

Improving Spectrum Based Fault Localisation Techniques

Gulsher Laghari, Alessandro Murgia and Serge Demeyer

BENEVOL`2015 - December 4, 2015



Ansymo

Antwerp Systems & Software Modelling
University of Antwerp



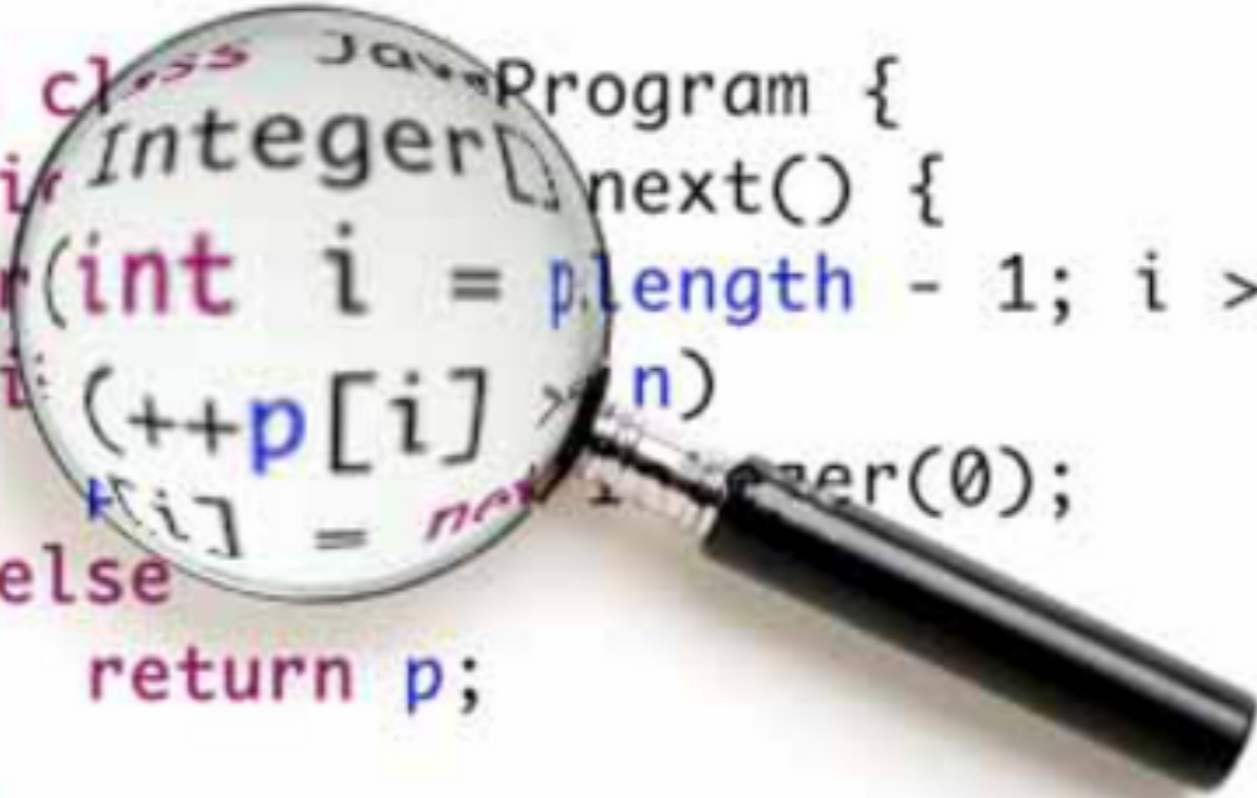
Universiteit
Antwerpen

Overview



Fault localisation

Fault Localisation an important step in the Debugging

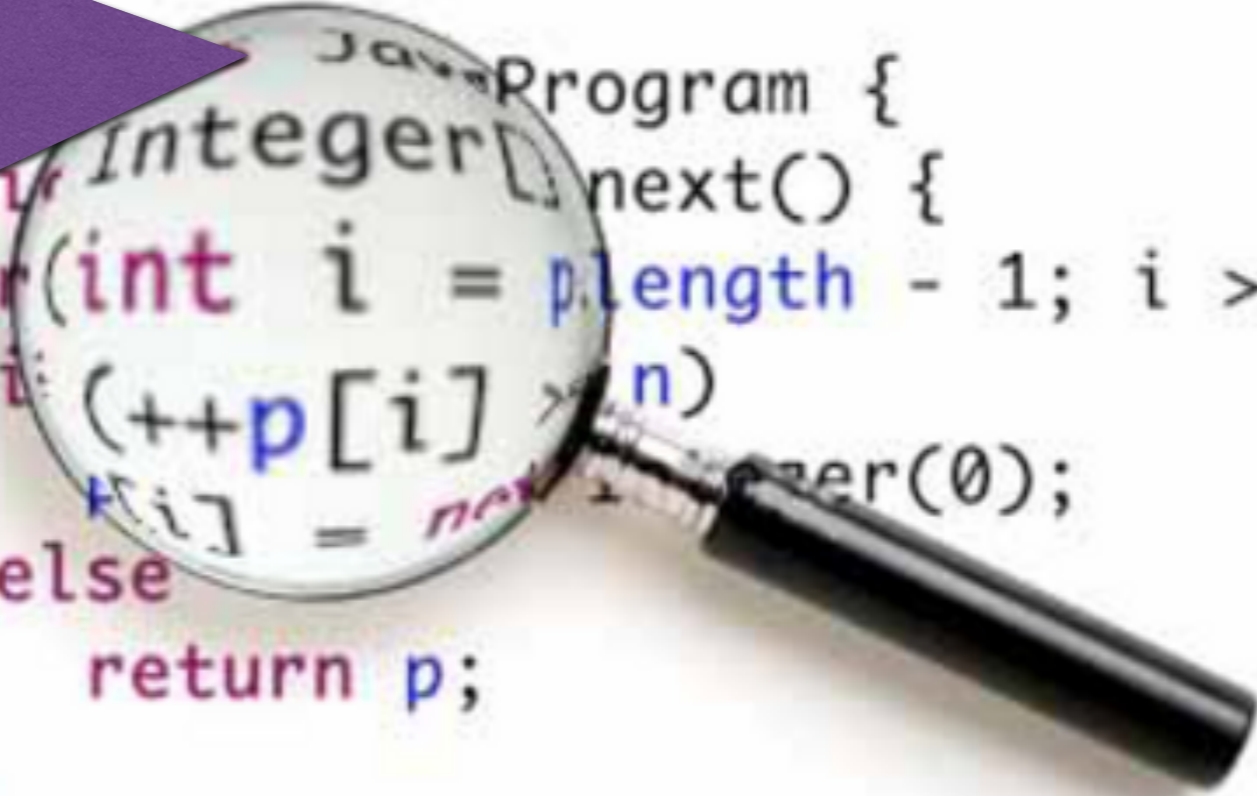


```
public class JavaProgram {  
    public Integer[] next() {  
        for (int i = p.length - 1; i >= 0;  
            i--)  
            if (++p[i] > n)  
                p[i] = nextInteger(0);  
        else  
            return p;  
    }  
    throw new NoSuchElementException();  
}
```

The image shows a magnifying glass with a black handle and a silver rim, positioned over a snippet of Java code. The lens of the magnifying glass is centered on the line `if (++p[i] > n)`, which is also highlighted in blue in the original image. The code is written in a monospaced font with syntax highlighting: keywords like `public`, `class`, `for`, `else`, and `throw` are in red; `Integer`, `int`, `return`, and `new` are in blue; and `next()`, `length`, `p`, `i`, `n`, and `0` are in black. The background is a light gray with a subtle grid pattern.

Fault localisation

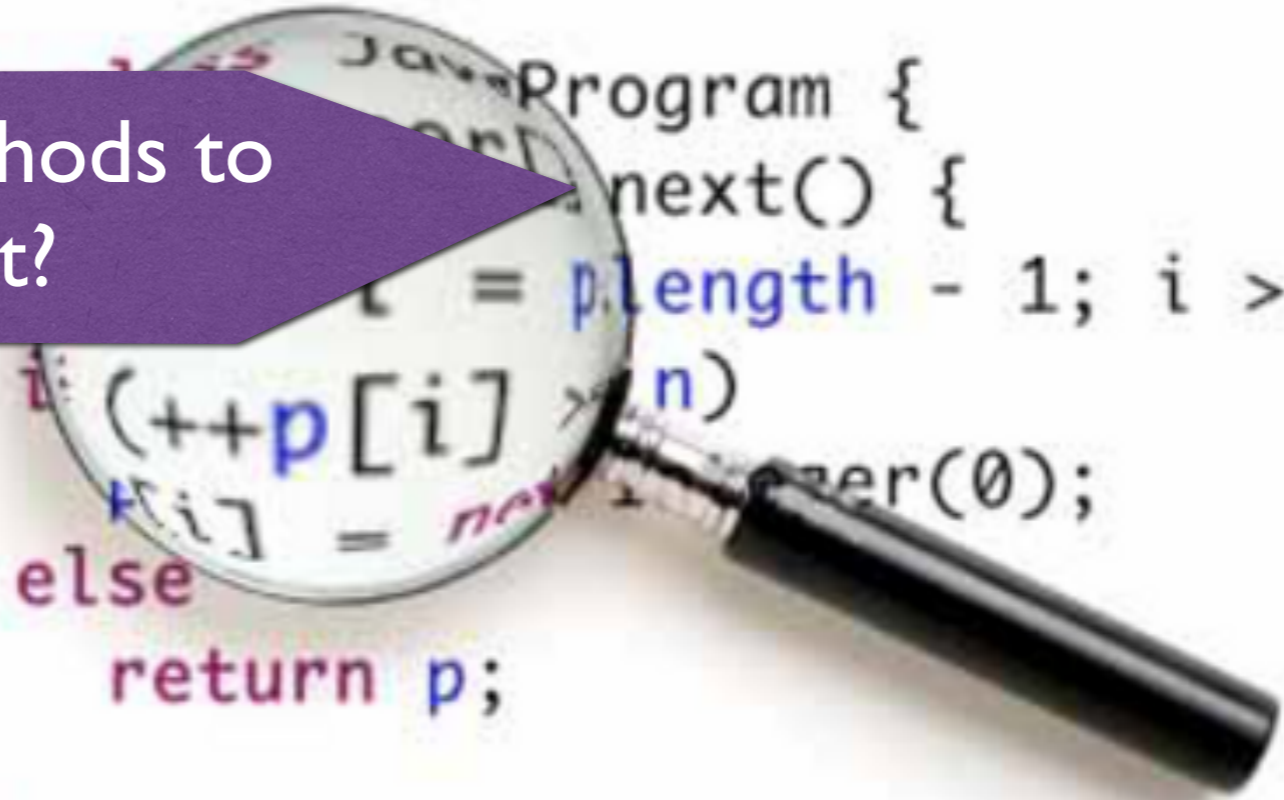
Which Classes to
look at?



```
JavaProgram {  
    Integer[] next() {  
        for (int i = p.length - 1; i >= 0;  
            i--)  
            if (++p[i] > n)  
                p[i] = nextInteger(0);  
        else  
            return p;  
    }  
    throw new NoSuchElementException();  
}
```

Fault localisation

Which Methods to
look at?

A magnifying glass with a black handle is positioned over a snippet of Java code. The lens is focused on the line `p[i] = nextElement(0);`. The code is color-coded: keywords like `else` and `return` are in red, `throw new` is in purple, and other tokens are in black. The background is a light gray with a subtle grid pattern.

```
JavaProgram {  
    next() {  
        i = p.length - 1; i >= 0;  
        if (i < 0) {  
            return null;  
        }  
        p[i] = nextElement(0);  
        else  
            return p;  
    }  
    throw new NoSuchElementException();  
}
```


Fault localisation

Which Statements
to look at?

```
public class JavaProgram {  
    public Integer[] next() {  
        for (int i = p.length - 1; i >= 0; i--)  
            if (p[i] > n)  
                p[i] = nextInteger(0);  
        else  
            return p;  
    }  
    throw new NoSuchElementException();  
}
```

Spectrum Based Fault Localisation

1. Input

2. Process

3. Output

Faulty Program



Test code



1
Run Tests and
Collect traces

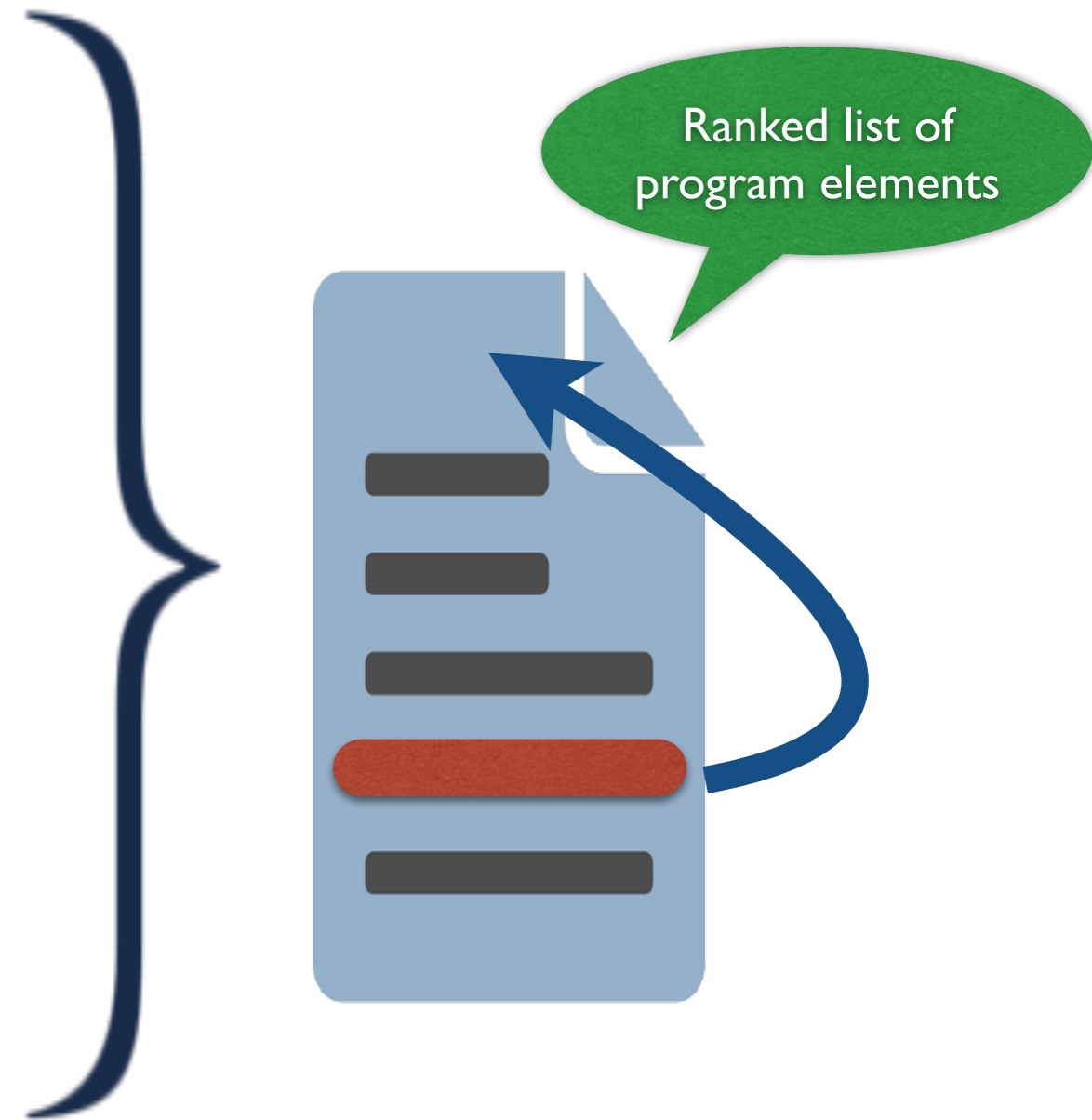
2
Build Program
Spectrum

3
Rank Program
Elements

Ranked list of
program elements



Spectrum Based Fault Localisation



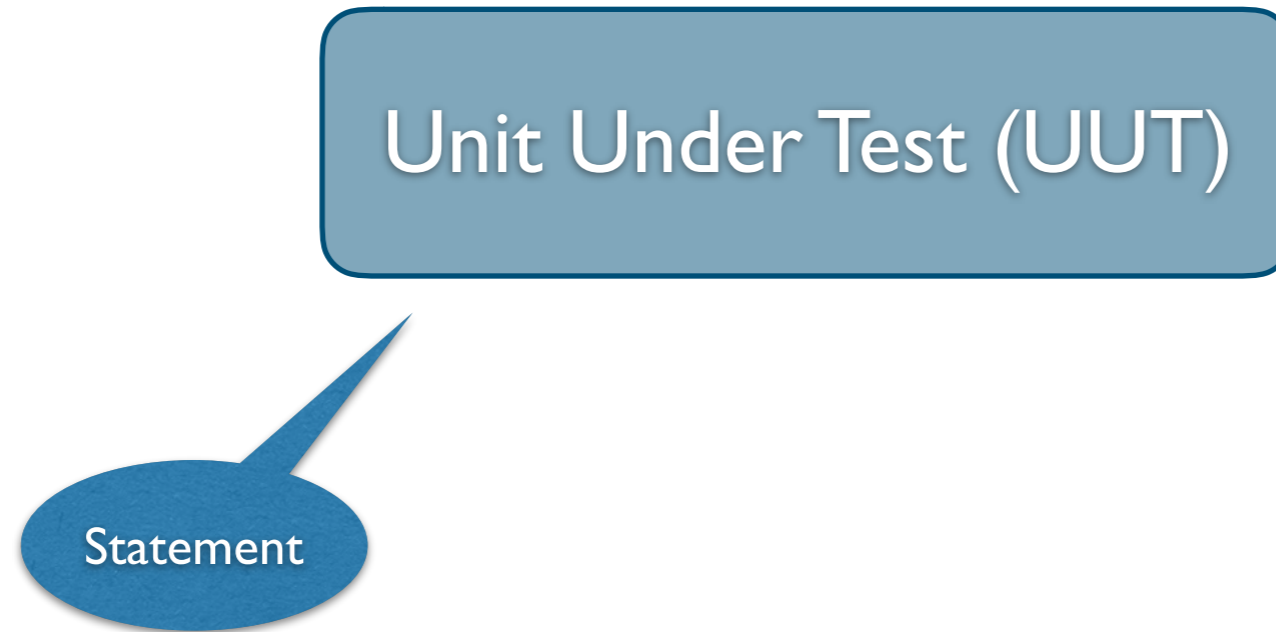
Spectrum Based Fault Localisation

I. Trace Collection

Unit Under Test (UUT)

Spectrum Based Fault Localisation

I. Trace Collection



Spectrum Based Fault Localisation

I. Trace Collection



Unit Under Test (UUT)

Block

Spectrum Based Fault Localisation

I. Trace Collection



Unit Under Test (UUT)

The diagram consists of a light blue rounded rectangular box containing the text 'Unit Under Test (UUT)'. A dark blue arrow points from the bottom right corner of this box to a dark blue oval containing the text 'Method'.

Method

Spectrum Based Fault Localisation

2. Program Spectrum

$$UUT_i = (e_f, e_p, n_f, n_p)$$

e_f = number of **failing test cases** that **execute** the UUT

e_p = number of **passing test cases** that **execute** the UUT

n_f = number of **failing test cases** that do **not execute** the UUT

n_p = number of **passing test cases** that do **not execute** the UUT

Spectrum Based Fault Localisation

2. Program Spectrum

Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p
UUT_1	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$				
...				
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$				

Spectrum Based Fault Localisation

2. Program Spectrum

Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p
UUT_1	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$				
...				
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$				

Spectrum Based Fault Localisation

2. Program Spectrum

Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p
UUT_1	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$	$\sum_{i=1}^m X_{i,j}$		$m - e_f$	
...				
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$				

Spectrum Based Fault Localisation

2. Program Spectrum

Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p
UUT_1	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$	$\sum_{i=1}^m X_{i,j}$	$\sum_{i=m+1}^n X_{i,j}$	$m - e_f$	$(n-m) - e_p$
...				
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$				

Spectrum Based Fault Localisation

3. Ranking

Suspiciousness of UUT

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum				Suspiciousness
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p	
UUT_1	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$					[0,1]
...					
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$					

Spectrum Based Fault Localisation

3. Ranking

Suspiciousness of UUT

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum				Suspiciousness
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p	
UUT_1	1	1	1	0	0	0					1
...					
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$					

Spectrum Based Fault Localisation

3. Ranking

Suspiciousness of UUT

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum				Suspiciousness
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p	
UUT_1	0	...	0	$X_{1,m+1}$...	$X_{1,n}$					0
...					
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$					

Spectrum Based Fault Localisation

3. Ranking

Suspiciousness of UUT

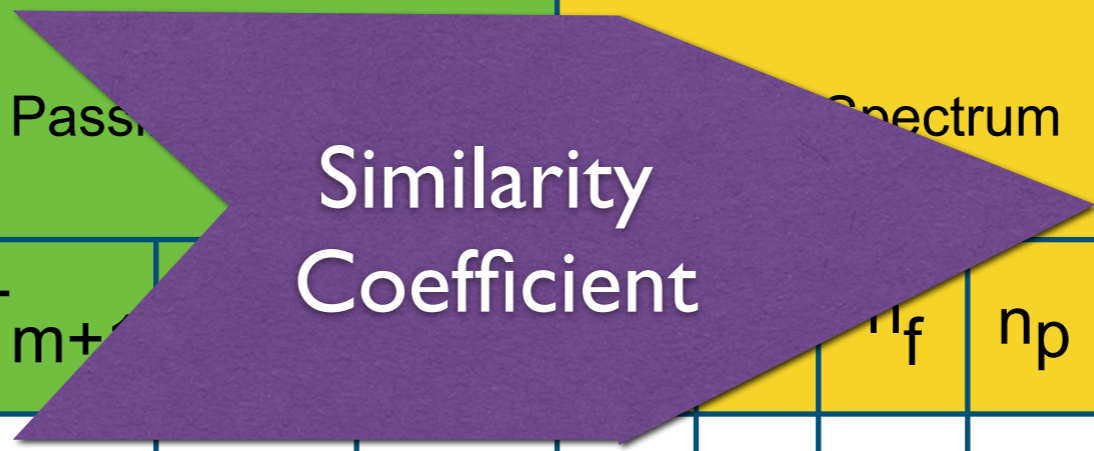
UUT	Failing Test Cases			Passing Test Cases			Program Spectrum				Suspiciousness
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p	
UUT_1	1	...	0	1	...	0					$0 < S < 1$
...					
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$					

Spectrum Based Fault Localisation

3. Ranking

Suspiciousness of UUT

UUT	Failing Test Cases			Passing Test Cases	Spectrum			Suspiciousness
	T_1	...	T_m	T_{m+1}	...	T_n	n_p	
UUT_1	1	...	0	1	...	0		
...		
UUT_N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$		



Spectrum Based Fault Localisation (The Problem)

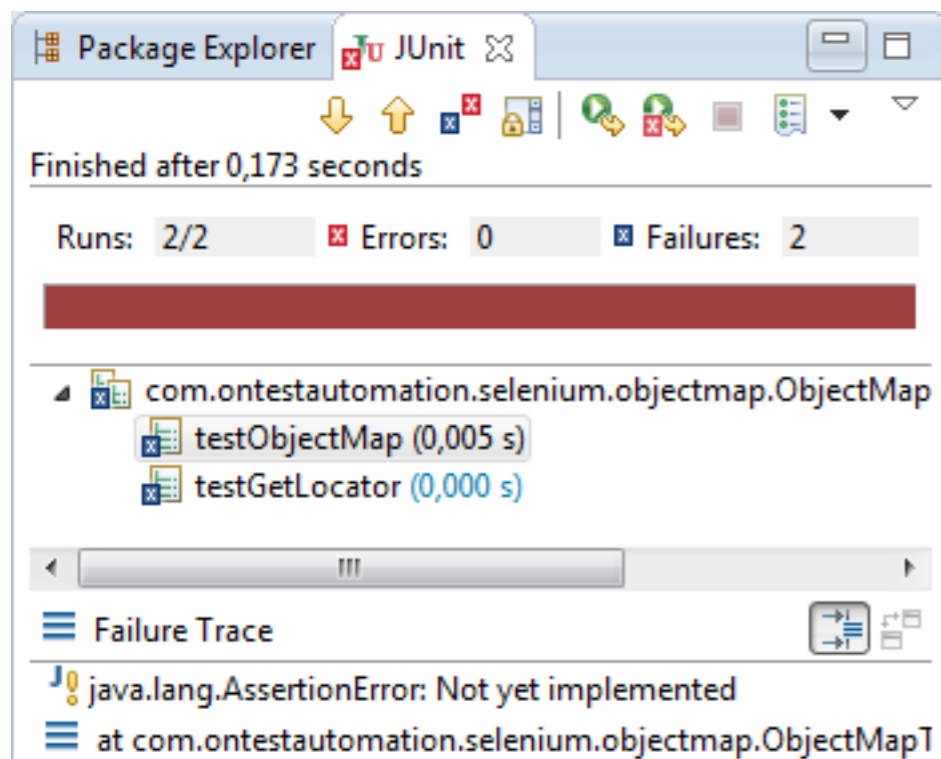
Context: when the **UUT** is a **method**

```
public void method() {  
    .....  
    methodA()  
    methodB()  
    if (condition) {  
        return  
    }  
    .....  
    methodC()  
    .....  
}
```



Spectrum Based Fault Localisation (The Problem)

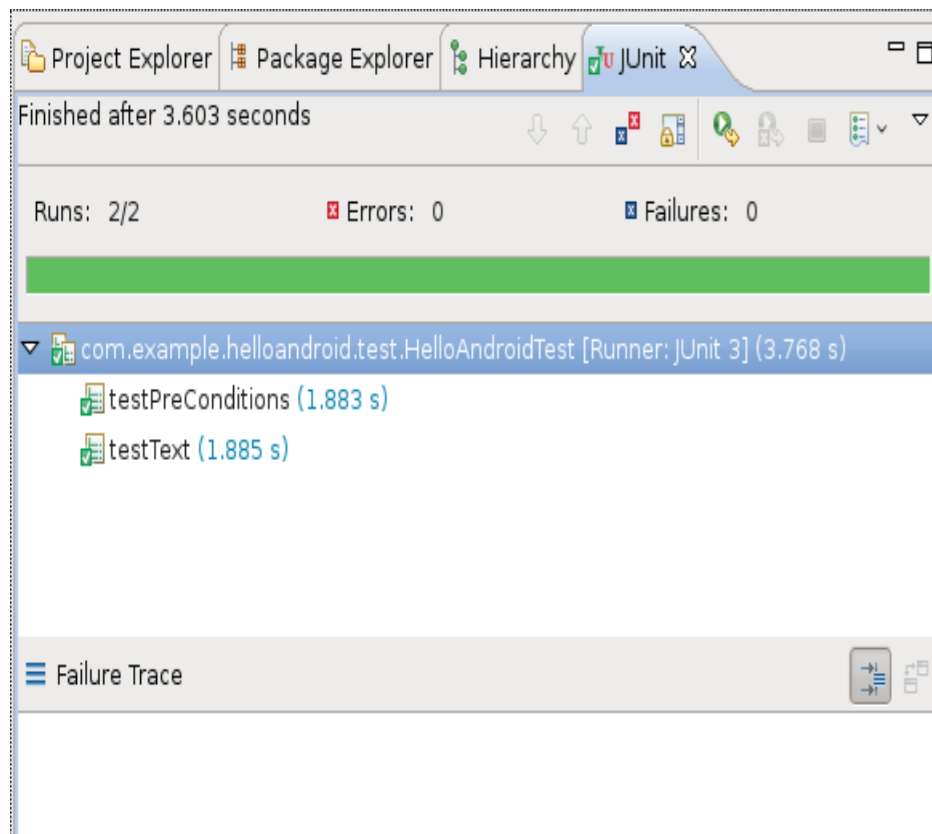
Context: when the **UUT** is a **method**



```
public void method() {  
    .....  
    methodA()   
    methodB()   
    if(condition) {  
        return  
    }  
    .....  
    methodC()   
    .....  
}
```

Spectrum Based Fault Localisation (The Problem)

Context: when the UUT is a method



```
public void method() {  
    .....  
    methodA()   
    methodB()   
    if (condition) {  
        return   
    }  
    .....  
    methodC()   
    .....  
}
```

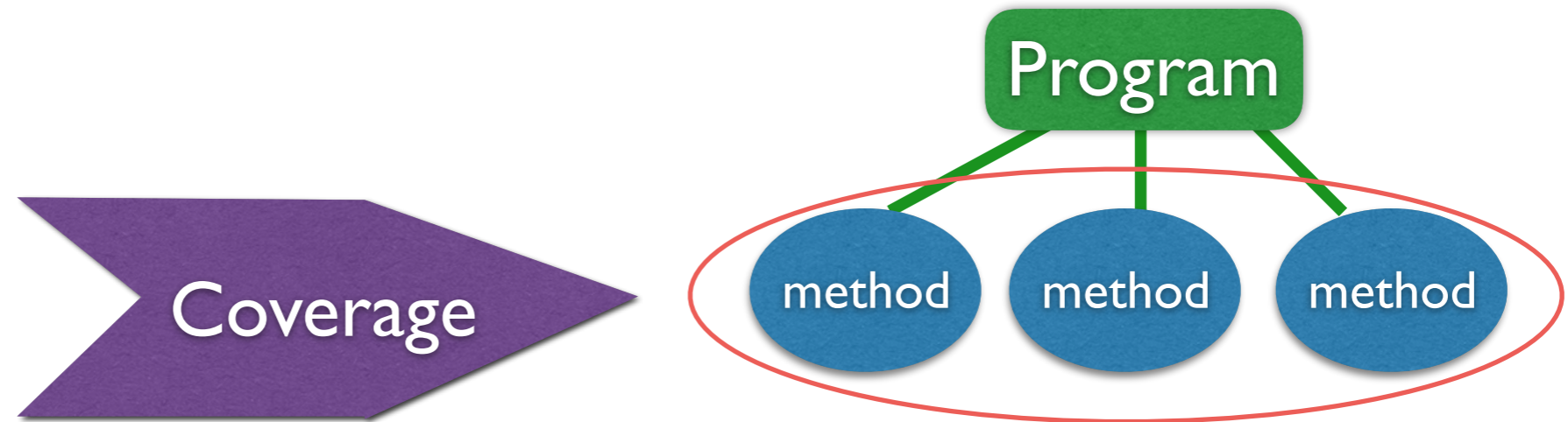
Spectrum Based Fault Localisation (The Problem)

Context: when the **UUT** is a **method**

```
public void method() {  
    .....  
    methodA()  
    methodB()  
    if (condition) {  
        return  
    }  
    .....  
    methodC()  
    .....  
}
```

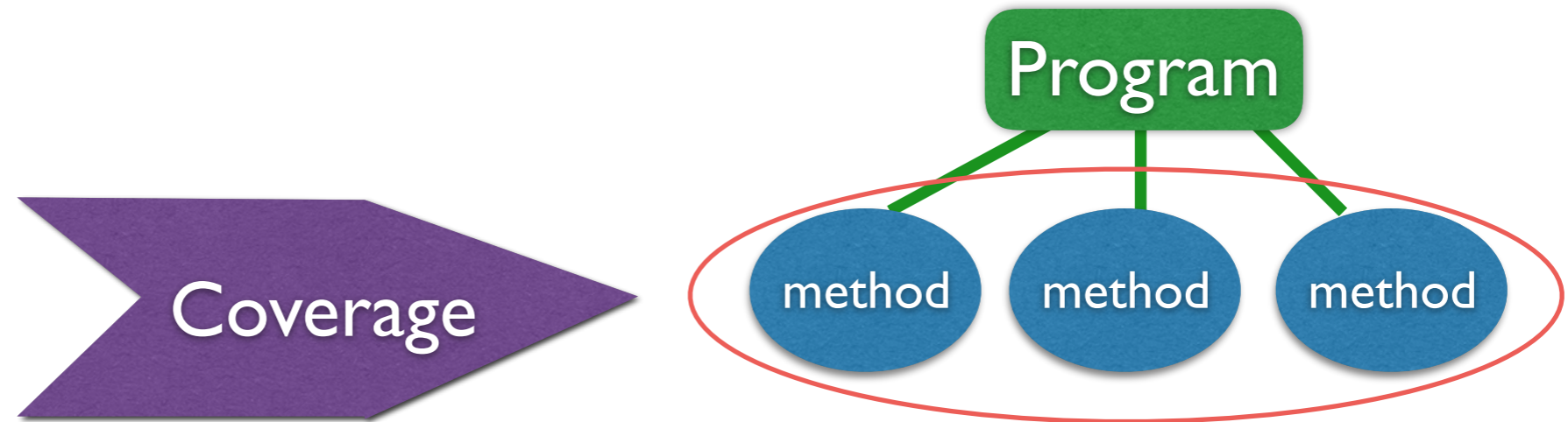
Spectrum Based Fault Localisation (The Problem)

Context: when the **UUT** is a **method**



Spectrum Based Fault Localisation (The Problem)

Context: when the **UUT** is a **method**



Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p
UUT ₁	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$				
...				
UUT _N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$				

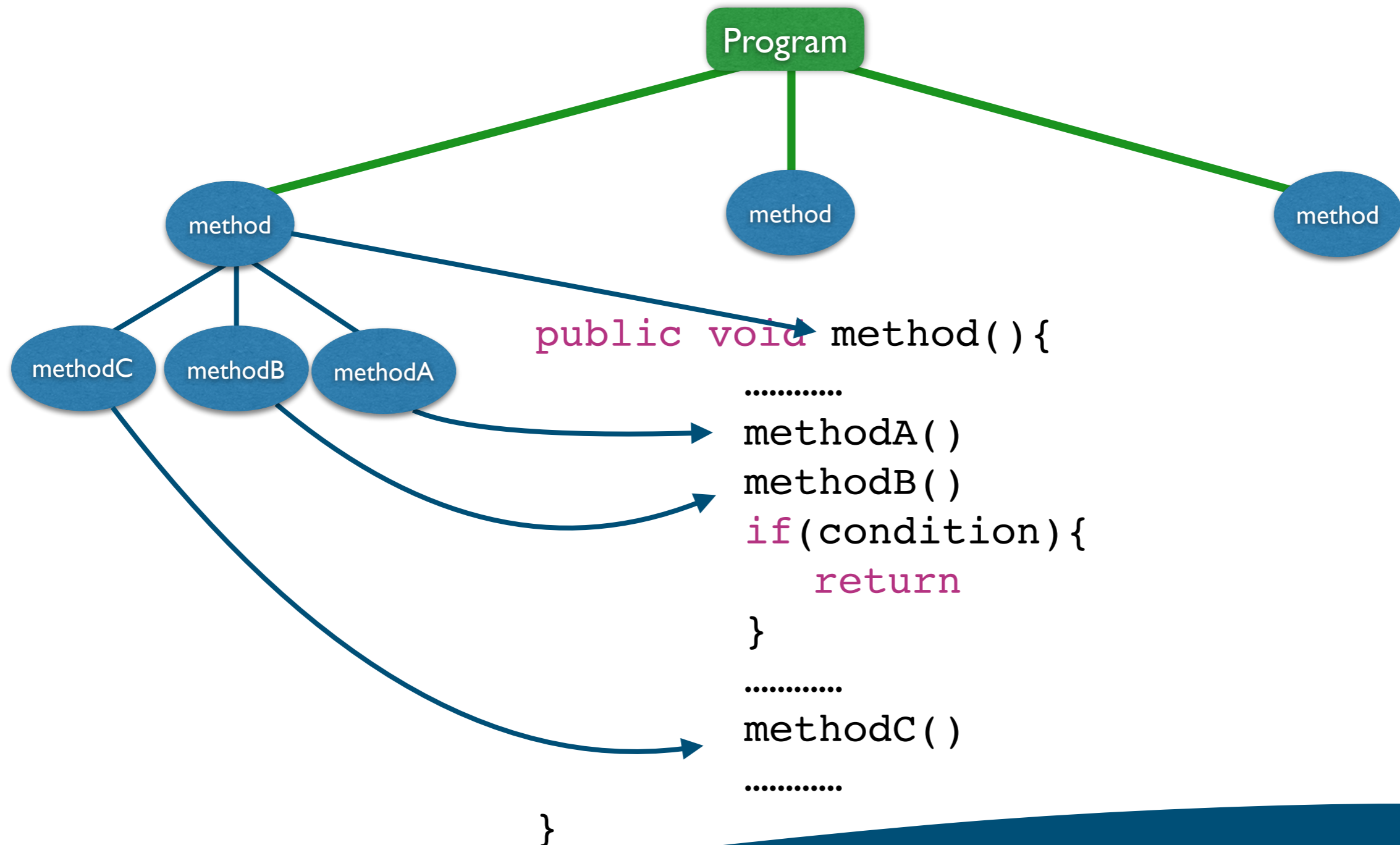
Spectrum Based Fault Localisation (The Solution)

Context: when the **UUT** is a **method**



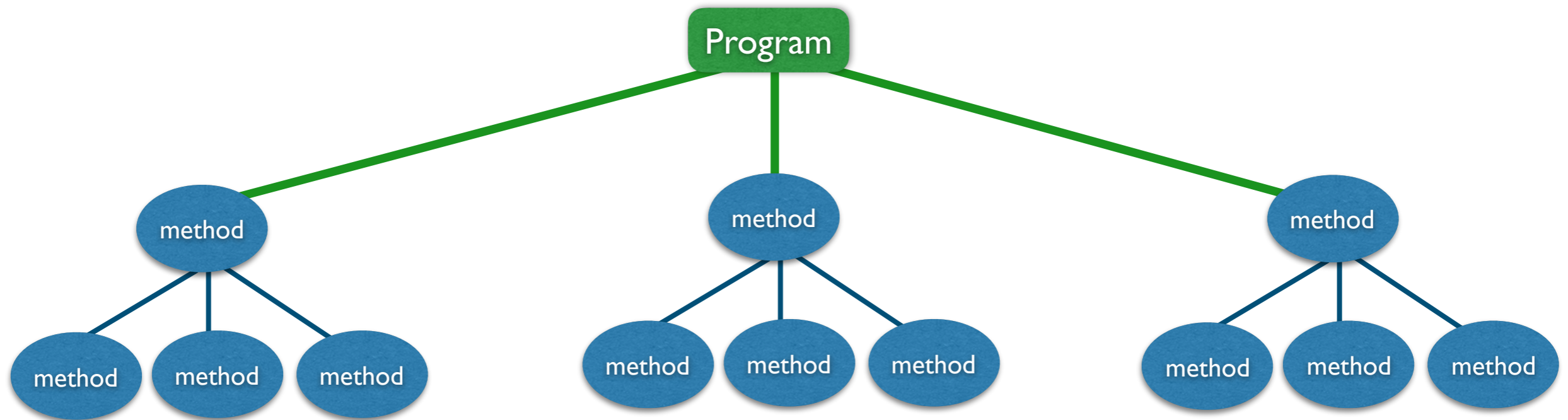
Spectrum Based Fault Localisation (The Solution)

Context: when the **UUT** is a **method**



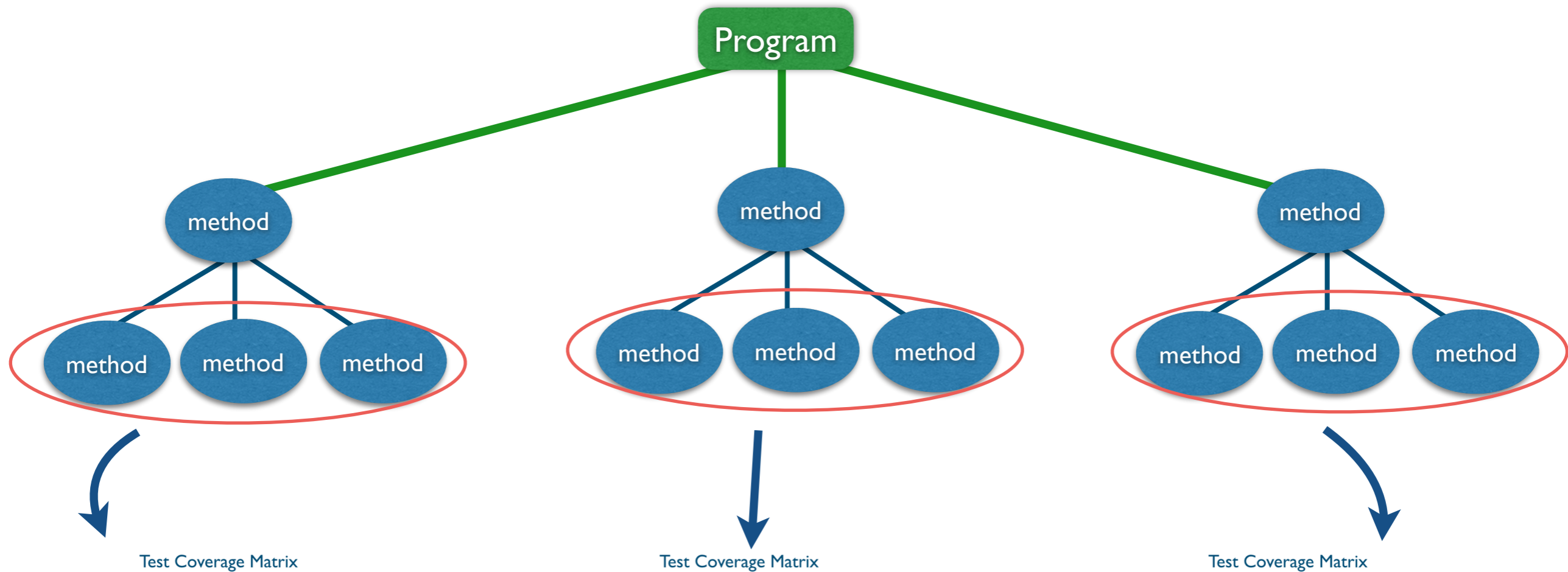
Spectrum Based Fault Localisation (The Solution)

Context: when the **UUT** is a **method**



Spectrum Based Fault Localisation (The Solution)

Context: when the UUT is a method



Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p
UUT ₁	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$				
...				
UUT _N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$				

Test Coverage Matrix

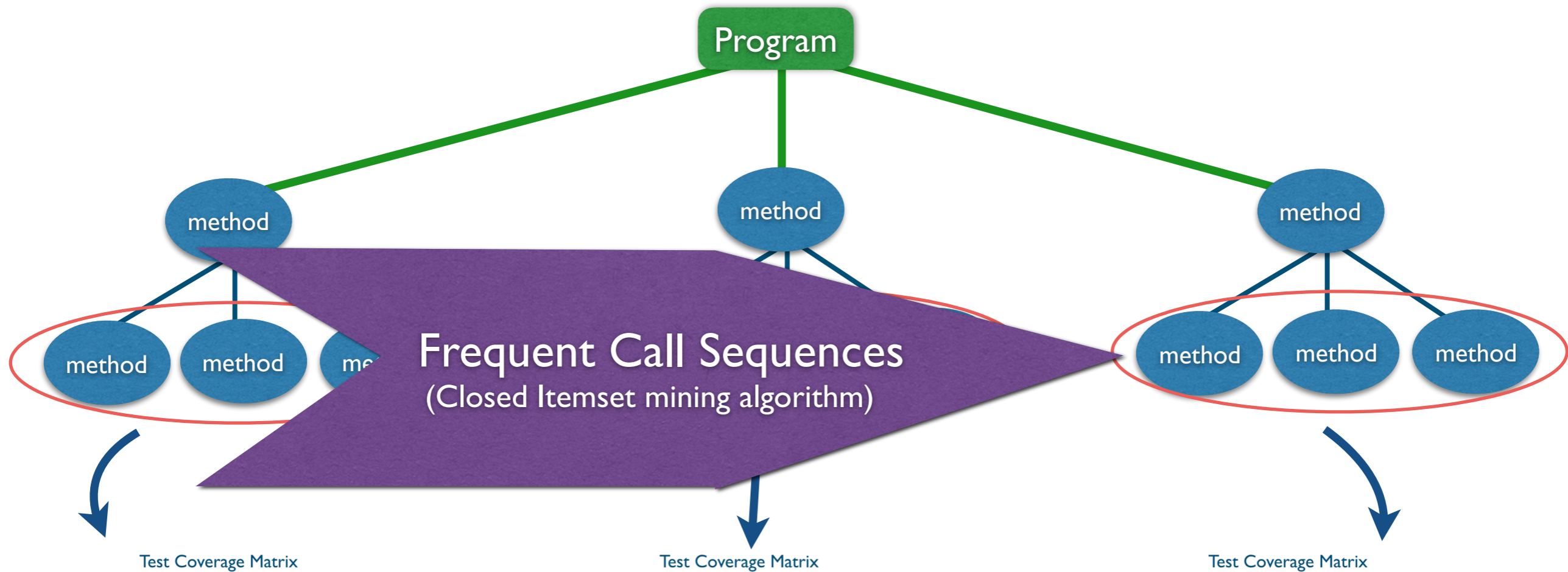
UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p
UUT ₁	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$				
...				
UUT _N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$				

Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T_1	...	T_m	T_{m+1}	...	T_n	e_f	e_p	n_f	n_p
UUT ₁	$X_{1,1}$...	$X_{1,m}$	$X_{1,m+1}$...	$X_{1,n}$				
...				
UUT _N	$X_{N,1}$...	$X_{N,m}$	$X_{N,m+1}$...	$X_{N,n}$				

Spectrum Based Fault Localisation (The Solution)

Context: when the UUT is a method



Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T ₁	...	T _m	T _{m+1}	...	T _n	e _f	e _p	n _f	n _p
UUT ₁	X _{1,1}	...	X _{1,m}	X _{1,m+1}	...	X _{1,n}				
...				
UUT _N	X _{N,1}	...	X _{N,m}	X _{N,m+1}	...	X _{N,n}				

Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T ₁	...	T _m	T _{m+1}	...	T _n	e _f	e _p	n _f	n _p
UUT ₁	X _{1,1}	...	X _{1,m}	X _{1,m+1}	...	X _{1,n}				
...				
UUT _N	X _{N,1}	...	X _{N,m}	X _{N,m+1}	...	X _{N,n}				

Test Coverage Matrix

UUT	Failing Test Cases			Passing Test Cases			Program Spectrum			
	T ₁	...	T _m	T _{m+1}	...	T _n	e _f	e _p	n _f	n _p
UUT ₁	X _{1,1}	...	X _{1,m}	X _{1,m+1}	...	X _{1,n}				
...				
UUT _N	X _{N,1}	...	X _{N,m}	X _{N,m+1}	...	X _{N,n}				

Spectrum Based Fault Localisation (The Solution)

Context: when the **UUT** is a **method**

Intuition

A method whose **call sequences** differ in failing and passing test cases is **more suspicious**

Experimental Setup 1/2

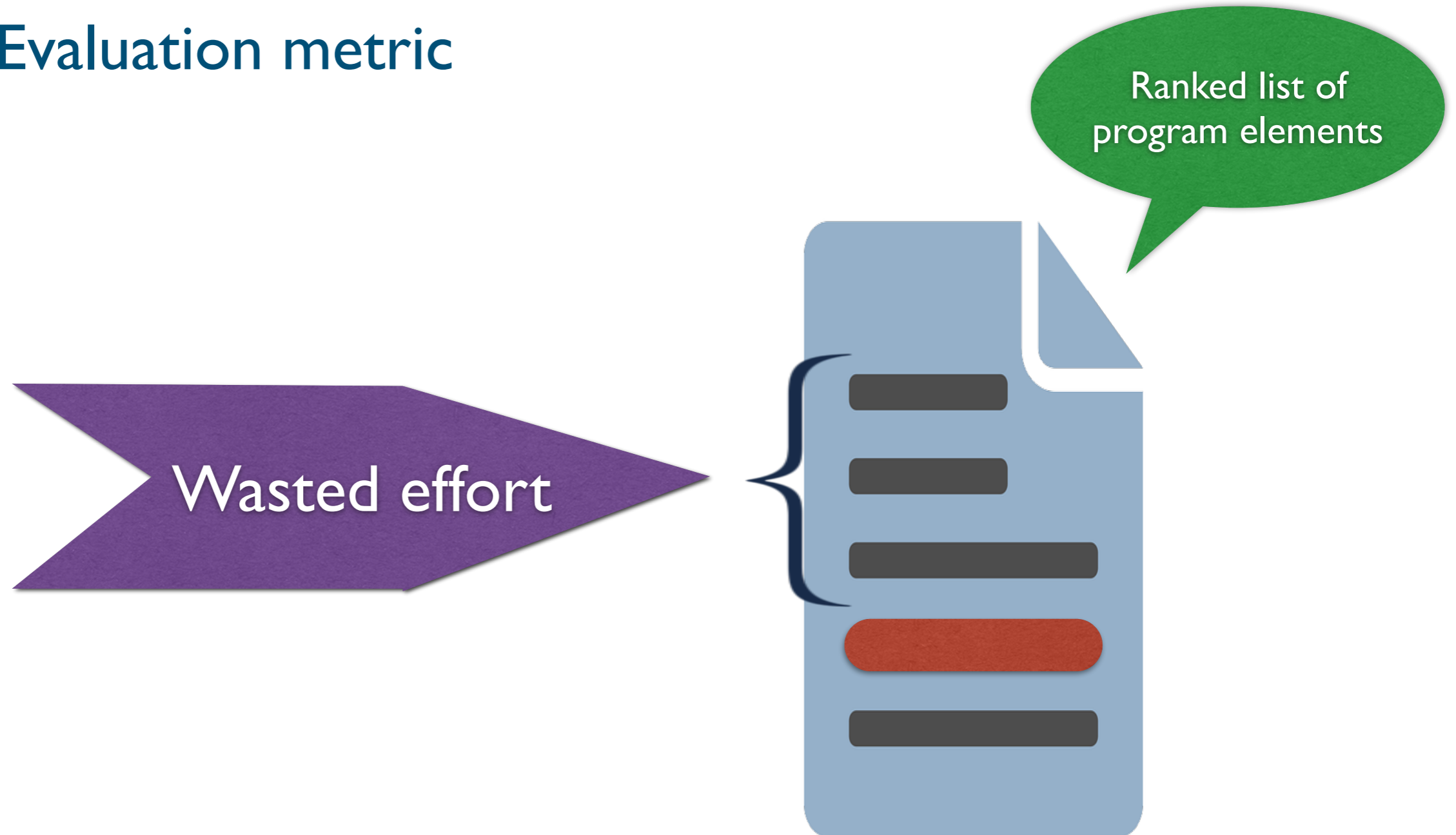
Project*	# of versions	LOC	# of faults
NanoXML	4	7646	16
JMeter	2	43400	3

* Downloaded from Software-artifact Infrastructure Repository (<http://sir.unl.edu>)

- One fault at a time
- Single failing and all passing test cases

Experimental Setup 2/2

Evaluation metric

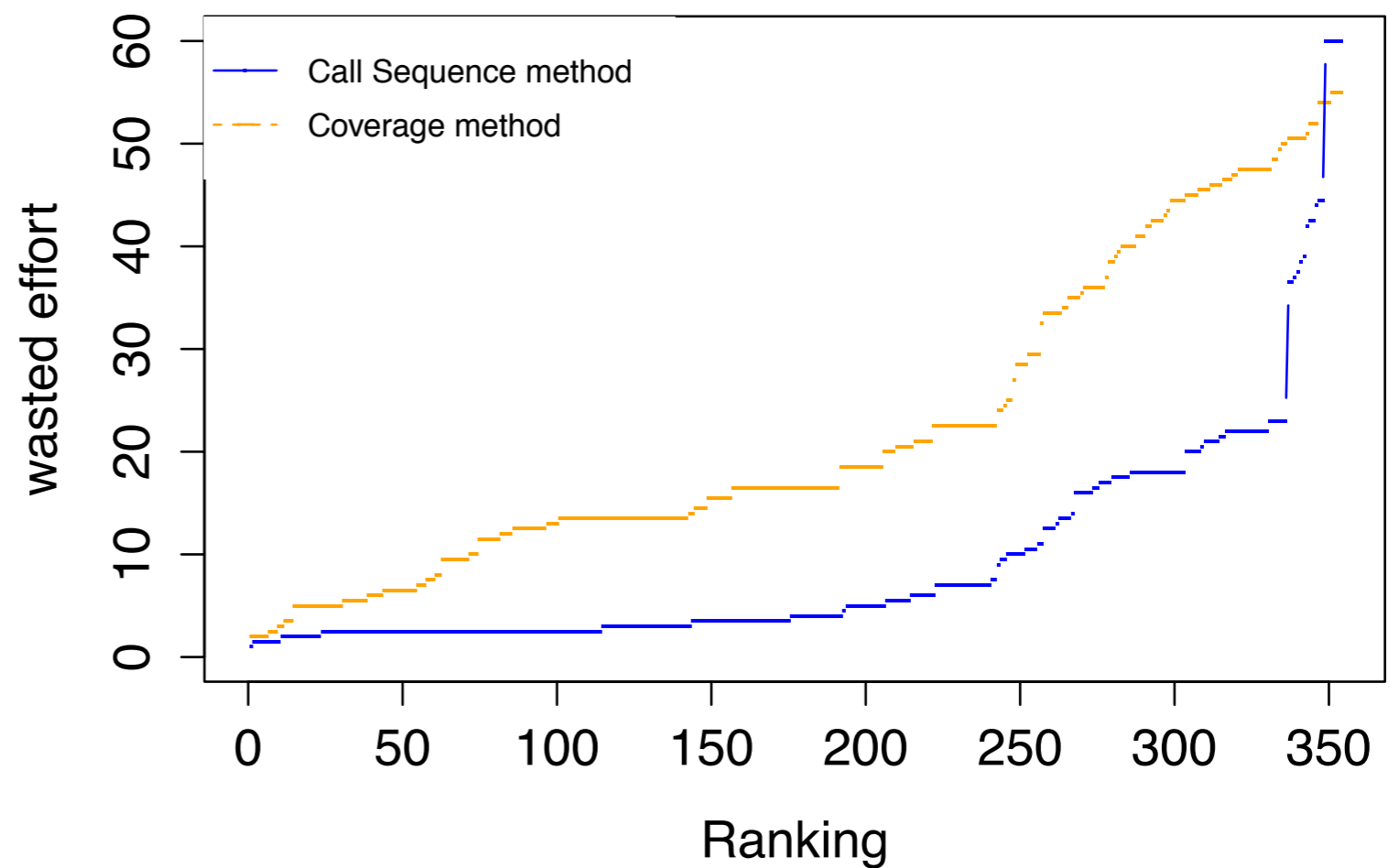


Results and Discussion - 1/2

Total 354 rankings

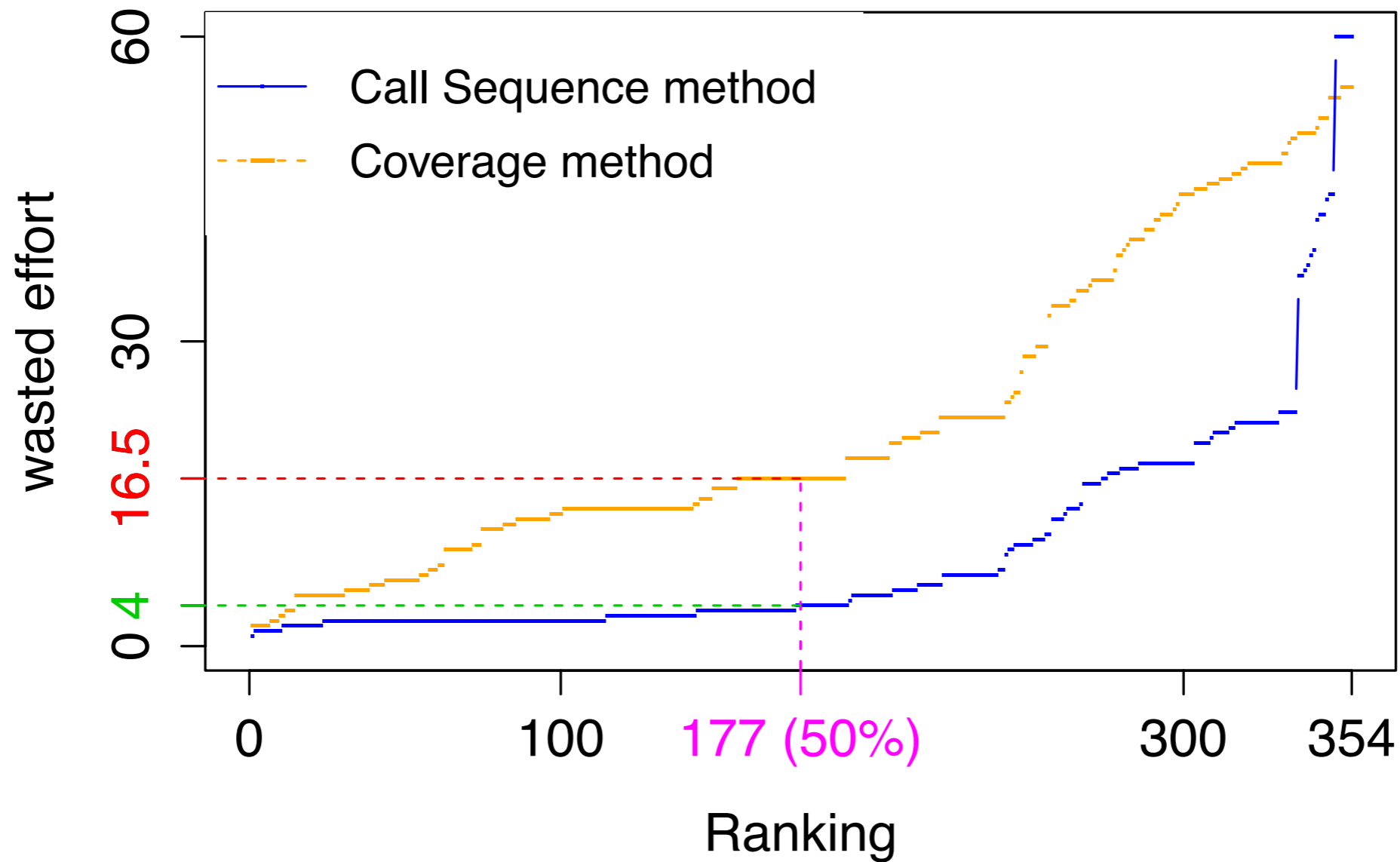
Average Wasted effort

	Call Sequence method	Coverage method
mean	9.3	22.2
Std. Dev	11.1	14.9



Results and Discussion - 2/2

Total 354 rankings



Summary

