

## Test Driven Development In Erlang

"I am rarely happier then when spending an entire day programming my computer to perform automatically a task that it would otherwise take me a good ten seconds to do by hand. Ten seconds, I tell myself, is ten seconds. Time is valuable and ten seconds' worth of it is well worth the investment of a day's happy activity working out a way to save it".

-- Douglas Adams, *Last Chance to See*



Erlang Training and Consulting Ltd  
[www.erlang-consulting.com](http://www.erlang-consulting.com)

Martin Carlson  
[martin@erlang-consulting.com](mailto:martin@erlang-consulting.com)

© 2005 Erlang Training and Consulting Ltd

## Test Driven Development In Erlang

- Test Driven Development
- Hypothesis
- The Prototype
- Research
- Results
- Thoughts & Conclusions



# Test Driven Development

- **Extreme Programming**
  - *Test Driven Development (TDD)*
- **Used By NASA In The 1960's**
- **Reduces Errors by 40%**
  - *Published papers on Java, C and C++*
- **Accelerated Problem Resolution**
- **Test Assets**
  - *Enables Changes*

## Hypothesis

- **Is Test Driven Development language specific?**
- **Can it Increase output of correct Erlang code?**
- **Does the combination of Erlang and TDD confer other spin-off benefits?**

# The Prototype

- Test Driven Development
- Traditional - Test Last Development
- Six Modules
- OTP Test Server - Tool



# Track and Trace Requirements

- **Food Traceability**
  - *EU Requirements since January 1<sup>st</sup>, 2005*
  - *Halal Requirements*
- **Interoperability**
  - *Gateway between traceability and supply chain management*
- **Marking Technology Agnostic**
- **Certification and Verification**
- **Highly Scalable**
- **Transaction Based**
  - *Micro Transactions (pay per mark)*



# Research Approach

- **Two development cycles**
  - *Similar Complexity*
- **Same Tools And Techniques**
- **Different Methodology**
  - *Test Driven Development*
  - *Test Last Development*

# Results

- **40% Error Reduction**
  - *Forced to design your code before developing it.*
  - *Better awareness of constraints.*
  - *Interface prerequisites are addressed early.*
- **Faster Implementation**
  - *Less bugs and troubleshooting.*
  - *Code testing optimization.*
- **Faster Refactoring**
  - *Automatic predefined test suites in place.*

## Thoughts & Conclusions

- **Short test and code iterations give:**
  - *Fast feedback.*
  - *Less code to locate bugs in.*
- **Refactoring benefits from a test suite repository.**
  - *Reuse of the test suite.*
  - *Incremental improvement of the test suite.*
- **Hidden faults**
  - *Supervisor restarts mask faults.*
- **Forced to think through the architecture differently.**

## Questions ?



# Test Driven Development In Erlang

- Test Driven Development
- Hypothesis
- The Prototype
- Research
- Results
- Thoughts and Conclusions

Erlang