Do Fewer, Do Smarter, Do Faster

- Mutation testing in practice

Smart Software Engineering Tools
Apr 25, 2019

Markus Borg
@mrksbrg
mrksbrg.com

RISE Research Institutes of Sweden AB

(CC Flickr: dalbera)

Who is Markus?

- Development engineer, **ABB** 2007-2010
  - Process automation
  - Editor and compiler development
- PhD student, **Lund University** 2010-2015
  - Requirements engineering and testing
  - Traceability, change impact analysis
- Senior researcher, **RISE** 2015-

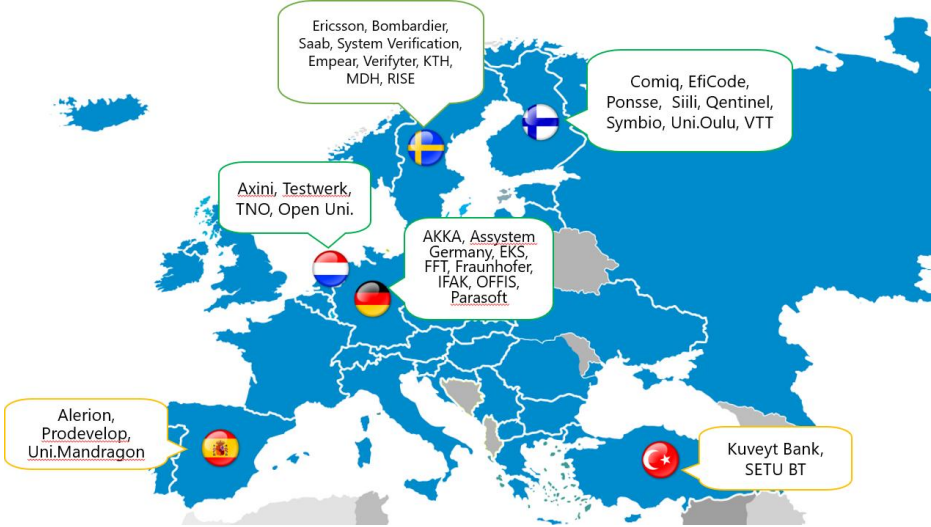
EASE

RI. SE Research Institutes of Sweden

Non-research Markus

- Adjunct lecturer (20%), Lund University
  - Teaching software engineering
- Member of the board, Swedsoft
  - Influence decision makers
  - Write comment letters
  - Facilitate networking

TESTOMAT PROJECT

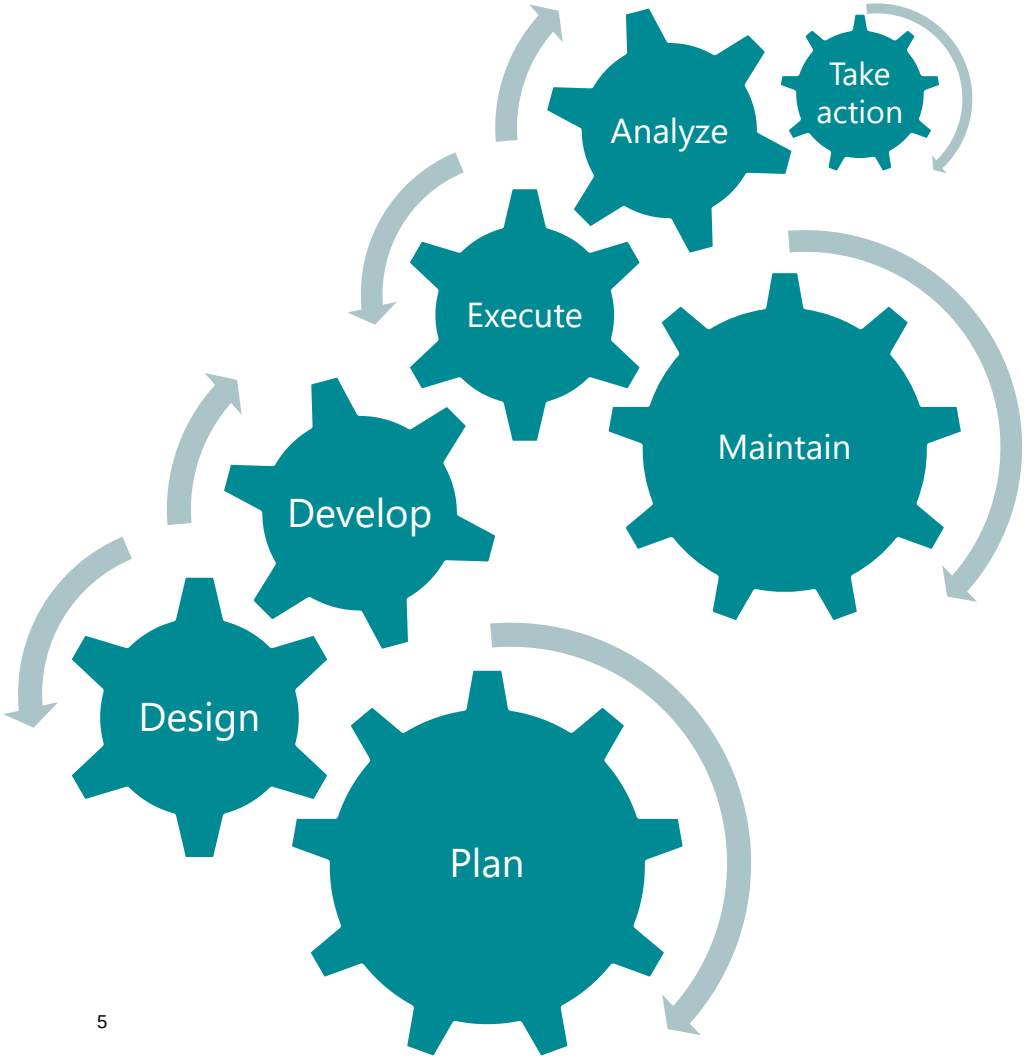


Contemporary dilemma. Modern software teams must optimize for both

- Few bugs
- Ease of change

Project Goal

- Help software teams to increase the development speed without sacrificing quality
- Advance the state-of-the-art in test automation



TESTOMAT PROJECT

Three years
34 partners
€ 21,752,000

Mutation Testing

RI.
SE

Do you trust your test cases?

Ali Parsai

Sten Vercammen

Universiteit Antwerpen

(Staff Sgt. Ryan Callaghan, US Air Force)

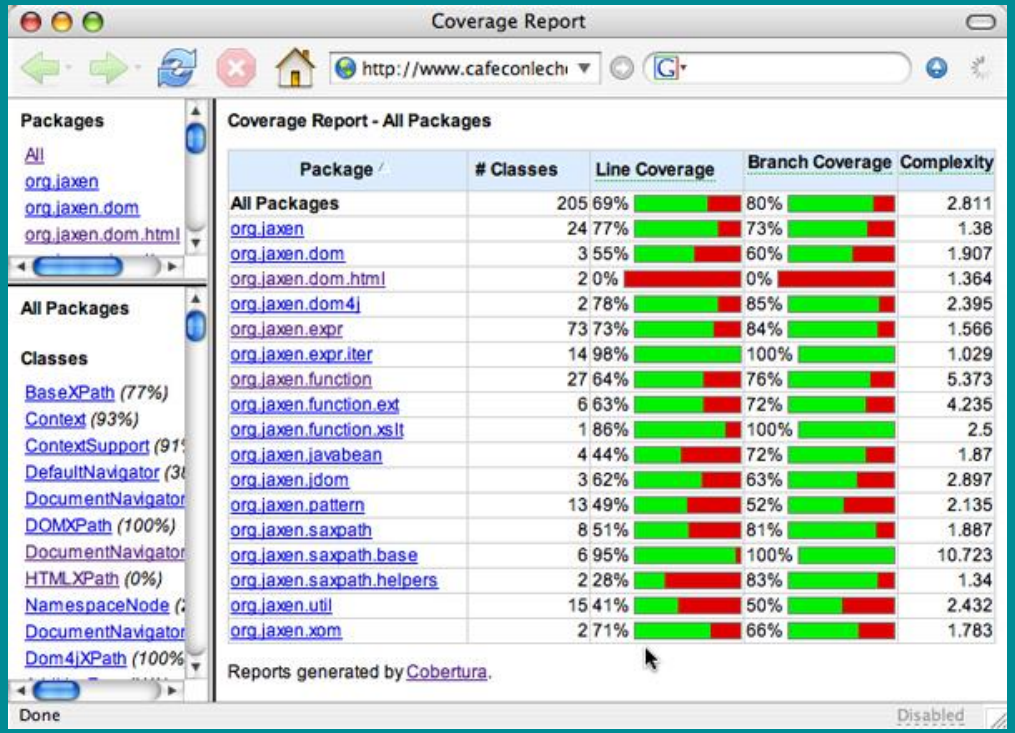
High Quality Test Suite

Fewer Bugs

How do you know that your test suite is good?

RI. SE

Go-to solution: Coverage



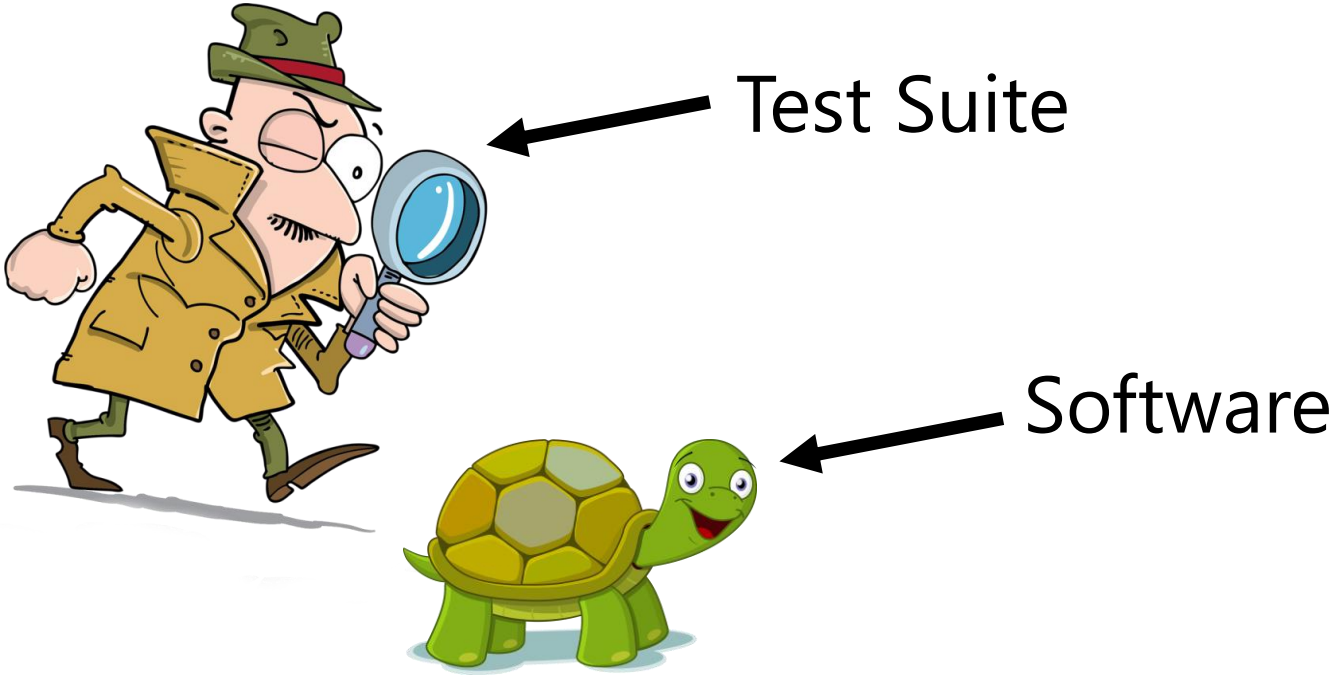
- Shows how test code exercises production code
- Does not reveal how well we tested the production code
  - Necessary, but not sufficient

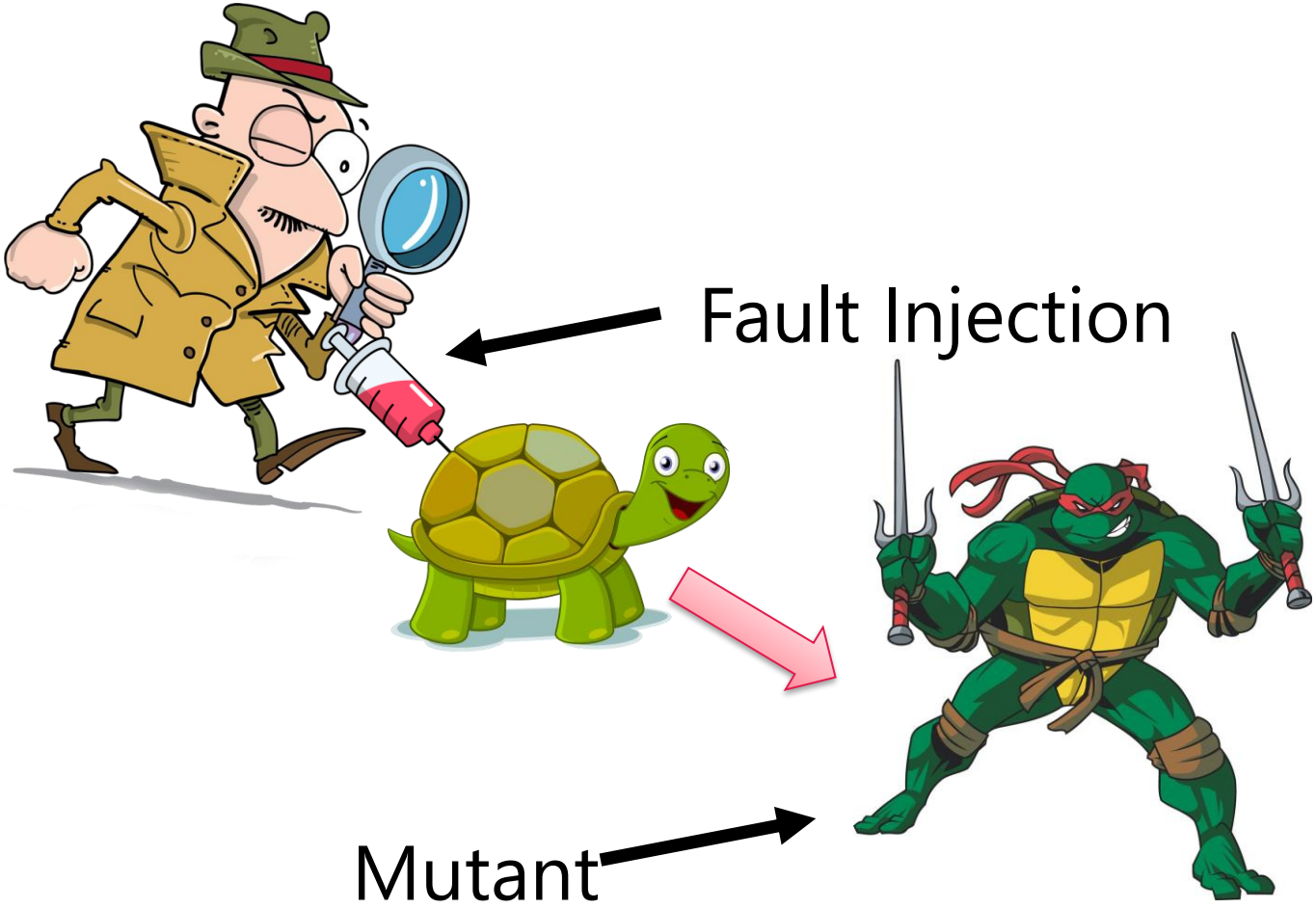
Try removing all assertions!

RI. SE

We need to test our test cases

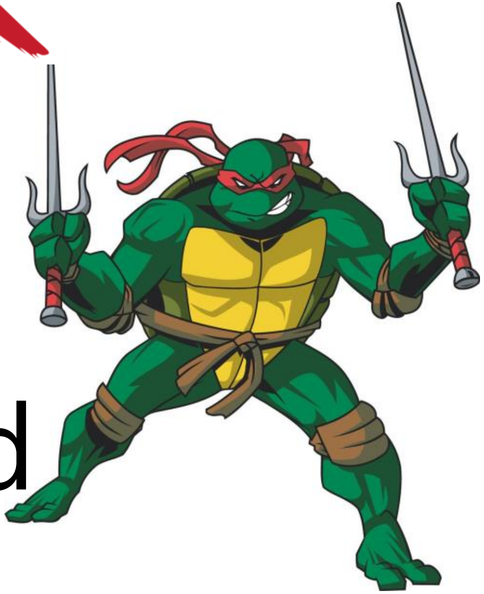
fault injection!


Test Suite
Software


Fault Injection
Mutant

Survived

Killed



Mutation Operator

a+b a-b

a>> a<<b

a==b a!=b

a<b a<=b

a !a

A::b B::b

a(b) a(b,c)

a&b a|b

RI.
SE

Competent programmer hypothesis

- Developers are skilled at programming
- Source code is almost correct
- Most software faults are due to small syntactic errors

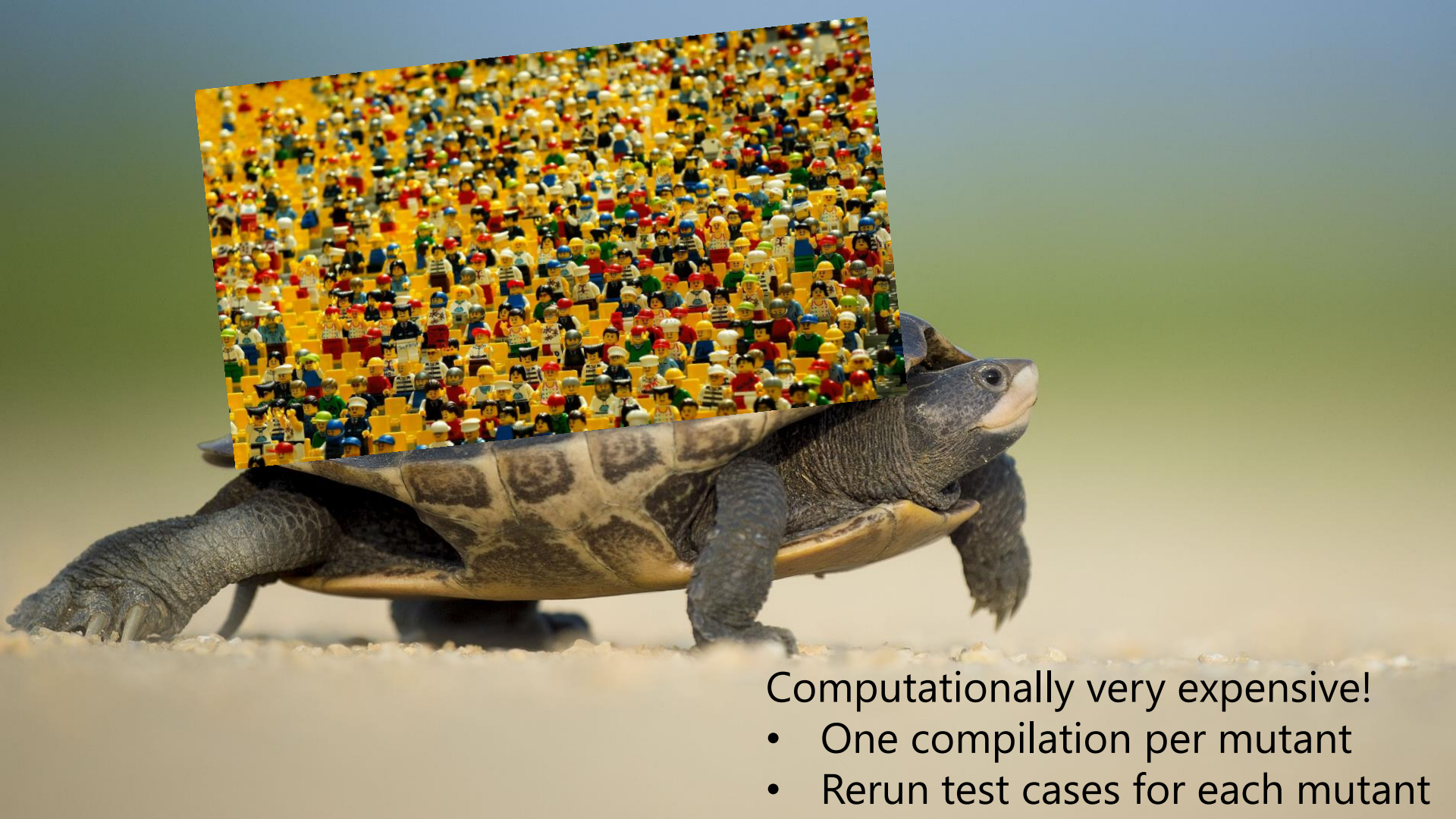
Coupling effect

- Simple faults cascade to form other emergent faults
- Tests that detect small syntactic errors also detect complex issues

Mutation score

$$\frac{\text{Number of killed mutants}}{\text{Total number of mutants}}$$

RI.
SE



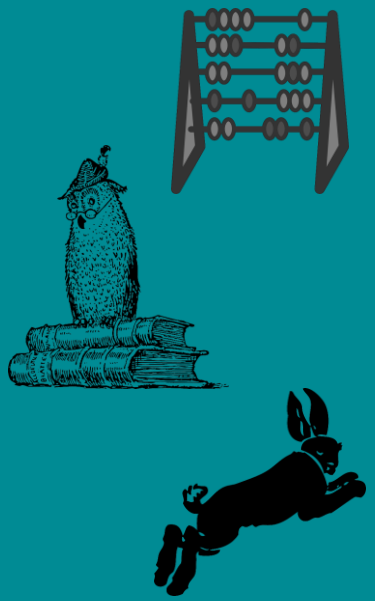
Computationally very expensive!

- One compilation per mutant
- Rerun test cases for each mutant

Do Fewer,
Do Smarter,
Do Faster

RI.
SE

Three key strategies to make it feasible



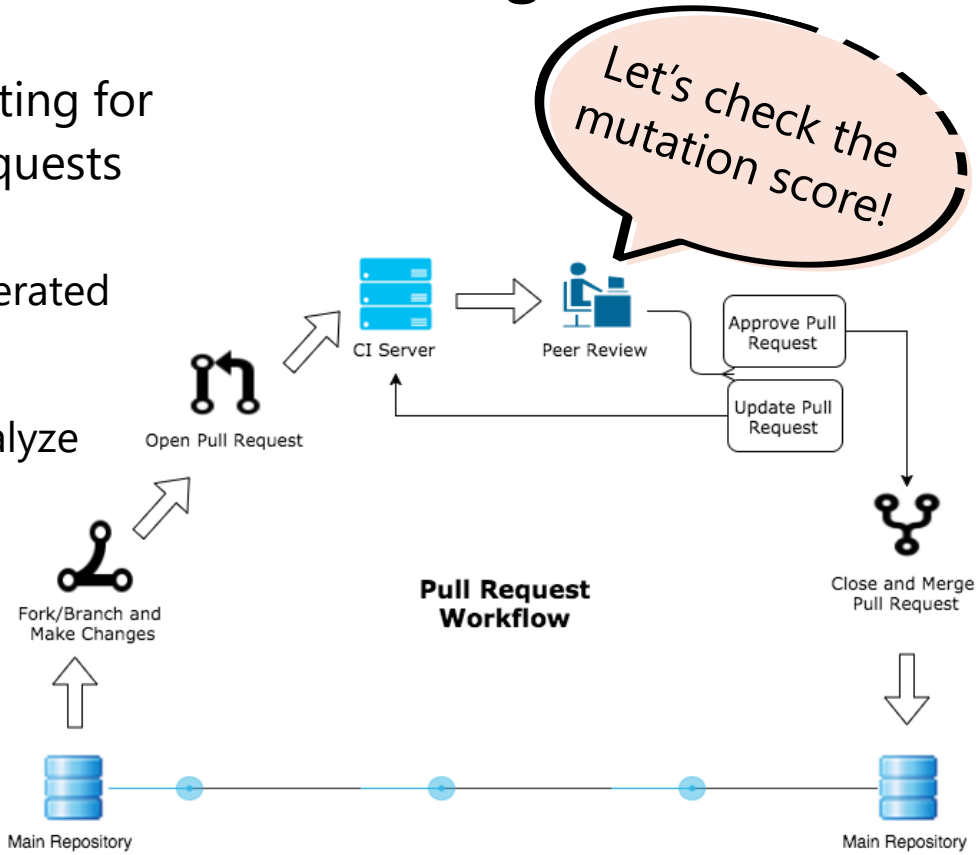
- Do *fewer* - reducing the number of mutants to execute
- Do *smarter* – run mutation testing in less naïve setups
- Do *faster* – optimize execution of mutation testing steps

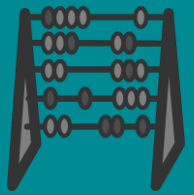
Change-based mutation testing

- Run mutation testing for individual pull requests

Fewer mutants generated

Fewer results to analyze





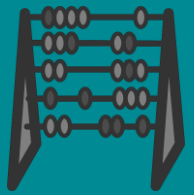
Do fewer

Change-based mutation testing

- Little Darwin (OSS)
  - https://littledarwin.parsai.net/

Ali Parsai

Universiteit Antwerpen



Do fewer

- C# mutation testing tool
  - By Mille Boström, not yet OSS

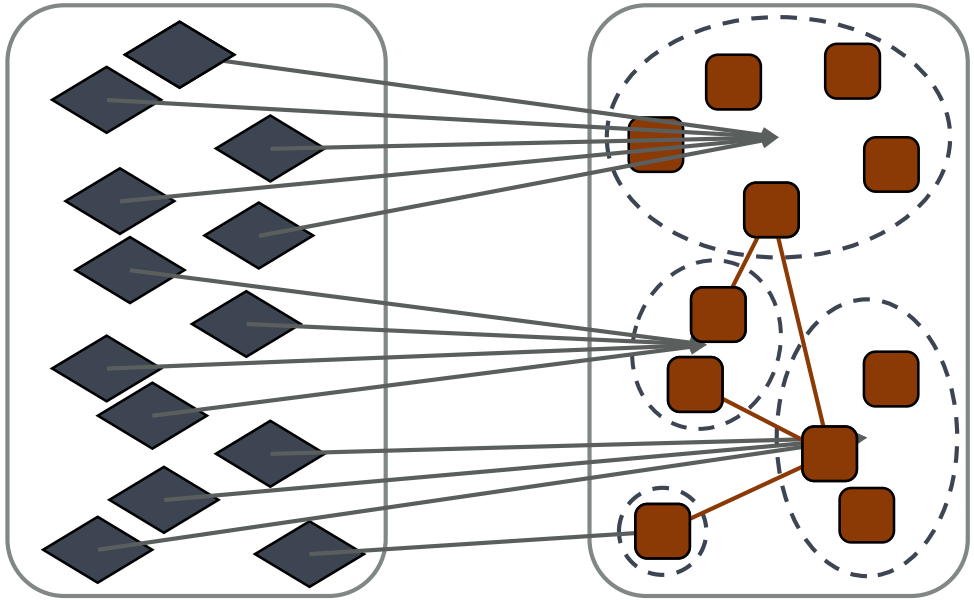
SYSTEM VERIFICATION

RI. SE

Focal methods = foci of test cases

- Run only test cases that actually test the code where you introduced the mutant

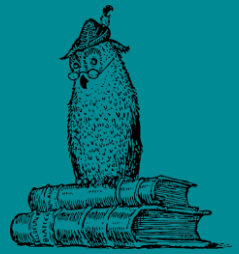
faster kills



Sten Vercammen

Universiteit Antwerpen

<APACHE ANT>

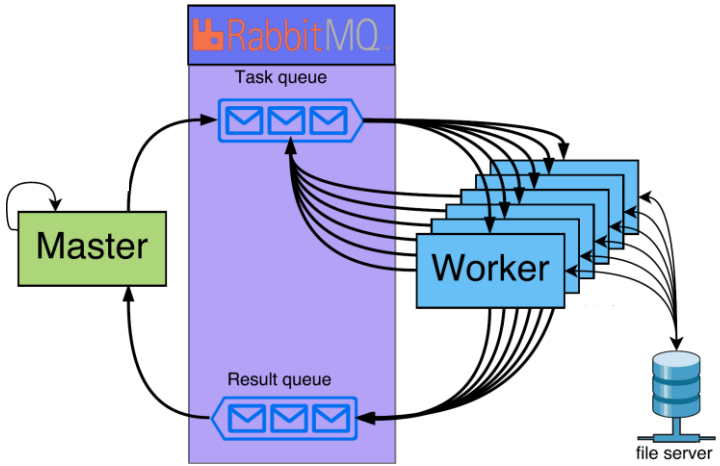


Do smarter

RI. SE

Toward mutation testing in the cloud

- Parallelize the work in DiMuTesTas tool (OSS)
  - https://github.com/Sten-Vercammen/DiMuTesTas
- Investigate speed-up and bottlenecks



Sten Vercammen

Universiteit Antwerpen

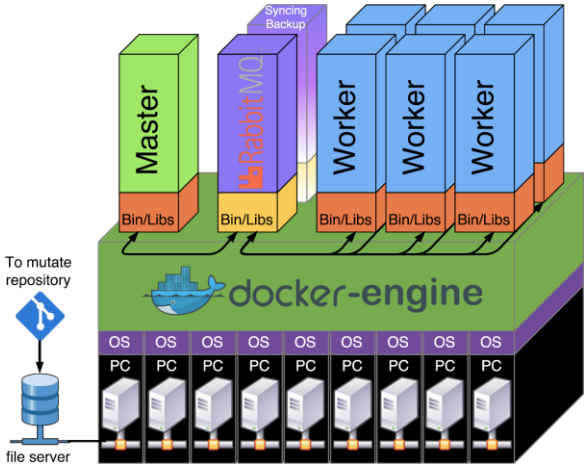
Do smarter

RI. SE

Lessons learned

- Time to compile time vs. time to test varies greatly

Speed-ups of 12x-13x with 16 workers



inventivedesigners

health CONNECT
ICT EXPERTISE THAT CARES

intris
WISETECH GLOBAL GROUP

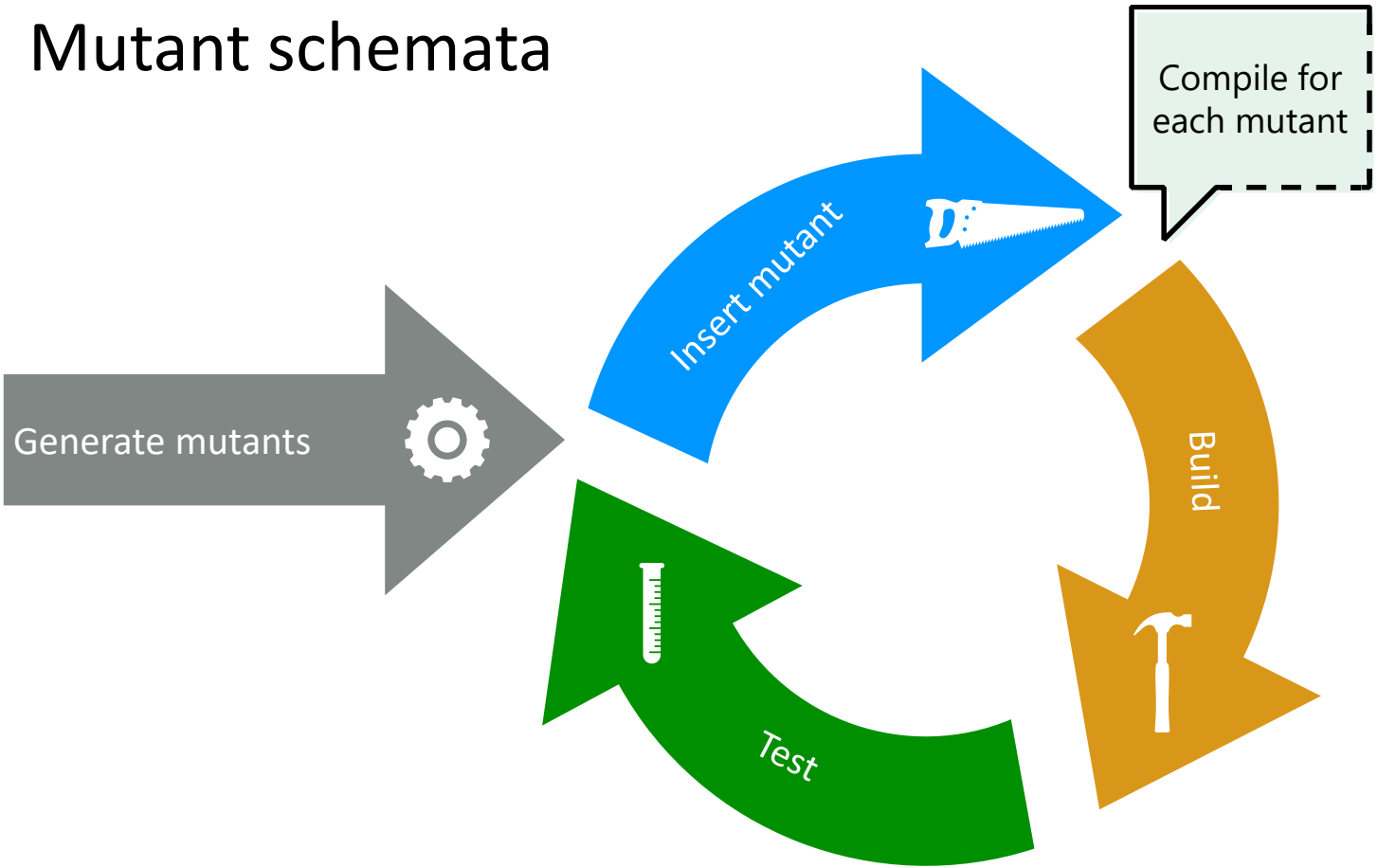
Sten Vercammen

Universiteit Antwerpen

Do smarter

RI. SE

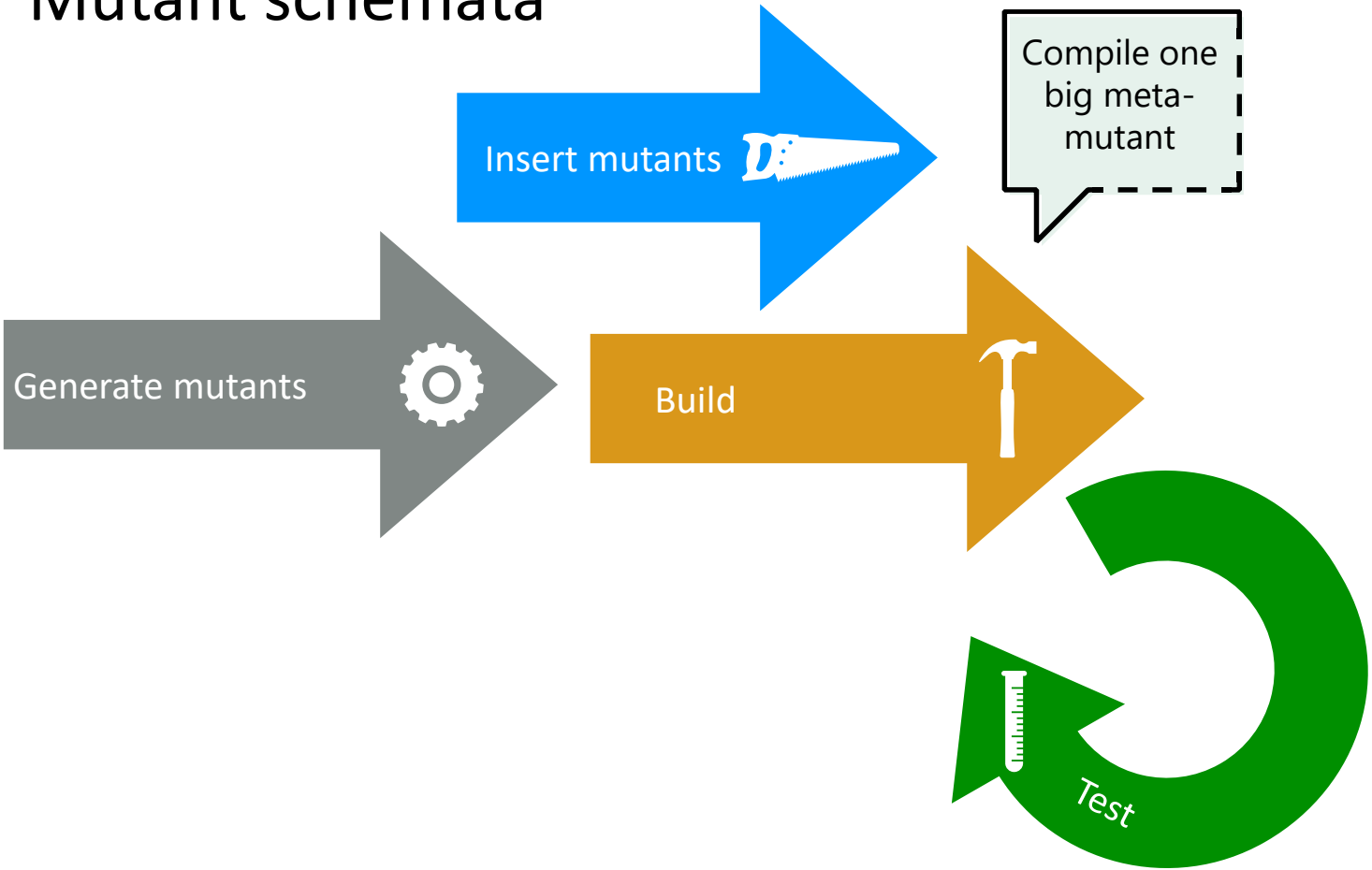
Mutant schemata



Do faster

26

Mutant schemata



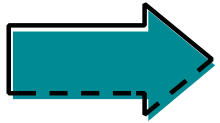
Do faster

RI. SE

Mutant schemata

- Big meta-mutants replace numerous seperate mutants
  - Add extra parameters to selectively activate mutants

a + b > c



arithOp(a, b, X) > c

```
int arithOp(int op1, int op2, int location) {
  switch(variant(location)) {
    case aoADD: return op1 + op2;
    case aoSUB: return op1 - op2;
    case aoMULT: return op1 * op2;
    …
  }
}
```

Reduced compilation time

Do faster

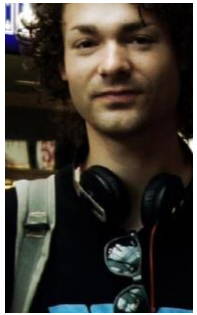
RI. SE

Mutant schemata

- Ongoing work with SAAB and University of Antwerp
- DexTool (OSS) C/C++
  - https://github.com/joakim-brannstrom/dextool

Do faster

Sten Vercammen



Christoffer Nylén

SAAB TECHNOLOGIES SAAB

Universiteit Antwerpen

RI. SE

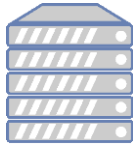
Wrap-up

Combined approaches => tech transfer!

Do Fewer

Change-based mutations

Affected mutants

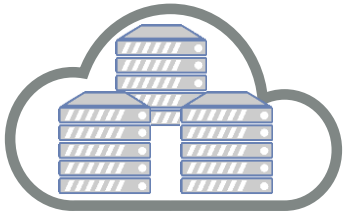


CI server

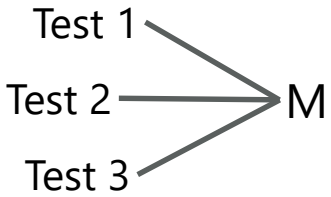
Prune equivalent, invalid, unreachable

Do Smarter

Distribution

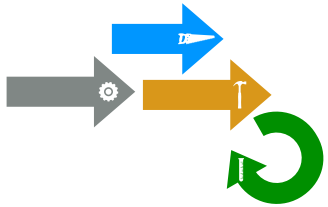


Test prioritization / Focal methods

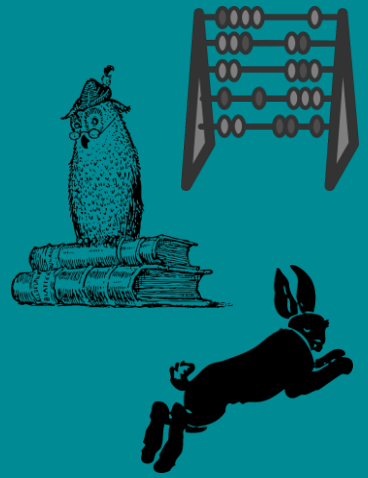


Do Faster

Mutant schemata



Ready for industry!



Do fewer,

Do smarter,

Do faster,

RI. SE

Our offer: Tools, support

- Little Darwin, DiMuTesTas (Java)
- DexTool (C/C++)
- Change-based Mutation (C#)

Our offer: Tools, support… and Sten!

- Little Darwin, DiMuTesTas (Java)
- DexTool (C/C++)
- Change-based Mutation (C#)
- Sten Vercammen
  - Joint PhD program Antwerp+Lund
  - 6 months in Sweden (2020 or later)

Universiteit Antwerpen

LUND UNIVERSITY

RI. SE

Our offer:

- Little Darwin, DiMuTesTas (Java)
- DexTool (C/C++)
- Change-based Mutation (C#)
- Sten Vercammen

RI.
SE
Markus Borg
@mrksbrg
mrksbrg.com
RISE Research Institutes of Sweden AB