Using Erlang for Testing non-Erlang Products

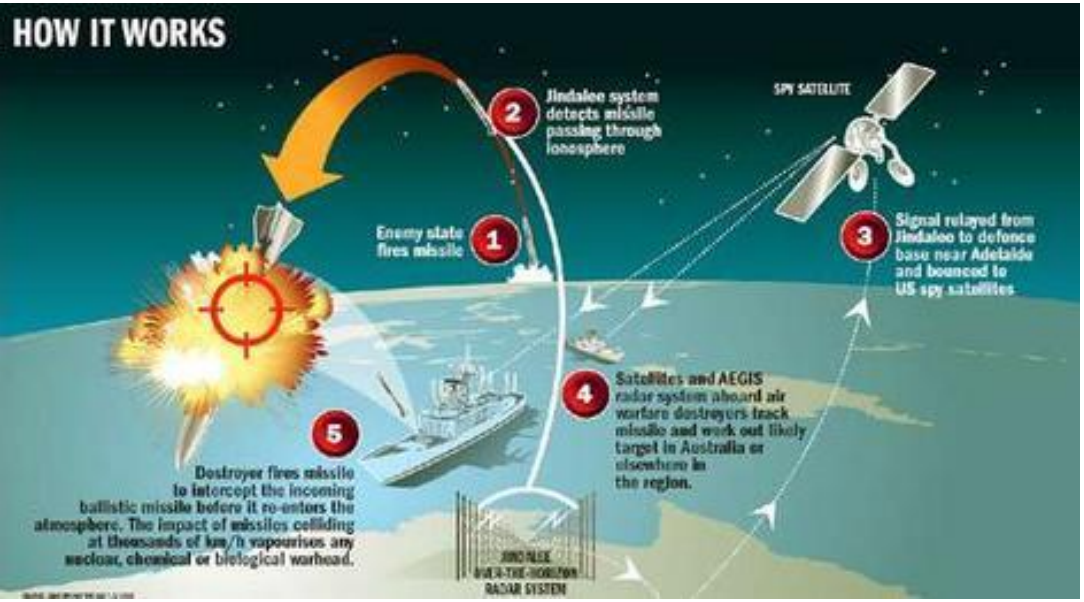
Test Automation and Test Generation

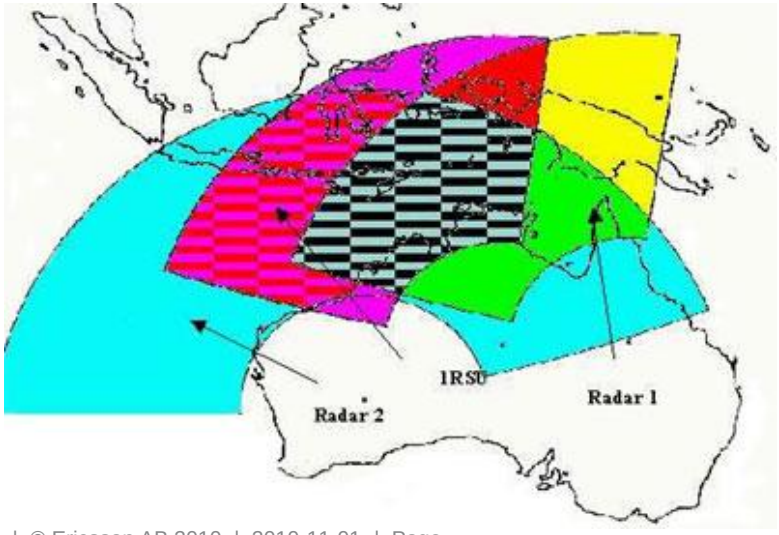
Graham Crowe

Introduction

- The story of my career:
  - my education
  - the roles I've had
  - the technology I have experienced
- This story will provide an insight as to why I champion Erlang for testing

Jindalee Operational Radar Network





CAND98/0153-25
JORN PROJECT RECEIVER SITE ANTENNA ARRAY, LAVERTON W.A.
PIC BY CPL DAVE BROOS, DEFENCE PUBLIC AFFAIRS.

The AXE 10 Years

- Structured and consistent
  - blocks, subsystems
- Proprietary language
  - PLEX (PASCAL flavoured)
  - ASA (assembler language)
- Execution Model
  - signaling between blocks
  - no shared data between blocks
  - global job buffers for signaling
- Debugging
  - “Test System;”
- Patchable (ASA)
- Forlopp

The Python Years

- Started working with WCDMA at the end of 2002
- Manual testing too hard
- Started learning Python
- Co-developed an automated test environment for black box testing
  - initially not concurrent
- Concurrency VITAL
  - class inheritance of an FSM
  - global mailbox for scheduling events (like AXE 10)
  - event jobs ran to completion
- Don't name a test environment NITE!

Erlang, WCDMA

- Changed jobs within WCDMA in 2005
- Old test environment replaced with Erlang
- Accused of heresy by my old python colleague!
- Soon impressed by the languages explicit expression of concurrency
- A mistake, we retained a part of the old test environment
  - a stub written in C
  - interfaces changed often
  - limited test scope possibilities
- Discovered QuickCheck

QuviQ

Erlang, LTE

- Started working with LTE in early 2007
- Introduced GTE for function test and node integration
  - G for generic, not Graham
  - white box testing
  - black box testing
  - Load testing (several processes)
  - Distributed erlang (if necessary)
- Test environment designed specifically with QuickCheck in mind
  - specification based testing
  - avoid scenario based testing

Summary

- Complex, concurrent products like Radio Base Stations need test environments that:
  - are, guess what, CONCURRENT
  - can explicitly test functionality of internal and external interfaces
    - simulate interface behaviour
    - selectively observe events
    - selectively alter interface behaviour
  - can implicitly test functionality
    - traffic models
    - statistical analysis
  - can generate test cases, test automation is only a fraction the story

ERICSSON