QuickCheck for EUnit

(using unit tests differently)

Thomas Arts

Quviq AB

Joined work with
Simon Thompson
Pablo Lamela

Unit tests

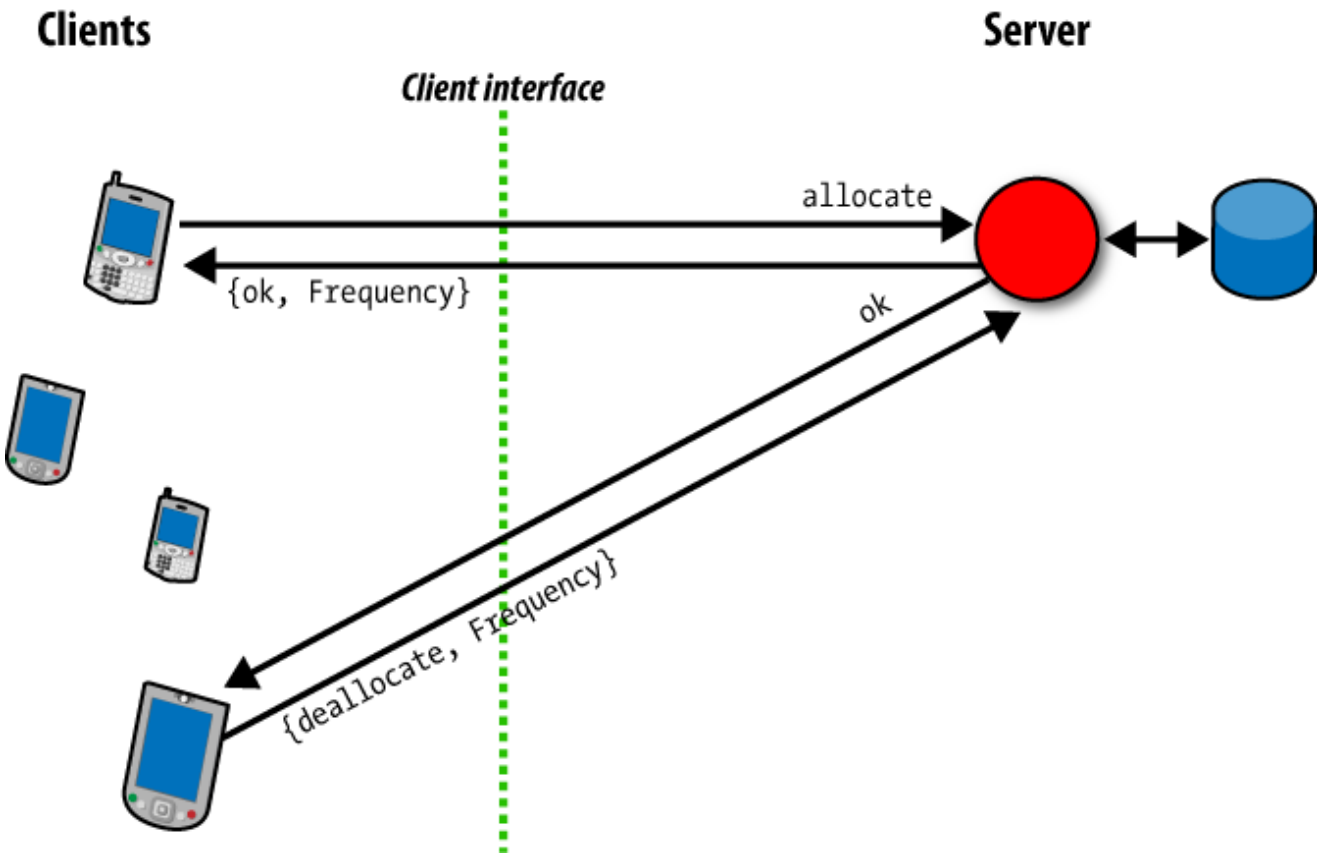
Developers write Unit tests… but never enough

How can we help them?

- Use the existing tests to learn (machine learning)
- Visualize the information in the tests
- Automatically propose new tests

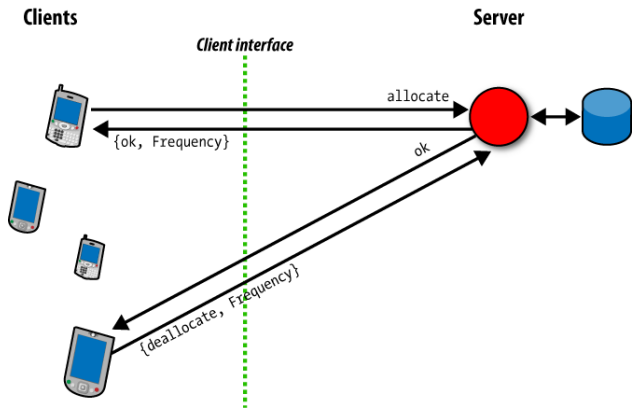
(for details: ACM Sigplan Erlang workshop publications, 2010 + 2011).

Server for mobile frequencies



Server for mobile frequencies

State-based system allows allocation and de-allocation of frequencies from an initial list, once system is started.

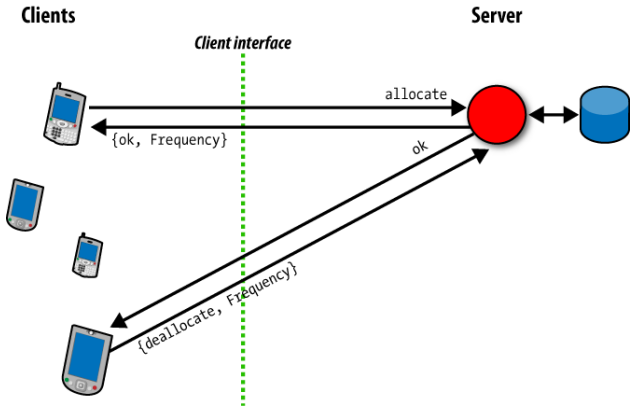



```
-spec start([integer()]) -> true.
-spec stop() -> ok.
-spec allocate() -> {ok,integer()} |
                    {error, no_frequency}.
-spec deallocate(integer()) -> ok.
```

Unit testing the server

EUnit is a unit testing framework for Erlang.

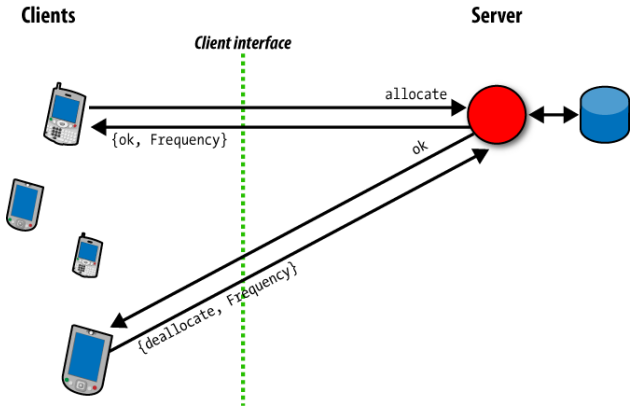
Test start / stop behaviour.




```
startstop_test() ->
    ?assertMatch( … ,start([])),
    ?assertMatch(ok,stop()),
    ?assertMatch( … ,start([1])),
    ?assertMatch(ok,stop()).
```

Which models meet the test?

What is the simplest FSM model which meets this test?



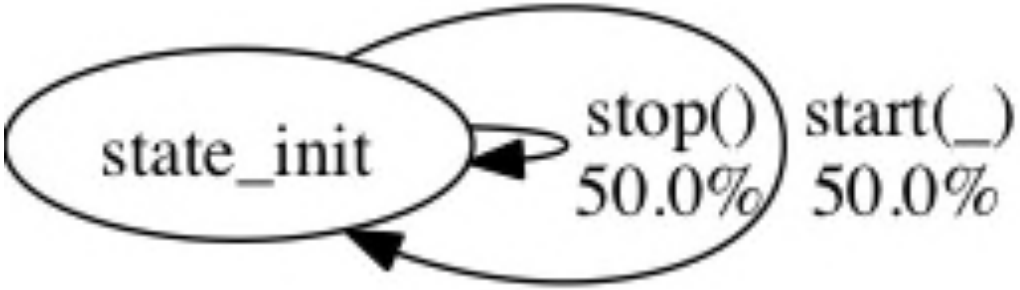

```
startstop_test() ->
    ?assertMatch( … ,start([])),
    ?assertMatch(ok,stop()),
    ?assertMatch( … ,start([1])),
    ?assertMatch(ok,stop()).
```

Models from unit tests

DEMO
- running the test(s)
- visualzing the model

Model 1

Simplest model meeting the test.




```
startstop_test_() ->
   ?assertMatch( … ,start([])),
   ?assertMatch(ok,stop()),
   ?assertMatch( … ,start([1])),
   ?assertMatch(ok,stop())]}.
```

QuickCheck Model

QuickCheck automatically generates test...

from a specification

The state machine is a specification

automatically create textual representation

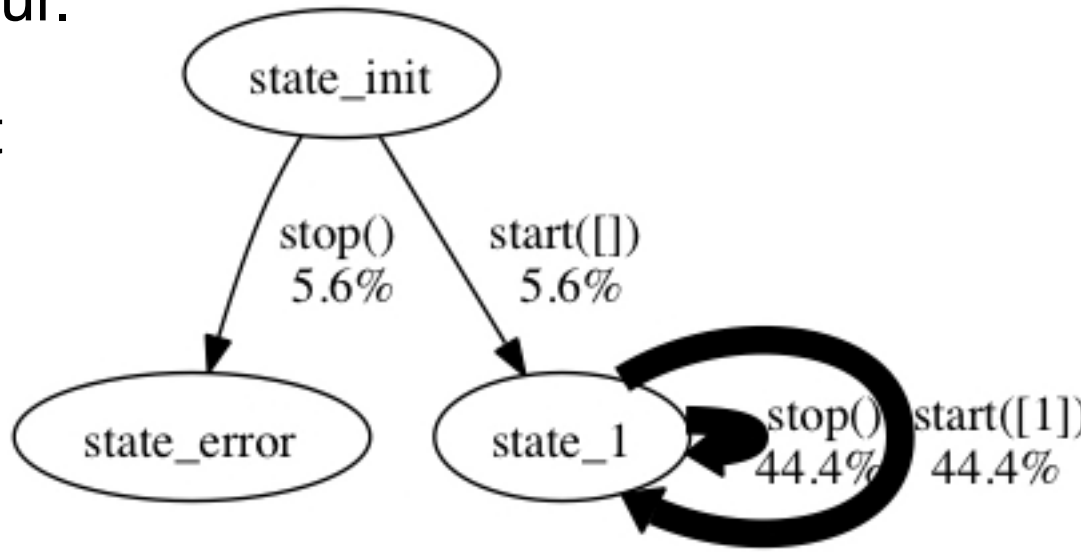
DEMO:

- show the representation
- run QuickCheck

Model 2

Test start / stop behaviour.

Can't stop a server that isn't running.




```
startstop_test() -> …

stopFirst_test() ->
     ?assertError(badarg,frequency:stop([])).
```

Demo

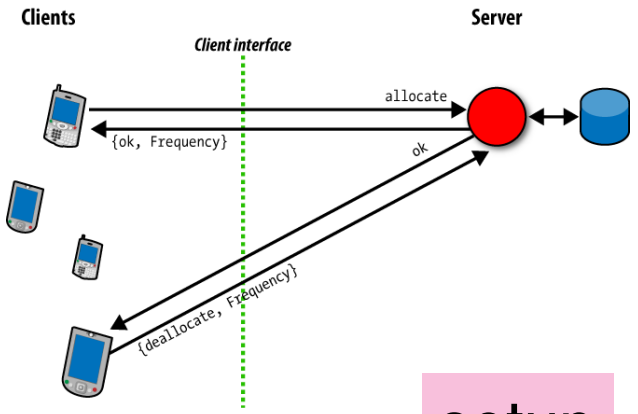
- Run QuickCheck again

Test set 3

Test start / stop behaviour.

Can't stop a server that isn't running.

Can't start a running server.




```
start_twice_test_() ->
      {setup,
       fun() -> start([]) end,
       fun(_) -> stop() end,
       fun() -> ?_assertError(_,start([])) end}.
```

setup

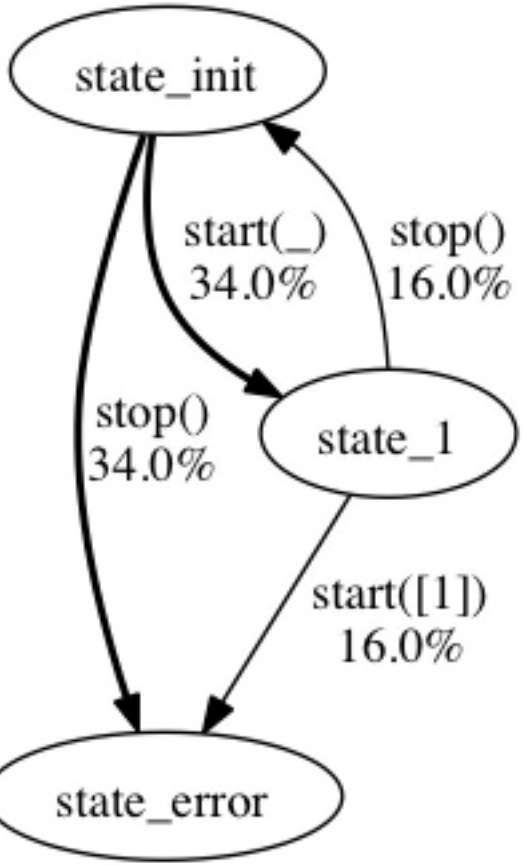
teardown

test

Demo

- Run QuickCheck again

Test set 3



Test start / stop behaviour.

Can't stop a server that isn't running.

Can't start a running server.

```
start_twice_test_() ->
     {setup,
      fun() -> start([]) end,
      fun(_) -> stop() end,
      fun() -> ?assertException(_,_,start([])) end}.
```

Tests for free!

- Build new tests from existing test suites.
- Property-based testing:

  Check the system with **any** path through the finite-state machine for **any** value of the API arguments.

Implementation

- Extract traces from Eunit tests: dynamic or static
- Add assertions to the trace, such that return values are extractable
- Use bluefringe algorithm for learning fsm
- Compute machine on an abstraction, but keep real values
- Introduce generator (oneof) when several arguments give same transition

Conclusion

Writing EUnit tests gets fun!