Assessing Testing Evidence: Do We Know How?

Mats Heimdahl

Professor, Computer Science and Engineering

Director, University of Minnesota Software Engineering Center

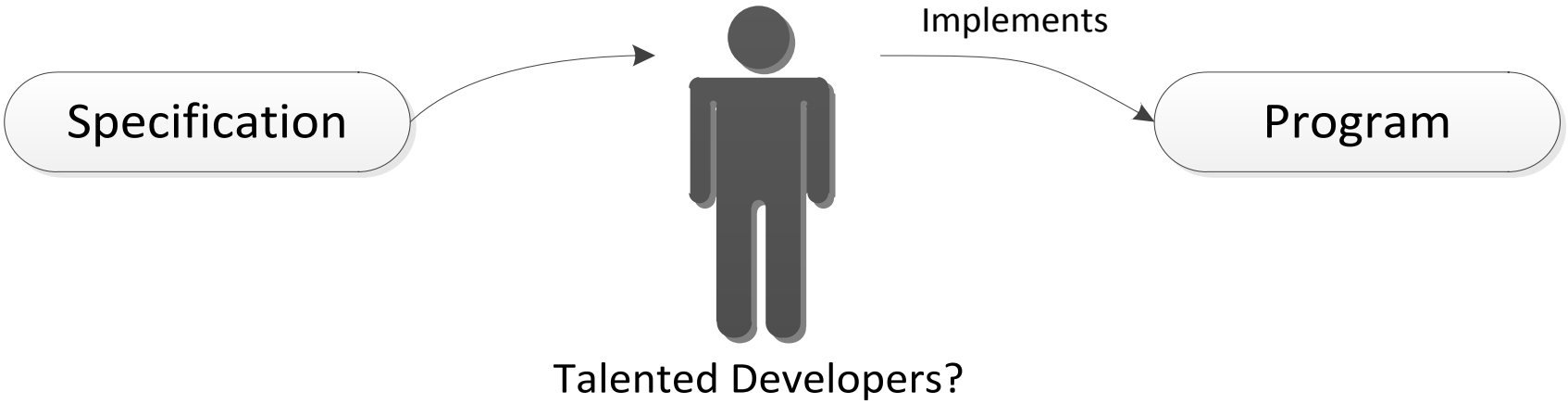
UNIVERSITY OF MINNESOTA

Software Engineering Center

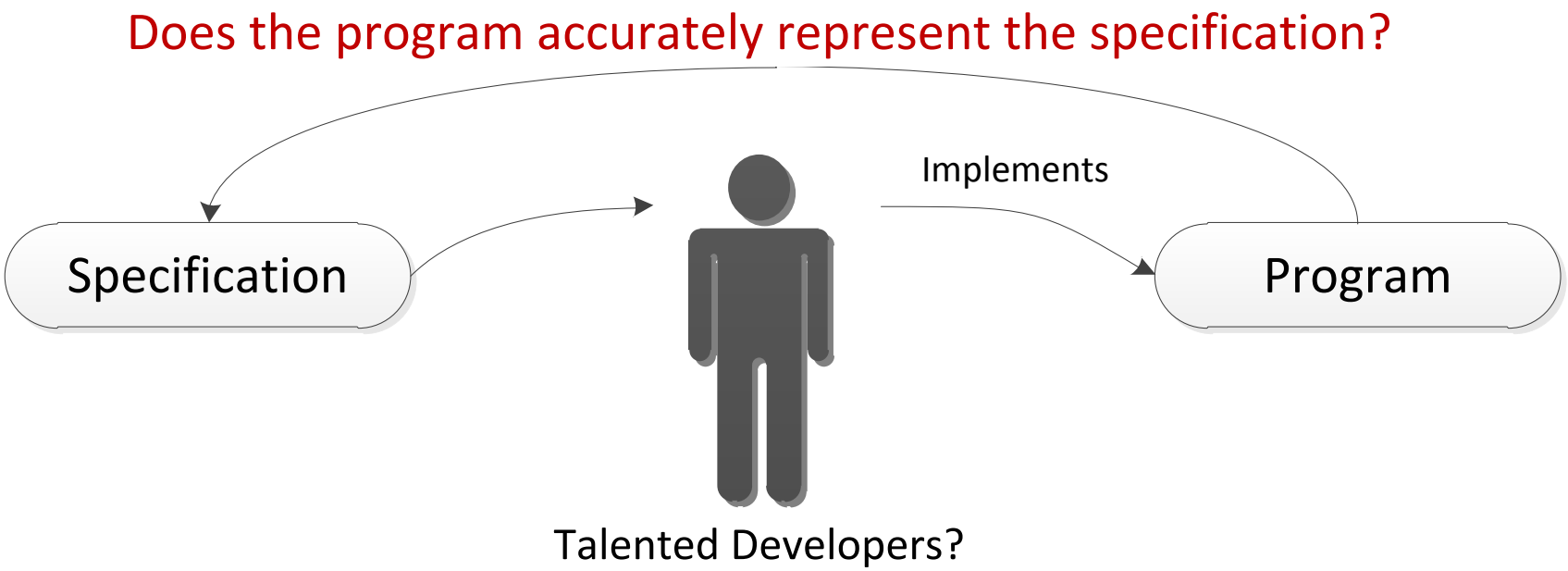
Funded by CNS-0931931 and CNS-1035715


NO

Software Development



The Big Question



McDermid:

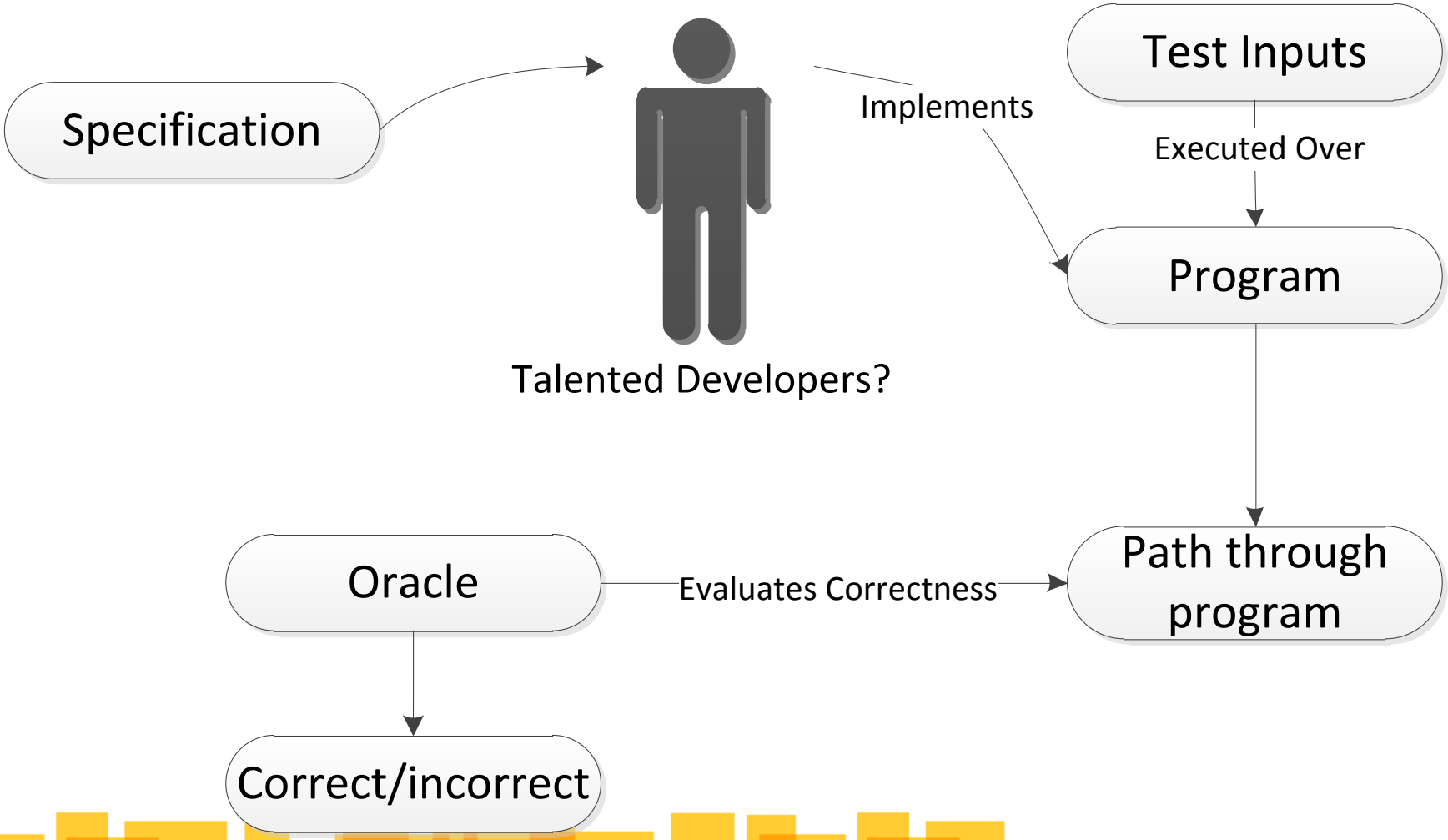
"Software Safety: Where is the evidence?"

- Bring the Evidence!!
- What Evidence????

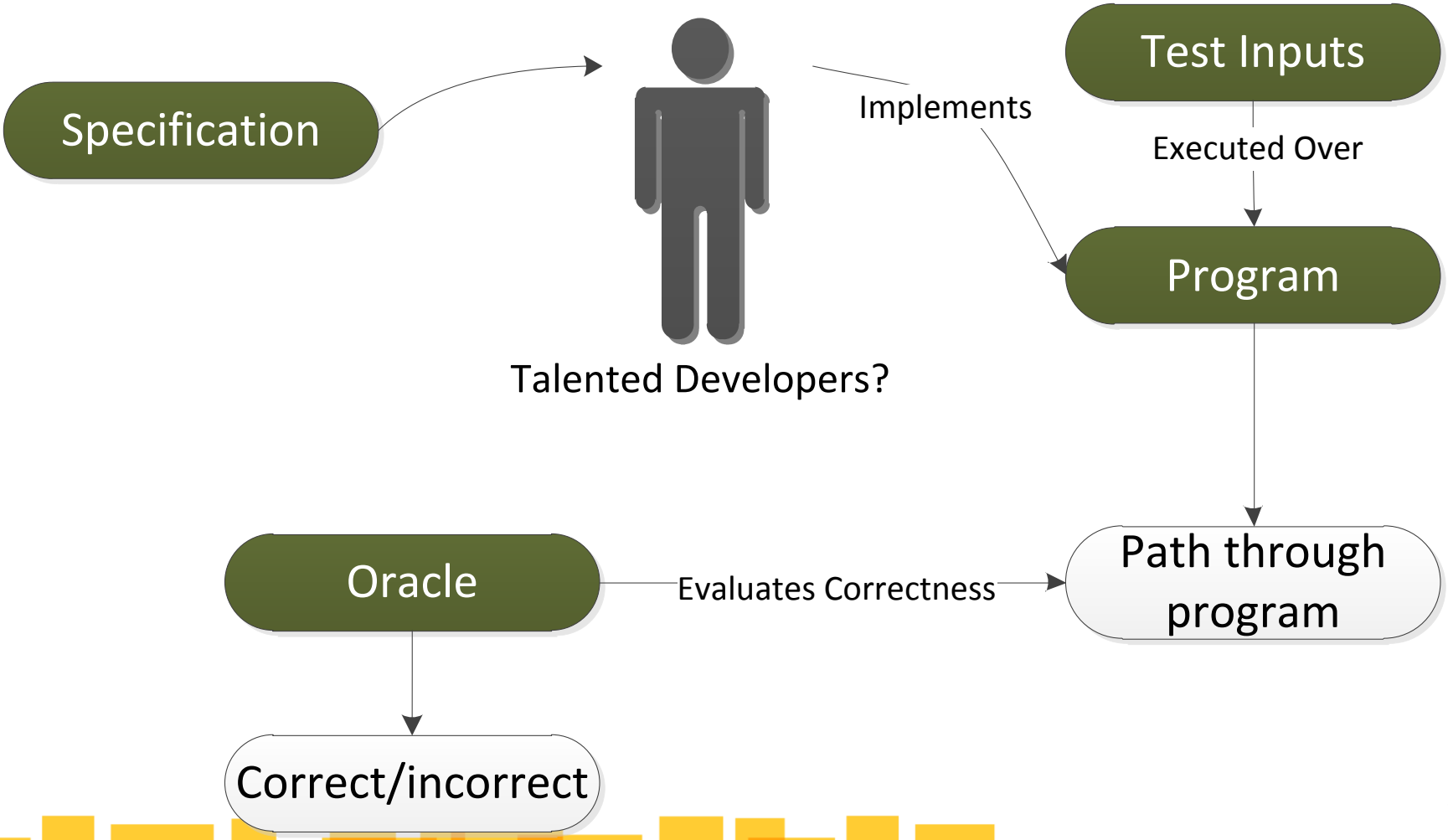
Software meets its requirements

1. Inspection
2. Testing
3. Formal Verification

Testing Process



Testing Process

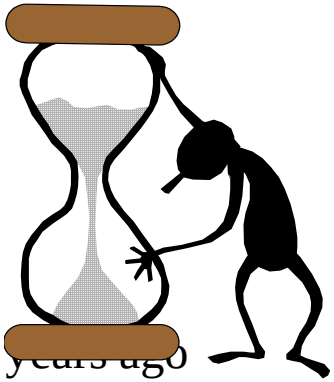


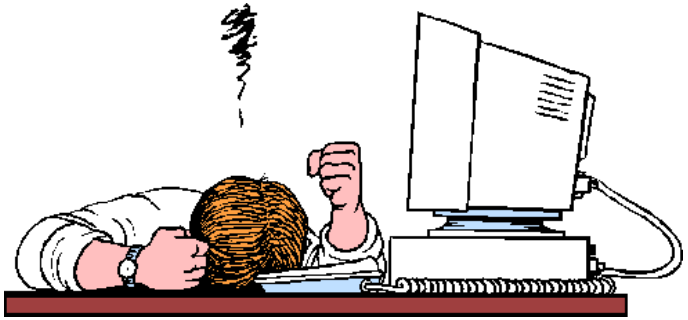
The **BIG** Question

Does this testing show the requirements are met?
Is this **adequate evidence**?

What About Testing??

- Statistical Testing
  - **Does not work**
    - Butler and Finelli 25 years ago
      - R. W. Butler and G. B. Finelli. "The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software"

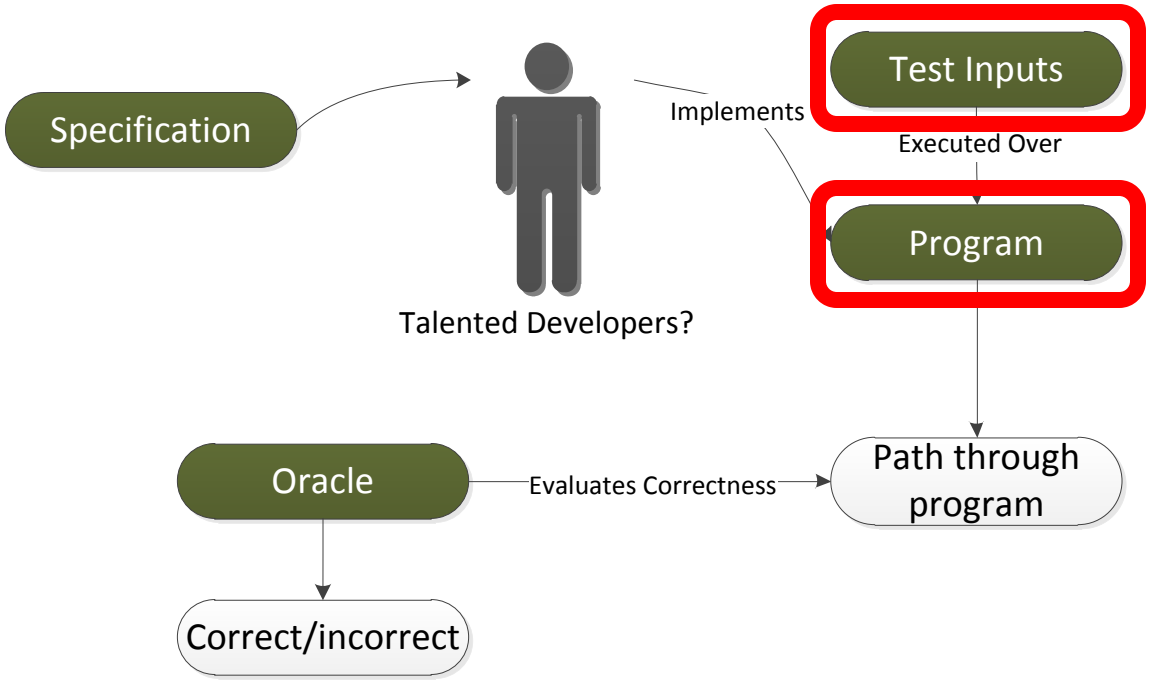




- Coverage Criteria
  - **Does not work** (yet)
    - As will be shown

- Engineering Judgment
  - Assisted by coverage measures
  - Not objective!!!

Current Coverage Criteria



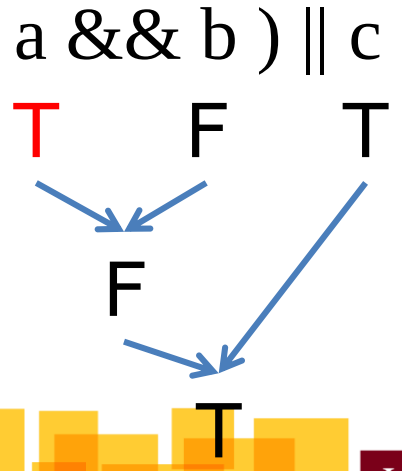
$$C : P \times 2^{T}$$

Modified Condition/Decision Coverage (MC/DC)

To satisfy MC/DC:

- Every basic condition in a decision in the model should take on all possible outcomes at least once, and
- Each basic condition should be shown to independently affect the decision's outcome

a = **T**
b = F
c = T

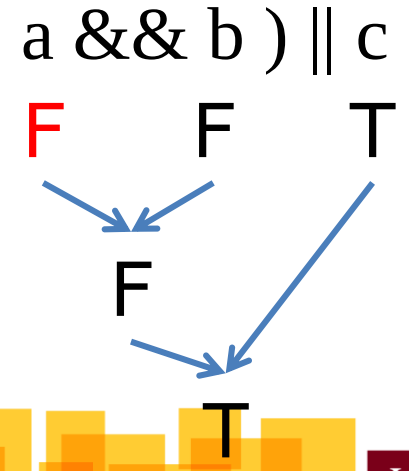


Modified Condition/Decision Coverage (MC/DC)

To satisfy MC/DC:

- Every basic condition in a decision in the model should take on all possible outcomes at least once, and
- Each basic condition should be shown to independently affect the decision's outcome

a = **F**
b = F
c = T

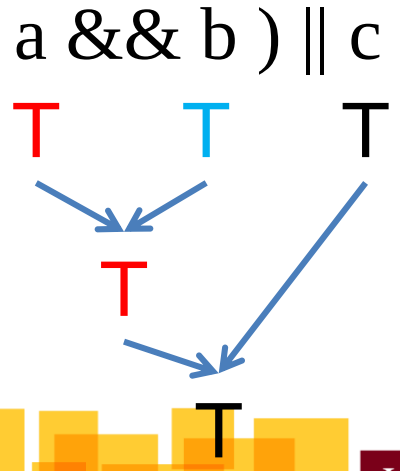


Modified Condition/Decision Coverage (MC/DC)

To satisfy MC/DC:

- Every basic condition in a decision in the model should take on all possible outcomes at least once, and
- Each basic condition should be shown to independently affect the decision's outcome

a = T
b = T
c = T

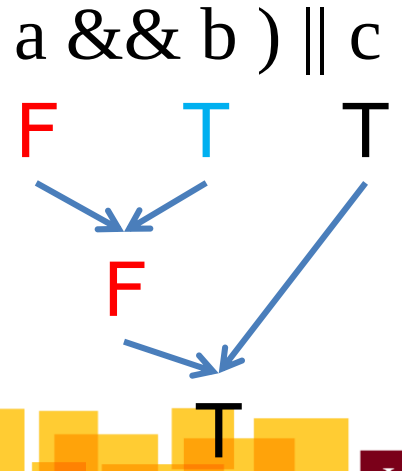


Modified Condition/Decision Coverage (MC/DC)

To satisfy MC/DC:

- Every basic condition in a decision in the model should take on all possible outcomes at least once, and
- Each basic condition should be shown to independently affect the decision's outcome

a = **F**
b = **T**
c = T

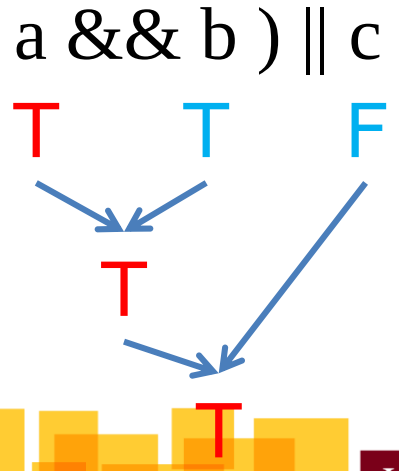


Modified Condition/Decision Coverage (MC/DC)

To satisfy MC/DC:

- Every basic condition in a decision in the model should take on all possible outcomes at least once, and
- Each basic condition should be shown to independently affect the decision's outcome

a = **T**
b = **T**
c = **F**

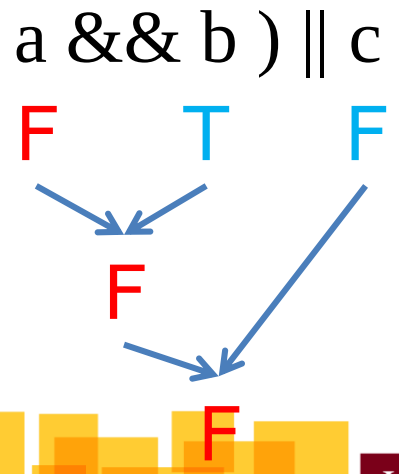


Modified Condition/Decision Coverage (MC/DC)

To satisfy MC/DC:

- Every basic condition in a decision in the model should take on all possible outcomes at least once, and
- Each basic condition should be shown to independently affect the decision's outcome

a = **F**
b = **T**
c = **F**



Masking and Measurement of MC/DC

Version 1:

Non-Inlined Implementation

expr1 = in1 **or** in2;

out1 = expr1 **and** in3;

Version 2:

Inlined Implementation

out1 = (in1 **or** in2) **and** in3;

Tests in green satisfy MC/DC for version 1 but not 2

In1	In2	In3	In1 or in2	(in1 or in2) and in3
F	F	F	F	F
F	F	T	F	F
F	T	F	T	F
F	T	T	T	T
T	F	F	T	F
T	F	T	T	T
T	T	F	T	F
T	T	T	T	T

Masking and Measurement of MC/DC

Version 1:

Non-Inlined Implementation

expr1 = in1 **and** in2;

out1 = expr1 **and** in3;

Version 2:

Inlined Implementation

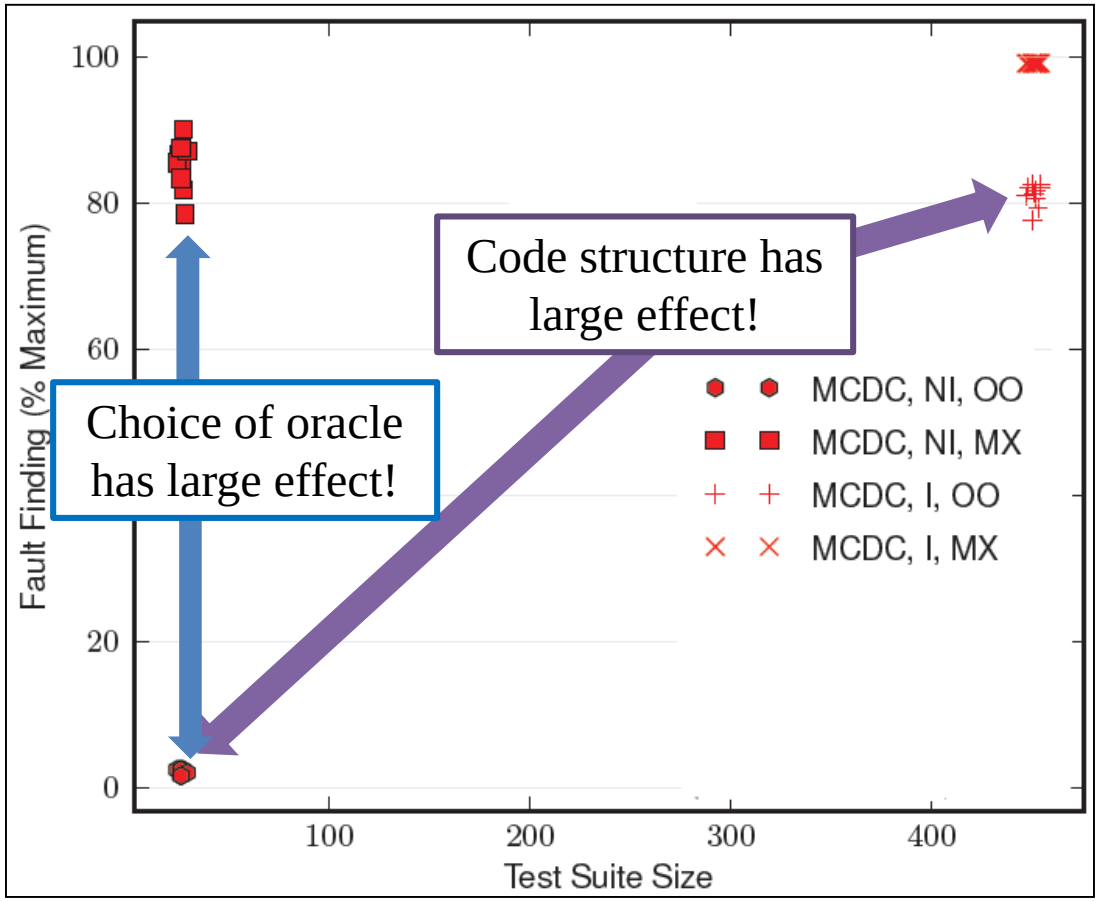
out1 = (in1 **or** in2) **and** in3;

Tests in green satisfy MC/DC for version 1 but not 2

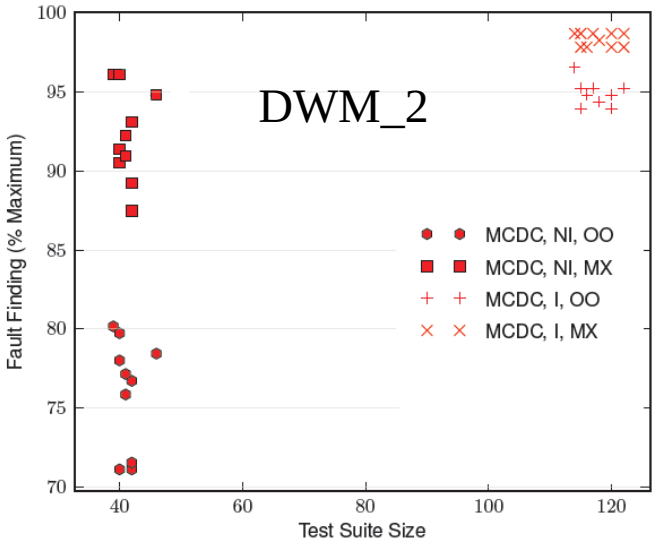
Tests still pass if we replace 'or' with 'and'

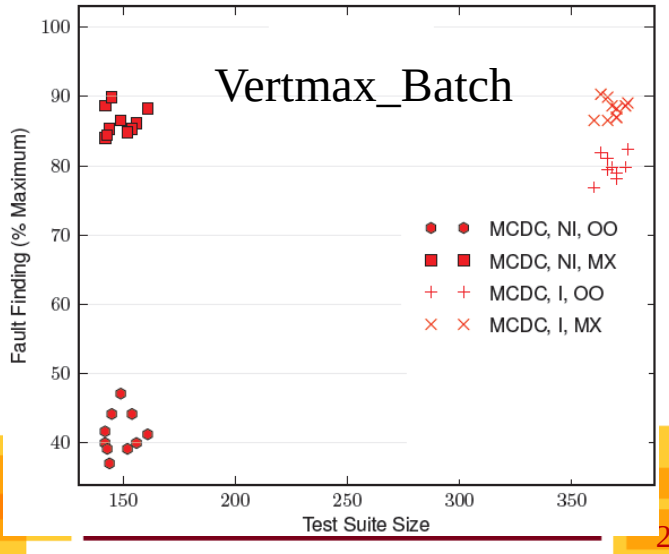
In1	In2	In3	In1 & in2	(in1 & in2) and in3
F	F	F	F	F
F	F	T	F	F
F	T	F	F	F
F	T	T	T	T
T	F	F	F	F
T	F	T	T	T
T	T	F	T	F
T	T	T	T	T

MC/DC Effectiveness



DWM_1





Oracle Data

Version 1:

Non-Inlined Implementation

expr1 = in1 **or** in2;

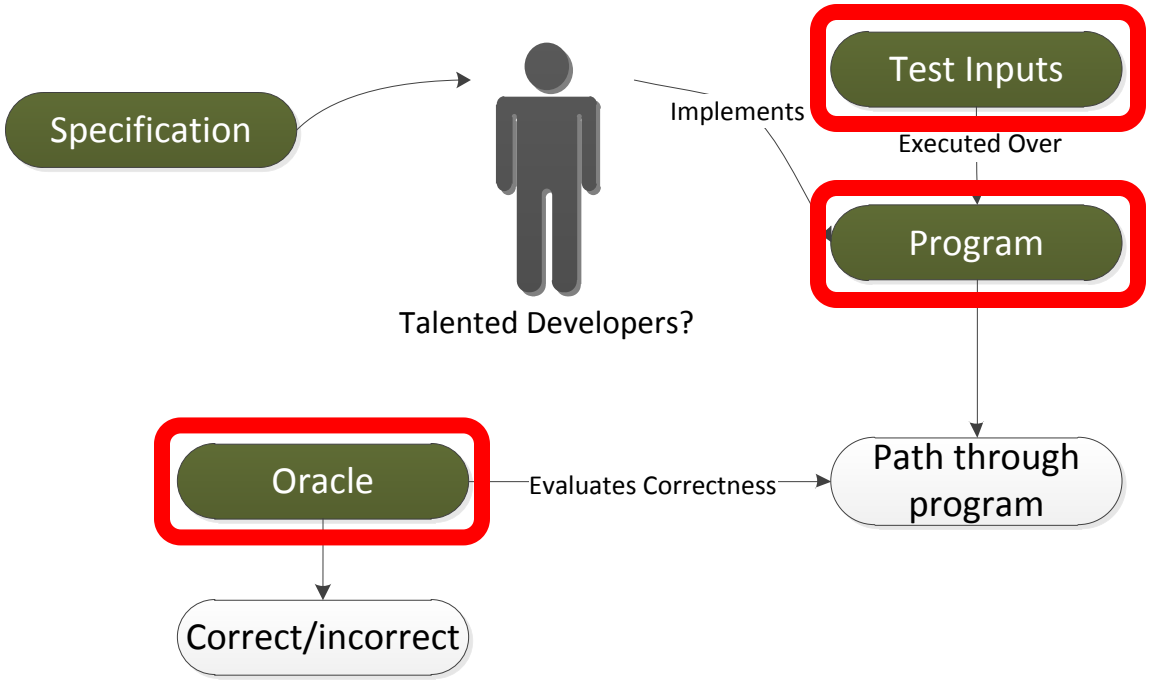
out1 = expr1 **and** in3;

Version 2:

Inlined Implementation

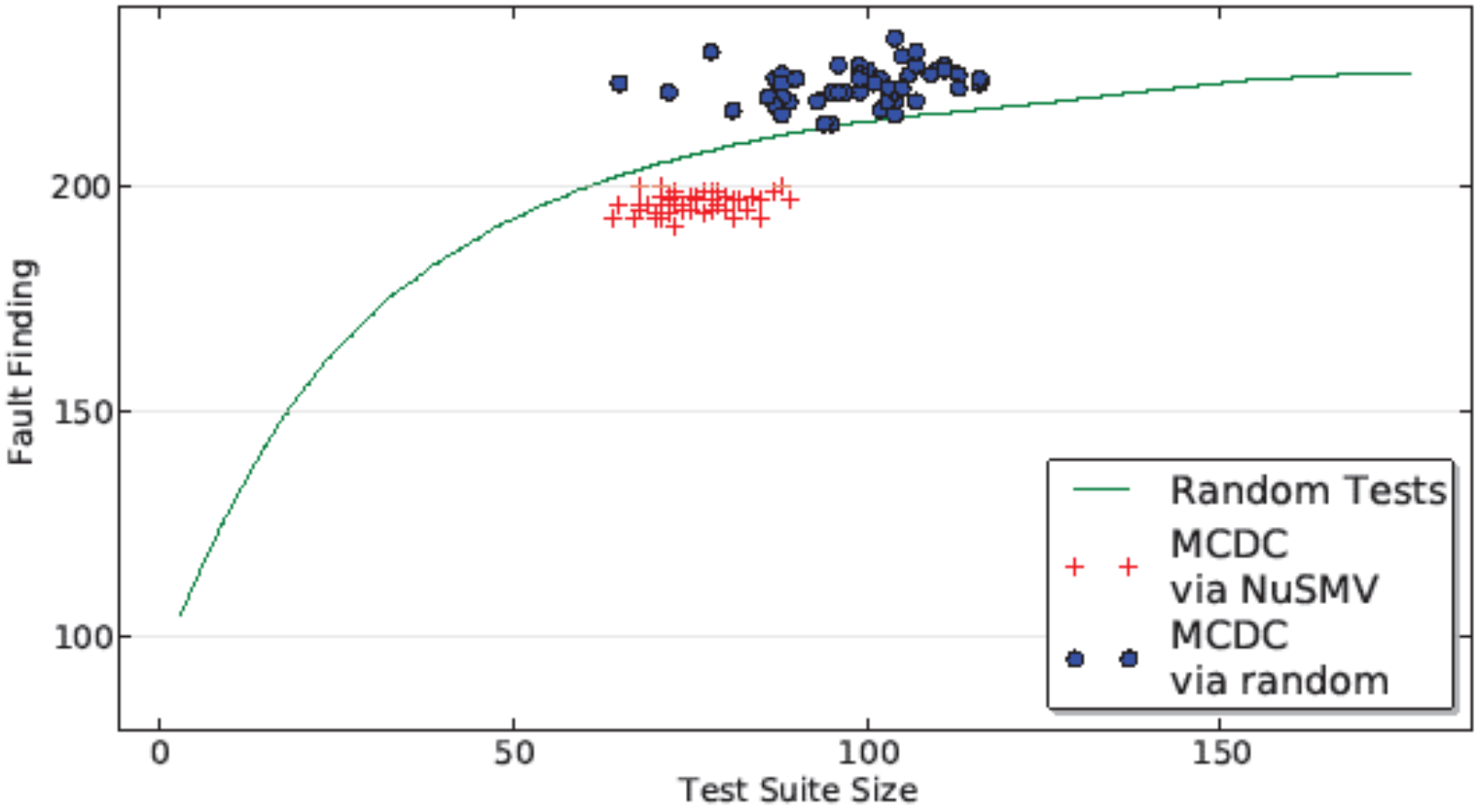
out1 = (in1 **or** in2) **and** in3;

The Oracle is Important

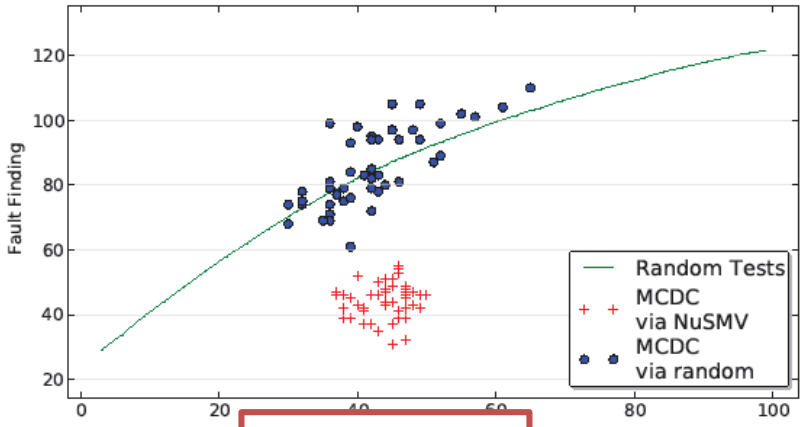


$$C : P \times 2^{T} \times O$$

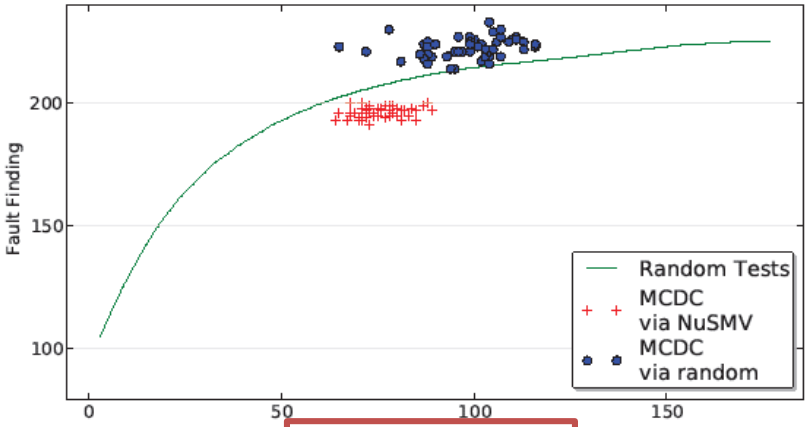
Is MC/DC Any Good?



Random Testing is Pretty Good

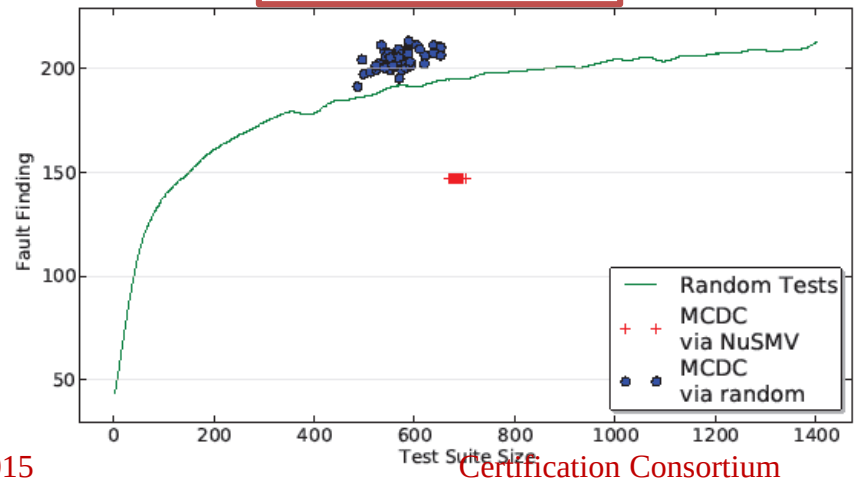


DWM_1

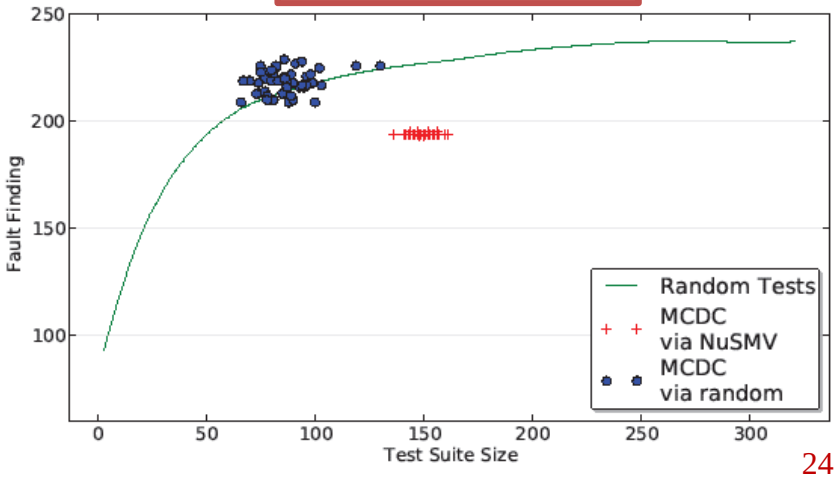


DWM_2

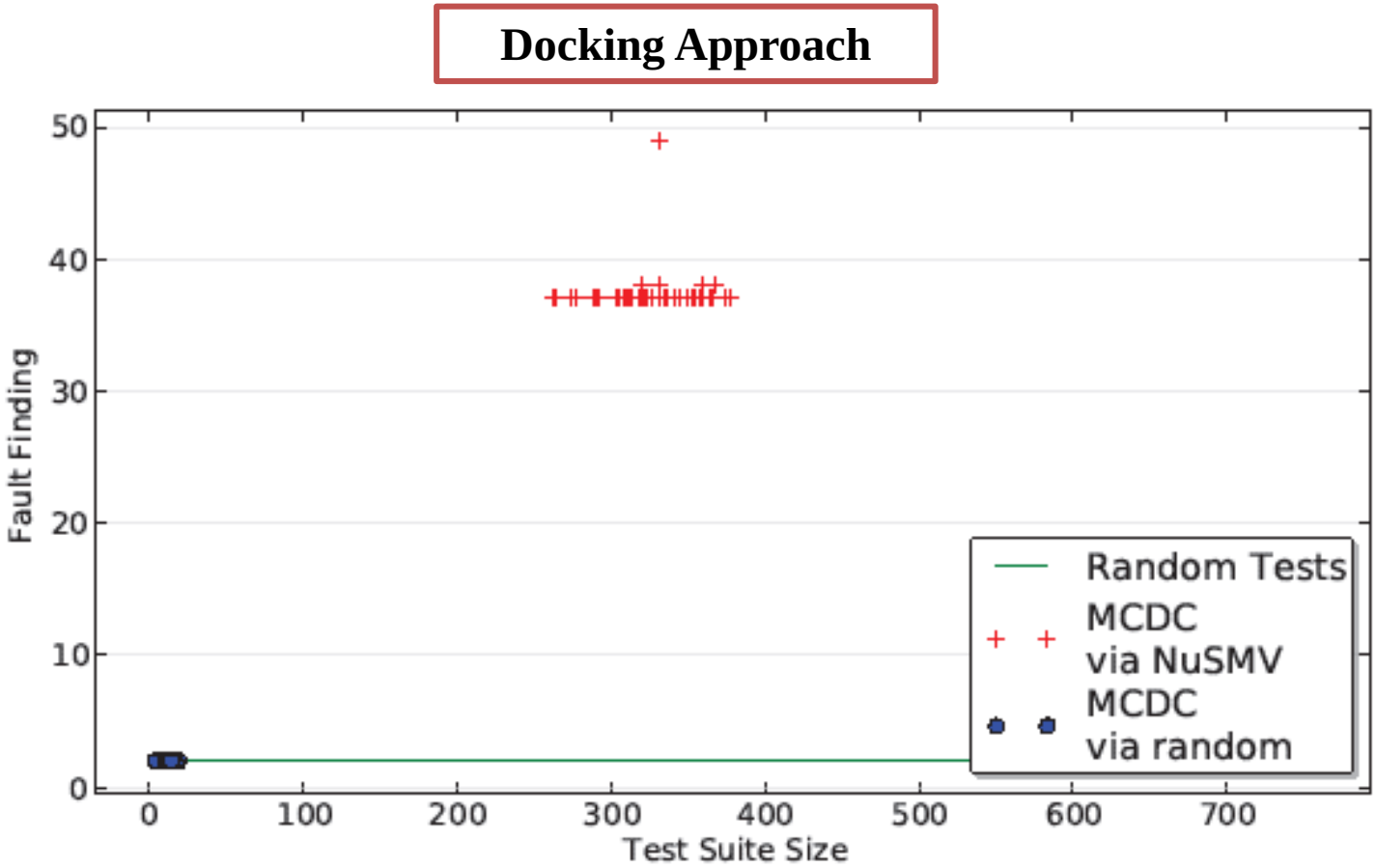
Vertmax_Batch



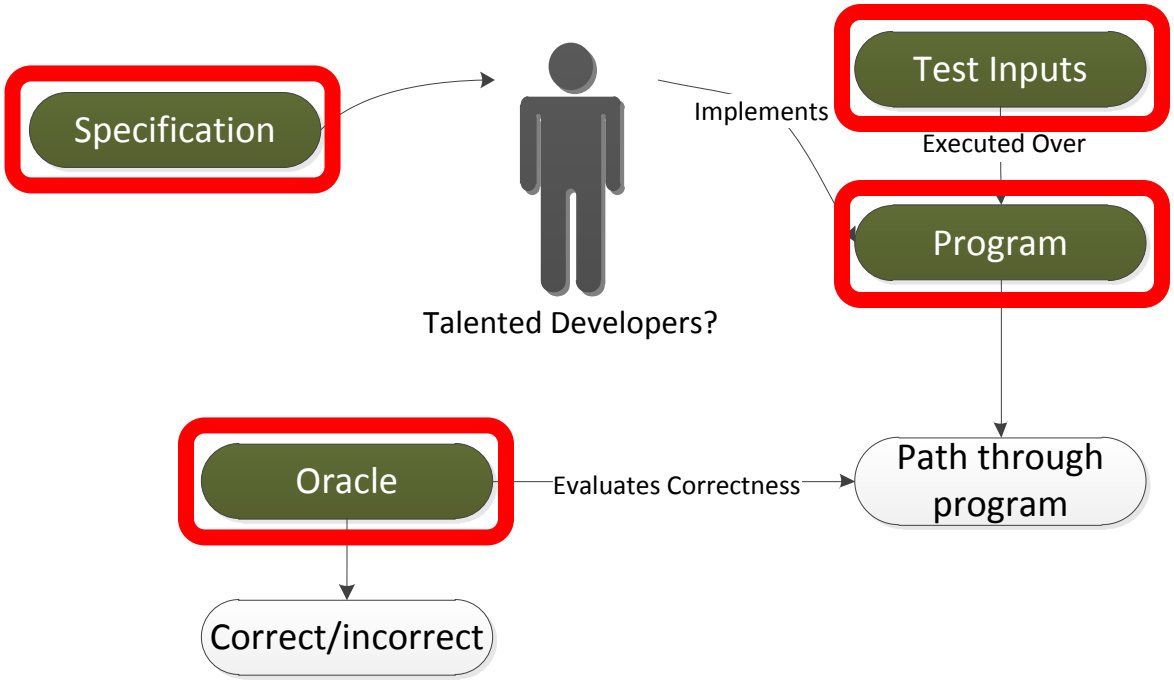
Latctl_Batch



Or Not…

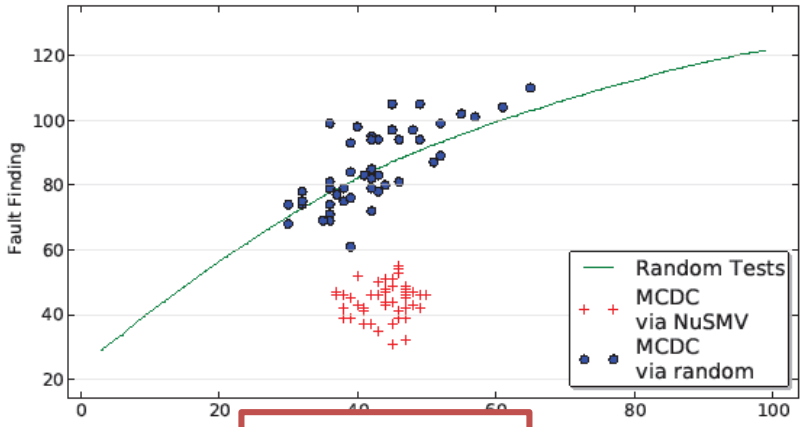

Docking Approach
Fault Finding
Test Suite Size
0
10
20
30
40
50
0
100
200
300
400
500
600
700
Random Tests
MCDC via NuSMV
MCDC via random

Complete Adequacy Criteria

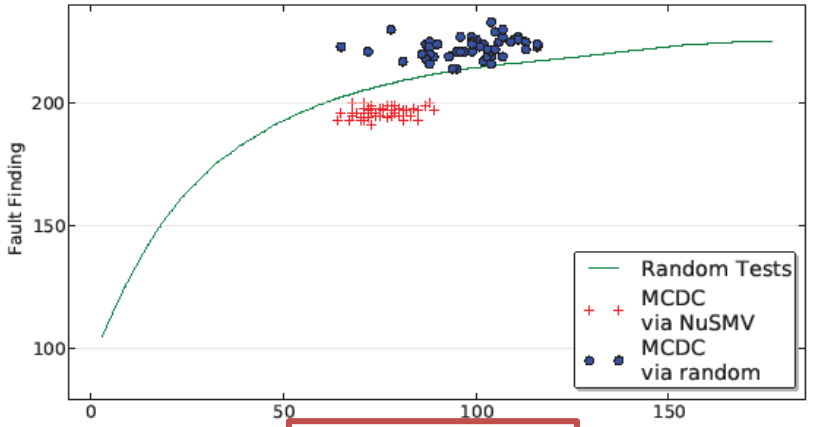


$$C: P \times S \times 2^T \times O$$

Random Testing is Pretty Good

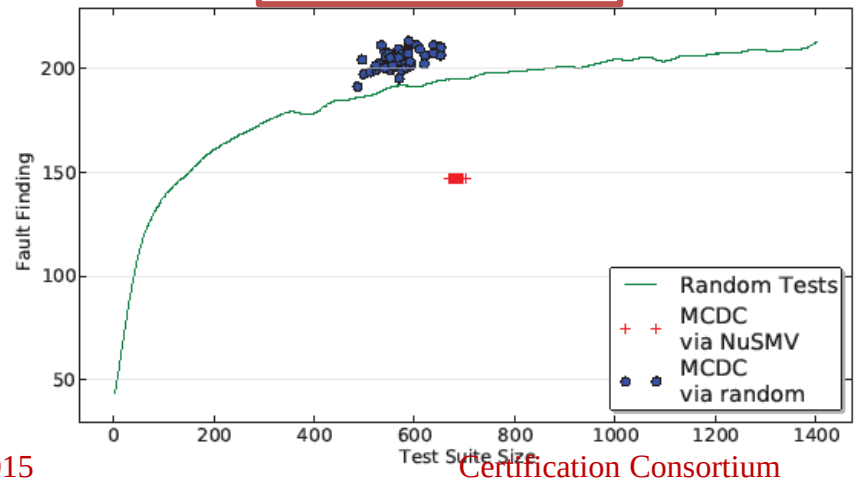


DWM_1

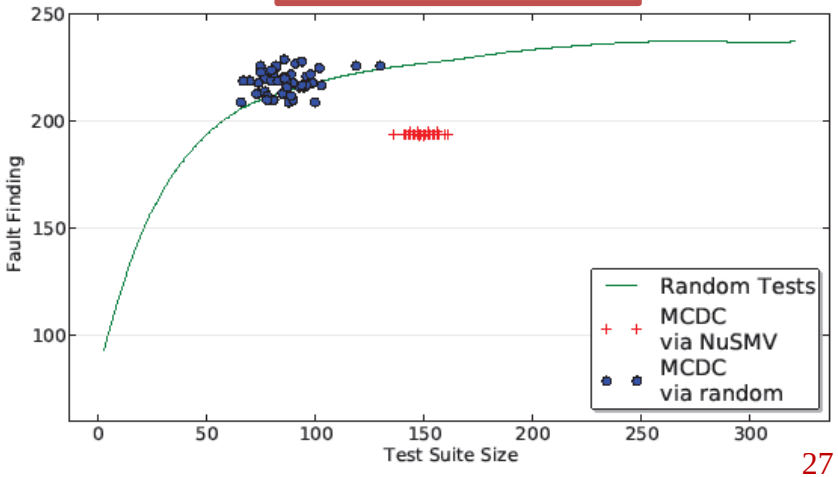


DWM_2

Vertmax_Batch



Latctl_Batch



MC/DC Coverage as Test Generation Target

- Model checker designed to generate understandable counterexamples
  - Simple, manipulate few variables
  - As short as possible

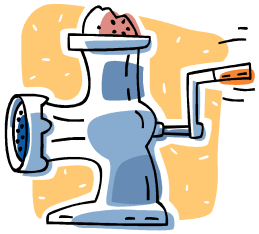
Counterexample Test

bool1	int1	bool2	bool3	int2
false	0	false	false	0
false	10	false	false	0

Random Test

bool1	int1	bool2	bool3	int2
false	-100	true	true	9164
false	10	false	false	16394
true	431	True	false	-7451
false	-1513	false	true	-1647

What We Need…



Generation Mechanism

Matt Staats, Michael W. Whalen, and Mats P.E. Heimdahl.
Programs, Tests, and Oracles: The Foundations of Testing Revisited.

Another Way to Look at MC/DC

- Masking MC/DC can be expressed:

$$(D(t_i) \neq D[true/c_n](t_i)) \wedge (D(t_j) \neq D[false/c_n](t_j))$$

Where $P[v/e_n]$ means, For program *P*, the computed value for the *nth* instance of expression *e* is replaced by value *v*

- Describes whether a condition is observable in a decision (i.e., not masked)
- **Problem**: we can rewrite programs to make decisions large or small (and MC/DC easy or hard to satisfy!)

Observable MC/DC

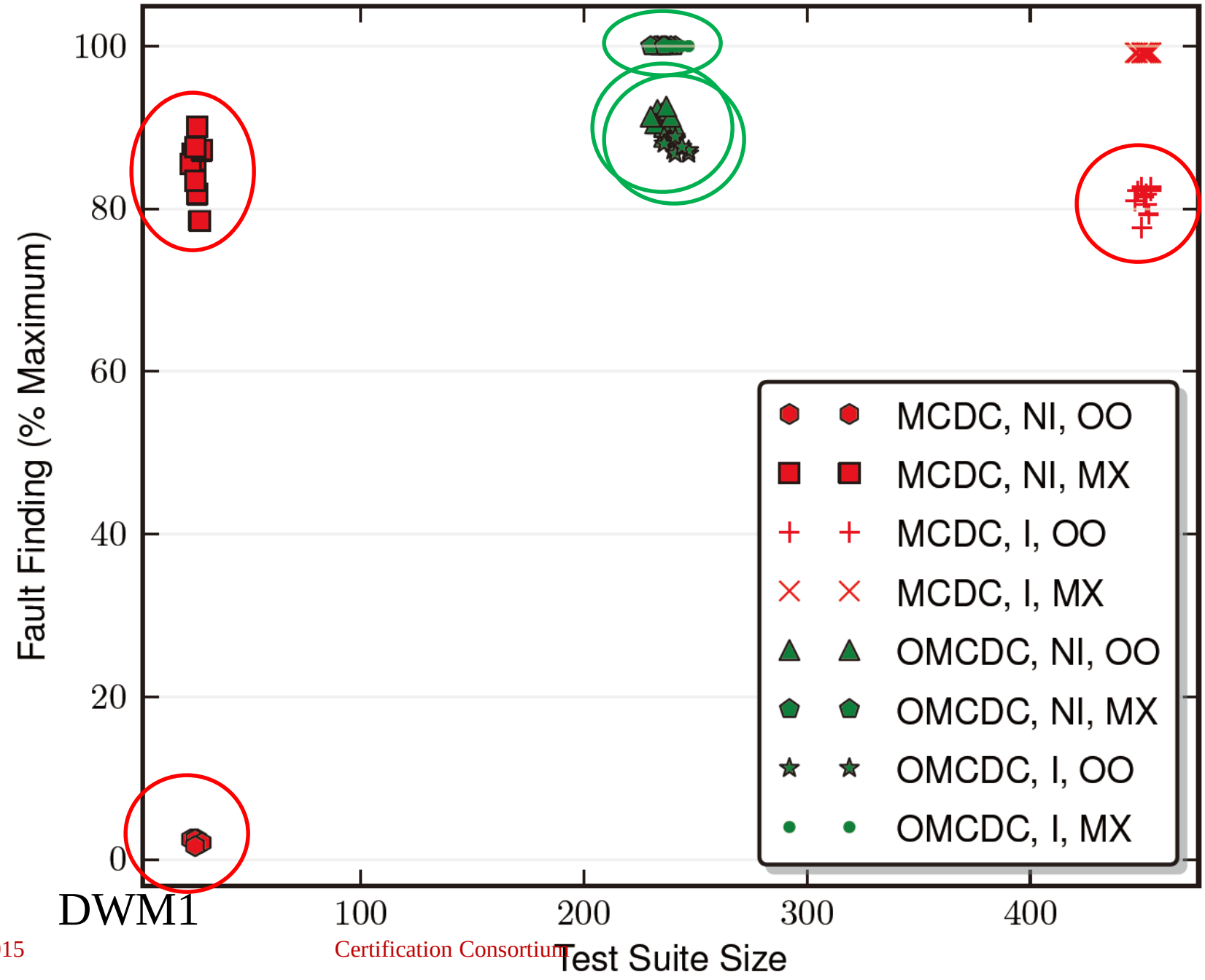
Michael W. Whalen, Gregory Gay, Dongjiang You, and Mats P.E. Heimdahl. Observable Modified Condition/Decision Coverage. Proceedings of the 35th ACM/IEEE International Conference on Software Engineering (ICSE'13). San Francisco, USA, May 2013.

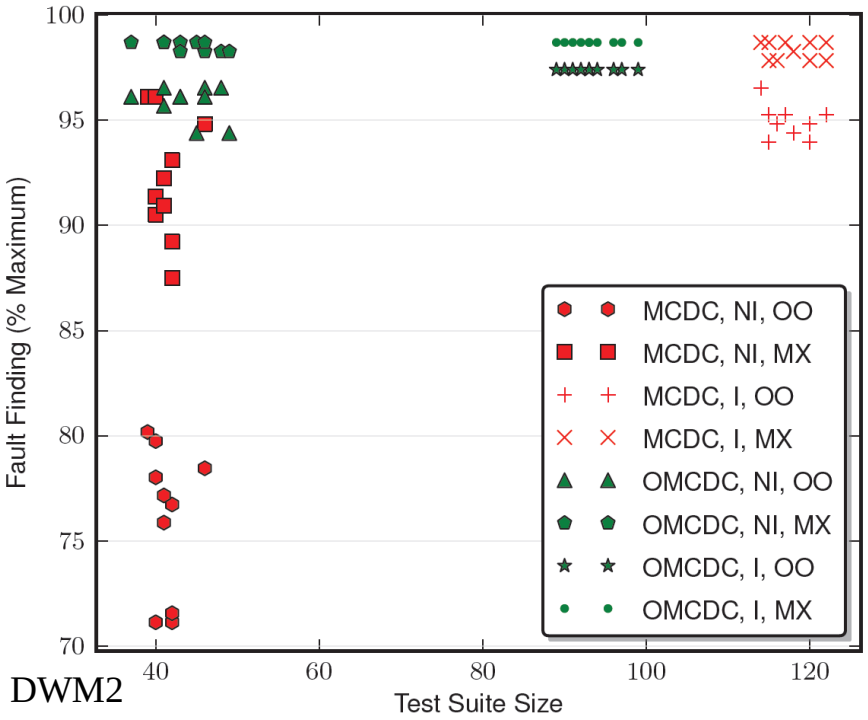
Idea: lift observability from decisions to programs

- Explicitly account for oracle
- Strength should be unaffected by simple program transformations (e.g., inlining)

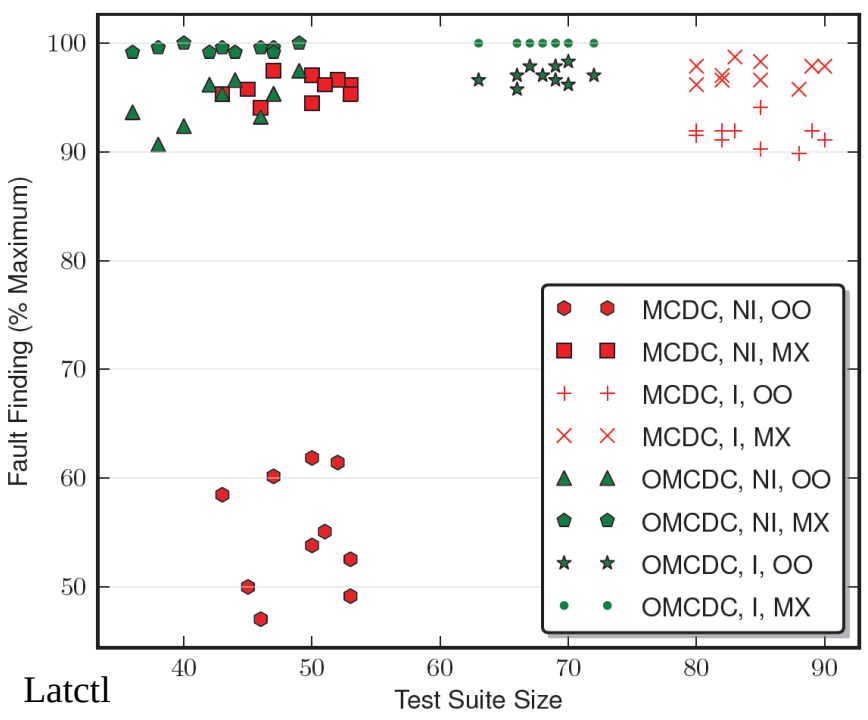
$$(\forall c_n \in Cond(P) \ .$$
$$(\exists t \in T \ . \ (P(t) \neq P[true/c_n](t))) \ \wedge$$
$$(\exists t \in T \ . \ (P(t) \neq P[false/c_n](t))))$$

where $Cond(P)$ is the set of all conditions in program P

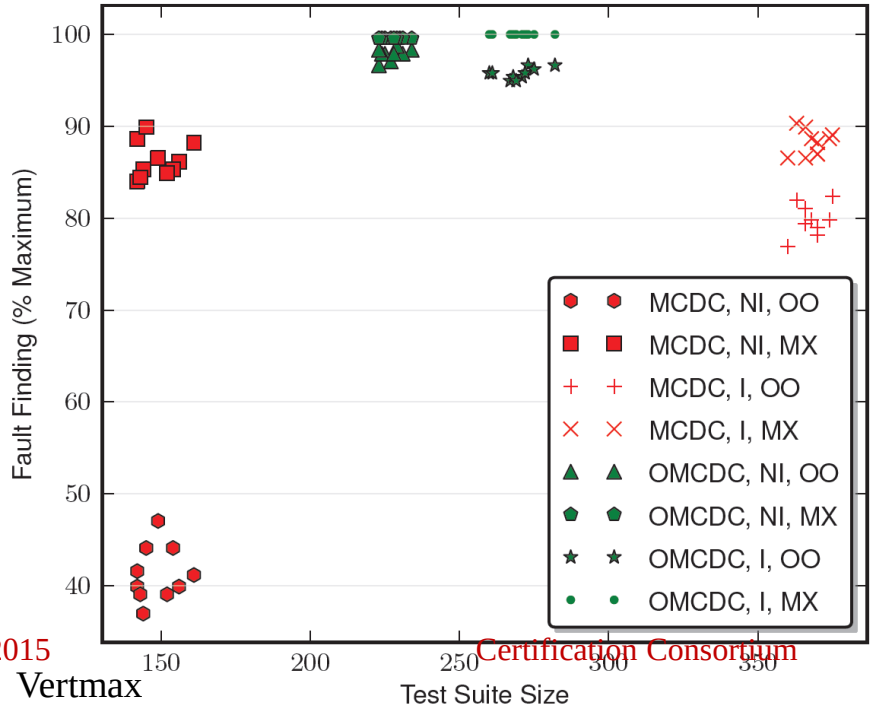

Fault Finding (% Maximum)
100
80
60
40
20
0
DWM1
100
200
300
400
Test Suite Size
MCDC, NI, OO
MCDC, NI, MX
MCDC, I, OO
MCDC, I, MX
OMCDC, NI, OO
OMCDC, NI, MX
OMCDC, I, OO
OMCDC, I, MX



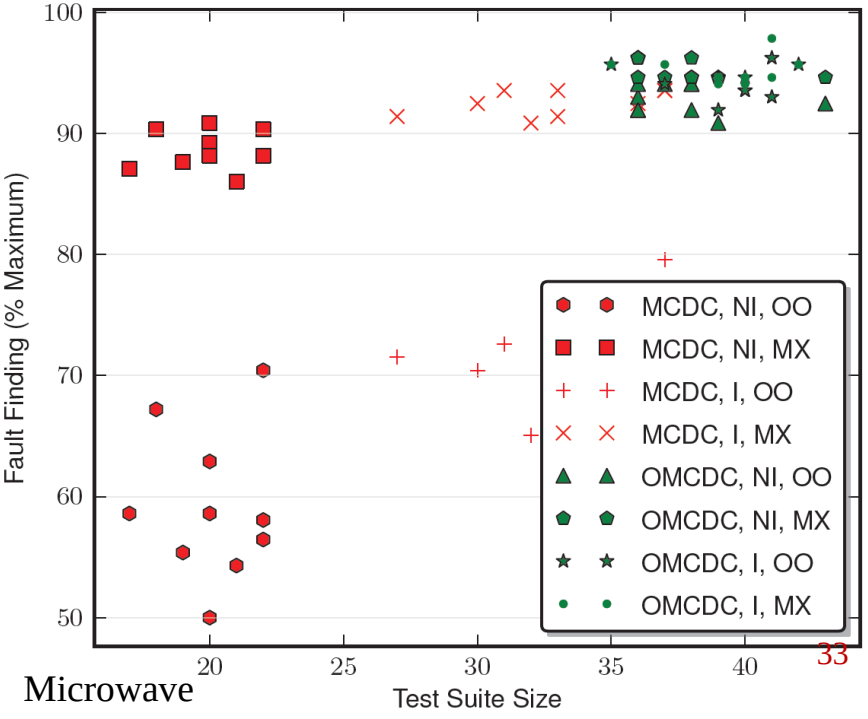
DWM2



Latctl



Vertmax



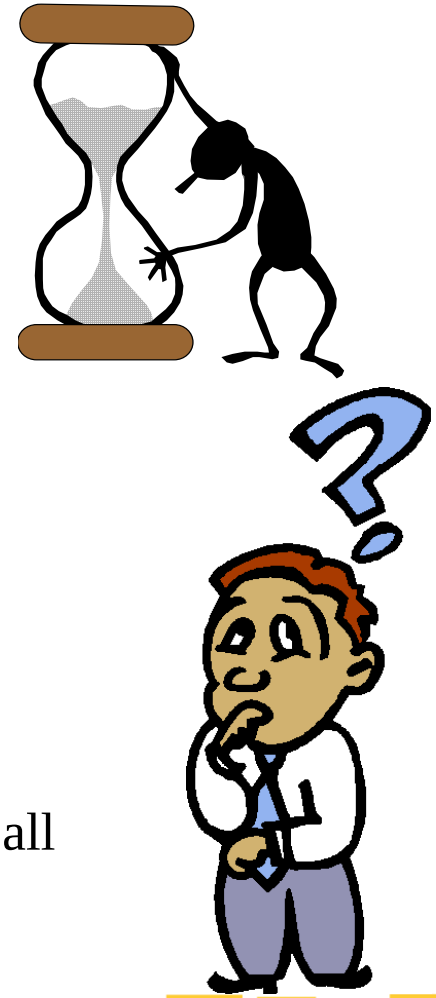
Microwave

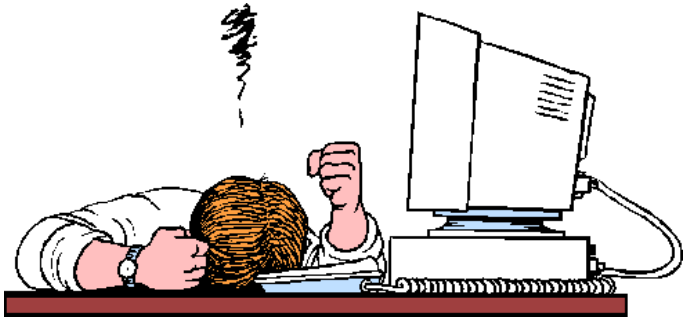
Achievable Obligations

	Structure	OMC/DC	MC/DC
DWM1	Non-Inlined	99.9%	100%
	Inlined	68.7%	98.1%
DWM2	Non-Inlined	89.8%	95.3%
	Inlined	**57.5%**	64.8%
Latctl	Non-Inlined	93.4%	100%
	Inlined	92.7%	99.6%
Vertmax	Non-Inlined	98.2%	100%
	Inlined	96.4%	99.1%
Microwave	Non-Inlined	**68.9%**	**98.9%**
	Inlined	72.2%	94.2%

What About Testing…..

- Statistical Testing
  - **Needs to be revisited**





- Coverage Criteria
  - **Will work better**

- Engineering Judgment
  - Maybe not so bad after all

Take Away Message

- We really do not have a good way of assessing adequacy of test effort
- Testing effectiveness is influenced by many factors
  - Interrelationship between Program, Specification, Test Set, and Oracle
- Potential benefits in examining other artifacts in software testing
  - Can we discover "good" combinations?
- **Much more work to be done!**

Discussion