Why test?

Industry Regulation

Avionics: DO-178B/C
Nuclear: IEC 60880
Rail: EN 50128
Why test?

Commercial risk

- Corporate and brand reputation
- Litigation

Cost of testing

- Test code early
- Increases overall software quality
- Reduces overall test costs
Objectives

I want an easy life

Write tests in a language that I already know

Test harness that is easy to reuse across projects

Results
Where do I start?

Dumb approach: start writing a system-level test

Why won’t this work? Coverage

Intelligent approach: break down my system & test individual components
Unit Testing

Imagine your system:

- Unit A
  - Unit B
  - Unit C
  - Unit D
    - Unit E
    - Unit F
    - Unit G
      - Unit H
      - Unit I
      - Unit J
Testing in isolation

Problem: dependencies

Smart solution: Stubs / mocks
Unit Testing

Dumb approach: simply write a test for each of my components
Fairly easy to do: but what does this prove?

Intelligent approach: Start from requirements
Highlight unwanted / unnecessary code
When do I stop?

Dumb approach: when I’m bored?
Once I’ve written a test for every component?
At home time?
Intelligent approach: I need an easy way to analyse how much of my code is exercised:
Analyse Coverage

Rule sets

Drill-down views
Analyse Coverage

Intelligent highlighting

```c
// Calculate averages and share (don't divide if c and d are zero)
if (system_valid() == OK)
{
  average_a = (first_a + second_a) / 2;
  average_b = (first_b + second_b) / 2;
  if (total == 0)
    share = ERROR;
    return ERROR;
  else
    share = (double)c / total;
}
```

Routine: int values_check(int ,int ,int ,int ,unsigned short ,unsigned short ,unsigned short )
Statement: if (total == 0) { share = ERROR; ...
Increase Coverage

How do I improve my coverage?

Write more tests and hope for the best
Increase Coverage

Intelligent approach: table-driven tests

Vary input parameters
Work with large data sets
Other tools

Auto-generation of tests

Test optimisation
Are we there yet?

Analysed Coverage
Maximise Coverage
Optimised Tests
White-box testing

“...is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality”

White-box testing

Accessing static data and functions

Must be unobtrusive

```c
#include "values.h"

/* Global data*/
int checked_status;
double share;

/* Local data*/
static double average_a = 0.0;
static double average_b = 0.0;

/* Local functions */
static int low_value_check()
{
    int low_value;
    if ((average_a < 11.25) || (average_b < 15.25))
    {
        low_value = 1;
    }
    else
    {
        low_value = 0;
    }
    return low_value;
}
```
Integration Testing

Looking at how the components interact

- Unit A
- Unit B
- Unit C
- Unit D
- Unit E
- Unit F
- Unit G
- Unit H
- Unit I
- Unit J
What is wrapping?

- **Source Code**
- **Called Object**

**Wrapper**
- **BEFORE Wrapper**
- **AFTER Wrapper**

- **Check Out Parameters**
- **Modify Out Parameters**
- **Modify In Parameters**
- **Check In Parameters**
Results

Multiple report formats

- Console Summary
- Test Results Explorer (drill down to each check, links to test script)
- XML Test Report
- ASCII text results with Outline

Detailed results

- Checks list actual against expected
- Summaries by Test projects, Test cases, Individual checks
- Reports Filtered for pass/fail/warnings
Summary

Easily analyse coverage

Table-driven tests

Auto-generate

Optimise

Check static data

Wrap function calls

Results
> Free Evaluation

- Full featured product
- Full technical support
- See how leading companies do intelligent testing
Thank you for your attention!

Any Questions?