Testing: Cable and Chain

Questions To Answer

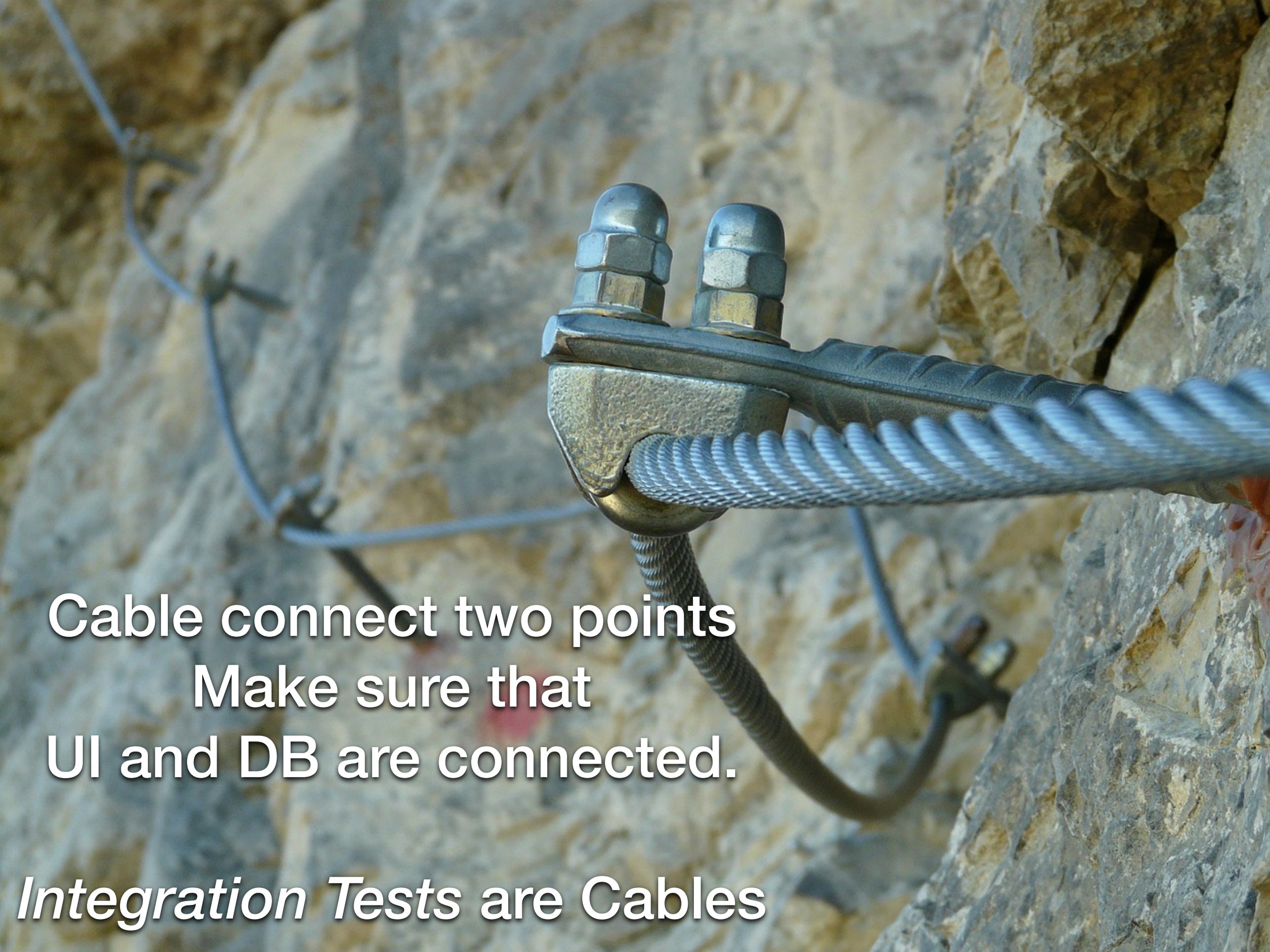
- Are tests helping or hurting your ability to code?
- Why progress slows with complexity?

About me

- Built and Maintained 100K membership site for ~8 year
- Built and Maintained a highly rated Scrum Training Application for ~12 years
- Elixir: CC Processing, Medical Research, Energy Grid
- Unit testing TDD since 1999 — sUnit !
  - Learned from Kent Beck and Bob Martin

Types of Tests

- Technical
  - ***Unit - Links***
  - Integration - Cables
    - Performance
    - Security
- User
  - Acceptance Tests - Checks Behavior
    - Is it done (Unit or Integration)



Cable connect two points
Make sure that
UI and DB are connected.

Integration Tests are Cables

Integration Tests

- Test that things connect correctly
- Not that they perform business tasks

cables()
:ok

- Keep them Minimal
- As few as possible to prove that things connect

Photo by Samuel Zeller on UnSplash

cables()
{:error, “update failed”}

- Hard to change
- Slow you down
- False Positive Coverage reporting

Photograph by Alexandre Baudoin on UnSplash

Incidental Coverage

- Need code example …

A chain is made up of links.

Each Link does a separate job.
If each link is strong,
the chain will be strong.

Unit Tests are Chain Links

Unit Tests

- Assert every possible thing that the function can do
- Every branch and behavior
- Link must be Strong and Reliable

Types of Tests

- Technical
  - Unit - Links - Checks Logic
  - Integration - Cables - Checks Connections
  - Performance
- User
  - Acceptance Tests - Checks Behavior
    - Is it done (Could be Unit or Integration)

Testing approach

1. Test every link throughly - Unit
2. Test that every link can connect - Integration

What is a *Unit* ?

Not a unit test IF

- It can't run in parallel with other unit tests
- It touches or needs …
- Database
- Network
- File system
- Environment - like config files

- ***A Set of Unit Testing Rules*** By Michael Feathers 2005

Things in Environment

- System Time
- Global Application.get_env
- Erlang Scheduler, so no multi-processes tests

- ***A Set of Unit Testing Rules*** By Michael Feathers 2005

Code Under Test

Should have no ***Side Effects***

Side-effects

- A function has a side effect if
  - it modifies a mutable data structure or variable
  - uses IO
  - throws an exception

https://dzone.com/articles/side-effects-1

Functional Programming

- … functions should not have side effects
- A function with side effects
  - is called a *Procedure*
  - can be unpredictable depending on the state of the system
  - *is hard to test*

https://dzone.com/articles/side-effects-1

Functions with no Side-Effects are called

Pure Functions

They are good !

no observable side effects

Pure Functions are easy to test

Tests for Pure Functions are

Unit Tests

But what if the function needs to talk to something ?

Higher-Order Functions

To the Rescue !

- Functions that take functions as params, or return functions

```
inc_function  =  fn(n) ->   n + 1   end

Enum.map( [ 1, 2, 3 ] ,  inc_function )
```

Test with Injection

- Needs Code example …

Refactor:
“Move Side-Effect Up”

- Needs Code Example …

Are your Tests … Large?

hard to write?

Using the DB?

The Environment?

You might be Bound in Cables

How do I Build Small tests ?

Try,
Test Driven Development

- From XP - ExtremeProgramming created in ~1997
- Inside out - Detroit style
- Outside in - London style
- Red - Green - Refactor, 5-10 mins & commit

Test Doubles

Test Double (think stunt double) is a generic term

- **Dummies:** passed around but never used. Fill parameter lists.
- **Fakes:** working implementations, like InMemoryTestDatabase.
- **Stubs:** canned answers to calls.
- **Spies:** Stubs that also record information. like An email service Stub that counts messages sent.
- **Mocks** are pre-programmed with expectations which form a specification of the calls they are expected to receive. They can throw an exception if they receive a call they don't expect and are checked during verification to ensure they got all the calls they were expecting.

- **TestDouble**, Bliki, by Martin Fowler, 2006

Independent Tests

- Easy to change, only touches Collaborators
- Rule #1: Only TestDouble for Collaborators

Local Stubs

Aka, “Mocks As Locals”

Although we have used the application configuration for solving the external API issue,

sometimes it is easier to just pass the dependency as argument. Imagine this example in Elixir where some function may perform heavy work which you want to isolate in tests

— José Valim

http://blog.plataformatec.com.br/2015/10/mocks-and-explicit-contracts/

Prefer Local Stubs

passing the dependency as argument is much simpler and should be preferred over relying on configuration files and Application.get_env/3

— José Valim

http://blog.plataformatec.com.br/2015/10/mocks-and-explicit-contracts/

Test Reflector

- See README.md
- Needs Fuller Code Examples …
-

```
test "using the Reflector to remove slow or side-effect dependency" do

  stubbed_data = "Something just for the test"

  MyRepoReflector.stash_some_data_i_need({:ok, stubbed_data})

  # When

  result = TargetCode.something(234, MyRepoReflector)

  assert {:ok, "Something just for the test234"} == result

  assert_receive :some_data_i_need

end
```