Allen Rohner

Breaking The Bank With Contract Testing

2024-05-09

nu bank

KR

Griffin

The bank you can build on.

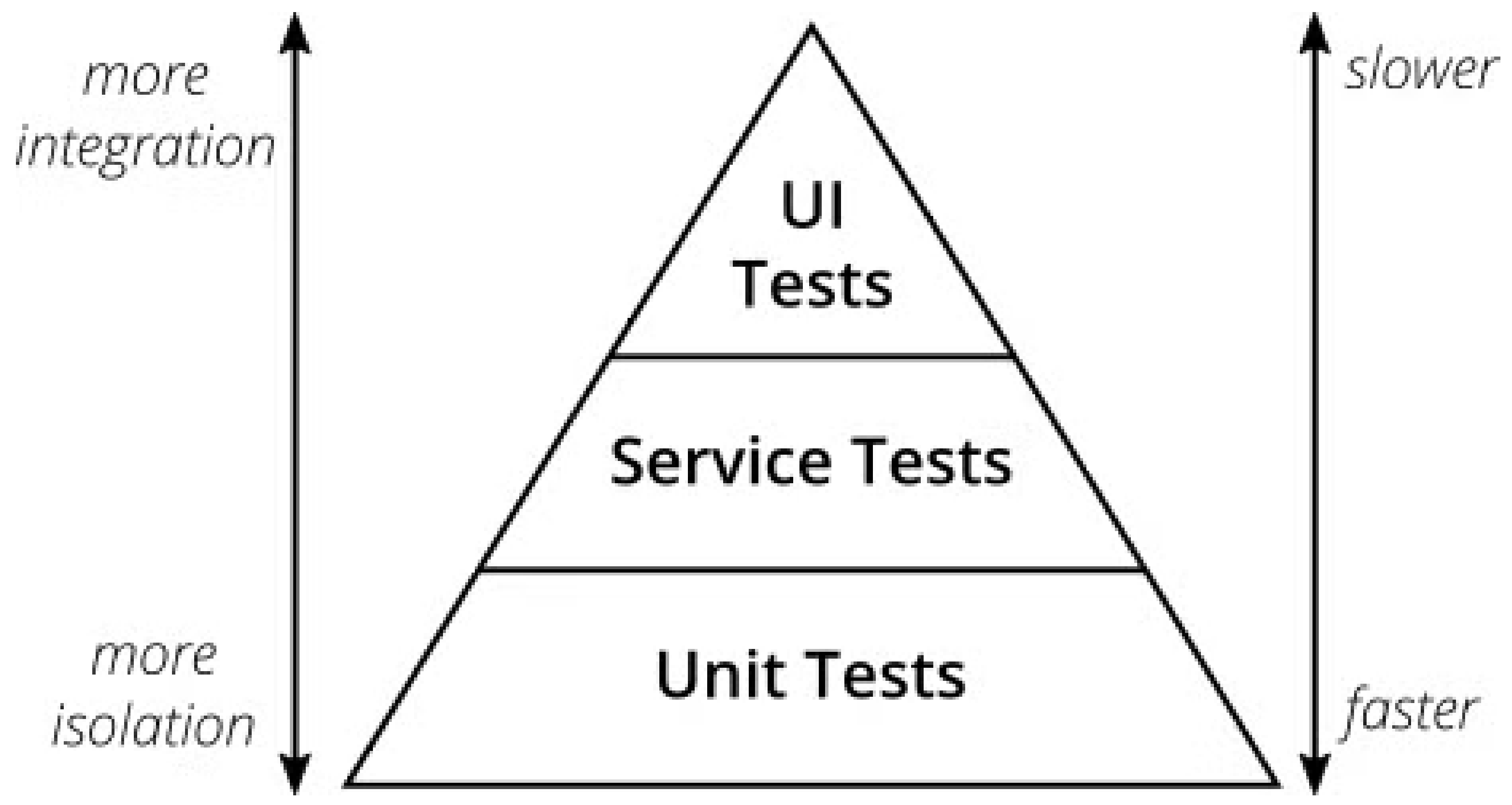
About Griffin

1. Fully authorised UK Bank
2. B2B bank for fintechs (regulated and not)
3. API-first
4. We built our own core banking system!

Test type	Level
Unit	function level
Integration	system level

Testing

- Slow
- Flakey



	pure	impure
function	unit	🤮
system	???	integration

You have an organization problem!

Organ (n)

a part ... that performs a particular job

from Latin organum

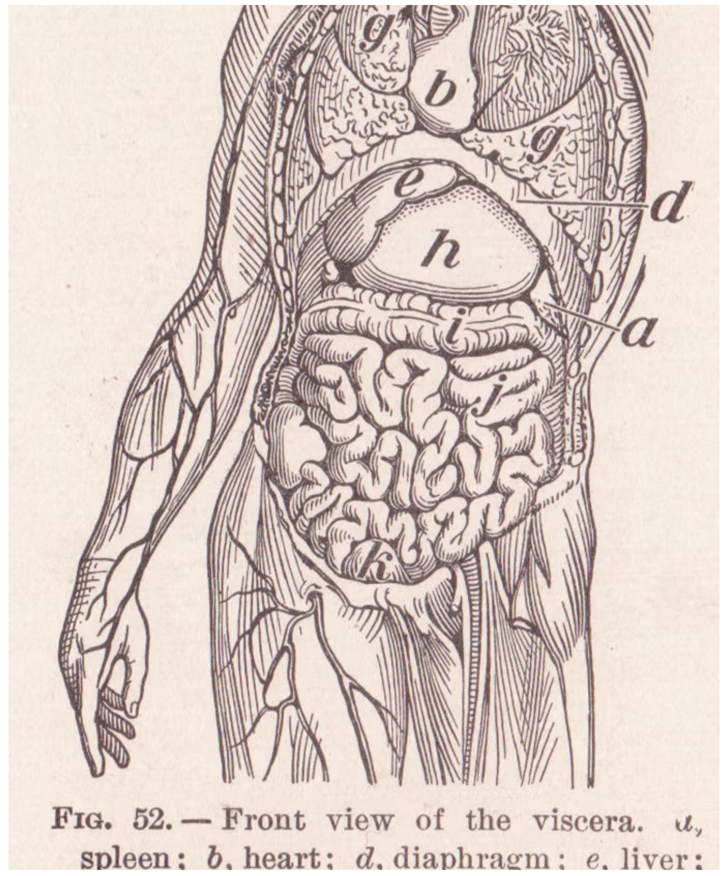
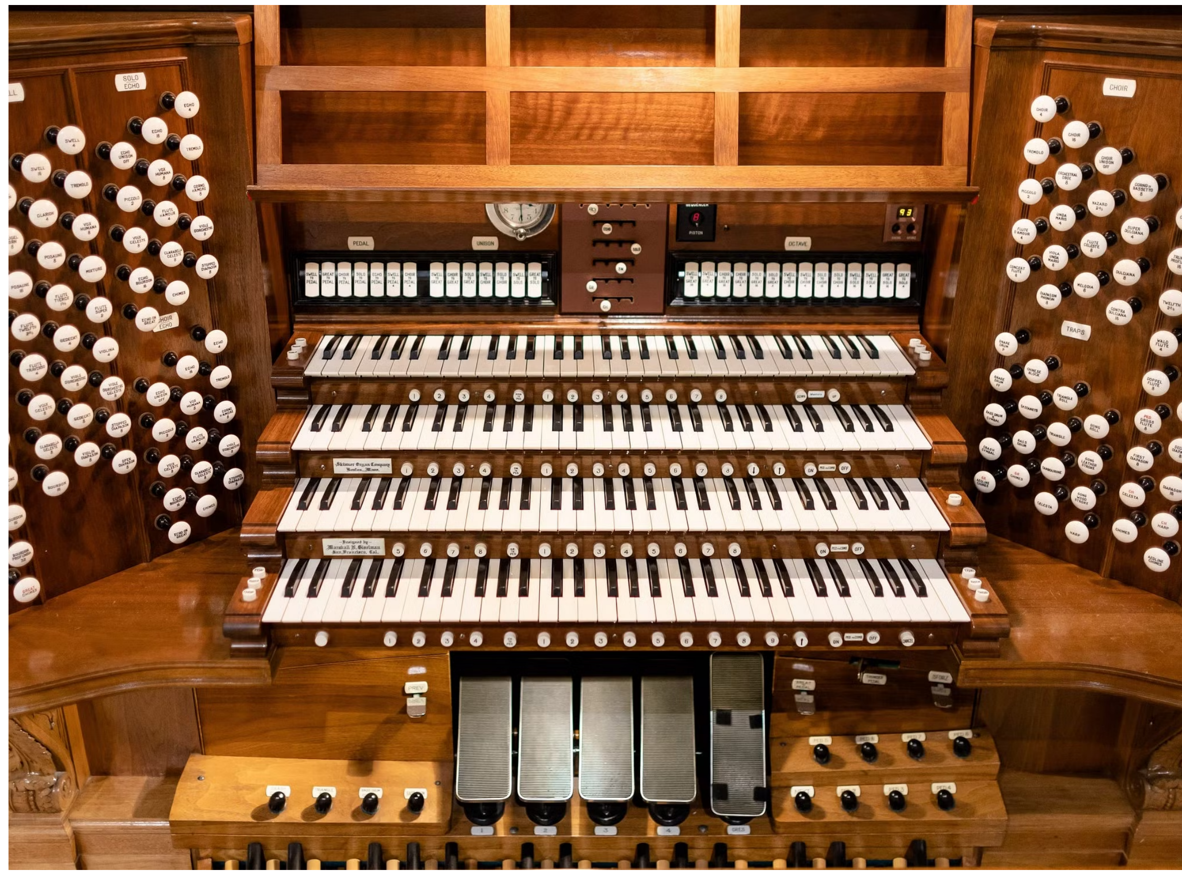


FIG. 52. — Front view of the viscera. a, spleen; b, heart; d, diaphragm; e, liver;



Your code is hard to test because it is not organized!

```
int foo(int a, int b);
```

```
foo  :: Integer -> Integer -> Integer
```

Quickcheck!

```clojure
(is (= 3 (+ 1 2)))
```

```clojure
(prop/for-all [x gen/nat
               y gen/nat]
  (let [z (+ x y)]
    (and (>= z x)
         (>= z y))))
```

QuickCheck

- quality requires coverage
- => large number of test cases
- => tests need to be fast
- => tests need to be deterministic
- Reproducibility requires determinism!

strangeloop

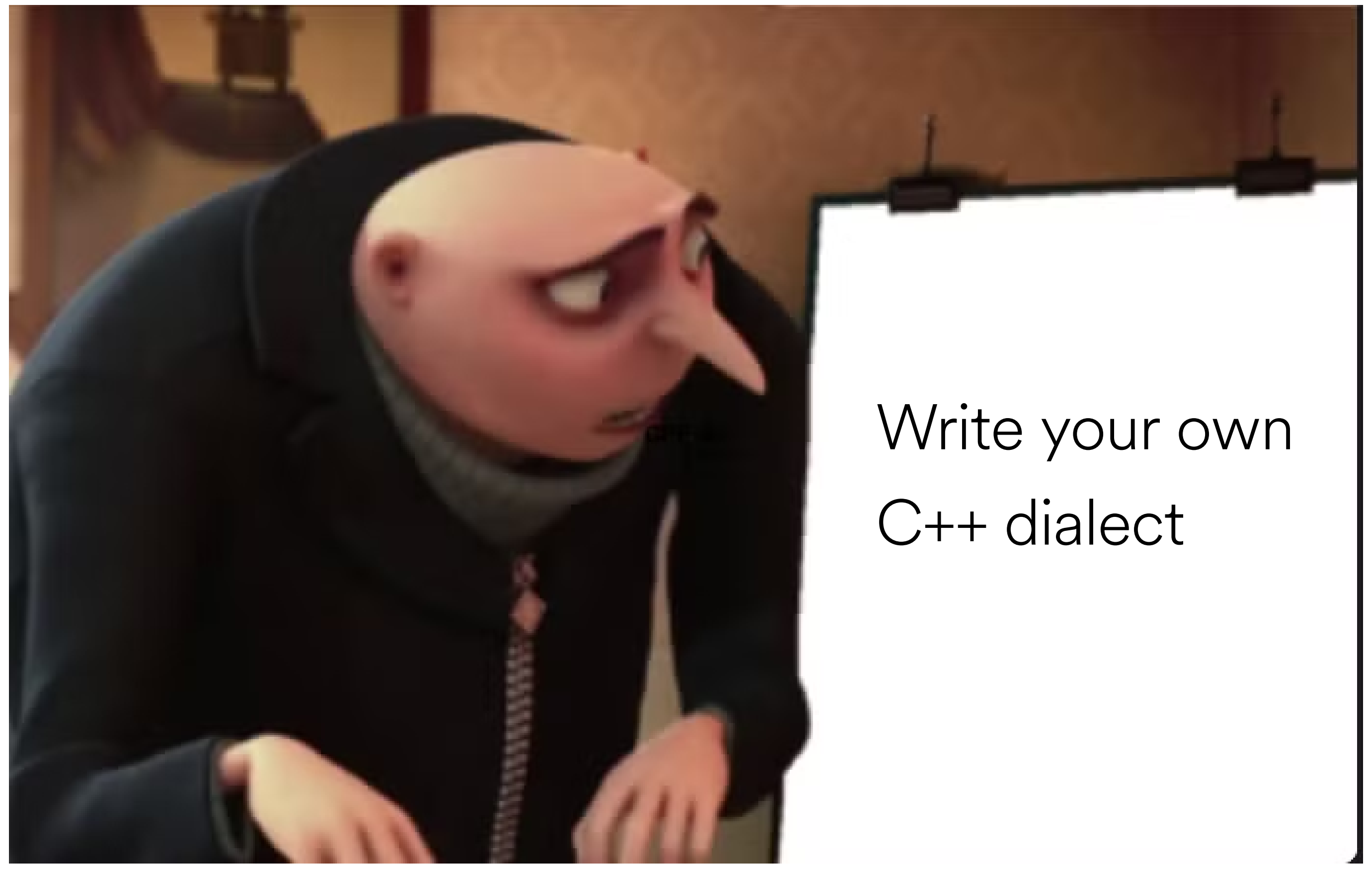
Sept 17-19, 2014 - St. Louis, MO
http://thestrangeloop.com

Test files

```
testTitle=SwizzledCycleTest
  testName=Cycle
  transactionsPerSecond=1000.0
  testDuration=30.0
  expectedRate=0.01

  testName=RandomClogging
  testDuration=30.0
  swizzle = 1

  testName=Attrition
  machinesToKill=10
  machinesToLeave=3
  reboot=true
  testDuration=30.0

  testName=ChangeConfig
  maxDelayBeforeChange=30.0
  coordinators=auto
```

FoundationDB Lessons Learned

- Write your own dialect of C++
- Isolate *all* sources of non-determinism
- hook it up to QuickCheck



test.contract

https://building.nubank.com.br/why-we-killed-our-end-to-end-test-suite/

Contracts

https://github.com/griffinbank/test.contract

Quickcheck testing of stateful services

Traditional options

- Mocks: brittle and can hallucinate
- Integration tests: slow and non-deterministic

AWS S3

```
(defprotocol S3
  (get-object [this name])

  (put-object [this name])

  (delete-object [this name])

  (list-objects [this]))

(do

  (put-object s3 "foo")

  (get-object s3 "foo"))

=> ???
```

Model

```
(c/model
   :methods [#'s3/put-object
                :args (fn [_] (gen/tuple gen/string gen/bytes))
                (c/method
                  (fn [state args]
                    (let [[filename content] args]
                      (c/return :spec (fn [status] (= 201 status))
                                :gen (gen/return 201)
                                :next-state (assoc-in state [:objects filename] content))))
```

Model Options

Verify

- gen sequence of calls using the model
- collect expected return values
- compare against a real implementation!
- run in CI or nightly

Mock

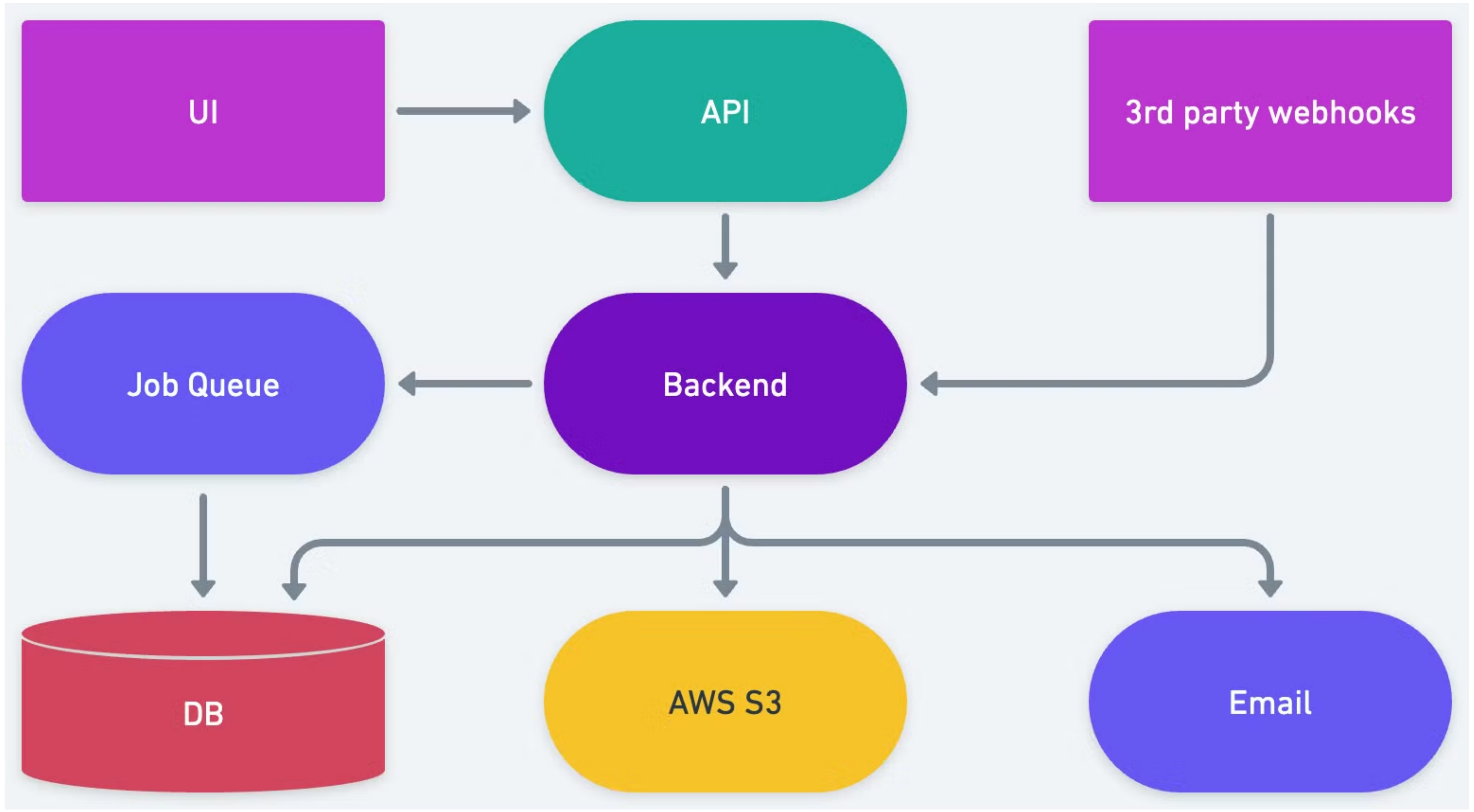
- Use in all other tests in the codebase
- Local development

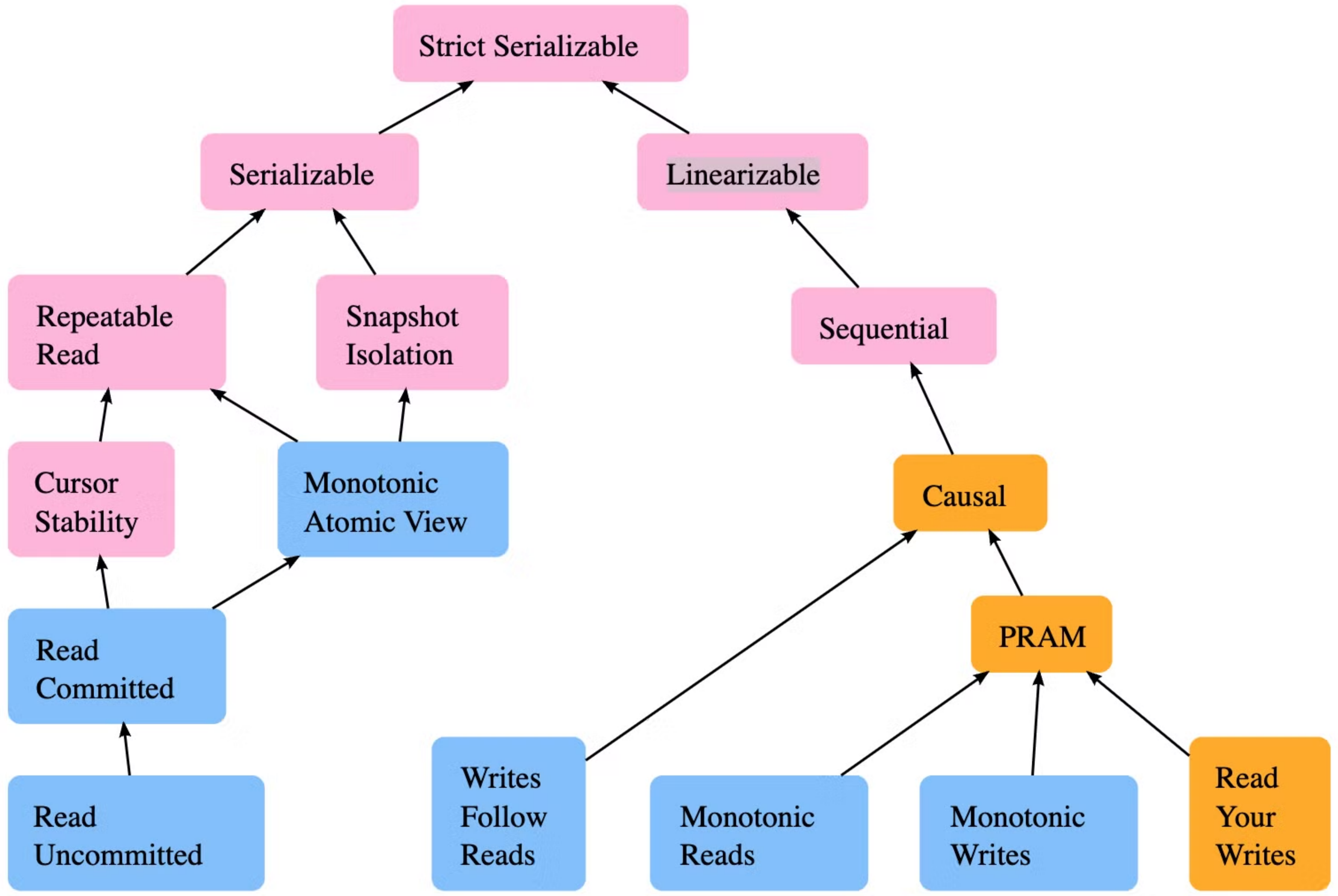
Test Proxy

- Call both implementations
- Extra validation on real-world use

Inject Errors

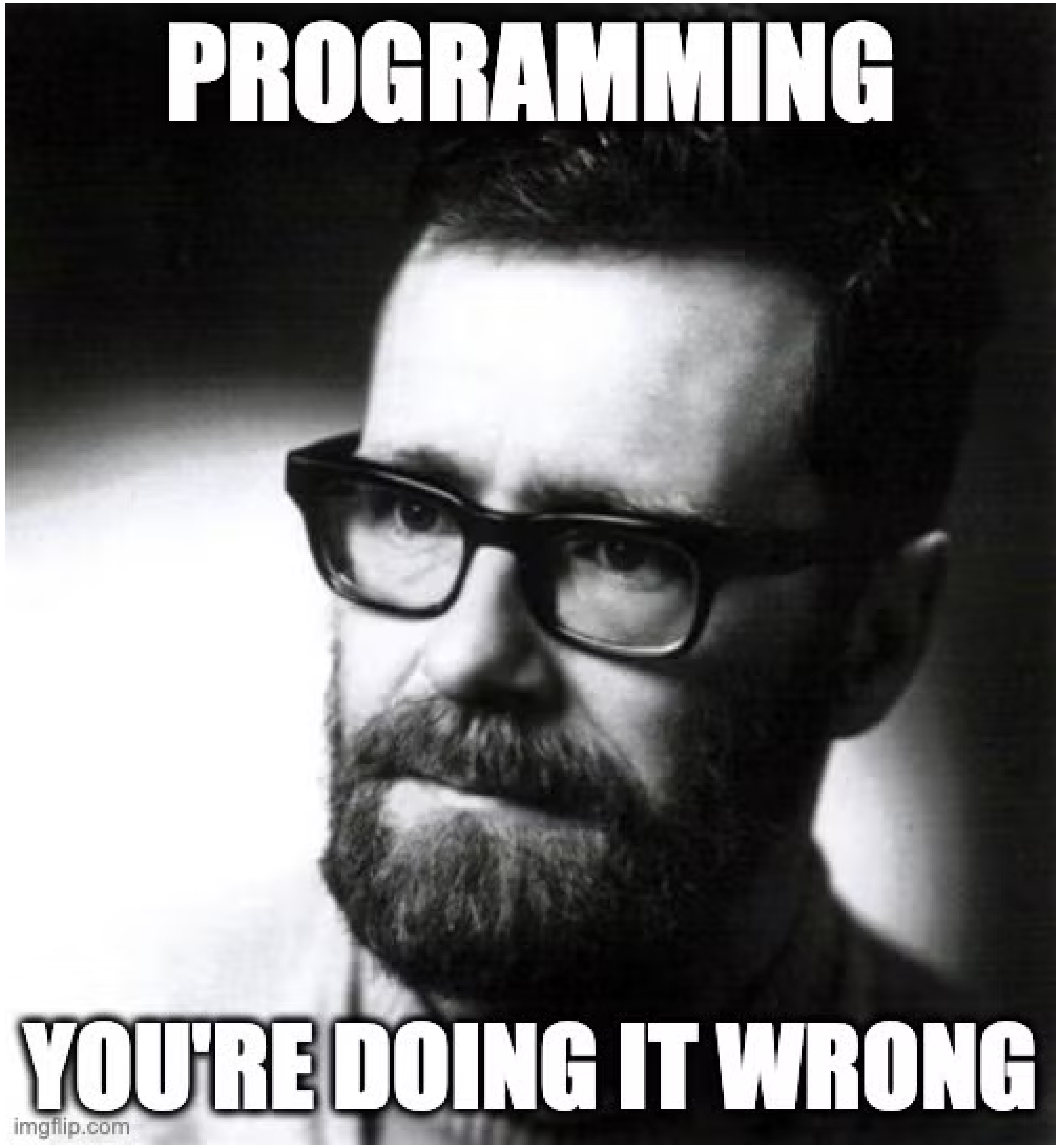
- Coming soon!


UI
API
3rd party webhooks
Job Queue
Backend
DB
AWS S3
Email


Strict Serializable
Serializable
Linearizable
Repeatable Read
Snapshot Isolation
Sequential
Cursor Stability
Monotonic Atomic View
Causal
Read Committed
PRAM
Read Uncommitted
Writes Follow Reads
Monotonic Reads
Monotonic Writes
Read Your Writes

In summary

1. Isolate *all* side effects and non-determinism
2. Quickcheck all the things
3. Be explicit about guarantees
4. Test your guarantees



Thank You

We're Hiring

https://griffin.com/careers