

White-Box Coverage and Automotive Testing

Achieving Standards for Safety-critical Automotive Software



RBCS

**TIME TESTED.
TESTING IMPROVED.**

www.RBCS-US.com



Introduction



The Boeing 787, the most advanced airliner, has about 8 million lines of code.



The Ford F150, the most popular truck, has about 150 million lines of code.



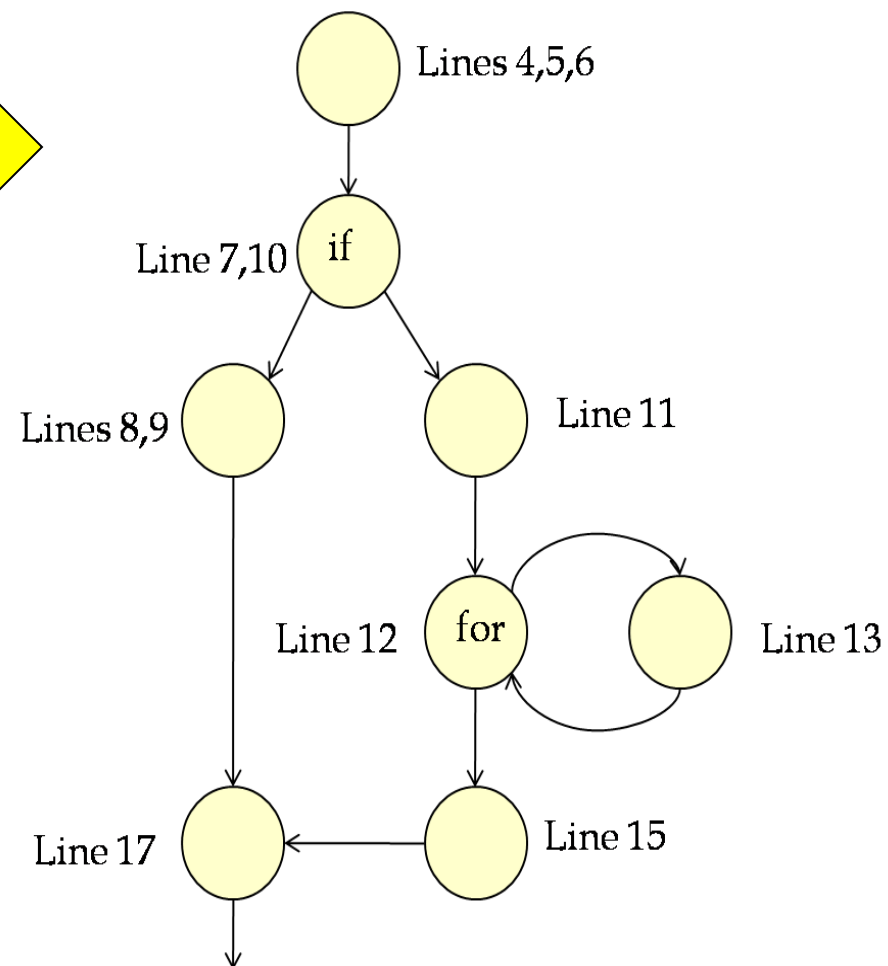
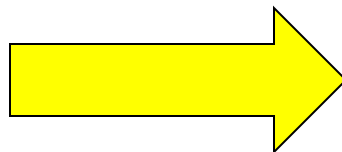
Control Flow Coverage

- ✦ Statement (instruction, code): every statement executed at least once
- ✦ Decision (branch): every possible outcome (T, F)
- ✦ Condition: each atomic condition once T, once F
- ✦ Decision/condition: condition and decision coverage
- ✦ Modified condition/decision: each condition alone determines the outcome twice (once T, once F)
- ✦ Multiple condition: all possible combinations of condition in a decision



Code to Control Flow Example

```
1 #include <stdio.h>
2 main()
3 {
4   int i, n, f;
5   printf("n = ");
6   scanf("%d", &n);
7   if (n < 0) {
8     printf("Invalid: %d\n", n);
9     n = -1;
10  } else {
11    f = 1;
12    for (i = 1; i <= n; i++) {
13      f *= i;
14    }
15    printf("%d! = %d\n", n, f);
16  }
17  return n;
18 }
```





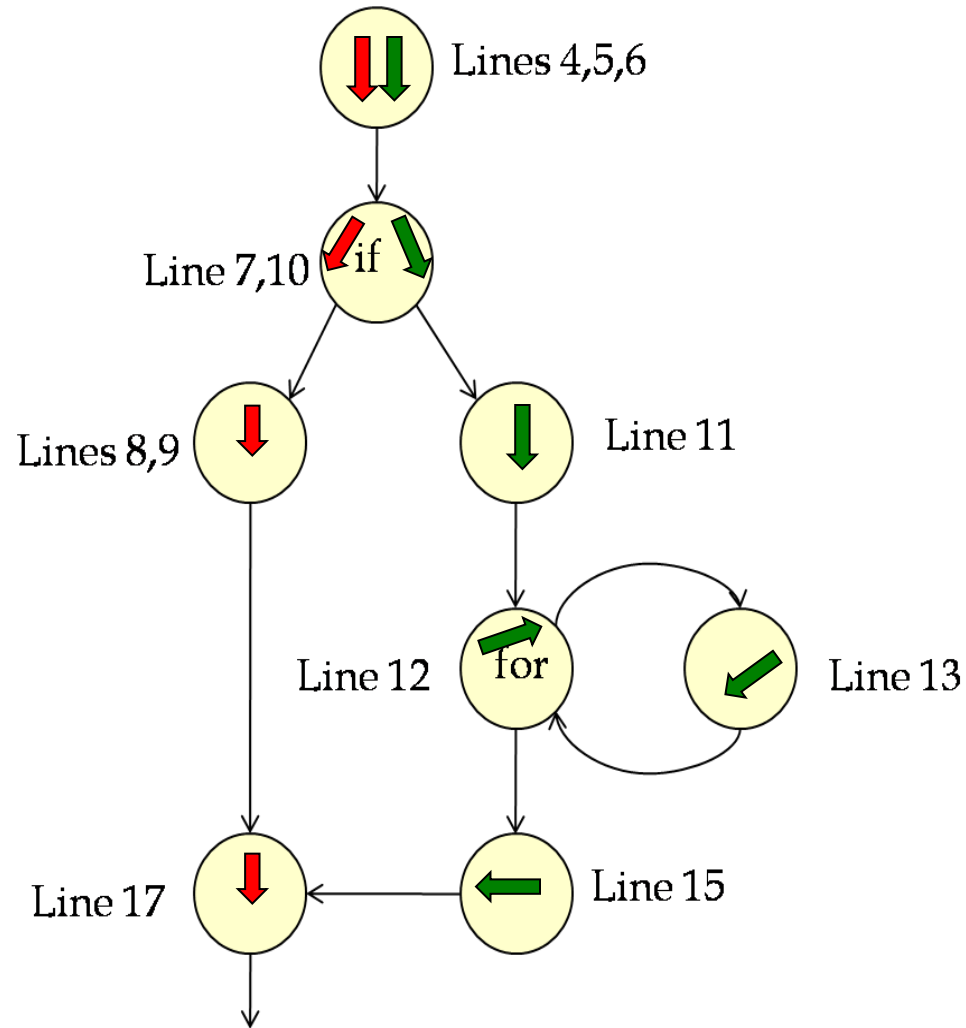
Example: Statement Coverage

```

1 #include <stdio.h>
2 main()
3 {
4   int i, n, f;
5   printf("n = ");
6   scanf("%d", &n);
7   if (n < 0) {
8     printf("Invalid: %d\n", n);
9     n = -1;
10  } else {
11    f = 1;
12    for (i = 1; i <= n; i++) {
13      f *= i;
14    }
15    printf("%d! = %d\n", n, f);
16  }
17  return n;
18 }

```

Test value: **n < 0**
n > 0





What is a Decision?

- ✦ The ability to decide to take one path rather than another is the real power of a computer
- ✦ The decision may be a simple Boolean expression

$a > b$

- ✦ Or, it might be arbitrarily complex

$(a > b) \ || \ (x + y == -1) \ \&\& \ ((d) \ != \ \text{TRUE})$

- ✦ Different programming languages have different decision constructs

- ✦ if then (else)
- ✦ while
- ✦ until do
- ✦ for
- ✦ switch/case
- ✦ and lots more...



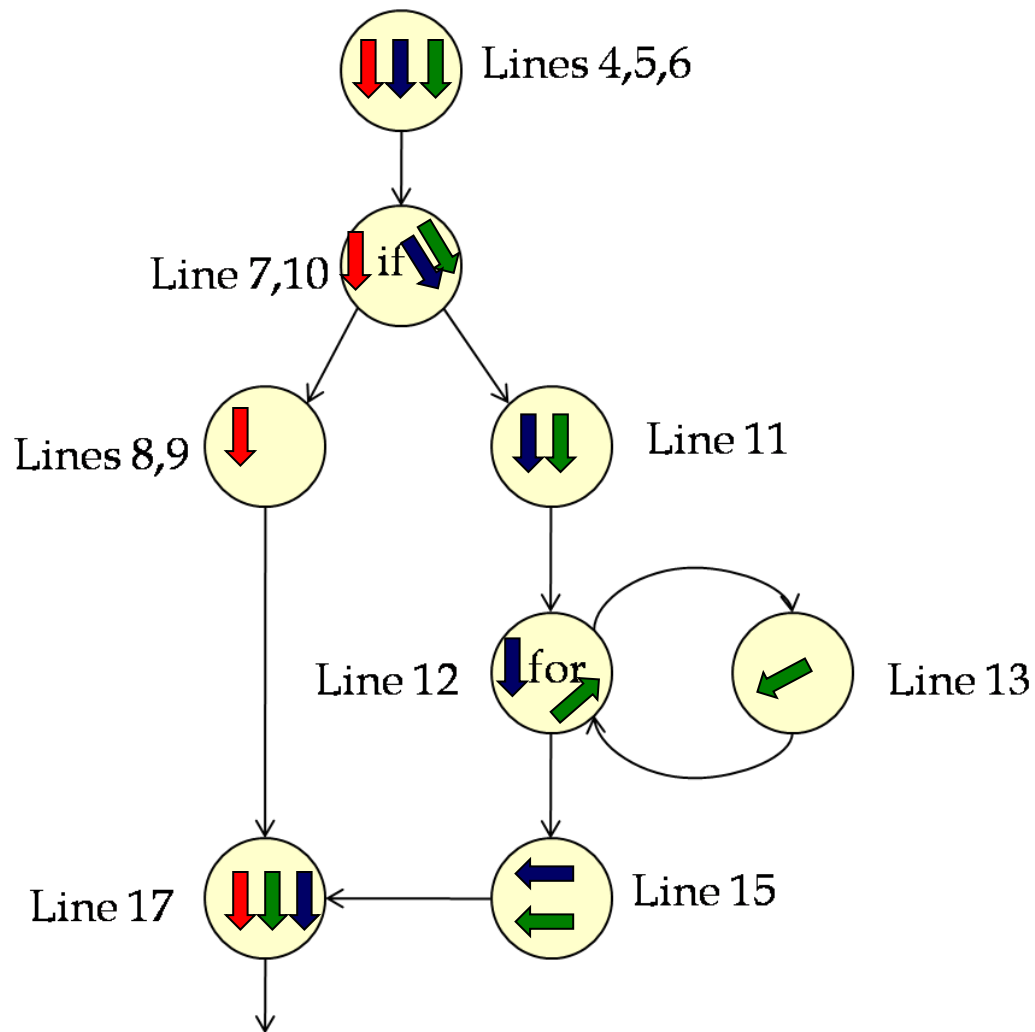
Example: Decision Coverage

```

1 #include <stdio.h>
2 main()
3 {
4   int i, n, f;
5   printf("n = ");
6   scanf("%d", &n);
7   if (n < 0) {
8     printf("Invalid: %d\n", n);
9     n = -1;
10  } else {
11    f = 1;
12    for (i = 1; i <= n; i++) {
13      f *= i;
14    }
15    printf("%d! = %d\n", n, f);
16  }
17  return n;
18 }

```

For statement coverage, we tested
(n < 0) and **(n > 0)**,
 which also covers decisions. For
 branch coverage we need **(n == 0)**





Condition Testing

- ⊕ Decision testing looks at the overall outcome
- ⊕ Condition testing looks at atomic conditions
- ⊕ Each atomic condition must be at least once true and once false
- ⊕ Applicability: mostly as part of understanding higher levels of coverage
- ⊕ Limitations/difficulties: condition coverage doesn't always lead to decision coverage
- ⊕ If a decision is based on a single atomic condition, condition testing is identical to decision testing



Atomic Conditions

- ⊕ A decision is made up of one or more atomic conditions

- ⊕ Examples:

x → TRUE

-- There is one atomic condition

D && F → FALSE

-- There are two atomic conditions

(A || B) && (C == D) → TRUE

-- There are three atomic conditions [A, B, (C==D)]

(a>b) || (x+y==-1) && ((d)!=TRUE) → TRUE

-- There are three atomic conditions

- ⊕ In each case, there is still only one decision for the whole expression



Condition Testing

- ✦ The rule of condition testing is to ensure that each atomic condition has been evaluated to both TRUE and FALSE in testing
- ✦ In the previous slide, **x**, **D**, **F**, **A** and **B** must all be tested TRUE and FALSE
 - ✦ **(C == D)** must be tested both TRUE and FALSE
 - ✦ **(a > b)** must be tested both TRUE and FALSE
 - ✦ **(x+y == -1)** must be tested both TRUE and FALSE
 - ✦ **((d) != TRUE)** must be tested both TRUE and FALSE



Decision Condition Testing

- ✦ Testing condition coverage and decision coverage
- ✦ A decision may be a simple Boolean expression

$a > b$

- ✦ Or, it might be arbitrarily complex

$(a > b) \ || \ (x+y == -1) \ \&\& \ (d) \ != \ \text{TRUE}$

- ✦ To achieve decision coverage, we only ever need two test cases; a test where the decision evaluates to TRUE and one FALSE



Example: Decision/Condition Coverage

- ✦ Consider the following code snippet again:

if (a AND b) then...

- ✦ Condition coverage is satisfied with

a == FALSE, b == TRUE → FALSE

a == TRUE, b == FALSE → FALSE

- ✦ Decision/Condition coverage achieved by adding

a == TRUE, b == TRUE → TRUE

- ✦ Is that sufficient testing?



Modified Condition/Decision Coverage

- ✚ MC/DC is decision/condition coverage plus at least one test where each atomic condition controls the decision outcome
- ✚ Generally, given N atomic conditions, requires N+1 tests
- ✚ Applicability: required for the most safety-critical of automotive systems (highest ASIL)
- ✚ Limitations/difficulties: not always achievable when conditions are coupled or when the compiler short-circuits



Systematic MC/DC Process

1. Draw a table with three columns, with the logical expression and the two possible outcomes as shown
2. Add a blank row for each condition, with the number of blanks equal to the total number of conditions
3. Fill the diagonal blanks in the second column with "T" and in the third column with "F"
4. Replace the remaining blanks with neutral values (F for OR, T for AND)
5. Delete any duplicate tests from the cells

a OR b	T	F
a	--	--
b	--	--

a OR b	T	F
a	<u>T</u> _	<u>F</u> _
b	_ <u>T</u>	_ <u>F</u>

a OR b	T	F
a	<u>T</u> <u>F</u>	<u>F</u> <u>F</u>
b	<u>F</u> <u>T</u>	<u>F</u> <u>F</u>

a OR b	T	F
a	<u>T</u> <u>F</u>	<u>F</u> <u>F</u>
b	<u>F</u> <u>T</u>	<i>Del</i>



Example: MC/DC Coverage

- ✦ Consider the following code snippet:

if ((a OR b) AND c) then...

- ✦ Decision/Condition coverage can be achieved with

a == TRUE, b == TRUE, c == TRUE → TRUE

a == FALSE, b == FALSE, c == FALSE → FALSE

- ✦ Notice that b never independently affects the outcome of the decision, so we don't have MC/DC coverage here
- ✦ Let's try the process...



Example: Achieving MC/DC

1. Draw a table with three columns, with the logical expression and the two possible outcomes as shown
2. Add a blank row for each condition, with the number of blanks equal to the total number of conditions
3. Fill the diagonal blanks in the second column with "T" and in the third column with "F"
4. Replace the remaining blanks with neutral values (F for OR, T for AND); careful with compound expressions like this one!
5. Delete any duplicate tests from the cells

(a OR b) AND c	T	F
a	---	---
b	---	---
c	---	---

(a OR b) AND c	T	F
a	<u>T</u> ---	<u>F</u> ---
b	-- <u>T</u>	-- <u>F</u>
c	-- <u>T</u>	-- <u>F</u>

(a OR b) AND c	T	F
a	<u>T</u> <u>F</u> <u>T</u>	<u>F</u> <u>F</u> <u>T</u>
b	<u>F</u> <u>T</u> <u>T</u>	<u>F</u> <u>F</u> <u>T</u>
c	<u>T</u> <u>F</u> <u>T</u>	<u>T</u> <u>F</u> <u>F</u>

(a OR b) AND c	T	F
a	<u>T</u> <u>F</u> <u>T</u>	<u>F</u> <u>F</u> <u>T</u>
b	<u>F</u> <u>T</u> <u>T</u>	<i>Del</i>
c	<i>Del</i>	<u>T</u> <u>F</u> <u>F</u>

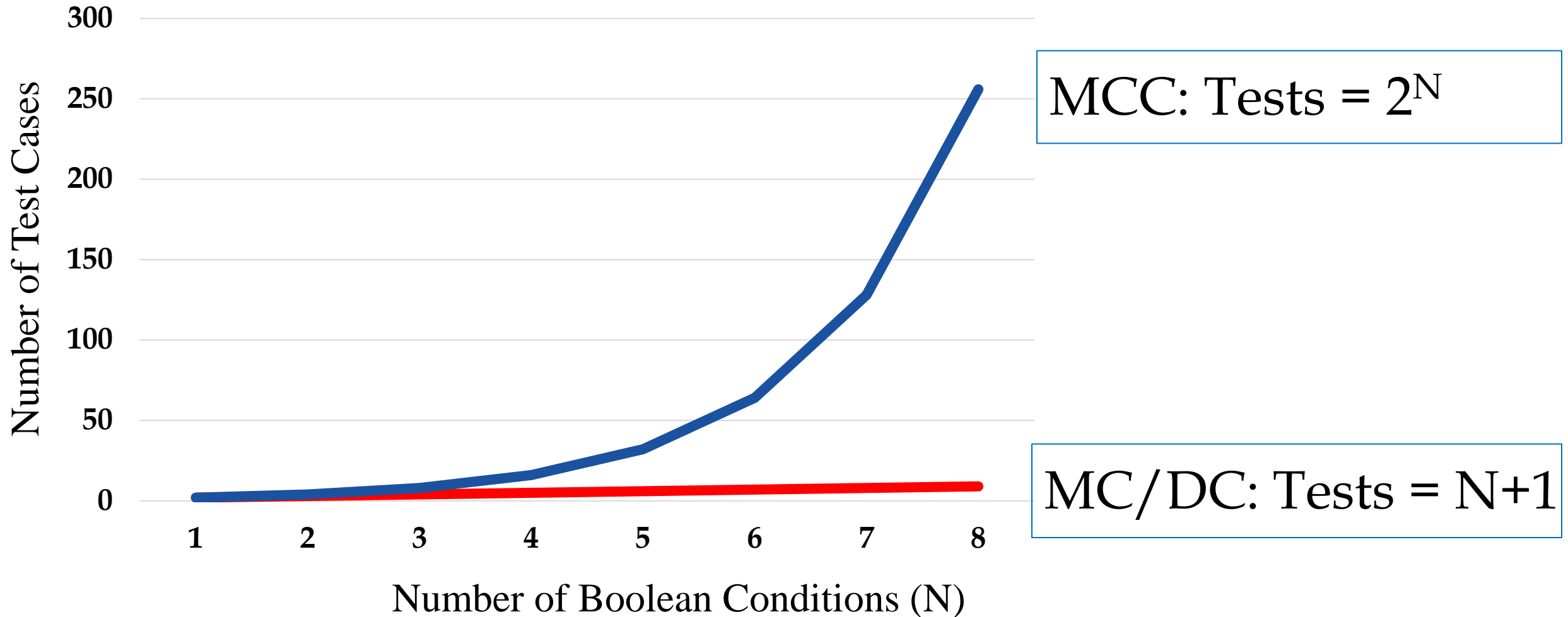


Multiple Condition Coverage

- ✦ Concept: exhaustively test every single combination of conditions that can affect an outcome
- ✦ Test derivation: use Boolean truth table to generate all possible combinations of values that must be tested (2^N)
- ✦ This level of coverage is not required in automotive testing, because the cost is very high without delivering any significant additional risk mitigation



First Rule of Testing: Never Talk About Testing Everything



MCC: Tests = 2^N

MC/DC: Tests = N+1



Conclusion

- ❖ In this webinar, you've learned the three code coverage levels you need to achieve for automotive testing
 - ❖ Statement (or instruction or code)
 - ❖ Decision (or branch)
 - ❖ Modified condition/decision coverage (MC/DC)
- ❖ The relevant level of coverage is based on Automotive Safety Integrity Level (ASIL)
- ❖ Knowing how to apply these techniques is essential to contributing as a tester, SDET, or developer in an automotive software project



...*Contact RBCS*

For over 25 years, RBCS has delivered consulting, training, and expert services to clients, helping them with software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS advises its clients, trains their employees, conducts product testing, builds and improves testing groups, provides advice, reports, and testimony for law firms in legal matters, and hires test staff for thousands of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. Find us at the coordinates below:

Address: RBCS, Inc.
31520 Beck Road
Bulverde, TX 78163-3911
USA

Phone: +1 (830) 438-4830

E-mail: info@rbc-us.com

Web: www.rbc-us.com

Twitter: @RBCS, @LaikaTestDog

Facebook: @TestingImprovedbyRBCS

LinkedIn: <https://www.linkedin.com/in/rex-black>

YouTube: <https://www.youtube.com/user/RBCSINC>

Podcasts: <https://rbc-us.com/resources/podcast/>
www.rbc-us.com