STAR: Stack-Trace based Automatic crash Reproduction

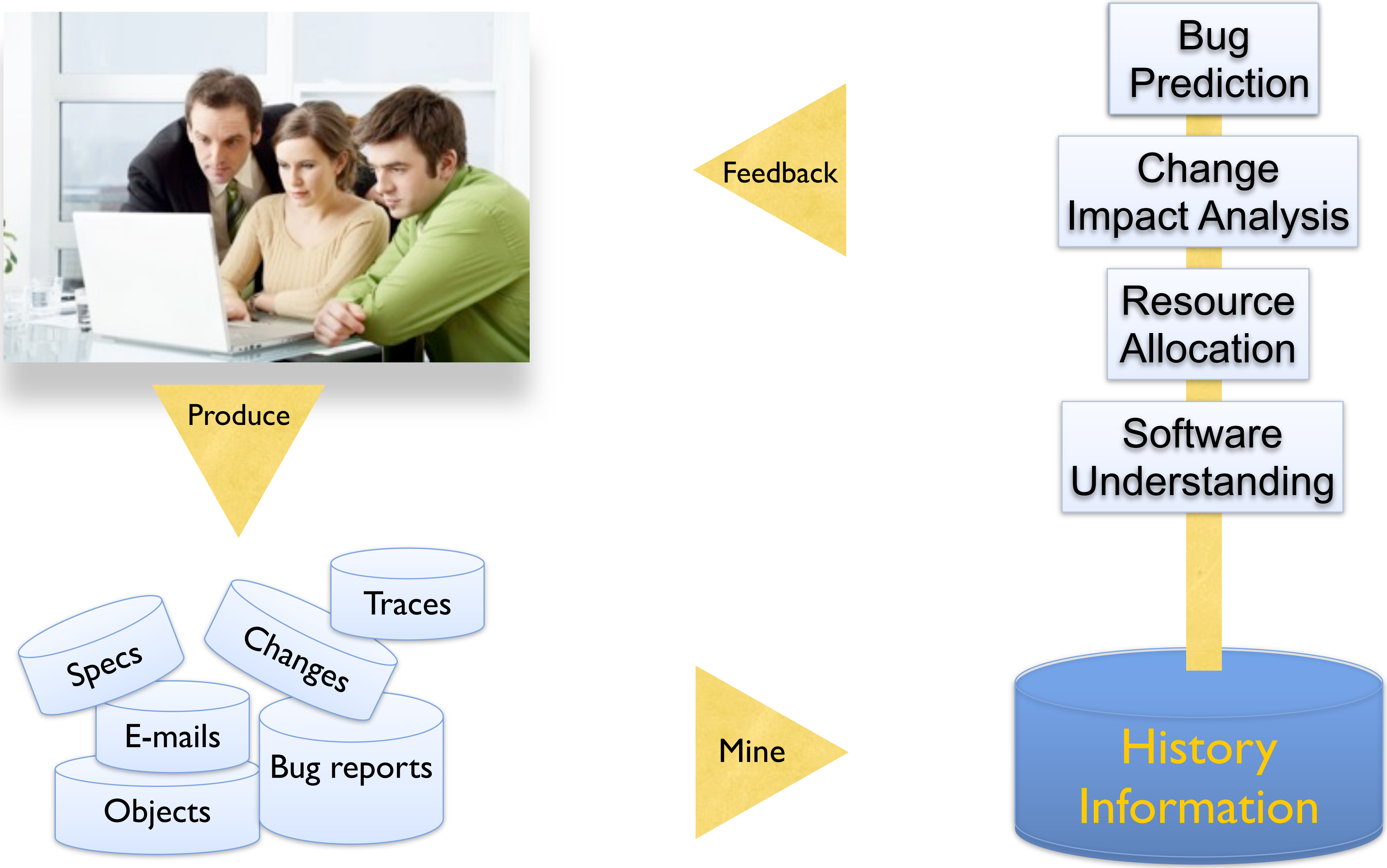
Sung Kim <hunkim@cse.ust.hk>
The Hong Kong University of Science and Technology
with Ning Chen and Hojun Jaygarl

ROSAEC Workshop Jan 6, 2011

Sung’s research areas

- MSR: Mining Software Repositories
  - Defect prediction (learning from repositories)
  - Bug triage/bug report mining
  - Crash report/stack trace mining
  - Code clones
- Static Analysis
  - Unit test generation
  - Crash reproduction
  - Patch generation

Mining Software Repositores



Feedback

Produce

Specs

Changes

Traces

E-mails

Bug reports

Objects

Mine

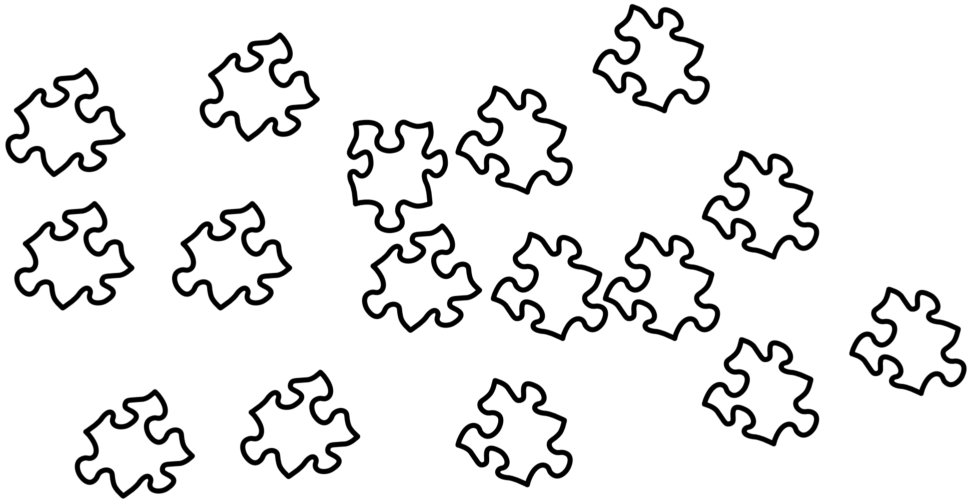
Bug Prediction

Change Impact Analysis

Resource Allocation

Software Understanding

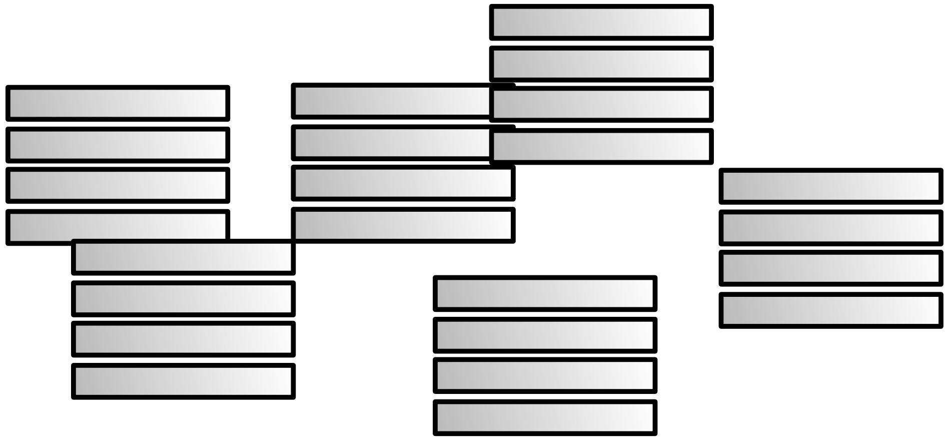
History Information



objects

Mine

Testing

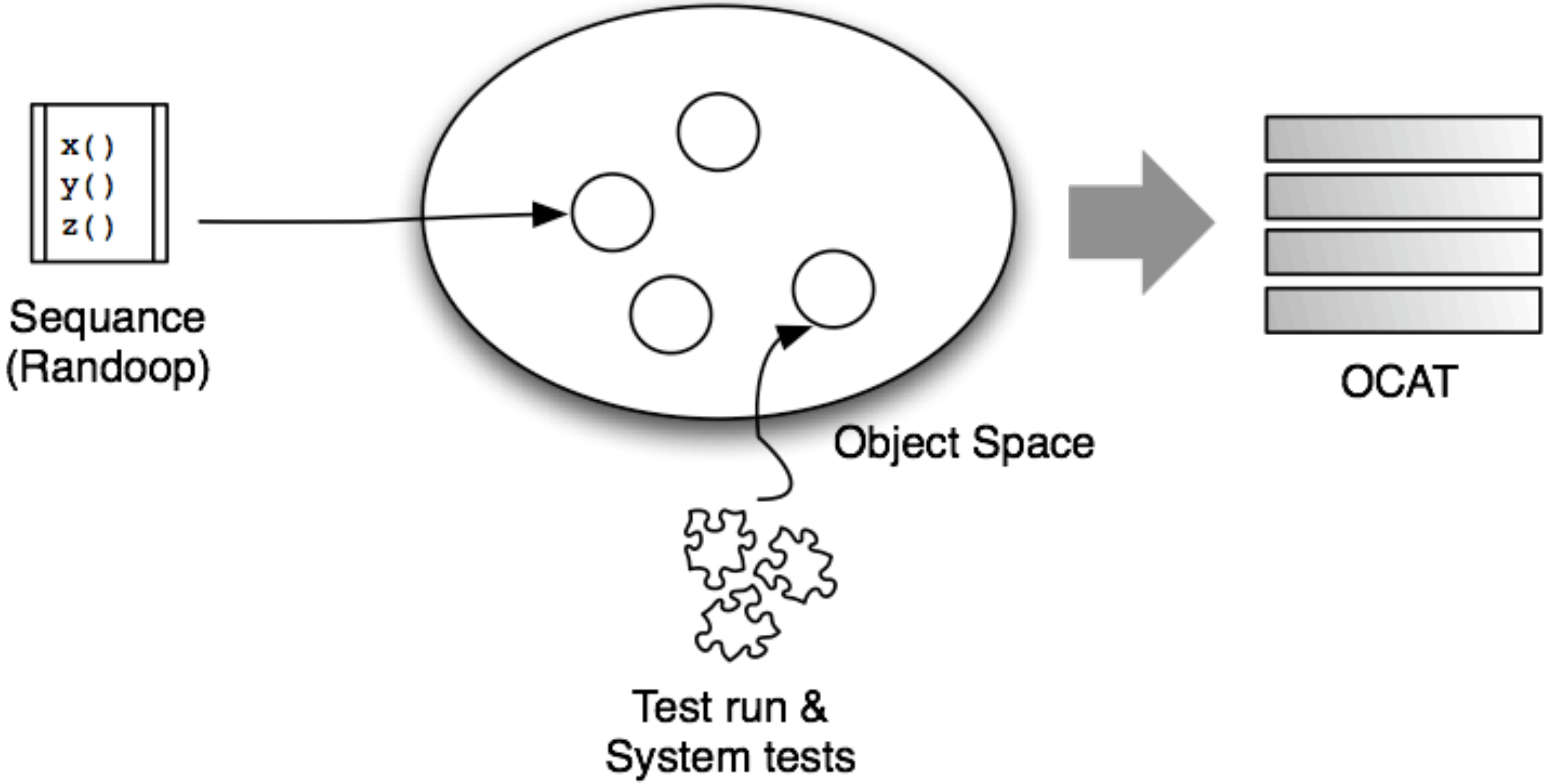


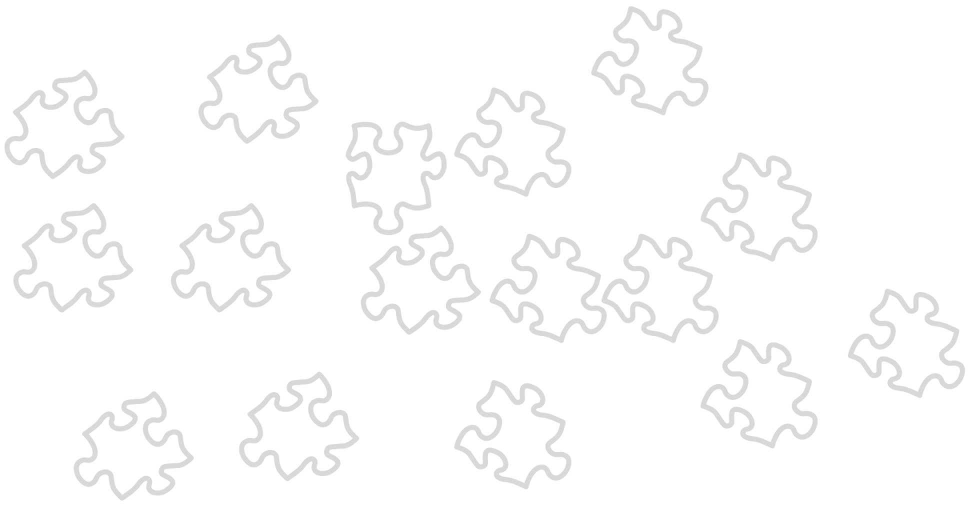
stack traces

Mine

Crash repro

ROSAEC Workshop 2010

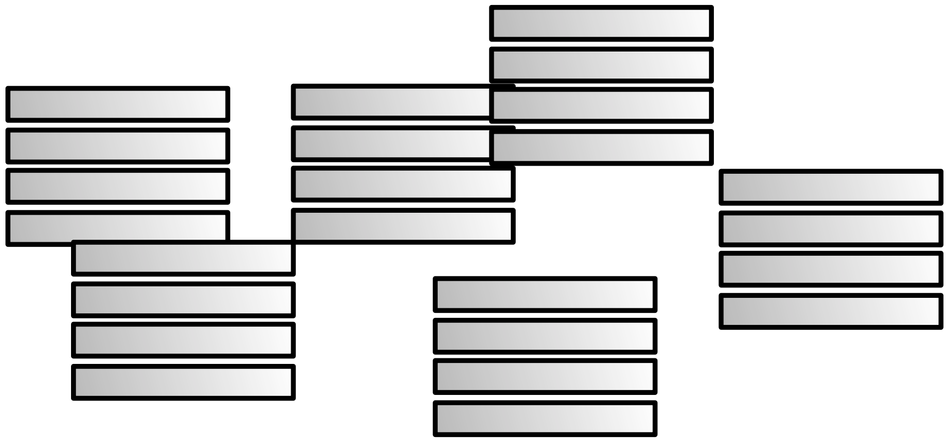




objects

Mine

Testing



stack traces

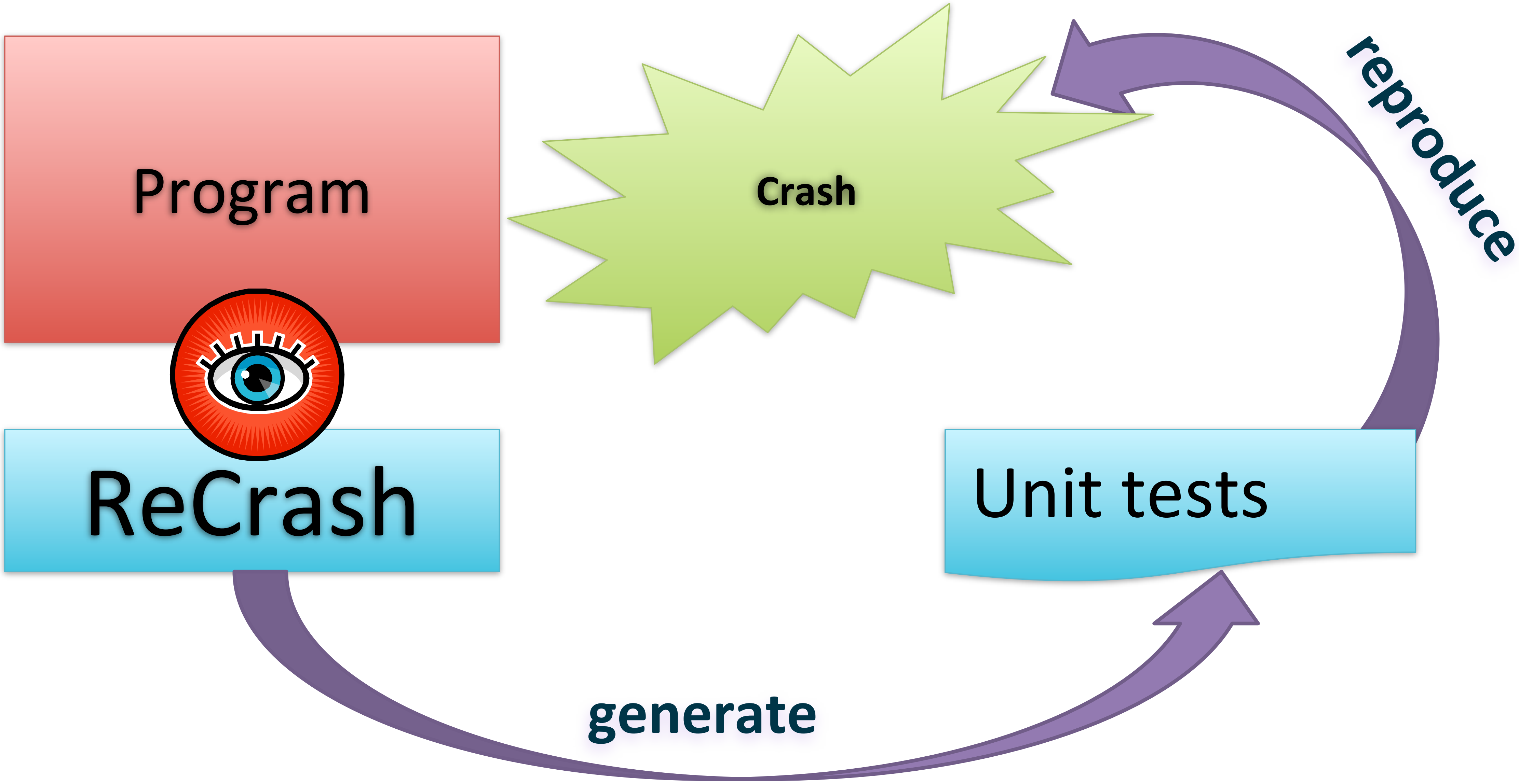
Mine

Crash repro

Reproducing Crashes

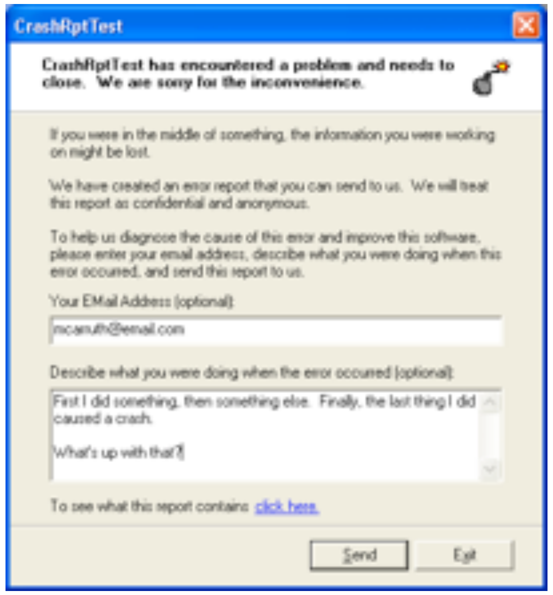
- Must be able to reproduce crashes for debugging
  - To fix bugs and validate fixes
- Reproducing crashes (faults) is hard!
  - Require the exact configuration of crash (in field)

ReCrash



6-64% performance overhead

Crash Reporting System



CrashRptTest has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To help us diagnose the cause of this error and improve this software, please enter your email address, describe what you were doing when the error occurred, and send this report to us.

Your EMail Address (optional):

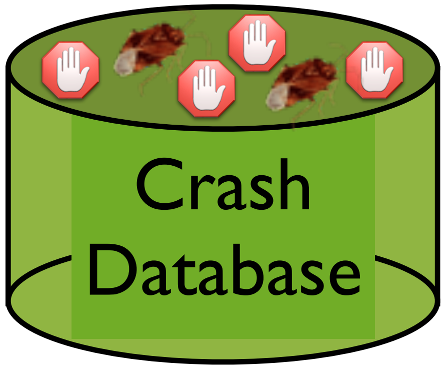
mcarruth@email.com

Describe what you were doing when the error occurred (optional):

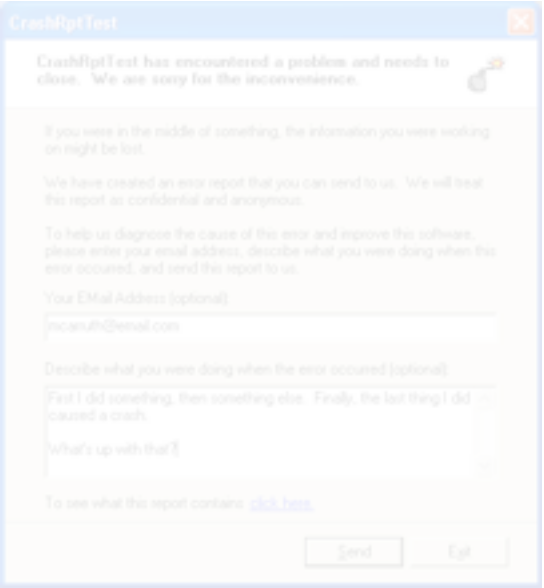
First I did something, then something else. Finally, the last thing I did caused a crash.

What's up with that?

To see what this report contains click here.



Crash Reporting System



CrashRptTest has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To help us diagnose the cause of this error and improve this software, please enter your email address, describe what you were doing when this error occurred, and send this report to us.

Your EMail Address (optional):

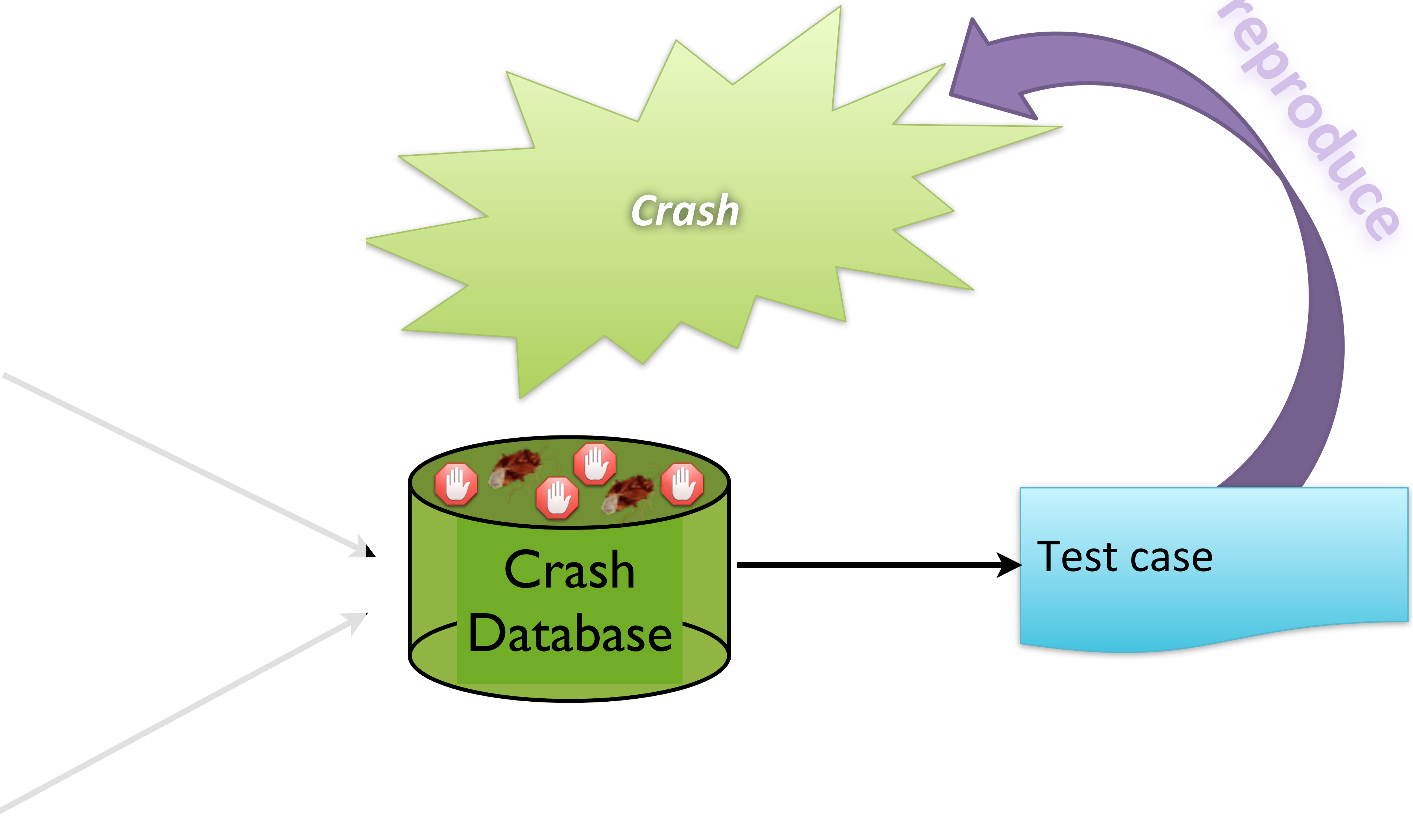
mcarruth@email.com

Describe what you were doing when the error occurred (optional):

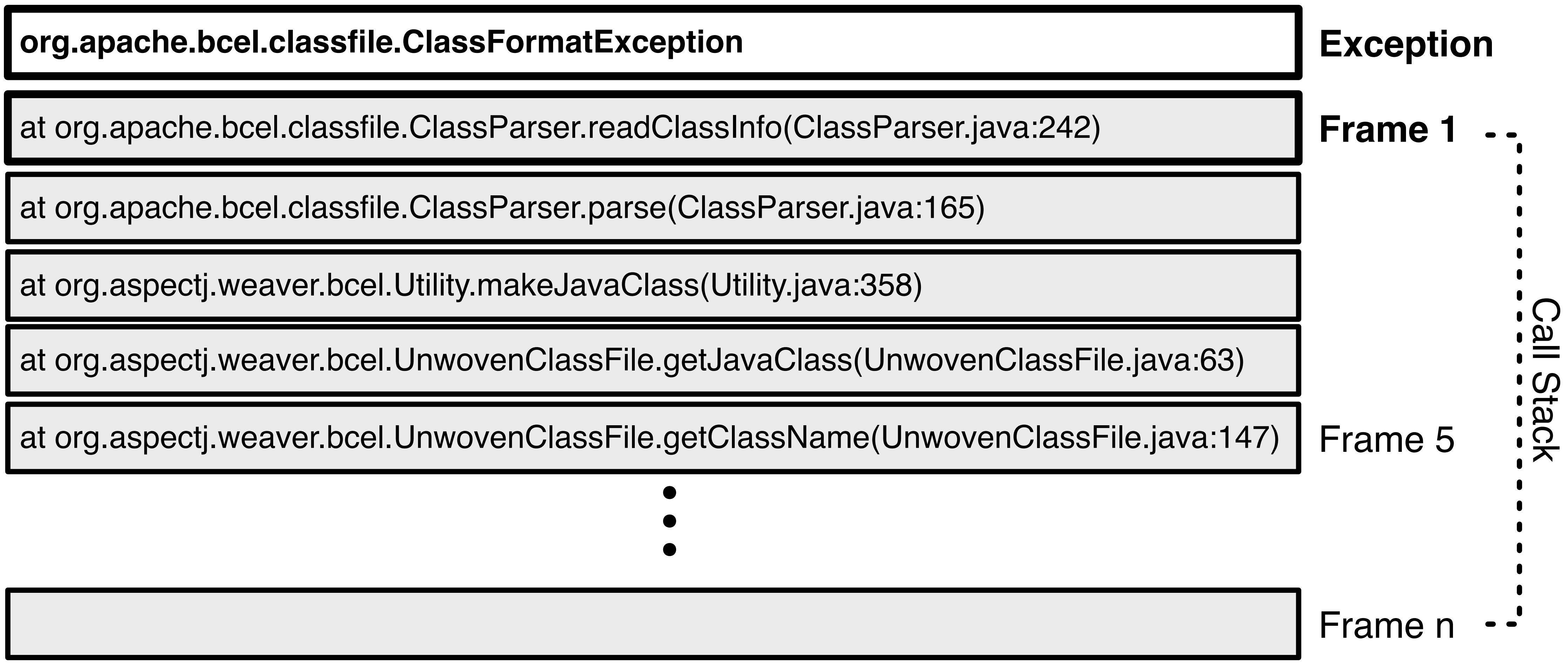
First I did something, then something else. Finally, the last thing I did caused a crash.

What's up with that?

To see what this report contains click here.



Crash Stack Traces



```
org.apache.bcel.classfile.ClassFormatException
at org.apache.bcel.classfile.ClassParser.readClassInfo(ClassParser.java:242)
at org.apache.bcel.classfile.ClassParser.parse(ClassParser.java:165)
at org.aspectj.weaver.bcel.Utility.makeJavaClass(Utility.java:358)
at org.aspectj.weaver.bcel.UnwovenClassFile.getJavaClass(UnwovenClassFile.java:63)
at org.aspectj.weaver.bcel.UnwovenClassFile.getClassName(UnwovenClassFile.java:147)
```

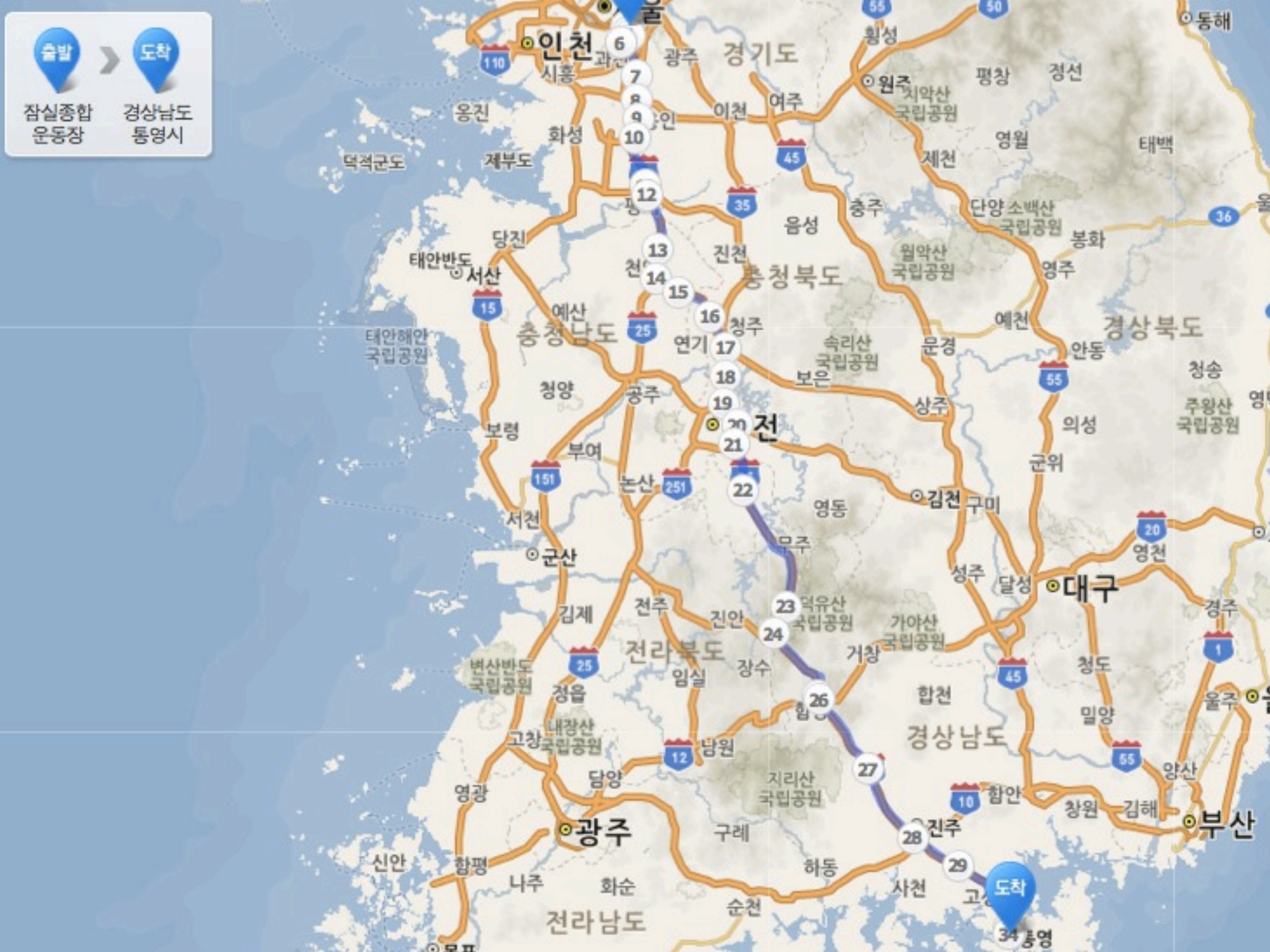
Exception

Frame 1

Frame 5

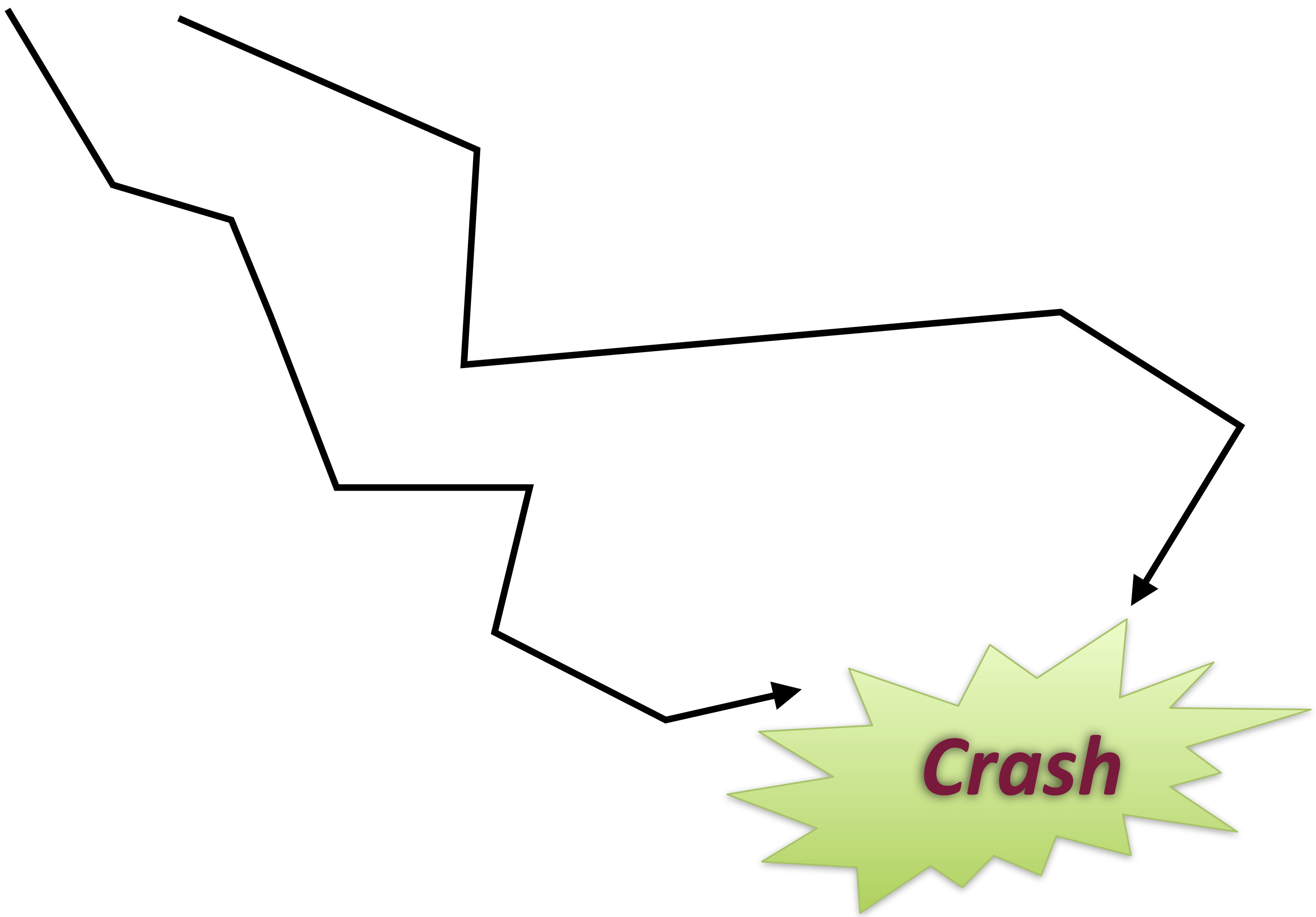
Frame n

Call Stack


출발
도착
잠실종합운동장
경상남도 통영시
인천
과천
시흥
광주
경기도
횡성
원주
치악산 국립공원
평창
정선
동해
옹진
화성
용인
이천
여주
덕적군도
제부도
영월
태백
제천
평택
당진
음성
충주
단양
소백산 국립공원
봉화
태안반도
서산
천안
진천
월악산 국립공원
충청북도
영주
예산
충청남도
청주
연기
속리산 국립공원
문경
예천
안동
경상북도
태안해안 국립공원
청양
공주
보은
상주
의성
청송
주왕산 국립공원
보령
대전
부여
논산
군위
영동
김천
구미
서천
무주
군산
영천
성주
달성
대구
경주
김제
전주
진안
덕유산 국립공원
가야산 국립공원
전라북도
거창
장수
임실
청도
변산반도 국립공원
정읍
함양
합천
밀양
울주
울산
내장산 국립공원
고창
남원
경상남도
지리산 국립공원
양산
영광
담양
함안
창원
김해
부산
광주
구례
진주
신안
함평
하동
사천
고성
나주
화순
순천
전라남도
목포
통영
110
15
25
151
251
35
45
55
12
10
20
1
36
50
6
7
8
9
10
12
13
14
15
16
17
18
19
20
21
22
23
24
26
27
28
29
34

Direction?

```
at org.apache.bcel.classfile.ClassParser.readClassInfo(ClassParser.java:242)
```



Crash Inputs

Frame 1:

ClassParser.*readClassInfo*(ClassInfo x)

(1) Receiver

➊

ClassParser.readClassInfo(ClassInfo x)

(2) Arguments

❶

```
ClassParser.readClassInfo(ClassInfo x)
```

❷

Problem Definition

```
void testCase() {




 cp.readClassInfo(x)
}
```

Problem Definition

```
void testCase() {
❶  ClassParser cp = ?


⚡  cp.readClassInfo(x)
}
```

Problem Definition

```
void testCase() {
❶  ClassParser cp = ?
❷  ClassInfo x = ?
⚡  cp.readClassInfo(x)
}
```

Three Approaches to Find Crash Inputs

- Feedback based random approach (feed)
- Object-capture based (objcap)
- Static analysis (precondition)

Feedback (Randoop)

- Find methods that return *Bar*
    - *Bar foo()* {..}
    - *Bar getBar(List x)* {..}

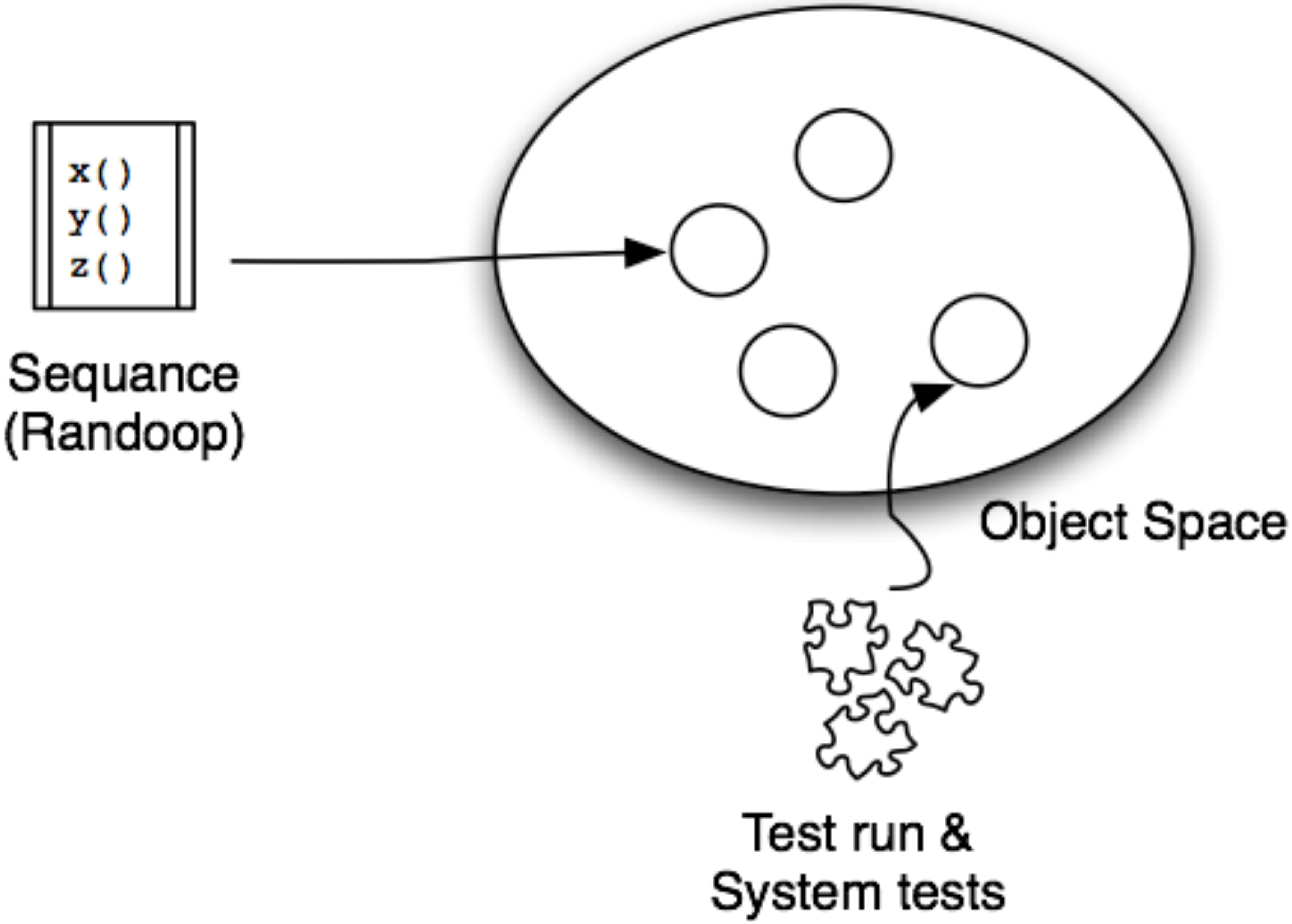
Feedback (Randoop)

- Find methods that return *Bar*
  - *Bar foo()* {..}
  - *Bar getBar(List x)* {..}
- Generates object instances recursively
  - foo = getFoo()
  - bar = get(foo)

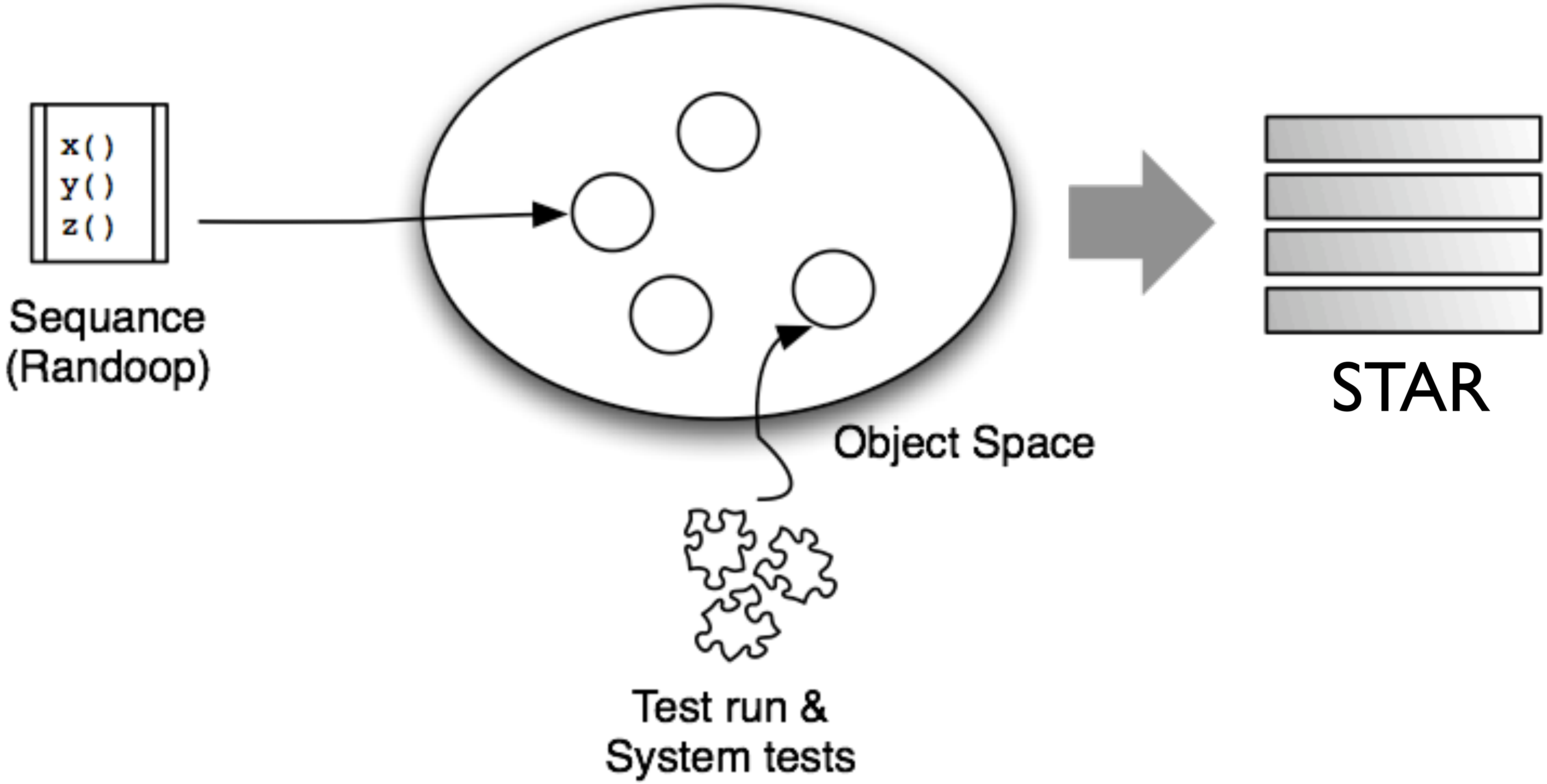
Feedback (Randoop)

- Find methods that return *Bar*
  - *Bar foo()* {..}
  - *Bar getBar(List x)* {..}
- Generates object instances recursively
  - foo = getFoo()
  - bar = get(foo)
- Mutate objects using method sequences
  - bar = get(Foo)
  - setBar(bar)
  - ...

OCAT



OCAT


x()
y()
z()
Sequance
(Randoop)
Object Space
Test run &
System tests
STAR

Mutating Object (precondition)

```
foo (Object o) {
  if (o.x>o.y) {
    o.x = o.x + o.y;
    o.y = o.x - o.y;
    o.x = o.x - o.y;
    if (o.x - o.y > 0) {
        // throw exception
    }
  }
}
```

Mutating Object (precondition)

- Identify crash condition (postcondition)
- Compute weakest precondition (wp)
- There is a wp rule for each statement in the programming language

wp rules: assignment

```
// precondition: ??
x = e;
// postcondition: Q
```

Precondition = Q with all (free) occurrences of x replaced by e

Example:

```
// assert:  ??
x = x + 1;
// assert x > 0
```

Precondition = $(x+1) > 0$

We write this as wp for "weakest precondition"

wp("`x=e;`", Q) = Q with x replaced by e

wp: if statement

```
// precondition:  ??
if (b) S1 else S2
// postcondition: Q
```

Essentially case analysis

wp(“`if (b) S1 else S2`”, Q) =

($\quad$ b ⇒ wp(“`S1`”, Q)

$\wedge$ ¬b ⇒ wp(“`S2`”, Q))

wp: composition

```
// precondition: ??
S1;        // some statement
S2;        // another statement
// postcondition:  Q
```

Work from back to front

Postcondition = wp(“**`S1; S2;`**”, Q) = wp(“**`S1;`**”, wp(“**`S2;`**”, Q))

Example:

```
// precondition: ??
x = 0;
y = x+1;
// postcondition: y > 0
```

wp example

```
foo (Object o) {          wp: o.x>o.y & o.y-o.x>0
  if (o.x>o.y) {          wp: x>y & ((x+y)-((x+y)-y))-((x+y)-y)>0
    o.x = o.x + o.y;      wp: ((x+y)-((x+y)-y))-((x+y)-y)>0
    o.y = o.x - o.y;      wp: (x-(x-y))-(x-y)>0
    o.x = o.x - o.y;      wp: (x-y)-y>0
    if (o.x - o.y > 0) {Q: x-y>0
        throw exception
    }
  }
}
```

Three Approaches to Find Crash Inputs

- Feedback based random approach (feed)
- Object-capture based (objcap)
- Static analysis (precondition)

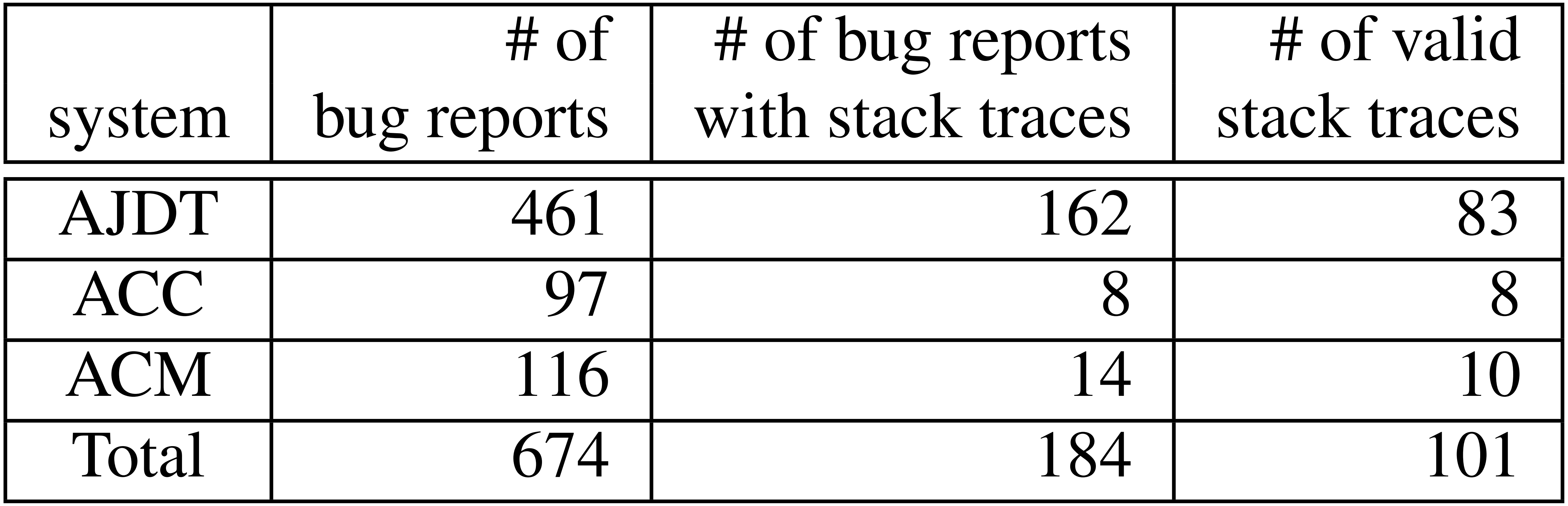
Final Test Case

```
void testCase() {
 ClassParser cp = createCP();  // random
 ClassInfo x = loadClassInfo(); // object-capture
 x. b = false;   // based on wp
cp.readClassInfo(x)
}
```

STAR Approach

- **Challenge1**: Crash points?
  - Crash reporting system (MSR)
- **Challenge II**: missing objects
  - Collect from normal execution (MSR)
- **Challenge III**: not suitable objects
  - Mutate objects (Static Analysis)

Experiments



system	# of bug reports	# of bug reports with stack traces	# of valid stack traces
AJDT	461	162	83
ACC	97	8	8
ACM	116	14	10
Total	674	184	101

Results

system	# of	# of reproducible crashes			
				d	total
AJDT 1.1.					10
AJDT 1.2.					4
AJDT 1.5.					24
ACC					5
ACM					2
total					45
percent		12.9%	+13.9%	+17.8%	44.6%

45%

Summary

- STAR approaches
  - Feedback based random approach (feed)
  - Object-capture based (objcap)
  - Static analysis (precondition)
- 45 % crash reproduction (with 0 overhead)
- Repository data (captured objects, crash traces) help static analysis

Future Work

- Common change patterns + autofix?
  - Most autofix approaches are random mutation based
- Translation + Static analysis
  - “press x% when %x is on”
- Any other combinations with MSR?

STAR: Stack-Trace based Automatic crash Reproduction

Sung Kim <hunkim@cse.ust.hk>
http://www.cse.ust.hk/~hunkim
http://www.se.or.kr (Korean Blog)