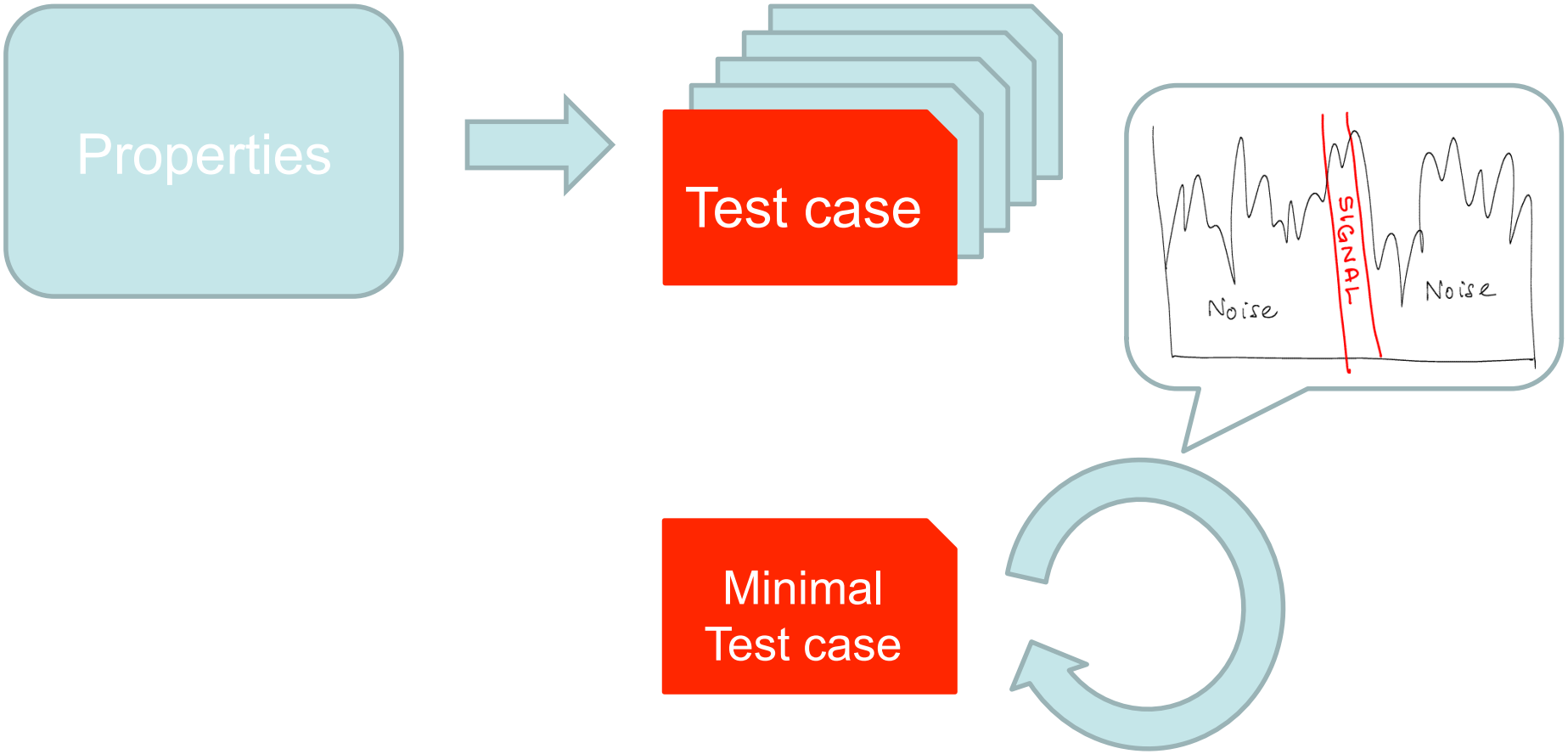
Testing Asynchronous APIs With QuickCheck

Thomas Arts

Quviq AB

QuickCheck in a Nutshell



Instead of writing test cases....

they are automatically generated from properties

Useful for

Unit Testing, Component Testing, System Testing

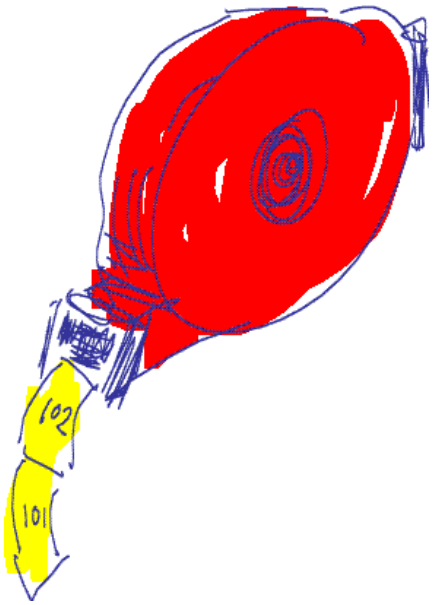
Less work, better testing, more fun

Most developers agree that writing unit tests is useful

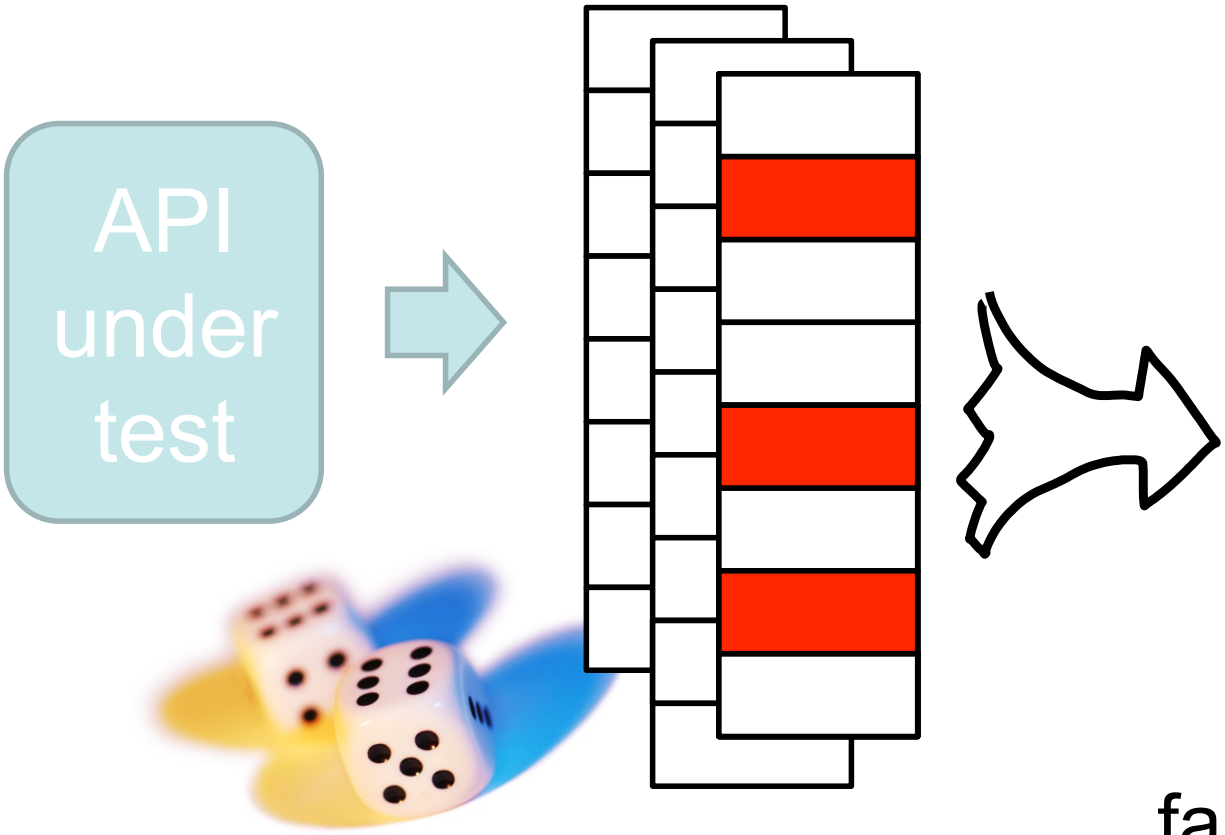
…. but also quickly gets boring …

An example:

API
take
reset



QuickCheck



A minimal
failing example

A Unit Test in Erlang

```
test_dispenser() ->
    ok =  reset(),
    1  =  take(),
    2  =  take(),
    3  =  take(),
    ok =  reset(),
    1  =  take().
```

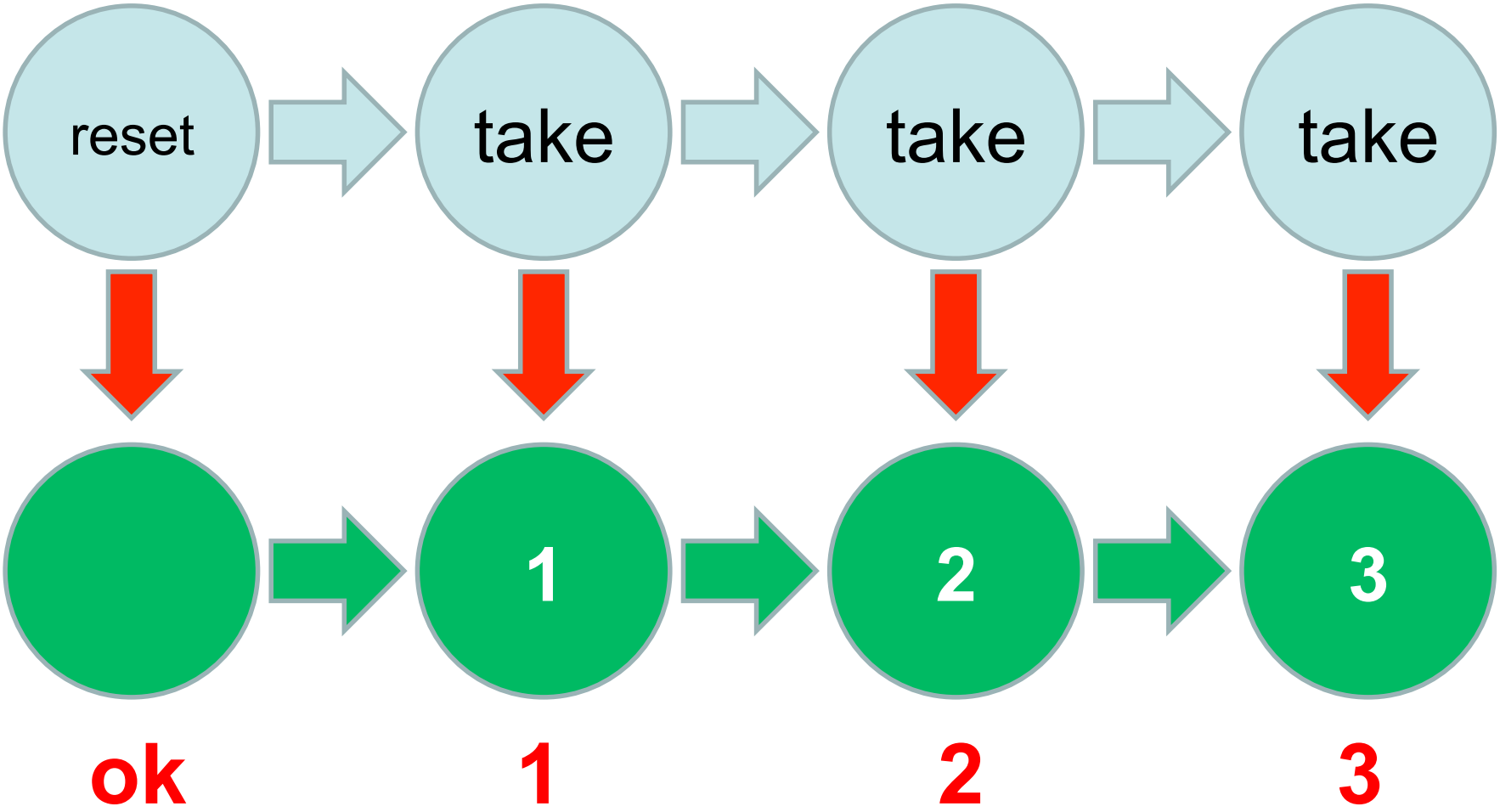
Expected results

State machine model

The *generator* for testing a sequence of commands is a state machine specification

The *property* is that a run of the generated sequence satisfies all postconditions.

Modelling the dispenser



State Machine for arbitrary sequence

Live

DEMO

```
initial_state() ->   undefined.
```

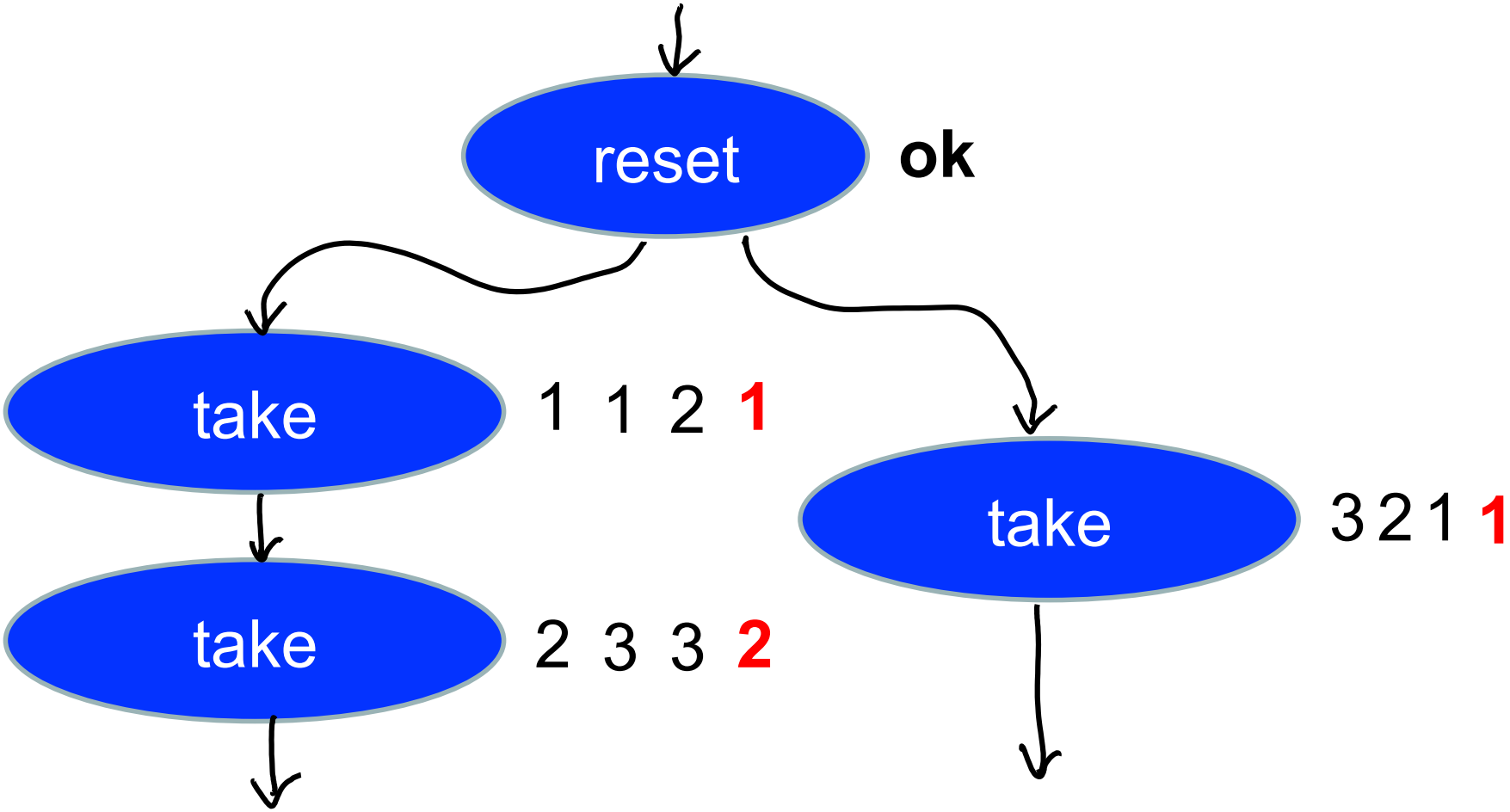
reset

```
reset_args(_State) ->  [].

reset() -> get("http://localhost:4000/reset").

reset_next(_State, _Result, []) -> 1.
```

take

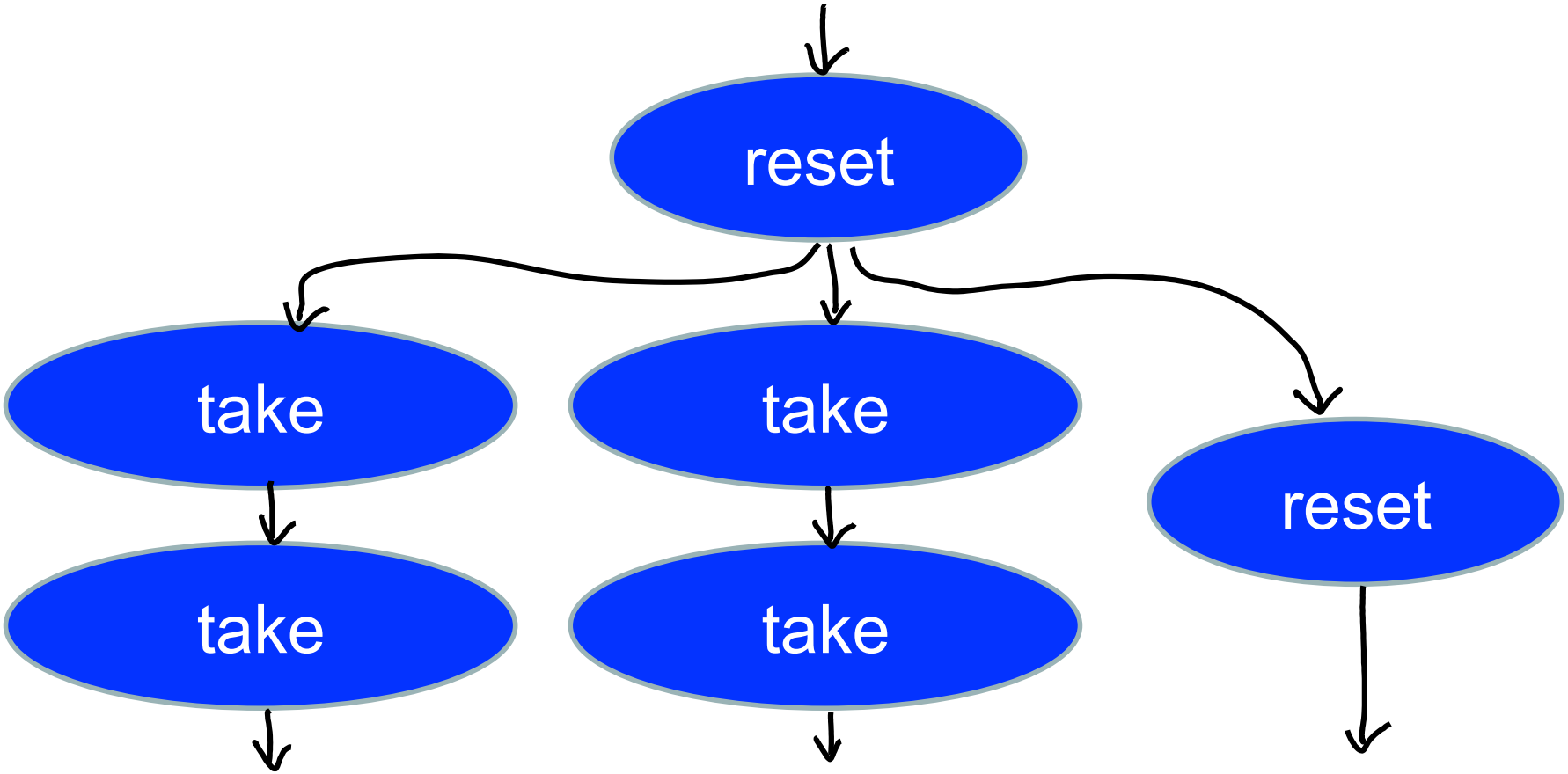
```
take_pre(State) ->  State /= undefined.

take_args(_State) ->  [].

take() -> get("http://localhost:4000/take").

take_next(State, _Result, []) -> State + 1.

take_post(State, [], Result) ->  eq(Result, State + 1).
```

A Parallel Unit Test



- Three possible correct outcomes!

Another Parallel Test



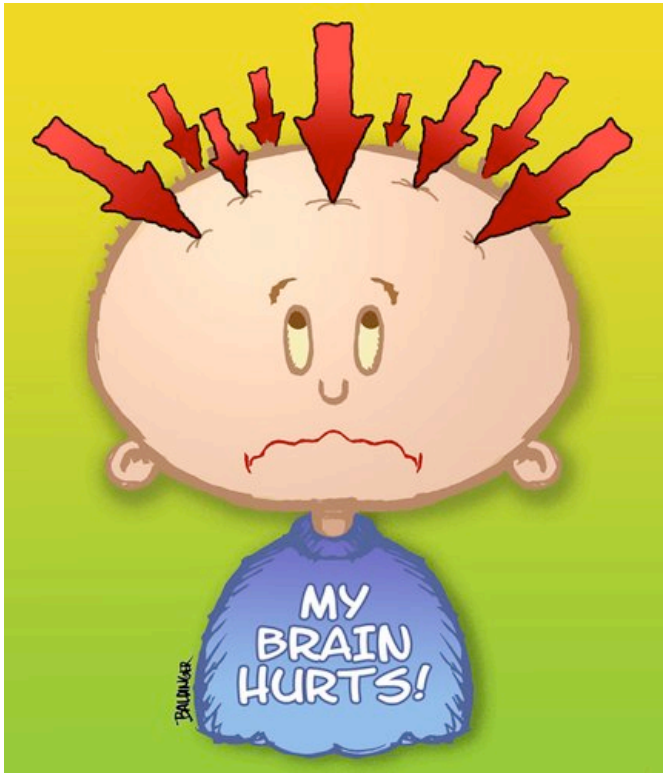
30 possible correct outcomes!

Writing unit tests for concurrent events:

Headache!

Thus, people don't!

QuickCheck does it for you!



Arbitrary sequences

Live

DEMO

```
prop_dispenser() ->
  ?FORALL(Cmds, parallel_commands(?MODULE),
    begin
       {H, S, Res} = run_parallel_commands(?MODULE, Cmds),
       pretty_commands(?MODULE, Cmds, {H, S, Res},
                       Res == ok)
    end).
```

QuickCheck properties:

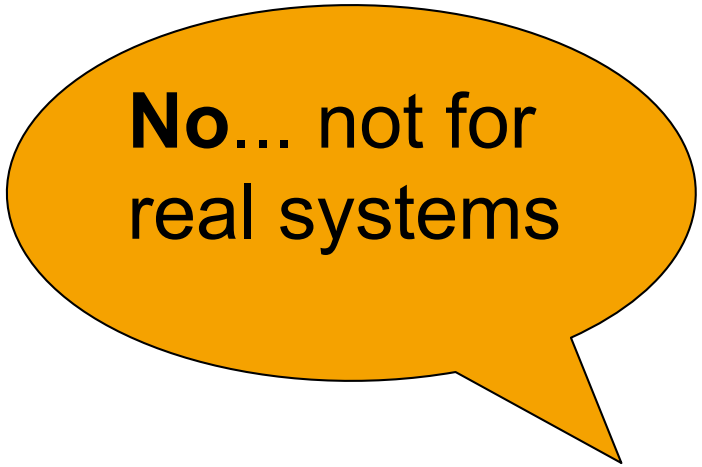
Property specifies behaviour of any command sequence

QuickCheck

- runs the sequences with different threads
- collect the results
- checks whether this can be explained from sequential behaviour

Commonly asked question:

Do we need to re-implement the software as a model?



How to test project-fifo?

project-fifo.net

A complete system

Project-fifo

- Cloud Orchestration
- Manage private and public clouds
- Based on SmartOS / Solaris Containers
- OSS and Commercial Support

- Self hosted, Distributed, Highly available, Eventually consistent

Architecture

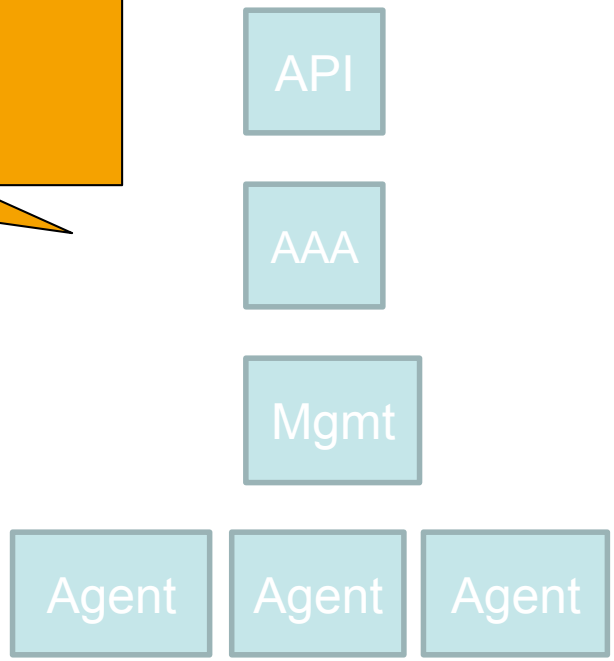
HTTP / RE

60,000 lines of code

AAA: OAuth2, RBAC

Business logic, database, tracking

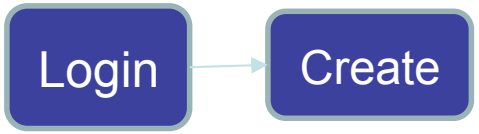
Agents to manage physical components

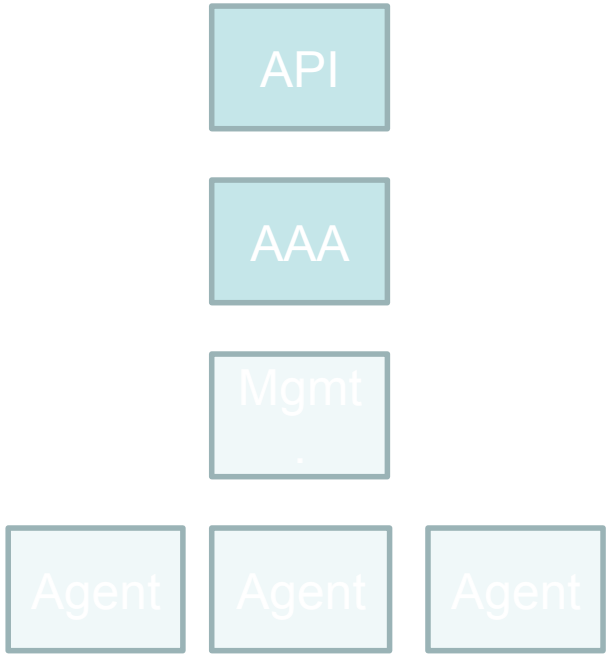


Lets test

The problem

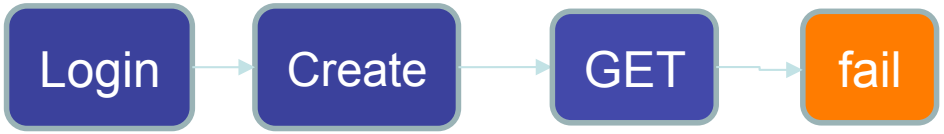
Lets test



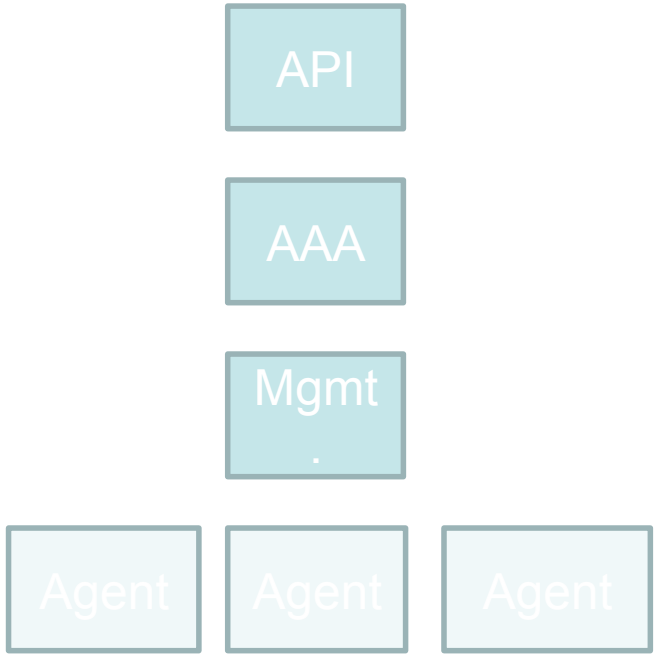


The problem

Lets test

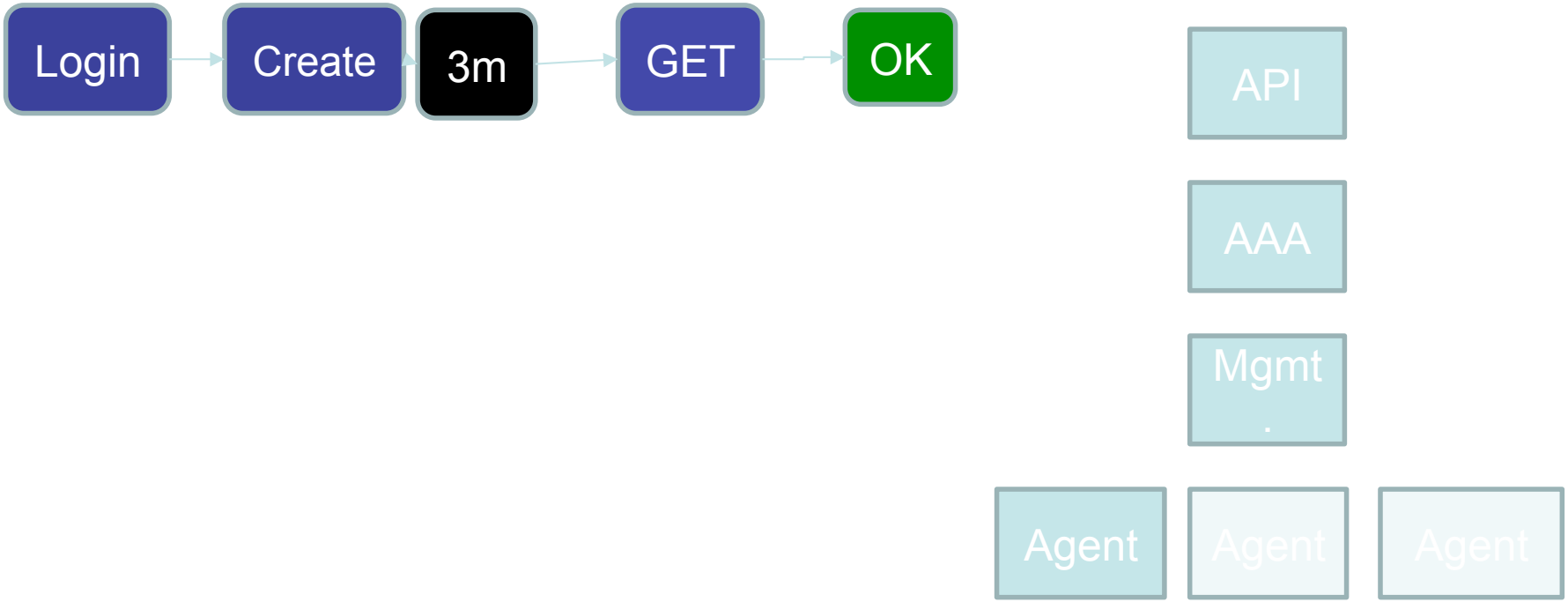


This fails, create is asynchronous



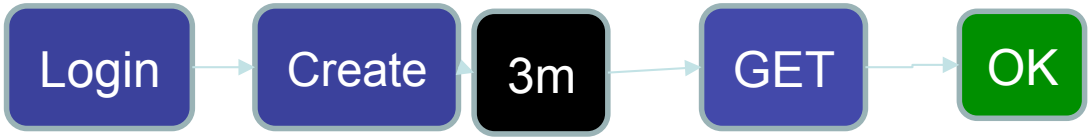
The problem

Lets test

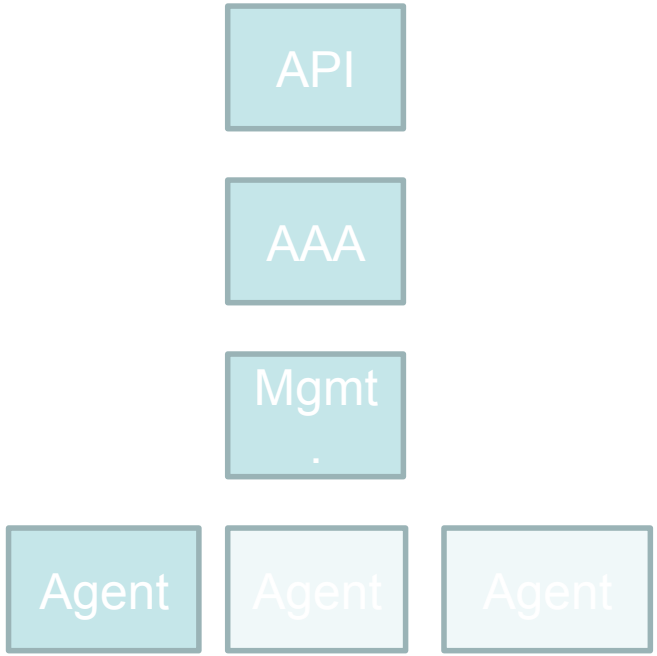


The problem

Lets test

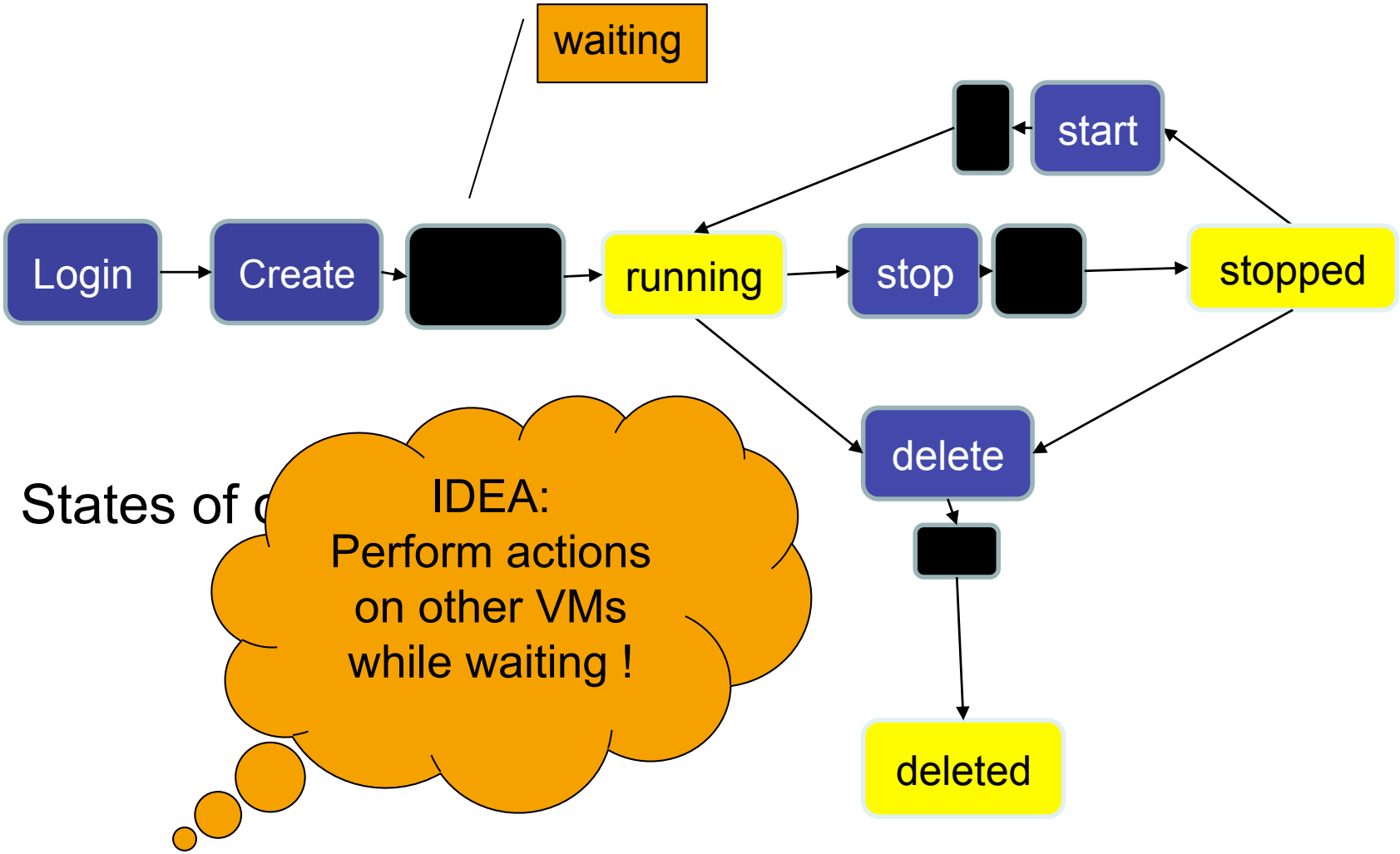


- utterly slow testing
- hard to test around "just started"
- forget race condition testing

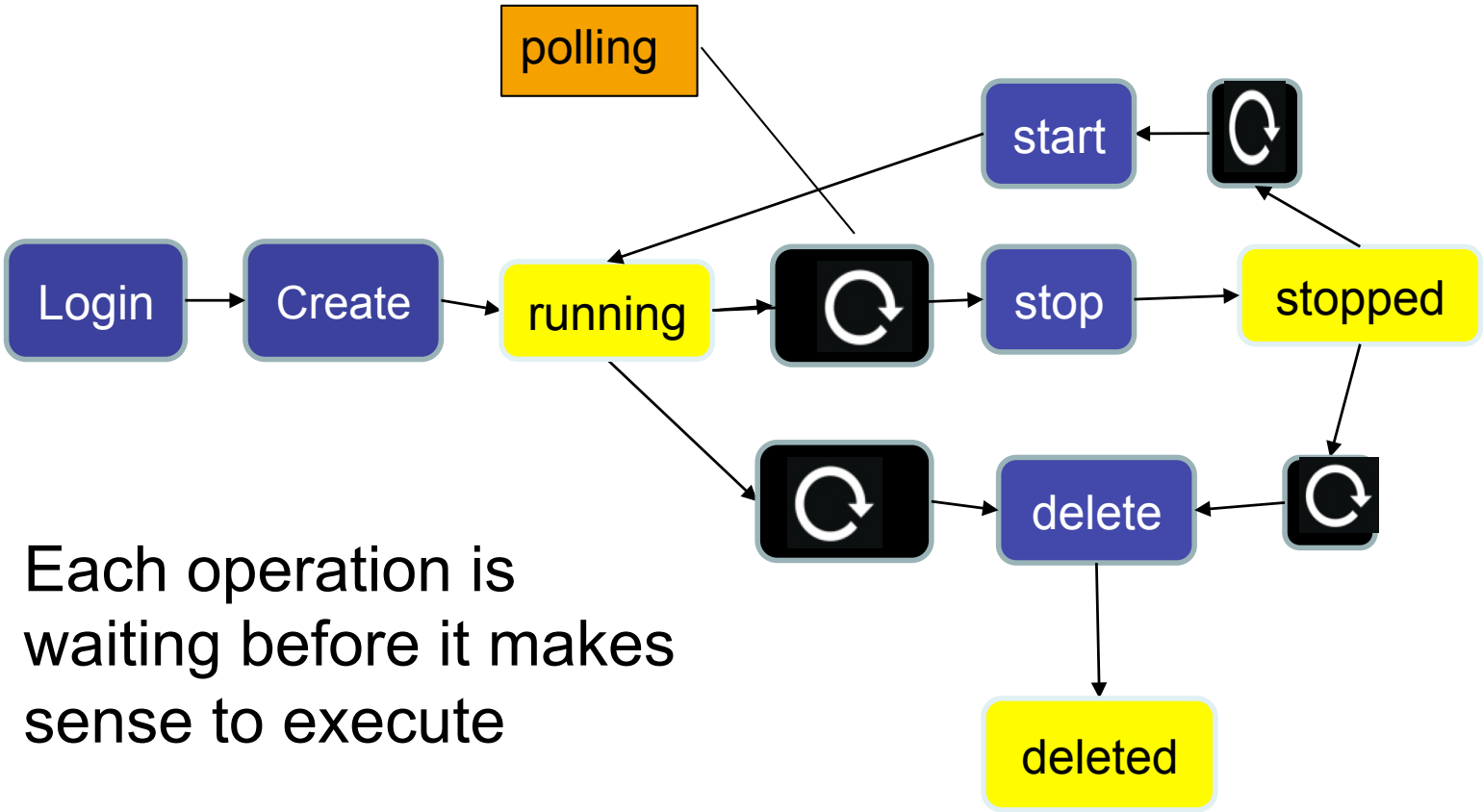


The problem

Asynchronous API



Asynchronous API



Each operation is waiting before it makes sense to execute

and add: sleep

Running QuickCheck tests

revealed 25 errors... all fixed now ☺

Timing errors, race conditions, type errors, incorrect use of library API, error in documentation, errors in the logic, system limits error, errors in fault handling...

and coincidently a hardware error

A complete system

60,000 lines of code

460 lines of QuickCheck

Any reasonable test suite would contain more lines of code...

... and find less errors.