Introduction to Data Coupling and Control Coupling
Established 1975

ISO 9001 certified company

Certified for use in safety related software development according to IEC 61508, IEC 62304, EN 50128, IEC 60880 and ISO 26262

Provider of Software Quality, Compliance Management and Testing Solutions

Active participants in standards e.g. DO-178C, MISRA C/C++, FACE and CERT
LDRA Standards Experience & Pedigree

Professor Mike Hennell
- Member of SC-205 / WG-71 (DO-178C) formal methods subgroup
- Member of MISRA C committee and MISRA C++ committee
- Member of the working group drafting a proposed secureC annex for the C language definition (SC 22 / WG14)

Bill St Clair
- Member of SC-205 / WG-71 (DO-178C) Object Oriented Technology subgroup

Shan Bhattacharya
- Member of FACE Consortium Technical Working Group Conformance Verification Matrix Subcommittee
- Member of FACE Consortium Integration Workshop Standing Committee

Dr Clive Pygott
- Member of ISO software vulnerabilities working group (SC 22 / WG 23)
- Member of MISRA C++ committee
- Member of the working group drafting a proposed secureC annex for the C language definition (SC 22 / WG14)

Liz Whiting
- Member of MISRA C committee language definition (WG14)

Chris Tapp
- Chairman of MISRA C++ committee
- Member of MISRA C committee language definition (WG14)
• DO-178C Overview

Delivering Software Quality and Security through Test, Analysis and Requirements Traceability
Linking Requirements, Code and Tests

Traceability from Requirements through Code and Tests
Traceability Across DAL Levels

System requirements allocated to Software

High-Level Requirements

Low-Level Requirements

Source Code

Executable Object Code

Review and Analysis Results

SW Architecture

Test Results

Test Procedures

Test Cases

Levels A - D

Levels A - C

Level A
Reviewing requirements and traceability data

The software shall allow the user to exit the application.

The software shall offer the user an input option to exit the application from the user input display using the letter 'q' for quit.
Mapping low-level requirements to source code

- Ensure low-level requirements are implemented
- Identifies implemented functionality missing in requirements
- Key to structural coverage analysis
- Correlates expected behavior and/or design details to test design and test measurement
Safety and Security with Coding Standards

Code

```c
int32_t a[10];
uint32_t i;
for (i = 0; i < 20; ++i) {
    a[i] = 0;
}
```

Result

Depending on the runtime environment (OS, etc), this will result in an exception or overwrite unrelated memory.

Pre-empt security and reliability issues early in the life-cycle

- Reduce cost of verification early in the process by developing more verifiable software
- Eliminate defects statically, either by pre-empting or detection is vastly cheaper
- Consistent coding style and form across teams
- Portability & reusability across varying architecture and compiler/tool chains

Security vs Reliability

- Standards families such as MISRA focus on reliability
- CERT C / CWE were developed to address security vulnerabilities
- Both categories of standards can be used in conjunction
- Tailorable for new development, legacy code, and runtime error checking
Considering Traceability Scenarios

Identifying potential traceability issues

- Improper decomposition
- The all encompassing requirement
- Many to one/few could reveal poor traceability analysis
Developing and documenting test cases

- Must be linked to requirements they verify
- Test case details must be reviewed against requirements
- Function testing at the system or integration scope can be managed from TBmanager
Measure Structural Coverage

Measure of Test Effectiveness

- Coverage through requirements based tests cases (functional and low-level)
- Identifies unreachable/infeasible code and gaps in requirements and test design
What is Structural Coverage?

Measurement of Test Effectiveness

- How effectively did tests exercise code?
- Exercised, entry points, statements, branches, compound conditionals, execution paths
- Systems requirement reliability levels up with one defect per $10^9$ operating hours
- Metric that helps determine when a system is adequately tested

Structural Coverage is Often Mandated

- DO-178B/C, DO-278(A) for Commercial/Defense avionics and ground systems
- IEC 61508 for industrial controls
- ISO 26262 for automotive
- IEC 62304 for medical devices
- EN 50128 for rail
- Company based standards (in-house)
Depending on the SIL or DAL level and functional safety standard being followed, coverage requirements and required methodology varies:

- Statement Coverage
- Branch Decision Coverage
- Modified Condition / Decision Coverage (MC/DC)
- Data Coupling and Control Coupling Coverage
- Object Code Coverage
- Linear Code Sequence And Jump Coverage – Test Path (LCSAJ)
Developing and documenting low-level test cases

- Must be linked to requirements they verify
- Test case details must be reviewed against requirements
- Typically written at the function interface
Unit Testing – Host / Target

Testing at the function interface

- Achieve structural coverage objectives
- Verify low level requirements
Why Unit Testing

Why Unit test?

- Allows you to find faults earlier
- Allows you to characterize the behavior of components as you develop – and make sure they behave the way they are supposed to
- Allows you to verify components before they are put into systems
- Gives you confidence the system will work correctly when it all fits together

Why Unit testing with structural coverage?

- Entry point, Statement, Branch, MC/DC, Path, assembly level coverage
- Provides a basis to know you have adequately tested
```c
uint32_t pulse=0U;
/* (M) DATAFLOW VIOLATION : 70 D : DU anomaly, variable value is not used. : pulse : 40F */
/* See also line 17 Dataflow.c(DATAFLOW) */
uint32_t iter;

if (cycles > 0U)
{
  for (iter=0U; iter<cycles; iter++)
  {
    pulse++;
    /* (M) DATAFLOW VIOLATION : 70 D : DU anomaly, variable value is not used. : pulse : 57T */
    /* See also line 17 Dataflow.c(DATAFLOW) */
  }
}
/* (O) DATAFLOW VIOLATION : 7 D : DU data flow anomalies found. */
```
• Data Coupling and Control Coupling
DO-178B A7-8 points to section 6.4.4.2c, states
• “The analysis should confirm the data coupling and control coupling between the code components.”

DO-178C section 6.4.4.2c states
• “Analysis to confirm that the requirements-based testing has exercised the data and control coupling between code components”

Analytical exercise vs measurement exercise
• The shift in emphasis from confirm the coupling (DO-178B) to confirm the exercising of the coupling (DO-178C) changes the DCCC objective from an analytical exercise against the test design to a measurement exercise against the test execution. This was a specific topic of conversation when the committee discussed this issue.

Confirming coupling
• Confirm control coupling by reviewing procedure call coverage achieved by requirements based tests across software component boundaries
• Confirm data coupling by reviewing dynamic data flow coverage report generated from requirements based tests
Two of the files contain a definition of a function foo

- May resolve all calls to definition in file A
- Or calls in File A to file A, calls in file B to file B, and calls in file C to either

Control Coupling defect as user may be unaware of ambiguity
Weak symbols are often useful in defining library functions that can be overridden

• GCC supports for Elf targets

In the example above one of the two functions will be linked

The symbol that is linked is decided arbitrarily

• Essentially picks up the symbol that is “closest”
  • Procedure call coverage is the ideal way to ensure the right instance is linked.

https://gcc.gnu.org/onlinedocs/gcc/Weak-Pragmas.html

#pragma weak symbol
This pragma declares symbol to be weak, as if the declaration had the attribute of the same name. The pragma may appear before or after the declaration of symbol. It is not an error for symbol to never be defined at all
#pragma weak symbol1 = symbol2
This pragma declares symbol1 to be a weak alias of symbol2. It is an error if symbol2 is not defined in the current translation unit.
C++ Challenges and potential Linking Issues

Inline function defined in a class with two different implementation (Cpp example)

- Arbitrary and essentially picks up the symbol that is “closest”
- Procedure call coverage is the ideal way to ensure the right instance is linked.
Control Coupling Example – Procedure Call Coverage (CC2)

- Control Coupling requires that all potential calls be executed at each point.
## CC2 Data Sets

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Var value</th>
<th>Glob value</th>
<th>Call Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set 1</td>
<td>0</td>
<td>1</td>
<td>foo ( &amp;func1) at case 2</td>
</tr>
<tr>
<td>Data Set 2</td>
<td>1</td>
<td>0</td>
<td>foo ( &amp;func2) at case 1</td>
</tr>
<tr>
<td>Data Set 3</td>
<td>0</td>
<td>0</td>
<td>foo ( &amp;func2) at case 2</td>
</tr>
<tr>
<td>Data Set 4</td>
<td>1</td>
<td>1</td>
<td>foo ( &amp;func1) at case 1</td>
</tr>
</tbody>
</table>

```c
30 int main ( void ){
    loop:
    get ( var ); get ( glob ); get(exit_var);
    if( glob == 1 ){
        foo( &func1 );
    } else{
        foo( &func2 );
    }
}
```

```c
20 void foo ( void (*pfunc)(void) ){
    if( var == 1 ) {
        pfunc();  /* case 1 */
        printf("\ncase 1\n");
    } else {
        pfunc();  /* case 2 */
        printf("\ncase 2\n");
    }
}
```
CC2 Data Sets

- Streaming information captured as code executes
CC1 Data Sets 1 and 2

Data Sets 1 and 2 yield 100% Statement and Branch/Decision Coverage

Less than 100% Procedure Call Coverage Achieved
Visualising Procedure Call Coverage

- Function foo has 100% Statement and Branch/Decision Coverage but all potential calls have not been executed at each point
- Dashed Red arrows from main to func1 and func2 indicate incomplete Procedure Call Coverage
CC1 Data Sets 3 and 4

Data Set 3

Data Set 4

- Data Sets 3 and 4 yield 100% Statement and Branch/Decision
- Less than 100% Procedure Call Coverage achieved
All Data Sets Combined (1 – 4)

- Control Coupling requires that all potential calls be executed at each point
- 100% Statement, Branch/Decision, and Procedure Call Coverage achieved
• Function foo has 100% Statement and Branch/Decision Coverage
• Callgraph arrows indicate 100% Procedure Call Coverage has been met
Data Coupling Analysis Scenarios and Examples

Supporting Information for DO-178C and DO-278A

RTCA DO-248C
December 13, 2011
Prepared by: SC-205
© 2011 RTCA, Inc.
Data Coupling Analysis by Test Case Example

- CalculateAirspeed and DisplayAirspeed are both invoked by runAirspeedCommand
- All three are in different files and represent different software components
- Test cases are created to verify commands are being set and achieve structural coverage

```c
static S_U32 airspeed;
#define CALCULATE_CMD 1
#define DISPLAY_CMD 2

/*
 * Data Coupling defects:
 * 1) It is possible to call displayAirspeed (a use operation of 'airspeed')
 *    before calculateAirspeed (a set operation of 'airspeed').
 *    Demonstrated with Test Case 1
 * 2) It is possible to call calculateAirspeed without a subsequent call to displayAirspeed
 *    Demonstrated with Test Case 2
 */

void runAirspeedCommand (S_U16 command)
{
    switch(command)
    {
        case CALCULATE_CMD:
            calculateAirspeed (airspeed);
            break;
        case DISPLAY_CMD:
            displayAirspeed (airspeed);
            break;
    }
}
```

The requirements state that CALCULATE_CMD is called before DISPLAY_CMD every time.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Procedure</th>
<th>Regression P/F</th>
<th>Name / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc 1</td>
<td>runAirspeedCommand</td>
<td>PASS</td>
<td>Invoke DISPLAY_CMD so that the current value of airspeed is set before calculation.</td>
</tr>
<tr>
<td>Tc 2</td>
<td>runAirspeedCommand</td>
<td>PASS</td>
<td>Invoke CALCULATE_CMD so that the airspeed is calculated before display.</td>
</tr>
<tr>
<td>Tc 3</td>
<td>runAirspeedCommand</td>
<td>PASS</td>
<td>Invoke a command value outside of DISPLAY_CMD and CALCULATE_CMD for verification.</td>
</tr>
</tbody>
</table>
In order to get 100% statement and 100% decision coverage of the “C” code, we needed to create three test cases that verify the requirement.
Statement, Branch, and Data Coverage Achieved

In aggregate the Dynamic Data Flow Coverage Report shows all data elements have been read and written to as expected.
DDFC By Test Case Reveals Control Flow Issues

Test case

Unexecuted code for the given test case

Unexecuted data reference for the given test case

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Call Depth / Parameter Name</th>
<th>File</th>
<th>Procedure</th>
<th>Type Code</th>
<th>Attribute Code</th>
<th>Used on lines...</th>
</tr>
</thead>
<tbody>
<tr>
<td>airspeed</td>
<td>Alias</td>
<td>AirspeedCommands.cpp</td>
<td>runAirspeedCommand</td>
<td>G</td>
<td>R</td>
<td>30 ****</td>
</tr>
<tr>
<td>command</td>
<td></td>
<td>AirspeedCommands.cpp</td>
<td>runAirspeedCommand</td>
<td>G</td>
<td>R</td>
<td>36</td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td>AirspeedCalculate.cpp</td>
<td>calculateAirspeed</td>
<td>G</td>
<td>R</td>
<td>31</td>
</tr>
</tbody>
</table>

On line 39 the reference to airspeed by displayAirspeed is not executed with this test case.
DDFC By Test Case Reveals Control Flow Issues

Test case

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY_CMD</td>
<td>command</td>
<td>S_U16</td>
</tr>
<tr>
<td>*** Value Retained ***</td>
<td>airspeed</td>
<td>S_U32</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unexecuted code for the given test case

```c
void runAirspeedCommand (S_U16 command)
{
    switch (command)
    {
        case CALCULATE_CMD:
            calculateAirspeed (airspeed);
            break;
        case DISPLAY_CMD:
            displayAirspeed (airspeed);
            break;
    }
}
```

Unexecuted data reference for the given test case

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Call Depth / Parameter Name</th>
<th>File</th>
<th>Procedure</th>
<th>Type Code</th>
<th>Attribute Code</th>
<th>Used on lines...</th>
</tr>
</thead>
<tbody>
<tr>
<td>airspeed</td>
<td></td>
<td>AirspeedCommands.cpp</td>
<td>runAirspeedCommand</td>
<td>G</td>
<td>R</td>
<td>36 *****</td>
</tr>
<tr>
<td>command</td>
<td></td>
<td></td>
<td></td>
<td>G</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>R</td>
<td>16 *****</td>
</tr>
</tbody>
</table>

On line 36 the define of airspeed by calculateAirspeed is not executed with this test case
DO-178C Text

• “Analysis to confirm that the requirements-based testing has exercised the data and control coupling between code components”
• “Test Coverage of software structure, both data and control coupling, is achieved”
• A measurement exercise against the test execution

A Form of Test Measurement

• Analogous to structural coverage but focus on data set and usage

Scope and Granularity

• Measured in aggregate or by test case
• Measured after execution of high level and/or low-level tests
### Global Variable Usage Summary

<table>
<thead>
<tr>
<th>Name</th>
<th>File</th>
<th>Attrs</th>
<th>Used On Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumZones</td>
<td>CELL.CPP</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSTEMDATA.CPP</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TUNNEL.CPP</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>NumZoneParams</td>
<td>CELL.CPP</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSTEMDATA.CPP</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TUNNEL.CPP</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>NumSystemParams</td>
<td>CELL.CPP</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSTEMDATA.CPP</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TUNNEL.CPP</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>NumLampTypes</td>
<td>CELL.CPP</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSTEMDATA.CPP</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TUNNEL.CPP</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>LampsPerLuminaire</td>
<td>CELL.CPP</td>
<td>G</td>
<td></td>
</tr>
</tbody>
</table>

- Generated global variables usage data can be used to filter DC/CC artifacts
- Results can be analysed in the context of requirements based tests
Data Coupling for Parametric Globals

Variable Fan-in/Fan-out Pairs

Variable Fan-in/Fan-out coverage (3 Sets and 3 Use)

- Execute every Set of the Set-Call list and every Use in the Set-Use list
- Variable Fan-in/Fan-out coverage is practically approachable much like MC/DC coverage (n+1 number of test cases)

Review of Fan-in/Fan-out Coverage in the context of requirements based tests to meet objective A-7.8
• Test Case 1
  • Glob Var is passed in call_setter to param_setter1 for setting
  • Glob Var is set in param_setter1

• Test Case 2
  • Glob Var is passed in call_user to param_use1 and is referenced
  • Glob Var is set in param_setter1

• Variable Fan-in/Fan-out Coverage of parametric global glob_var
  • By reference for set and by value for use
  • NOTE: Add test cases in hidden slide
**Parametric Global**

- Execution of the six text cases above achieve 100% Variable Fan-In/Fan-Out Coverage

- **Glob_var** being set in param_setter (1..3), the Set-Call list

- **Glob_var** being used in param_use (1..3), the Set-Use list
Data Coupling for Parametric Globals

Glob Set 1
Glob Set 2
Glob Set 3

Glob Passed as a parameter

Glob Use 1
Glob Use 2
Glob Use 3

Set-Use combinations

<table>
<thead>
<tr>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use 1</td>
<td>Use 1</td>
<td>Use 1</td>
<td>Use 2</td>
<td>Use 2</td>
<td>Use 2</td>
<td>Use 3</td>
<td>Use 3</td>
<td>Use 3</td>
</tr>
</tbody>
</table>

Set-Use combinations (9 Set and 9 Use)

- Ensure that all combinations of the Set-Use pairs in these lists are executed (9 in total)
- Set-Use combinations result in a combinatorial explosion much like Branch Combination Coverage (2^n number of test cases)
The Benefits of Using LDRA Tools for DDFC

Dramatic reduction of time necessary for DCCC analysis

Clear, repeatable, methodology for DCCC that has been reviewed and accepted by DERs
- Reduces risks of methodology ambiguities during SOI audits
- Consistent with the expectations DO-178C as a test measurement exercise

Defined artifact set for archival and review

By utilising a DO-178C harmonised DDFC qualification package, the review burden for this process is vastly reduced

Reduced cost of DDCC activities during incremental releases
Question: From Certification perspective, what are the artifacts to be produced as an evidence to the A-7.8 Data Coupling and Control Coupling objective?

- **Data Coupling** - Dynamic Data Flow Coverage Report and methodology as described in the software verification plan (SVP), showing that results are reviewed against requirements based tests to ensure global and parametric variable set/use pairs are clearly understood and exercised.

- **Control Coupling** - Procedure Call Coverage Report – Ensuring 100% procedure call coverage. Review of the procedure call coverage report:
  - To ensure the linking phase has resolved all calls correctly. Potential duplicate definitions and other linking issues should be considered
  - Ensure all potential calls are executed at each point. Function pointers can be particularly problematic.
Question: If one achieves 100% Test Coverage for High Level Requirements and Low Level Requirements, and 100% Structural Coverage for Statement and Branch/Decision Coverage from Requirements Based Tests, is this not sufficient to take credit for A-7.8?

- Data Coupling – “Test coverage of software structure, both data coupling and control coupling…” (6.4.4 d), requires a test measurement of data coupling. 100% Structural Coverage Data doesn’t ensure that all relevant set/use pairs of data elements are exercised per requirements, especially global variables passed as parameters and “used” further downstream in the call tree.

- Additionally 100% Statement and Branch/Decision coverage doesn’t necessarily imply 100% Procedure Call Coverage, which is necessary to ensure that all potential calls are executed at each point.

- The specifics of the methodology used to meet objective A-7.8 should be documented in the SVP, and typically communicated and agreed upon up front with your DER.
Question: Did you present your approach to any FAA/EASA or Certification DER representative, if so, what was their feedback?

- Feedback has been consistently positive. This is an area where tooling was lacking when DO-178B was written and this resulted in a lot of manual effort and confusion. DO-178C’s changes, more clearly reflect the committees original intentions. When the Dynamic data flow coverage report has been shown to DERs, ACOs, and other FAA/EASA leadership, LDRA’s has been told that this type of transparent reporting of test measurement of data definitions and references (set/use operations) aligns very well with the intent of the standard. Additionally we received many comments that this approach and technology saves both the applicant and DER significant time and effort to meet the objective. The qualification package was added to cost effectively qualify the DC/CC analysis and reporting produced by the LDRA tool suite and further reduce the review effort required.
Are there any Questions?
For further information:

www.ldra.com   info@ldra.com

@ldra_technology   LDRA Software Technology   LDRA Limited