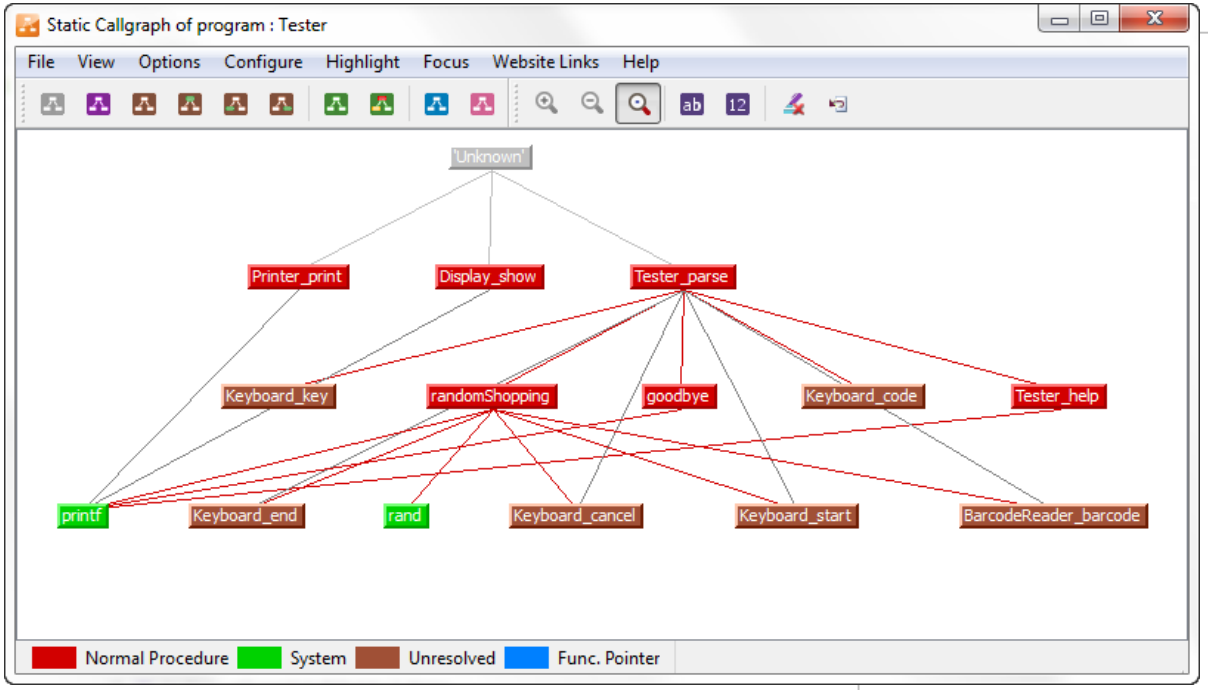
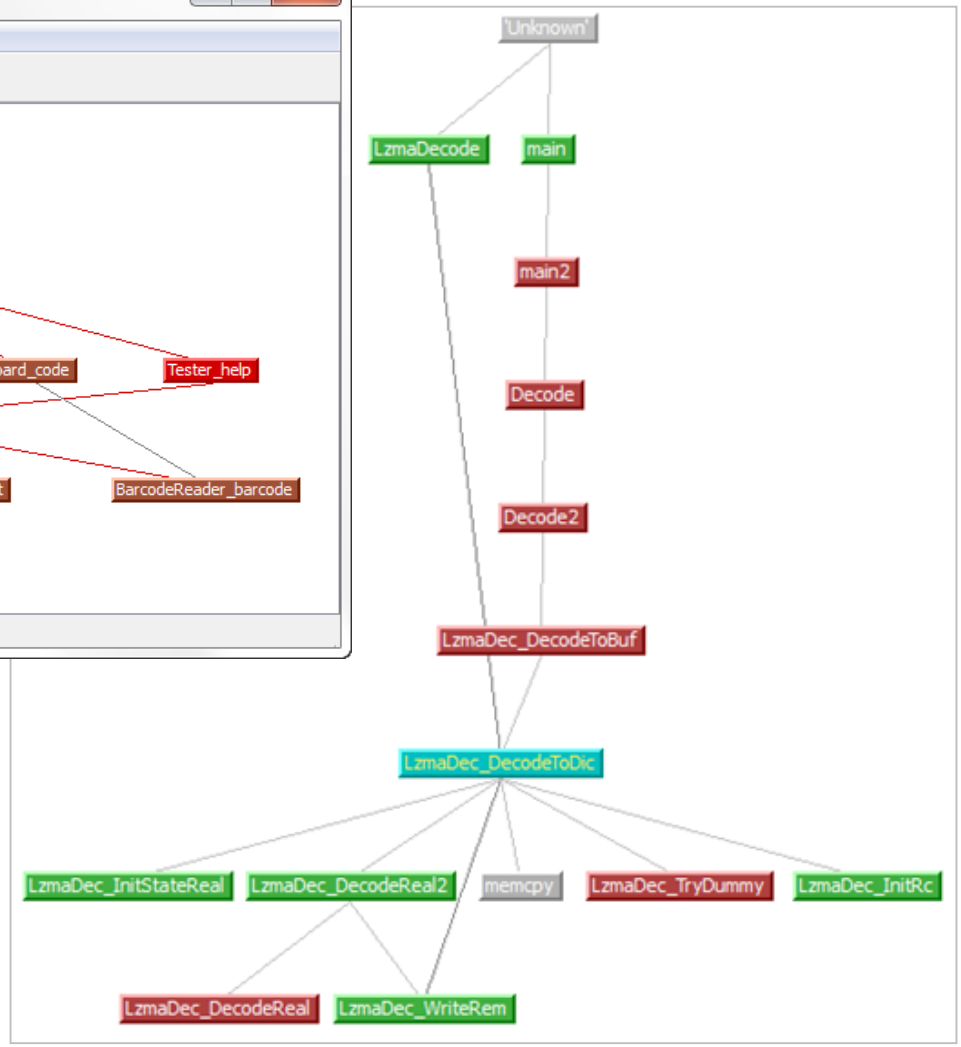
Performing Data and Control Coupling Analysis with the LDRA tool suite

Control Flow Analysis – Tbvision/Tbsafe/TBmanager





Data Flow Analysis – Tbvision/Tbsafe/TBmanager

- How is data used?

```
169  #pragma vector=ADC12_VECTOR
170  __interrupt void ADC12ISR (void)
171  {
172    switch(__even_in_range(ADC12IV,34))
173    {
174    case  0: break;
175    case  2: break;
176    case  4: break;
177    case  6:
178          adc12_result = ADC12MEM0;
179          adc12_data_ready = 1;
180          _BIC_SR_IRQ(LPM3_bits);
181          break;
182    case  8: break;
```

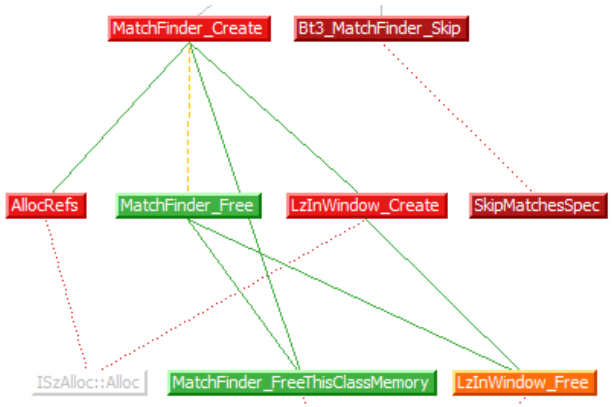
adc12_result (Pri)		u16	adc12.c	<Global scope>	G	E	60				
			adc12.h/adc12.c		G	E	62				
				ADC12ISR	G	D	178				
				adc12_single_conversion	G	R	157				

Data and Control Coupling Coverage - TBsafe

- DO-178C section 6.4.4.2 c states:

  "Analysis to confirm that the requirements-based testing has exercised the data and control coupling between code components"

- Control coupling **coverage** is ensuring that every invocation of a function has been exercised



- Data coupling **coverage** is ensuring that we have exercised every access to the data

(p)->cyclicBufferSize		MatchFinder_Init	P	R	278	
		MatchFinder_SetLimits	P	R	249	
(p)->dic	LzmaDec.c	LzmaDec_Allocate	P	R	954	958
			P	D	957	
		LzmaDec_DecodeReal	P	R	141	178
(p)->dic		LzmaDec_DecodeToBuf	P	R	869	
		LzmaDec_FreeDict	P	R	888	
			P	D	889	
		LzmaDec_TryDummy	P	R	512	522
		LzmaDec_WriteRem	P	R	432	
(p)->dicBufSize	LzmaDec.c	LzmaDec_Allocate	P	R	954 *****	
			P	D	964	
		LzmaDec_DecodeReal	P	R	142	

Data Coupling Coverage - TBsafe

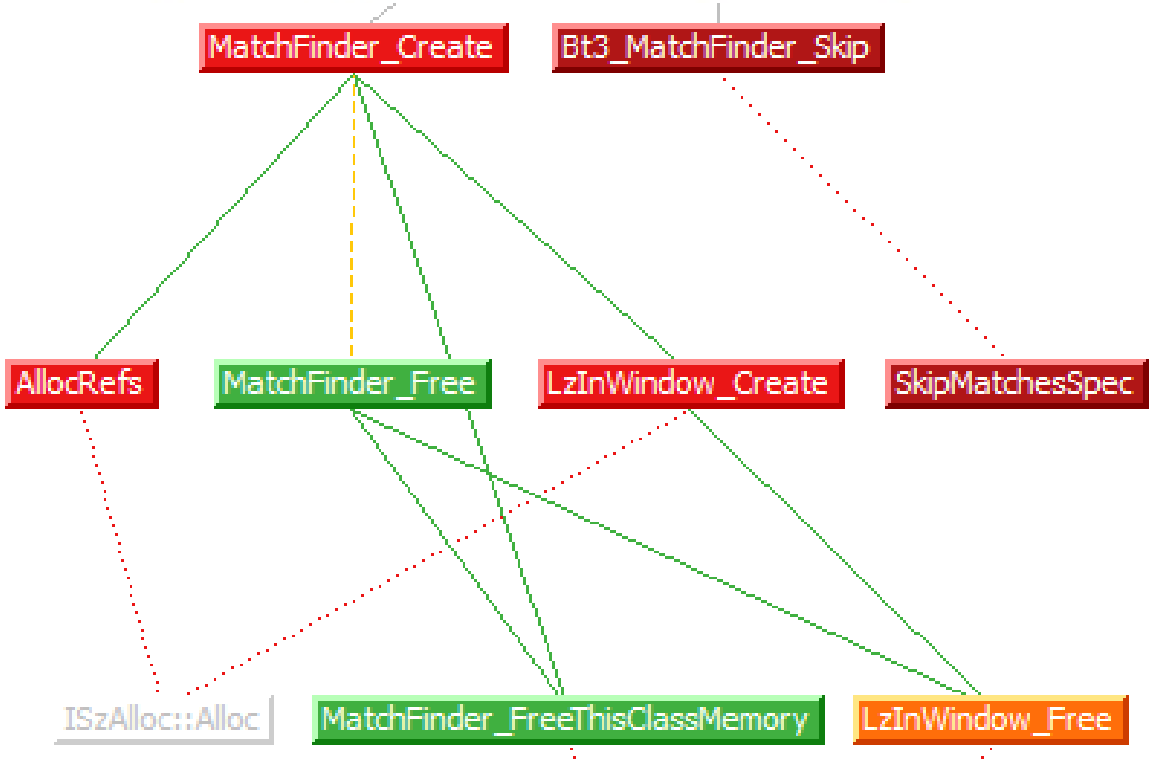
- Table A-7 5 - Test coverage of software structure (modified condition/decision coverage) is achieved - Unfulfilled
- Table A-7 6 - Test coverage of software structure (decision coverage) is achieved - Unfulfilled
- Table A-7 7 - Test coverage of software structure (statement coverage) is achieved - Unfulfilled
- Table A-7 8 - Test coverage of software structure (data coupling and control coupling) is achieved - Fulfilled - 2 assets
  - Software Verification Results fulfilled by 2 items
    - Callgraph - Pass/Fail Coverage
    - DataCouplingReport.html
- Table A-7 9 - Verification of additional code that can not be traced to Source Code, is achieved - Unfulfilled

data access not exercised!

	Call Depth / Parameter Name										
Variable Name	**Alias**	**File**	**Procedure**	**Type Code**	**Attribute Code**	**Used on lines...**					
OutBound.itsIException		Calculator.cpp	Calculator::calc_C::setItsIException	P	D						
	1 itsIException	Calculator.cpp	Calculator::calc_C::OutBound_C::setItsIExceptio	P	D	118					
_p_		Calculator.cpp	Calculator::calc_C::calc_C	P	D	128					
		Tester.cpp	Tester::calc_C::calc_C	P	D	130					
arg		TesterBuilder.cpp	TesterBuilder::cancelTimeout	P	E	58					
				P	R	60 *****					
arg1		Calculator.cpp	Calculator::add	P	E	177					
				P	R	180					
arg1			Calculator::calc_C::InBound_C::add	P	E	26					
				P	R	29					
	1 arg1		Calculator::add	P	E	177					
				P	R	180					
arg1			Calculator::calc_C::InBound_C::divide	P	E	34					
				P	R	37					

Control Coupling Coverage - TBsafe

- Table A-7 5 - Test coverage of software structure (modified condition/decision coverage) is achieved - Unfulfilled
- Table A-7 6 - Test coverage of software structure (decision coverage) is achieved - Unfulfilled
- Table A-7 7 - Test coverage of software structure (statement coverage) is achieved - Unfulfilled
- Table A-7 8 - Test coverage of software structure (data coupling and control coupling) is achieved - Fulfilled - 2 assets
  - Software Verification Results fulfilled by 2 items
    - Callgraph - Pass/Fail Coverage
    - DataCouplingReport.html
- Table A-7 9 - Verification of additional code that can not be traced to Source Code, is achieved - Unfulfilled



Data and Control Coupling Coverage Expanded

LDRA

- DO-178C section 6.4.4.2 c states:

  "Analysis to confirm that the requirements-based testing has exercised the data and control coupling between code components"

- Confirm control coupling by reviewing procedure call coverage achieved by requirements based tests across software component boundaries, with the requirements that are implemented in the components
- Confirm data coupling by reviewing dynamic data flow coverage report generated from requirements based tests, with the requirements implemented by the components

Calculate and Display Airspeed Example

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

```
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 states that CALCULATE_CMD is called before DISPLAY_CMD every time

Test Case View

Test Case	Procedure	Regression P / F	Name / Description
Tc 1	runAirspeedCommand	PASS	Inovke DISPLAY_CMD so that the current value of airs...
Tc 2	runAirspeedCommand	PASS	Inovke CALCULATE_CMD so that the airspeed is calc...
Tc 3	runAirspeedCommand	PASS	Inovke a command value outside of DISPLAY_CMD an...

Testing Requirements to Achieve Coverage

- 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

Variable I/O View

Value	Name	Type
CALCULATE_CMD	command	S_U16
*** Value Retained ***	airspeed	S_U32
0	airspeed	S_U32

Variable I/O View

Value	Name	Type
DISPLAY_CMD	command	S_U16
*** Value Retained ***	airspeed	S_U32

Variable I/O View

Value	Name	Type
*** Value Retained ***	airspeed	S_U32
0	command	S_U16
0	airspeed	S_U32

AirspeedCommands.cpp
- Statement Coverage - Current - 100% - Combined - 100%
- Branch Decision Coverage - Current - 100% - Combined - 100%
- LCSAJ Coverage - Current - 100% - Combined - 100%
- Global Variables
- runAirspeedCommand - void - Combined - S 100% - B 100% - L 100%

Test Case View

Test Case	Procedure	Regression P / F	Name / Description
Tc 1	runAirspeedCommand	PASS	Inovke DISPLAY_CMD so that the current value of airs...
Tc 2	runAirspeedCommand	PASS	Inovke CALCULATE_CMD so that the airspeed is calc...
Tc 3	runAirspeedCommand	PASS	Inovke a command value outside of DISPLAY_CMD an...

Statement, Branch, and Data Coverage Acheived

LINE NUMBER REF. (SOURCE)	STATEMENT	PREVIOUS RUNS	CURRENT RUN	COMBINED
94 (31)	void	-	-	-
95	runAirspeedCommand (	6	3	9
96	S_U16 command)	-	-	-
97 (32)	{	-	-	-
98 (33)	switch (	6	3	9
99				
100				
101 (34)				
102 (35)				
103 (36)				
104 (37)				
105 (38)				
106 (39)				
107 (40)				
108 (41)				
109 (42)	}			

LINE NUMBERS: FROM	REFORMATTED (SOURCE) TO	PREVIOUS RUNS	CURRENT RUN	COMBINED	CODE PRECEDING DECISION POINT
101 (34)	102 (35)	2	1	3	command) {
101 (34)	105 (38)	2	1	3	
101 (34)	109 (42)	2	1	3	
104 (37)	109 (42)	2	1	3	break ;
107 (40)	109 (42)	2	1	3	break ;

	Call Depth / Parameter Name					
Variable Name	Alias	File	Procedure	Type Code	Attribute Code	Used on lines...
airspeed		AirspeedCommands.cpp	runAirspeedCommand	G	R	39
				G	D Definition	36
command		AirspeedCommands.cpp	runAirspeedCommand	P	E	31
				P	R	33
factor		AirspeedCalculate.cpp	calculateAirspeed	G	R	16
sensorReading		AirspeedCalculate.cpp	calculate		R	16
speed		AirspeedCalculate.cpp	calcul			14
						16
		AirspeedDisplay.cpp	displ			12
						14
						14

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

LDRA

Test case

Variable I/O View

Value	Name	Type
CALCULATE_CMD	command	S_U16
*** Value Retained ***	airspeed	S_U32
0	airspeed	S_U32

Unexecuted code for the given test case

```
31  void runAirspeedCommand  (S_U16 command)
32  {
33      switch(command)
34      {
35          case CALCULATE_CMD:
36          calculateAirspeed (airspeed);
37          break;
38          case DISPLAY_CMD:
39          displayAirspeed (airspeed);
40          break;
41      }
42  }
```

```
void
 runAirspeedCommand (
 S_U16 command )
 {
  switch (
   command
   )
   {
   case 1 :
    calculateAirspeed ( airspeed ) ;
    break ;
   case 2 :
    displayAirspeed ( airspeed ) ;
    break ;
   }
 }
```

Unexecuted data reference for the given test case

	Call Depth / Parameter Name					
Variable Name	**Alias**	**File**	**Procedure**	**Type Code**	**Attribute Code**	**Used on lines...**
airspeed		AirspeedCommands.cpp	runAirspeedCommand	G	R	39 *****
						36
command		AirspeedCommands.cpp	runAirspeedCo			31
						33
factor		AirspeedCalculate.cpp	calculateAirspeed			16

On line 39 the reference to airspeed by displayAirspeed is not executed with this test case

DDFC by test case reveals control flow issues

LDRA

Test case

Variable I/O View

Value	Name	Type
I DISPLAY_CMD	command	S_U16
I *** Value Retained ***	airspeed	S_U32
O 0	airspeed	S_U32

Unexecuted code for the given test case

```
31  void runAirspeedCommand  (S_U16 command)
32  {
33      switch(command)
34      {
35          case CALCULATE_CMD:
36          calculateAirspeed (airspeed);
37          break;
38          case DISPLAY_CMD:
39          displayAirspeed (airspeed);
40          break;
41      }
42  }
```

```
void
  runAirspeedCommand (
  S_U16 command )
  {
    switch (
      command
      )
      {
      case 1 :
        calculateAirspeed ( airspeed ) ;
        break ;
      case 2 :
        displayAirspeed ( airspeed ) ;
        break ;
    }
  }
```

Unexecuted data reference for the given test case

	Call Depth / Parameter Name					
Variable Name	Alias	File	Procedure	Type Code	Attribute Code	Used on lines...
airspeed		AirspeedCommands.cpp	runAirspeedCommand	G	R	39
				G	D	36 *****
command					E	31
				P	R	33
factor				G	R	16 *****

On line 36 the define of airspeed by calculateeAirspeed is not executed with this test case

Additional methodology notes

- DCCC analysis and methods described here can be done at any scope
  - Within TBrun with functions that effect data flow and control flow at the boundary of software components
  - At a higher level using dynamic data flow coverage and structural coverage data against high level tests
- By utilising a DO-178C harmonized DDFC qualification package, the review burden for this process is dramatically reduced
- Various other DCCC scenarios come up. Some of them are discussed in a provided white paper written by Professor Mike Hennell

The Benefits of using LDRA tools for DDFC

- Dramatic reduction of time necessary for DDCC analysis
- Clear, repeatable, methodology for DDCC that has been reviewed and accepted by DERs
  - Reduces risks of methodology ambiguities during SOI audits
  - Consistent with the expectations DO-178 C as a test measurement exercise
- Defined artifact set for archival and review
- Dramatically reduced cost of DDCC activities during incremental releases

For further information:

www.ldra.com info@ldra.com

@ldra_technology LDRA Software Technology LDRA Limited