Test Selection with Moose In Industry: Impact of Granularity

Vincent Blondeau

25 – August – 16

INVENTEURS DU MONDE NUMÉRIQUE

RMod

Université de Lille 1 SCIENCES ET TECHNOLOGIES

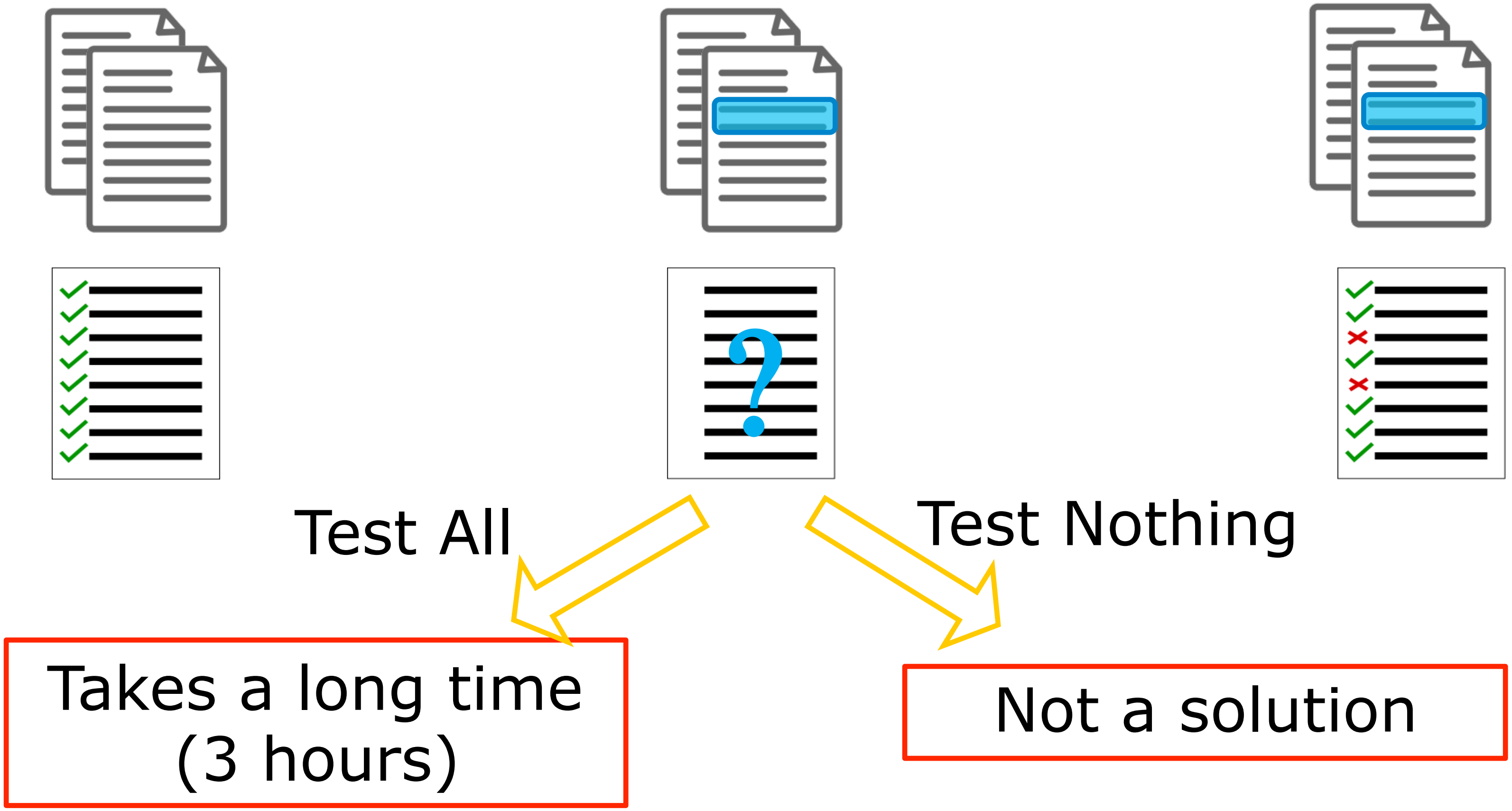
cnrs dépasser les frontières

worldline e-payment services

Context

- Industrial PhD in a major international IT company
  - +7 000 employees
  - 17 countries
  - Problems from the field

Test Case Selection After a Change



Test All

Test Nothing

Takes a long time
(3 hours)

Not a solution

Test Case Selection After a Change

- **Question**
  - Does test selection modify developers’ habits and enhance software quality?
- **Goal**
  - Save time and improve quality
  - Select tests to relaunch after a change in the source code
  - Any kind of test: End to end, performance…
- **Comparison of approaches on real changes**
  - Call graph analysis by static approach
  - Comparison with dynamic approach

Two approaches

- **Static Approach**
  - Create a model of the system with Moose
  - Navigate the call graph from a changed source code to find the tests

Properties

- Allows to find multiple tracks to the changed source
- No test execution

Two approaches

- **Dynamic Approach**
  - Execute the tests
    - Map the tests to the covered code
  - Relaunch the tests related to changed source code

Properties

- Dependent to the test data
- The tests have to be executed

Issue Classification

- Problems in test selection approaches arise when there is a break in the dependency graph representing the system.
  - *Third-party breaks*
  - *Multi-program breaks*
  - *Dynamic breaks*
  - *Polymorphism breaks*

Experiment

- **Hypothesis**: Dynamic approach is the oracle
  - With some flaws:
    - Does not work on failing or in error tests
    - Requires time to be performed

- **Approach**
  - Compare influence of real source code changes
  - Simulate code change on several existing projects

Experiment

- Consider real commits by mining repositories
  - Weight each covered method by the number of commits
  - Group covered methods in commits
    - Considered real method commit groups

Projects: Metrics

Metric	P1	P2	P3
KLOC Core	447	716	302
# Green Tests	5 323	168	3 035
# Total Methods	9 808	56 661	45 671
# Methods Covered	4720 (48%)	3 261 (6%)	8 143 (18%)
#Commits	2 217	467	2 115
Avg Methods/Commit	24	129	37
Avg Files/Commit	7	18	17

Metrics

- Number of selected tests
  - Ratio of the total test suite to relaunch
- Precision
  - How many selected tests are relevant?
- Recall
  - How many relevant tests are selected?

Weighting of methods with the number of commits

	#Selected Tests			Precision			Recall	
	1 Meth.		Weig.	1 Meth.		Weig.	1 Meth.		Weig.
P1	3%	→	3%	43%	↘	42%	91%	↗	92%
P2	0.8%	↗	1%	61%	↘	59%	64%	↘	62%
P3	2%	→	2%	34%	↘	33%	41%	↘	39%

Methods grouped in commits

	#Selected Tests		Precision		Recall	
	1 Meth.	Commit	1 Meth.	Commit	1 Meth.	Commit
P1	3%	↗ 4%	43%	↗ 55%	91%	↘ 81%
P2	0.8%	↗ 3%	61%	↘ 45%	64%	↘ 45%
P3	2%	↗ 6%	34%	↗ 49%	41%	↗ 56%

Conclusion

- Considering commits instead of individual methods tends to worsen the results
- Impact on projects is different
- Low ratio of selected tests, so still acceptable

Future steps

- Better understand how tests are used by developers
- Provide a tool for developers to select tests

Thanks!

25-August-16