Mutant Selection Problem

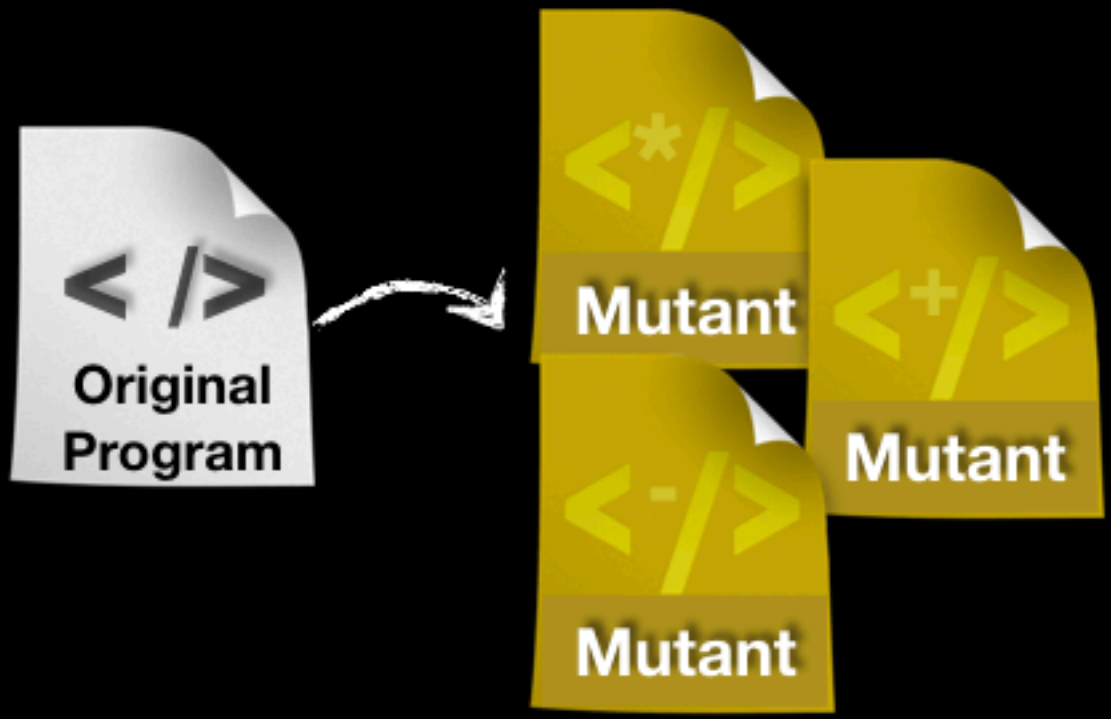
Mike Papadakis
University of Luxembourg
SnT Center

57th CREST Open Workshop

*Part of the presentation were made by Dr. Yue Jia

MUTATION ANALYSIS

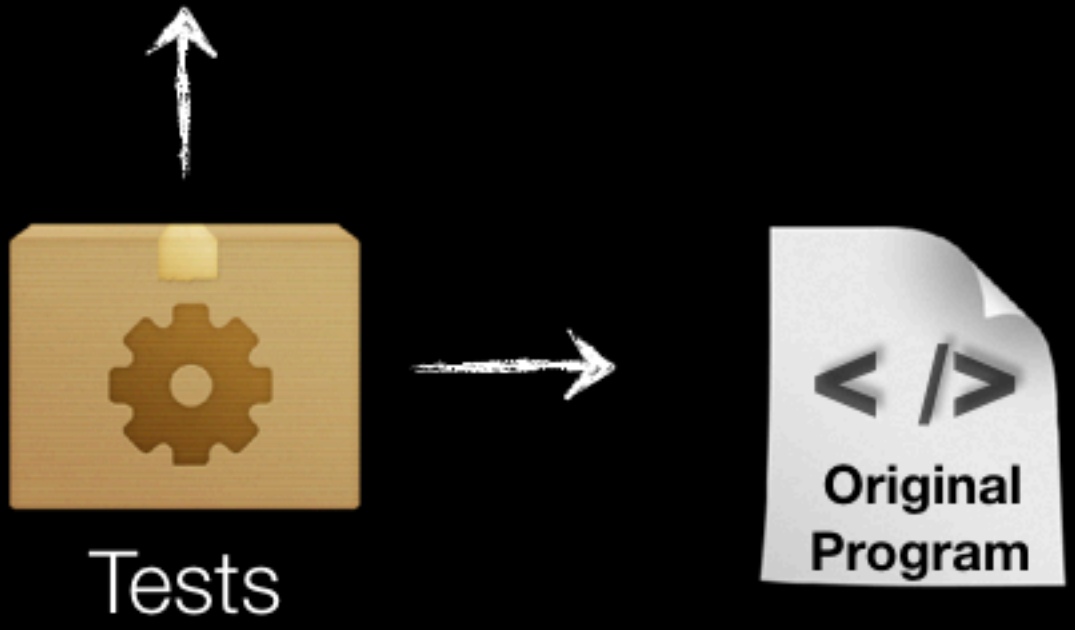
Mutation Testing



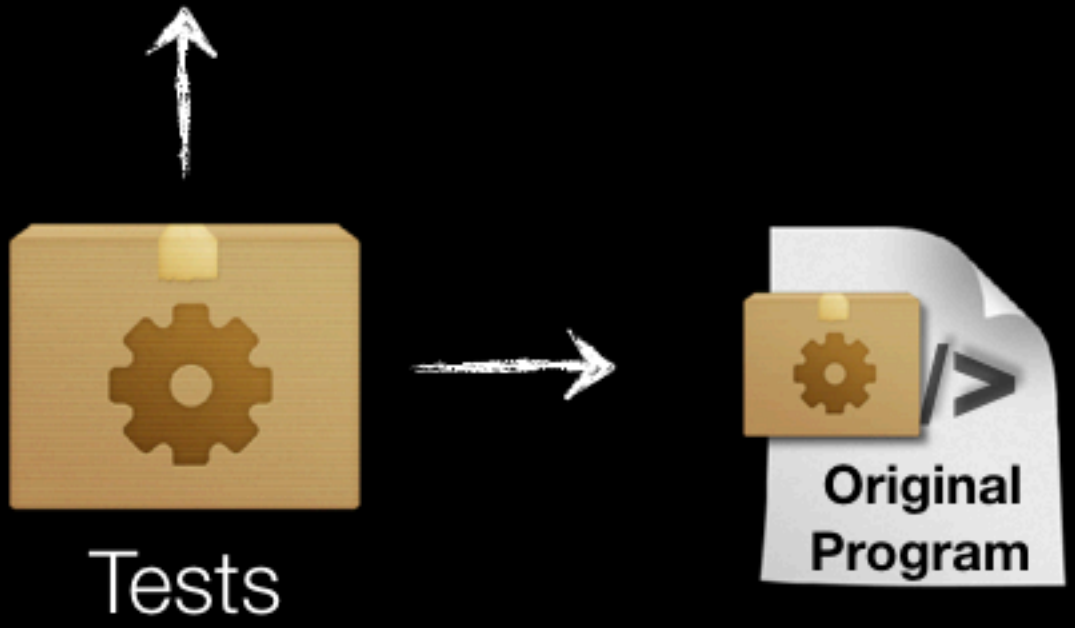

```
foo(int a, int b)
{
    if ( a < b )
        return a;
    else
        return b;
}
```

ROR: Relational Operator Replacement

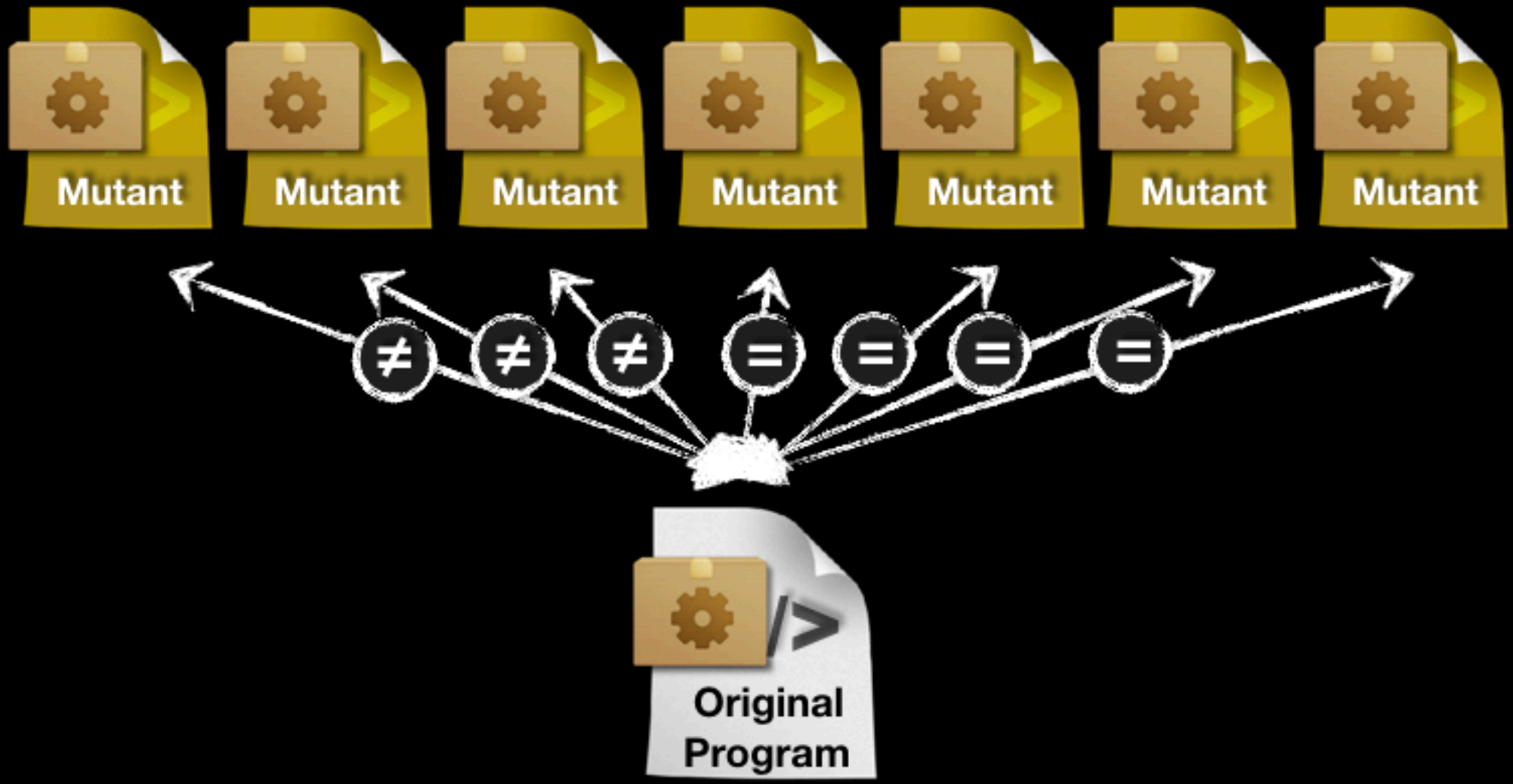
Code-based Mutation Testing

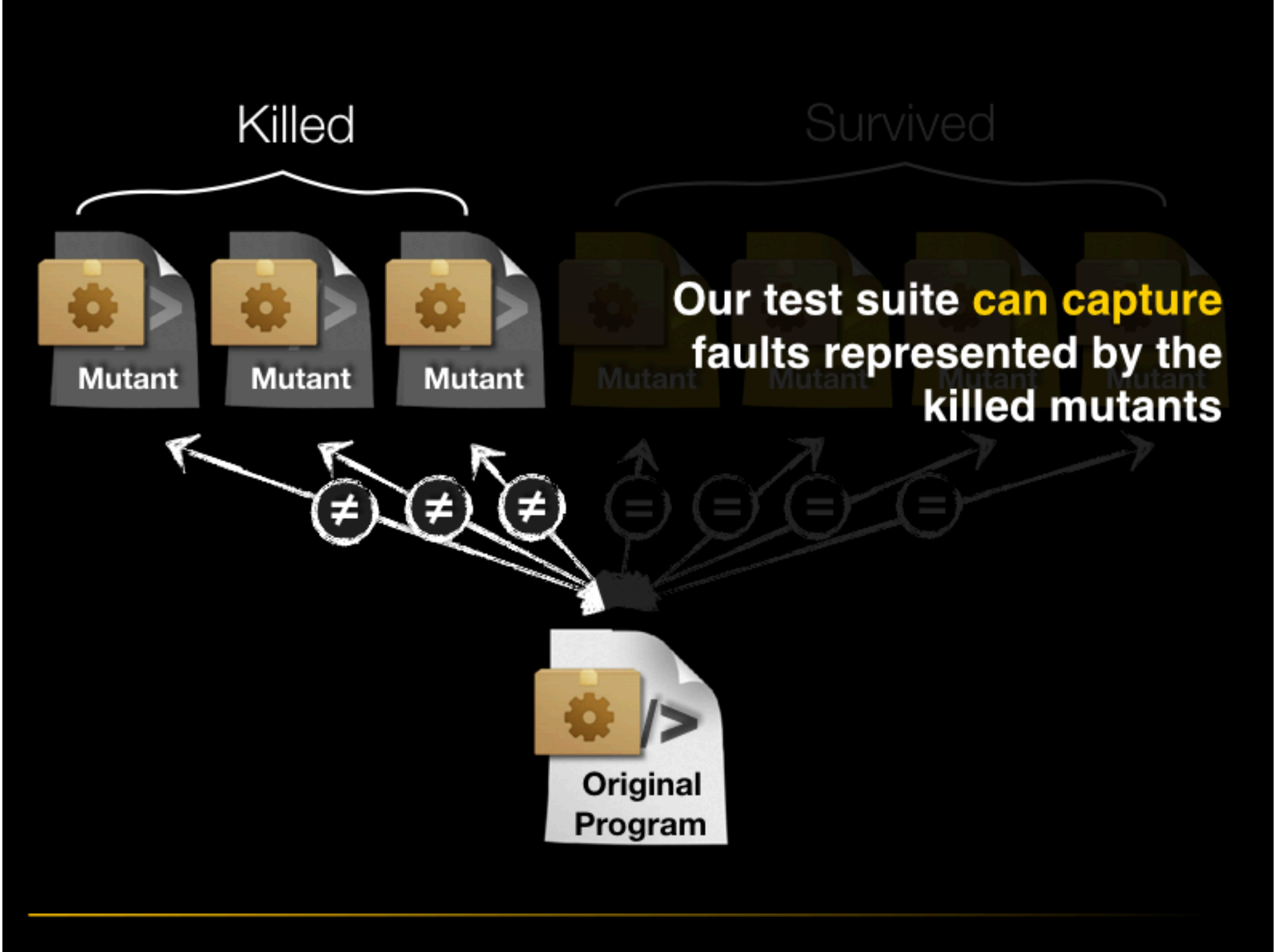


Code-based Mutation Testing



Code-based Mutation Testing

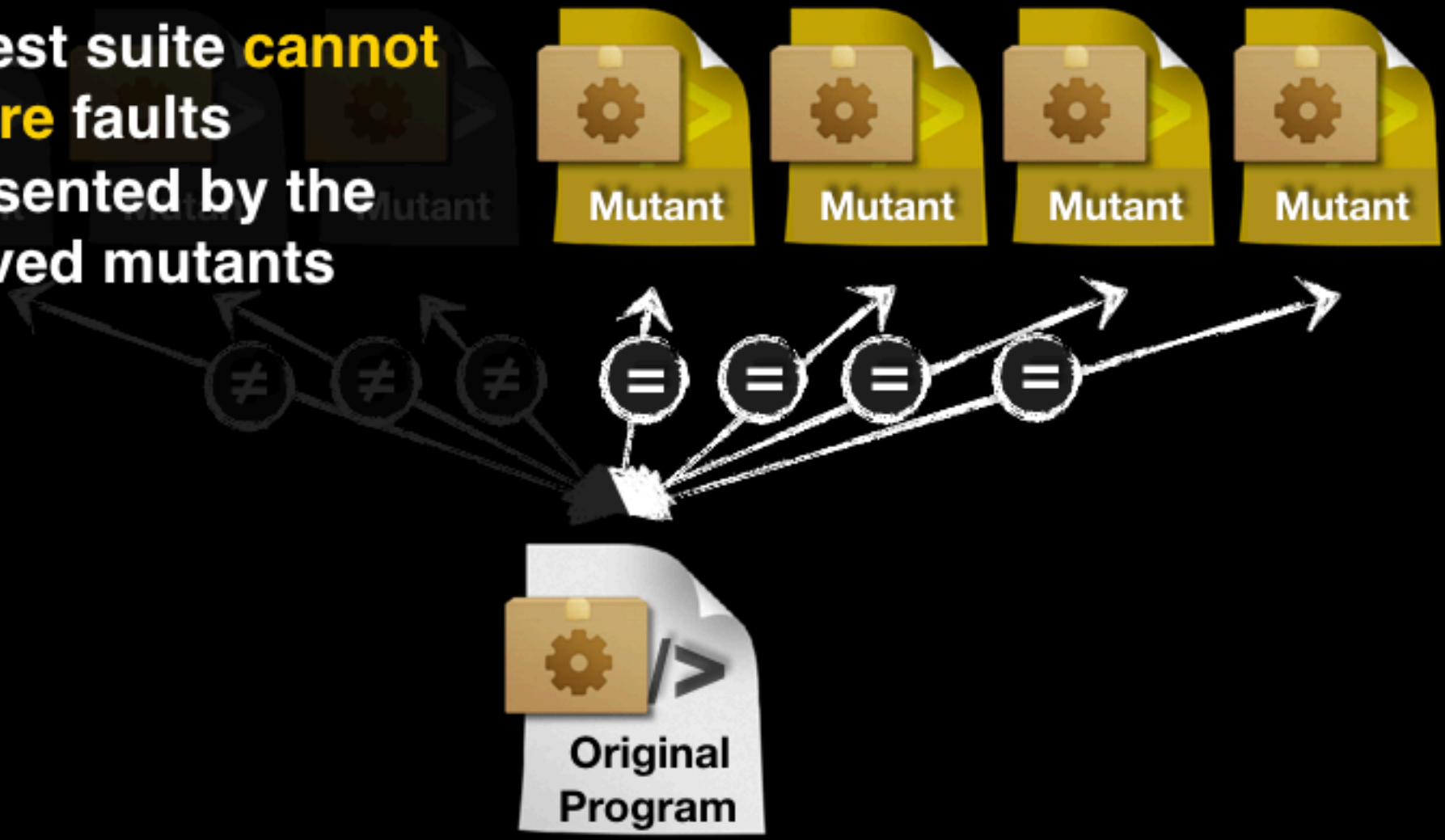


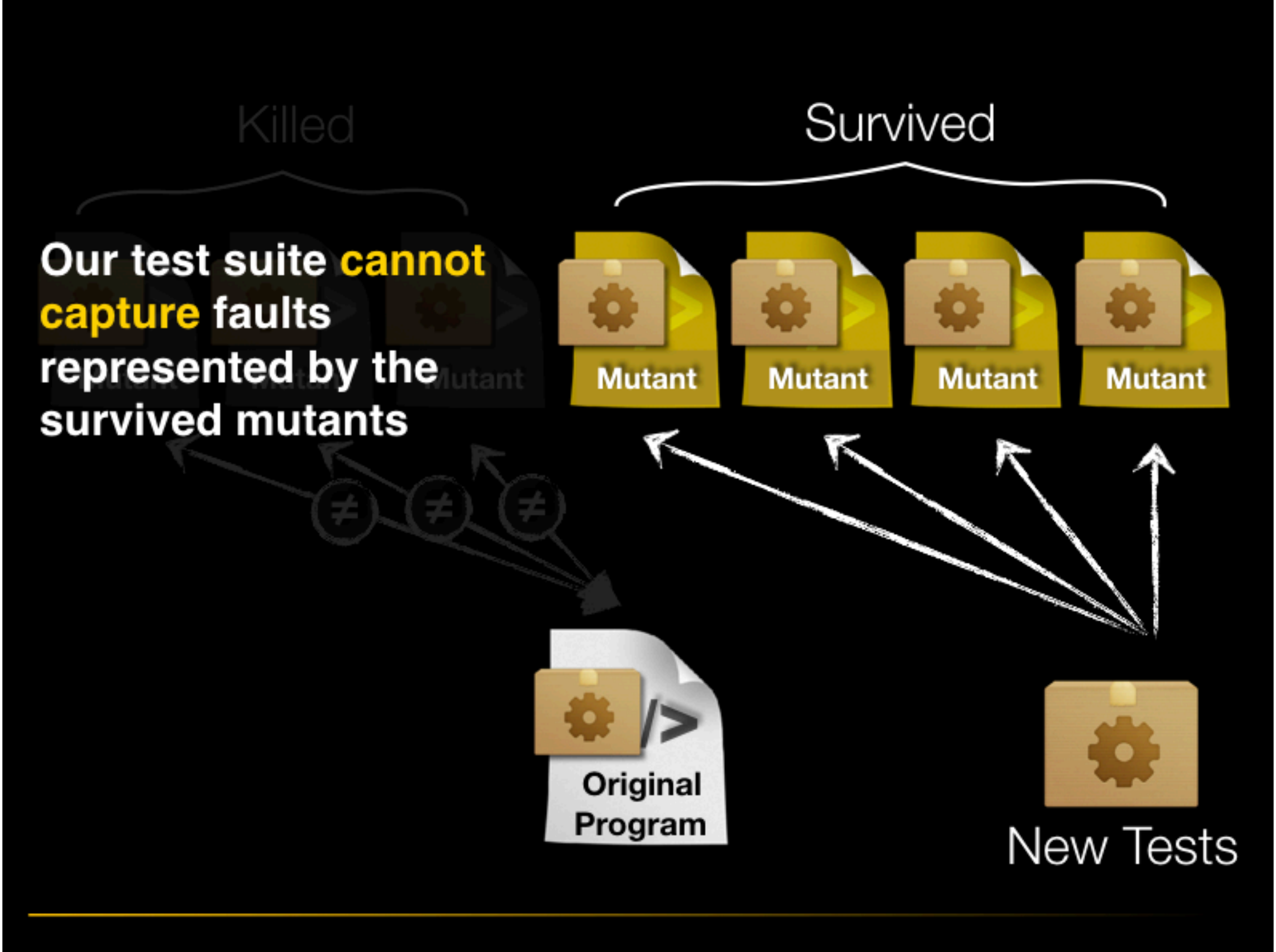

Killed
Survived
Mutant
Mutant
Mutant
Mutant
Mutant
Mutant
Our test suite can capture faults represented by the killed mutants
≠
≠
≠
=
=
=
=
Original Program

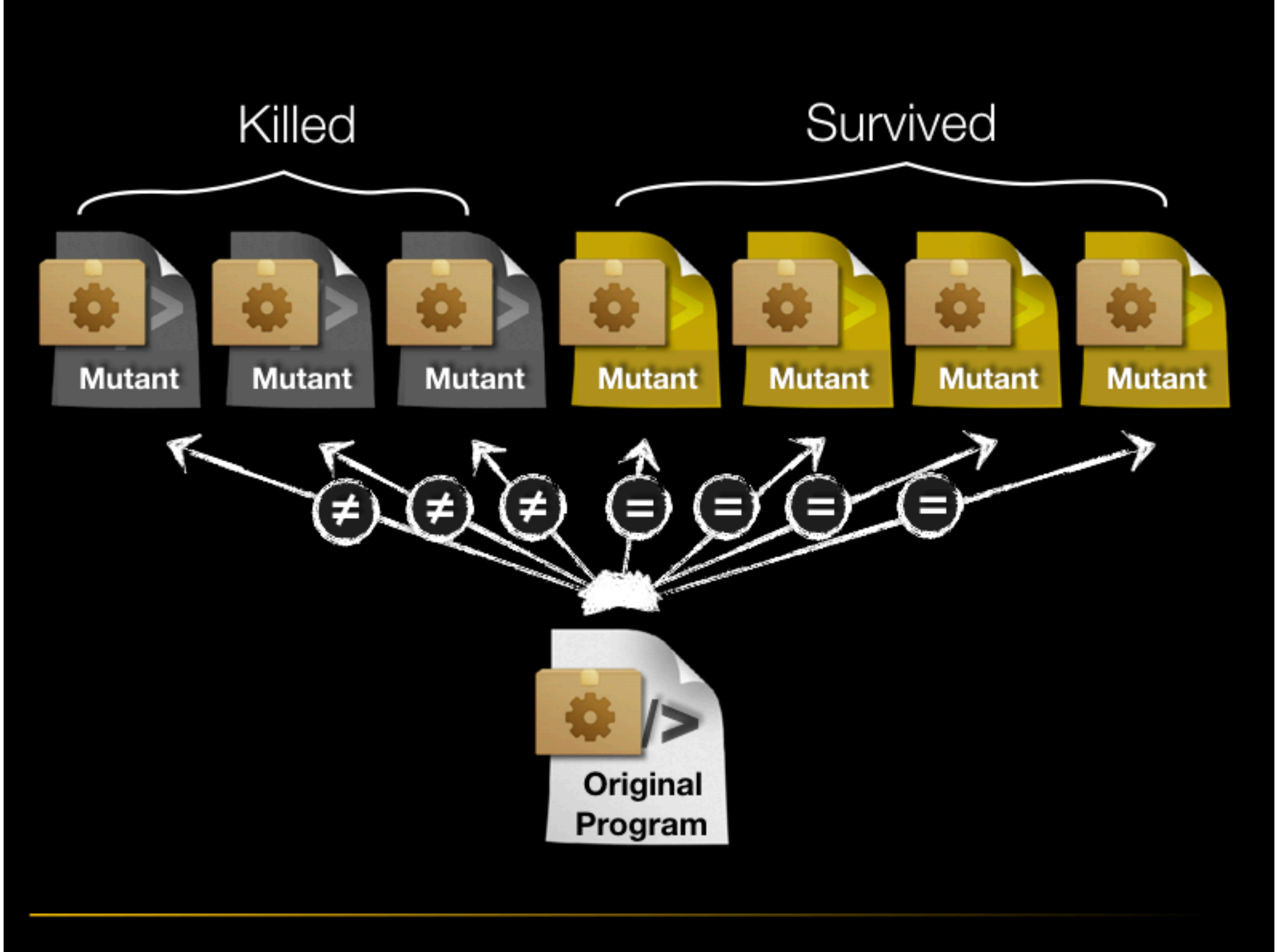
Killed

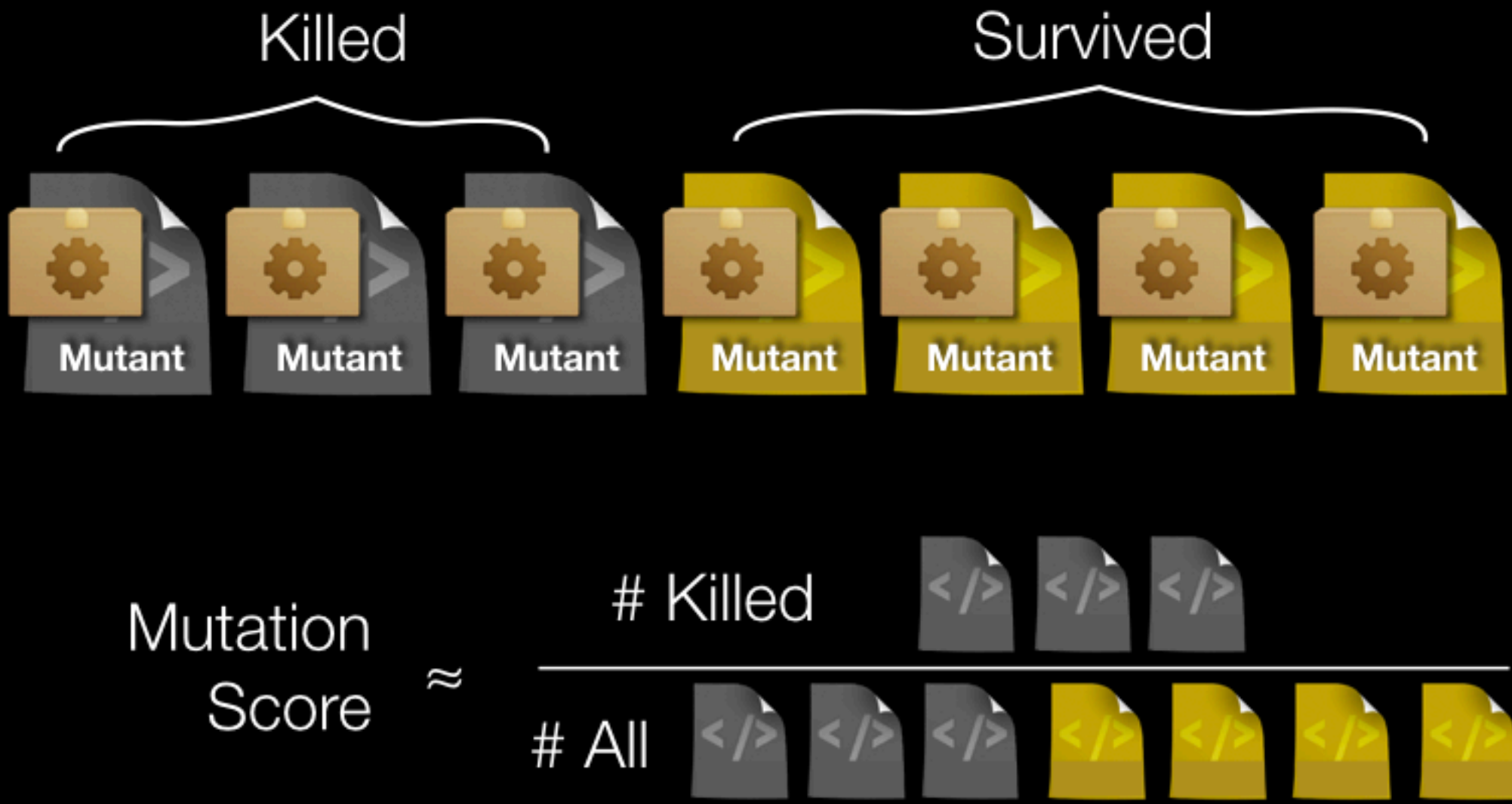
Survived

Our test suite cannot capture faults represented by the survived mutants




Killed
Survived
Our test suite cannot capture faults represented by the survived mutants
Mutant
Mutant
Mutant
Mutant
≠
≠
≠
Original Program
New Tests


Killed
Survived
Mutant
Mutant
Mutant
Mutant
Mutant
Mutant
Mutant
≠
≠
≠
=
=
=
=
Original
Program


Killed
Survived
Mutant
Mutant
Mutant
Mutant
Mutant
Mutant
Mutant
Mutation Score
≈
Killed
All
≈ 0.43

Example - Coordinated Universal Time

```
public static DateTimeZone forOffsetHoursMinutes(int hoursOffset, int minutesOffset)
        throws IllegalArgumentException {
```

```
/**
 * Gets a time zone instance for the specified offset to UTC in hours and minutes.
 * This method assumes 60 minutes in an hour, and standard length minutes.
 * <p>
 * This factory is a convenient way of constructing zones with a fixed offset.
 * The hours value must be in the range -23 to +23.
 * The minutes value must be in the range -59 to +59.
 * The following combinations of sign for the hour and minute are possible:
 * <pre>
 *  Hour    Minute    Example    Result
 *
 *  +ve     +ve       (2, 15)    +02:15
 *  +ve     zero      (2, 0)     +02:00
 *  +ve     -ve       (2, -15)   IllegalArgumentException
 *
 *  zero    +ve       (0, 15)    +00:15
 *  zero    zero      (0, 0)     +00:00
 *  zero    -ve       (0, -15)   -00:15
 *
 *  -ve     +ve       (-2, 15)   -02:15
 *  -ve     zero      (-2, 0)    -02:00
 *  -ve     -ve       (-2, -15)  -02:15
 * </pre>
```

*Apache Joda-Time

A **Real** fault!

```
public static DateTimeZone forOffsetHoursMinutes(int hoursOffset, int minutesOffset)
    throws IllegalArgumentException {
    if (hoursOffset == 0 && minutesOffset == 0) {
        return DateTimeZone.UTC;
    }
    if (hoursOffset < -23 || hoursOffset > 23) {
        throw new IllegalArgumentException("Hours out of range: " + hoursOffset);
    }
    if (minutesOffset < 0 || minutesOffset > 59) {
        throw new IllegalArgumentException("Minutes out of range: " + minutesOffset);
    }
    int offset = 0;
    try {
        int hoursInMinutes = hoursOffset * 60;
        if (hoursInMinutes < 0) {
            minutesOffset = hoursInMinutes - minutesOffset;
        } else {
            minutesOffset = hoursInMinutes + minutesOffset;
        }
        offset = FieldUtils.safeMultiply(minutesOffset, DateTimeConstants.MILLIS_PER_MINUTE);
    } catch (ArithmeticException ex) {
        throw new IllegalArgumentException("Offset is too large");
    }
    return forOffsetMillis(offset);
}
```

How Developers **fixed**?

```
public static DateTimeZone forOffsetHoursMinutes(int hoursOffset, int minutesOffset)
    throws IllegalArgumentException {
    if (hoursOffset == 0 && minutesOffset == 0) {
        return DateTimeZone.UTC;
    }
    if (hoursOffset < -23 || hoursOffset > 23) {
        throw new IllegalArgumentException("Hours out of range: " + hoursOffset);
    }
    if (minutesOffset < 0 || minutesOffset > 59) {
        throw new IllegalArgumentException("Minutes out of range: " + minutesOffset);
    }
    int offse
    try {
        int
        if (
        } el
```

Let's **test** this program

```
-       if (m
+       if (minutesOffset < -59 || minutesOffset > 59) {
            throw new IllegalArgumentException("Minutes out of range: " + minutesOffset);
        }
+       if (hoursOffset > 0 && minutesOffset < 0) {
+           throw new IllegalArgumentException("Positive hours must not have negative minu
+       }
        int offset = 0;
        try {
            int hoursInMinutes = hoursOffset * 60;
            if (hoursInMinutes < 0) {
-               minutesOffset = hoursInMinutes - minutesOffset;
+               minutesOffset = hoursInMinutes - Math.abs(minutesOffset);
            } else {
```

What **Coverage** Can Do?

```
public static DateTimeZone forOffsetHoursMinutes(int hoursOffset, int minutesOffset) thr
    if (hoursOffset == 0 && minutesOffset == 0) {
        return DateTimeZone.UTC;
    }
    if (hoursOffset < -23 || hoursOffset > 23) {
        throw new IllegalArgumentException("Hours out of range: " + hoursOffset);
    }
    if (minutesOffset < 0 || minutesOffset > 59) {
        throw new IllegalArgumentException("Minutes out of range: " + minutesOffset);
    }
    int offset = 0;
    try {
        int hoursInMinutes = hoursOffset * 60;
        if (hoursInMinutes < 0) {
            minutesOffset = hoursInMinutes - minutesOffset;
        } else {
            minutesOffset = hoursInMinutes + minutesOffset;
        }
        offset = FieldUtils.safeMultiply(minutesOffset, DateTimeConstants.MILLIS_PER_MIN
    } catch (ArithmeticException ex) {
        throw new IllegalArgumentException("Offset is too large");
    }
    return forOffsetMillis(offset);
}
```

The branch-adequate tests **CANNOT** find this bug!

DC = 100%

What about **Mutation?**

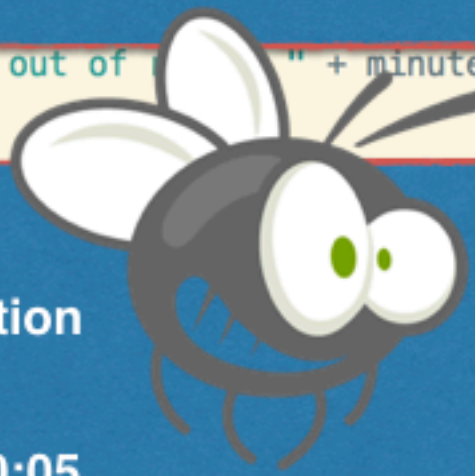
```
272        public static DateTimeZone forOffsetHoursMinutes(int hoursOffset, int minutesOffset) th
273  26        if (hoursOffset == 0 && minutesOffset == 0) {
274  1             return DateTimeZone.UTC;
275            }
276  42        if (hoursOffset < -23 || hoursOffset > 23) {
277  10            throw new IllegalArgumentException("Hours out of range: " + hoursOffset);
278            }
279  35        if (minutesOffset < 0 || minutesOffset > 59) {
280  10            throw new IllegalArgumentException("Minutes out of range: " + minutesOffset);
281            }
282  3         int offset = 0;
283            try {
284  19            int hoursInMinutes = hoursOffset * 60;
285  14            if (hoursInMinutes < 0) {
286  17                minutesOffset = hoursInMinutes - minutesOffset;
287                } else {
288  17                minutesOffset = hoursInMinutes + minutesOffset;
289                }
290  14            offset = FieldUtils.safeMultiply(minutesOffset, DateTimeConstants.MILLIS_PER_MI
291            } catch (ArithmeticException ex) {
292  1             throw new IllegalArgumentException("Offset is too large");
293            }
294  7         return forOffsetMillis(offset);
295        }
```

MS = 78%

```
public static DateTimeZone                                    int hoursOffset, int minutesOffset)
    throws IllegalArgumentException {
    if (hoursOffset == 0 && minutesOffset == 0) {
        return DateTimeZone.UTC;
    }
    if (hoursOffset < -23 || hoursOffse
        throw new IllegalArgumentExcept
    }
    if (minutesOffset < 0 || minutesOff
        throw new IllegalArgumentException("Minutes out of                " + minutesOffset);
    }
    int offset = 0;
    try {
        in
        if
            }
        }
        o                                                                    ER_MINUTE
    } cat
        th
    }
    return
```

minutesOffset <= 0

DateTimeZone.forOffsetHoursMinutes(0, -5)

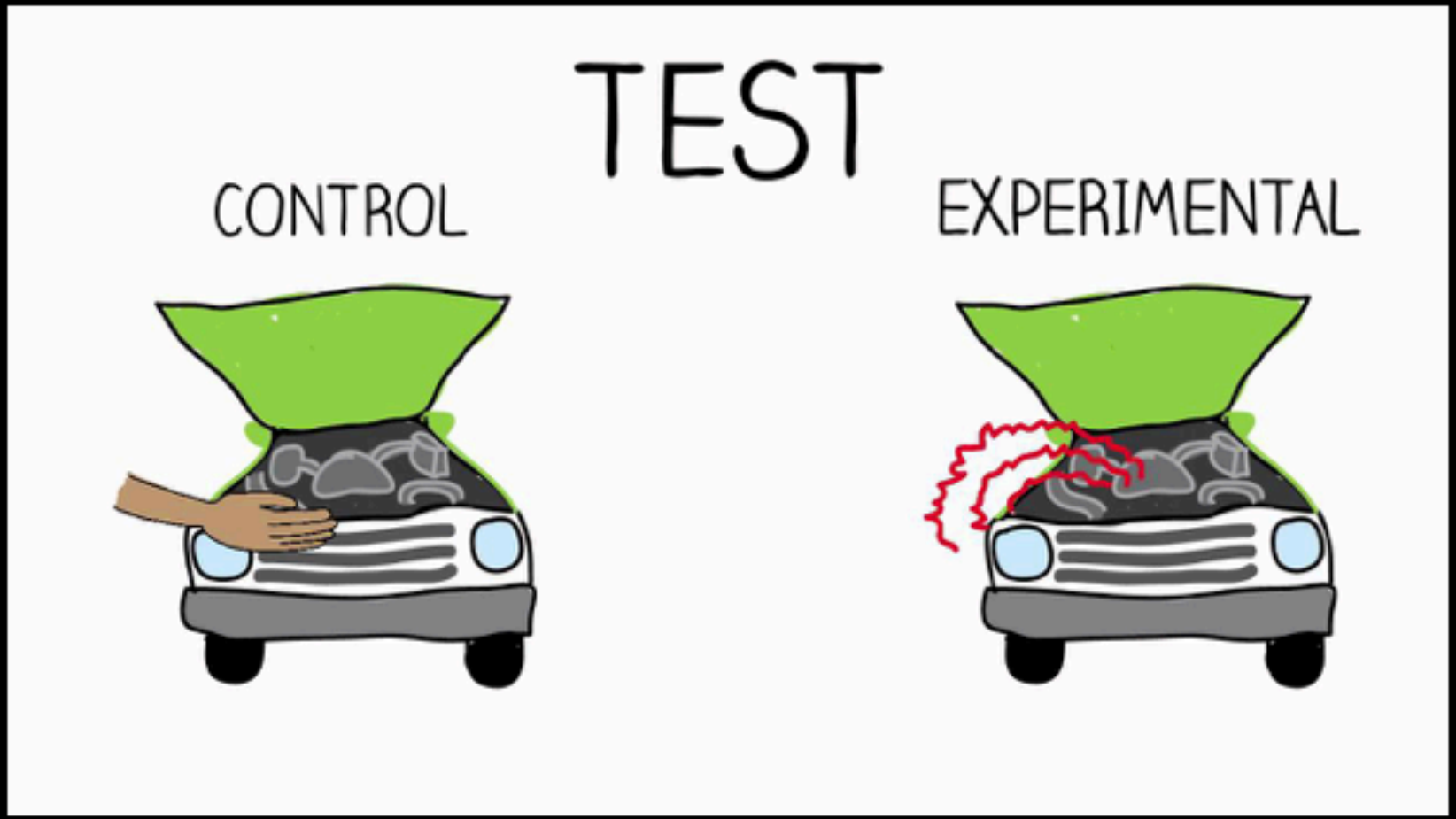


➡Program under test: **throws exception**

➡Mutant: UTC

The correct program would return **-00:05**

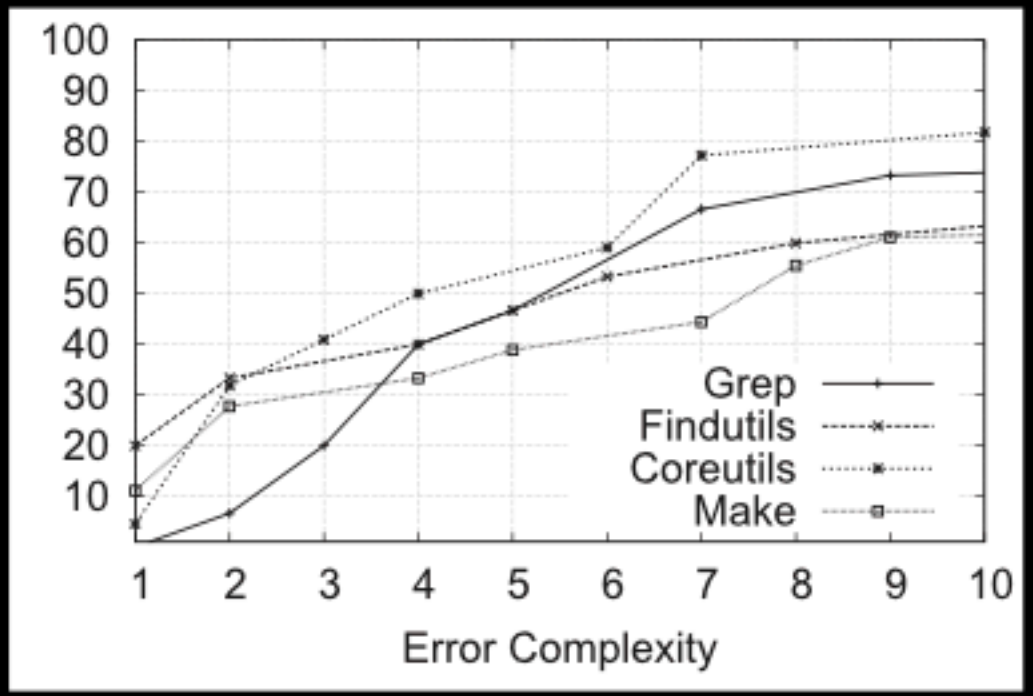
Experiment



CoREBench: Realisitic, Complex Errors

Subject	Size in kLoC	Maturity 1st commit	#Commits total (last year)	#Tests	#Bug Reports marked fixed	Extract. Period recent 1k commits	#RErrors extracted
Coreutils	83.1	Oct. 1992	27,807 (290)	4772	832	08.05.11 – 06.10.13	**22**
Findutils	18.0	Feb. 1996	2,031 (43)	1054	312	01.08.05 – 26.10.13	**15**
Grep	9.4	Nov. 1989	1,307 (31)	1582	66	25.09.01 – 26.10.13	**15**
Make	35.3	Apr. 1988	2,288 (134)	528	305	01.03.96 – 24.11.13	**18**

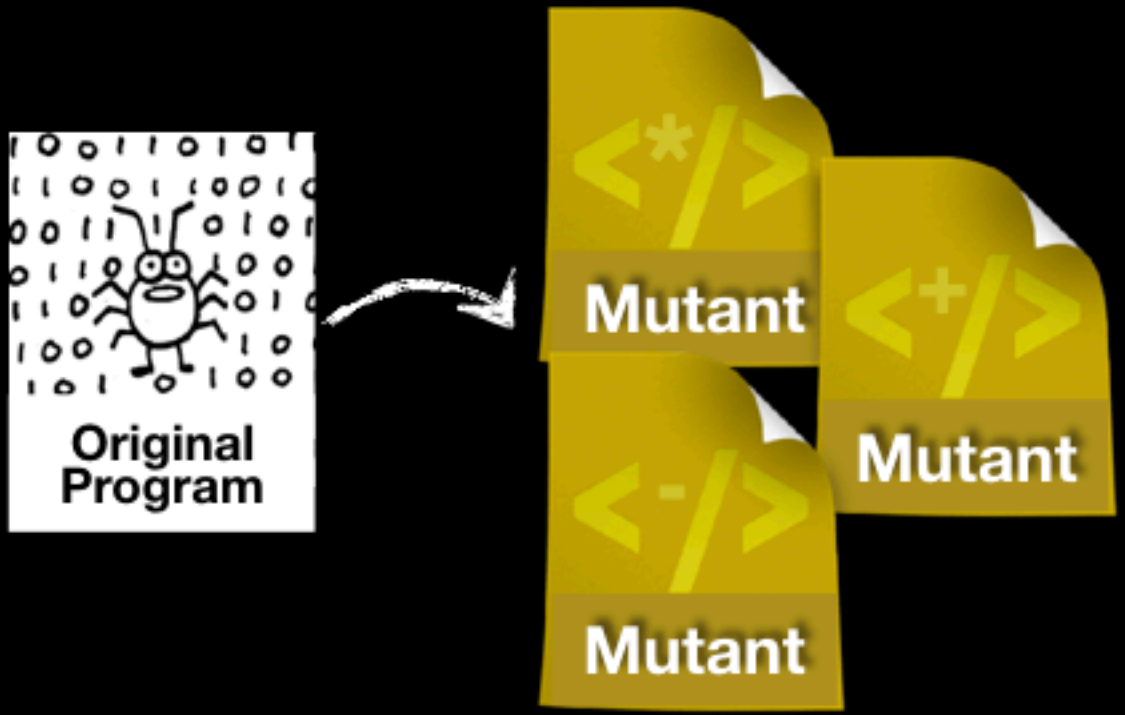
CoREBench



Bohme et al. **"CoREBench: Studying Complexity of Regression Errors"**, ISSTA'14
http://www.comp.nus.edu.sg/~release/corebench/

Fault Revelation Ability

Subject	Size in kLoC	Maturity 1st commit	#Commits total (last year)	#Tests	#Bug Reports marked fixed	Extract. Period recent 1k commits	#RErrors extracted
Coreutils	83.1	Oct. 1992	27,807 (290)	4772	832	08.05.11 – 06.10.13	**22**
Findutils	18.0	Feb. 1996	2,031 (43)	1054	312	01.08.05 – 26.10.13	**15**
Grep	9.4	Nov. 1989	1,307 (31)	1582	66	25.09.01 – 26.10.13	**15**
Make	35.3	Apr. 1988	2,288 (134)	528	305	01.03.96 – 24.11.13	**18**

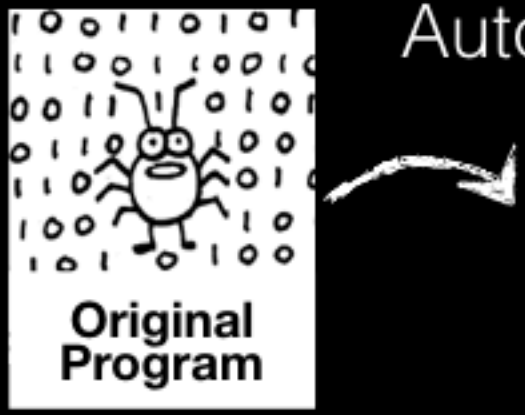


Fault Revelation Ability

Subject	Size in kLoC	Maturity 1st commit	#Commits total (last year)	#Tests	#Bug Reports marked fixed	Extract. Period recent 1k commits	#RErrors extracted
Coreutils	83.1	Oct. 1992	27,807 (290)	4772	832	08.05.11 – 06.10.13	**22**
Findutils	18.0	Feb. 1996	2,031 (43)	1054	312	01.08.05 – 26.10.13	**15**
Grep	9.4	Nov. 1989	1,307 (31)	1582	66	25.09.01 – 26.10.13	**15**
Make	35.3	Apr. 1988	2,288 (134)	528	305	01.03.96 – 24.11.13	**18**

Automated Test Gen

Developer Tests



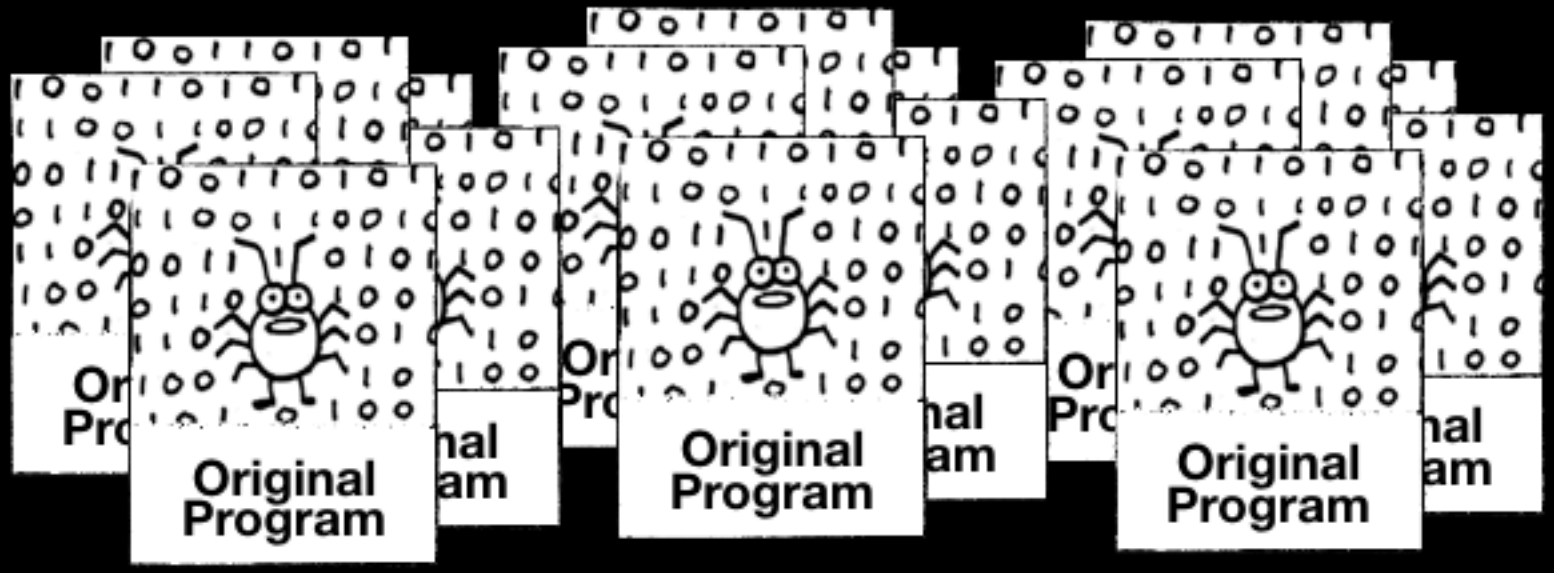
Manual Test Gen

Fault Revelation Ability

Mutants, Branches, statements

Test pool

Fault Revelation Ability

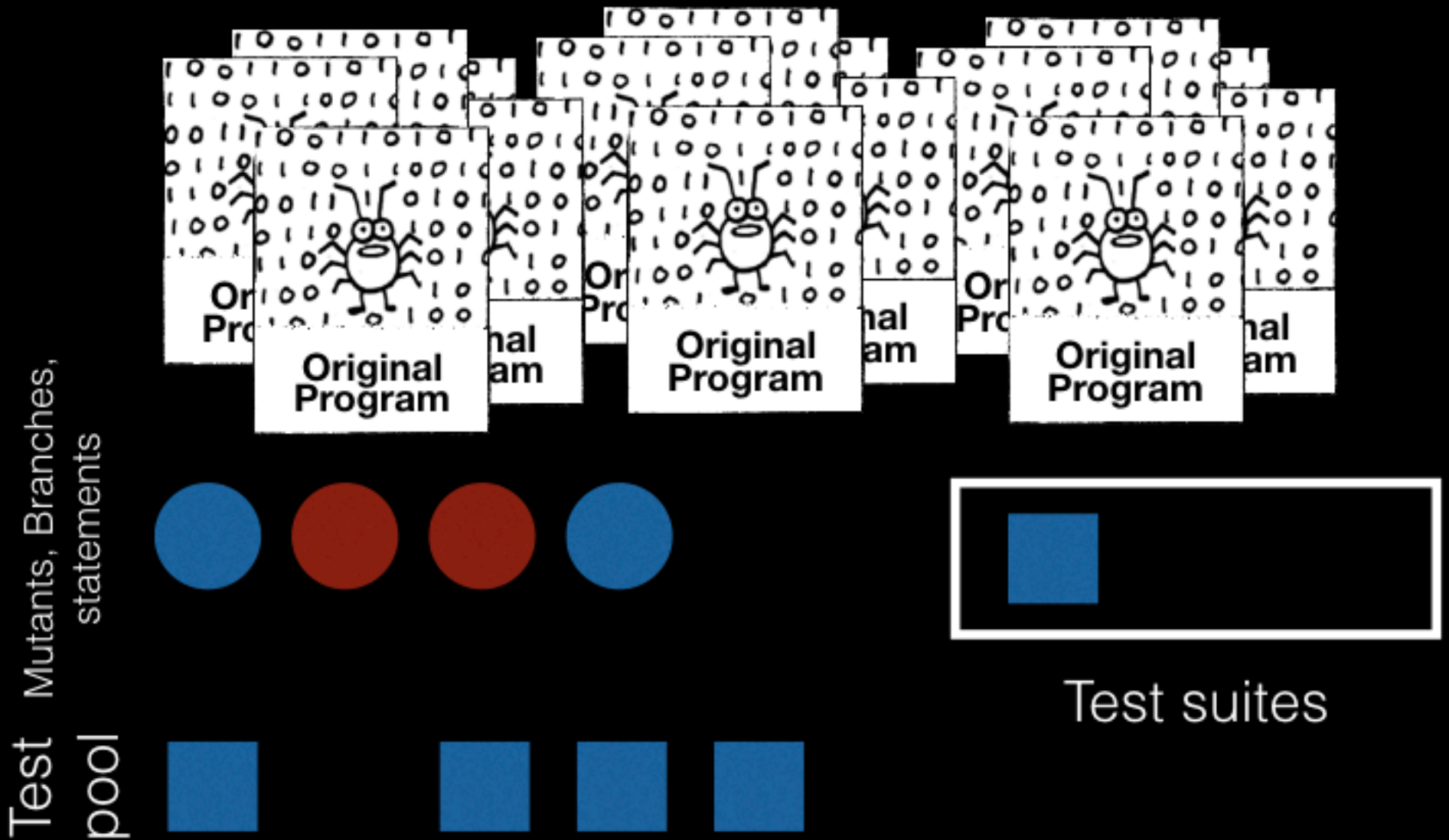


Test pool

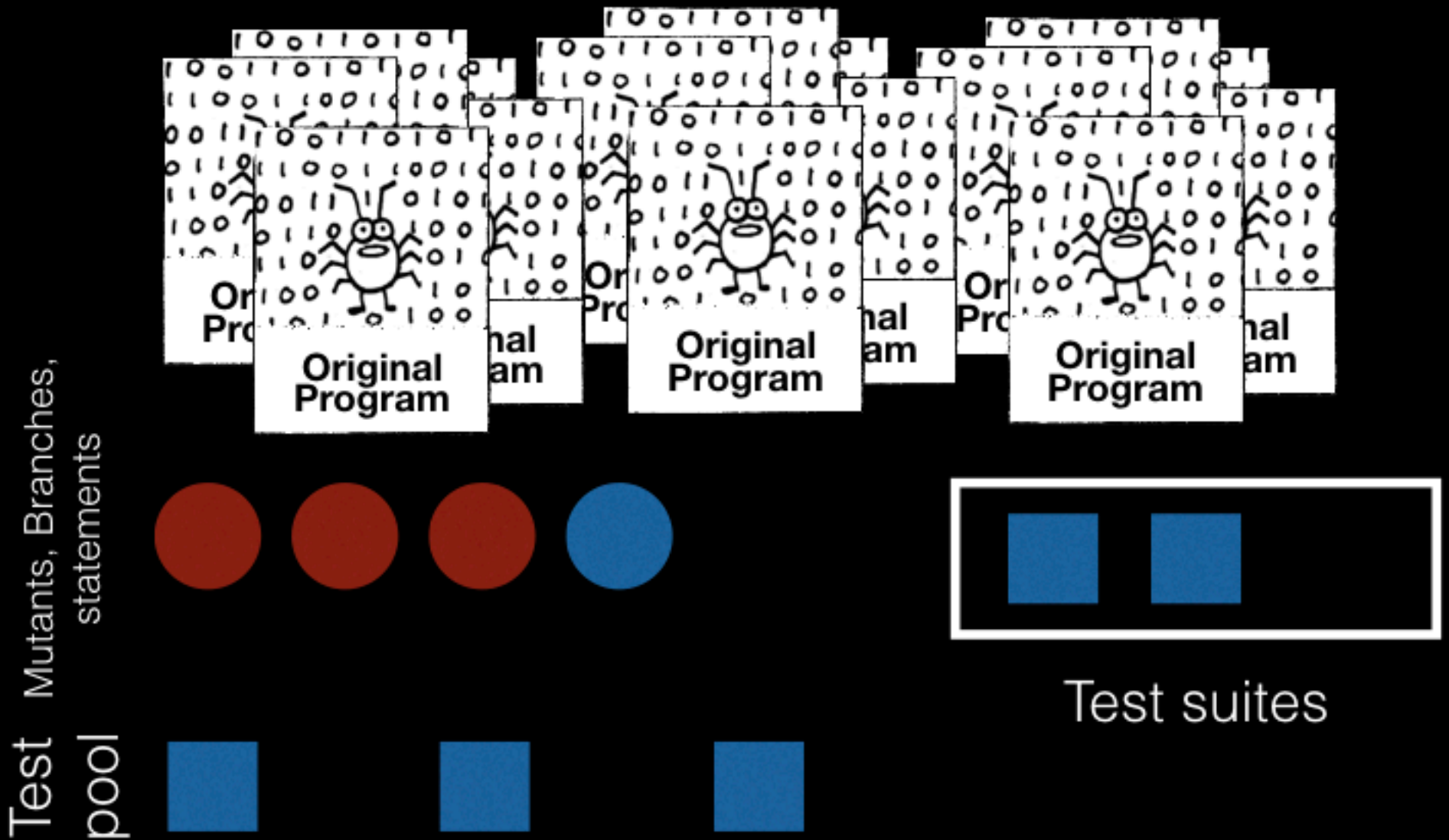
Mutants, Branches, statements

Test suites

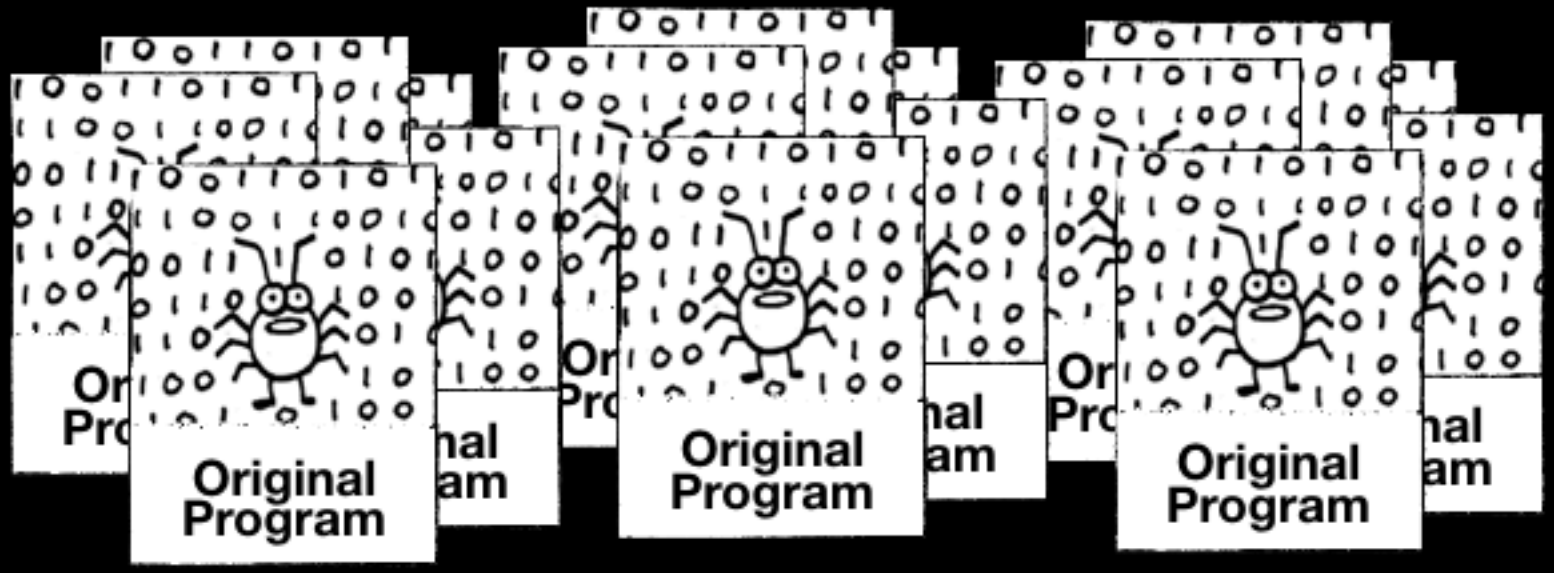
Fault Revelation Ability



Fault Revelation Ability



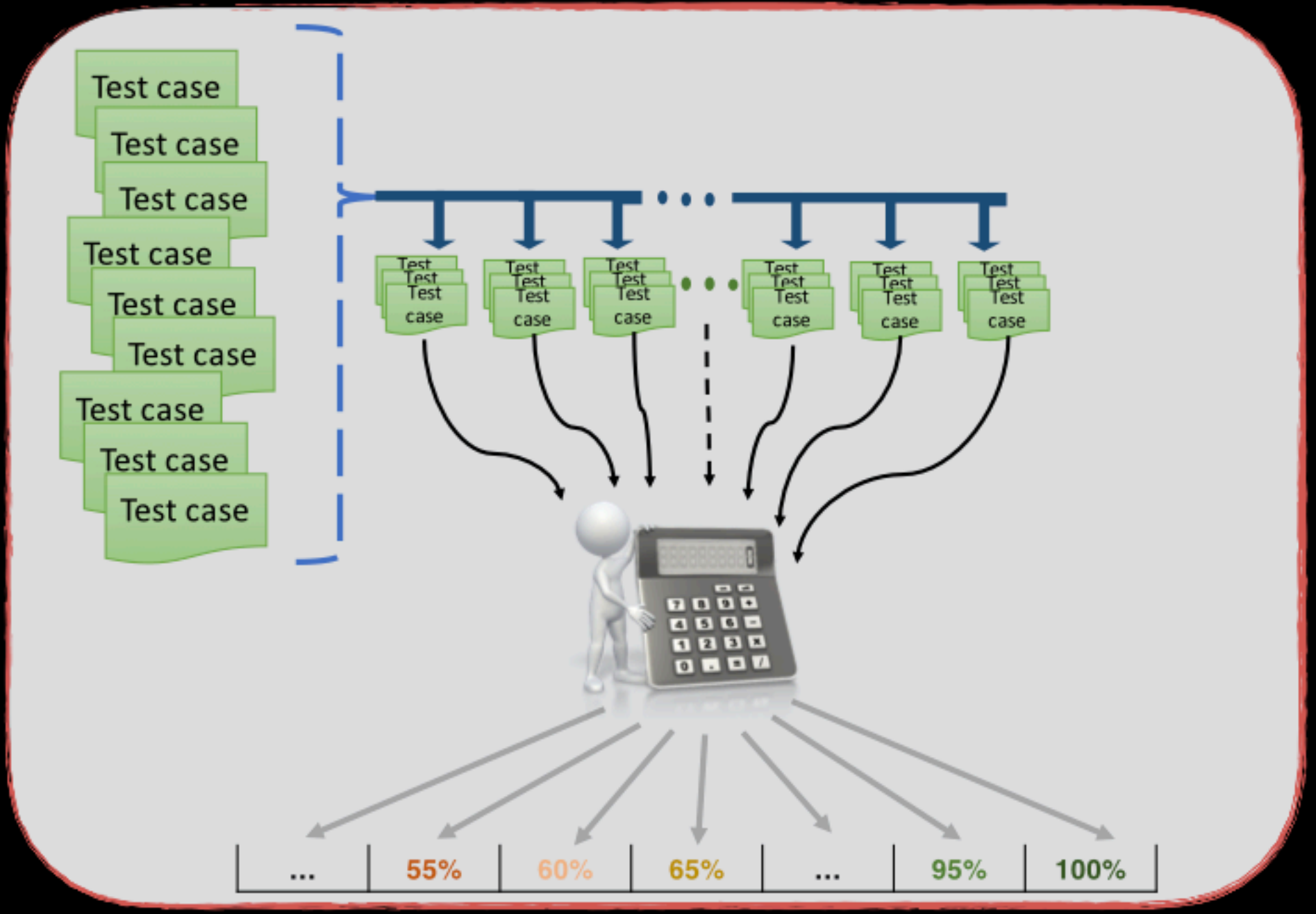
Fault Revelation Ability

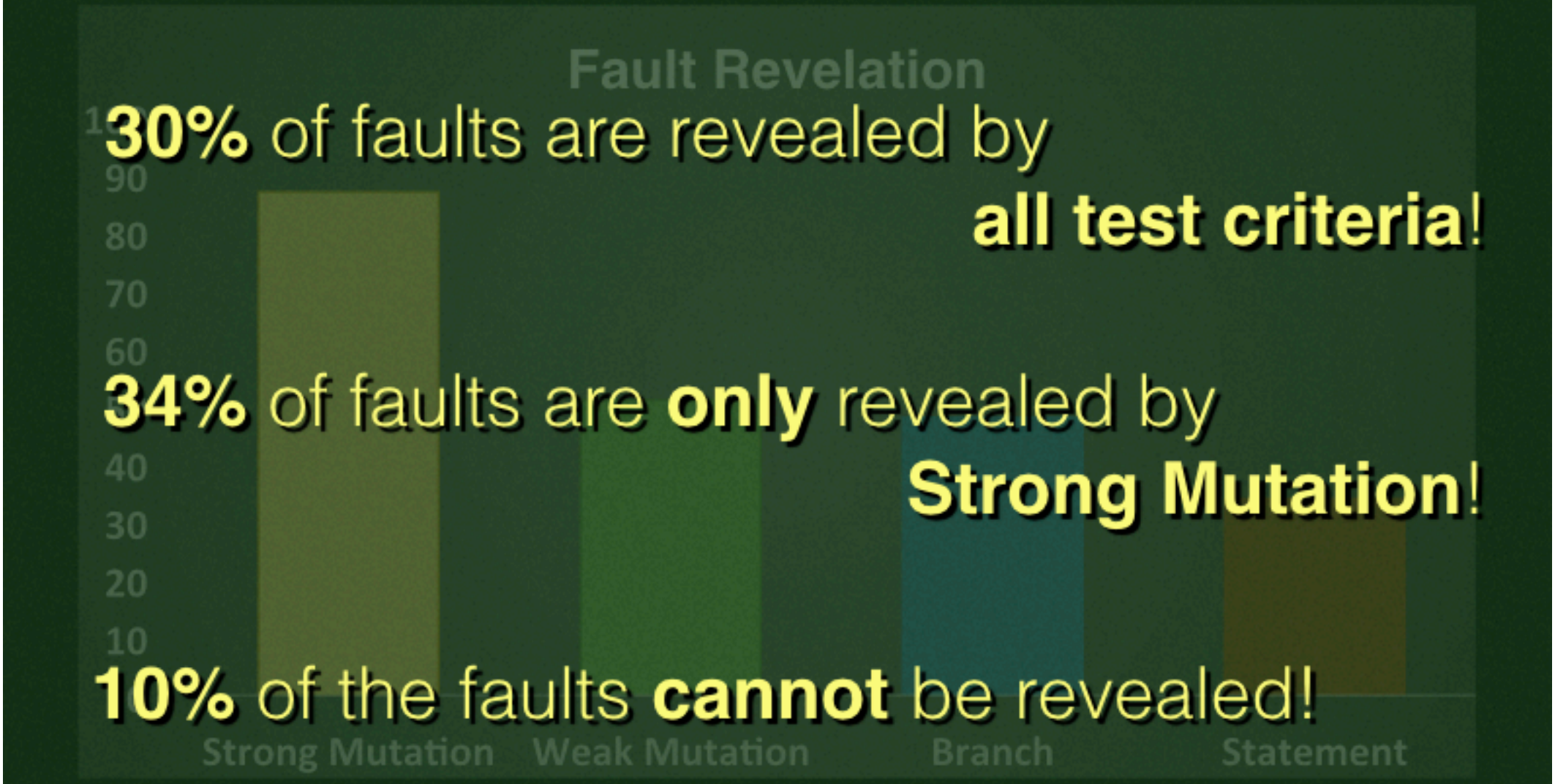


Test pool

Mutants, Branches, statements

Test suites


Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
Test case
...
55%
60%
65%
...
95%
100%



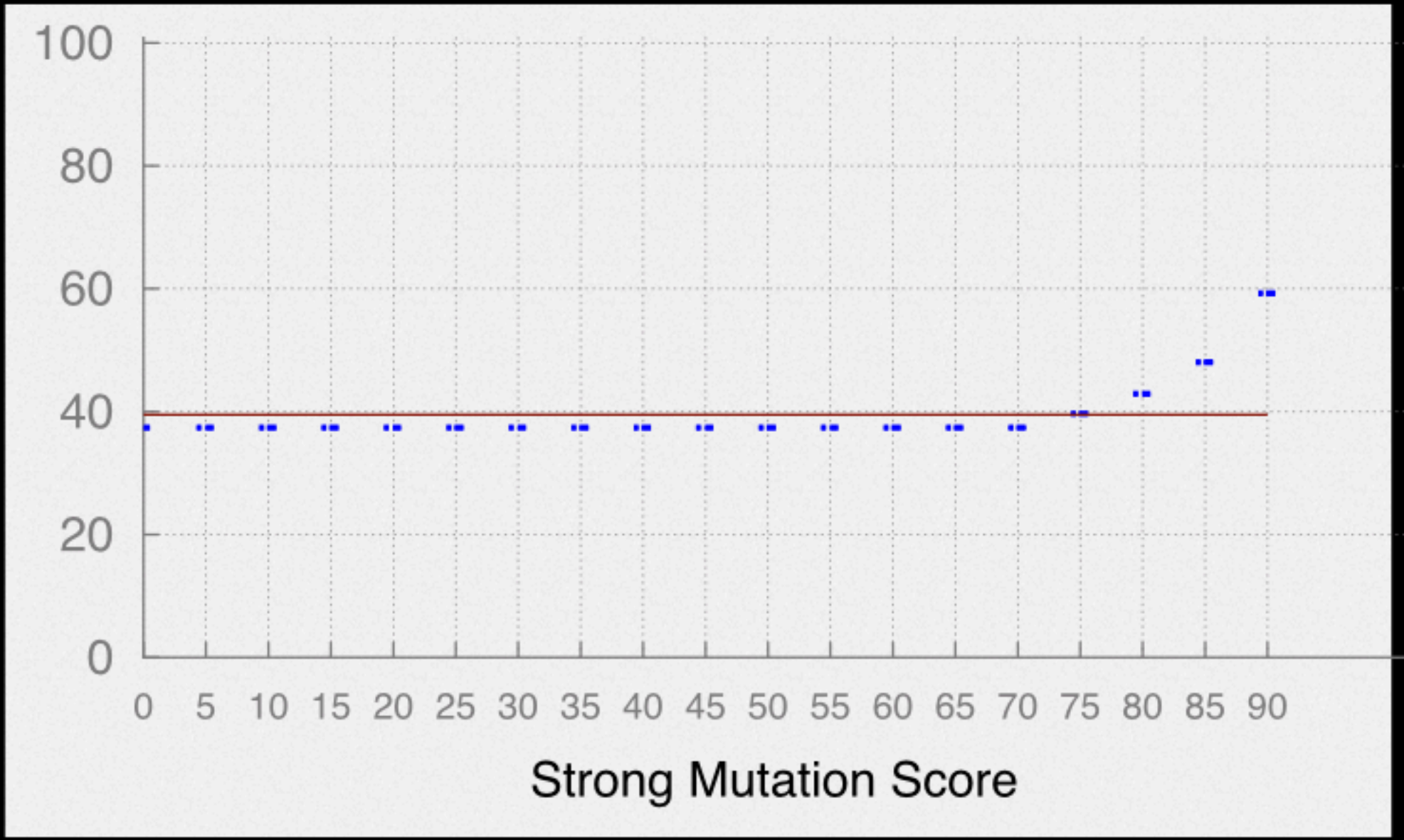
30% of faults are revealed by **all test criteria**!

34% of faults are **only** revealed by **Strong Mutation**!

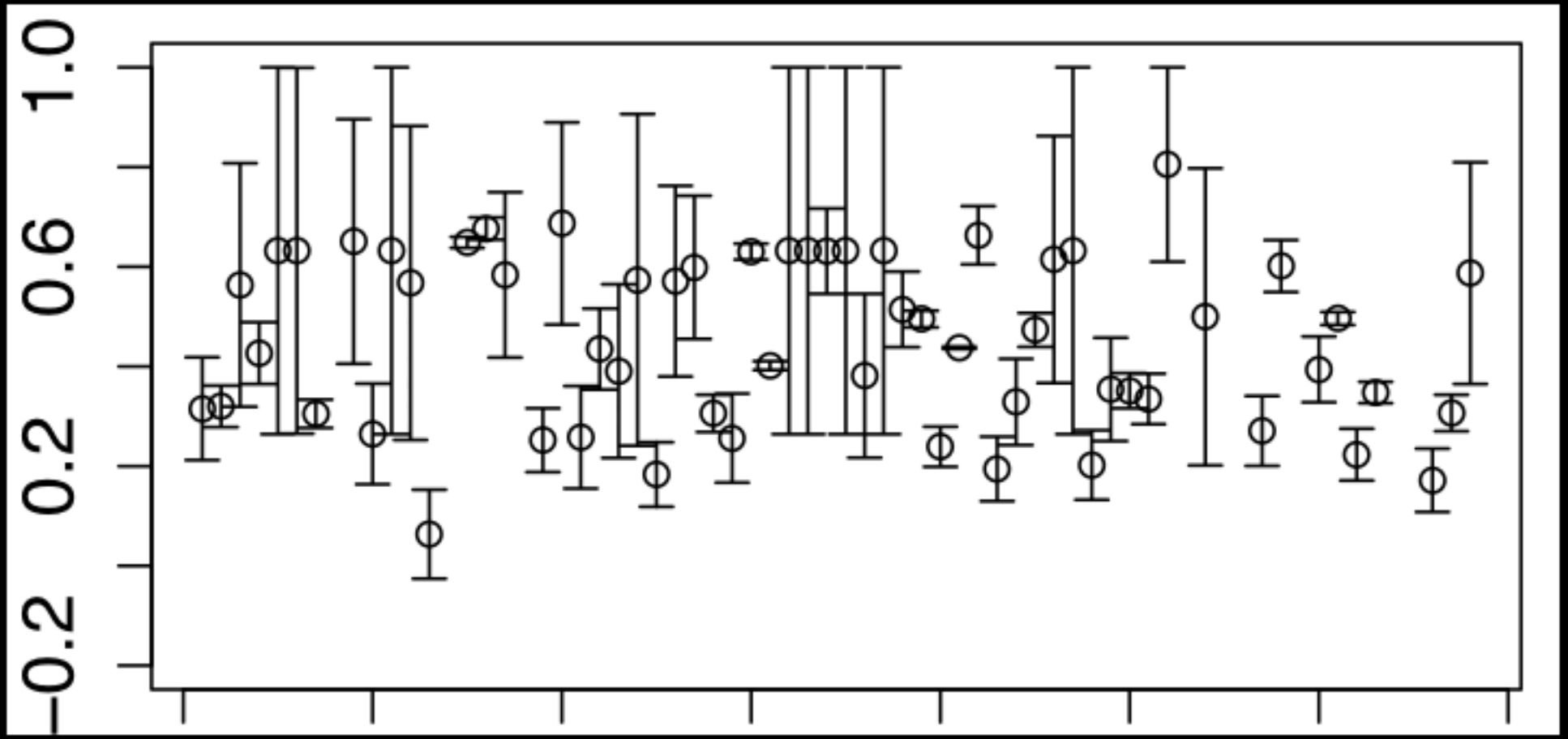
10% of the faults **cannot** be revealed!

Titcheu et al. **Empirical Study on Mutation, Statement and Branch Coverage Fault Revelation that Avoids the Unreliable Clean Program Assumption,** ICSE '17.

Fault Revelation at highest score levels



Fault Revelation Improvement at top 10% scores



CoREBench (http://www.comp.nus.edu.sg/~release/corebench/)

Are Mutation Scores Correlated with Real Fault Detection? A Large Scale Empirical study on the Relationship Between Mutants and Real Faults, ICSE'18

Defects4J
(https://github.com/rjust/defects4j)

Defects4J -- version 1.1.0 build canceled

Defects4J is a collection of reproducible bugs and a supporting infrastructure with the goal of advancing software engineering research.

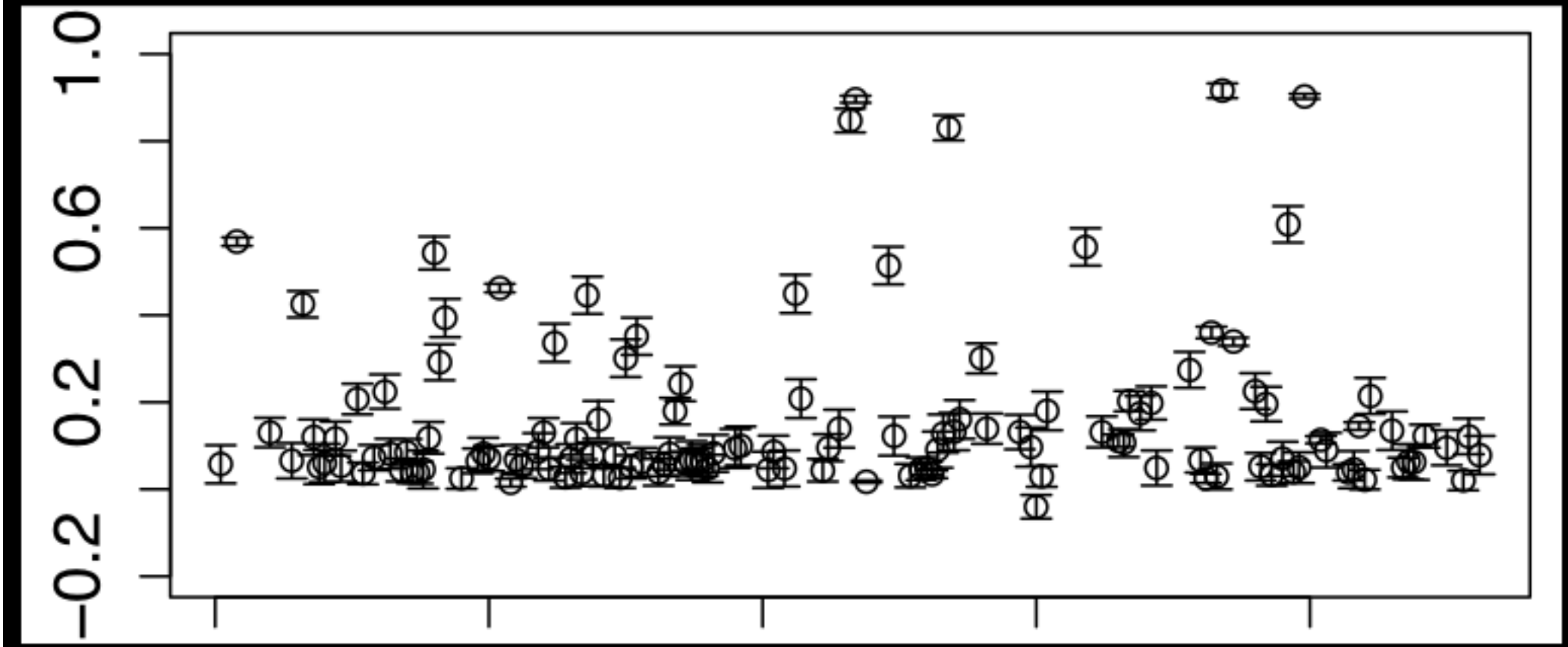
The projects

Defects4J contains 395 bugs from the following open-source projects:

Identifier	Project name	Number of bugs
Chart	JFreeChart	26
Closure	Closure compiler	133
Lang	Apache commons-lang	65
Math	Apache commons-math	106
Mockito	Mockito	38
Time	Joda-Time	27

The bugs

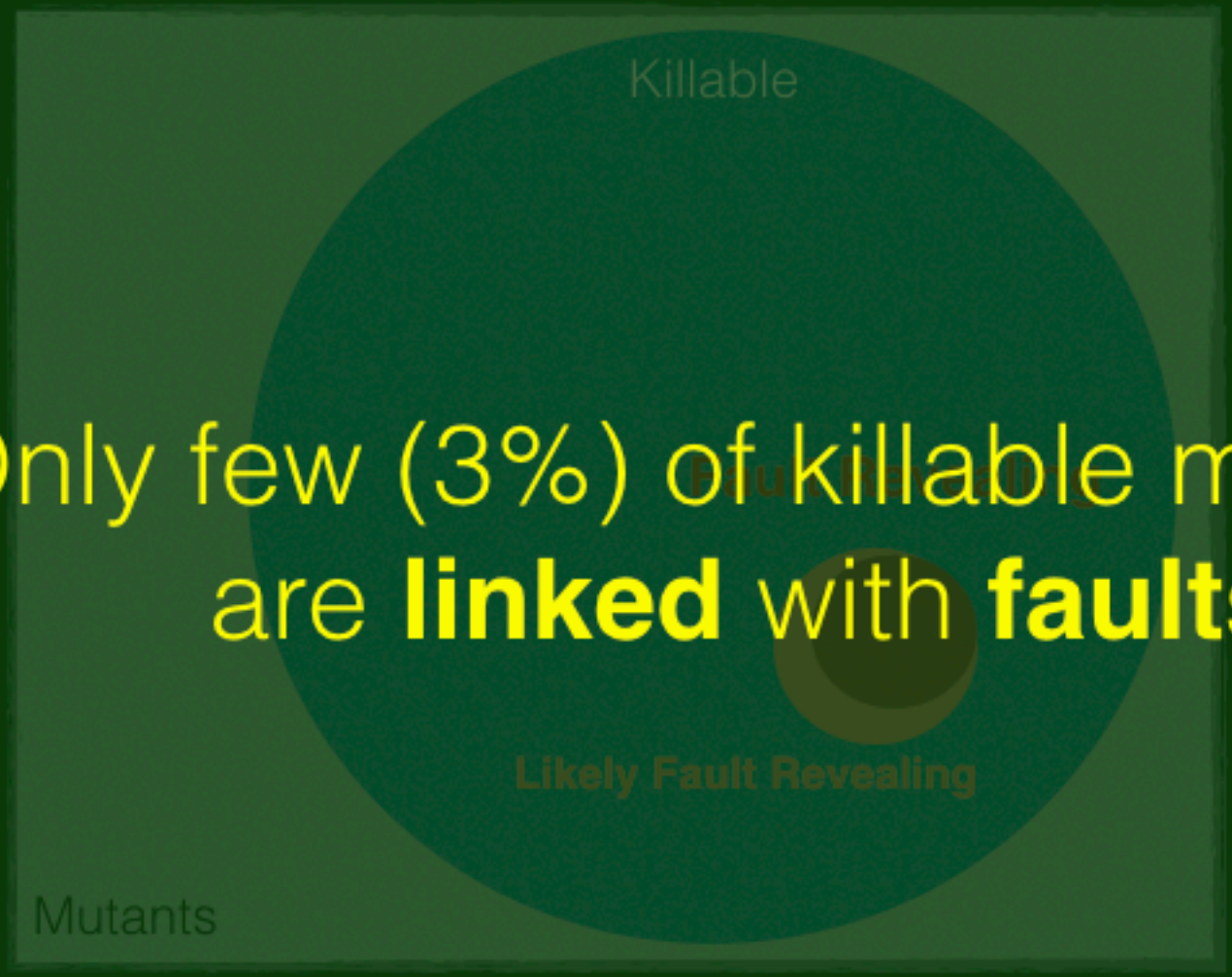
Fault Revelation Improvement



Defects4J (https://github.com/rjust/defects4j)

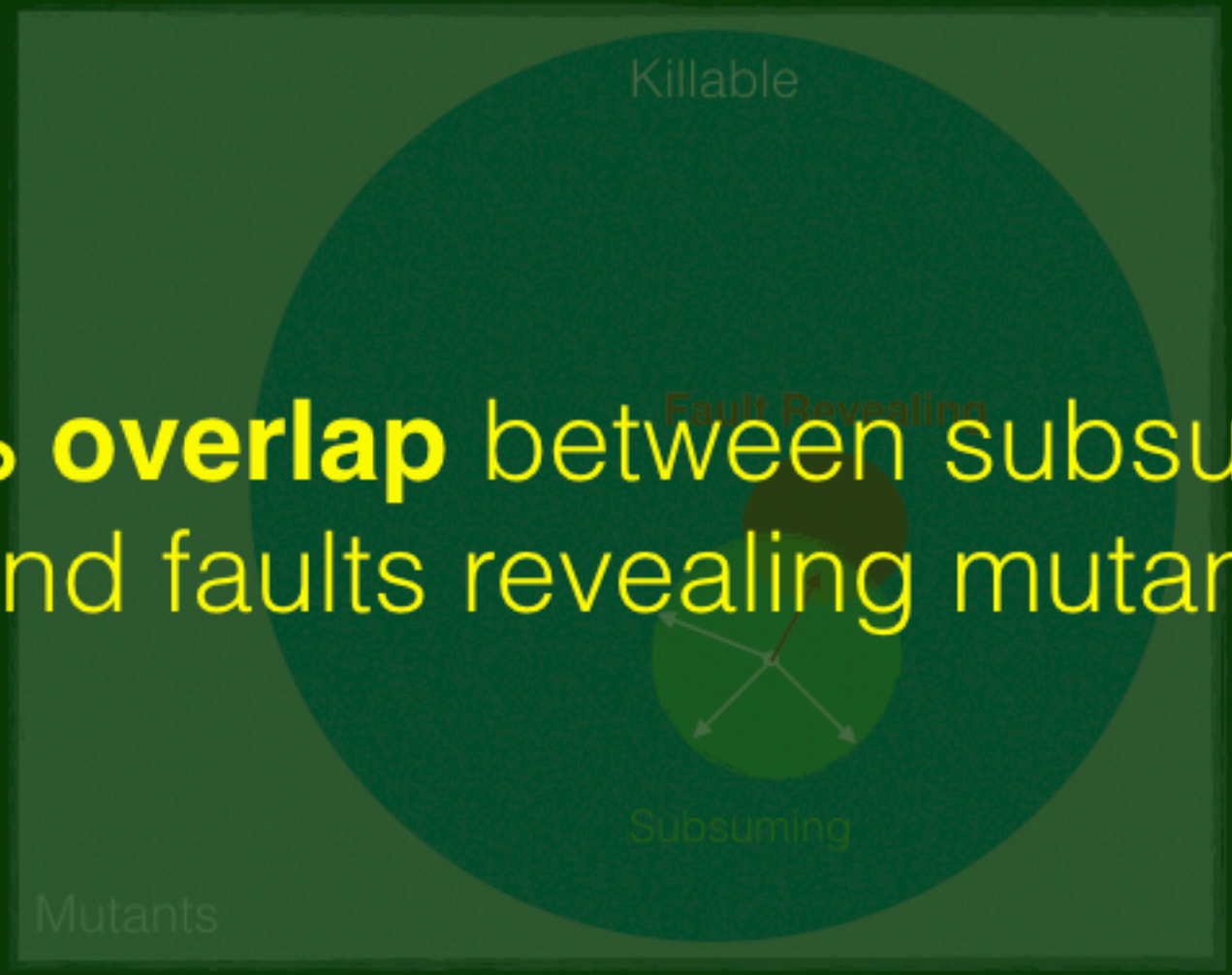
Are Mutation Scores Correlated with Real Fault Detection? A Large Scale Empirical study on the Relationship Between Mutants and Real Faults, ICSE'18

Link between Mutants and Faults



Only few (3%) of killable mutants are **linked** with **faults**

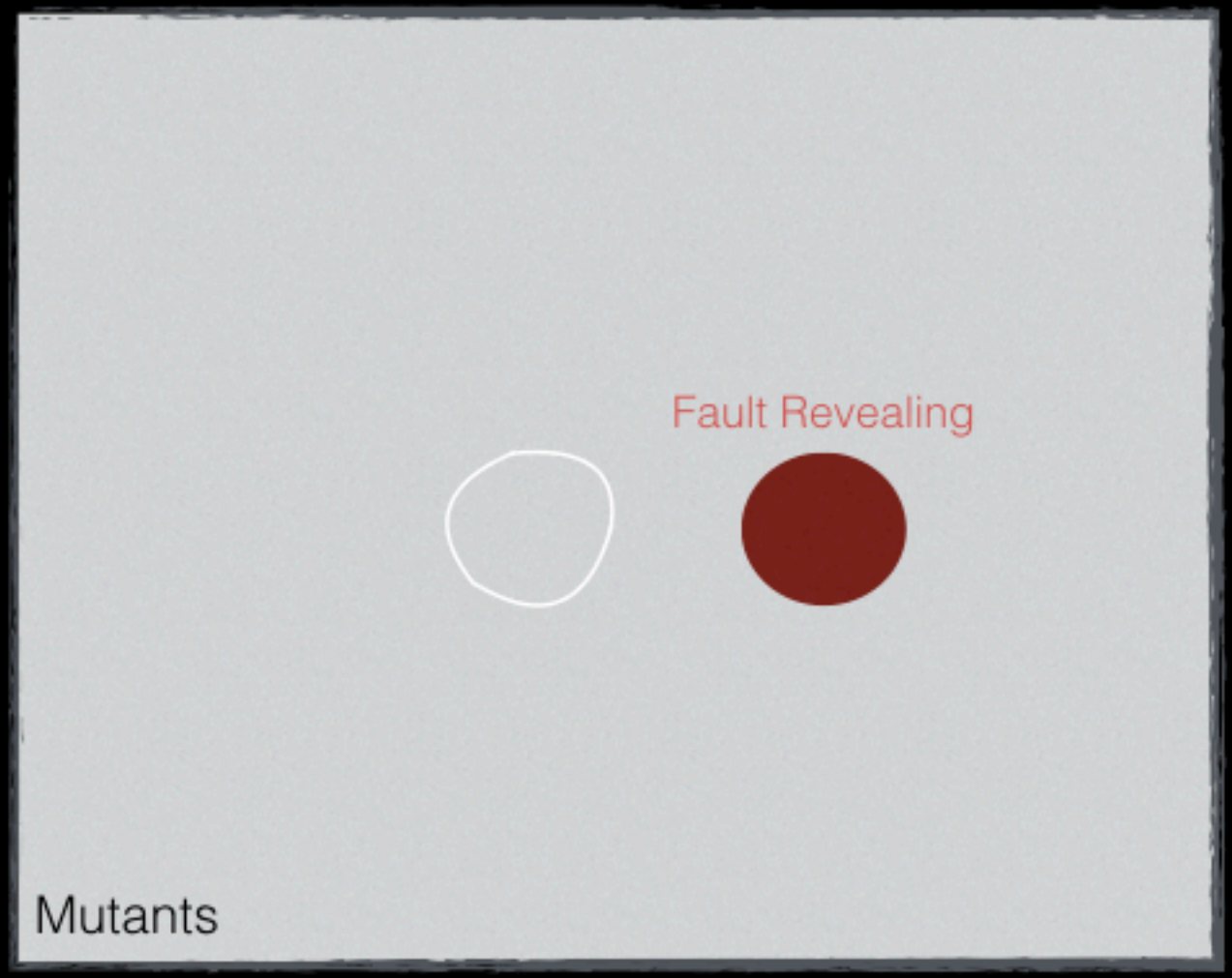
Subsuming mutants and Faults



10% overlap between subsuming and faults revealing mutants

Selection Problem

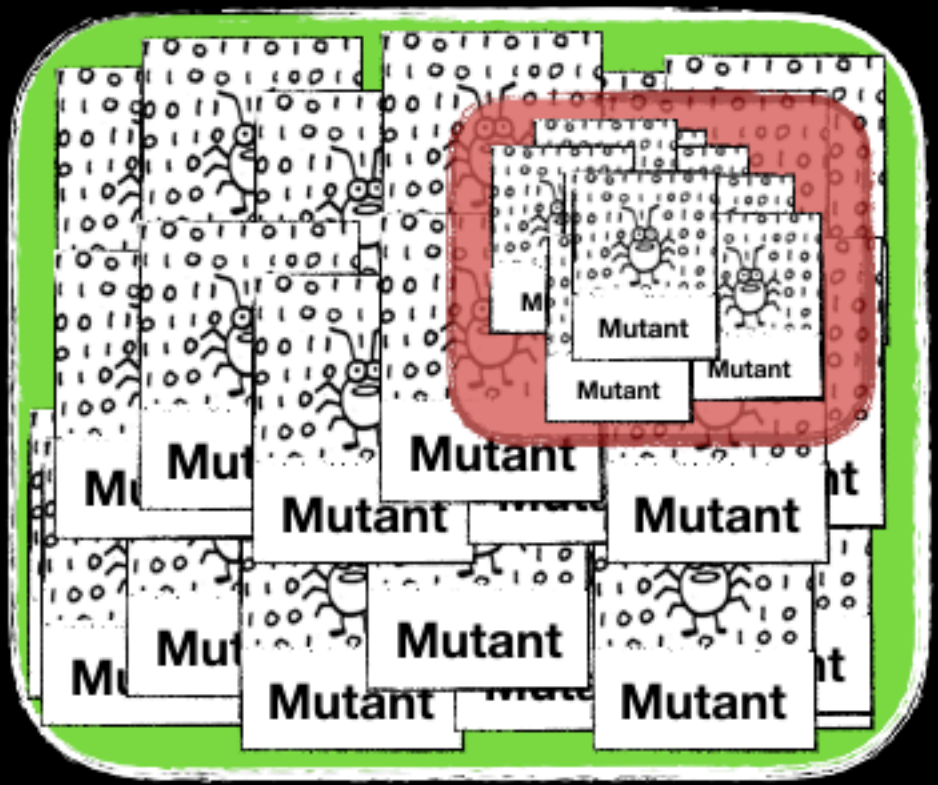
Selection Problem



Problem

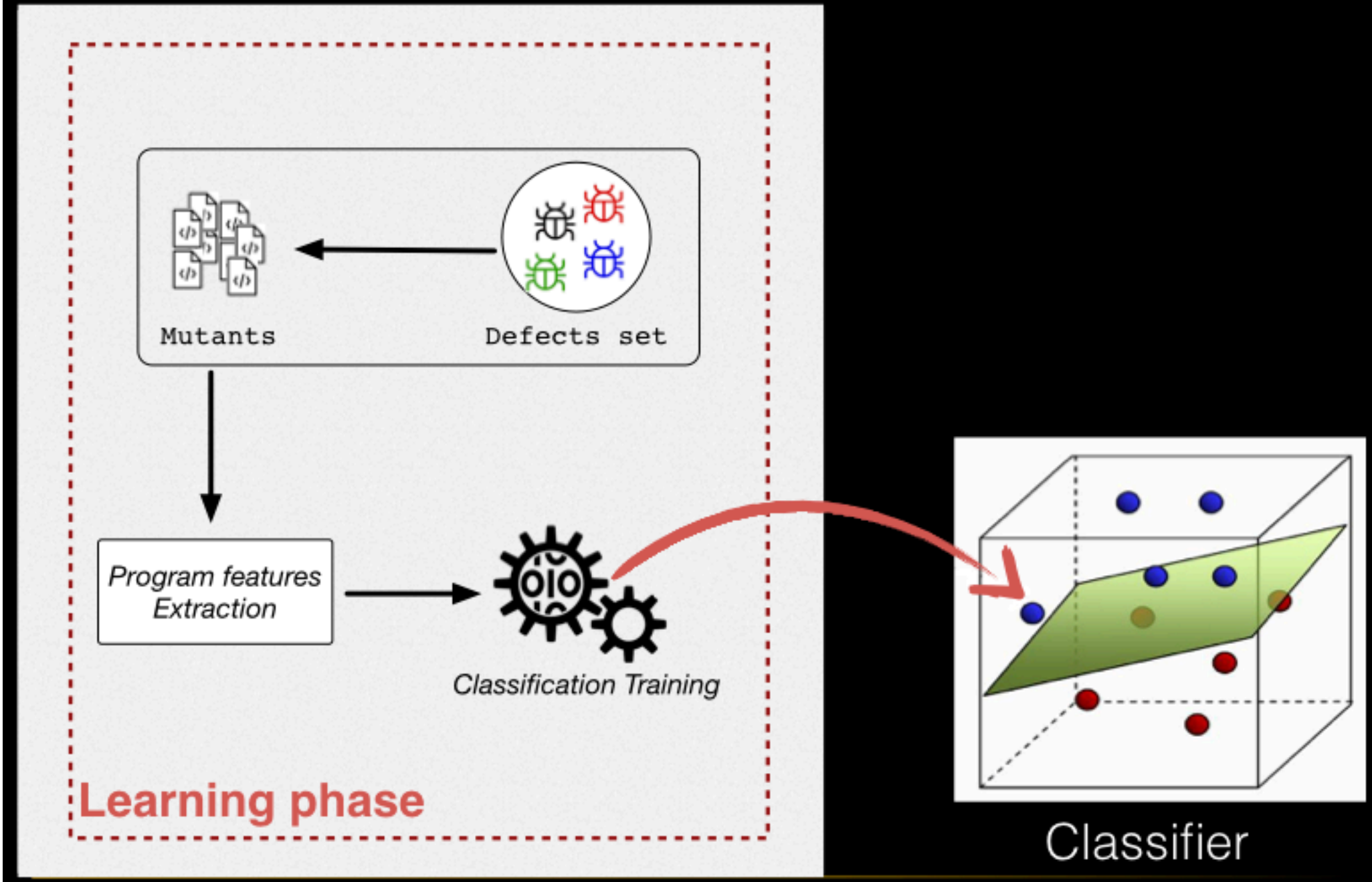
Set selection problem

Prioritisation Problem

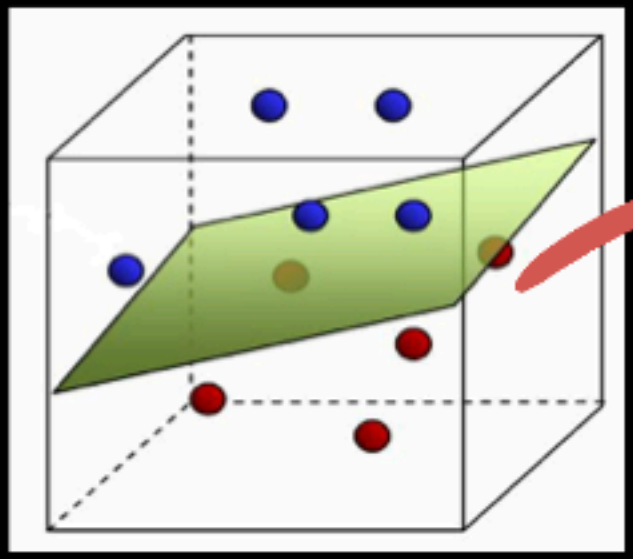


Prediction Modelling

Machine Learning



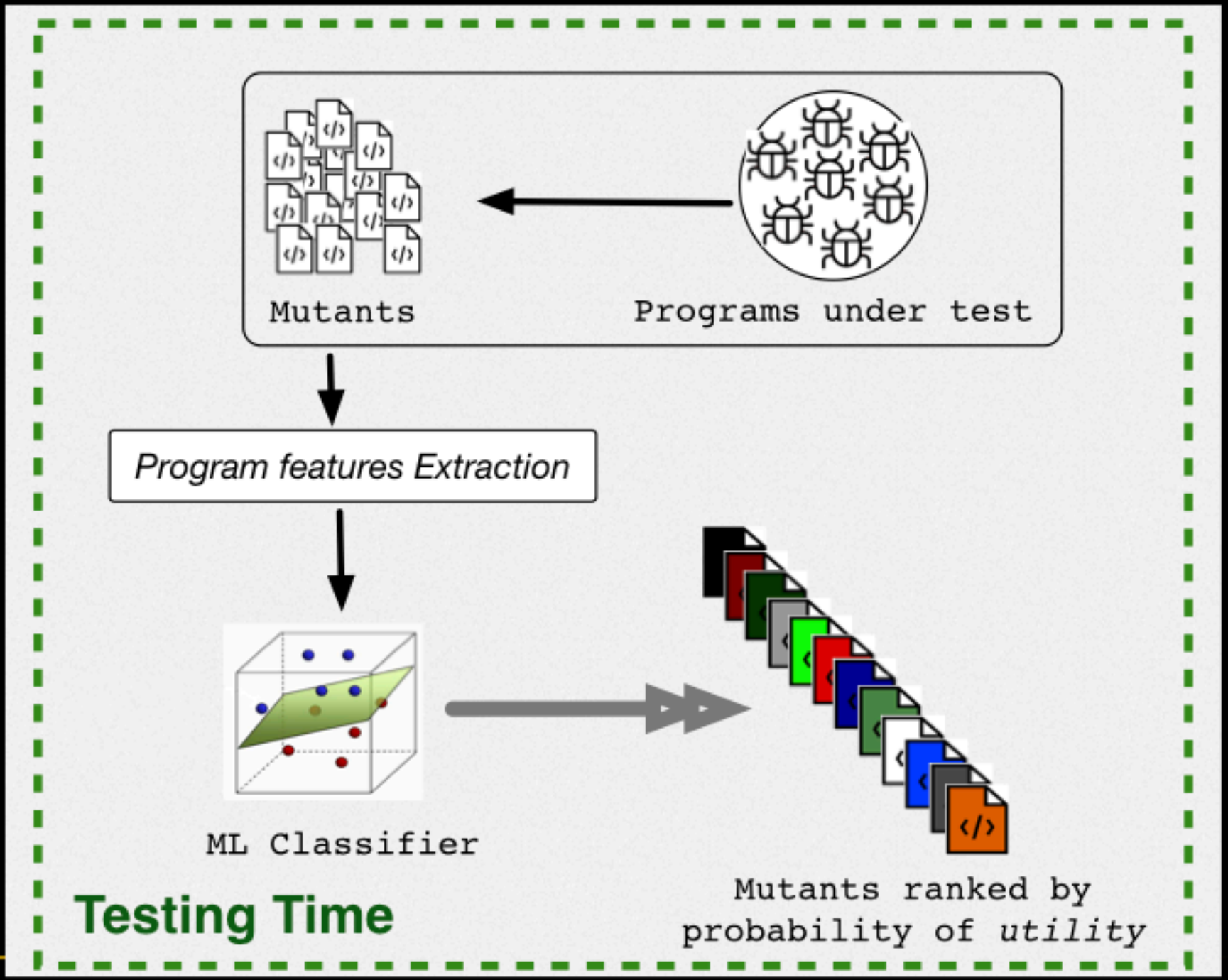
Classifier that predicts mutants' Utility



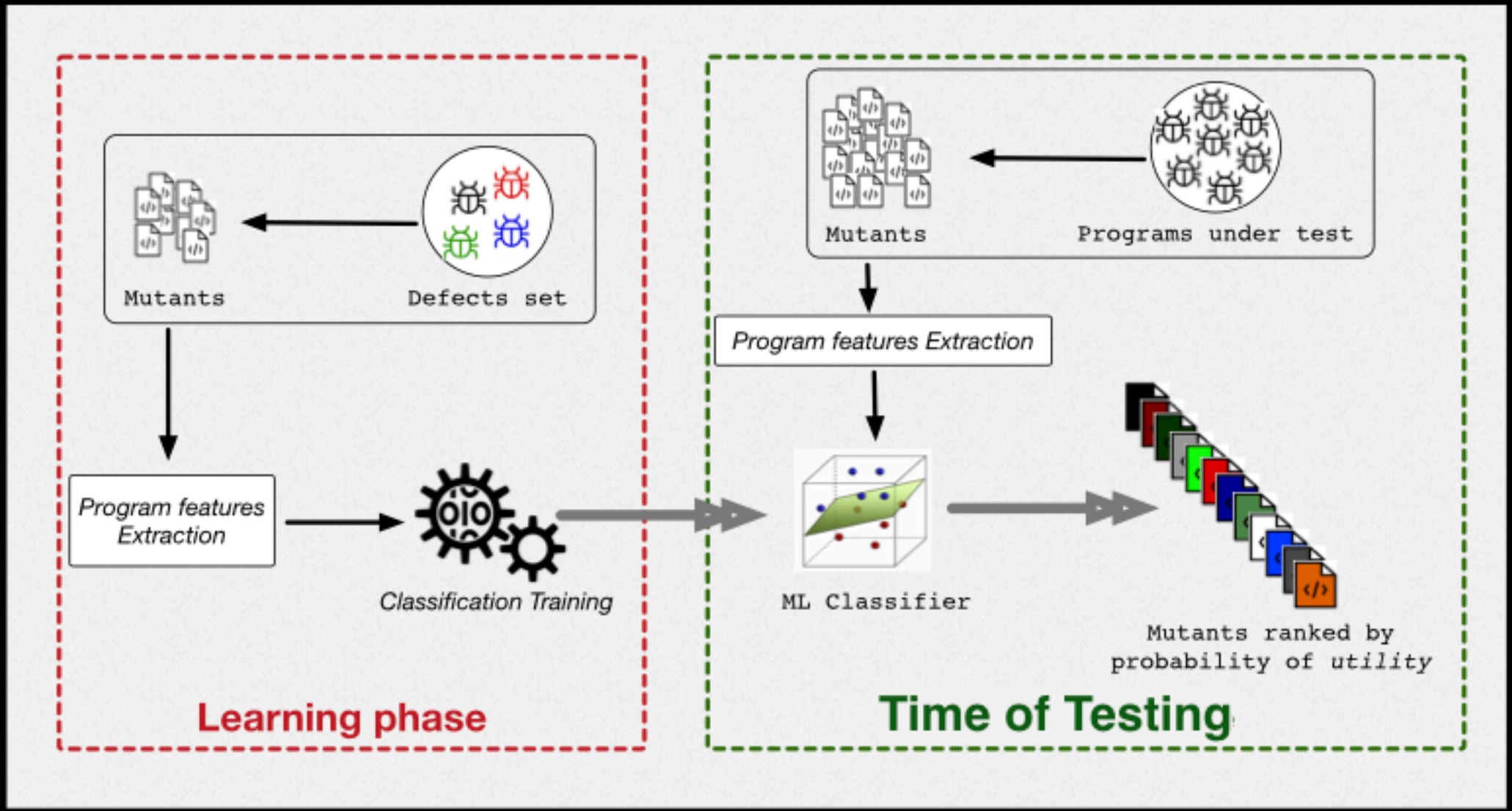
Classifier

Testing Process

Learning-to-rank Mutants



Learning-to-rank Mutants



Assumptions

1. We have (**enough**) historical data
2. Causes of defects are **repeated**
3. Mutants are **linked** with defects
4. Mutants' utility can be captured by **static** features

Features

- Depth in the CFG of ***B***
- Complexity of **S** as its number of mutants
- Mutant type of **M**
- Types of mutants on P_S
- AST type of P_S
- Number of predecessor/successor Basic Blocks of ***B*** on the CFG
- Number of AST parents of ***S***
- Type of ***B***
- Type of P_S' basic block
- Number of mutants on ***S***
- Number of mutants of statement control out-dependent to **S**
- Number of mutants of statement control in-dependent to **S**
- Number of mutants of statement data out-dependent to **S**
- Number of mutants of statement data in-dependent to **S**
- Number of mutants of statement control out-dependent to P_S
- Number of mutants of statement control in-dependent to P_S
- Number of mutants of statement data out-dependent to P_S
- Number of mutants of statement data in-dependent to P_S
- Mutant type and statement type of statement control out-dependent to **S**
- Mutant type and statement type of statement control in-dependent to **S**
- Mutant type and statement type of statement data out-dependent to **S**
- Mutant type and statement type of statement data in-dependent to **S**

M denotes a mutant on statement S in basic block B. P_S denotes the AST parent (of S)

Mutant Type

```
int maximum (int a, int b, int
1. int max = a;
2. if (b > max)
3.   max = b;
4. if (c > max)    // M1: if (c < max)
5.   max = c;
6. return max;
}
```

Mutant Type:

$> \rightarrow <$

AST R

AST Parent Mutant Type:

$M1$ mutates relation

$M2$ mutate AST parent of relation: $if(e) \rightarrow if(!e)$

The mutant type of $M2$ is the feature's value for $M1$

```
int maximum (int a, int b, int
1. int max = a;
2. if (b > max)
3.   max = b;
4. if (c > max)     // M1: if (c > max)
                    // M2: if (!(c > max))
5.   max = c;
6. return max;
}
```

CFG related

```
int maximum (int a, int b, i
1. int max = a;
2. if (b > max)
3.   max = b;
4. if (c > max)     // M1: if (c < max)
5.   max = c;      // M2: max = c + 1
6. return max;
}
```

Depth in CFG:

$M1$ has depth 2

$M2$ has depth 3

Control & Data Dependences

```
int maximum (int a, int b, int
1. int max = a;
2. if (b > max)
3.   max = b;
4. if (c > max)     // M1: if (c < max)
5.   max = c;       // M2: max = c + 1
6. return max;     // M3 return 0;
}
```

Dependence:

Control : $M1 \rightarrow M2$

Data : $M2 \rightarrow M3$

Implementation

Use **Gradient Boosted Decision** Trees

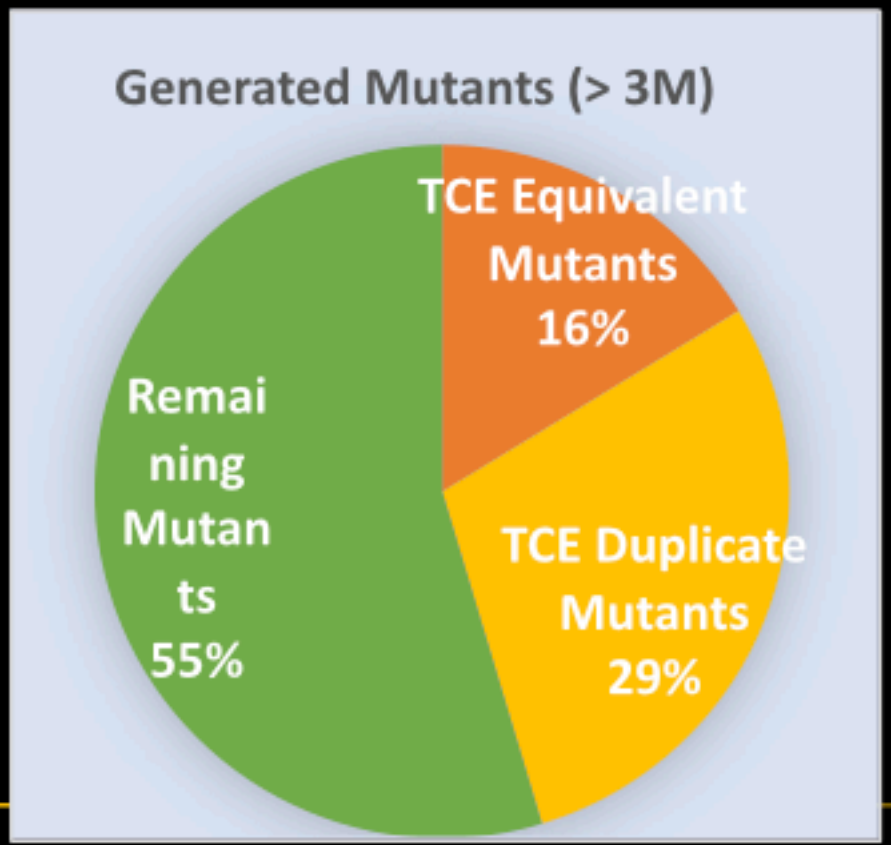
LLVM-based mutation tool

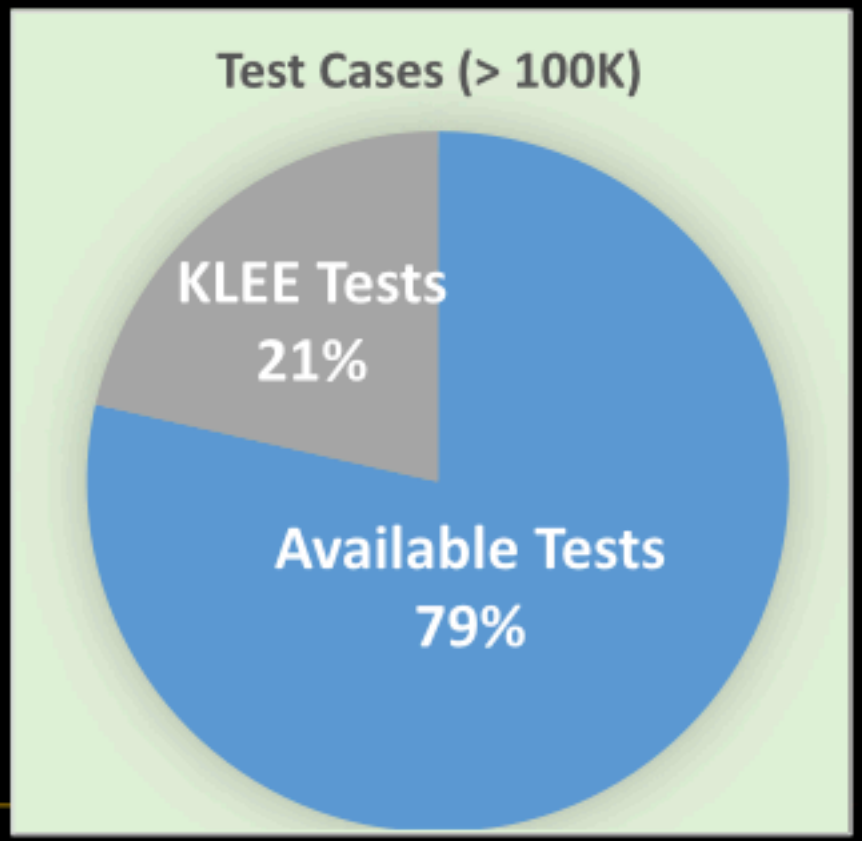
Mutant Types

Mutated Instruction	Original Instruction Type	Mutated Instruction Type
STATEMENT	WHOLE STMT	TRAPSTMT
	WHOLE STMT	DELSTMT
	CALL STATEMENT	SHUFFLEARGS
	SWITCH STATEMENT	SHUFFLECASESDESTS
	SWITCH STATEMENT	REMOVECASES
EXPRESSION	SCALAR.ATOM	UNARY
	SCALAR.BINARY	BINARY
	SCALAR.BINARY	UNARY
	SCALAR.ATOM	BINARY
	SCALAR.BINARY	TRAPSTMT
	POINTER.BINARY	BINARY
	SCALAR.BINARY	DELSTMT
	DEREFERENCE.BINARY	BINARY
	SCALAR.UNARY	UNARY
	POINTER.BINARY	UNARY
	DEREFERENCE.BINARY	UNARY
	POINTER.ATOM	UNARY
	POINTER.UNARY	UNARY

Codeflaws *(https://codeflaws.github.io/)*

3902 defects from
the **Codeforces** online database
1 to 322 LOCs with average 36 LOCs

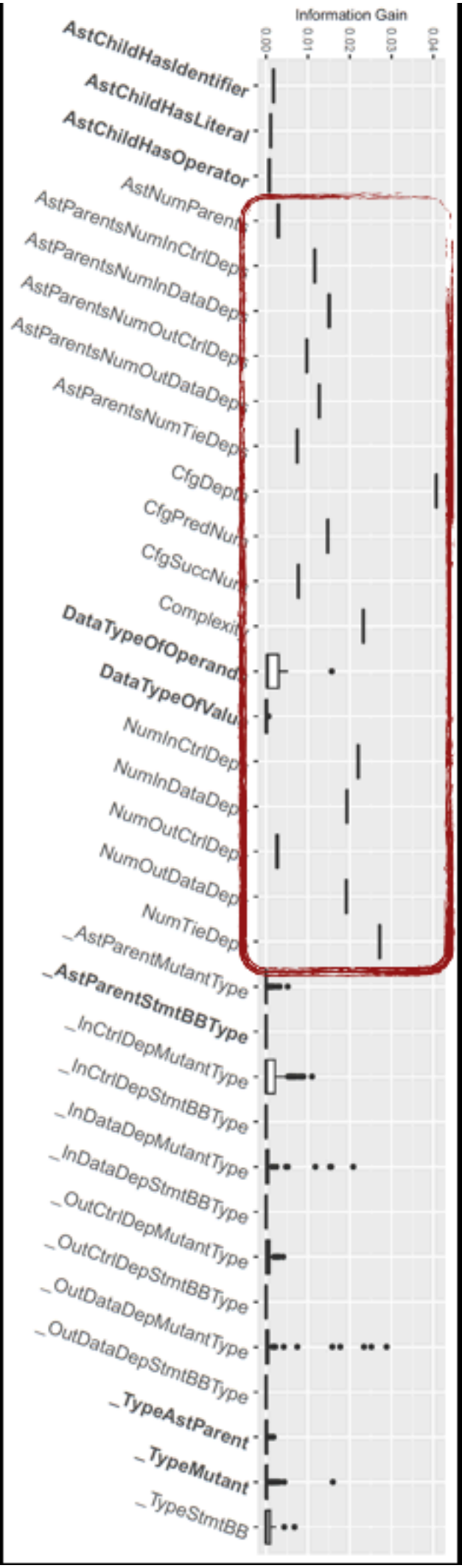


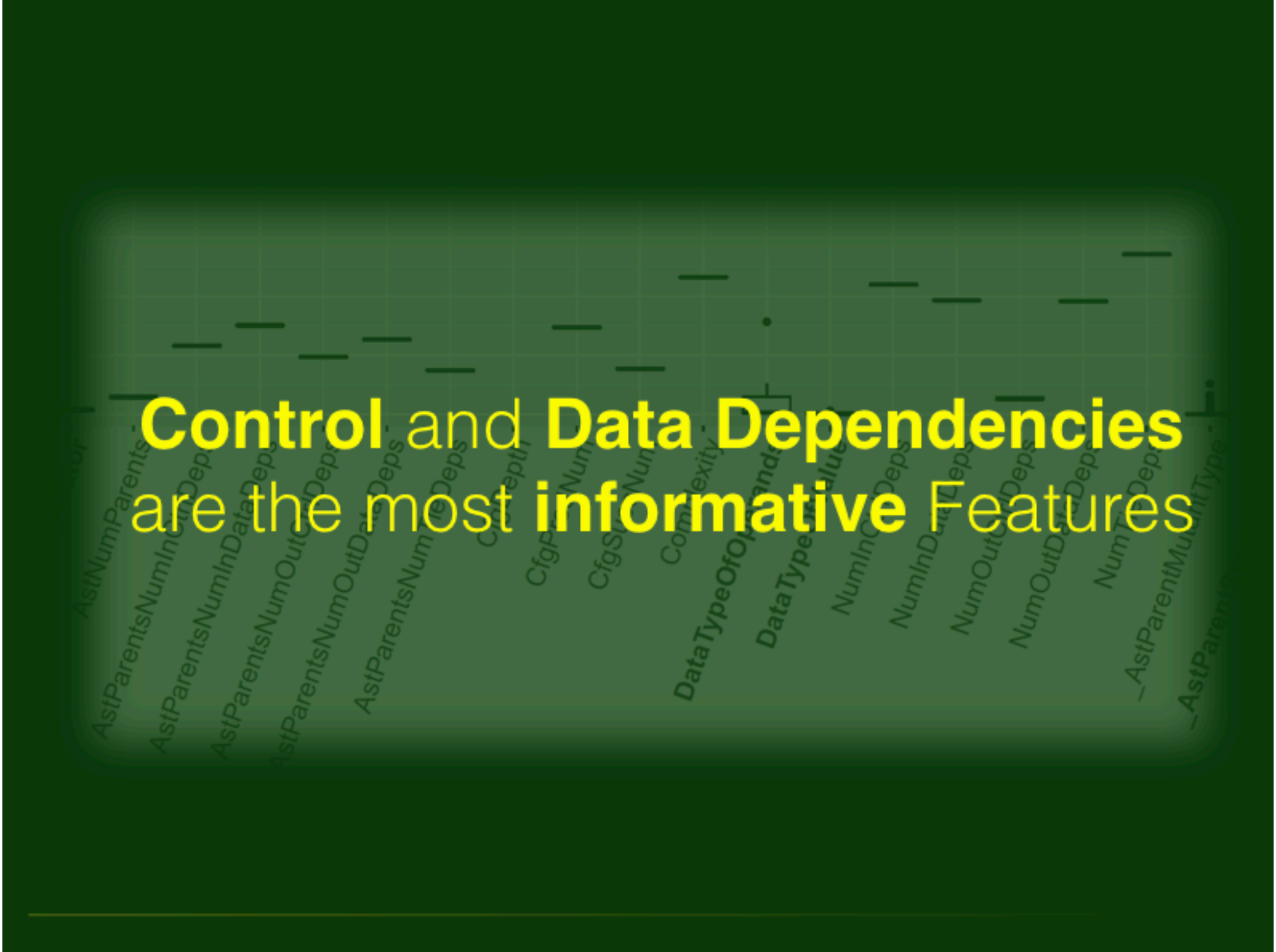


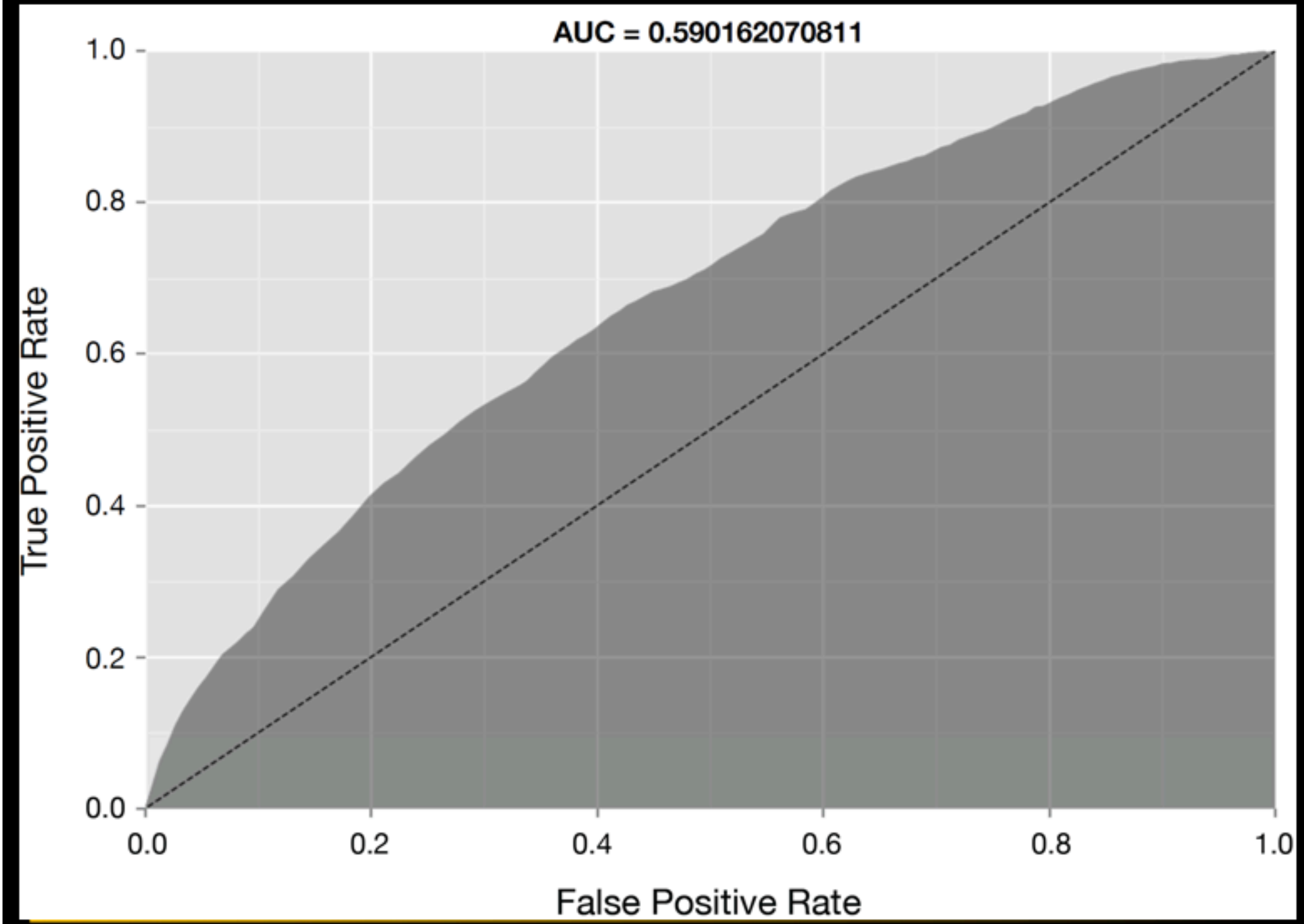
Fault Set

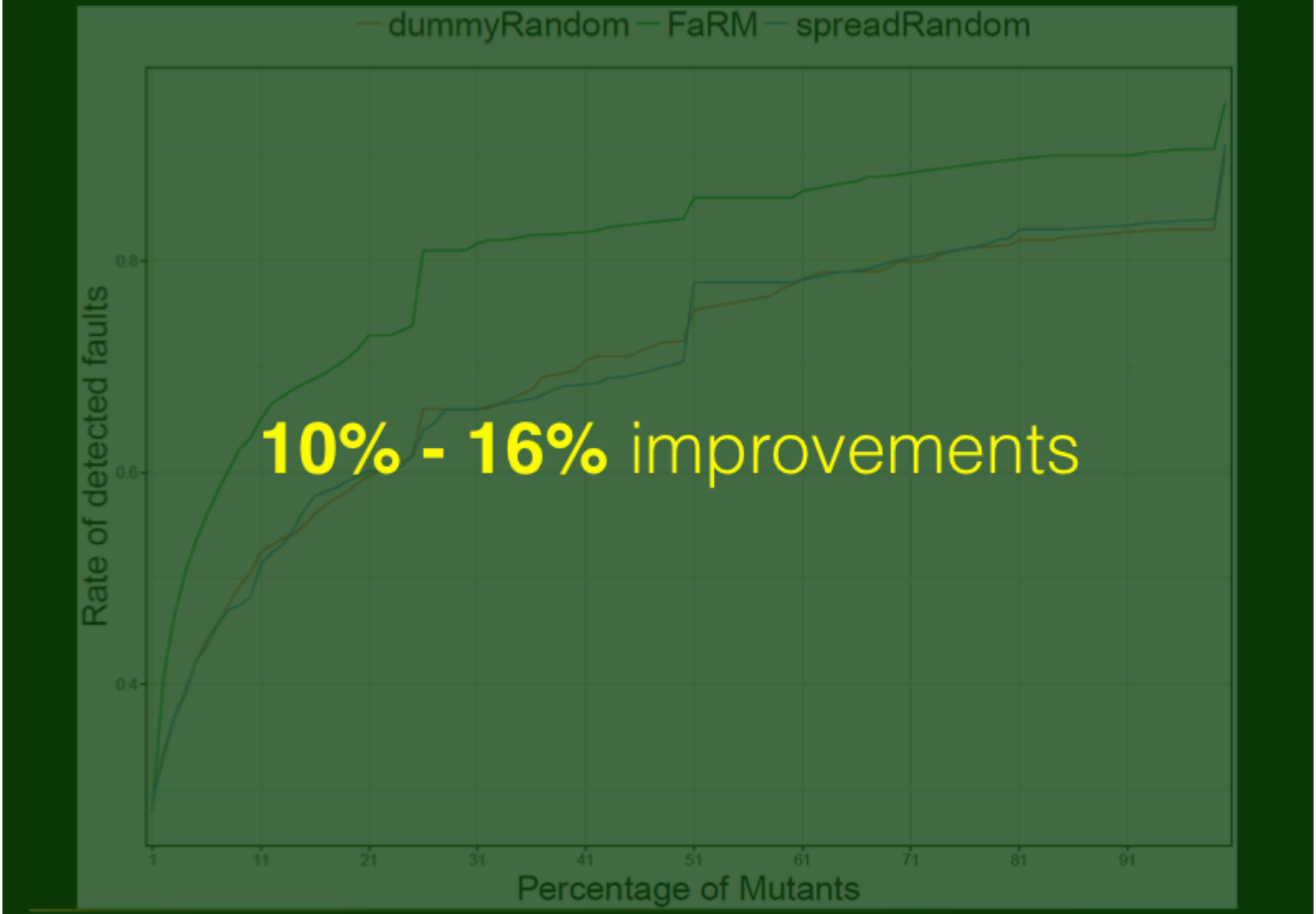
TABLE I: Fault Classes

AST Type	Fault Class	Example
Higher Order	Expression HEXP	$\ominus\ if(C)$ $\oplus\ if(C \mid\mid \mathbf{D})$
	Non-branch Stmt HDMS	$\ominus\ printf(s);$ $\ominus\ x = y + 3;$
	Combination HCOM	$\ominus\ rep(i, n)$ $\oplus\ for(...)$
	Non-branch Stmt HIMS	$\oplus\ printf(s);$ $\oplus\ x = y + 3;$
	Branch Stmt HBRN	$\oplus\ if(C)\ \{printf(s);\}$
	Others HOTH	$\ominus\ g(0);$ $\oplus\ for(...)\{f(i);\}$
Statement	Function Call SISF	$\oplus\ scanf(\text{“}\%d'', \&n);$
	Type STYP	$\ominus$ **int** a $\oplus$ **long** a
	Control Flow SRIF	$\ominus\ if(a > b)$ $\oplus\ if(\mathbf{g}(a) > b)$
	Data Flow SISA	$\oplus\ t = 0$
	Move SMOV	$\ominus\ f(x);$ g(x); $\oplus\ f(x);$
Operand	Variable DRWV	$\ominus\ \mathbf{b} = 0;$ $\oplus\ \mathbf{a} = 0;$
	Array DCCA	$\ominus\ int\ x[\mathbf{2}];$ $\oplus\ int\ x[\mathbf{20}];$
	Variable DRVA	$\ominus\ if(\mathbf{i} > 0)$ $\oplus\ if(\mathbf{k} > 0)$
	Constant DCCR	$\ominus\ if(x > \mathbf{4})$ $\oplus\ if(x > \mathbf{3})$
Operator	Control Flow ORRN	$\ominus\ if(a > 0)$ $\oplus\ if(a >= 0)$
	Arithmetic OAAN	$\ominus\ v2$ **-=** 2 $\oplus\ v2$ **+=** 2
	Function Call OFPF	$\ominus\ f(\text{“}\%\mathbf{d}'', i);$ $\oplus\ f(\text{“}\%\mathbf{ld}'', i);$
	Control Flow OILN	$\ominus\ if(x)$ $\oplus\ if(x\mathbf{\&\&f(x)})$
	Arithmetic OAIS	$\ominus\ x$ **+=** y $\oplus\ x$ **+=** y **/ 2**

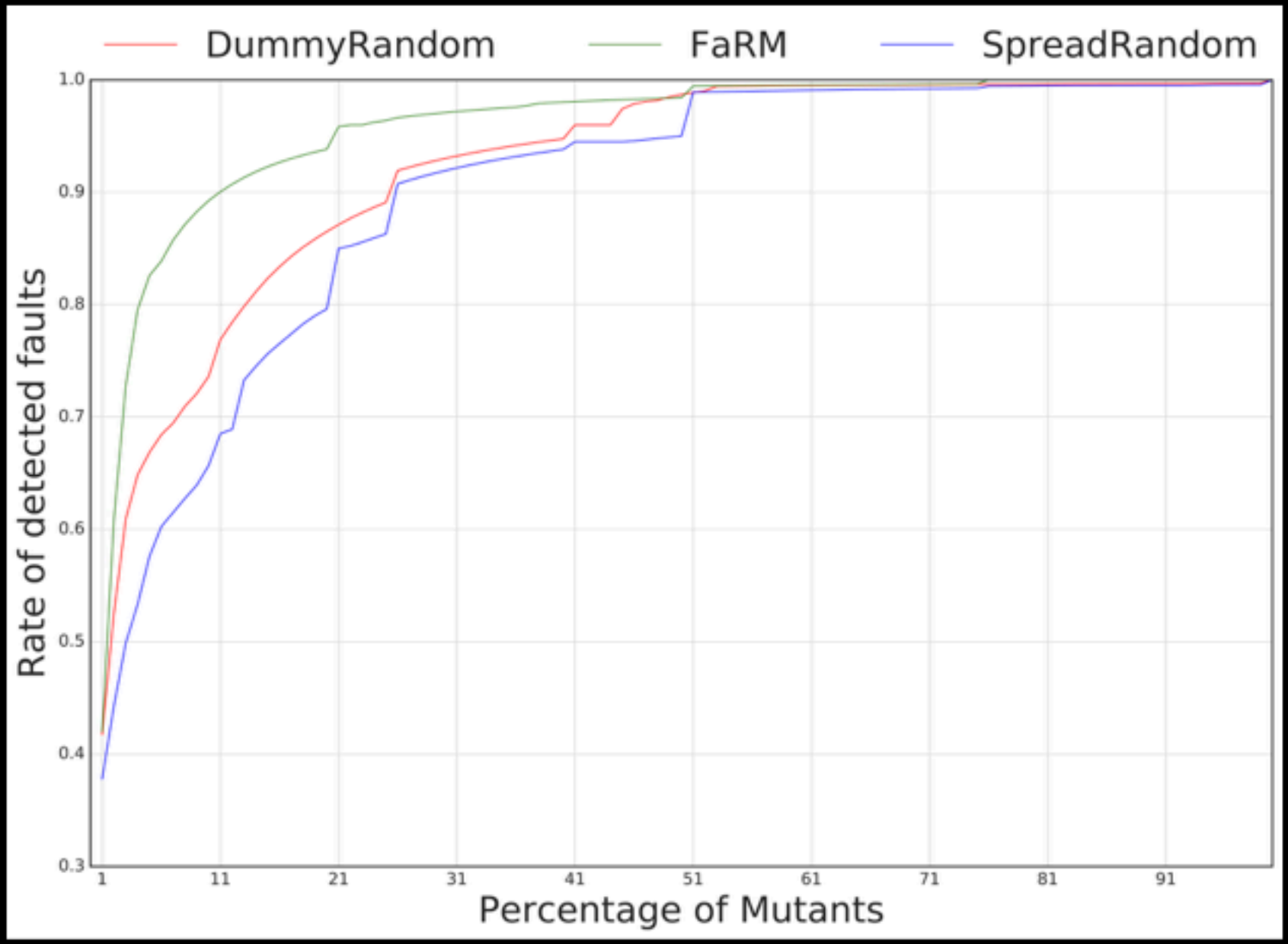

Information Gain
0.00
0.01
0.02
0.03
0.04
AstChildHasIdentifier
AstChildHasLiteral
AstChildHasOperator
AstNumParents
AstParentsNumInCtrlDeps
AstParentsNumInDataDeps
AstParentsNumOutCtrlDeps
AstParentsNumOutDataDeps
AstParentsNumTieDeps
CfgDepth
CfgPredNum
CfgSuccNum
Complexity
DataTypeOfOperand
DataTypeOfValu
NumInCtrlDep
NumInDataDep
NumOutCtrlDep
NumOutDataDep
NumTieDep
_AstParentMutantType
_AstParentStmtBBType
_InCtrlDepMutantType
_InCtrlDepStmtBBType
_InDataDepMutantType
_InDataDepStmtBBType
_OutCtrlDepMutantType
_OutCtrlDepStmtBBType
_OutDataDepMutantType
_OutDataDepStmtBBType
_TypeAstParent
_TypeMutant
_TypeStmtBB


Control and **Data Dependencies**
are the most **informative** Features

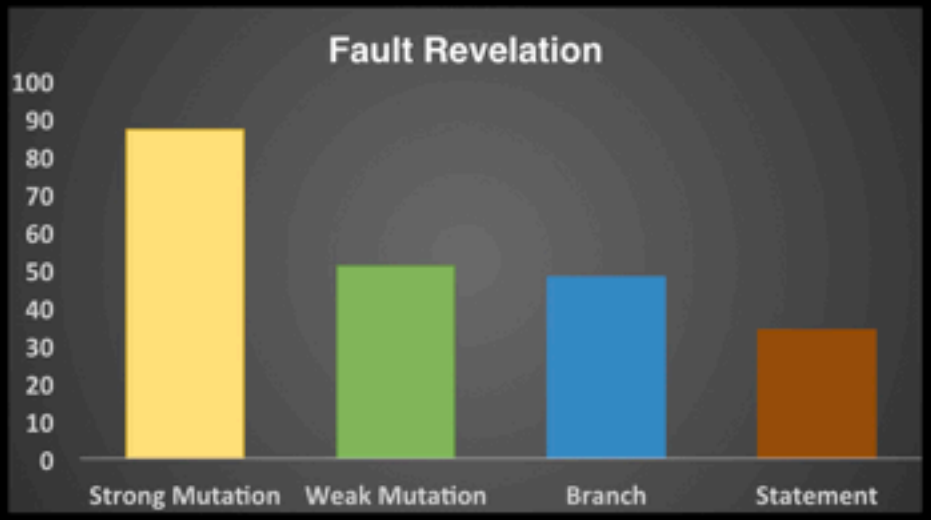

AUC = 0.590162070811
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


dummyRandom
FaRM
spreadRandom
Rate of detected faults
0.8
0.6
0.4
1
11
21
31
41
51
61
71
81
91
Percentage of Mutants
10% - 16% improvements

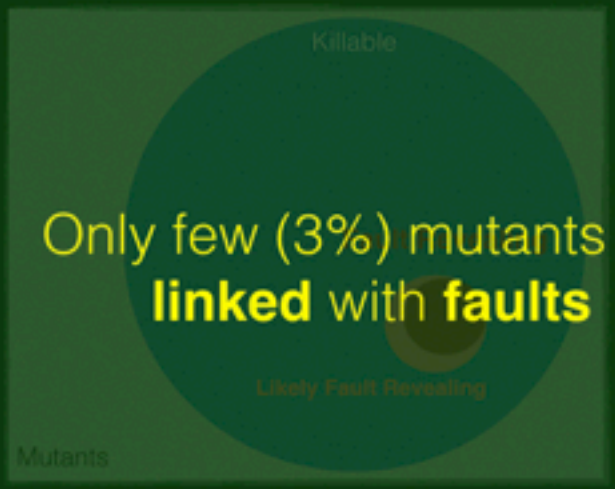
Results on CoREBench



Bohme et al. **"CoREBench: Studying Complexity of Regression Errors"**, ISSTA'14
http://www.comp.nus.edu.sg/~release/corebench/



Link between Mutants and Faults

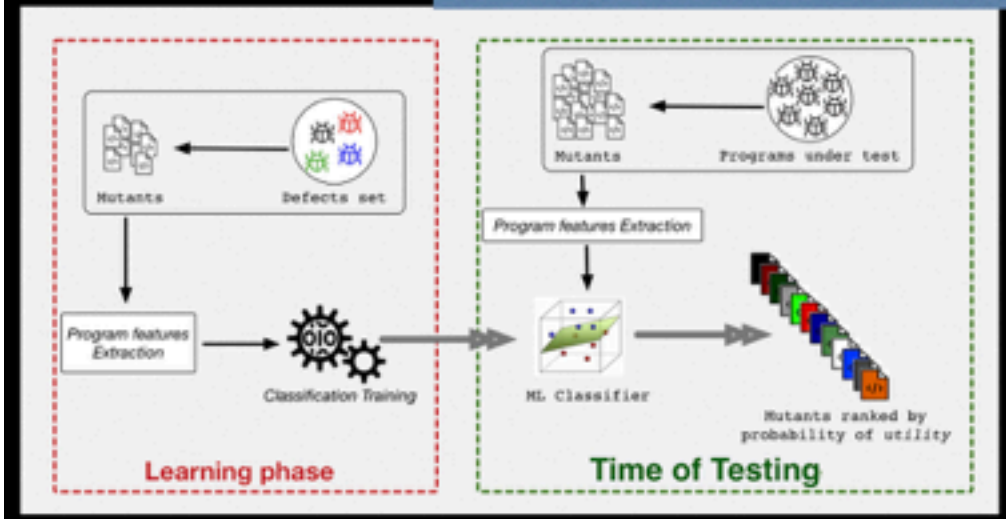


Only few (3%) mutants are **linked** with **faults**

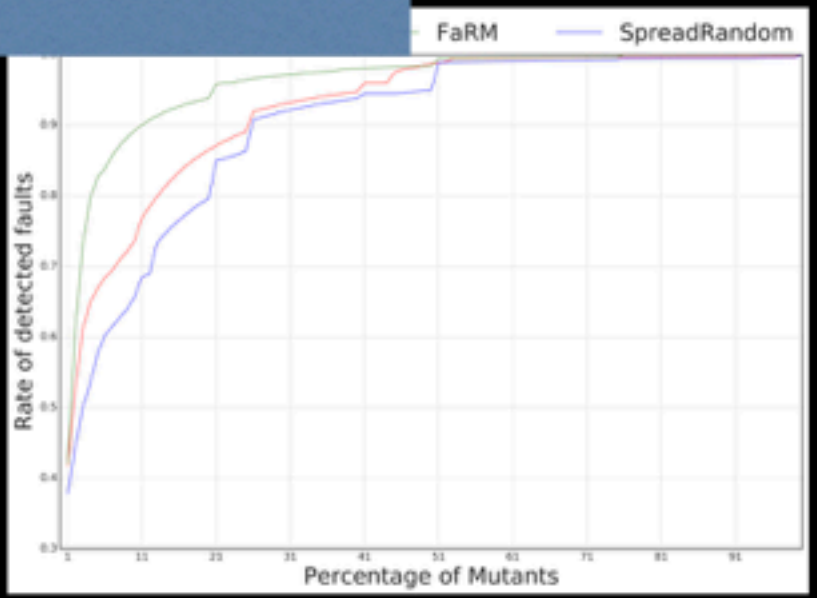
Titcheu et al. **Empirical Study o**
Revelation that Avoids the Un

Thank you for your attention!!!

Learning-to-rank



nch



Bohme et al. **"CoREBench: Studying Complexity of Regression Errors"**, ISSTA'14
http://www.comp.nus.edu.sg/~release/corebench/

Hard-to-propagate & Faults