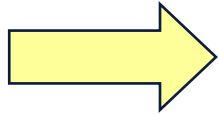


# Erhellendes und Erschreckendes zur Codeüberdeckungsmessung

Frank Büchner, Hitex GmbH, Karlsruhe





## Code Coverage Introduction

### Coverage In Standards

- Entry Point Coverage

- Statement Coverage

- Branch Coverage

- Decision Coverage

- Condition Coverage (MC/DC)

### Advanced Topics

- Calculating MC/DC

- Calculating Branch Coverage

- Names

- Process Step

- Calculations

- Omissions

- Test Cases from the Code

- Test Case Quality

# Code Coverage Introduction

*Caution: Not valid for all items!*

$$\text{Code coverage} = \frac{\text{Number of items exercised}}{\text{Total number of items}}$$

Code coverage = Structural coverage

# Tool for Reference → TESSY

## ■ Unit / module / integration testing of embedded software

The screenshot displays the TESSY software interface for a project named 'Hitex-E1'. The interface includes a menu bar (File, Edit, Window, Help), a toolbar, and several panes:

- Test Project:** A tree view showing the project structure with folders like 'Examples' and 'Experiments', and files like 'ABCD\_fo', 'Impr', 'func', 'func1', 'func2', 'ABD', and 'Assignm'.
- func1:** A central pane showing a flowchart diagram for the 'func1' function. The diagram includes a decision diamond and several process boxes.
- MC/DC Coverage:** A table showing coverage data for the 'func1' function. The table has columns 'a', 'b', and 'Teststep'.
- Branch (C1) Coverage:** A table showing branch coverage data for the 'func1' function. The table has columns 'Test Case / Test ...' and 'Coverage'.
- Assignment-to-decision.c:** A code editor showing the source code for the 'func1' function.

The 'MC/DC Coverage' table shows the following data:

a	b	Teststep
0	0	
0	1	
1	-	1.1

The 'Branch (C1) Coverage' table shows the following data:

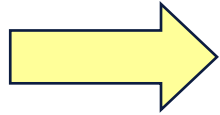
Test Case / Test ...	Coverage
1 (1)	50.00 %

The 'Assignment-to-decision.c' code editor shows the following code:

```

7
8 decision = a || b;
9 if (decision)
10     ret_val = 1;
11 else
12     ret_val = 0;
    
```

The status bar at the bottom indicates the project root is 'C:\Projects\Tessy3\Hitex-E1' and the configuration file is '\$(PROJECTROOT)\tessy\config\E1\_configuration.xml'.



## Code Coverage Introduction

### Coverage In Standards

- Entry Point Coverage

- Statement Coverage

- Branch Coverage

- Decision Coverage

- Condition Coverage (MC/DC)

### Advanced Topics

- Calculating MC/DC

- Calculating Branch Coverage

- Names

- Process Step

- Calculations

- Omissions

- Test Cases from the Code

- Test Case Quality

# Coverage In Standards

	IEC 61508	ISO 26262	DO-178
Entry Point Coverage	X		
Statement Coverage	X	X	X
Branch Coverage	X	X	
Decision Coverage			X
Modified Condition / Decision Coverage (MC/DC)	X	X	X

# Entry Point Coverage

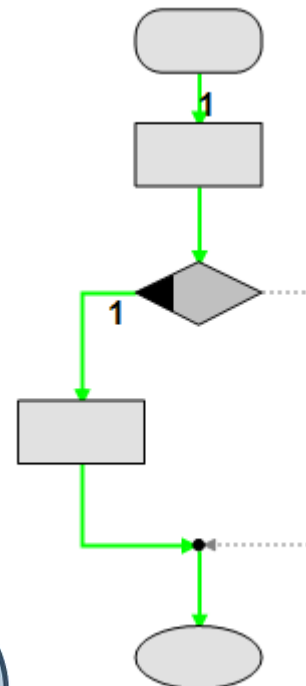
- In C, functions are entry points (“Eingangspunkte”)

No.	Name	EPC	Test Cases	Result
	Hitex-E1	90.9 %	2 of 11 failed	✗
	Hitex-Examples	90.9 %	2 of 11 failed	✗
	Coverage	90.9 %	2 of 11 failed	✗
	Examples	90.9 %	2 of 11 failed	✗
	EntryPoint	90.9 %	2 of 11 failed	✗
	Folder1	88.88 %	2 of 9 failed	✗
	A+B	83.33 %	1 of 6 failed	✗
1	f1A	-	1 of 1 passed	✓
2	f2A	-	-	⚠
3	f3A	-	1 of 1 passed	✓
4	f4B	-	1 of 1 passed	✓
5	f5B	-	1 of 1 failed	✗
6	f6B	-	1 of 1 passed	✓
	C	100 %	1 of 3 failed	✗
7	f7C	-	1 of 1 failed	✗
8	f8C	-	1 of 1 passed	✓
9	f9C	-	1 of 1 passed	✓
	Folder2	100 %	2 of 2 passed	✓
	D	100 %	2 of 2 passed	✓
10	f10D	-	1 of 1 passed	✓
11	f11D	-	1 of 1 passed	✓

# Statement Coverage

- Statement coverage ("Anweisungsüberdeckung")
- Statement coverage is a **weak measure**

```
int func(int decision)
{
    int a = 0;
    if (decision)
    {
        a = 1;
    }
    return 1 / a;
}
```



Statement (C0) Coverage

Total Coverage: 100.00 %

Test Case / Test Step	Coverage
1 (1)	100.00 %

100%

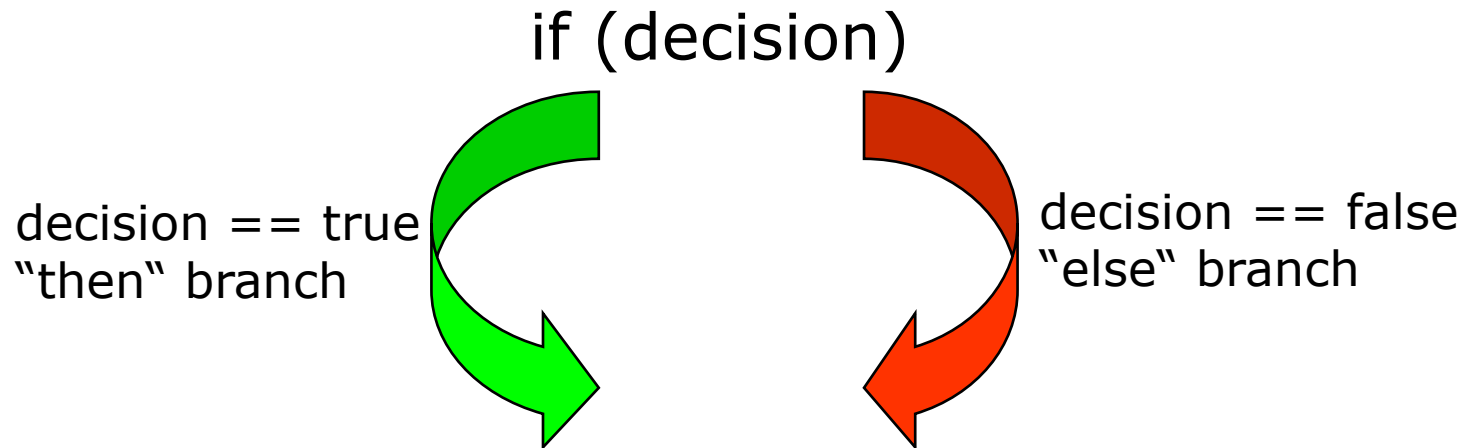
Measured by TESSY

What happens, if *decision* would evaluate to false?



# Branch Coverage

- Branch coverage (“Zweigüberdeckung”)

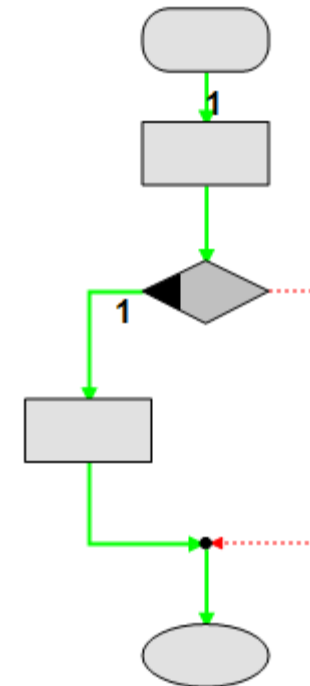


- More relevant than statement coverage

# Branch Coverage

## ■ Branch Coverage for the previous example

```
int func(int decision)
{
    int a = 0;
    if (decision)
    {
        a = 1;
    }
    return 1 / a;
}
```



Branch (C1) Coverage		Statement (C0) Coverage
Total Coverage: 50.00 %		
Test Case / Test Step	Coverage	
▶ 1 (1)	50.00 %	



Measured by TESSY

# Branch Coverage

## ■ Shortcomings of branch coverage

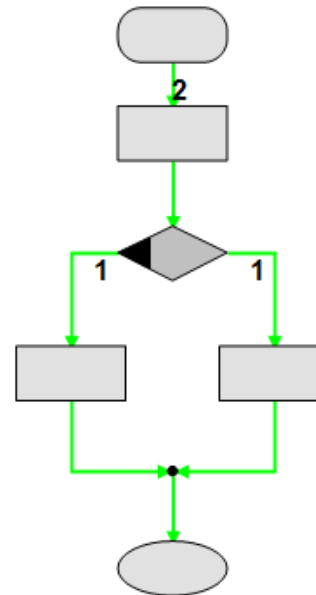
```
int func(int A, int B)
{
    int a = 0;
    if (A && B)
    {
        a = 1 * B;
    }
    else
    {
        a = 2 * B;
    }
    return 1 / a;
}
```

	A	B
1	F	T
2	T	T

# Branch Coverage

## ■ Shortcomings of branch coverage

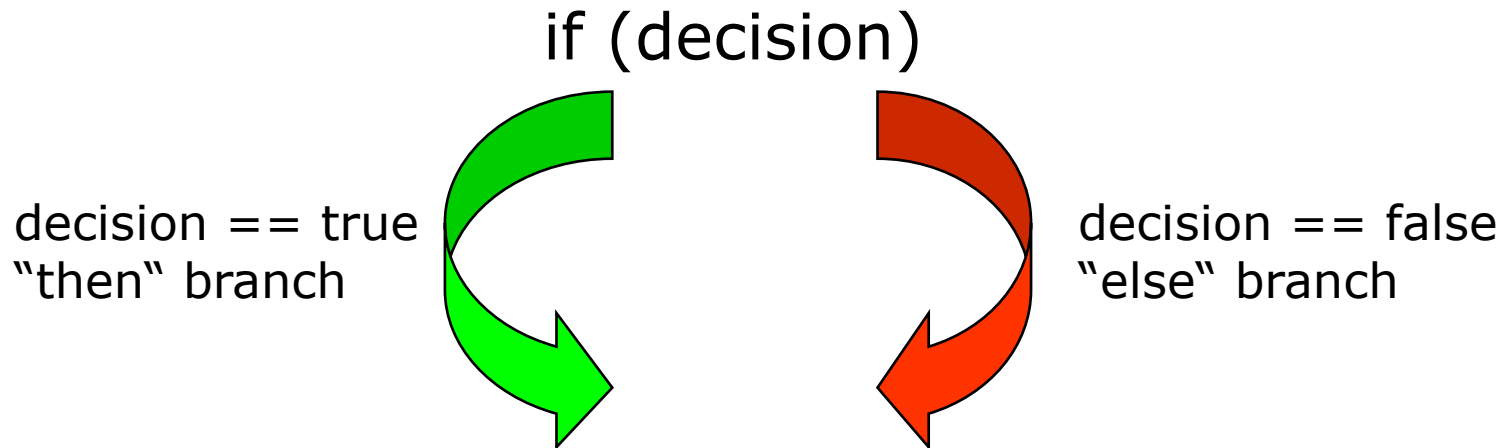
```
int func(int A, int B)
{
    int a = 0;
    if (A && B)
    {
        a = 1 * B;
    }
    else
    {
        a = 2 * B;
    }
    return 1 / a;
}
```



Two test cases yield 100% branch coverage! What happens, if B is false?

# Decision Coverage

- Decision coverage ("Entscheidungsüberdeckung")



- Similar to branch coverage

# Decision Coverage

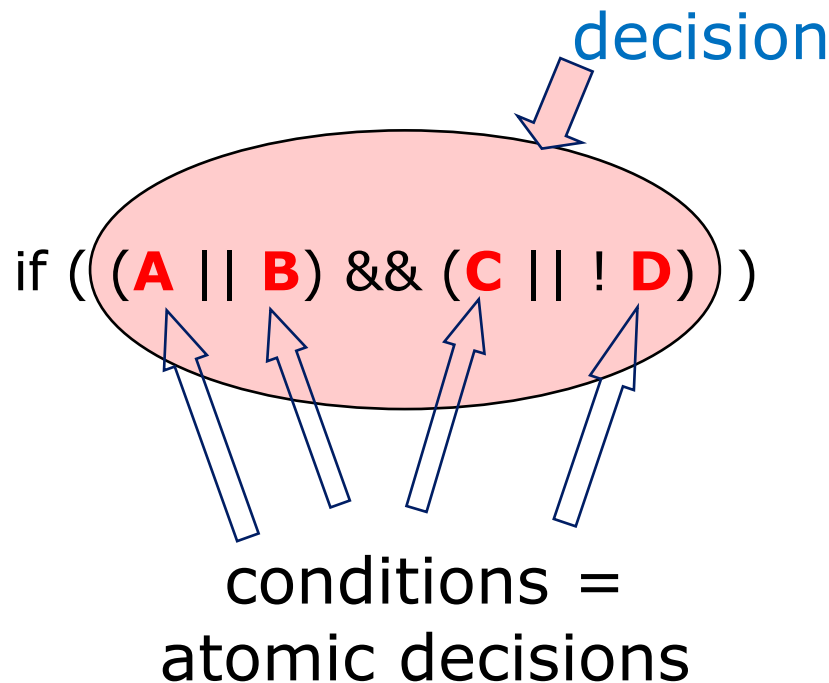
## ■ Relation between Branch Coverage and Decision Coverage

100% branch coverage ~~→~~ 100% decision coverage

**Branch coverage != Decision coverage**

# Condition Coverage

- Condition coverage (“**Bedingungsüberdeckung**”)
  - Reveals, if all conditions (i.e. atomic decisions) in a decision affect the outcome.



- Logical operators: AND, OR

- Modified Condition / Decision Coverage (MC/DC)  
("Modifizierte Bedingungs-/Entscheidungsüberdeckung")
  - Each condition requires a pair of test cases, that
    1. Differs in the Boolean value for that condition, and
    2. Has the same Boolean value for all other conditions, and
    3. Produces true and false in the outcome of the decision
  - **n** conditions require **n + 1** test cases



Code Coverage Introduction

Coverage In Standards

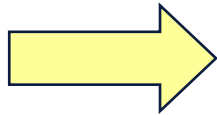
Entry Point Coverage

Statement Coverage

Branch Coverage

Decision Coverage

Condition Coverage (MC/DC)



Advanced Topics

Calculating MC/DC

Calculating Branch Coverage

Names

Process Step

Calculations

Omissions

Test Cases from the Code

Test Case Quality

# Calculating MC/DC

## ■ Two methods of calculating MC/DC

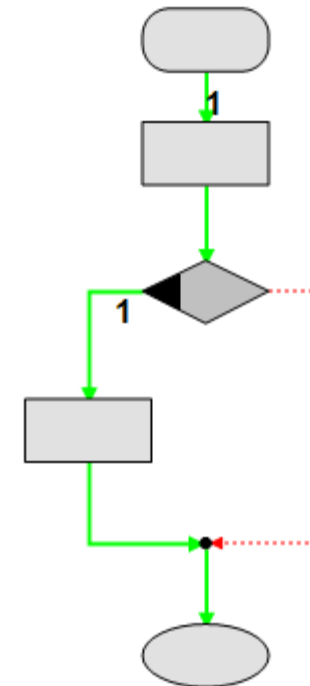
	A	B	C	A && (B    C)
1	f	-	-	F
2	t	f	f	F
3	t	f	t	T
4	t	t	-	T



# Calculating Branch Coverage

## ■ Branch Coverage measured by TESSY

```
int func(int decision)
{
    int a = 0;
    if (decision)
    {
        a = 1;
    }
    return 1 / a;
}
```

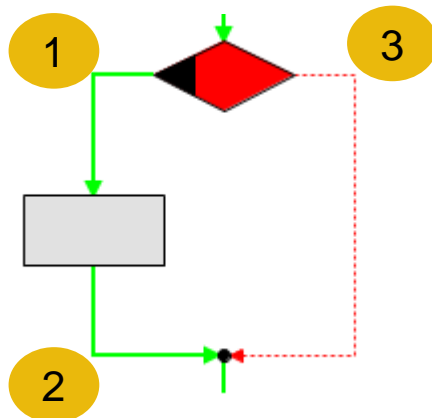


Branch (C1) Coverage	
Statement (C0) Coverage	
Total Coverage: 50.00 %	
Test Case / Test Step	Coverage
▶ 1 (1)	50.00 %

50%

# Calculating Branch Coverage

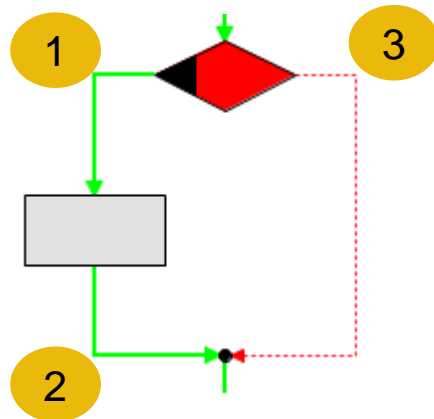
- Branch coverage according to [Spillner 2012], p. 152
  - IF-Anweisung mit leerem ELSE. ... Der THEN-Teil besteht aus zwei Zweigen und einem Knoten, der ELSE-Teil nur aus einem Zweig ohne Knoten (ohne Anweisung). Insgesamt besteht ein IF-Statement mit leerem ELSE-Teil somit aus drei Zweigen. Wird nun der Testfall *true* ausgeführt, sind zwei der drei Zweige überdeckt, also **66%** Überdeckungsgrad sind erreicht.



Three branches:  
One test case → 66% branch coverage!

# Calculating Branch Coverage

- TESSY measurement is based on “primitive” branches
  - Primitive = essential
  - Either #1 or #2 is not primitive



Two primitive branches:  
One test case → 50% branch coverage!

## ■ Names are misleading / ambiguous

	[SPILLNER]	[BEIZER]
Statement coverage	C0	C1
Branch coverage	C1	C2


[SPILLNER]		[LIGGESMEYER]
Condition determination testing	=	Modified Condition / Decision Coverage (MC/DC)
Definierter Bedingungstest / Minimaler Mehrfachbedingungstest	=	Modifizierter Bedingungs- / Entscheidungsüberdeckungstest

# Process Step

- Code coverage should be measured during unit testing

```
void f1(void)
{
    int *pi;

    pi = ...
    if (NULL != pi)
        f2(pi);
    .
    .
}
```



```
int f2(int *pi)
{
    if (NULL == pi)
        return ERR_NULL_POINTER;
    .
}
```

# Calculations

- Code coverage is insensitive to calculations



```
long double sinus(long double x_deg)
{
    int i;
    long double temp, x_rad;
    int sign = -1;

    x_rad = x_deg / 180 * pi ;
    temp = x_rad;

    for(i=3; i<=(MAX_FAC-2); i+=2)
    {
        temp += sign * pot(x_rad,i) / fac(i);
        sign *= -1;
    }
    return(temp);
}
```

One test case with input `x_deg = 0` yields 100%



# Omissions

- Code coverage does not detect omissions



```
long double sinus(long double x_deg)
{
    int i;
    long double temp, x_rad;
    int sign = -1;

    while (x_deg >= 360)x_deg -= 360;

    x_rad = x_deg / 180 * pi ;
    temp = x_rad;

    for(i=3; i<=(MAX_FAC-2); i+=2)
    {
        temp += sign * pwr(x_rad,i) / fac(i);
        sign *= -1;
    }
    return(temp);
}
```

# Test Cases from the Code

- Test cases for 100% shall not be derived from the code
  - Will not find omissions
  - Assumes the code as correct → needs test oracle
  - Not recommended by standards

Methods		ASIL			
		A	B	C	D
1a	Analysis of requirements	++	++	++	++
1b	Generation and analysis of equivalence classes	+	++	++	++
1c	Analysis of boundary values	+	++	++	++
1d	Error guessing	+	+	+	+

ISO 26262:2011, part 6, table 11

- Not recommended
  - ~~Random~~

# Test Case Quality

- Code coverage does not measure the quality of the test cases



*Beware of fool's gold!*

- Test case quality is measured by mutation testing (= error seeding)

## Contact & Additional Information

Frank Büchner  
Dipl.-Inform.  
Principal Engineer Software Quality  
Hitex GmbH  
Greschbachstr. 12  
Karlsruhe  
Germany

**Any questions?**



Tel.: +49 / 721 / 9628 – 125  
[frank.buechner@hitex.de](mailto:frank.buechner@hitex.de)

Information on TESSY: [www.hitex.com/tessy](http://www.hitex.com/tessy)

# Test Case Quality by Mutation Testing

## ■ Mutation testing = Error seeding

```
int lower_limit = 10;  
int boundary    = 35;  
int upper_limit = 50;  
  
if ((lower_limit <= v1) && (v1 < boundary))  
    return category1;
```



```
int lower_limit = 10;  
int boundary    = 25;  
int upper_limit = 50;  
  
if ((upper_limit < v1) || (v1 <= boundary))  
    return category1;
```

- Different constant
- Different variable
- Arithmetic
- Logic

# MC/DC error detection rate

- Case study from the automotive domain [KANDL]

MC/DC [%]	Detected <b>Value</b> Mutations [%]	Detected <b>Variable Name</b> Mutations [%]	Detected <b>Operator</b> Mutations [%]
92 (= max)	100	<b>78</b>	92

# Contact & Additional Information

Frank Büchner  
Dipl.-Inform.  
Principal Engineer Software Quality  
Hitex GmbH  
Greschbachstr. 12  
Karlsruhe  
Germany



Tel.: +49 / 721 / 9628 – 125  
[frank.buechner@hitex.de](mailto:frank.buechner@hitex.de)

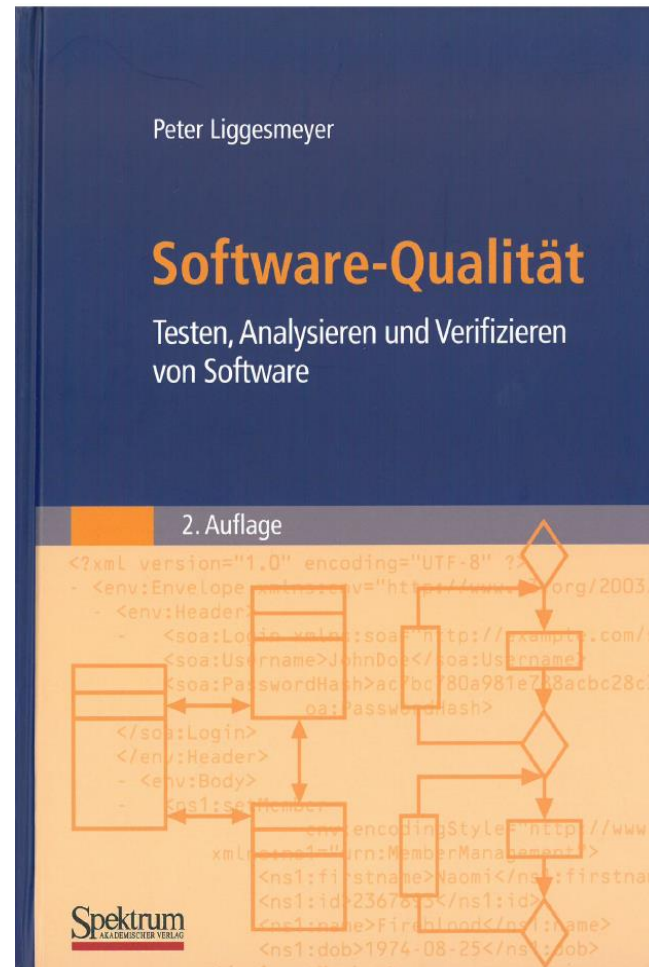
Information on TESSY: [www.hitex.com/tessy](http://www.hitex.com/tessy)

## ■ Liggesmeyer

Ed. 1, 2002



Ed. 2, 2009



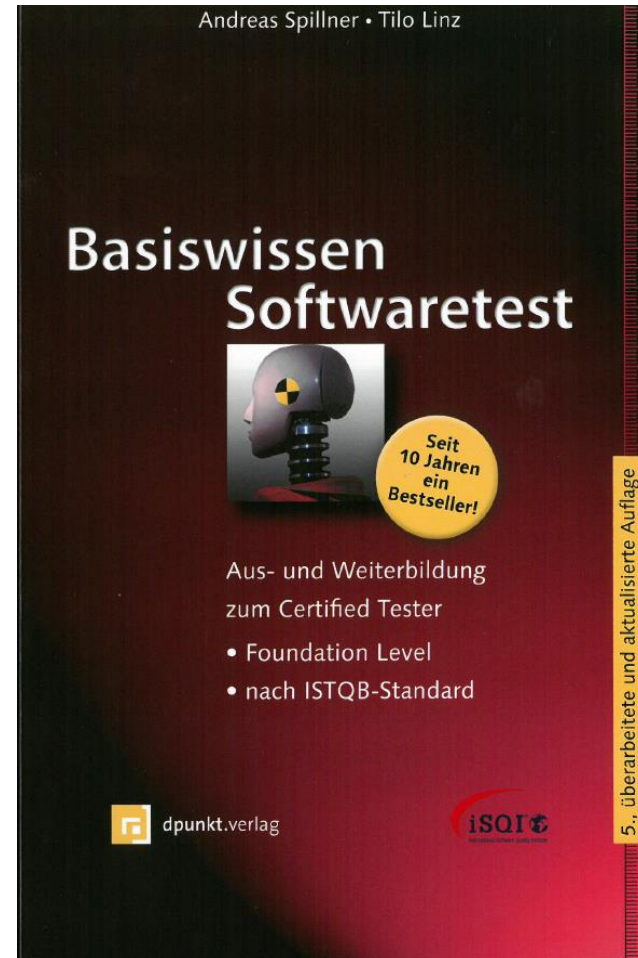


## ■ Spillner, Linz

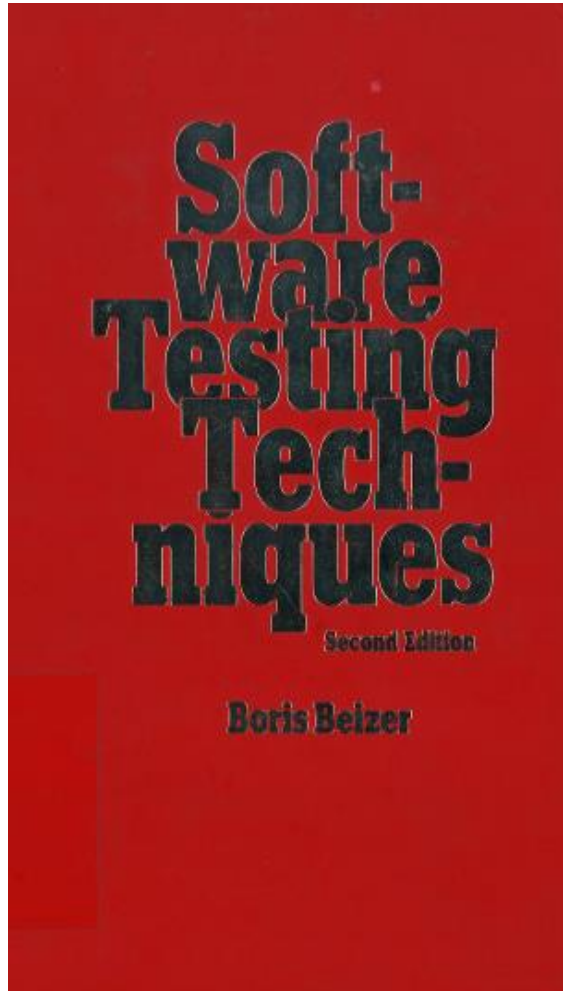
Ed. 1, 2003



Ed. 5, 2012



■ Beizer, 1990



# Literature

- [DO-178B]: Software Considerations In Airborne Systems And Equipment Certification, RTCA, 1992.
- [CAST-10]: Position Paper, Certification Authorities Software Team (CAST), 2002.
- [Kandl]: Susanne Kandl und Raimund Kirner: Error Detection Rate of MC/DC for a Case Study from the Automotive Domain, Institute of Computer Engineering, Vienna University of Technology, Austria. 2011.
- [BCS SIGIST]: Standard for Software Component Testing, British Computer Society - Specialist Interest Group in Software Testing (BSC SIGIST), Working Draft 3.4, April 2001.  
[www.testingstandards.co.uk](http://www.testingstandards.co.uk)