Scent Intensification for Testing & Debugging

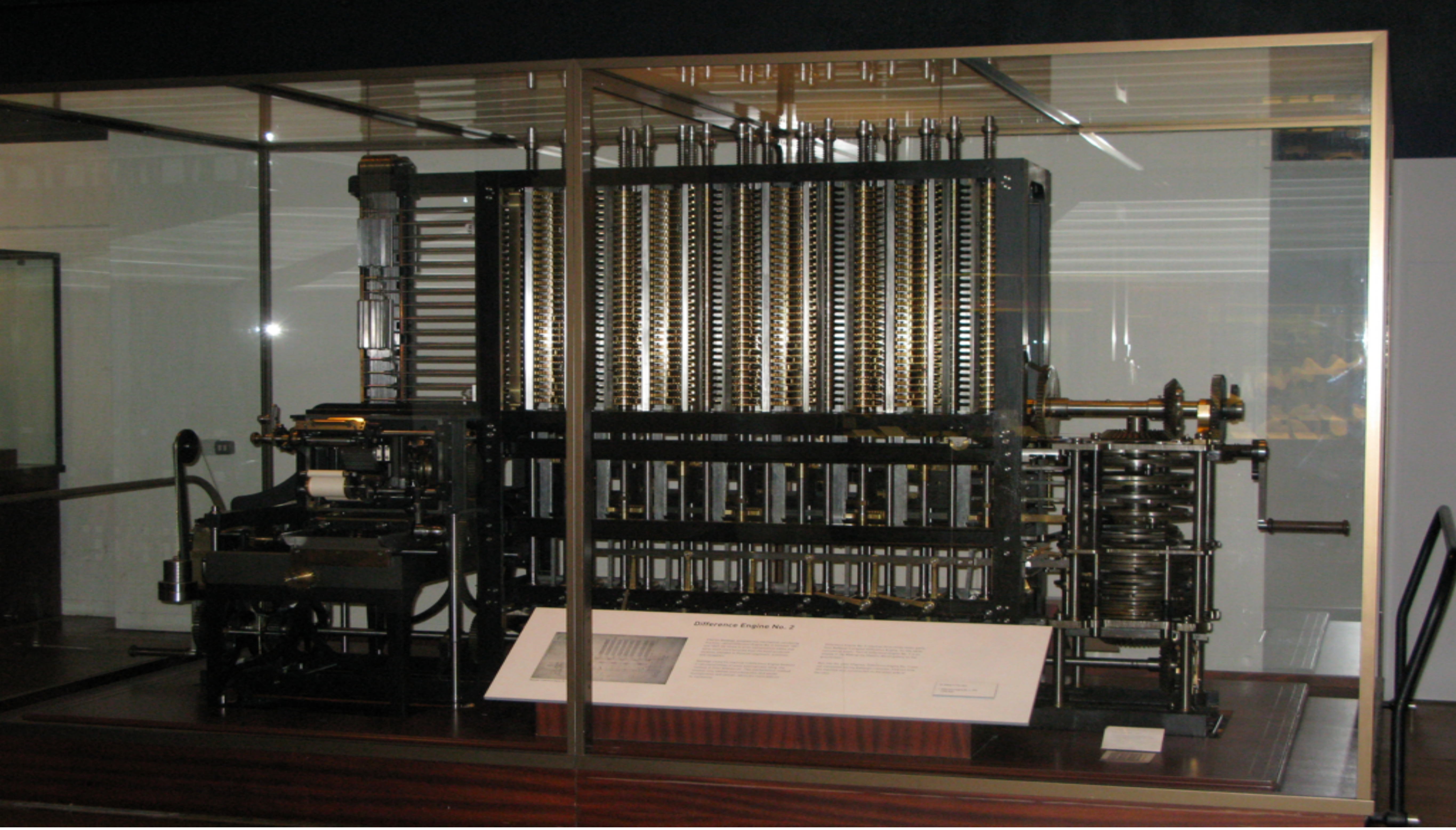
Rui Abreu

parc®
Palo Alto Research Center

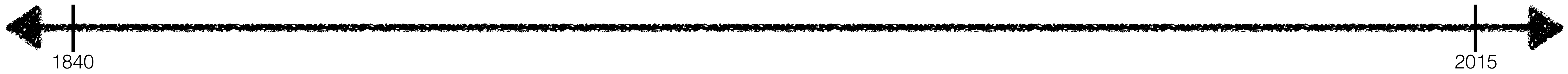
Economic Relevance

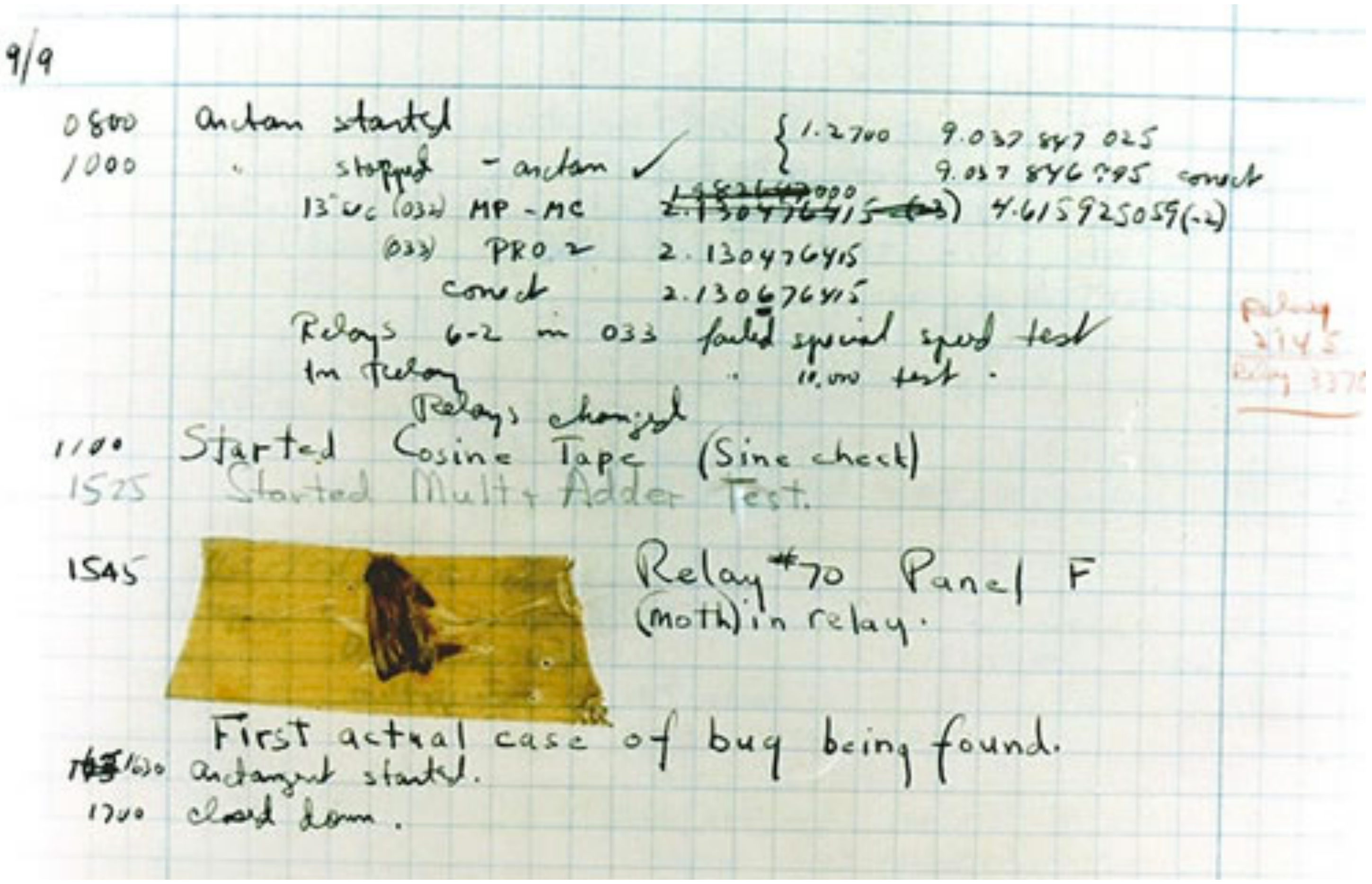
- [Embedded] Software
    - Exponential ***increase*** LOC
    - Despite thorough design / testing, **constant fault density**
    - Typically 5-15bugs / KLOC, 75 min / bug ➤ **$4K/KLOC**
    - Development cost $15-30K / KLOC ➤ **15-25% diagnostic cost**
- Residual defects cost **US $60B/year** [NIST 2002]
    - estimated **20%** due to **fault diagnosis** (downtime, labor)

The birth of debugging: your guess?



Software Errors mentioned in Ada Byron's notes on Charles Bababage's analytical engine



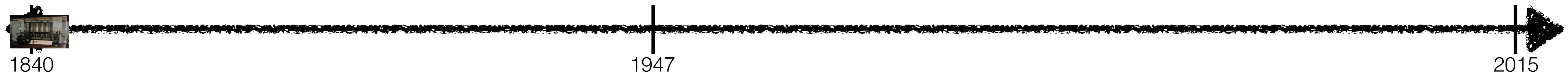


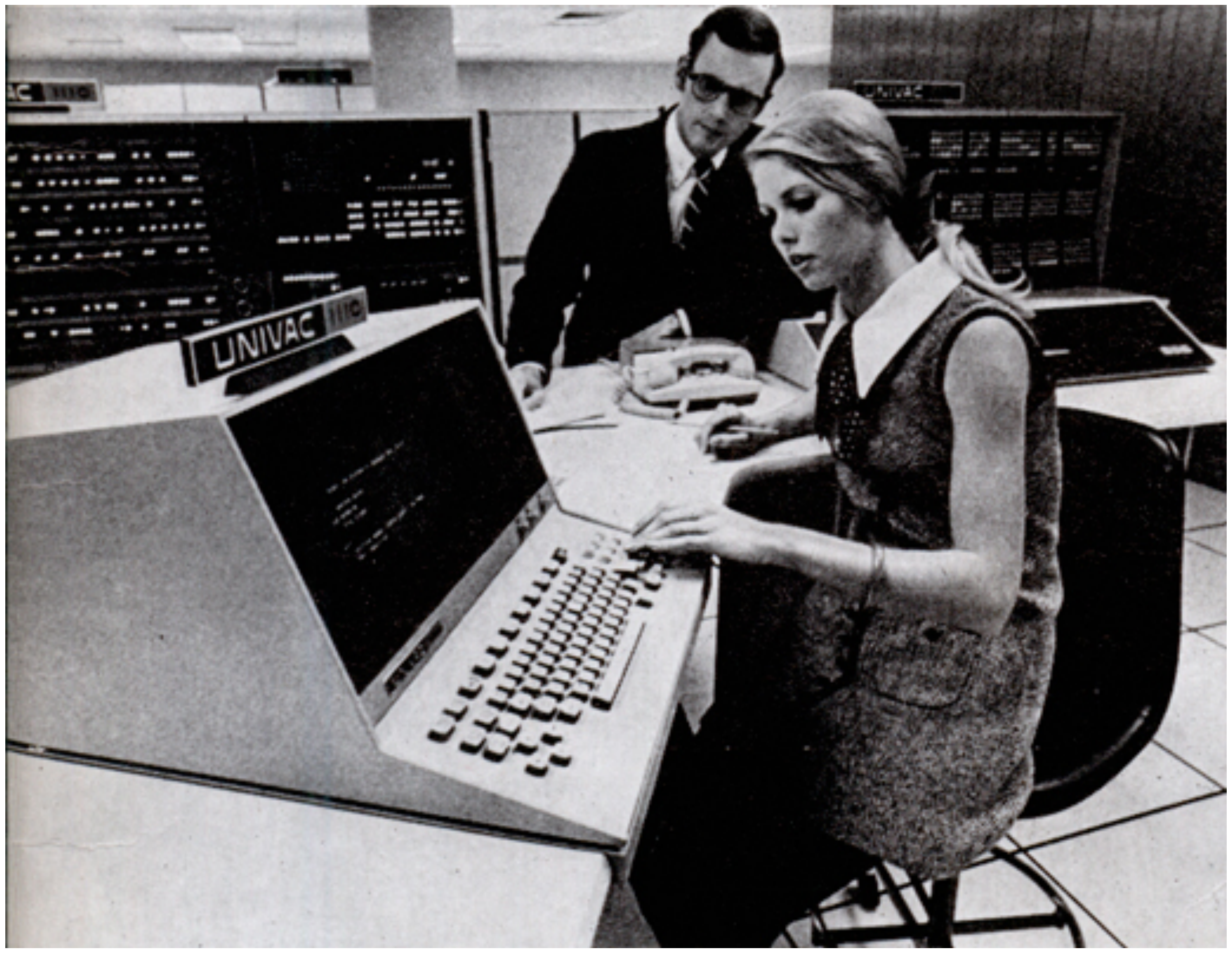
9/9

0800 antan started
1000 " stopped - antan ✓ {1.2700 9.037 847 025
9.037 846 995 const
13" uc (032) MP - MC ~~1.982147000~~
~~2.130476415 (-3)~~ 4.615925059(-2)
(033) PRO 2 2.130476415
const 2.130676415
Relays 6-2 in 033 failed special speed test
in relay " 11,000 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1630 antangent started.
1700 closed down.

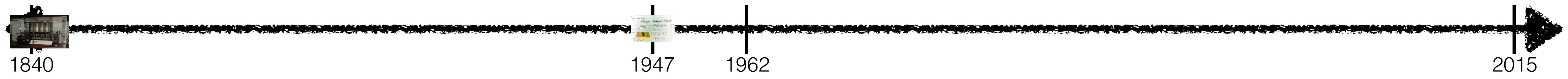
Relay 2145
Relay 3370

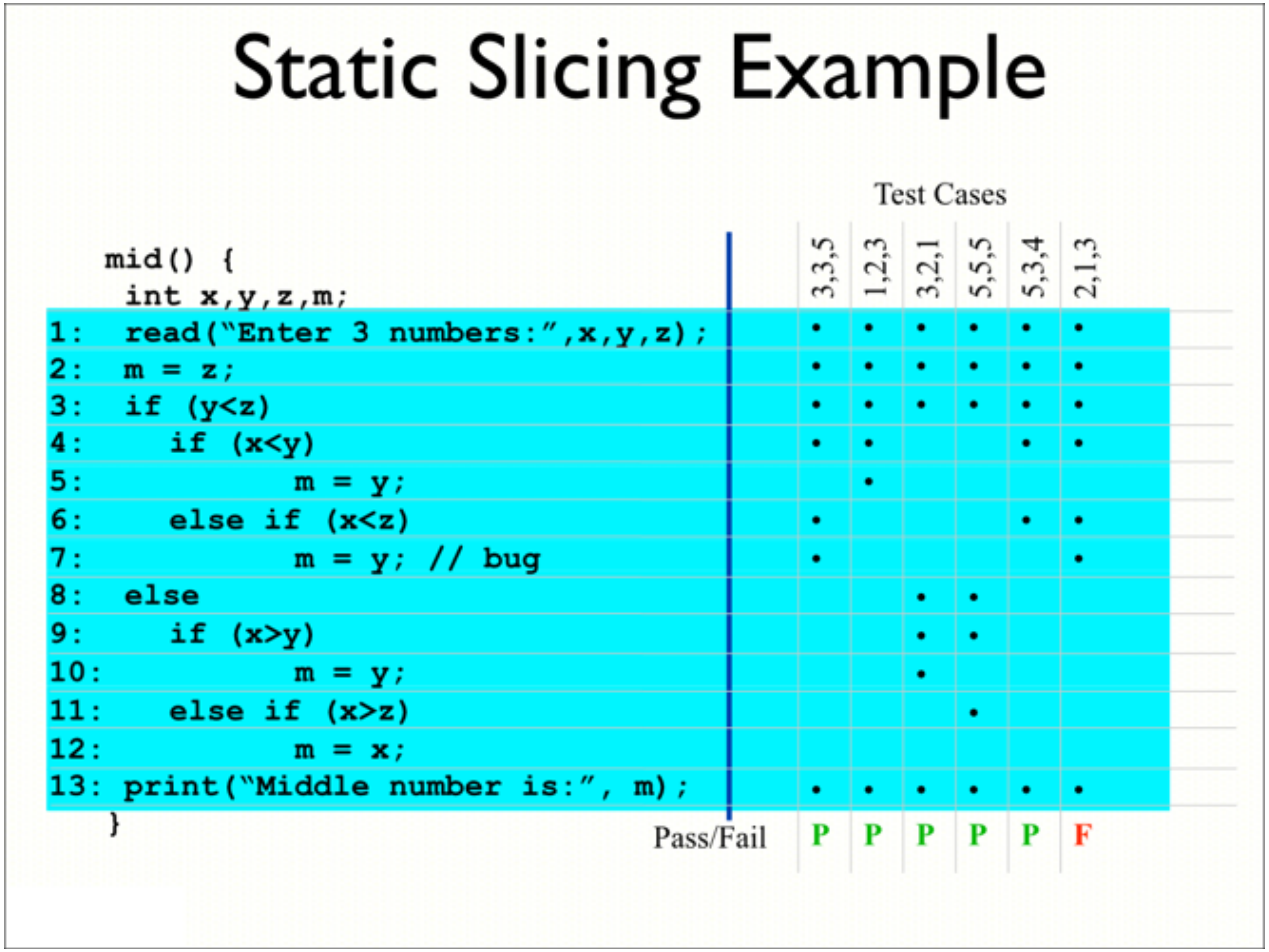
First actual bug and actual debugging: Admiral Grace Hopper's associates working on Mark II Computer at Harvard University





UNIVAC 1100's FLIT - Fault Localization by Interpretive Testing

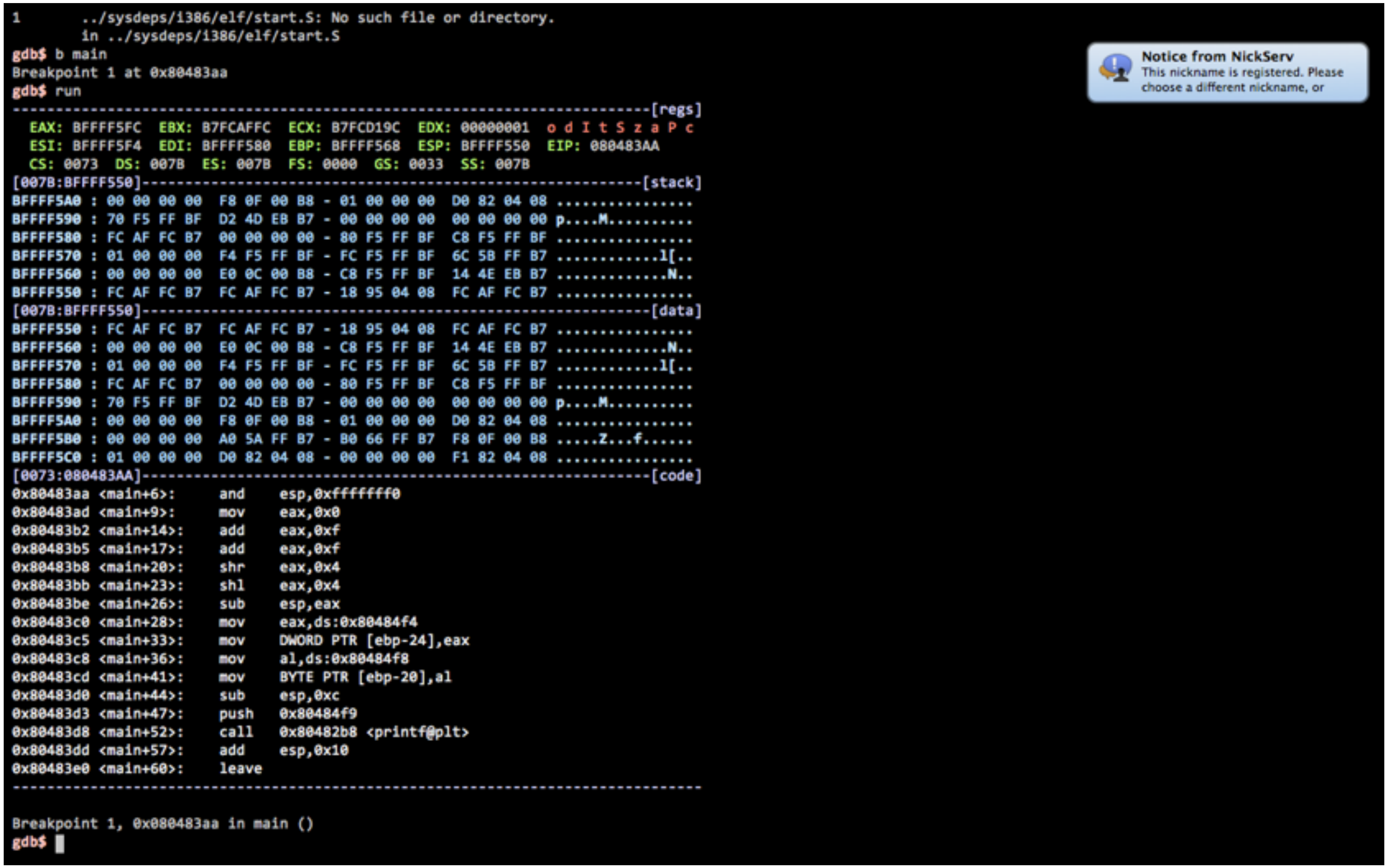




Static Slicing Example

	Test Cases					
	3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
`mid() {`						
`int x,y,z,m;`						
`1: read("Enter 3 numbers:",x,y,z);`	•	•	•	•	•	•
`2: m = z;`	•	•	•	•	•	•
`3: if (y<z)`	•	•	•	•	•	•
`4:    if (x<y)`	•	•			•	•
`5:          m = y;`		•				
`6:    else if (x<z)`	•				•	•
`7:          m = y; // bug`	•					•
`8: else`			•	•		
`9:    if (x>y)`			•	•		
`10:         m = y;`			•			
`11:   else if (x>z)`				•		
`12:         m = x;`						
`13: print("Middle number is:", m);`	•	•	•	•	•	•
`}`						
Pass/Fail	P	P	P	P	P	F

Weiser's Breakthrough paper.
Input: source code and program point

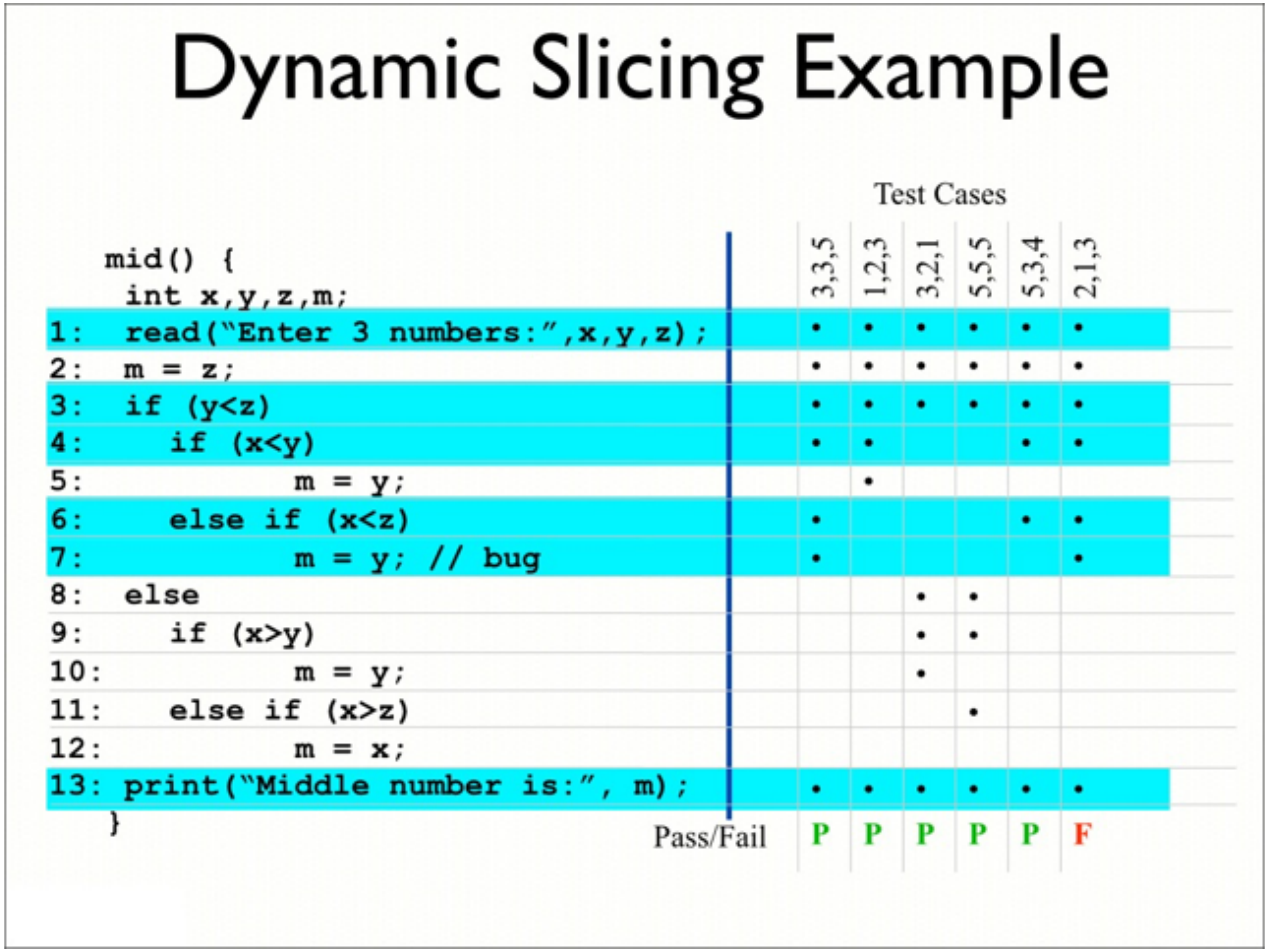


```
1       ../sysdeps/i386/elf/start.S: No such file or directory.
        in ../sysdeps/i386/elf/start.S
gdb$ b main
Breakpoint 1 at 0x80483aa
gdb$ run
--------------------------------------------------------------------------[regs]
  EAX: BFFFF5FC  EBX: B7FCAFFC  ECX: B7FCD19C  EDX: 00000001  o d I t S z a P c
  ESI: BFFFF5F4  EDI: BFFFF580  EBP: BFFFF568  ESP: BFFFF550  EIP: 080483AA
  CS: 0073  DS: 007B  ES: 007B  FS: 0000  GS: 0033  SS: 007B
[007B:BFFFF550]---------------------------------------------------------[stack]
BFFFF5A0 : 00 00 00 00  F8 0F 00 B8 - 01 00 00 00  D0 82 04 08 ................
BFFFF590 : 70 F5 FF BF  D2 4D EB B7 - 00 00 00 00  00 00 00 00 p....M..........
BFFFF580 : FC AF FC B7  00 00 00 00 - 80 F5 FF BF  C8 F5 FF BF ................
BFFFF570 : 01 00 00 00  F4 F5 FF BF - FC F5 FF BF  6C 5B FF B7 ............l[..
BFFFF560 : 00 00 00 00  E0 0C 00 B8 - C8 F5 FF BF  14 4E EB B7 .............N..
BFFFF550 : FC AF FC B7  FC AF FC B7 - 18 95 04 08  FC AF FC B7 ................
[007B:BFFFF550]----------------------------------------------------------[data]
BFFFF550 : FC AF FC B7  FC AF FC B7 - 18 95 04 08  FC AF FC B7 ................
BFFFF560 : 00 00 00 00  E0 0C 00 B8 - C8 F5 FF BF  14 4E EB B7 .............N..
BFFFF570 : 01 00 00 00  F4 F5 FF BF - FC F5 FF BF  6C 5B FF B7 ............l[..
BFFFF580 : FC AF FC B7  00 00 00 00 - 80 F5 FF BF  C8 F5 FF BF ................
BFFFF590 : 70 F5 FF BF  D2 4D EB B7 - 00 00 00 00  00 00 00 00 p....M..........
BFFFF5A0 : 00 00 00 00  F8 0F 00 B8 - 01 00 00 00  D0 82 04 08 ................
BFFFF5B0 : 00 00 00 00  A0 5A FF B7 - B0 66 FF B7  F8 0F 00 B8 .....Z...f......
BFFFF5C0 : 01 00 00 00  D0 82 04 08 - 00 00 00 00  F1 82 04 08 ................
[0073:080483AA]----------------------------------------------------------[code]
0x80483aa <main+6>:     and    esp,0xfffffff0
0x80483ad <main+9>:     mov    eax,0x0
0x80483b2 <main+14>:    add    eax,0xf
0x80483b5 <main+17>:    add    eax,0xf
0x80483b8 <main+20>:    shr    eax,0x4
0x80483bb <main+23>:    shl    eax,0x4
0x80483be <main+26>:    sub    esp,eax
0x80483c0 <main+28>:    mov    eax,ds:0x80484f4
0x80483c5 <main+33>:    mov    DWORD PTR [ebp-24],eax
0x80483c8 <main+36>:    mov    al,ds:0x80484f8
0x80483cd <main+41>:    mov    BYTE PTR [ebp-20],al
0x80483d0 <main+44>:    sub    esp,0xc
0x80483d3 <main+47>:    push   0x80484f9
0x80483d8 <main+52>:    call   0x80482b8 <printf@plt>
0x80483dd <main+57>:    add    esp,0x10
0x80483e0 <main+60>:    leave
--------------------------------------------------------------------------------

Breakpoint 1, 0x080483aa in main ()
gdb$
```

Notice from NickServ
This nickname is registered. Please choose a different nickname, or

Stallman's GDB

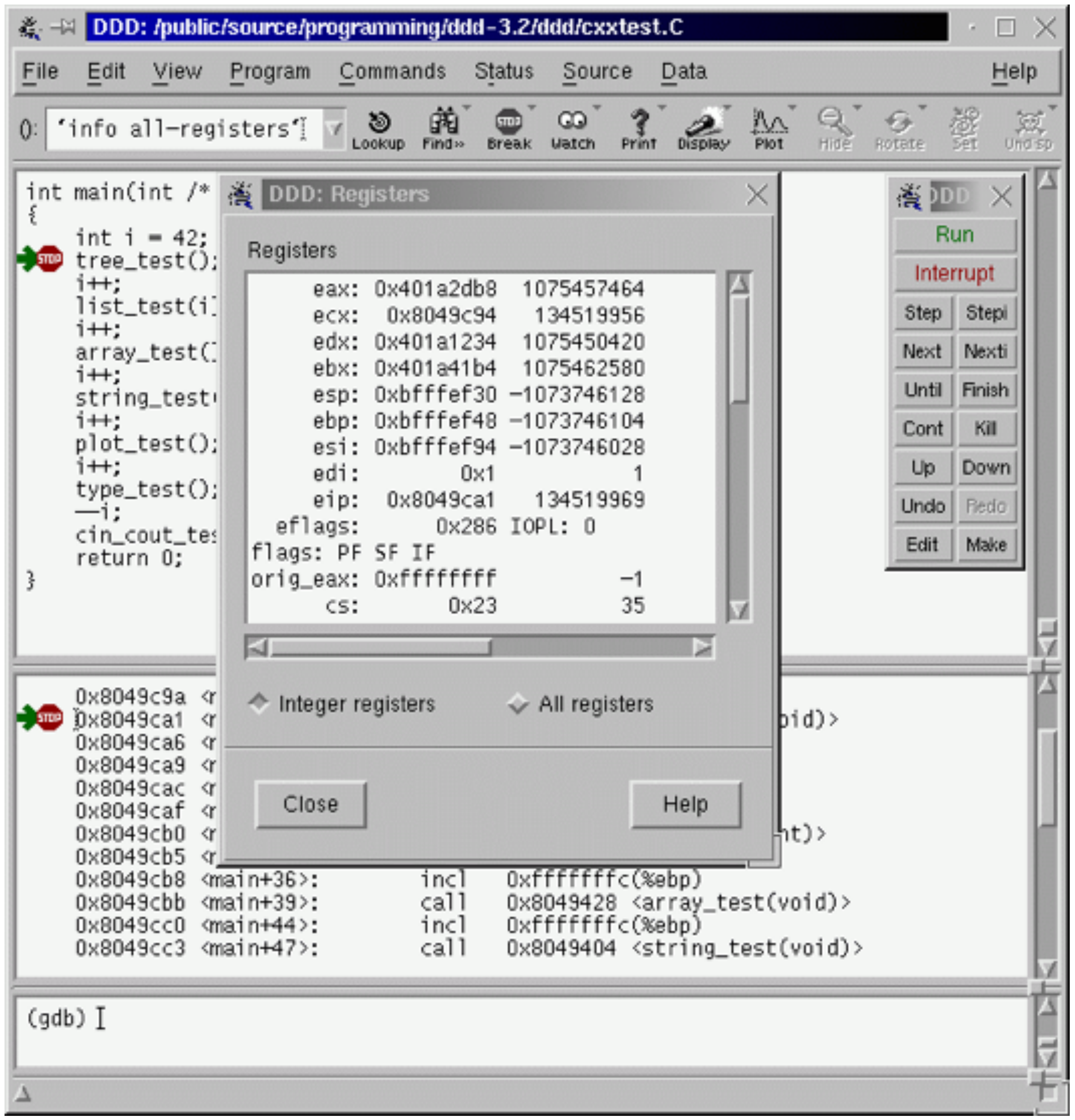
Input: faulty program and 1 failed test case



Dynamic Slicing Example

		Test Cases					
		3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
	`mid() {`						
	`int x,y,z,m;`						
1:	`read("Enter 3 numbers:",x,y,z);`	•	•	•	•	•	•
2:	`m = z;`	•	•	•	•	•	•
3:	`if (y<z)`	•	•	•	•	•	•
4:	`if (x<y)`	•	•			•	•
5:	`m = y;`		•				
6:	`else if (x<z)`	•				•	•
7:	`m = y; // bug`	•					•
8:	`else`			•	•		
9:	`if (x>y)`			•	•		
10:	`m = y;`			•			
11:	`else if (x>z)`				•		
12:	`m = x;`						
13:	`print("Middle number is:", m);`	•	•	•	•	•	•
	`}`						
	Pass/Fail	P	P	P	P	P	F

Korel and Laski's dynamic slicing
Agrawal
Input: source code and failed test case



```
int main(int /*
{
    int i = 42;
    tree_test();
    i++;
    list_test(i]
    i++;
    array_test(
    i++;
    string_test
    i++;
    plot_test();
    i++;
    type_test();
    --i;
    cin_cout_tes
    return 0;
}
```

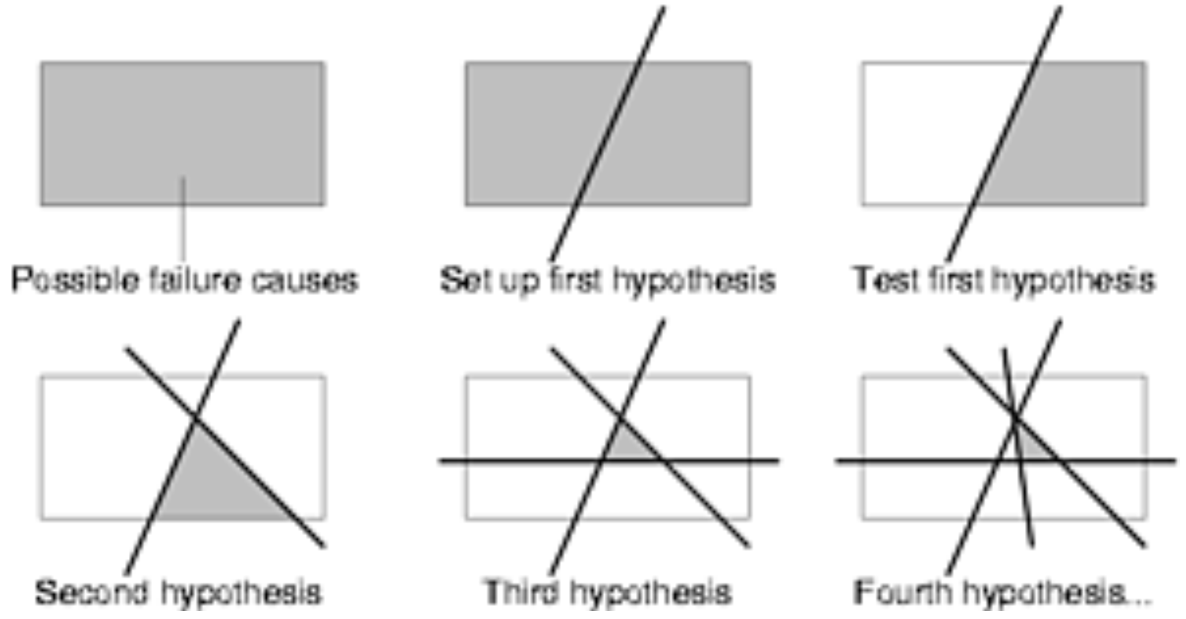
```
      eax: 0x401a2db8  1075457464
      ecx:  0x8049c94   134519956
      edx: 0x401a1234  1075450420
      ebx: 0x401a41b4  1075462580
      esp: 0xbfffef30 -1073746128
      ebp: 0xbfffef48 -1073746104
      esi: 0xbfffef94 -1073746028
      edi:        0x1           1
      eip:  0x8049ca1   134519969
   eflags:      0x286 IOPL: 0
flags: PF SF IF
orig_eax: 0xffffffff          -1
       cs:       0x23          35
```

```
0x8049c9a <r
0x8049ca1 <r                                      oid)>
0x8049ca6 <r
0x8049ca9 <r
0x8049cac <r
0x8049caf <r
0x8049cb0 <r                                      nt)>
0x8049cb5 <r
0x8049cb8 <main+36>:        incl   0xfffffffc(%ebp)
0x8049cbb <main+39>:        call   0x8049428 <array_test(void)>
0x8049cc0 <main+44>:        incl   0xfffffffc(%ebp)
0x8049cc3 <main+47>:        call   0x8049404 <string_test(void)>
```

```
(gdb)
```

DDD

Input: faulty program and failed test case



Delta Debugging

Input: faulty program, 1 failed and 1 passed test case

The Berkeley/Stanford Recovery-Oriented Computing (ROC) Project

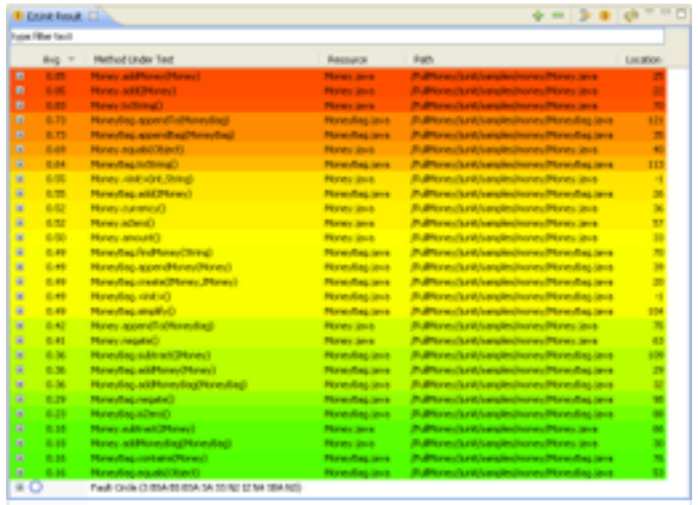
TARANTULA

Fault Localization via Visualization

Statistical Debugging

Input: faulty program, test suite

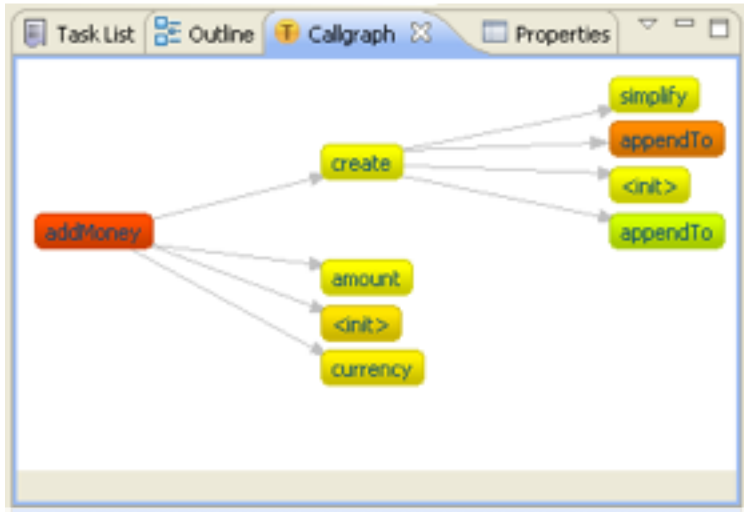
EzUnit

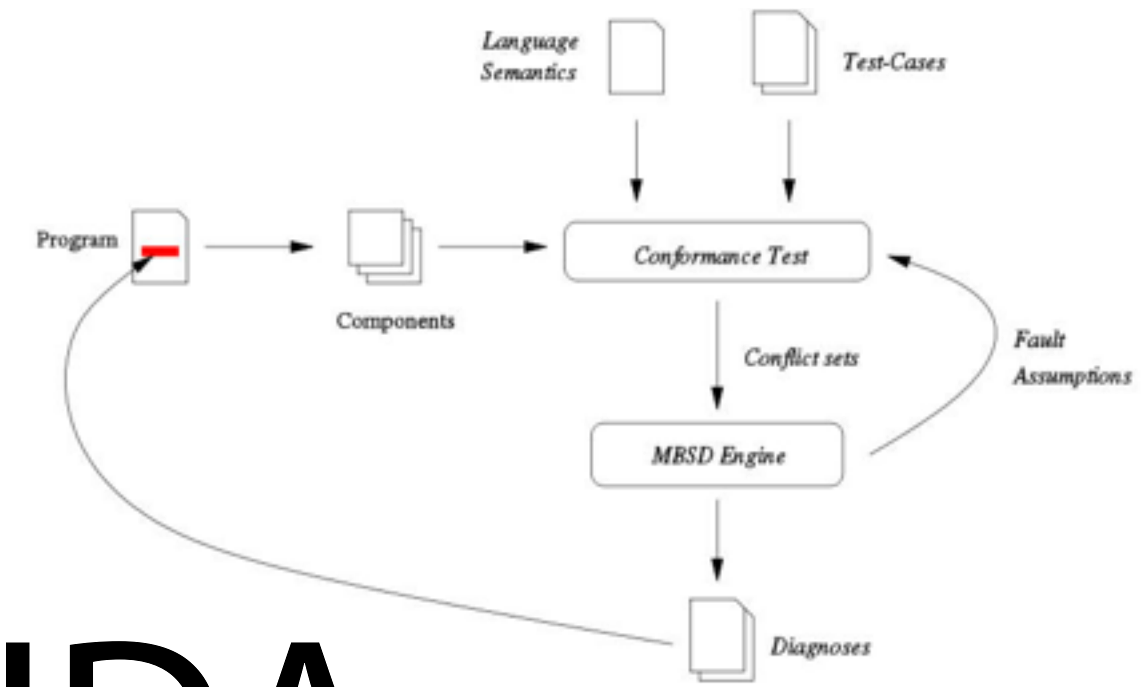


EzUnit Result

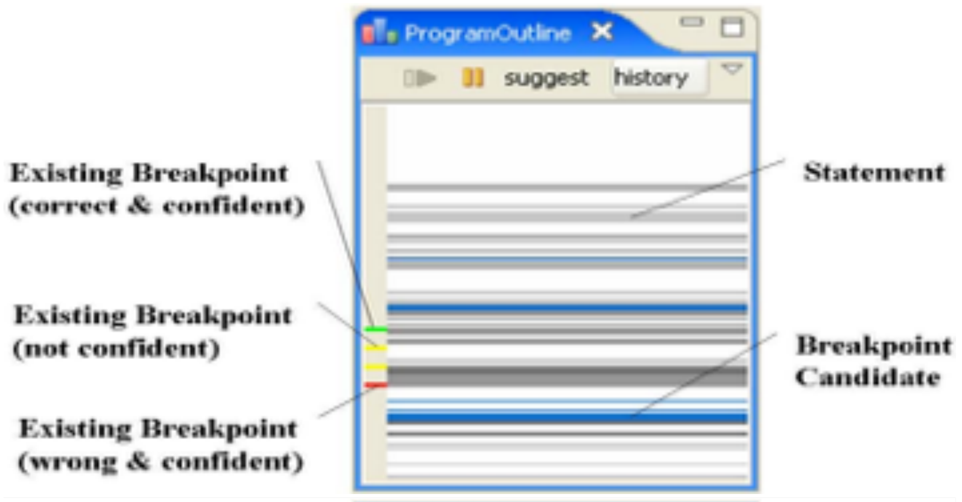
type filter text

	Avg	Method Under Test	Resource	Path	Location
⊞	0.85	Money.addMoney(Money)	Money.java	/FullMoney/junit/samples/money/Money.java	25
⊞	0.85	Money.add(IMoney)	Money.java	/FullMoney/junit/samples/money/Money.java	22
⊞	0.83	Money.toString()	Money.java	/FullMoney/junit/samples/money/Money.java	70
⊞	0.73	MoneyBag.appendTo(MoneyBag)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	121
⊞	0.73	MoneyBag.appendBag(MoneyBag)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	35
⊞	0.69	Money.equals(Object)	Money.java	/FullMoney/junit/samples/money/Money.java	40
⊞	0.64	MoneyBag.toString()	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	113
⊞	0.55	Money.<init>(int, String)	Money.java	/FullMoney/junit/samples/money/Money.java	-1
⊞	0.55	MoneyBag.add(IMoney)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	28
⊞	0.52	Money.currency()	Money.java	/FullMoney/junit/samples/money/Money.java	36
⊞	0.52	Money.isZero()	Money.java	/FullMoney/junit/samples/money/Money.java	57
⊞	0.50	Money.amount()	Money.java	/FullMoney/junit/samples/money/Money.java	33
⊞	0.49	MoneyBag.findMoney(String)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	70
⊞	0.49	MoneyBag.appendMoney(Money)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	39
⊞	0.49	MoneyBag.create(IMoney, IMoney)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	20
⊞	0.49	MoneyBag.<init>()	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	-1
⊞	0.49	MoneyBag.simplify()	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	104
⊞	0.42	Money.appendTo(MoneyBag)	Money.java	/FullMoney/junit/samples/money/Money.java	75
⊞	0.41	Money.negate()	Money.java	/FullMoney/junit/samples/money/Money.java	63
⊞	0.36	MoneyBag.subtract(IMoney)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	109
⊞	0.36	MoneyBag.addMoney(Money)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	29
⊞	0.36	MoneyBag.addMoneyBag(MoneyBag)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	32
⊞	0.29	MoneyBag.negate()	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	98
⊞	0.23	MoneyBag.isZero()	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	88
⊞	0.18	Money.subtract(IMoney)	Money.java	/FullMoney/junit/samples/money/Money.java	66
⊞	0.18	Money.addMoneyBag(MoneyBag)	Money.java	/FullMoney/junit/samples/money/Money.java	30
⊞	0.16	MoneyBag.contains(Money)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	76
⊞	0.16	MoneyBag.equals(Object)	MoneyBag.java	/FullMoney/junit/samples/money/MoneyBag.java	23
⊞ ○		Fault Circle (3 B5A B5 B5A 5A 55 N2 I2 N4 5BA N3)			




Language Semantics
Test-Cases
Program
Components
Conformance Test
Conflict sets
Fault Assumptions
MBSD Engine
Diagnoses

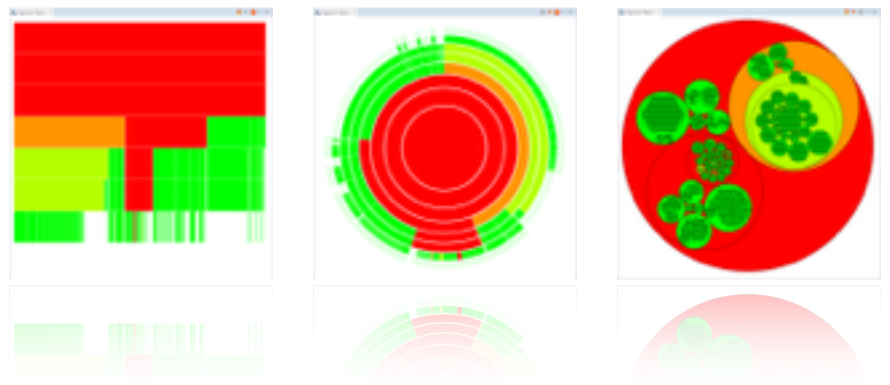
VIDA


ProgramOutline
suggest
history
Existing Breakpoint (correct & confident)
Existing Breakpoint (not confident)
Existing Breakpoint (wrong & confident)
Statement
Breakpoint Candidate

ZOLTAR
SPEAKS

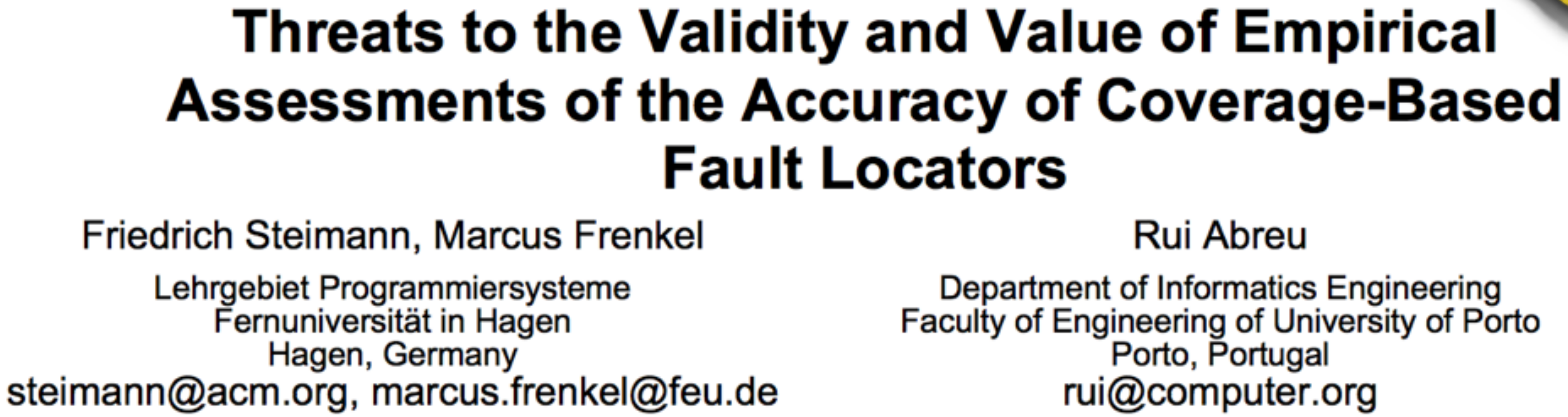
1840
1947
1962
1981
1986
1988
1993
1996
2002
2007
2009
2015

GZOLTAR



1840
1947
1962
1981
1986
1988
1993
1996
2002
2007
2009
2011/12
2015

Also a survey paper is under review at TSE. More than 300 works cited.



Threats to the Validity and Value of Empirical Assessments of the Accuracy of Coverage-Based Fault Locators

Friedrich Steimann, Marcus Frenkel
Lehrgebiet Programmiersysteme
Fernuniversität in Hagen
Hagen, Germany
steimann@acm.org, marcus.frenkel@feu.de

Rui Abreu
Department of Informatics Engineering
Faculty of Engineering of University of Porto
Porto, Portugal
rui@computer.org

Focus of this talk

- Techniques that take into account **spectra**
    - aka abstraction of program traces
    - Spectrum-based Fault Localization (**SFL**)
        - **Statistical vs. reasoning**
- **Lightweight, scalable**

SFL: Principle (1)

Test suite

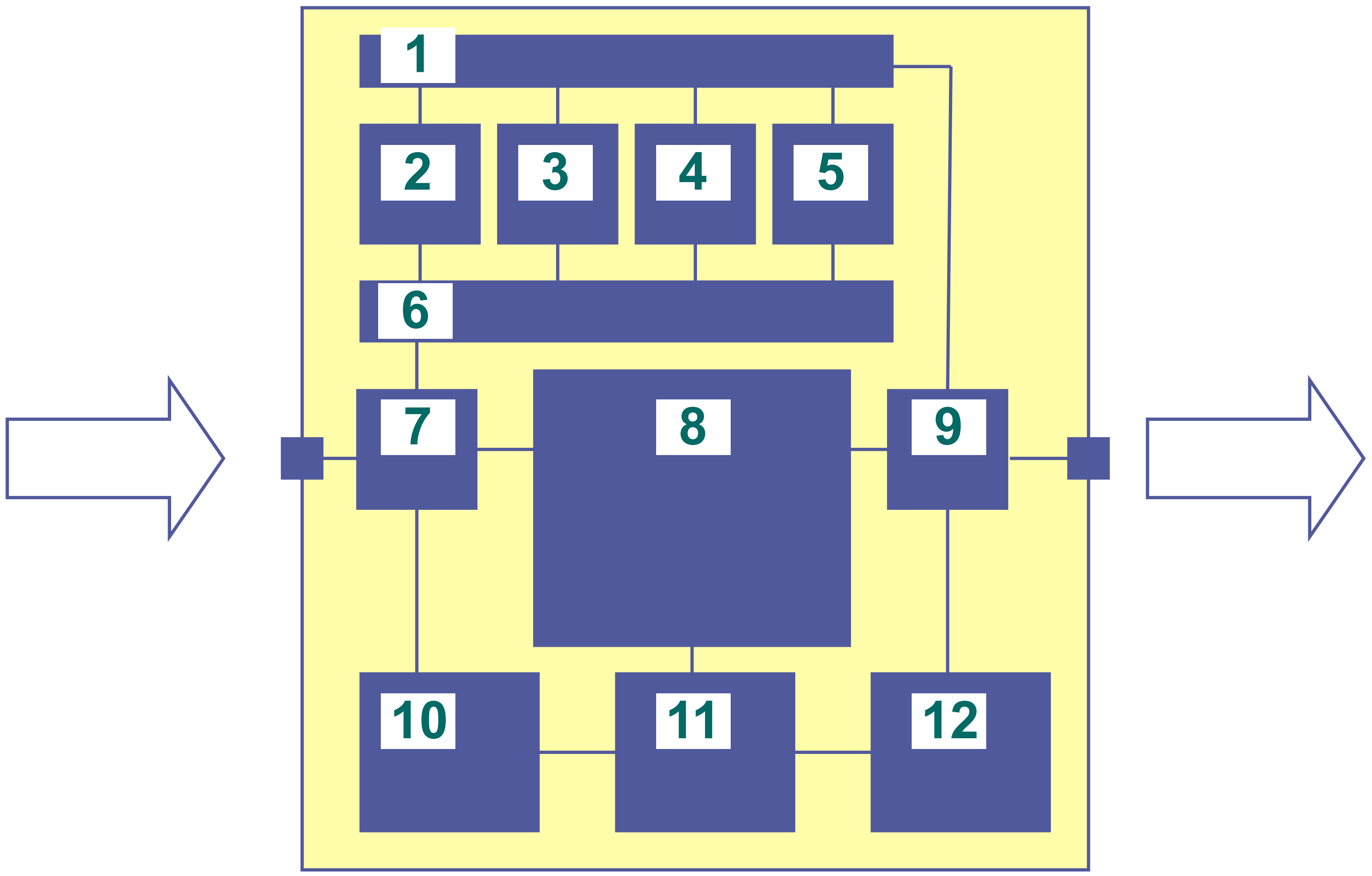
t1

t2

t3

t4

t5



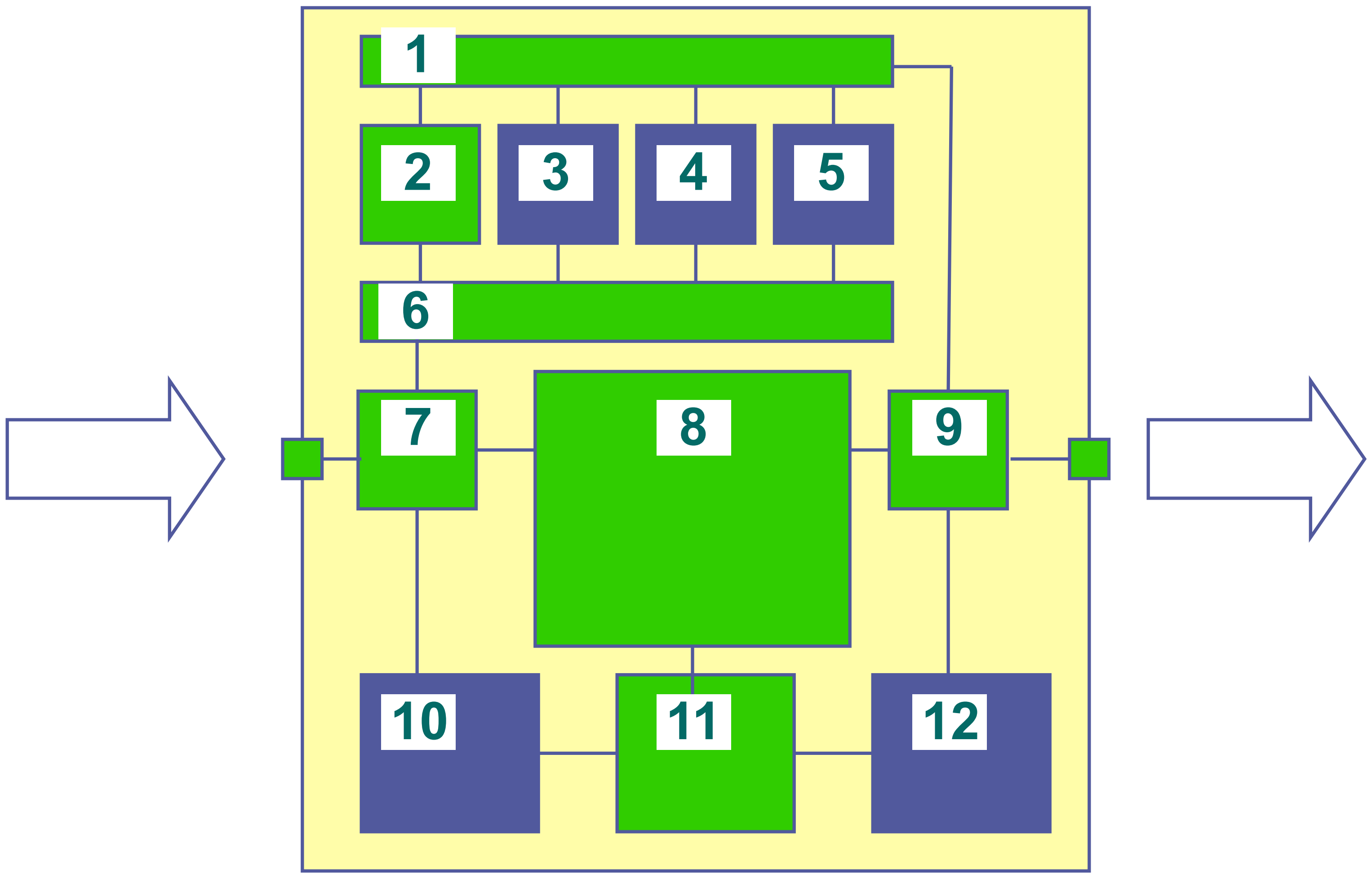
1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

🟦	**Not touched**
🟩	**Touched, pass**
🟥	**Touched, fail**

SFL: Principle (2)

Test suite

t2
t3
t4
t5



Status

t1 ✓

1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	0	0	1	1	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0

🟦	**Not touched**
🟩	**Touched, pass**
🟥	**Touched, fail**

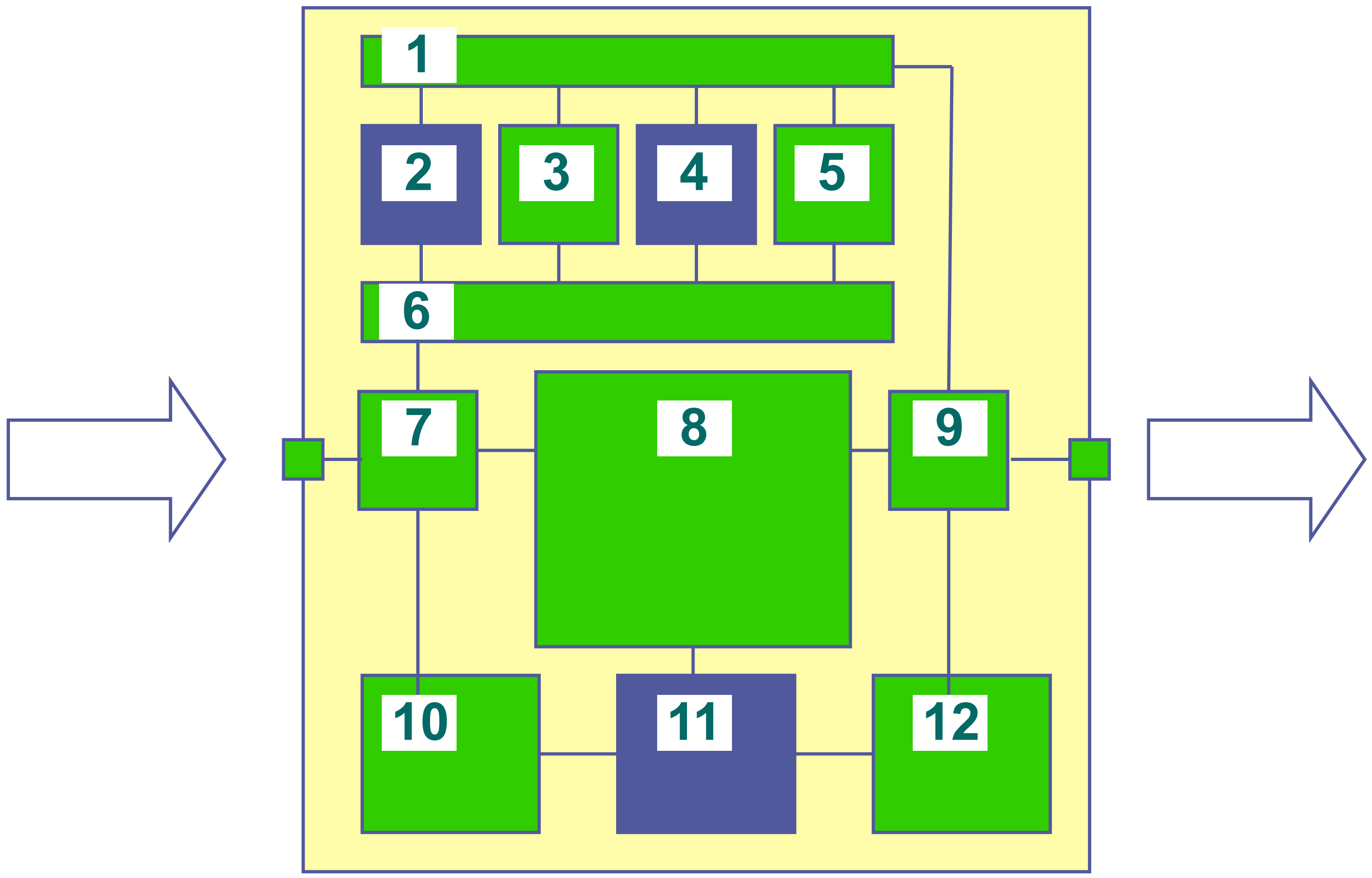
SFL: Principle (3)

Test suite

t3

t4

t5



Status

t1 ✓

t2 ✓

1	2	3	4	5	6	7	8	9	10	11	12
2	1	1	0	1	2	2	2	2	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0

🟦	Not touched
🟩	Touched, pass
🟥	Touched, fail

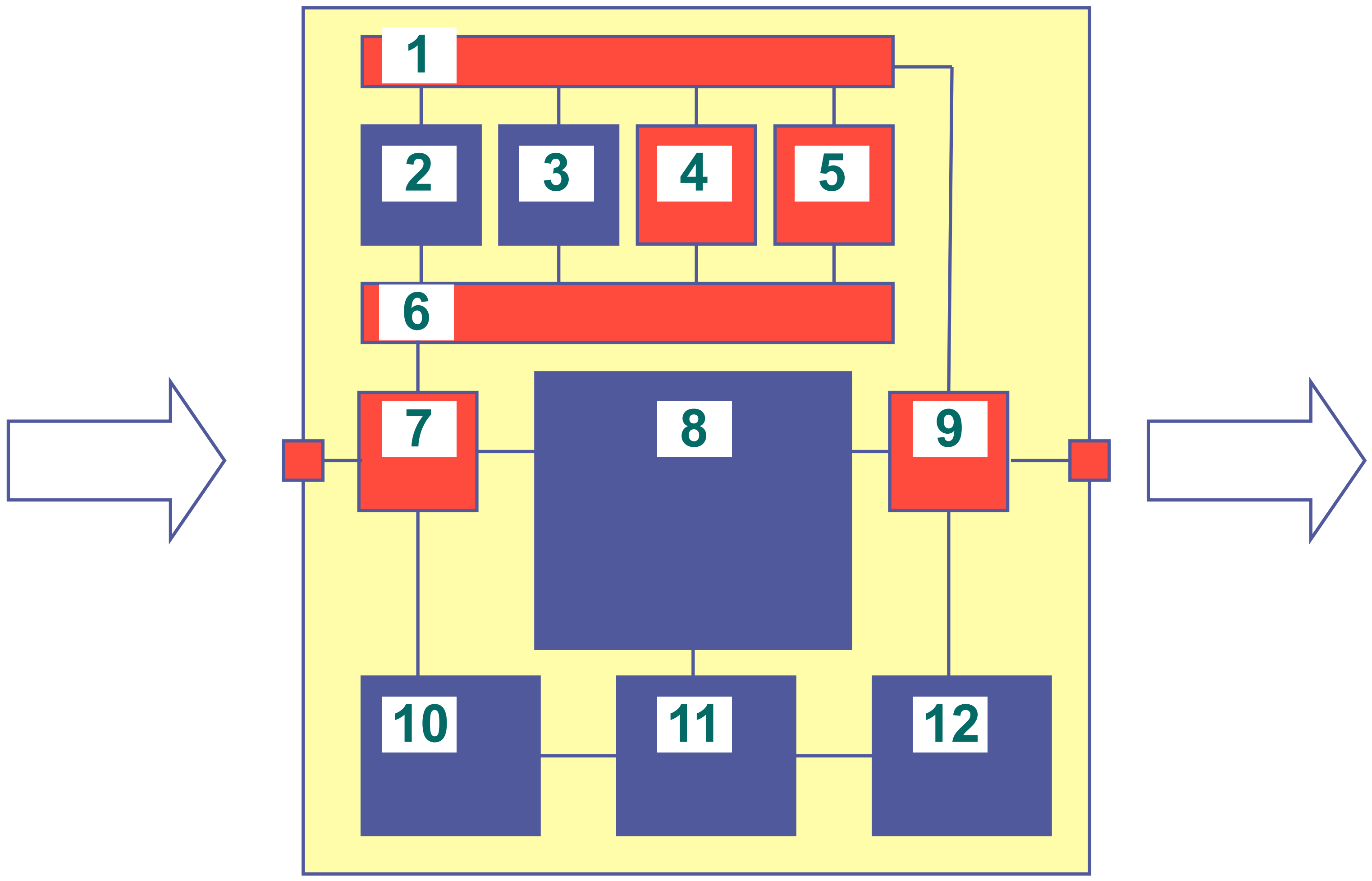
Integrates well with testing

SFL: Principle (4)

Test suite

t4

t5



Status

t1	✓
t2	✓
t3	✘

1	2	3	4	5	6	7	8	9	10	11	12
2	1	1	0	1	2	2	2	2	1	1	1
1	0	0	1	1	1	1	0	1	0	0	0

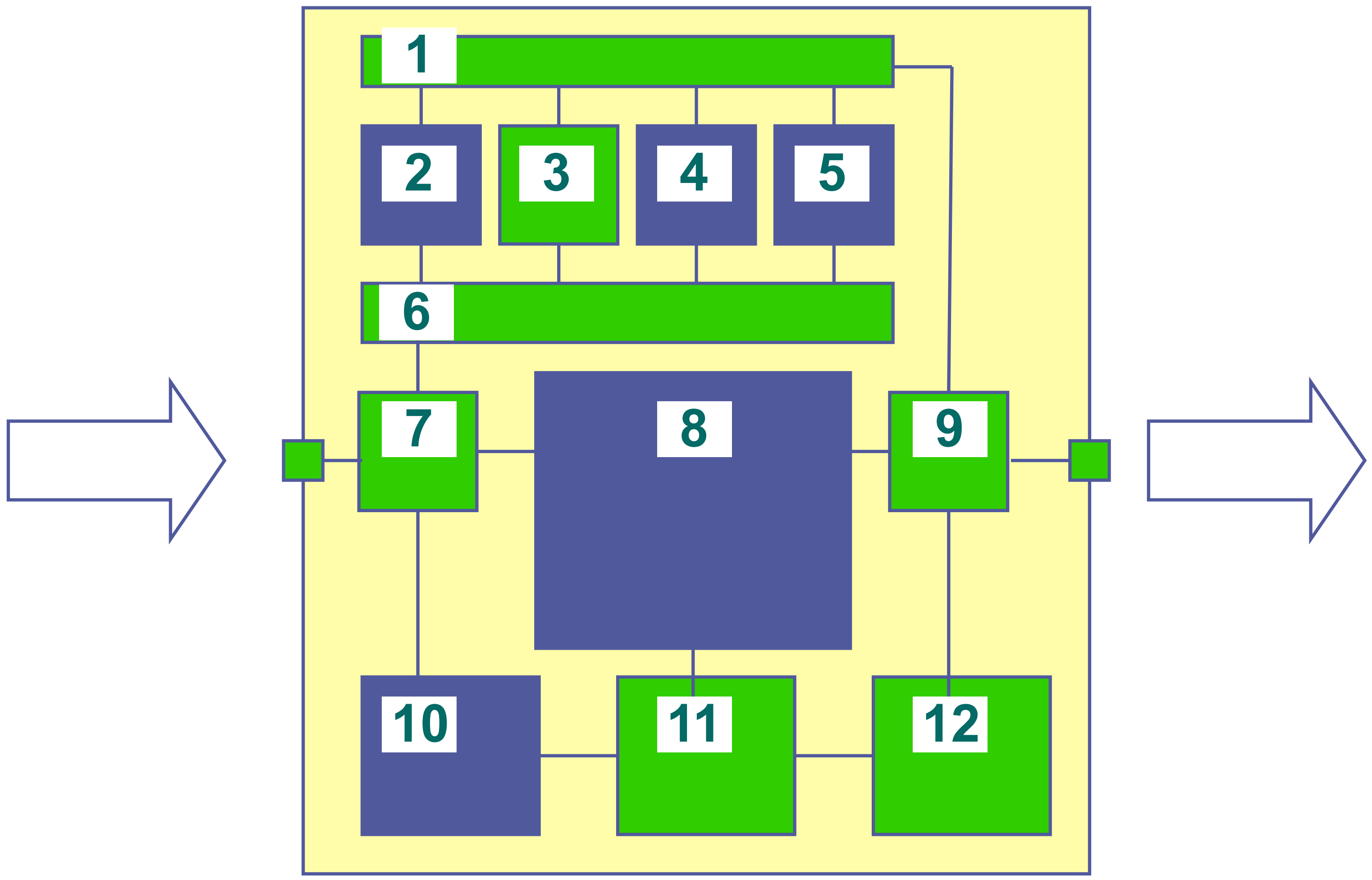
🟦	Not touched
🟩	Touched, pass
🟥	Touched, fail

Integrates well with testing

SFL: Principle (5)

Test suite

t5



Status

t1	✓
t2	✓
t3	✗
t4	✓

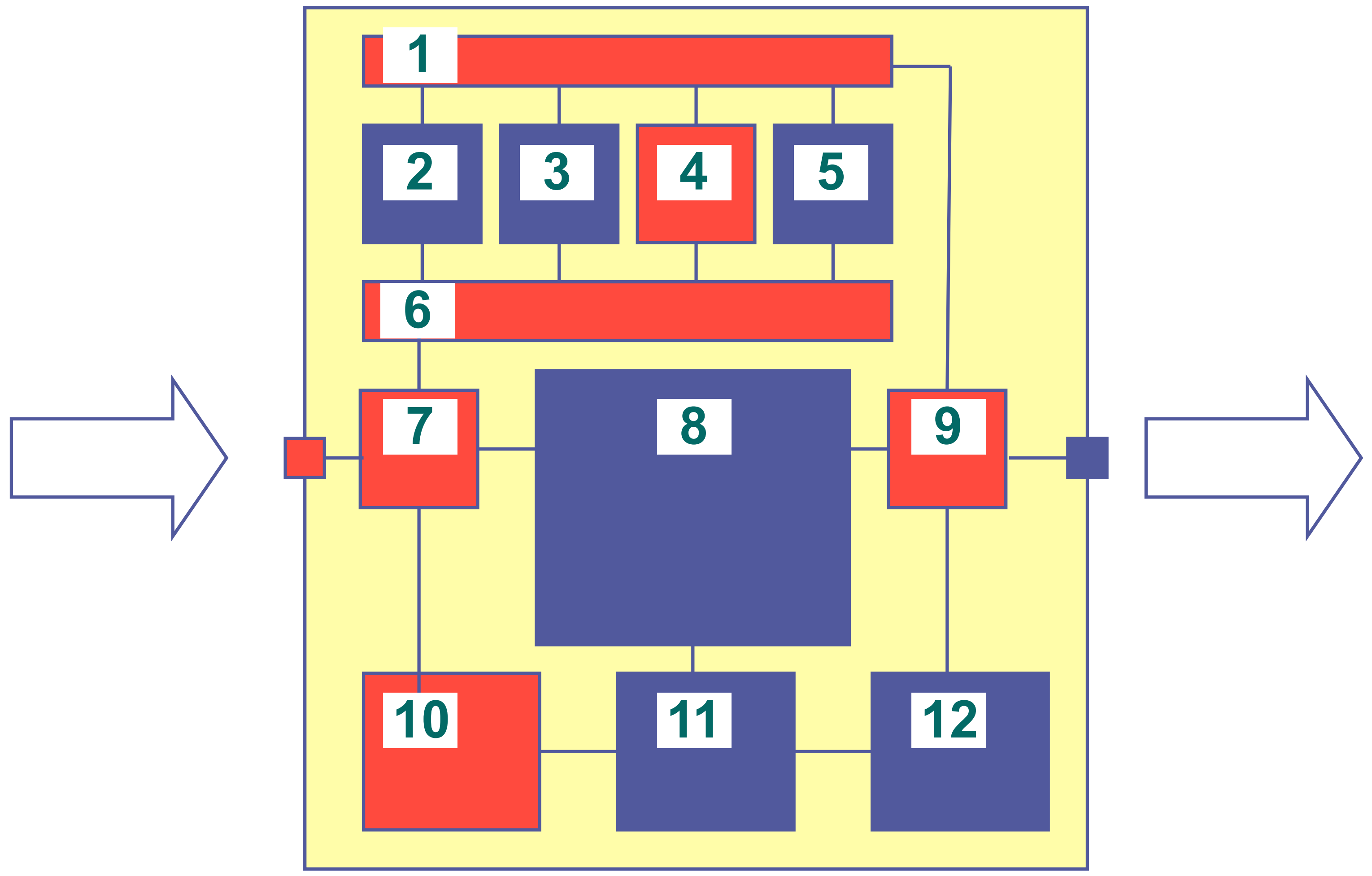
1	2	3	4	5	6	7	8	9	10	11	12
3	1	1	0	1	3	3	2	3	1	3	3
1	0	0	1	1	1	1	0	1	0	0	0

🟦	Not touched
🟩	Touched, pass
🟥	Touched, fail

Integrates well with testing

SFL: Principle (6)

Test suite



Status

t1	✓
t2	✓
t3	✗
t4	✓
t5	✗

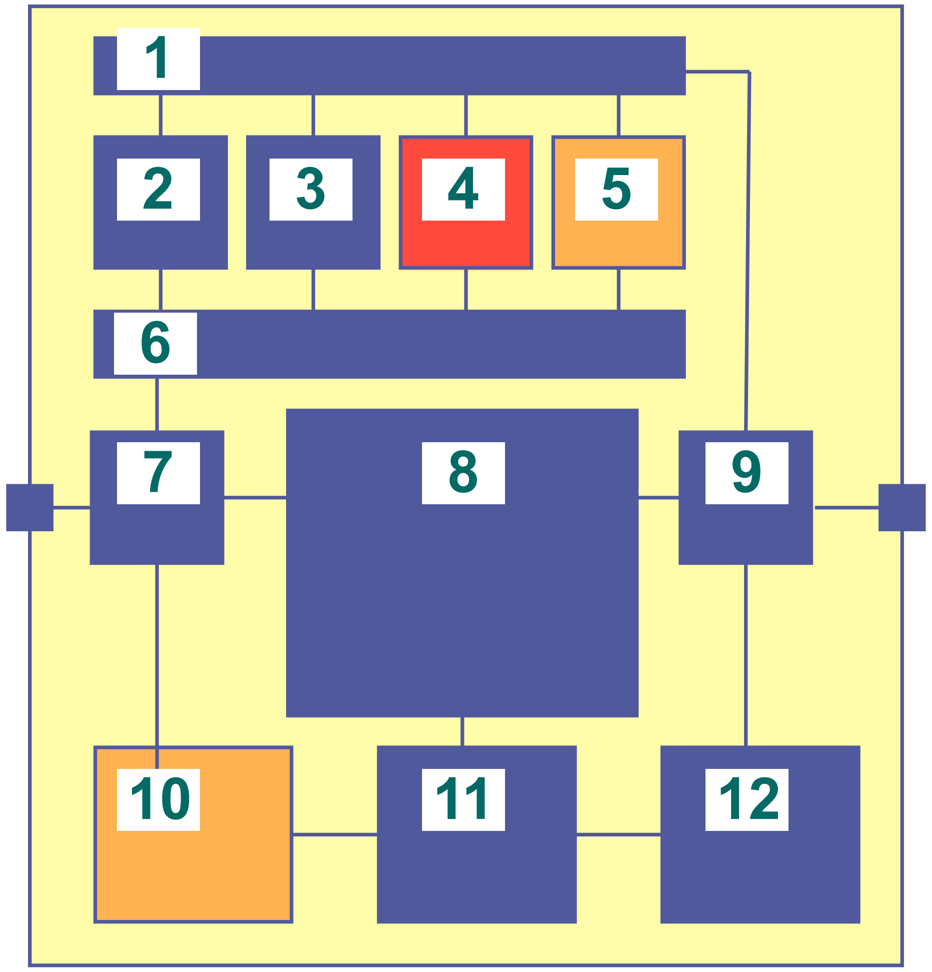
1	2	3	4	5	6	7	8	9	10	11	12
3	1	1	0	1	3	3	2	3	1	3	3
2	0	0	2	1	2	2	0	2	1	0	0

🟦	Not touched
🟩	Touched, pass
🟥	Touched, fail

Integrates well with testing

SFL: Principle (7)

Components are **ranked** according to the likelihood of causing detected errors



Status

t1	✓
t2	✓
t3	✗
t4	✓
t5	✗

1	2	3	4	5	6	7	8	9	10	11	12
3	1	1	0	1	3	3	2	3	1	3	3
2	0	0	2	1	2	2	0	2	1	0	0

🟦	Not touched
🟩	
🟥	Touched, fail

Program

Test Suite

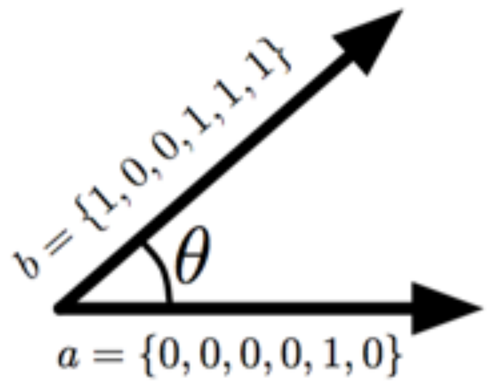
class Triangle {… static int type(int a, int b, int c) {	t_1	t_2	t_3	t_4	t_5	t_6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;	●						0.10001
else if ((a*a) == ((b*b) + (c*c)))		●	●	●			0.09999
type = RIGHT;			●				0.10001
else if ((a == b) || (b == a)) /* FAULT */		●		●			**0.10000**
type = ISOSCELES;		●					0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }					●	●	0.10000

Fault

Spectra

Suspiciousness score

- Each component (row) is **ranked** according to their **similarity** to the **error vector**
  - Many similarity coefficients exist.



$$\mathit{Ochiai}(a,b) = \cos(\theta)$$

- **Ochiai** similarity is equivalent to the cosine of the angle between two vectors in a n-dimensional space

Abreu, R., Zoeteweij, P., Golsteijn, R., & Van Gemund, A. J. (2009). A practical evaluation of spectrum-based fault localization. Journal of Systems and Software, 82(11), 1780-1792.
Lucia, L., Lo, D., Jiang, L., Thung, F., & Budi, A. (2014). Extended comprehensive study of association measures for fault localization. Journal of Software: Evolution and Process, 26(2).

Diagnostic Performance

Rank Position	Suspicious Statement	Line number	Suspiciousness
1º	`type = EQUILAT`	3	0.10001
2º	`type = RIGH`	5	0.10001
3º	`type = ISC   ES;`	7	0.10001
4º	`else if (   b)   ==   /* FAULT *`	6	0.10000
5º	`double s =   )/2.0;`	9	0.10000
6º	`return Math.sqr   b)*(s-c));`	10	0.10000
7º	`else if ( (a*a) == ((b*b) + (c*c)) )`	4	0.09999
8º	`int type = SCALENE;`	1	0.09998
9º	`if ( (a == b) && (b == c) )`	2	0.09998
10º	`return type; }`	8	0.09998

C_d = 4

Can we do better?

- Statistics-based SFL does not reason in terms of multiple faults

c_1	c_2	c_3	P/F
1	0	0	1 (F)
0	1	0	1 (F)
1	0	1	1 (F)
0	1	1	1 (F)
1	1	0	0 (P)

Diagnostic report = $< c_3, c_1, c_2 >$

Reasoning-based Approach

- Barinel is a reasoning-based approach
    - Integrates the best of model-based diagnosis with spectra

c_1	c_2	c_3	P/F
1	**0**	**0**	**1 (F)**
0	1	0	1 (F)
1	0	1	1 (F)
0	1	1	1 (F)
1	1	0	0 (P)

c_1 must be faulty
c_2 cannot be single fault
c_3 cannot be single fault
c_2, c_3 cannot be double fault

Reasoning-based Approach

- Barinel is a reasoning-based approach
  - Integrates the best of model-based diagnosis with spectra

c_1	c_2	c_3	P/F
1	0	0	1 (F)
0	**1**	**0**	**1 (F)**
1	0	1	1 (F)
0	1	1	1 (F)
1	1	0	0 (P)

c_2 must be faulty
c_1 cannot be single fault
c_1 cannot be single fault
c_1, c_3 cannot be double fault

Reasoning-based Approach

- Barinel is a spectrum-based reasoning approach
  - Integrates the best of model-based diagnosis with spectra

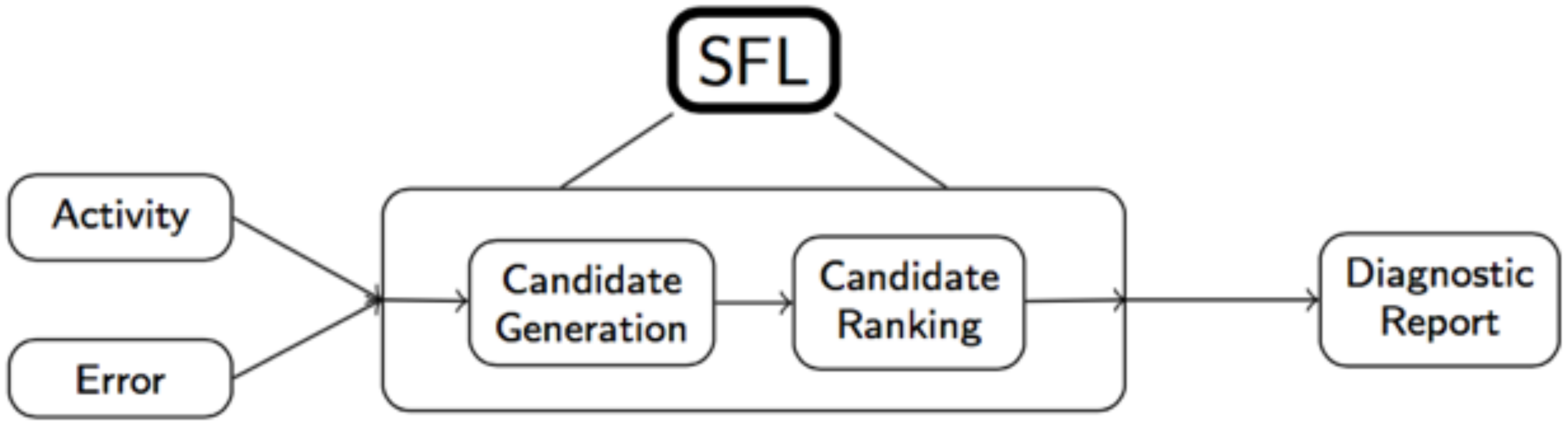
c_1			
1			
0			
1			
0	1	1	1 (F)
1	1	0	0 (P)

Summary:

c1, c2 faulty, but not single-fault
c1, c2 can be double-fault
c1,c3 nor c2,c3 can be double-fault
so {c1,c2} is the only diagnosis possible
(subsuming the triple fault {c1,c2,c3})

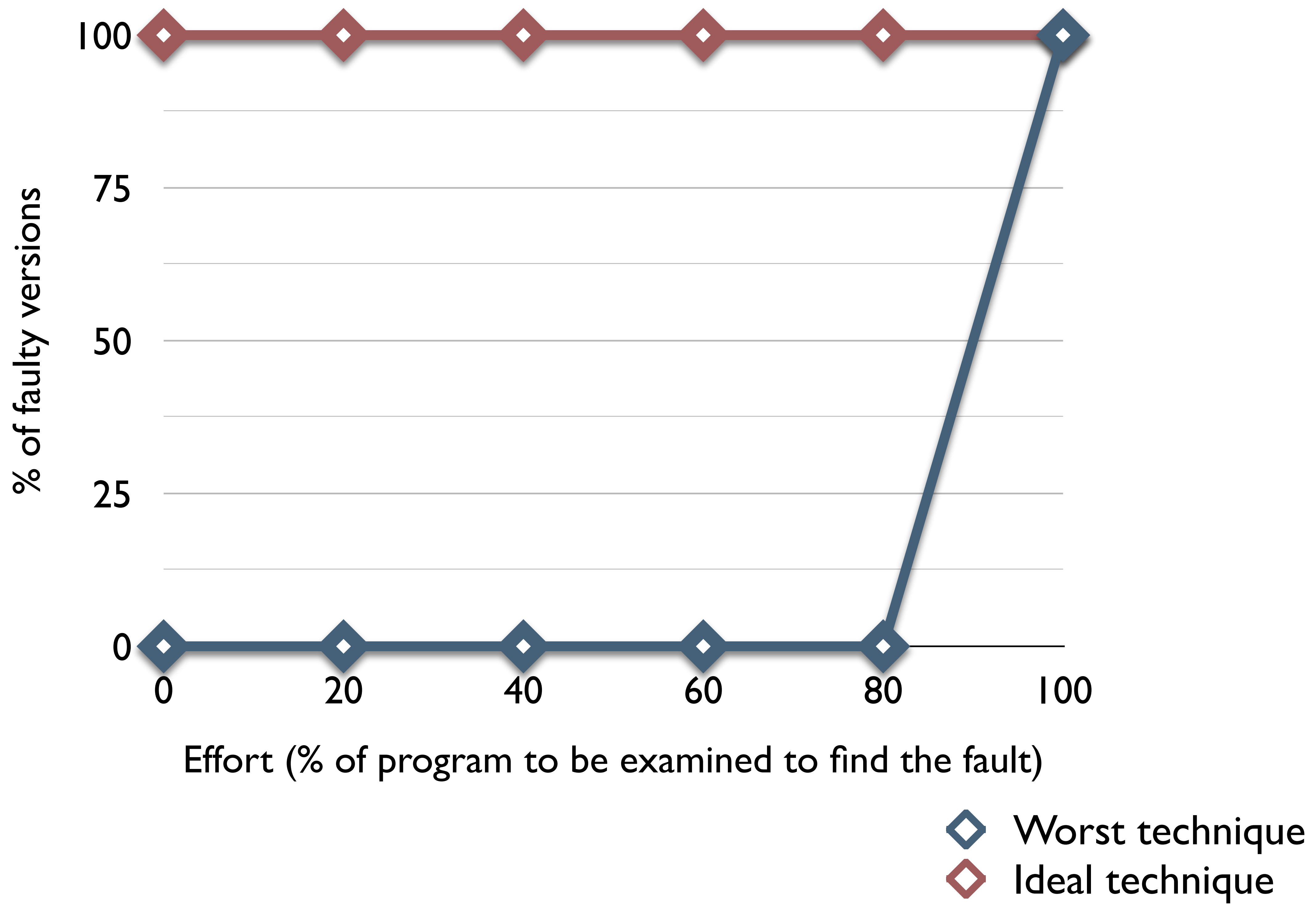
Spectrum-based reasoning

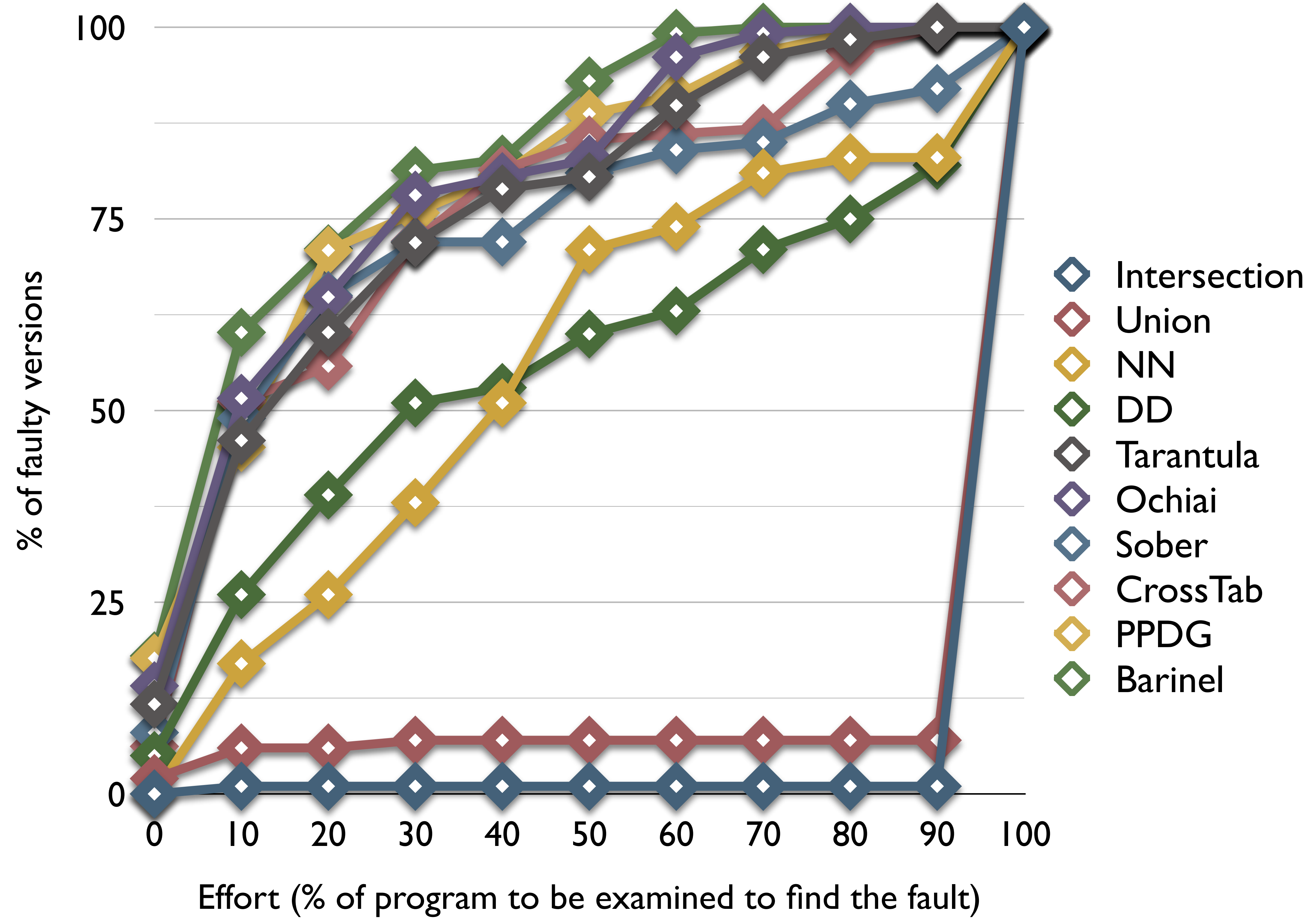
1. Generate sets of components that *explain* observed erroneous behavior
    - Equivalent to compute minimal hitting set (Staccato/MHS2**)
    - Given failed executions
2. Rank candidates according to their probability of being the true fault explanation ➤ Baye's rule
    - Given both passed and failed executions



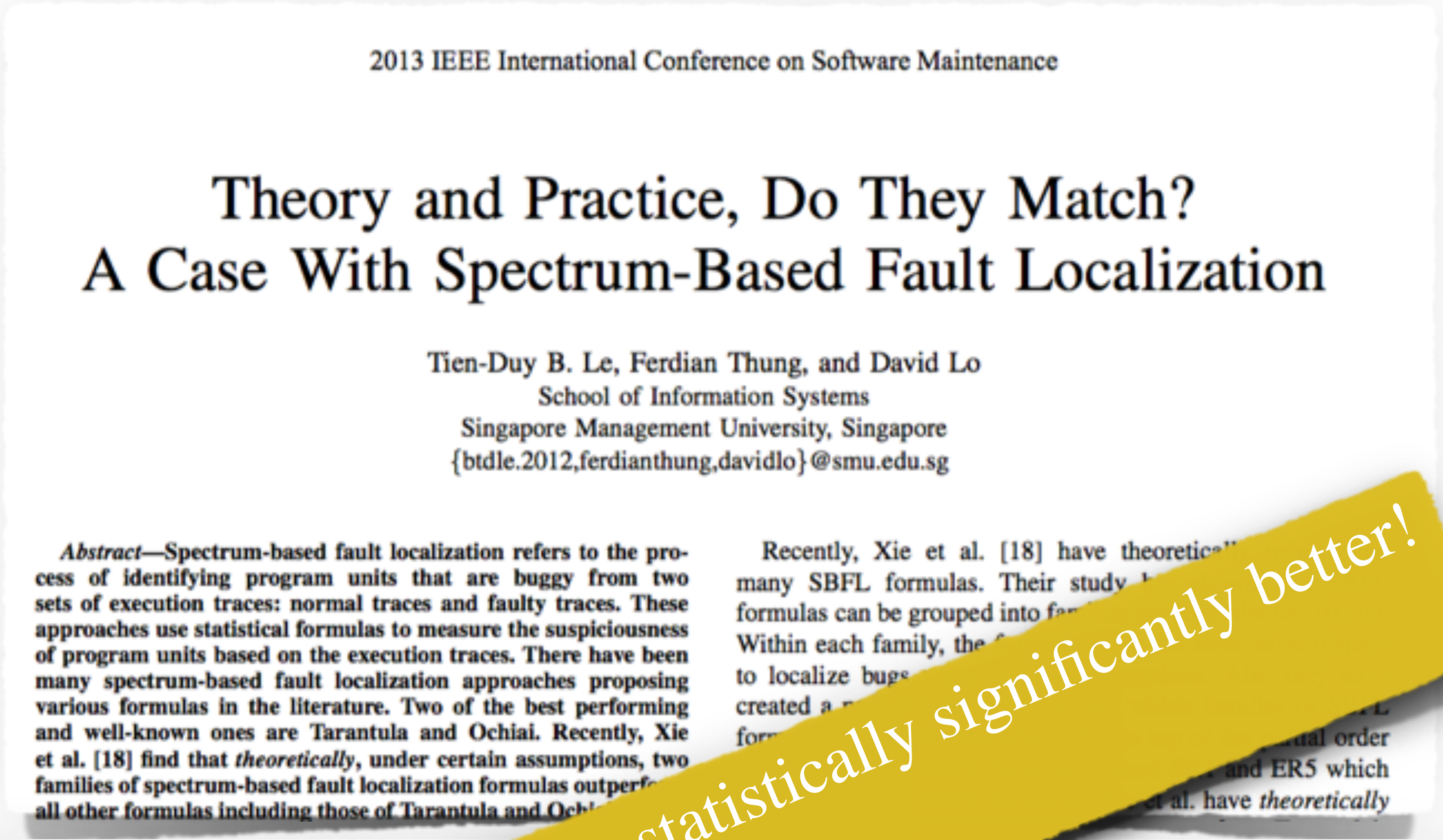
R. Abreu, P. Zoeteweij, and A. J. van Gemund, "Spectrum-Based Multiple Fault Localization", ASE '09
**https://github.com/npcardoso/MHS2 (citable via https://zenodo.org/record/10037) ➤ contribute to the project; send pull requests; email us!

Diagnostic Performance




100
75
50
25
0
% of faulty versions
0
10
20
30
40
50
60
70
80
90
100
Effort (% of program to be examined to find the fault)
Intersection
Union
NN
DD
Tarantula
Ochiai
Sober
CrossTab
PPDG
Barinel

No similarity coefficient is statistically significantly better!



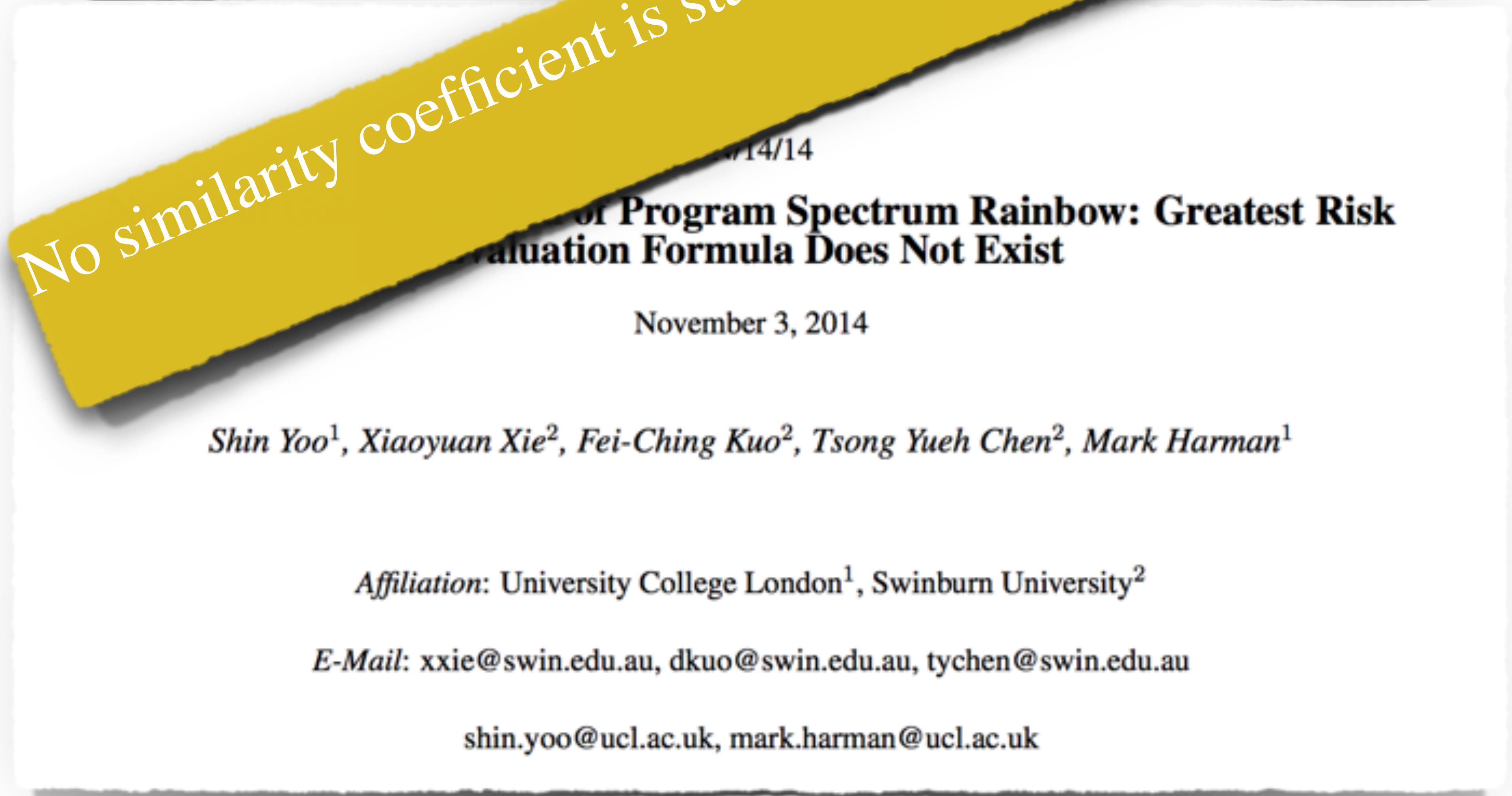
2013 IEEE International Conference on Software Maintenance

Theory and Practice, Do They Match?
A Case With Spectrum-Based Fault Localization

Tien-Duy B. Le, Ferdian Thung, and David Lo
School of Information Systems
Singapore Management University, Singapore
{btdle.2012,ferdianthung,davidlo}@smu.edu.sg

***Abstract*—Spectrum-based fault localization refers to the process of identifying program units that are buggy from two sets of execution traces: normal traces and faulty traces. These approaches use statistical formulas to measure the suspiciousness of program units based on the execution traces. There have been many spectrum-based fault localization approaches proposing various formulas in the literature. Two of the best performing and well-known ones are Tarantula and Ochiai. Recently, Xie et al. [18] find that *theoretically*, under certain assumptions, two families of spectrum-based fault localization formulas outperf all other formulas including those of Tarantula and Och**

Recently, Xie et al. [18] have theoretic many SBFL formulas. Their study formulas can be grouped into fa Within each family, the to localize bugs created a for order and ER5 which et al. have *theoretically*



14/14

of Program Spectrum Rainbow: Greatest Risk
aluation Formula Does Not Exist

November 3, 2014

Shin Yoo[1], *Xiaoyuan Xie*[2], *Fei-Ching Kuo*[2], *Tsong Yueh Chen*[2], *Mark Harman*[1]

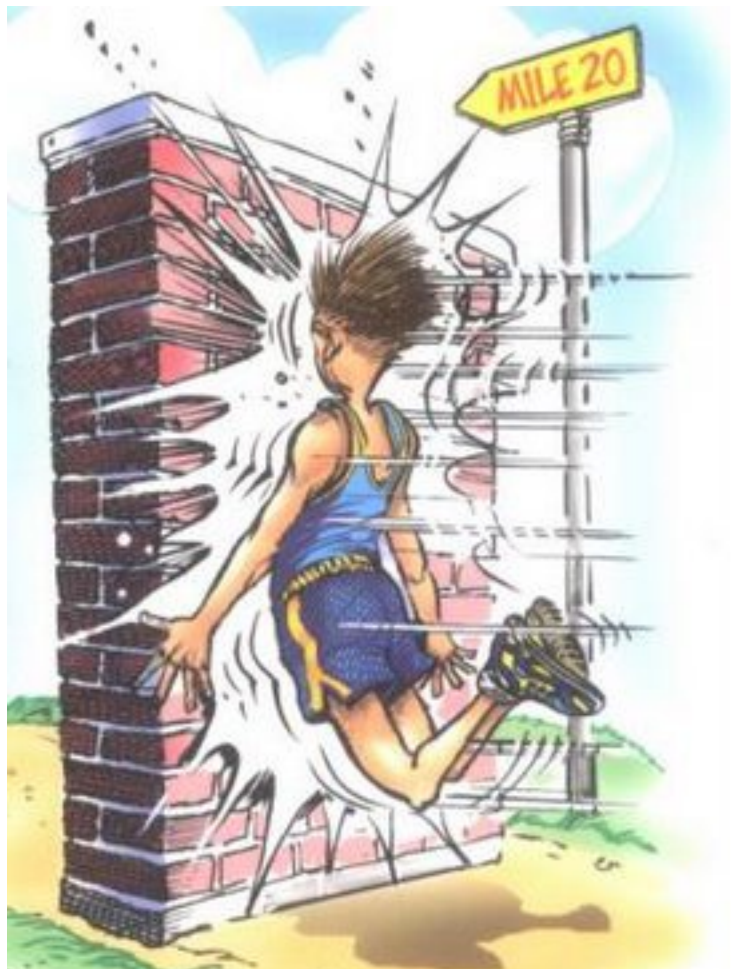
Affiliation: University College London[1], Swinburn University[2]

E-Mail: xxie@swin.edu.au, dkuo@swin.edu.au, tychen@swin.edu.au

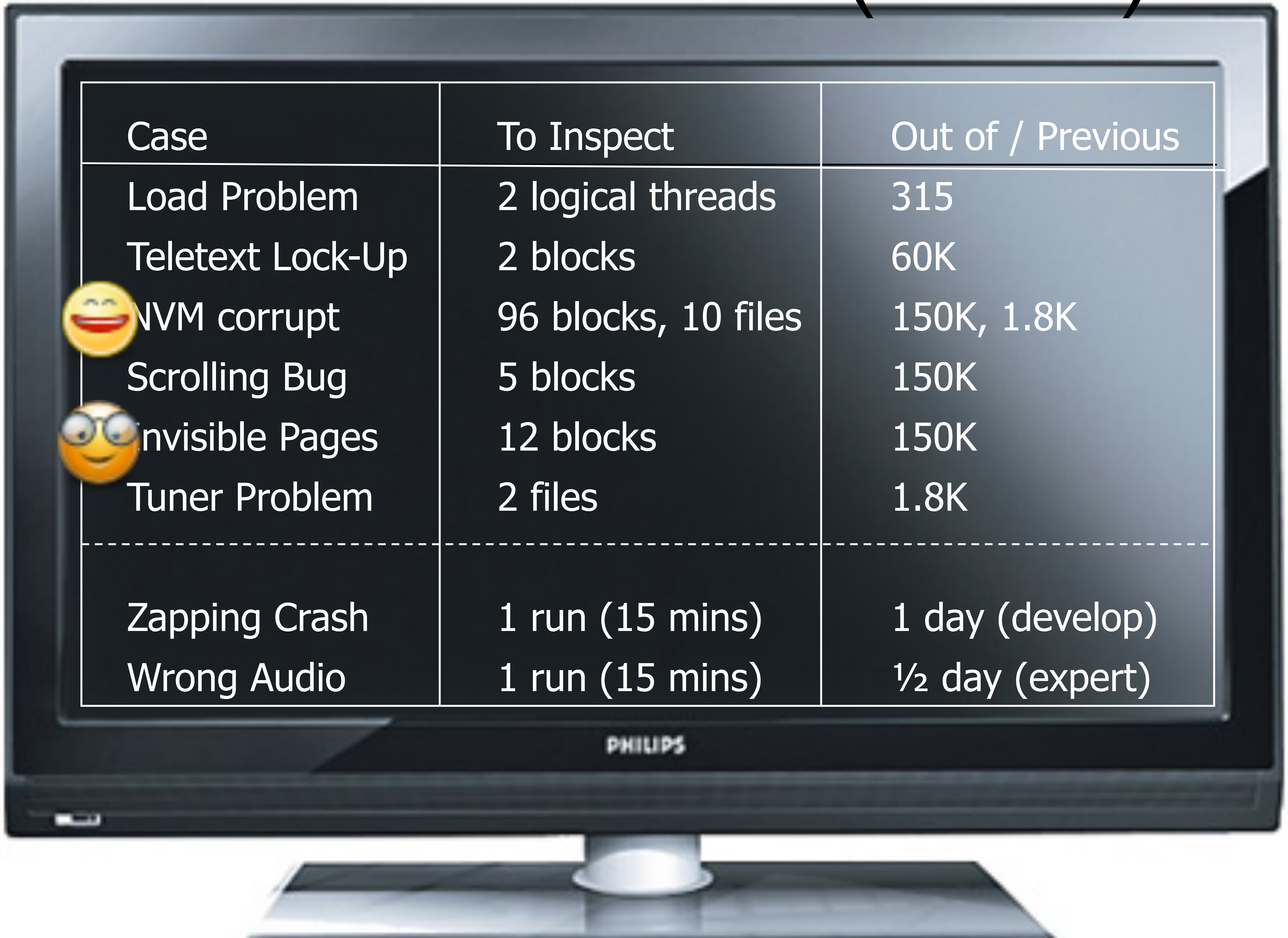
shin.yoo@ucl.ac.uk, mark.harman@ucl.ac.uk

How good are we?

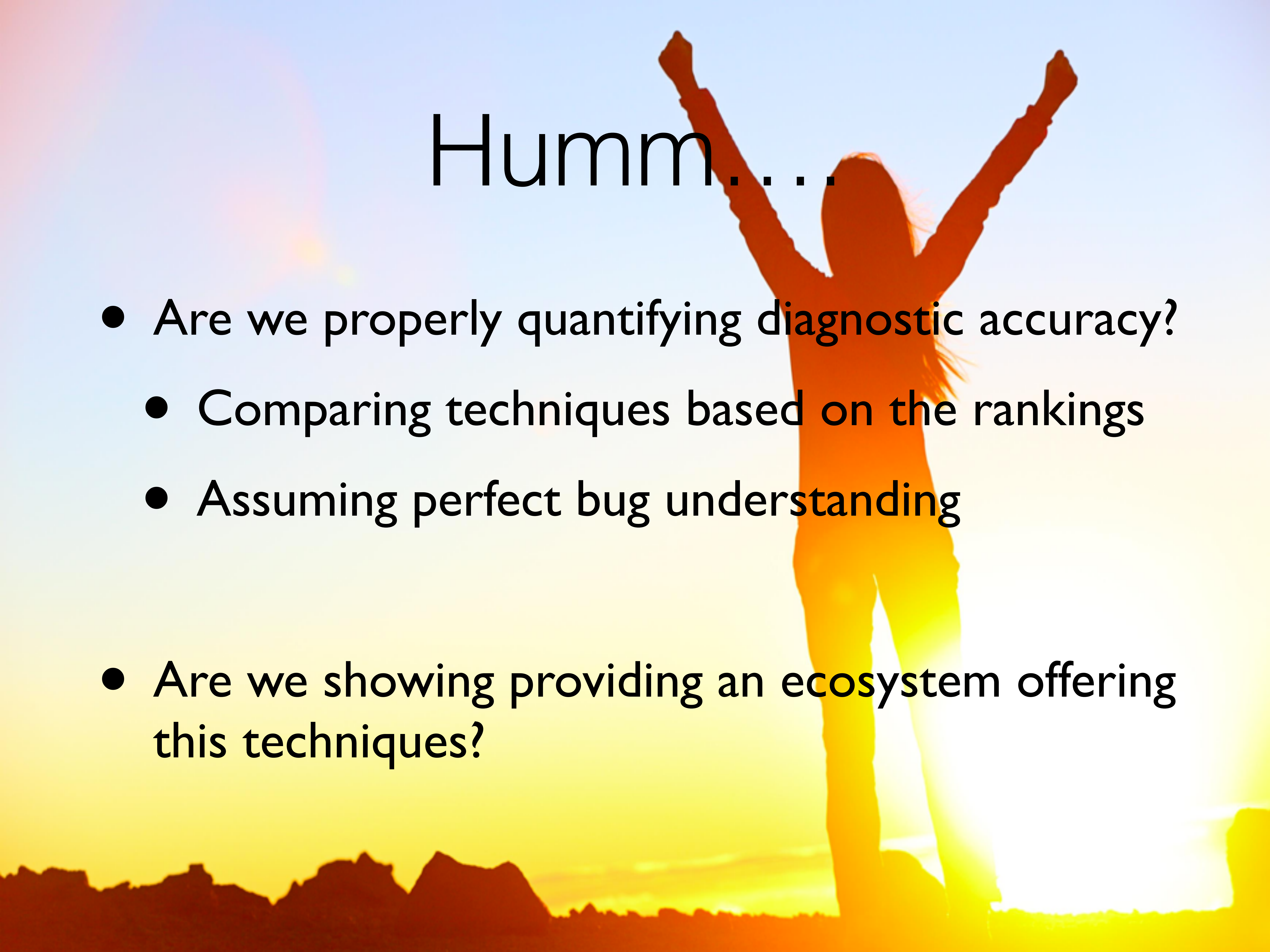
- Best Performing techniques still require to inspect 10% of the code…
  - 100 LOC ➤ 10LOC
  - 10,000 LOC ➤ 1,000LOC
  - 1,000,000 LOC ➤ 10,000LOC



Case Studies (NXP)



Case	To Inspect	Out of / Previous
Load Problem	2 logical threads	315
Teletext Lock-Up	2 blocks	60K
NVM corrupt	96 blocks, 10 files	150K, 1.8K
Scrolling Bug	5 blocks	150K
Invisible Pages	12 blocks	150K
Tuner Problem	2 files	1.8K
Zapping Crash	1 run (15 mins)	1 day (develop)
Wrong Audio	1 run (15 mins)	½ day (expert)

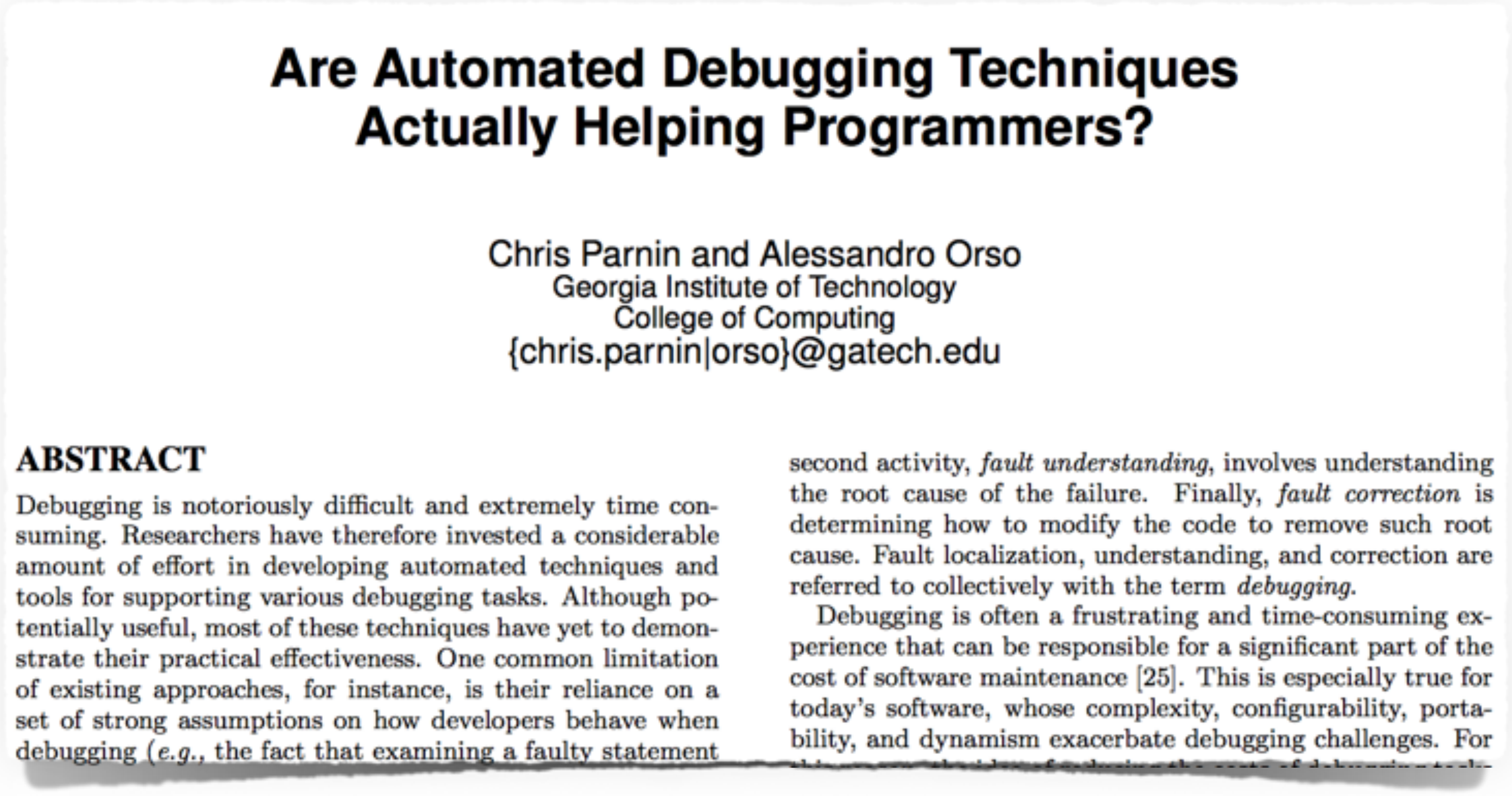


Humm….

- Are we properly quantifying diagnostic accuracy?
  - Comparing techniques based on the rankings
  - Assuming perfect bug understanding
- Are we showing providing an ecosystem offering this techniques?

Human Studies

Parnin & Orso *et al* observed that there is a lack of human studies! (ISSTA'11)



Are Automated Debugging Techniques Actually Helping Programmers?

Chris Parnin and Alessandro Orso
Georgia Institute of Technology
College of Computing
{chris.parnin|orso}@gatech.edu

ABSTRACT

Debugging is notoriously difficult and extremely time consuming. Researchers have therefore invested a considerable amount of effort in developing automated techniques and tools for supporting various debugging tasks. Although potentially useful, most of these techniques have yet to demonstrate their practical effectiveness. One common limitation of existing approaches, for instance, is their reliance on a set of strong assumptions on how developers behave when debugging (*e.g.*, the fact that examining a faulty statement

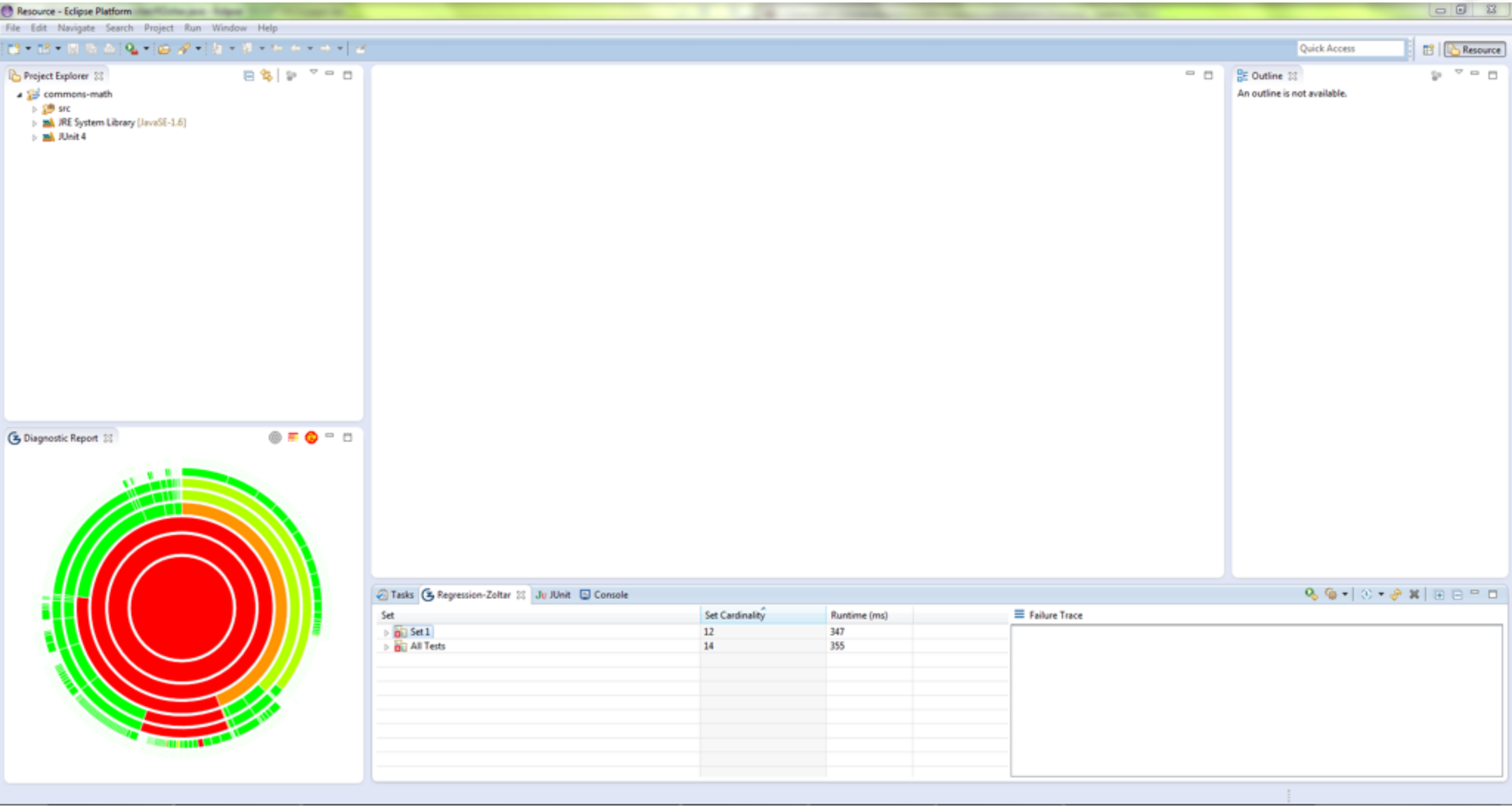
second activity, *fault understanding*, involves understanding the root cause of the failure. Finally, *fault correction* is determining how to modify the code to remove such root cause. Fault localization, understanding, and correction are referred to collectively with the term *debugging*.

Debugging is often a frustrating and time-consuming experience that can be responsible for a significant part of the cost of software maintenance [25]. This is especially true for today's software, whose complexity, configurability, portability, and dynamism exacerbate debugging challenges. For

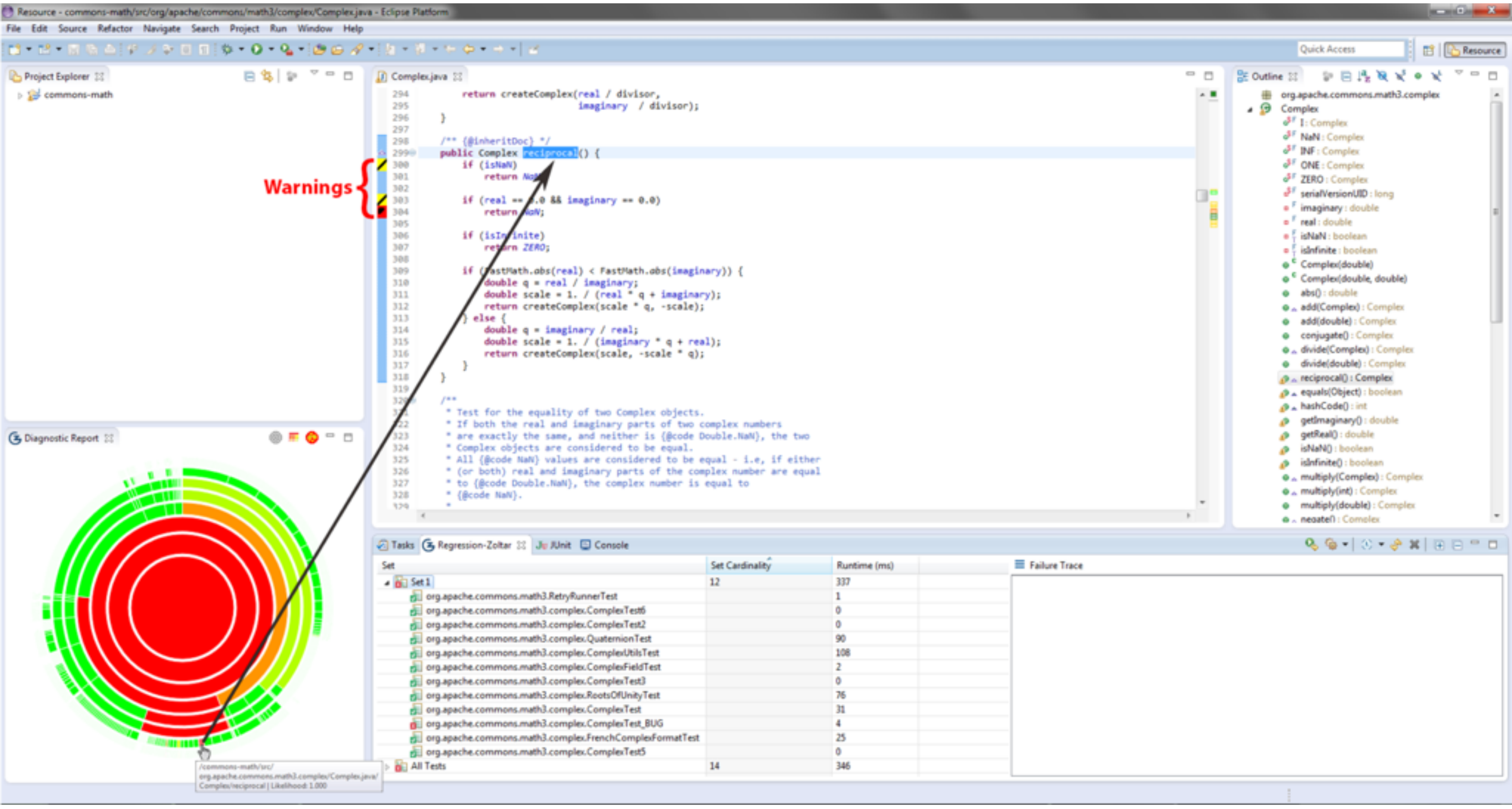
Crowbar

— http://www.crowbar.io —

Previously known as GZoltar

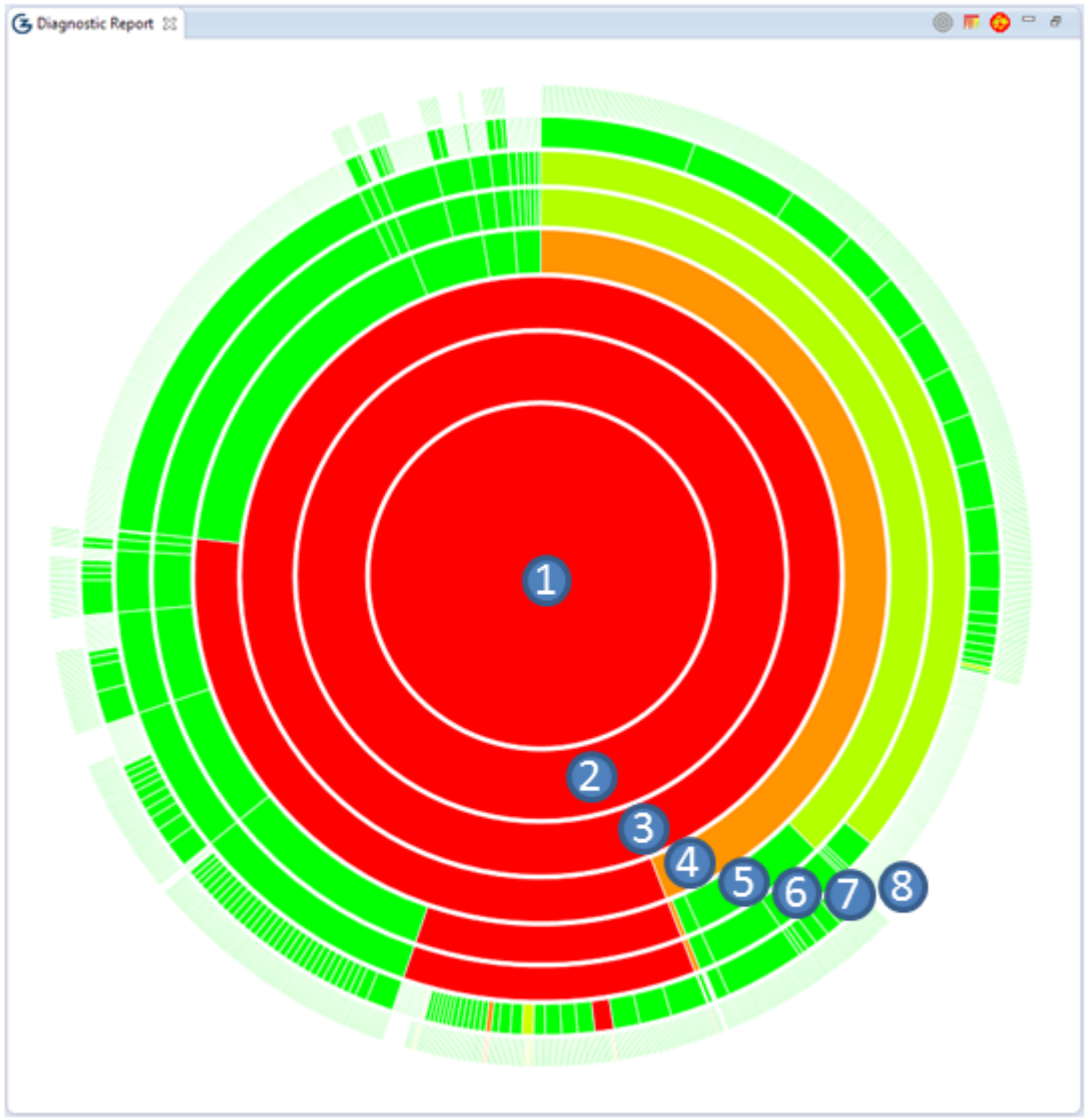

Resource - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Quick Access
Resource
Project Explorer
commons-math
src
JRE System Library [JavaSE-1.6]
JUnit 4
Outline
An outline is not available.
Diagnostic Report
Tasks
Regression-Zoltar
JUnit
Console
Failure Trace

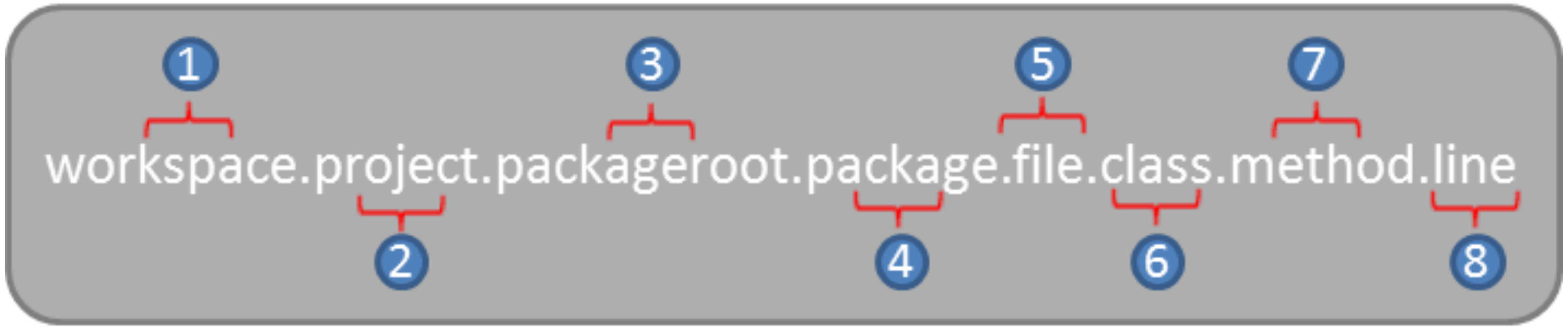
Set	Set Cardinality	Runtime (ms)
Set 1	12	347
All Tests	14	355



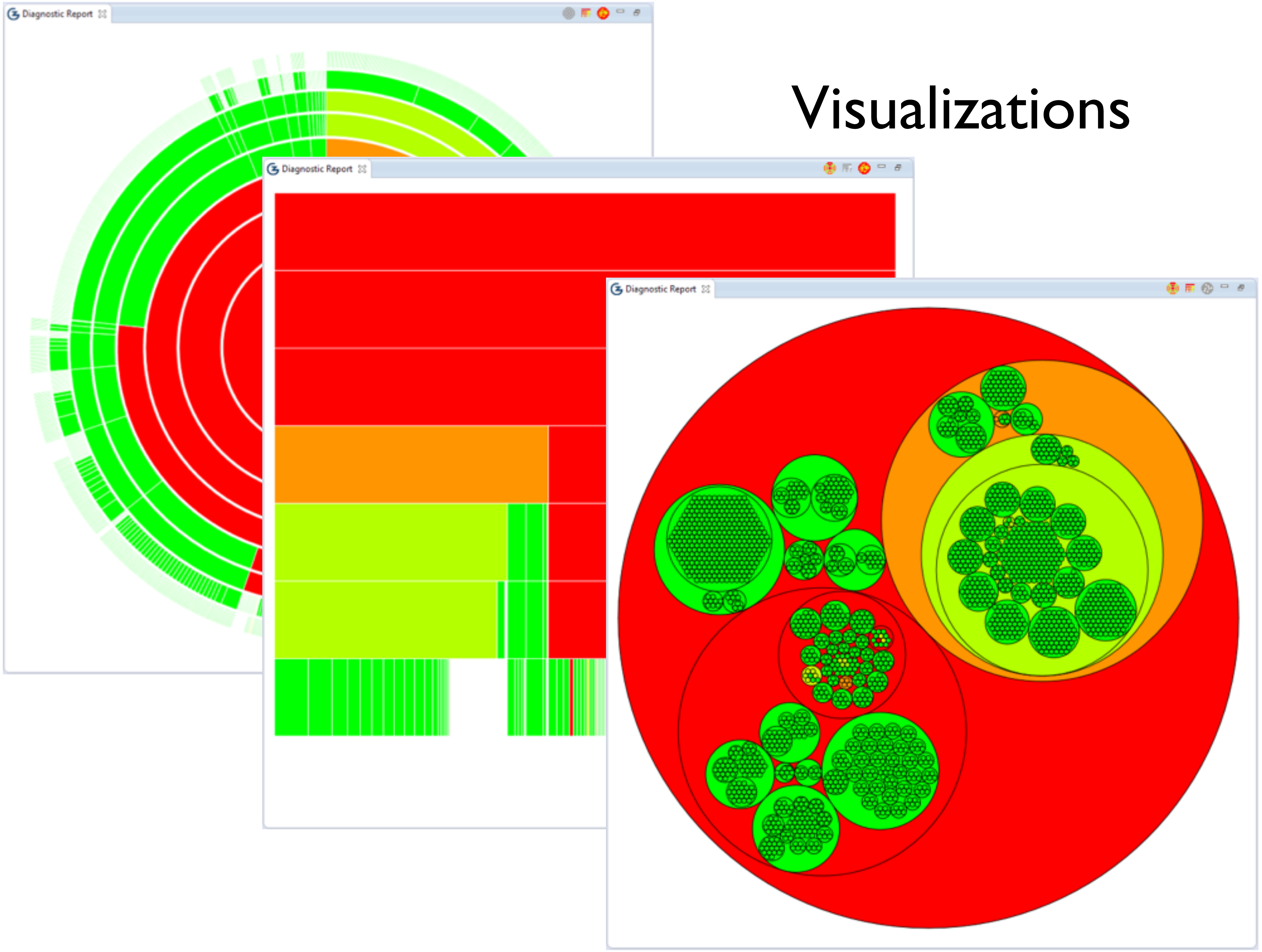
```java
            return createComplex(real / divisor,
                                 imaginary  / divisor);
    }

    /** {@inheritDoc} */
    public Complex reciprocal() {
        if (isNaN)
            return NaN;

        if (real == 0.0 && imaginary == 0.0)
            return NaN;

        if (isInfinite)
            return ZERO;

        if (FastMath.abs(real) < FastMath.abs(imaginary)) {
            double q = real / imaginary;
            double scale = 1. / (real * q + imaginary);
            return createComplex(scale * q, -scale);
        } else {
            double q = imaginary / real;
            double scale = 1. / (imaginary * q + real);
            return createComplex(scale, -scale * q);
        }
    }

    /**
     * Test for the equality of two Complex objects.
     * If both the real and imaginary parts of two complex numbers
     * are exactly the same, and neither is {@code Double.NaN}, the two
     * Complex objects are considered to be equal.
     * All {@code NaN} values are considered to be equal - i.e, if either
     * (or both) real and imaginary parts of the complex number are equal
     * to {@code Double.NaN}, the complex number is equal to
     * {@code NaN}.
     *
```

Set	Set Cardinality	Runtime (ms)
Set 1	12	337
org.apache.commons.math3.RetryRunnerTest		1
org.apache.commons.math3.complex.ComplexTest6		0
org.apache.commons.math3.complex.ComplexTest2		0
org.apache.commons.math3.complex.QuaternionTest		90
org.apache.commons.math3.complex.ComplexUtilsTest		108
org.apache.commons.math3.complex.ComplexFieldTest		2
org.apache.commons.math3.complex.ComplexTest3		0
org.apache.commons.math3.complex.RootsOfUnityTest		76
org.apache.commons.math3.complex.ComplexTest		31
org.apache.commons.math3.complex.ComplexTest_BUG		4
org.apache.commons.math3.complex.FrenchComplexFormatTest		25
org.apache.commons.math3.complex.ComplexTest5		0
All Tests	14	346


Diagnostic Report
1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8
workspace.project.packageroot.package.file.class.method.line

Visualizations

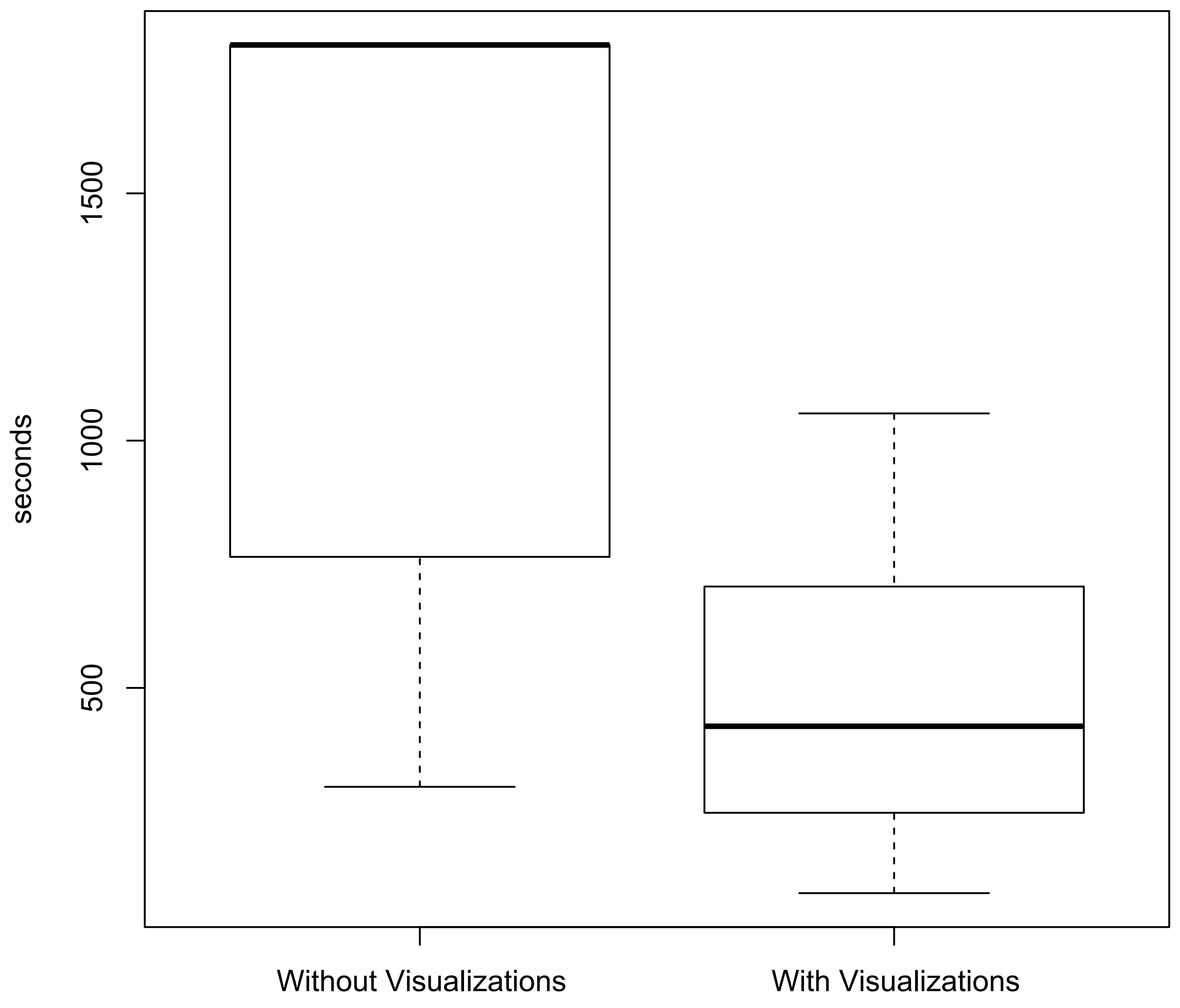


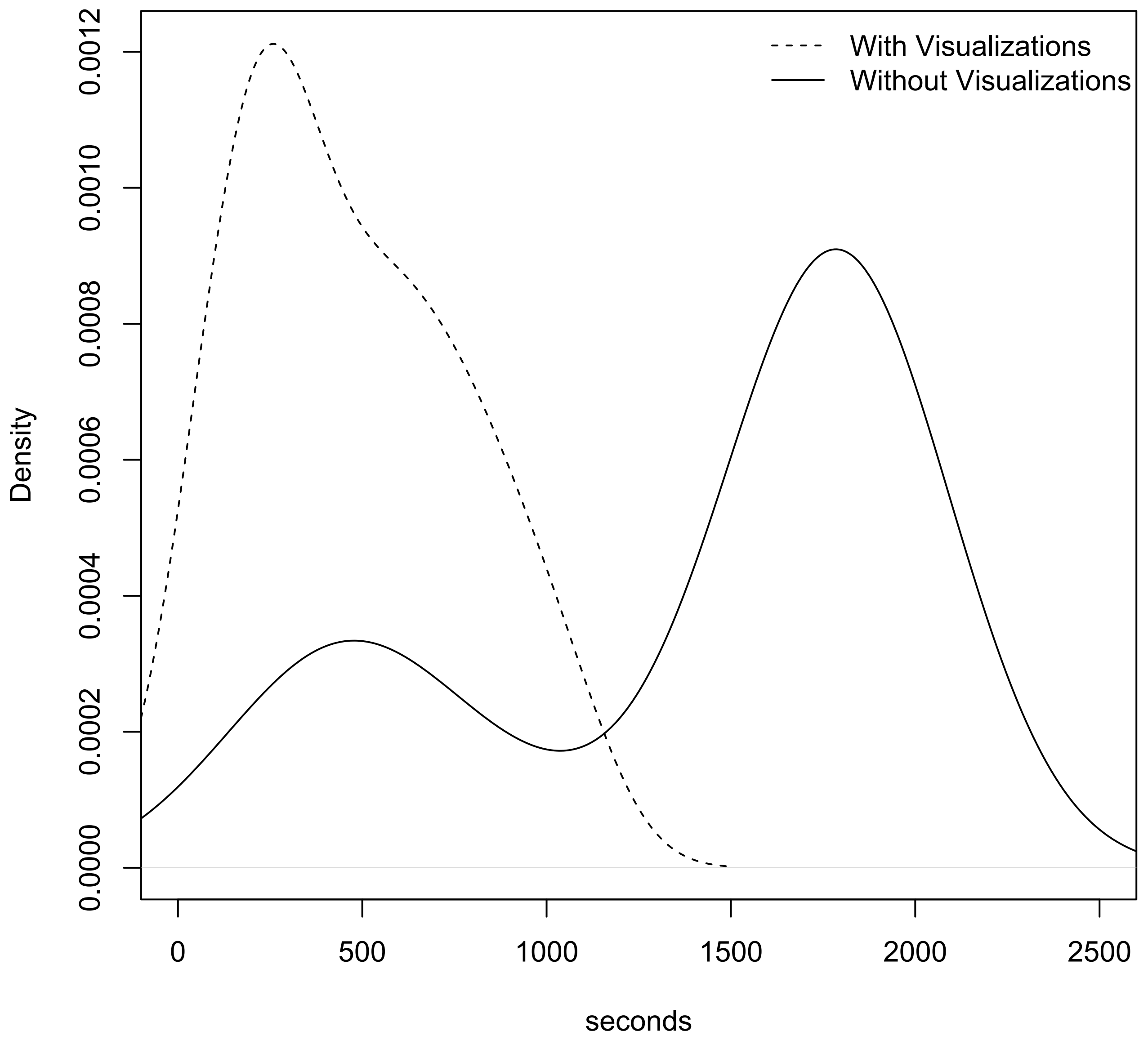
User Study: Setup

- 40 participants
- Intention: GZoltar vs. IDE's features
- Program: Xtream
  - 17,389 LOC
  - 306 classes and 22 packages
  - 1418 unit test cases
  - Injected 1 logical fault

User Study: Results

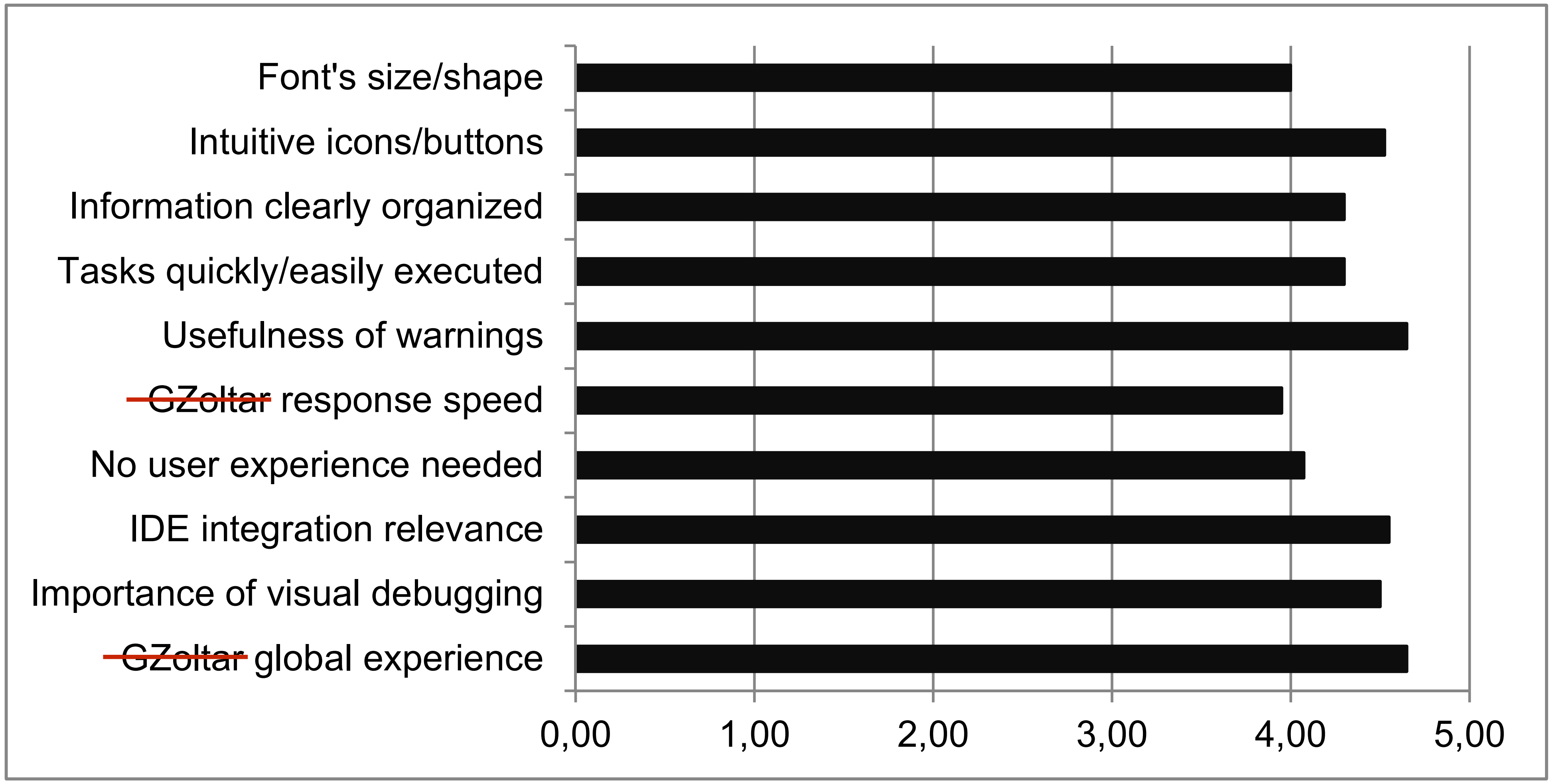
RQ1: Do the proposed visualizations efficiently aid the user to quickly find a fault?





User Study: Results

RQ2: Is Crowbar a usable toolset?



Importance of Testing

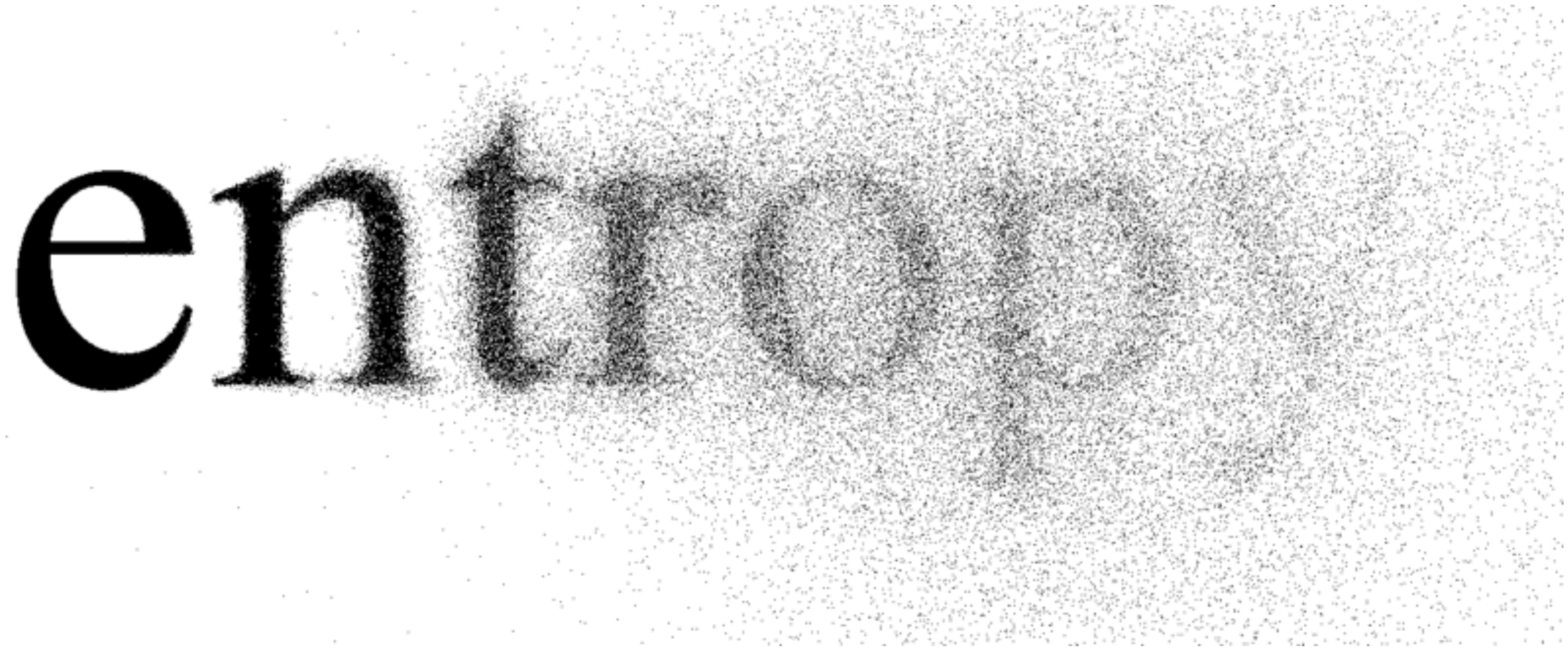
class Triangle {… static int type(int a, int b, int c) {	t1	t2	t3	t4	t5	t6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;	●						0.10001
else if ((a*a) == ((b*b) + (c*c)))		●	●	●			0.09999
type = RIGHT;			●				0.10001
else if ((a == b) || (b == a)) /* FAULT */		●		●			0.10000
type = ISOSCELES;		●					0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } … }					●	●	0.10000

“A confounding factor for the usefulness of SFL is the dependency on the **quality of the existing test suite**”

Diagnostic Performance

$C_d = 4$

Rank Position	Suspicious Statement	Line number	Suspiciousness
1º	`type = EQUILAT`	3	0.10001
2º	`type = RIGH`	5	0.10001
3º	`type = ISO ES;`	7	0.10001
4º	`else if ( b) == /* FAULT */`	6	0.10000
5º	`double s = )/2.0;`	9	0.10000
6º	`return Math.sqr b)*(s-c));`	10	0.10000
7º	`else if ( (a*a) == ((b*b) + (c*c)) )`	4	0.09999
8º	`int type = SCALENE;`	1	0.09998
9º	`if ( (a == b) && (b == c) )`	2	0.09998
10º	`return type; }`	8	0.09998



$$\mathcal{H}(D) = -\sum_{d_k \in D} \Pr(d_k) \cdot \log_2(\Pr(d_k)),\ 0 \leq \mathcal{H} \leq \log_2(M)$$

A. Gonzalez-Sanchez, R. Abreu, H.-G. Gross, and A. J. van Gemund, “Spectrum-Based Sequential Diagnosis”, AAAI ’11

Measuring Entropy

Rank Position	Suspicious Statement	Line number	Suspiciousness
1º	`type = EQUILATERAL;`	3	0.10001
2º	`type = RIG` [illegible]	5	0.10001
3º	`type =` [illegible] `LES;`	7	0.1[illegible]001
4º	`else i` [illegible] `(a =` [illegible] `b)` [illegible] `(b` [illegible] `) ) /*` [illegible] `FA` [illegible] `LT */`	[illegible]	0.100[illegible]0
5º	`double` [illegible] `(a+` [illegible] `0;`	[illegible]	0.100[illegible]0
6º	`return` [illegible] `qrt(s*(s-` [illegible] `b)*(s-c));`	10	0.[illegible]0000
7º	`else if ( (` [illegible] `) + (c*c)) )`	4	0.09999
8º	`int type = SCALENE;`	1	0.09998
9º	`if ( (a == b) && (b == c) )`	2	0.09998
10º	`return type; }`	8	0.09998

$\mathcal{H} = 3.322$

The **variety** of test cases is the major factor to have **uncertainty in the ranking**

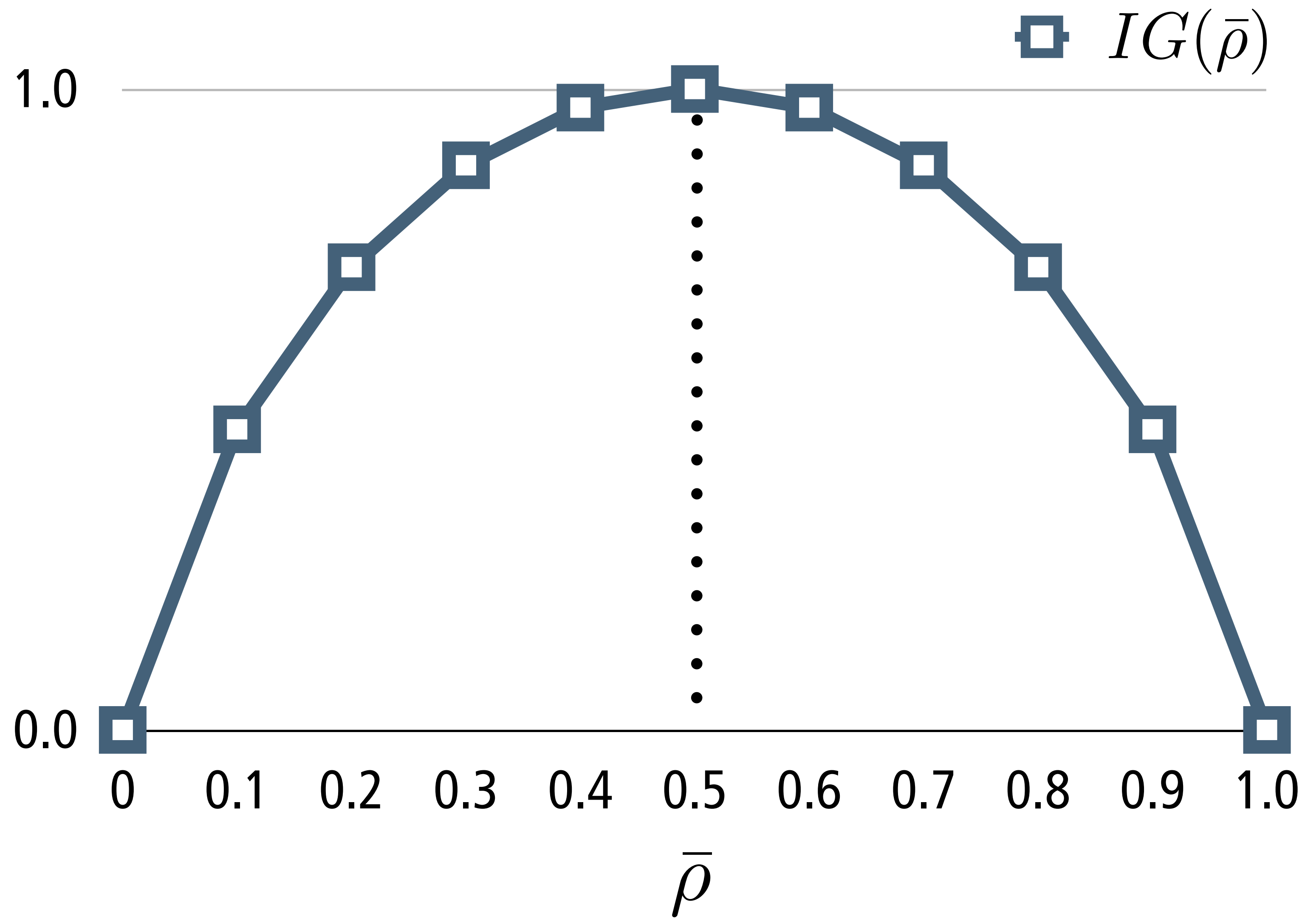
Density of a Test Suite

class Triangle {… static int type(int a, int b, int c) {	t1	t2	t3	t4	t5	t6	Suspiciousness
int type = SCALENE;	●	●	●	●			0.09998
if ((a == b) && (b == c))	●	●	●	●			0.09998
type = EQUILATERAL;	●						0.10001
else if ((a*a) == (([illegible] + (c*c)))	[illegible]	●	●	●			0.09999
type = RIGHT;	[illegible]	[illegible]	●	[illegible]			0.10001
else if ((a == b) | [illegible] = a)) /* FA [illegible]	[illegible]	●	[illegible]	●			0.10000
type = ISOSCELES;	[illegible]	●	[illegible]	[illegible]			0.10001
return type; }	●	●	●	●			0.09998
static double area(int a, int b, int c) {							
double s = (a+b+c)/2.0;					●	●	0.10000
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }					●	●	0.10000

?

$\bar{\rho} = 0.4$

A. Gonzalez-Sanchez, R. Abreu, H.-G. Gross, and A. J. van Gemund, "Prioritizing tests for fault localization through ambiguity group reduction", ASE '11 17

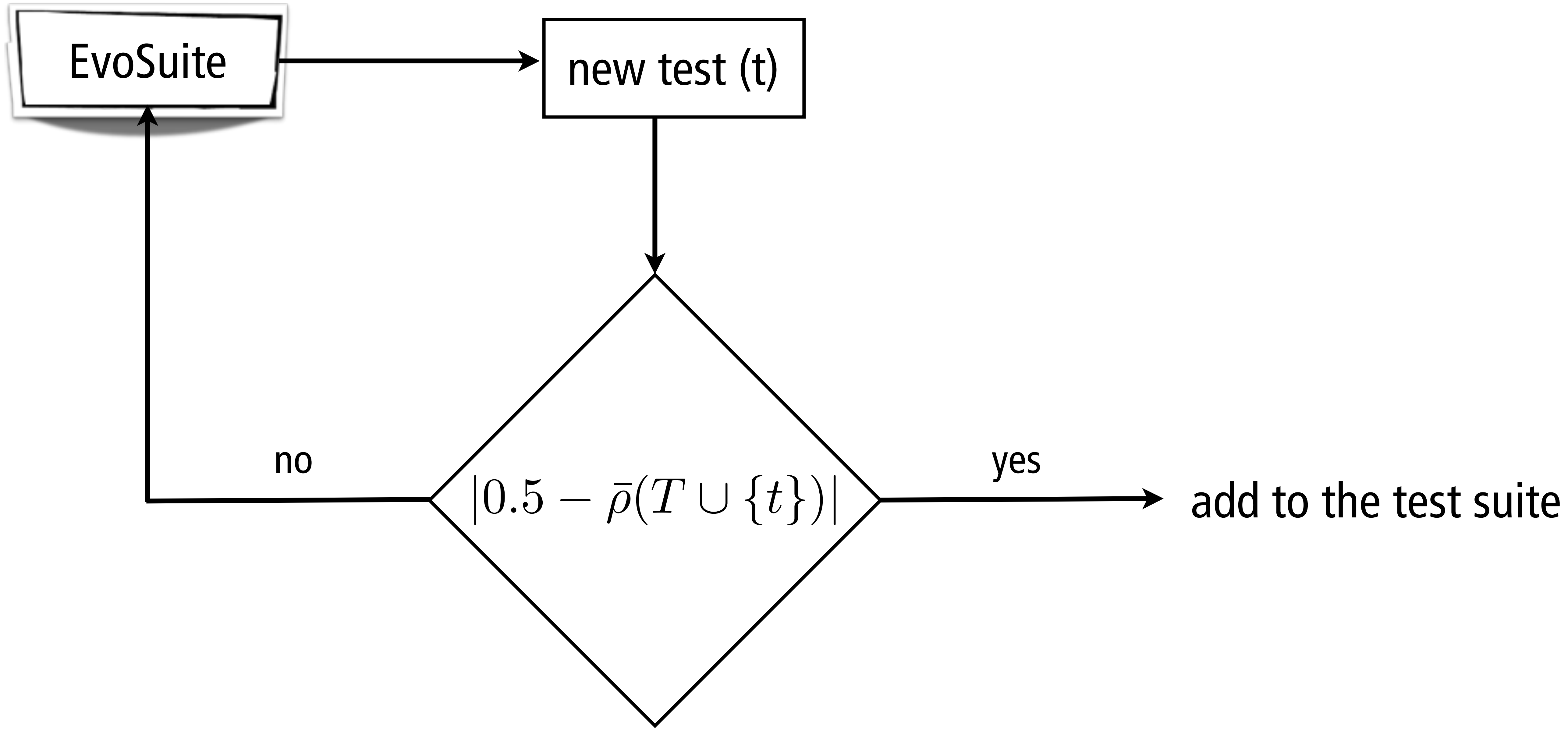


$$IG(\bar{\rho}) = -\bar{\rho} \cdot \log_2(\bar{\rho}) - (1 - \bar{\rho}) \cdot \log_2(1 - \bar{\rho})$$

R. A. Johnson, "An information theory approach to diagnosis", IRE Transactions on Reliability and Quality Control, no. 1, pp. 35–35, 1960

A fitness function based on **entropy** to guide search-based **test generation** and to optimize the quality of ranking reports

ENTBUG


EvoSuite
new test (t)
$|0.5 - \bar{\rho}(T \cup \{t\})|$
no
yes
add to the test suite

$\bar{\rho}$ = 0.500

$\mathcal{H}\downarrow$-27%

C_d = 0.0

class Triangle {… static int type(int a, int b, in	T		T + {t7}			T + {t_7, t_8}			T + {t_7, t_8, t_9}		
	º	Suspiciousness	t_7	º	Suspiciousness	t_8	º	Suspiciousness	t_9	º	Suspiciousness
int type = SCALENE;	8	0.09998	●	6	0.03629	●	6	0.02354	●	5	0.04347
if ((a == b) && (b == c))	9	0.09998	●	7	0.03629	●	7	0.02354	●	6	0.04347
type = EQUILATERAL;	1	0.10001				●					
else if ((a*a) == ((b*b) + (c*c)))	7	0.09999	●	5	0.08466		3	0.10983	●	2	0.17391
type = RIGHT;	2	0.10001	●	1	0.29033		1	0.37666			
else if ((a == b) || (b == */	4	0.10000	●	2	0.17204		2	0.22320	●	1	0.34782
type = ISOSCELES;	3	0.10001									
return type; }	10	0.09998	●	8	0.03629	●	8	0.02354	●	7	0.04347
static double area(int a, int int c) {											
double s = (a+b+c)/2.0;	5	0.10000	●	3	0.17204	●	4	0.10983	●	3	0.17391
return Math.sqrt(s*(s-a)*(s-b)*(s-c)); } ... }	6	0.10000	●	4	0.17204	●	5	0.10983	●	4	0.17391
Test case outcome (pass = ✓, fail = ✗)			✗			✓			✗		

	T	T + {t7}	T + {t_7, t_8}	T + {t_7, t_8, t_9}
$\bar{\rho}$	0.400	0.457	0.475	0.500
$\mathcal{H}$	3.322	2.651	2.445	2.437
C_d	4.000	2.000	1.000	0.000

http://www.gzoltar.com

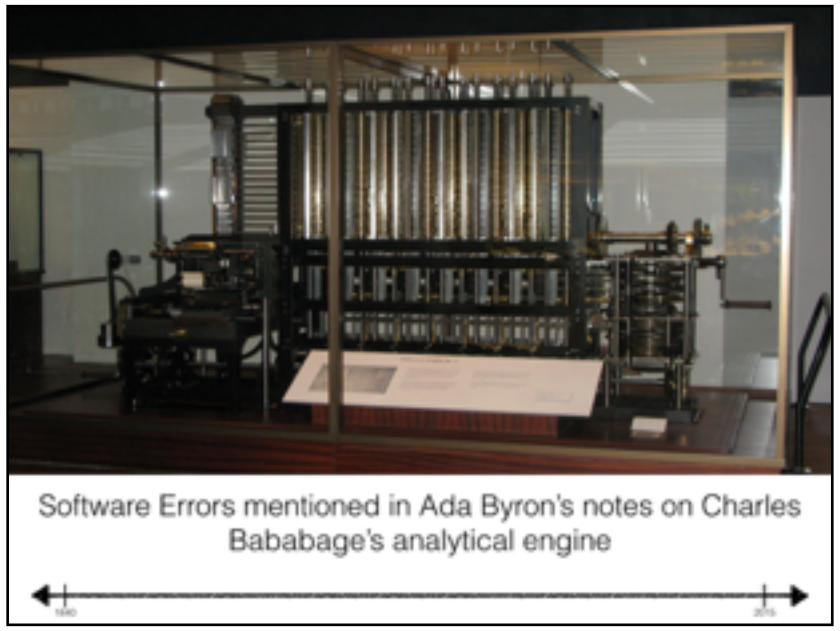
- Available as an Eclipse plug-in
  - a Visual Studio plugin will be released soon
- Also available as a library
  - Instrumentation and diagnosis
- Testing features are yet to be deployed
  - Only test suite minimization available

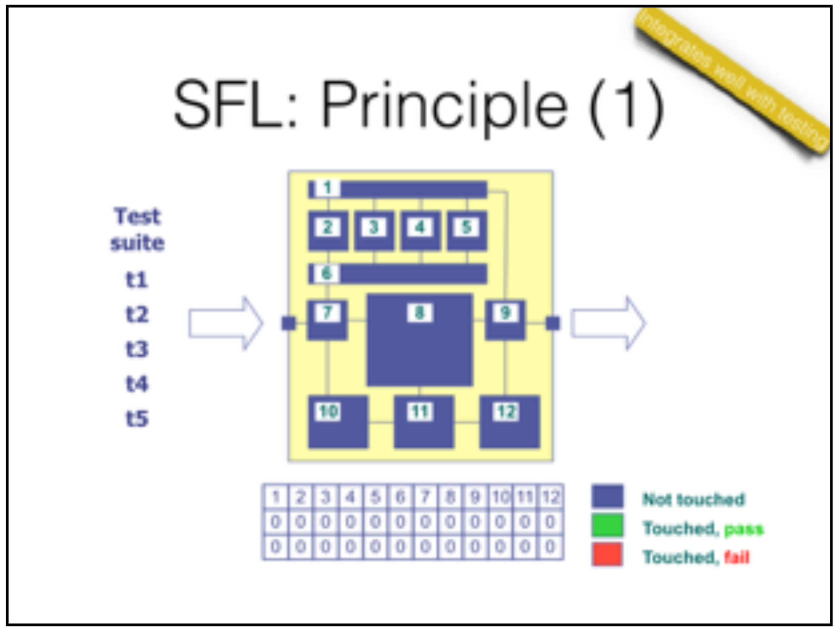
Let's use it

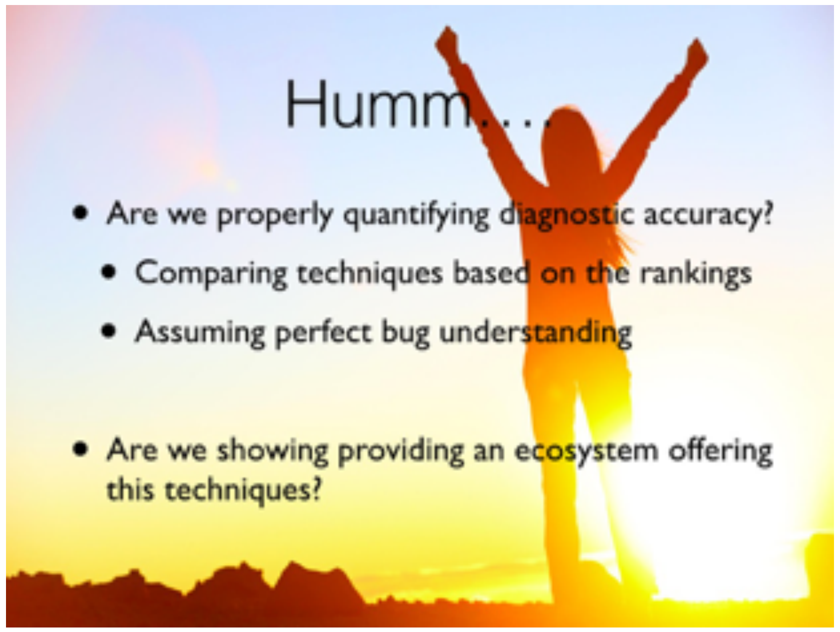
- Open Eclipse
- Install Crowbar
  - Help ➤ Install New Software
    - http://crowbar.io/plugin/tarot/
  - Window ➤ Other… ➤ Crowbar Views ➤ Diagnostic Reports
- Import (as maven project) buggy yodaTime
  - http://crowbar.io/plugin/tarot/buggy_yodatime.zip
- Find the bug!

Opportunities and Challenges

- Integration with software repository mining
- Use fitness function in test suite prioritization and minimization
- Generation: How to solve the oracle problem?
  - Human in the loop
  - AutoSeer project: leverage program invariants
- Explore idiosyncrasies of mobile devices

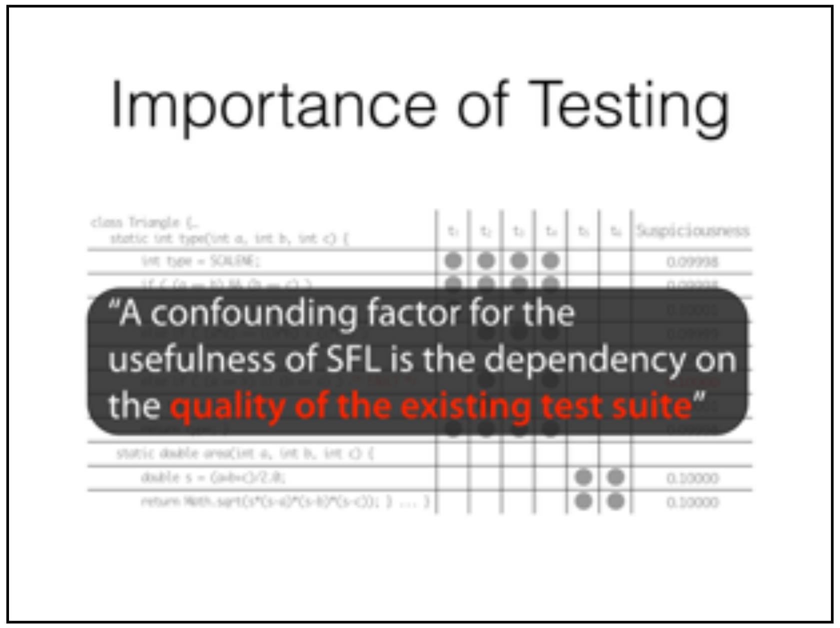

Software Errors mentioned in Ada Byron's notes on Charles Bababage's analytical engine


SFL: Principle (1)

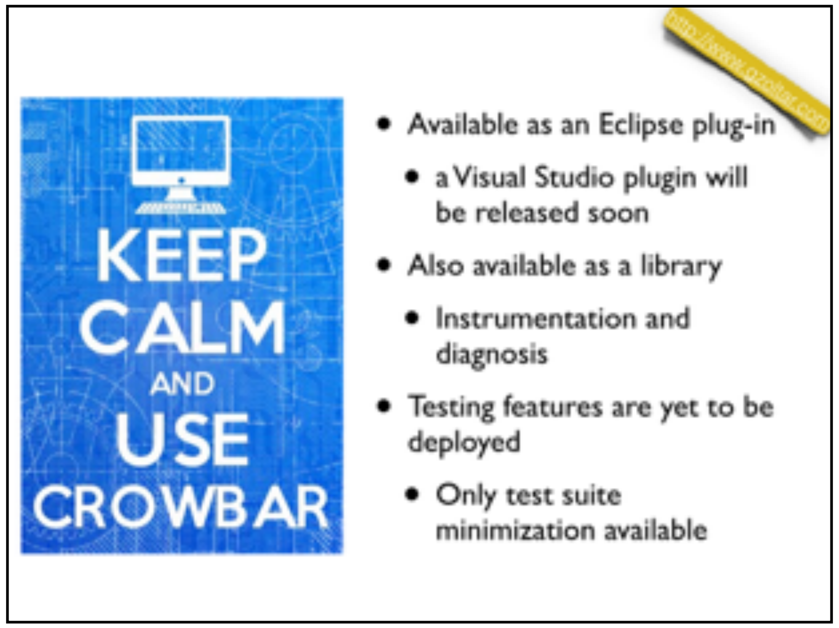

Humm…

- Are we properly quantifying diagnostic accuracy?
  - Comparing techniques based on the rankings
  - Assuming perfect bug understanding
- Are we showing providing an ecosystem offering this techniques?

Questions?


Importance of Testing

"A confounding factor for the usefulness of SFL is the dependency on the quality of the existing test suite"



- Available as an Eclipse plug-in
  - a Visual Studio plugin will be released soon
- Also available as a library
  - Instrumentation and diagnosis
- Testing features are yet to be deployed
  - Only test suite minimization available

Let's use it!

- Open Eclipse
- Install Crowbar
  - Help ➤ Install New Software
    - http://crowbar.io/plugin/tarot/
  - Window ➤ Other… ➤ Crowbar Views ➤ Diagnostic Reports
- Import (as maven project) buggy yodaTime
  - http://crowbar.io/plugin/tarot/buggy_yodatime.zip
- Find the bug!