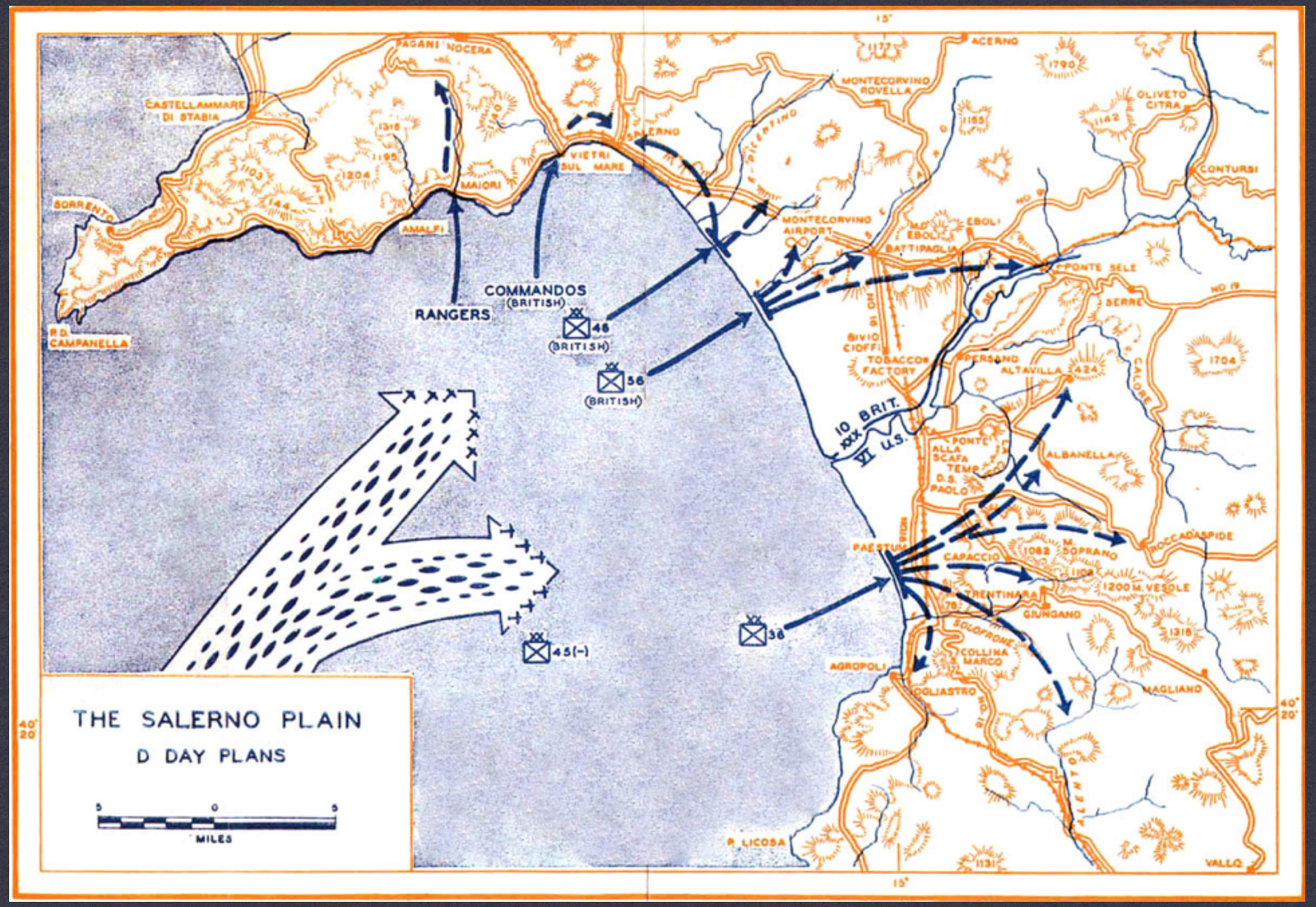
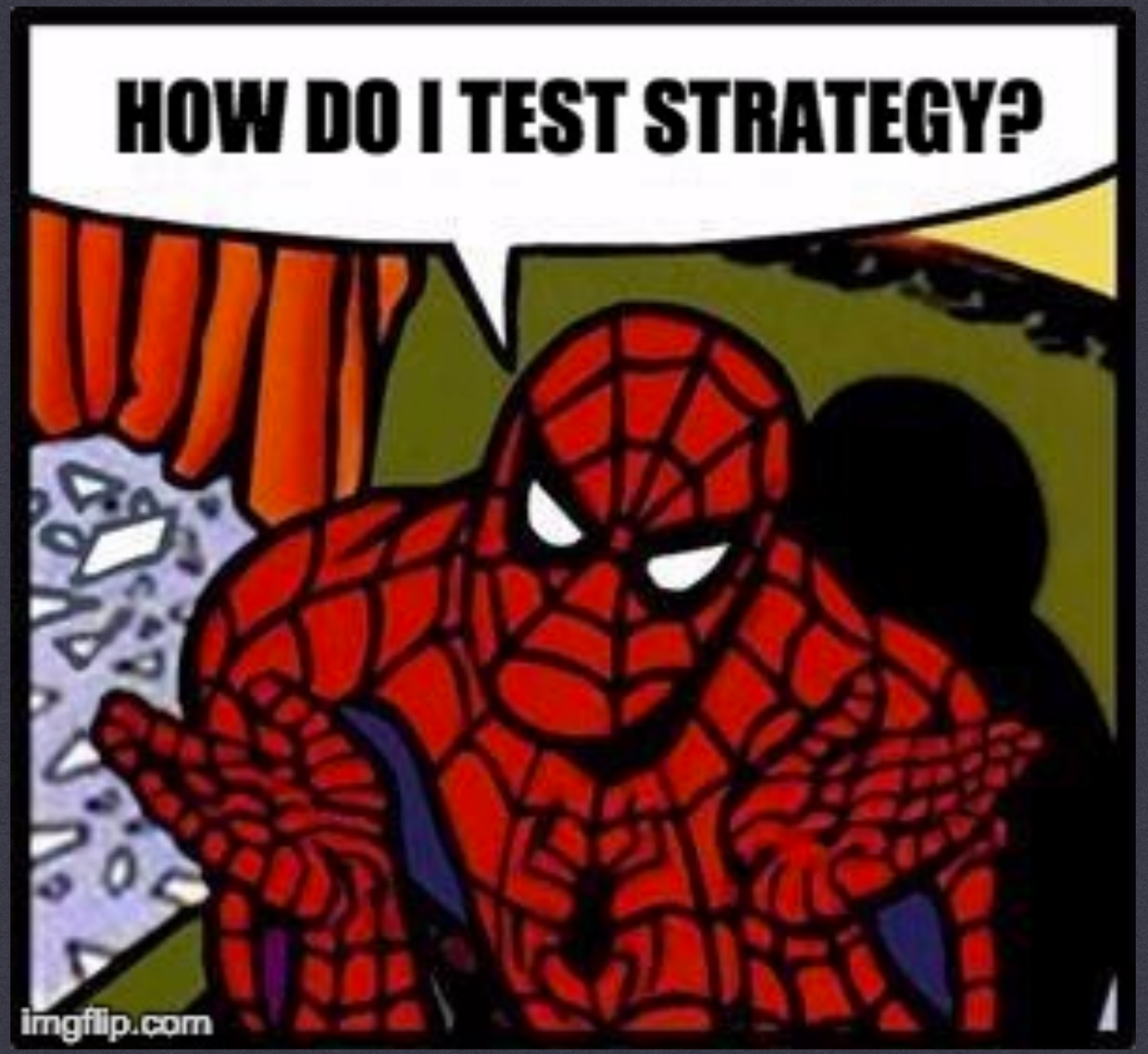
STRATEGIC TESTING

@DmitrySharkov

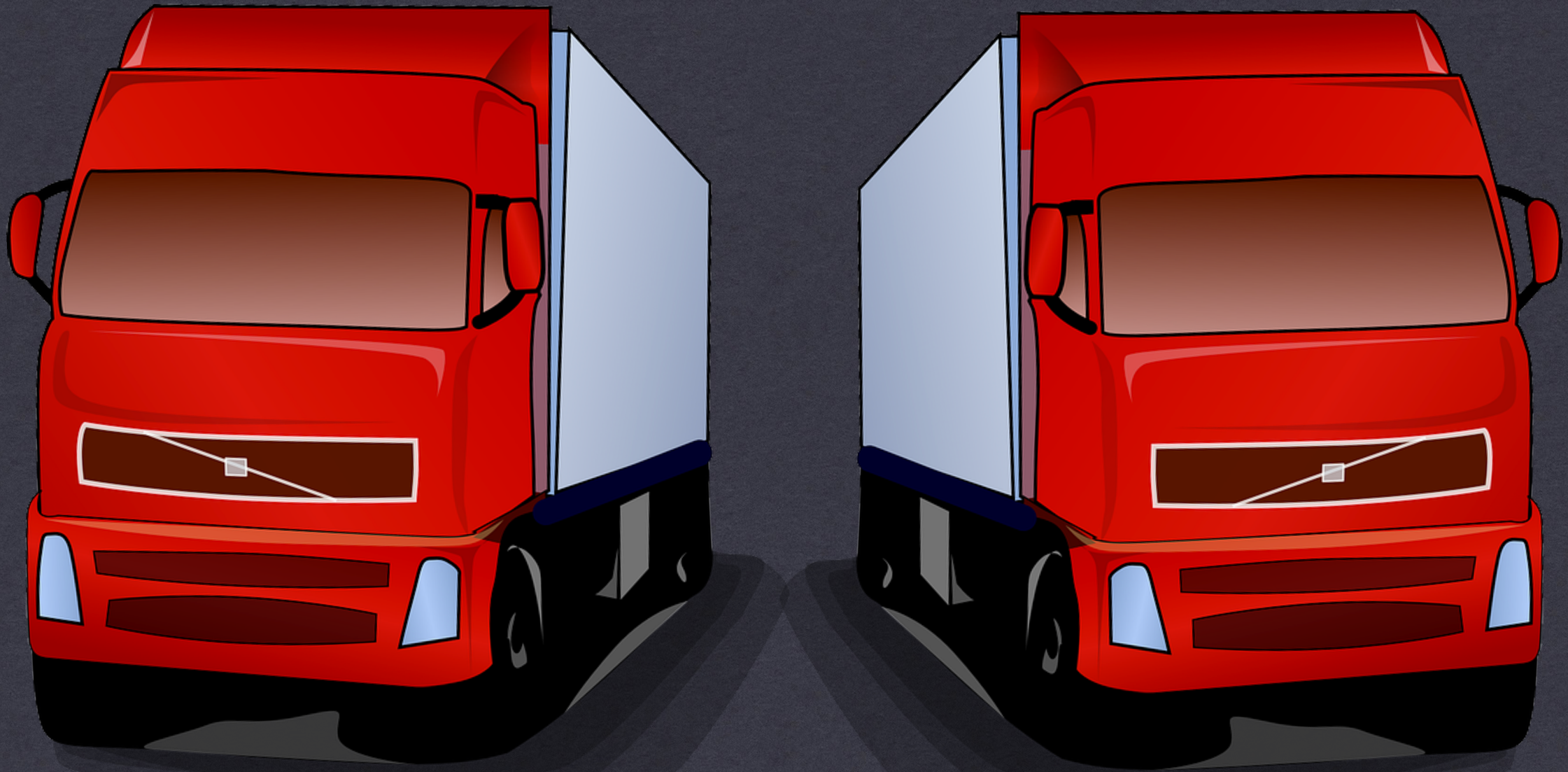
WHAT IS A TEST STRATEGY?

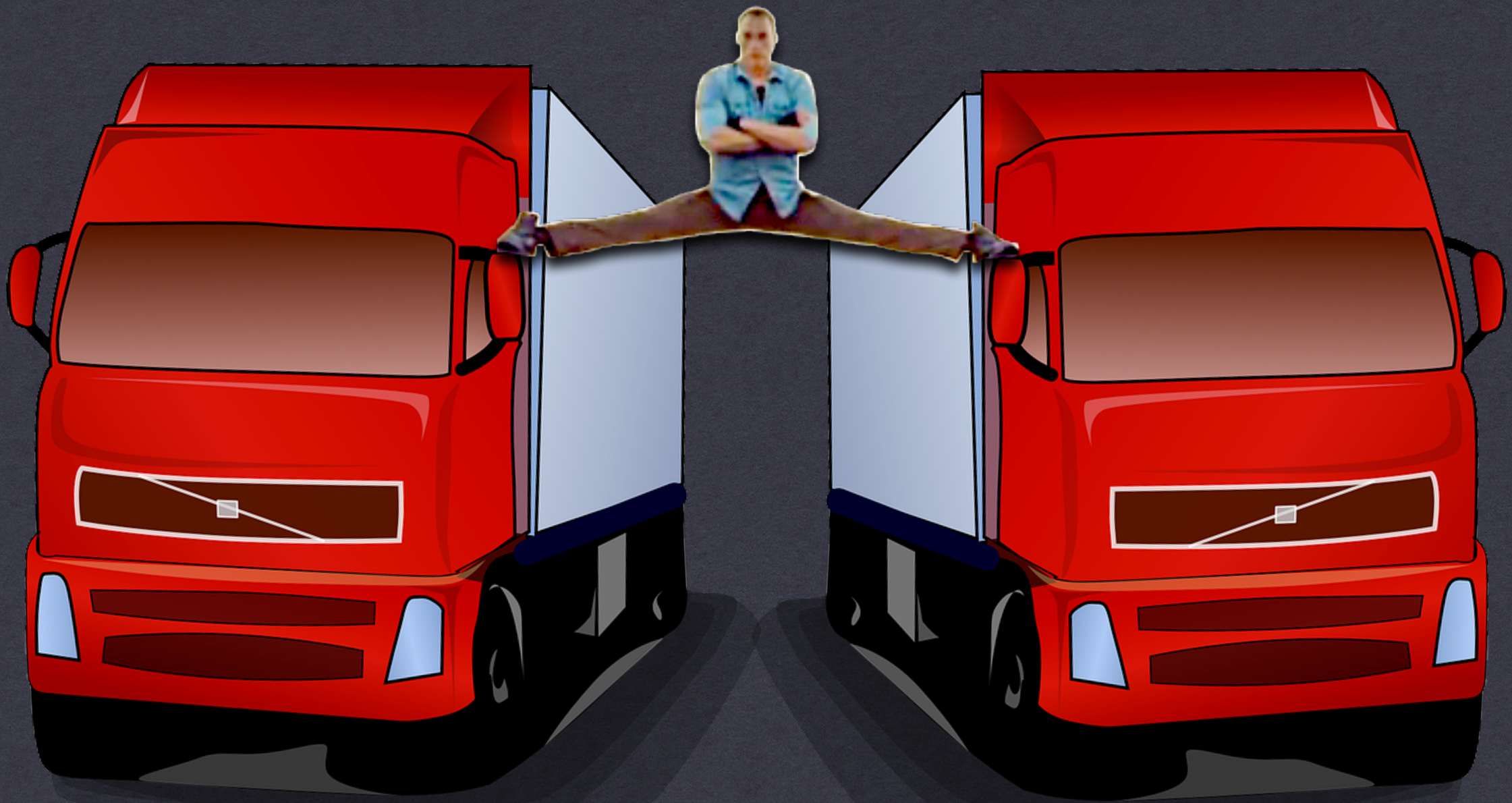



HOW DO I TEST STRATEGY?
imgflip.com

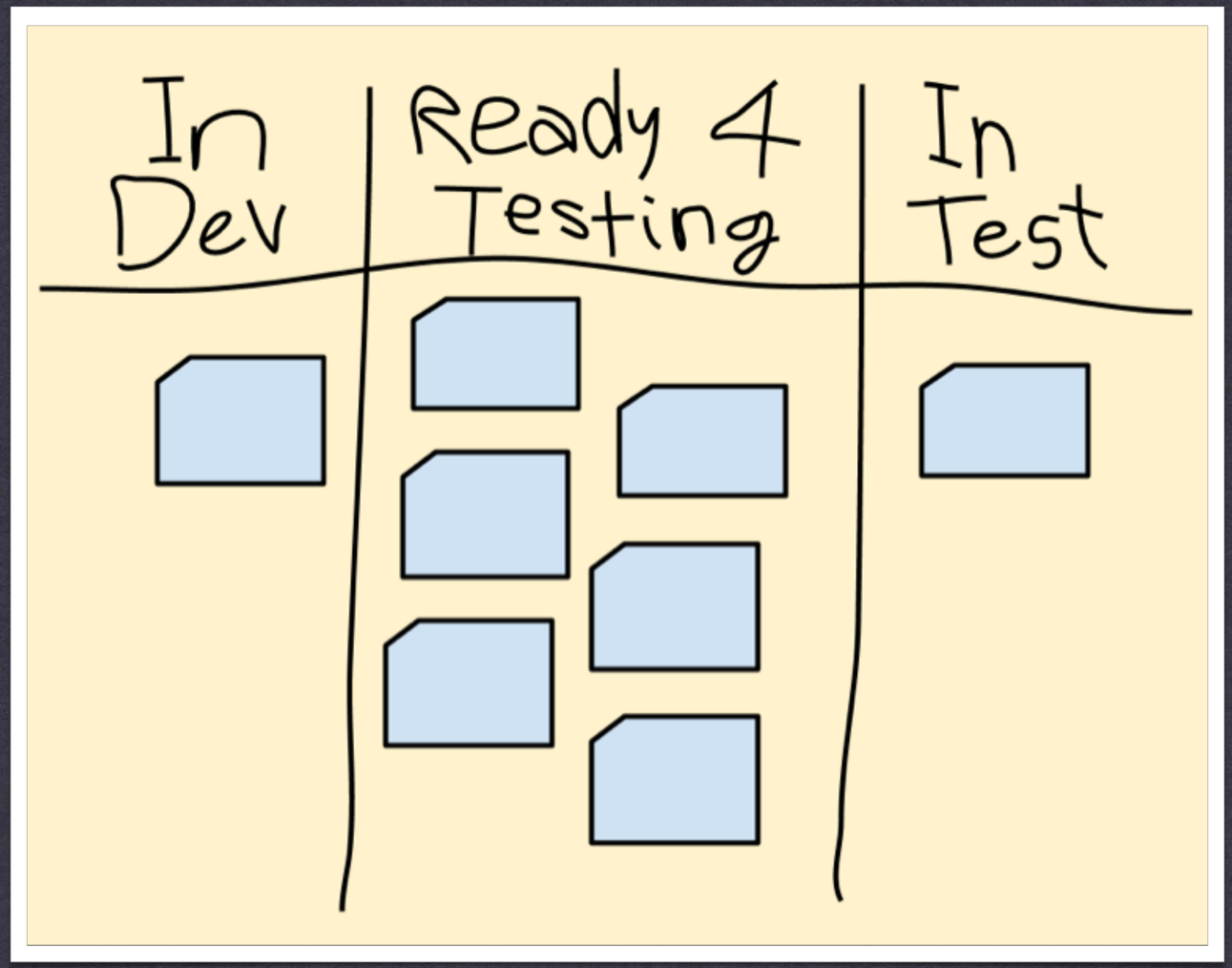

SILVER CAR
852357-BO
domenico.monaco@kiurz.it


95





SYMPTOMS OF POOR TEST STRATEGY



IT’S NOT NECESSARILY LOW QUALITY.

IT’S THE COST OF HIGH QUALITY.

TEST SMARTER, NOT HARDER.

test the right things

at the right time

with the right tools

TESTING TOOLBOX — TYPES OF TESTS

- unit (small)
- integration (medium)
- end-to-end (large automated)
- end-to-end (large manual)
- unscripted/sapient
- nonfunctional ("-ility") tests

TESTING TOOLBOX — TECHNOLOGIES

Test Type	Sample Technologies
UNIT	JUnit, RSpec, NUnit, Spock, Mockito, Jasmine, Mocha
INTEGRATION	same as above; custom
MANUAL E2E	HP Quality Center, Gherkin, unstructured
AUTOMATED E2E	HP Unified Functional Testing, Cucumber, custom
UNSCRIPTED	unstructured
"ILITY" TESTS	Gatling, sqlmap, Selenium Grid, WAVE

GOOD TEST STRATEGY

WE HAVE THE TOOLS. HOW DO WE USE THEM?

PROPERTIES OF A GOOD TEST STRATEGY

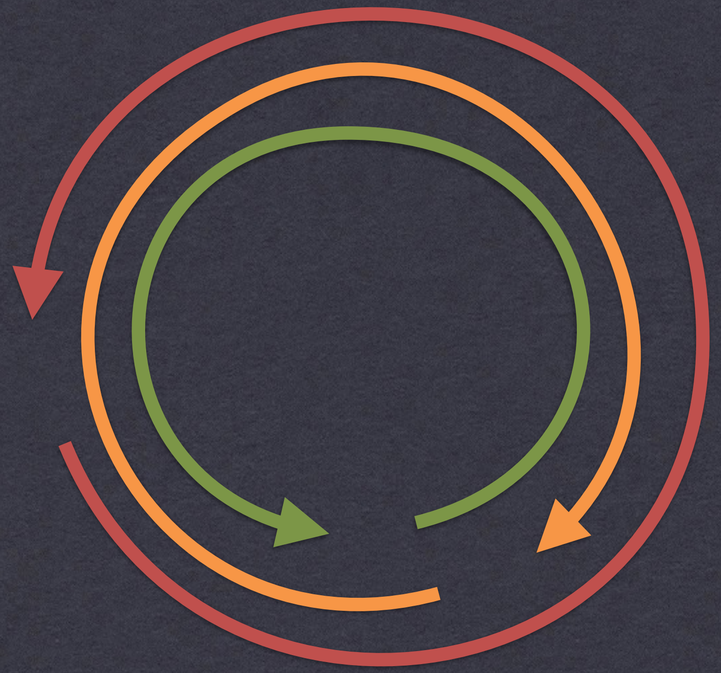
- Minimizes waste.
- Puts burden of testing primarily on devs.
- Sustainable.
- Repeatable.
- Yields tests that maximize speed of feedback, reliability, and focus.

PRINCIPLES OF TEST STRATEGY (VERSION 0.3.4)

- Favor tighter feedback loops.
- Think in terms of unknowns.
- Automate, automate, automate.
- Avoid redundancy.
- Don't be redundant.
- Address nonfunctional requirements early.

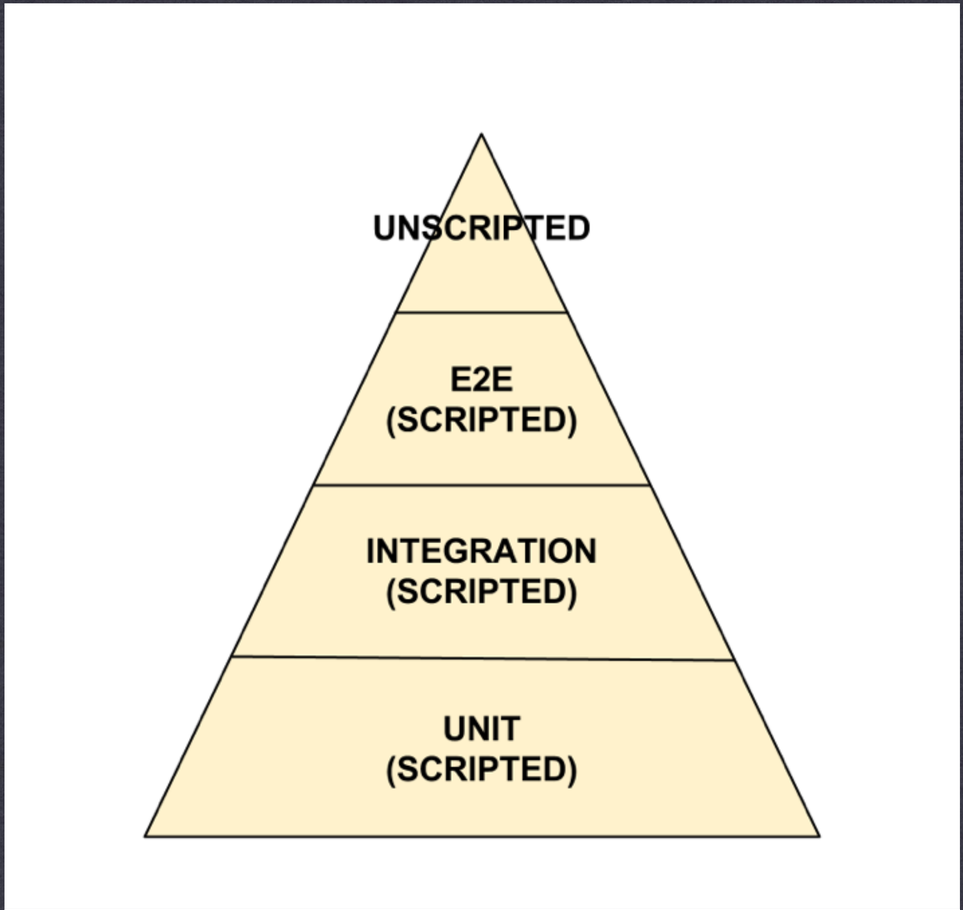
Principle 1

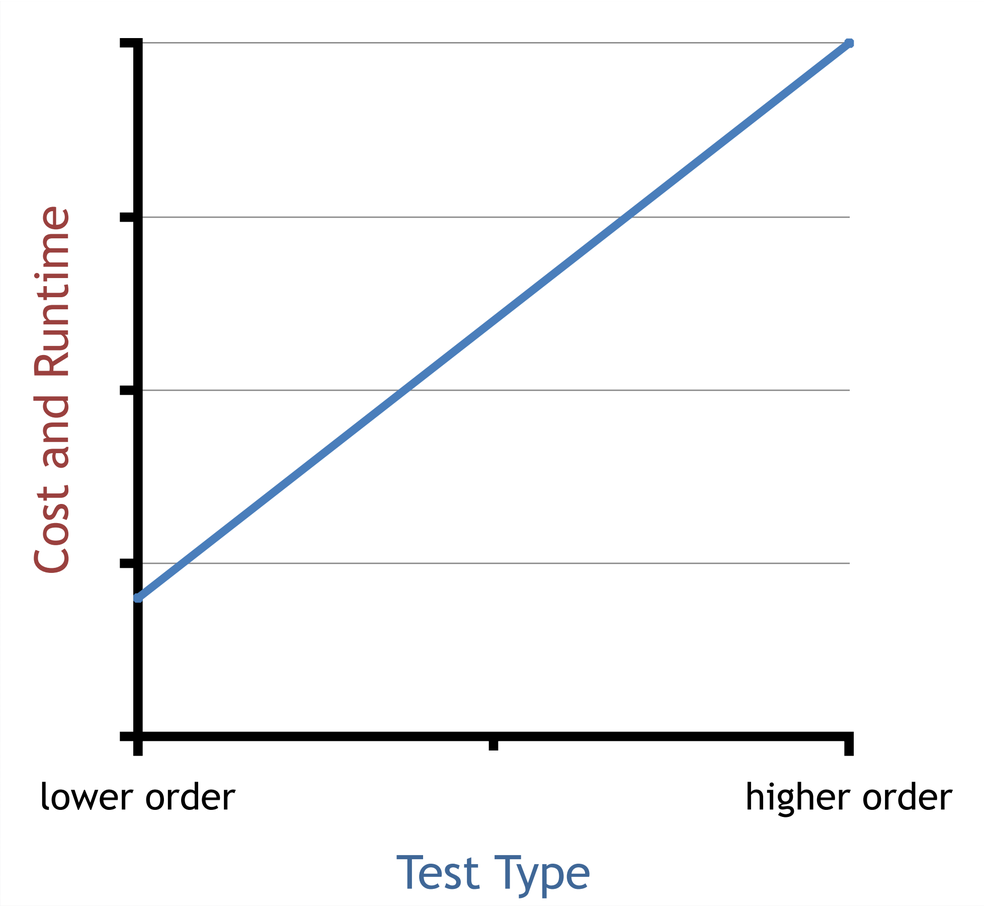
Favor tighter feedback loops.



Principle 1

Favor tighter feedback loops.





Principle 1

Favor tighter feedback loops.

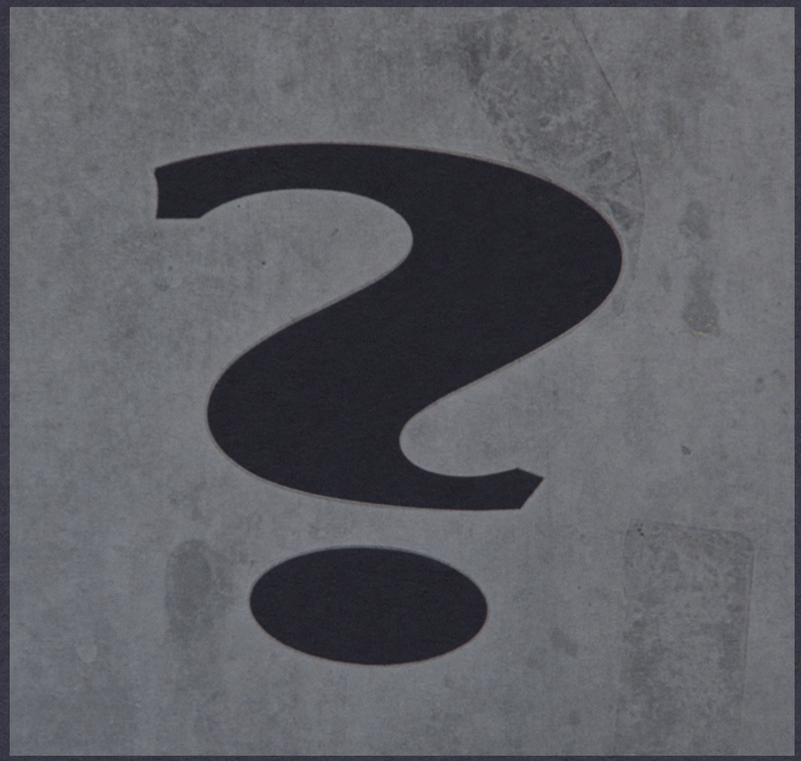
Testing whether a seat belt locking mechanism works…

Principle 2

Think in terms of unknowns.

What questions do you answer with…

- **unit tests?**
- **service-level integration tests?**
- **basic end-to-end automated tests?**
- **testing Safari on iOS?**
- **testing against production data?**

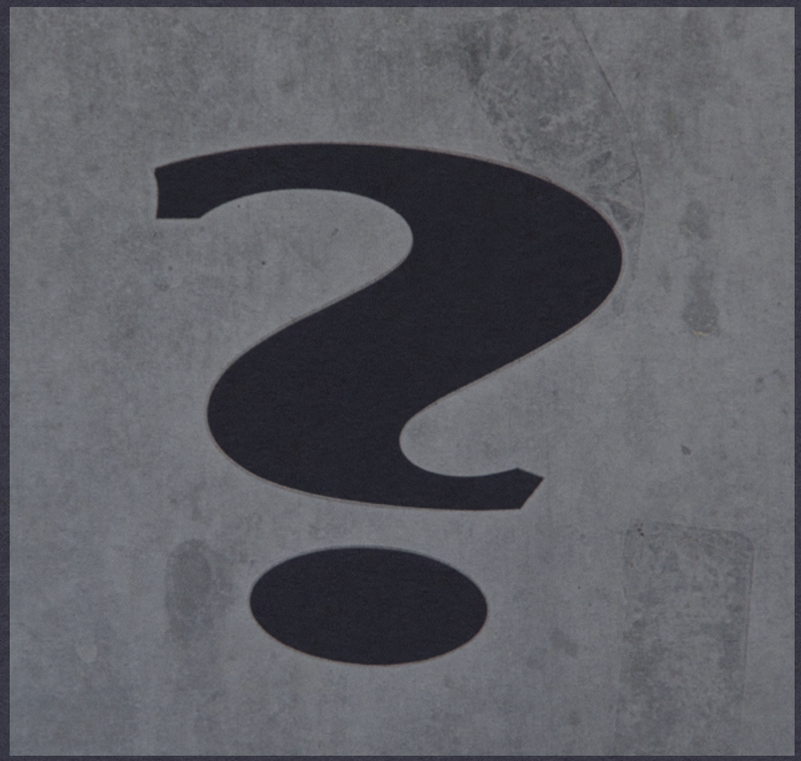


Principle 2

Think in terms of unknowns.

What raises questions?
(What are sources of unknowns?)

- **acceptance criteria — are they met?**
- **architecture — is integration exercised?**
- **environments — how is app affected?**
- **non-functional reqs — are they met?**
- **??? (unknown unknowns)**



Principle 3

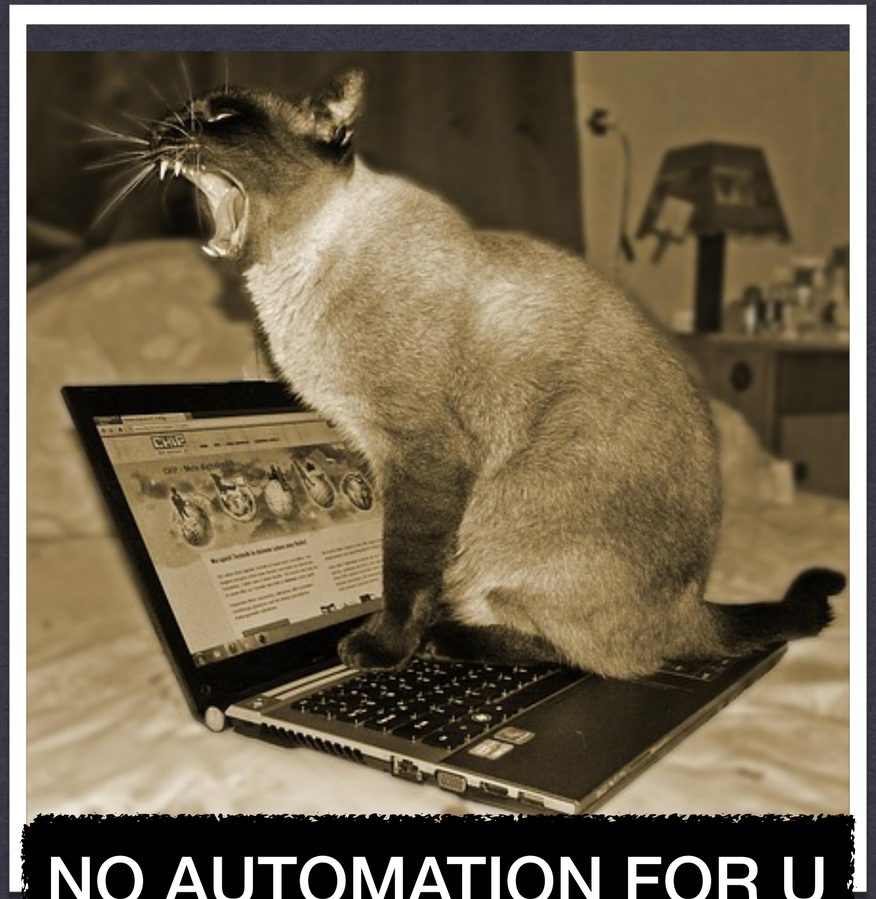
Automate, automate, automate.

Except for…

- Look and feel tests (maybe)
- Graphical consistency tests
- Wildly complex integrations
- Tests with risky side effects
- Stuff too hard to automate

Principle 3

Automate, automate, automate.



Principle 4

Avoid redundancy.

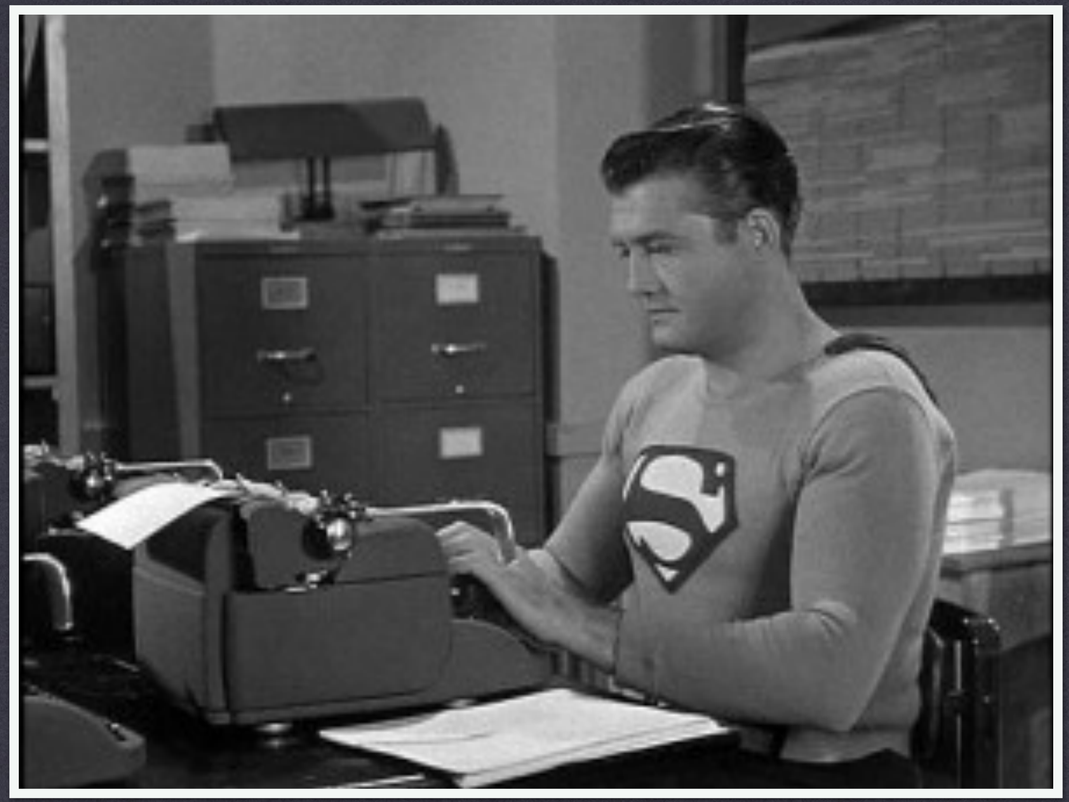
Principle 4

Avoid redundancy.

Track integration points.

Principle 5

Address nonfunctional requirements early.

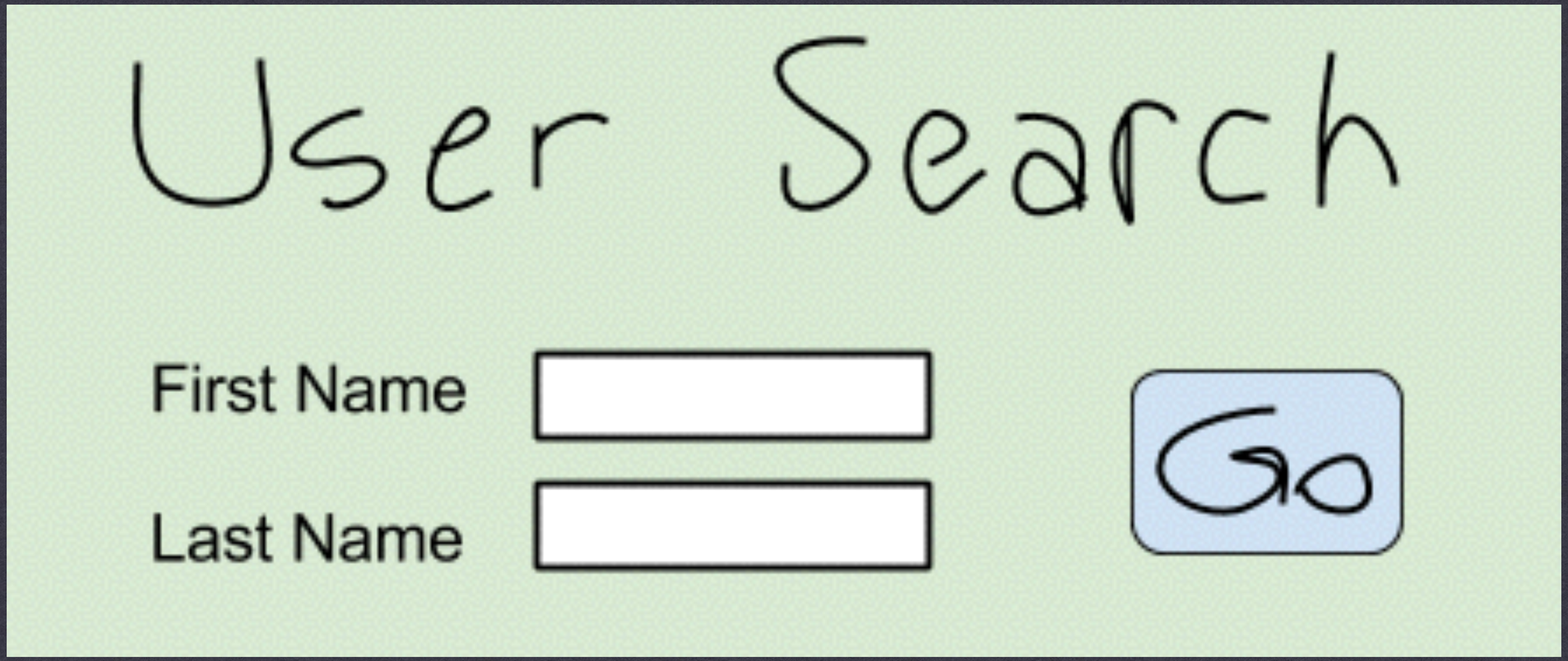


Applying Strategy… Tactics!

GOALS

- Coverage of all acceptance criteria.
- Sufficient coverage of non-functional requirements.
- Fast combined run of tests. (Fast feedback loops.)
- Inexpensive (to write and maintain) test suite.

EXAMPLE: USER SEARCH



USER SEARCH — ACCEPTANCE CRITERIA

Search by Last Name and (optionally) First Name.

Search is a case-insensitive "wildcard" search (uses terms as substrings).

Search is unavailable when offline.

Search button is disabled unless these search criteria are met:

- If searching by Last Name only, at least 3 characters required.
- If searching by both, at least 1 character required for each.

Search ignores non-alpha characters (`[^A-Za-z]`).
This includes punctuation, numbers, and all "special" chars (`á`, `ä`, `ñ` etc.)

If a server-side error occurs, display sad emoji and error message.

…

USER SEARCH — STORY 1

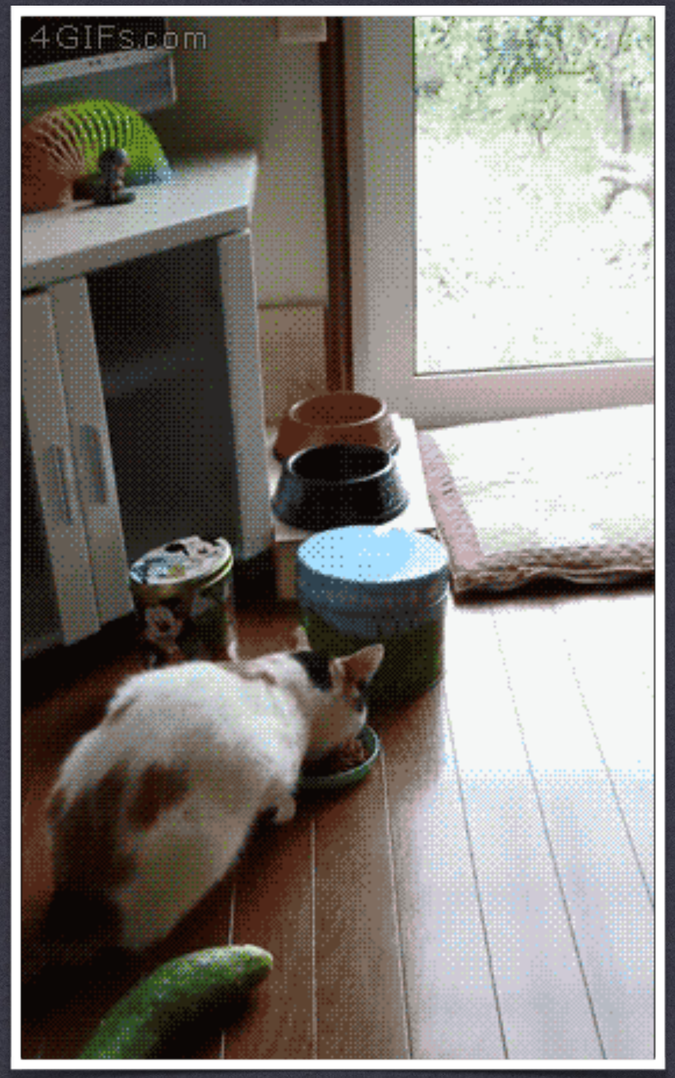
Search by Last Name and (optionally) First Name.
Search is a case-insensitive "wildcard" search (uses terms as substrings).
Search is unavailable when offline.
…

Start with a conversation about unknowns.

STORY 1 — QUESTIONS TO ANSWER

- Is the user able to enter search terms and press the search button?
- If results come back from the server, do they appear correctly?
- Do results return from the server successfully?
- Where do the results appear in the UI?
- Does the search service hit the data store correctly?
- Does the app search through the data correctly…
  - …based on Last Name?
  - …based on Last Name + First Name?
- Does the app return search results quickly enough?

HOW DO WE ANSWER THESE?



HOW DO WE ANSWER THESE?

Unit Tests

Integration Tests

E2E Tests

Performance Test

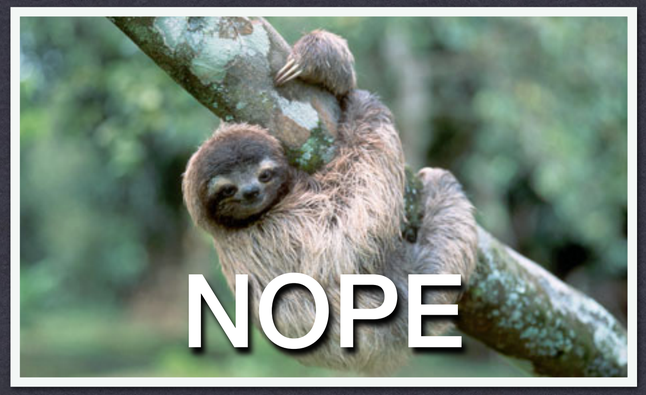
- Is the user able to enter search terms and press the search button?
- If results come back from the server, do they appear correctly?
- Do results return from the server successfully?
- Where do the results appear in the UI?
- Does the search service hit the data store correctly?
- Does the app search through the data correctly…
  - …based on Last Name?
  - ….based on Last Name + First Name?
- Does the app return search results quickly enough?

What can you isolate?

SOME ANSWERS RAISE FURTHER QUESTIONS

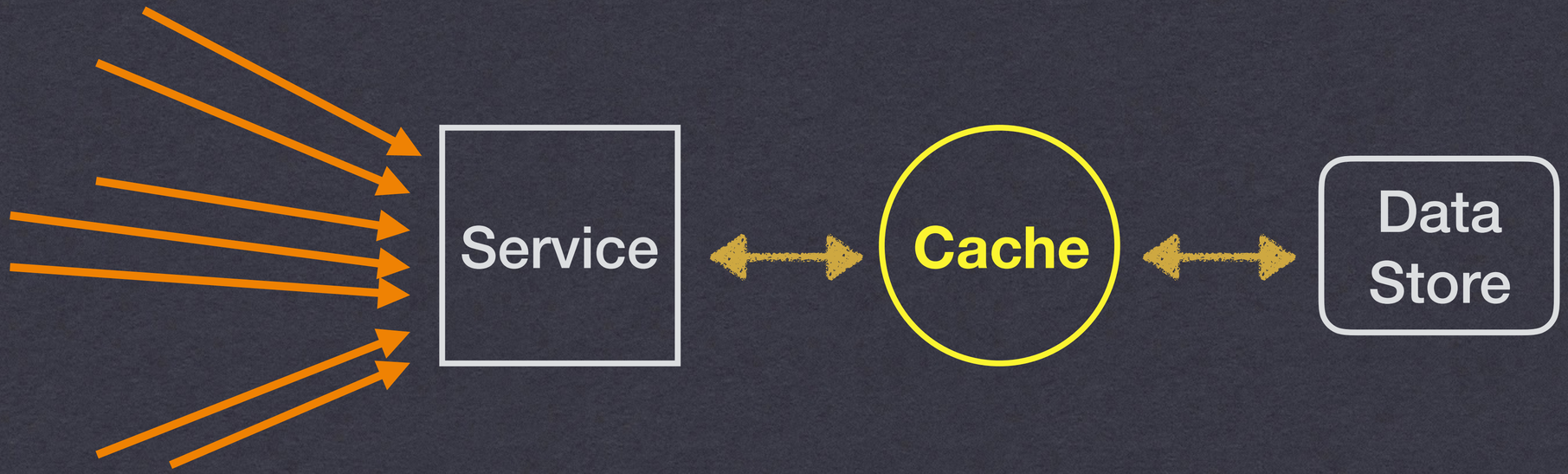
Performance Test

- Does the app return search results quickly enough?



ok then…

SOME ANSWERS RAISE FURTHER QUESTIONS



New Integration Test

- Does the search service leverage the cache properly?

USER SEARCH — NEXT STORY!

Search by Last Name and (optionally) First Name.
Search is a case-insensitive "wildcard" search (uses terms as substrings).
Search is unavailable when offline.
...

Oh! Oh! I know how to test it!

```
Scenario Outline: user search
    Given I go to the user search page
    When I search for users by <first_name> and <last_name>
    Then I get back correct search results

Examples:
    | first_name | last_name |
    | Han        | Solo      |
    | *k*        | Skywalker |
```

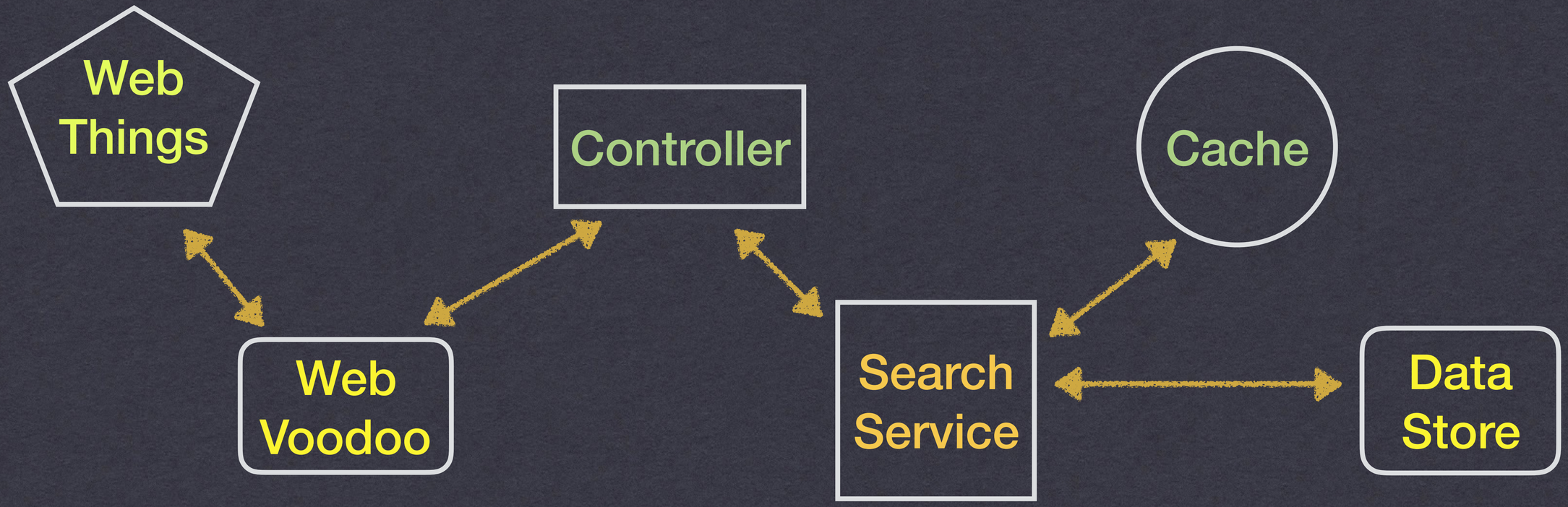
NO!

Search is a case-insensitive "wildcard" search (uses terms as substrings).

What unknowns does this acceptance criterion raise?

WHAT DO WE KNOW?

Consider components and integration points.



Does wildcard search introduce new ones?
(Avoid redundancy.)

STORY 2 — QUESTIONS TO ANSWER

Favor tight feedback loops!

Unit Tests

Integration Tests

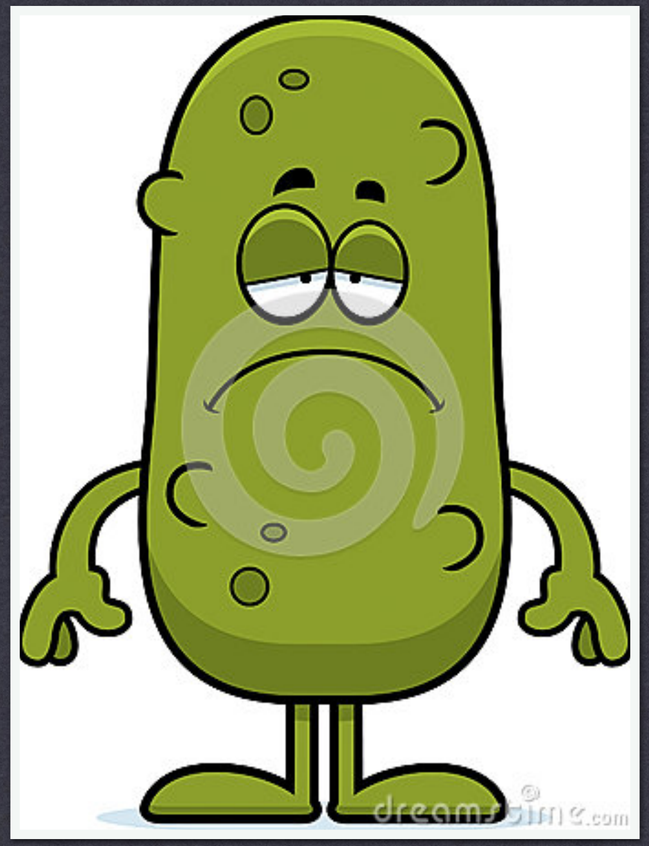
- Does the search service handle wildcards correctly…
- …in the Last Name?
- …in the First Name?
- Does it pass the wildcard to the data layer properly?

USER SEARCH — THIRD STORY!

Search by Last Name and (optionally) First Name.
Search is a case-insensitive "wildcard" search (uses terms as substrings).
Search is unavailable when offline.
...

Does the UI respond correctly if the server becomes unavailable?
Does the UI respond correctly if the server becomes available?

YOU DON'T NEED TO USE E2E TESTS HERE.



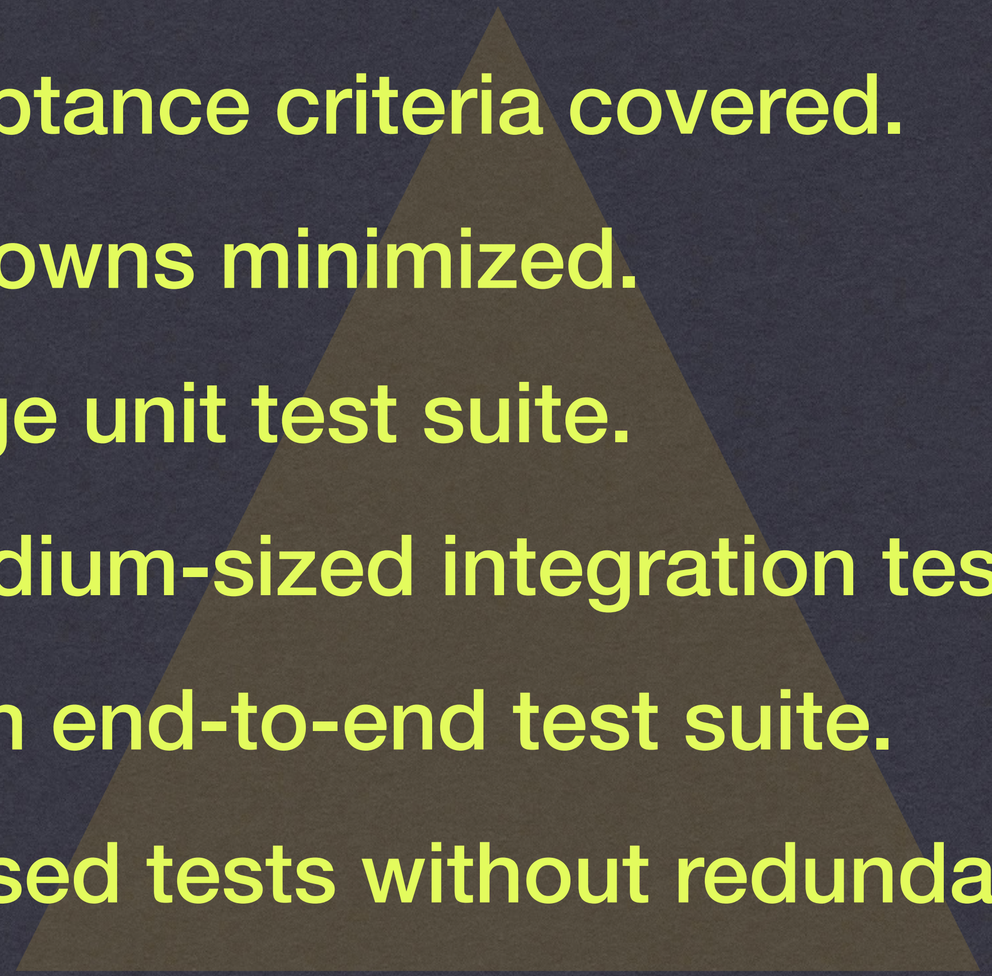
Simplify. Mock server responses. Inspect UI elements.

SOME RULES OF THUMB (TACTICS)

What are you testing?

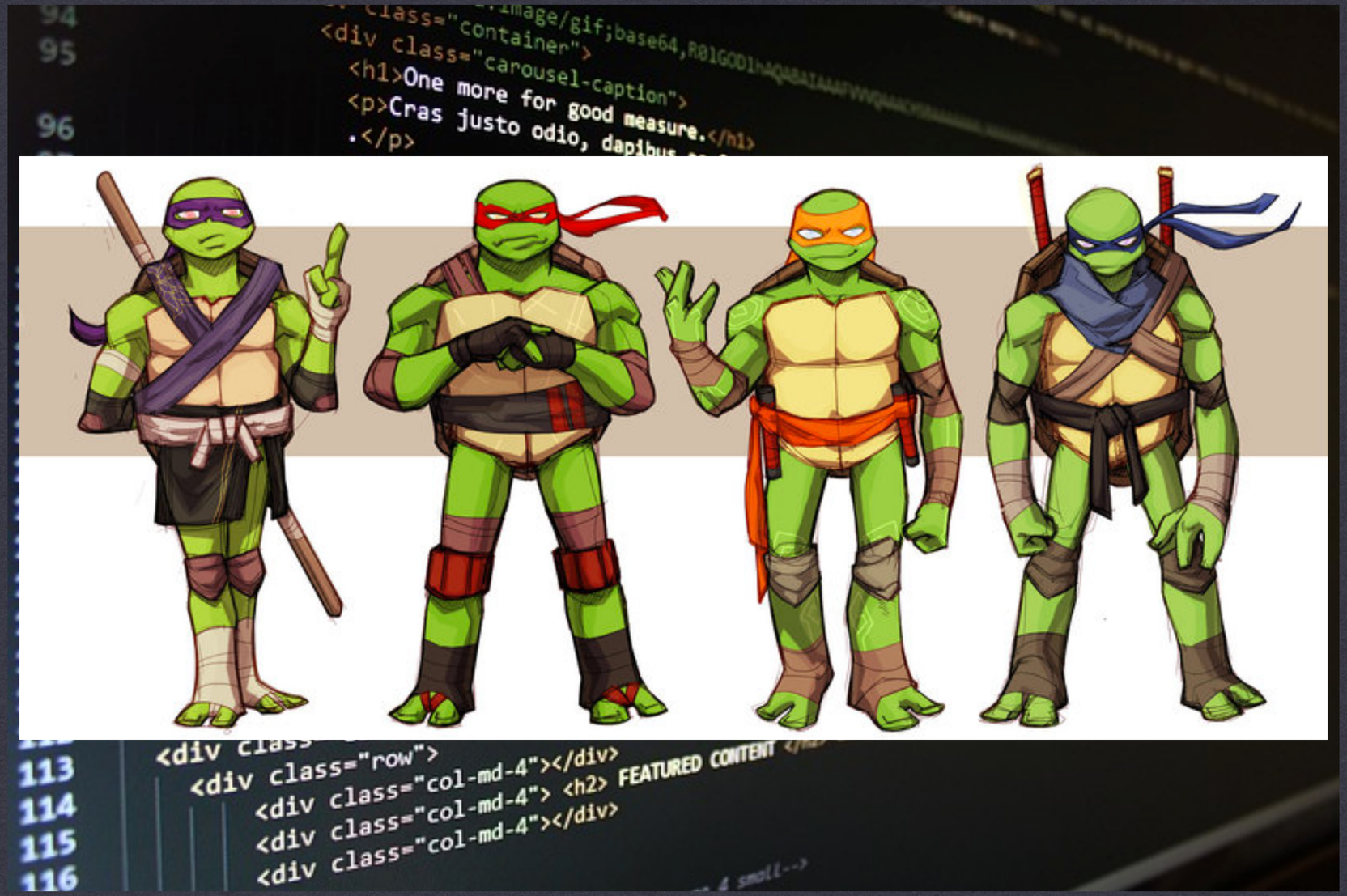
- Logic you can isolate — **unit tests**
  - Validation
  - Display logic
  - Service logic
  - Boundary conditions
- Medium-sized collaboration — **integration tests**
- Logic you cannot isolate — **e2e tests**

WHAT YOU SHOULD END UP WITH…



- Acceptance criteria covered.
- Unknowns minimized.
- A large unit test suite.
- A medium-sized integration test suite.
- A lean end-to-end test suite.
- Focused tests without redundancies.

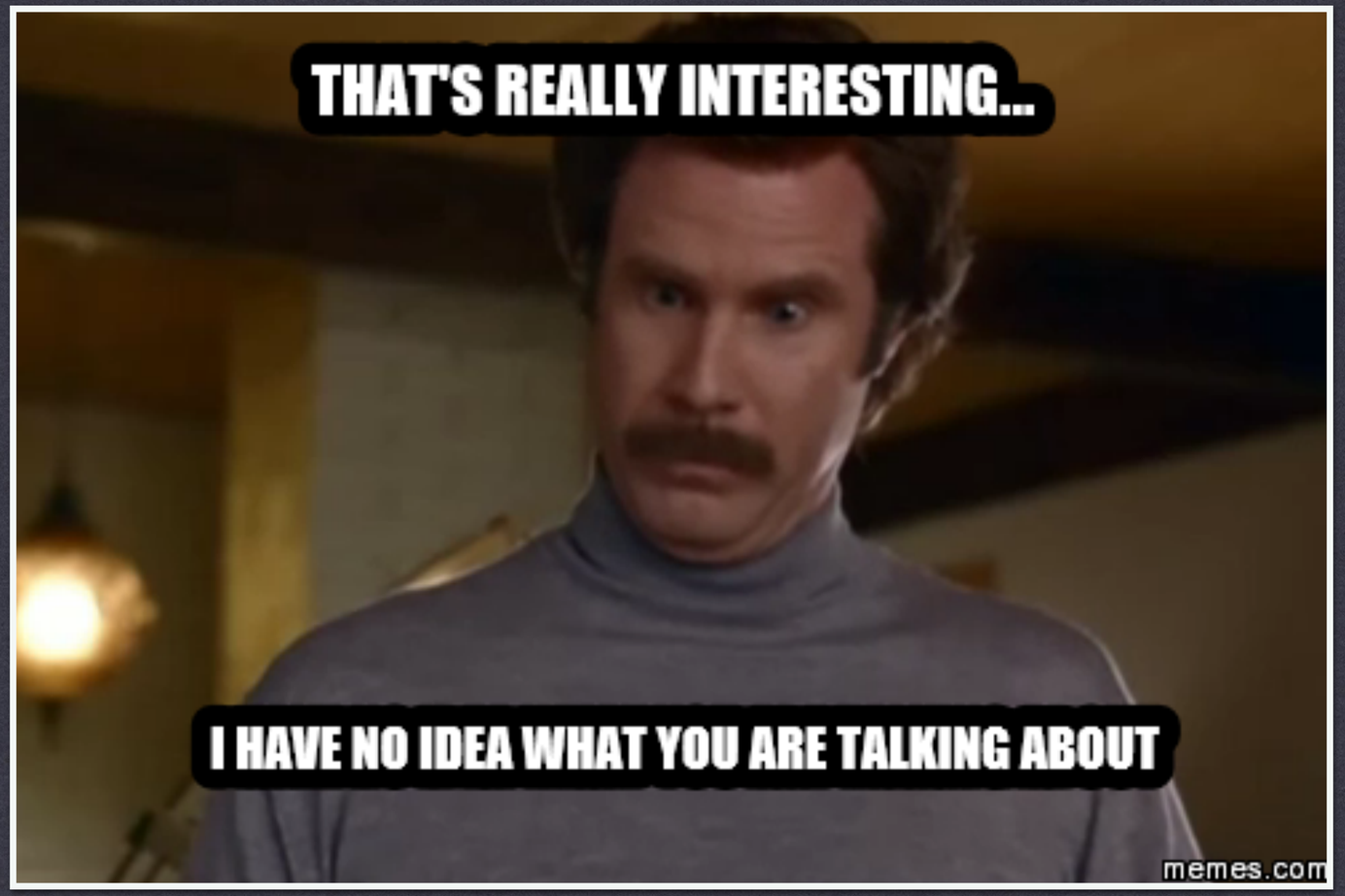
WHAT IT TAKES



COMMON OBSTACLES

Lack of trust.

COMMON OBSTACLES

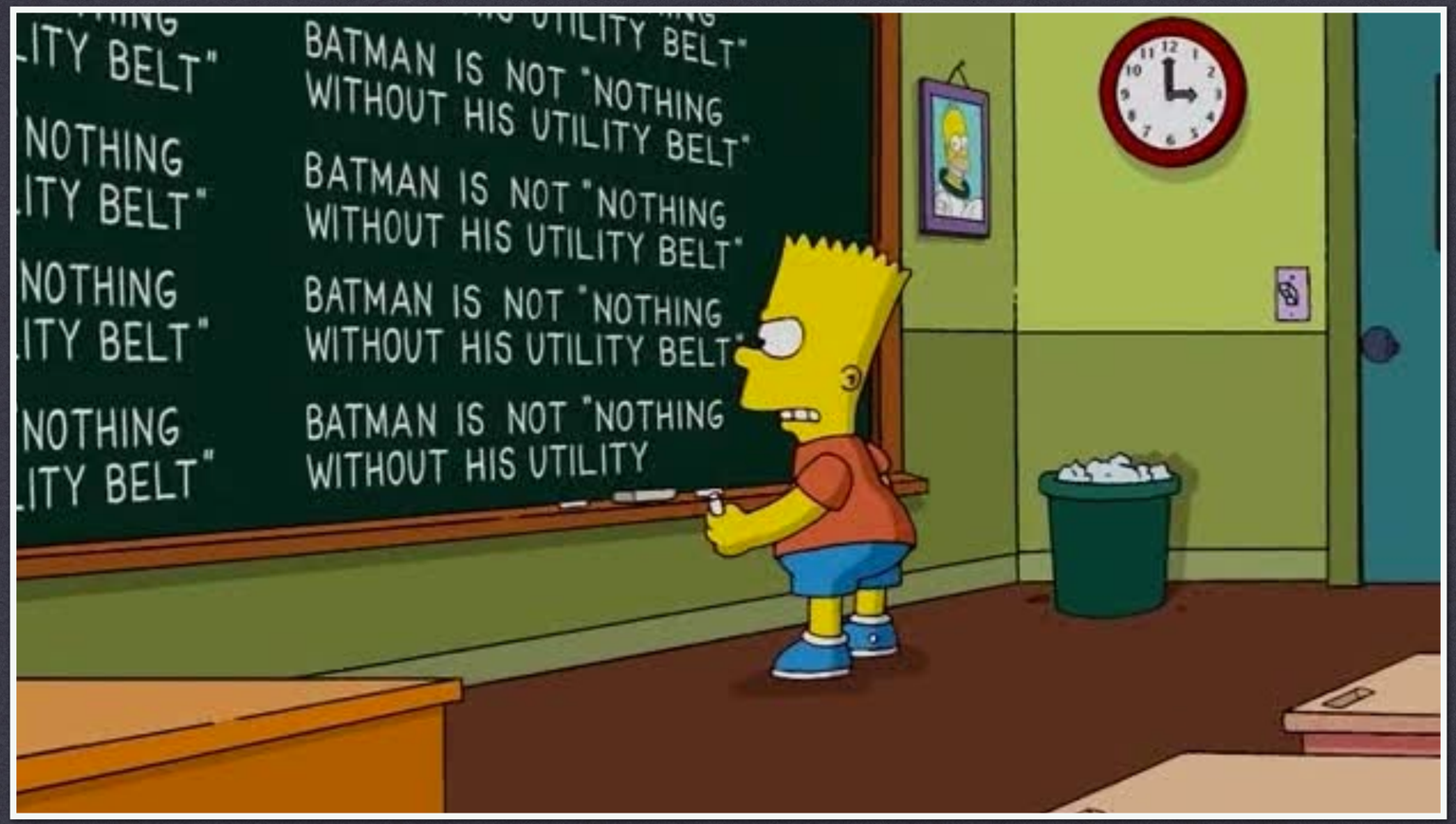


Lack of common understanding.

COMMON OBSTACLES

Working in silos.

COMMON OBSTACLES



Lack of appropriate tools.

BACK AT WORK…

- Make a testing "pyramid" of current state.
- Put testers and implementers together
- Look for redundancy across test suites
- Move tests to as low a level as possible
- Look for signs that tests are missing
- Encourage knowledge transfer
- Start improving and never stop

Thank you!

@DmitrySharkov