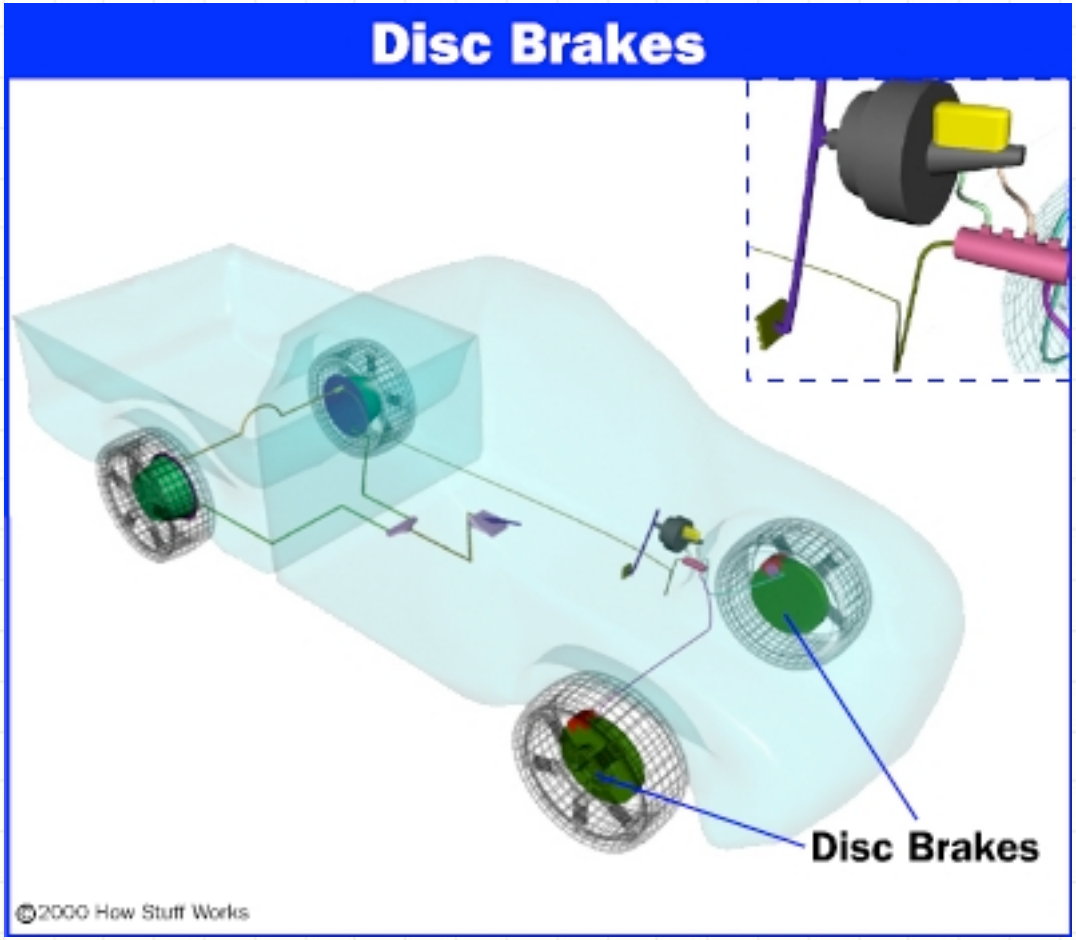
Unit-Test

Example:

CartModifier add(Customer, Item)

KitchenBean process(Order)

Functional Test



Functional Test

Example:

CartWebService
addItemToCustomerCart(String, Item)

Functional - specs

GIVEN

The Cookie Factory server and its Webservices

The customer Bob

WHEN

Bob calls the CartWebService addItemToCart WS with 3 chocolate cookies

THEN

- Bob’s cart is updated (incremented by 3 items)
- TCF inventory is updated (3 cookies removed)
- …

Unit vs Functional

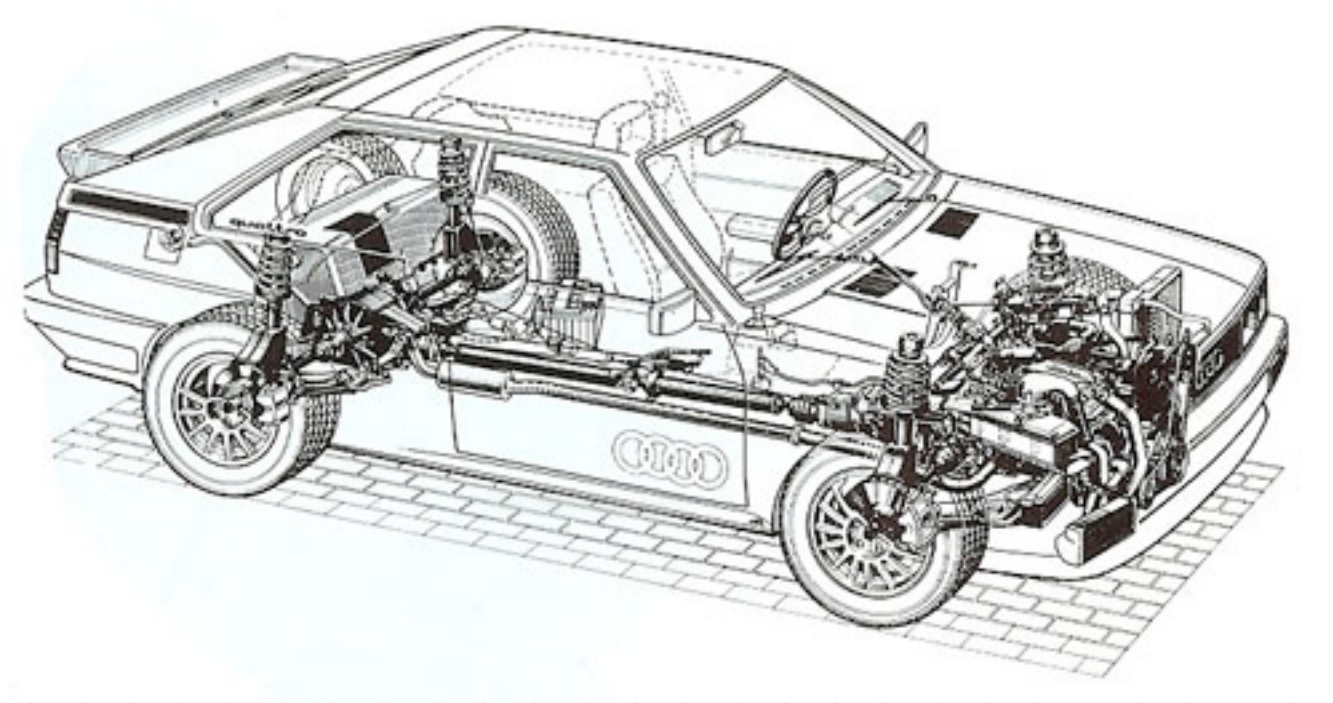
Technology <-> Function

Programmer <-> User

Does it work? <-> Does it implement the function I need?

Small <-> Bigger

Integration Test



Integration Test

Example:

CLI

OrderCookie.execute

Integration Test - specs

GIVEN
- TCF server, connected to “a” (*) bank system
- The CLI client
- The right sample data (Bob, some cookies…)

WHEN
- Bob builds a cookie order and calls OrderCookie.execute

THEN
- TCF inventory is updated (3 cookies removed)
- Bob’s cart is updated (added 3 chocolate cookies)
- Bob’s bank accounted is billed

(*: real or mocked)

Unit vs Integration

http://imgur.com/qSN5SFR

Scenario Testing

Scenario Testing - specs

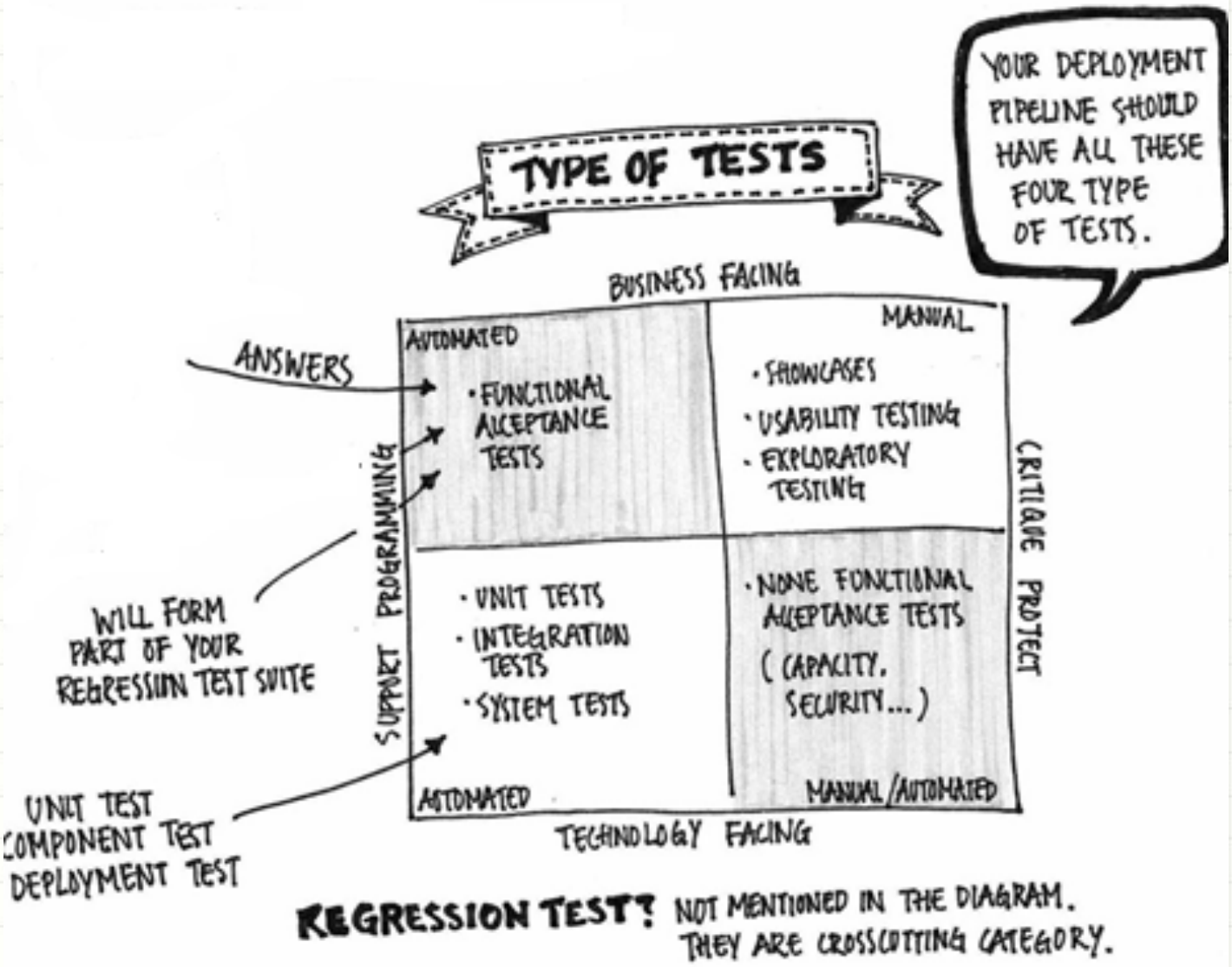
GIVEN

- TCF server, complete
- The CLI client

SCENARIO

- Create a customer Bob through the CustomerCare API
- Add some cookies in the factory
- Bob uses the CLI to order some cookies
- If TCF has enough cookies, fulfill the order and check for appropriate updates
- If TCF is running short on cookies, check for error conditions
- Keep going with more tests

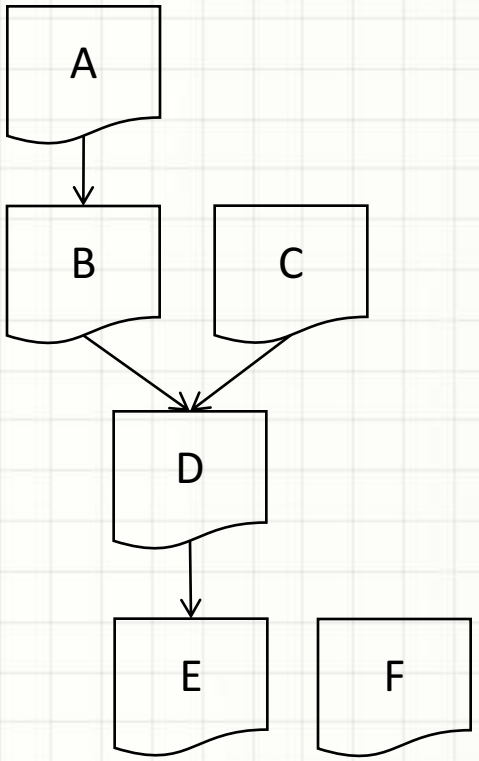
- → What tool would you use?



(From: Nhan Ngo)

Building components

- What does B consume of A?
- Contract
  - API
  - Teams
- Enforced through tests



WRITING TESTS

Things to consider

- Start state
- Sample data
- Asserts
- Clean up

Asserts

```
testSomething {
Customer c = customerRegistry.find(“Bob”);
assertNotNull(c,”didn’t get Bob”);
String orderRef =“FR12345”;
Order o = Util.findOrder(c.getOrders(),
orderRef)
assertNotNull(o, “didn’t get order”);
PaymentService ps = …
assertNotNull(ps “didn’t get PS”);
boolean status = ps.pay(o,c);
assertNotNull(status, “payment didn’t work”);
```

Asserts

- Too many or too few asserts?
- Only assert what you're really testing
- But write tests to cover for other aspects
- Each test should have ONE purpose

More things to consider

- Test maintenance
- Reusability
- → key for functional testing
- → and integration testing
  - Nesting test functions!

SCENARIO
- Create a customer Bob through the CustomerCare API
- Add some cookies in the factory
- Bob uses the CLI to order some cookies
- If TCF has enough cookies, fulfill the order and check for appropriate updates
- If TCF is running short on cookies, check for error condit
- Keep going with more tests

Where to run

- Unit
  - Dev command line / dev. Env
  - Build plan, right after compile
  - Prevents artifactory publication
  - Because
    - it’s fast
    - It’s the contract with dependents

Where to run

- Functional
  - Dev command line / dev. Env
  - Build plan, dedicated
    - Use profiles
  - Prevents artifactory publication – or not…
  - Not so fast anymore
  - May require to be run in-container
    - Arquilian, Cactus → server-side

Where to run

- Integration
  - Dedicated env
    - Use VM if possible
  - Separate build plan
  - Long running
  - Requires prod-like env.

QUESTIONS?