Mission Critical with Erlang And QuickCheck *Quality Never Sleeps*

Raghav Karol
Motorola Solutions

Torben Hoffmann
Motorola Solutions

Setting the stage

Pop Quiz

- Which of the items returned to Gordon Gecko represent QuickCheck?
  - Silk handkerchief
  - Gold watch
  - Ring
  - Gold money clip with no money in it
  - Mobile phone

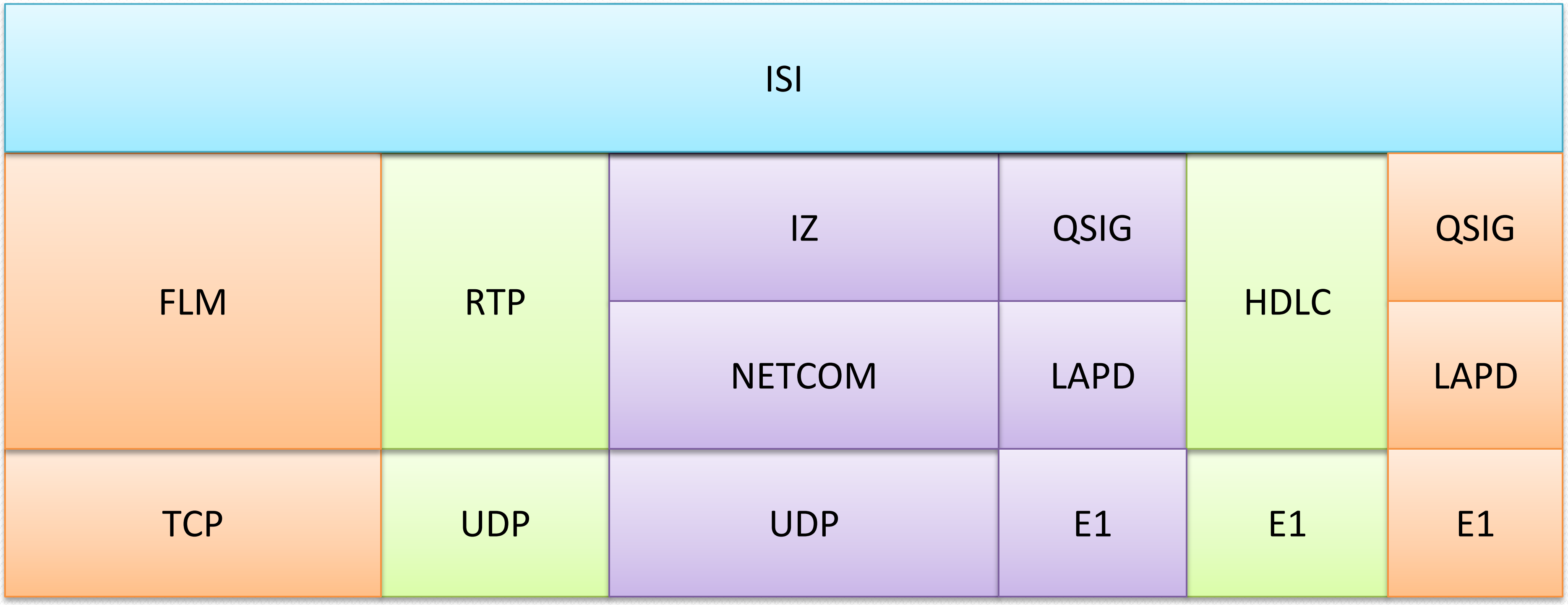
Overview

- What did we have to do?
- What did QuickCheck help us with?
- How is it to use Erlang?
- How productive is Erlang/OTP?

What did we have to do?

- ISI Project
  - Gateway to interconnect two TETRA systems
    - Migration for TETRA
  - State-full protocol conversion
    - Motorola Proprietary Call Control and Mobility
    - ISI – ITSI **I**nter**S**ystem **I**nterface open standard
    - Q.SIG, HDLC, LAPD, E1
    - Mobility and Resource Management
  - High concurrency requirement
  - High reliability – required to connect to live customer system

ISI Stack


ISI
FLM
RTP
IZ
QSIG
NETCOM
LAPD
HDLC
QSIG
LAPD
TCP
UDP
UDP
E1
E1
E1

Motivation and Background

- Working prototype to be delivered as a product
  - Existing codebase created for IOP certification
- Connect to live system
  - Main requirement – Should **not** crash existing system
  - Specification of legacy protocols not complete
- Small team, 5 members, no dedicated test resources
- ISI application Erlang + C
  - Enter Property based testing and QuickCheck

Romania

And he’s just the requirements guy ☺!

Unit test versus Property based testing

- Testing using xUnit like tools
  - Setup Fixture, execute test case, Teardown Fixture
    - Works well --- Used with good success before using property based testing
    - Important to have good test cases
    - Test cases and scenarios easily overlooked
    - Maintenance and refactoring of test cases also always required
  - How is property based testing different
    - *Specify* rules of generating a test case
    - *Specify* pre-conditions when above rules can be valid
    - *Model* expectations once a rule executes
    - Check post-conditions once a rule executes
  - What!?! Can this even work?

Quickcheck

- QuickCheck
  - Generators
  - Properties
  - Stateless testing

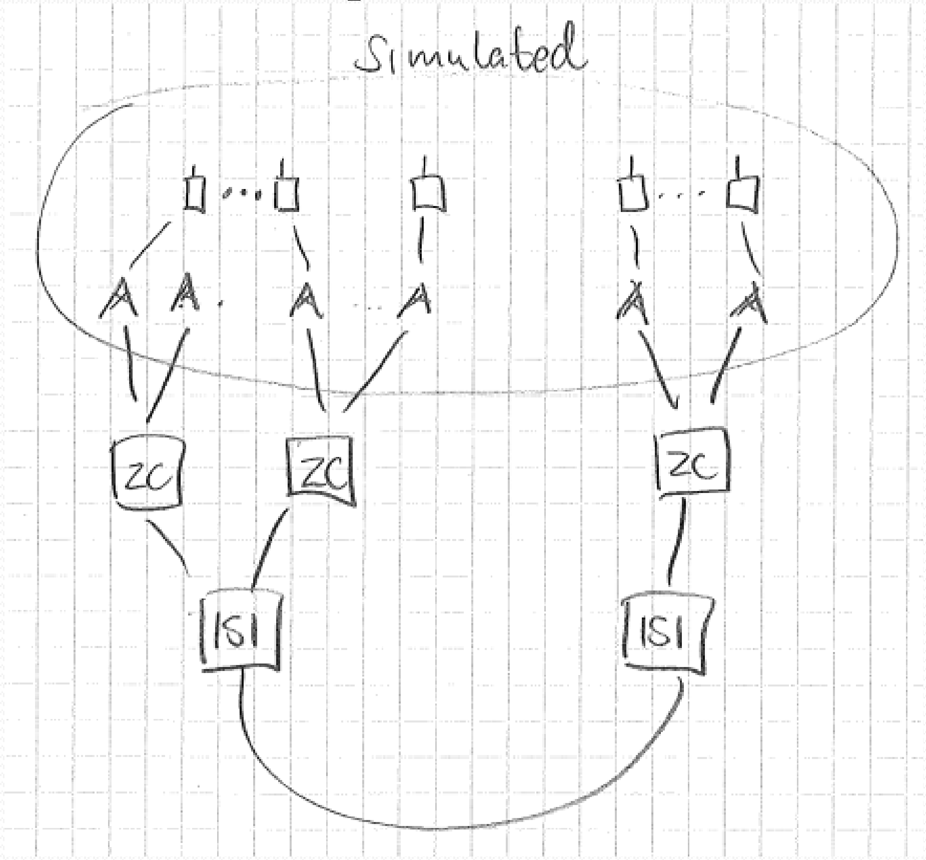
    Symbolic test cases
  - State full testing
    - State machines based
  - Shrinking
  - Also available for C --- Not used in our project.

Property based testing in ISI

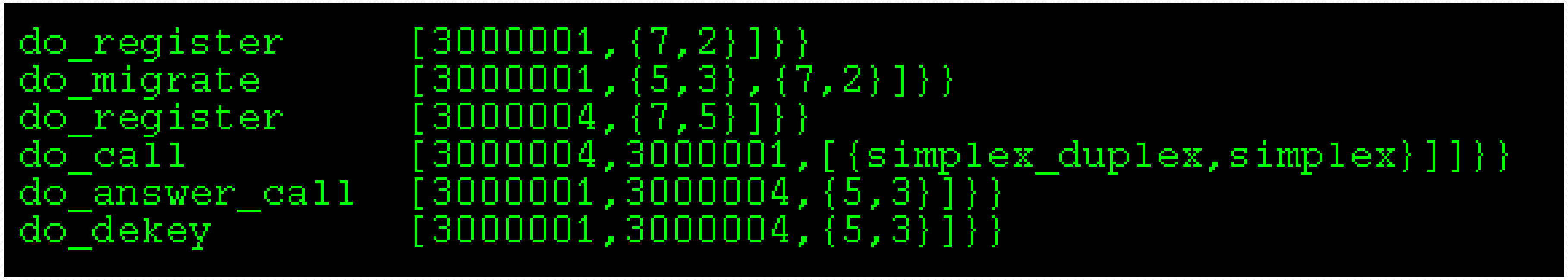
- Unit Level
  - Queue data structure
  - Resource management and allocation server
- Component level
  - NetComm Layer
- Black box – Box test level

Black box testing of the ISI GW

- Simulated sites and mobiles
  - Implementation of the ZC-Site protocol
  - Pseudo Air Interface protocol



Black box test example



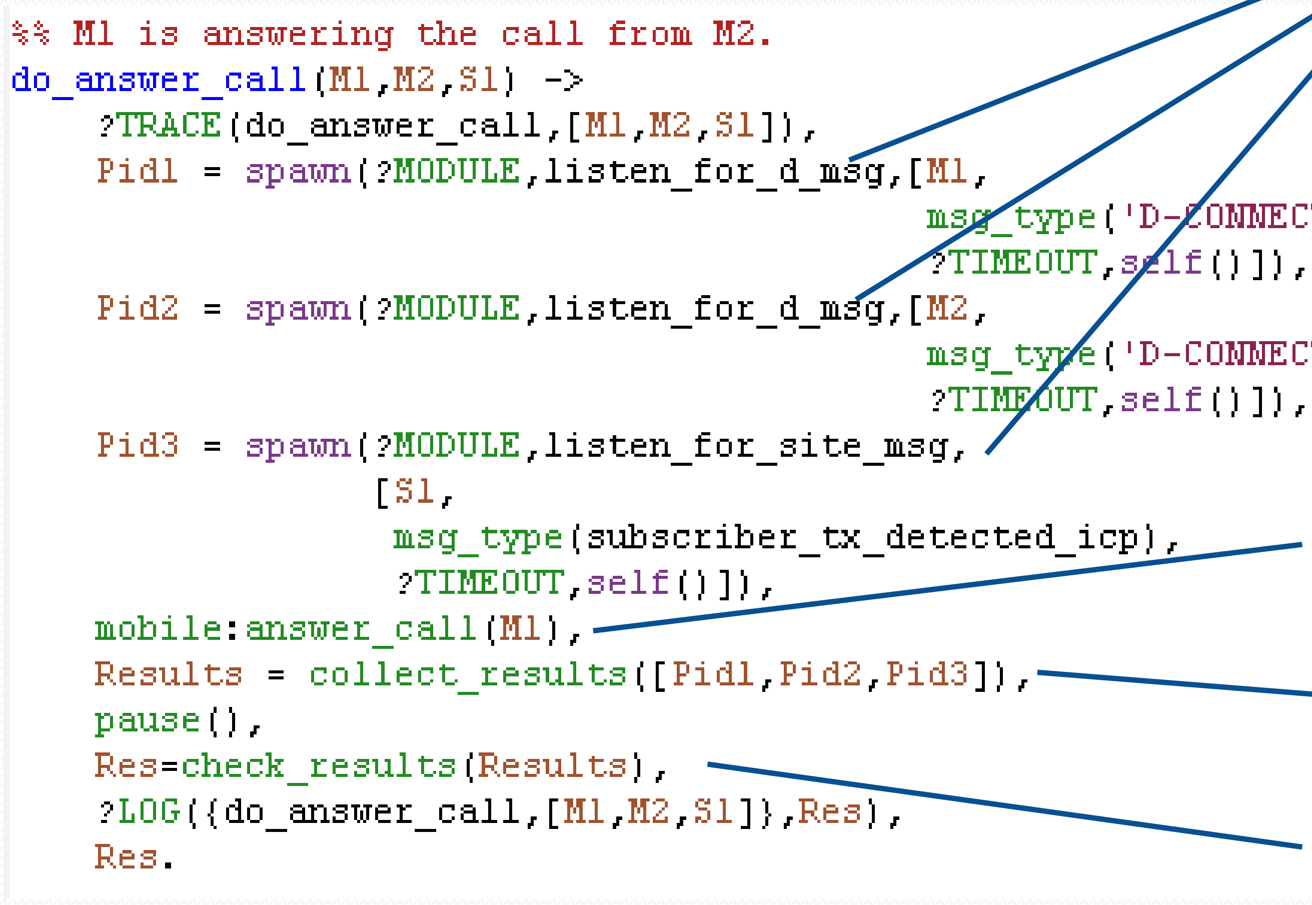
```
do_register       [3000001,{7,2}]}}
do_migrate        [3000001,{5,3},{7,2}]}}
do_register       [3000004,{7,5}]}}
do_call           [3000004,3000001,[{simplex_duplex,simplex}]]}}
do_answer_call    [3000001,3000004,{5,3}]}}
do_dekey          [3000001,3000004,{5,3}]}}
```

- The test cases are expressed in terms of actions taken by the mobiles

Handling Concurrency in QC

- In many cases the success of an operation required several messages to occur
- Enter...
  - hooks
- A hook spawns a listener processes for each message that is expected

Hook Example



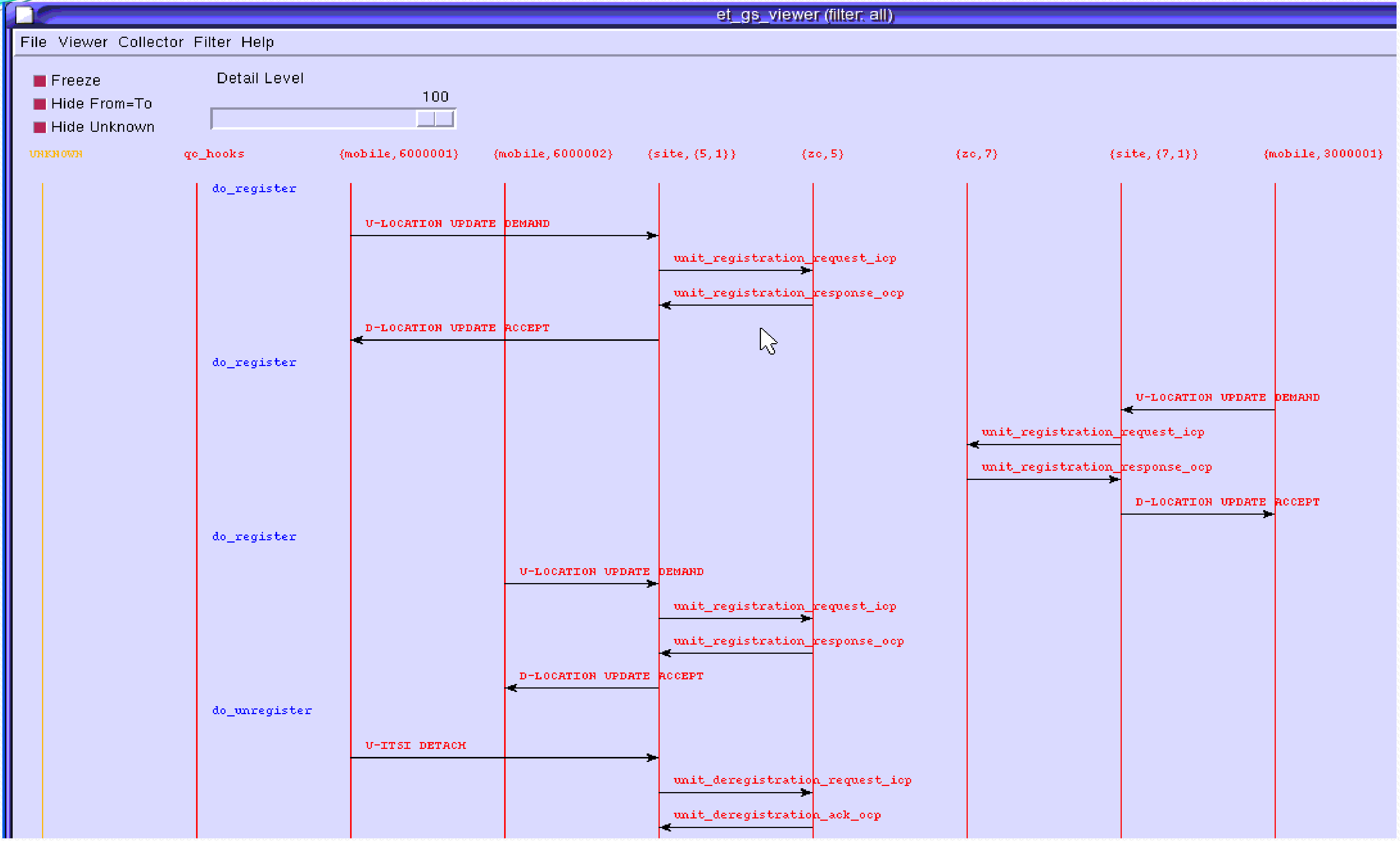
```
%% M1 is answering the call from M2.
do_answer_call(M1,M2,S1) ->
    ?TRACE(do_answer_call,[M1,M2,S1]),
    Pid1 = spawn(?MODULE,listen_for_d_msg,[M1,
                                            msg_type('D-CONNECT ACKNOWLEDGE'),
                                            ?TIMEOUT,self()]),
    Pid2 = spawn(?MODULE,listen_for_d_msg,[M2,
                                            msg_type('D-CONNECT'),
                                            ?TIMEOUT,self()]),
    Pid3 = spawn(?MODULE,listen_for_site_msg,
                 [S1,
                  msg_type(subscriber_tx_detected_icp),
                  ?TIMEOUT,self()]),
    mobile:answer_call(M1),
    Results = collect_results([Pid1,Pid2,Pid3]),
    pause(),
    Res=check_results(Results),
    ?LOG({do_answer_call,[M1,M2,S1]},Res),
    Res.
```

Listen until stated message recived

The Operation

Gather results from listeners

Was everything as expected?



Freeze
Hide From=To
Hide Unknown

Detail Level
100

QuickCheck Quality Never Sleeps

- So you made a testing framework – I can do that!
- Yes – every software department in our company, including us has
- Why QuickCheck
  - More thinking, specify, specify, specify, less work
  - Randomness

Visualizing auto tests

Visualizing auto tests

Erlang/OTP Quality never sleeps

- Auto-enforcement of a coding standard with Erlang/OTP
- Semantics of framework uncomfortably simple
- Ease of Distribution
- Supervisors - error handling and reliability
- Common case ALWAYS works
- Understanding software behavior
- Resolving Issues
  - Resolving issues in the field
  - Integration issues with other vendor forces: panic, stress, our reputation

Evaluation of performance

- The eternal problem:
  - Which approach is better?
- Key problem:
  - What is the size of the problem?
- One of the best size measures for software:
  - Function Points
    - Measures input and output and treats the software as a black box
    - Not widely used since it is time consuming to generate FP estimates for a system and even harder to check how many the final system has

Backfiring

- Backfiring:
  - Counting Function Points by looking at the actual code
- Our approach:
  - Use epp_dodger to extract incomming messages
  - Use xref to extract outgoing messages
  - Post-process in Excel to ensure counting the correct messages

Comparing with others

- Function Points are often used by estimation tools
  - Construx Estimate
  - COCOMO II
- Basic project estimation:
  - Inputs:
    - Function Points
    - Programming language
    - Type of project (Telecommunications)
  - Output:
    - Staff Months (SM) to complete the project

Erlang vs X

Language	Vs Erlang effort
Java	3x (2.3-3.9)
C++	4x (3.4-5.3)
C	7x (5.9-9.3)

So for a telecommunications project Erlang/OTP seems to be the right choice…

Conclusions

- For us Erlang solved
  - Complex technical issues
  - Communication in the team!
- Lowering Costs
  - Development
  - O&M
- QuickCheck
  - Leverages Erlang language features
  - Future of testing