

Software Quality Analysis

2018-2019

formerly known as
"Research Topics in Software Quality"



VRIJE
UNIVERSITEIT
BRUSSEL

Coen De Roover
cderoove@vub.ac.be
<http://soft.vub.ac.be/~cderoove/>

0. Introduction

Source Code Quality Defects

9/9


0800 Anttan started
 1000 " stopped - anttan ✓

1300 (033) MP - MC ~~1.482147000~~ { 1.2700 9.037847025
 (033) PRO 2 2.130476415 } 9.037846995 connect
 connect 2.130676415

Relays 6-2 in 033 failed special speed test
 in relay " 10,000 test.

Relays changed

1100 Started Cosine Tape (Sine check)
 1525 Started Multi Adder Test.

1545  Relay #70 Panel F
 (moth) in relay.

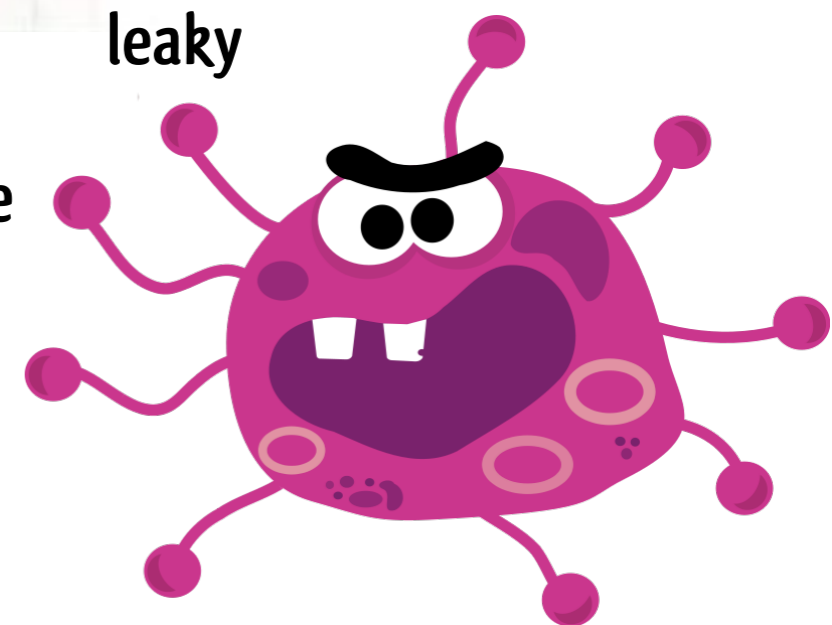
First actual case of bug being found.

1630 Anttan started.
 1700 closed down.

Relay 2145
 Relay 33

At 15:45 on the 9th of September 1947, Grace Murray Hopper records the first computer bug in her log book. The problem was traced to a moth stuck between a relay in the Harvard Mark II.

leaky
 unmaintainable
 buggy



Source Code Quality Defects

2005 Toyota Camry

data flow spaghetti:

> 11.000 global variables

control flow spaghetti:

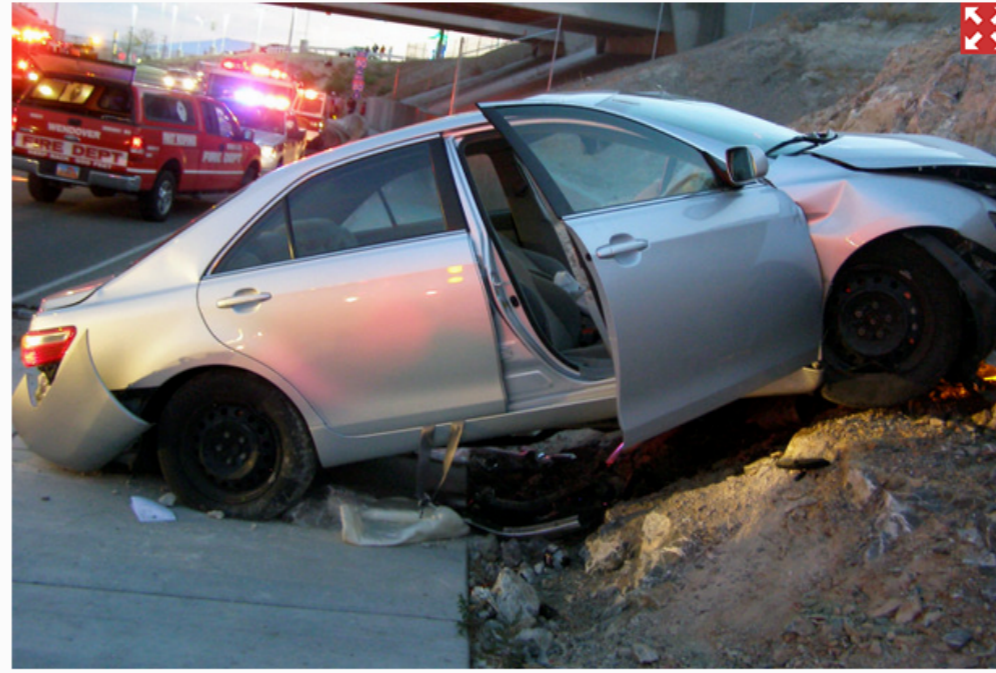
> 100 independent paths through throttle function

67 functions > 50 paths

Toyota to pay \$1.2B settlement in vehicle acceleration lawsuit

By Bob Fredericks and Post Wires

March 19, 2014 | 9:19am



A Toyota Camry that crashed as it exited Interstate 80 in Wendover, Utah, in 2010.

Photo: AP

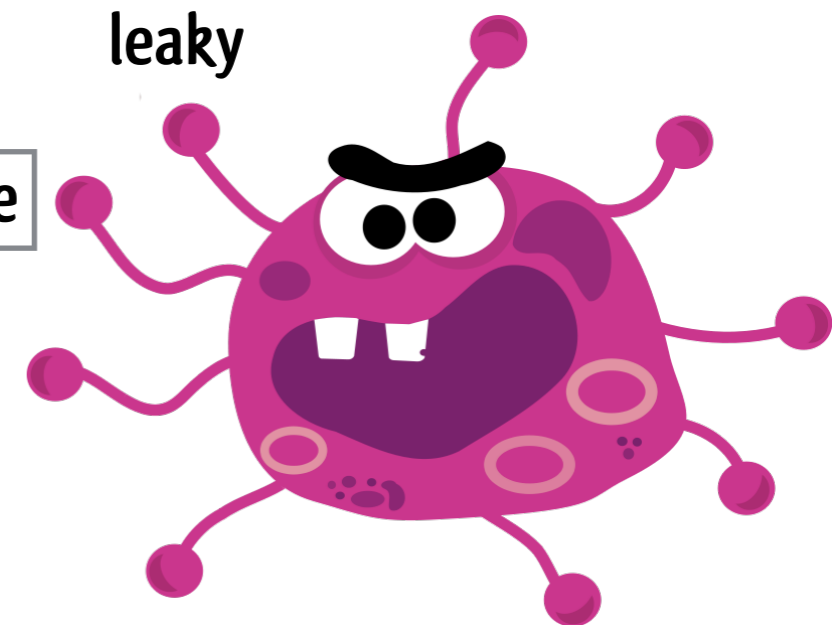
“Some of them are even so complex that they are what is called unmaintainable, which means that if you go in to fix a bug or to make a change, you’re likely to create a new bug in the process.”

[Barr, testimony 15]

unmaintainable

buggy

leaky



Source Code Quality Defects

```
x = Location.getLongitude();
```



```
y = toAttackerURL(x);  
URL.openConnection(y);
```

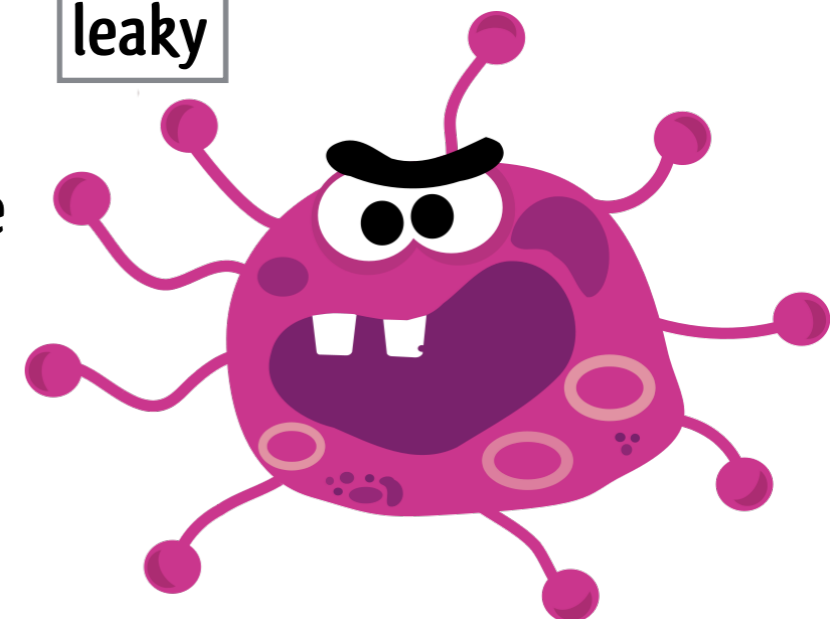


“AndroidLeaks found 57,299 potential privacy leaks in over 7,400 applications, out of which we have manually verified that 2,342 applications leak private data.” [Gibler et al., Trust12]

unmaintainable

buggy

leaky



Research Topics in Software Quality



automated techniques for **evaluating**, **improving**, and **assuring** the **quality of software**

initial focus on **foundations**, each with an example application

- **symbolic execution** for generating tests
- **program transformation** for renovating programs
- **program querying** for enforcing design invariants
- **program analysis** for detecting bugs
- **abstract interpretation** for detecting security leaks
- ...

later focus on how these are evolving to cope with modern software



Program transformation

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TDDConnectFinal	94.7 %	755	42	797
src	100.0 %	372	0	372
com.packtpublishing.tddjava.ch05connect4	100.0 %	372	0	372
Connect4TDD.java	100.0 %	372	0	372
Connect4TDD	100.0 %	372	0	372
Connect4TDD(PrintStream)	100.0 %	37	0	37
checkColumn(int)	100.0 %	19	0	19
checkPositionToInsert(int, int)	100.0 %	17	0	17
checkWinner(int, int)	100.0 %	166	0	166
getCurrentPlayer()	100.0 %	15	0	15
getNumberOfDiscs()	100.0 %	8	0	8
getNumberOfDiscsInColumn(int)	100.0 %	10	0	10
getWinner()	100.0 %	3	0	3
isFinished()	100.0 %	8	0	8
printBoard()	100.0 %	27	0	27
putDiscInColumn(int)	100.0 %	29	0	29
switchPlayer()	100.0 %	13	0	13

“Instruction coverage for a test is computed by running the test and measuring how many instructions of the program are executed.”

requires adding instructions to the program!



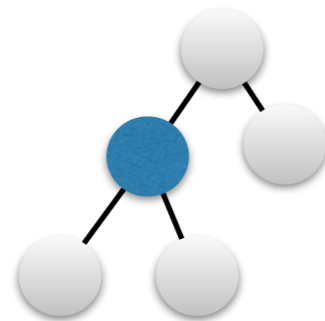
Kiama library for language processing

<https://bitbucket.org/inkeytonik/kiama>

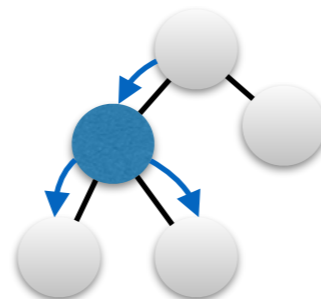


source

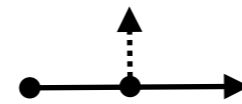
parsing



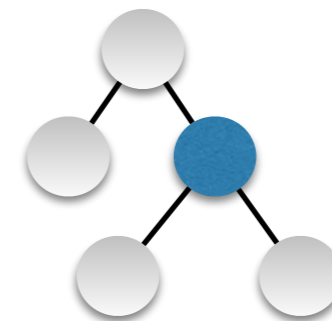
abstract syntax tree



queries



transformation



abstract syntax tree'

pretty
printing



source'

Program querying

```
REPL @ 127.0.0.1:54676 (damp.ekeko)
=> (ekeko* [?inv ?exp]
      (ast :MethodInvocation ?inv)
      (has :expression ?inv ?exp))
#<QueryView baristau.views.queryResult.QueryView@61a89758>
```

conditions on values for ?inv and ?exp

Ekeko Query Results

Query Variables

Mark Results

Query Stats

Table Columns Tree

?exp	?inv
startFigure ...	startFigure.displayBox() ...
endFigure ...	endFigure.connectionInsets() ...
nil	endFigure() ...
figure ...	figure.canConnect() ...
view() ...	view().addToSelection(fAnchorFigure) ...
palette ...	palette.add(createToolButton(IMAGES + "OCONN", "Elbow Connection Tool", tool)) ...
m ...	m.setBounds(x,y,d.width,d.height) ...
nil	repaint() ...

pairs of values satisfying constraints



e.g., lapsed listeners preventing subject from being reclaimed



Ekeko library for program querying
<https://github.com/cderoove/damp.ekeko/>

Program analysis

```
abstract class Animal {
  abstract void makeNoise();
}
```



```
class Dog extends Animal {
  @Override //bark
  void makeNoise() {}
}
```

```
class Horse extends Animal {
  @Override //neigh
  void makeNoise() {}
}
```

```
Animal frosti = new Horse();
frosti.makeNoise();
```

```
(define map
  (lambda (f l)
    (if (pair? l)
        (cons (f (car l))
              (map f (cdr l)))
        '()))))
```



Scheme

```
(define amicalled (lambda () 'yes))
```

```
(map (lambda (x) (x))
     (list amicalled))
```

- Which functions are applied at a call site?
- Which variables will have the same values?
- Which procedures have no observable side effects?
- Which expressions can be executed in parallel?

Examination



no test, but: 3 obligatory written reports on assignments (20% + 20% + 20%)

assigned throughout the year

deadline is start of examination session

1 obligatory oral presentation synthesizing 2 recent publications (40%)

to be selected from a predetermined list

your presence is mandatory during presentations from other students

failing to hand in an assignment or failing to present => ABSENT mark

How to read and present research in software quality

- what is the **motivation** for the work?
 - why is there no trivial solution?
 - what are the shortcomings of the previous solutions?
- what is the **proposed solution**?
 - why is it believed to work?
 - how is it believed to improve upon existing ones?
- how is the **solution evaluated**?
 - how is the solution technically implemented?
 - what are the results and their threats to validity?
- how do the authors and you perceive the **contributions and shortcomings** of the work?
 - what is your analysis of the problem, solution and its evaluation?
 - how solid, novel, applicable are the ideas?
- what is the **message to take away**?
 - why will this work still be relevant in the next decade?
 - are there future directions for this research?
- what techniques surveyed in the course does this **work relate to**?
 - would it be possible to combine the work with techniques from the course?
 - are there other application areas for the work?