Profes 2012

Improving Unfamiliar Code with Unit-Tests: An Empirical Investigation on Tool-Supported and Human-Based Testing

Dietmar Winkler[1] Martina Schmidt[1] Rudolf Ramler[2] Stefan Biffl[1]

[1]Vienna University of Technology
Institute of Software Technology, Christian Doppler Laboratory
“Software Engineering Integration for Flexible Automation Systems”

[2]Software Competence Center Hagenberg

dietmar.winkler@tuwien.ac.at
http://qse.ifs.tuwien.ac.at/~winkler

Motivation & Goals

Motivation

- Software testing is a well-established quality assurance approach to introduce unit tests on different levels during software development projects.
- Existing software solutions often suffer from a lack of unit tests due time restrictions and/or resource limitations.
- A lack of unit tests can hinder effective and efficient maintenance processes.

Goals:

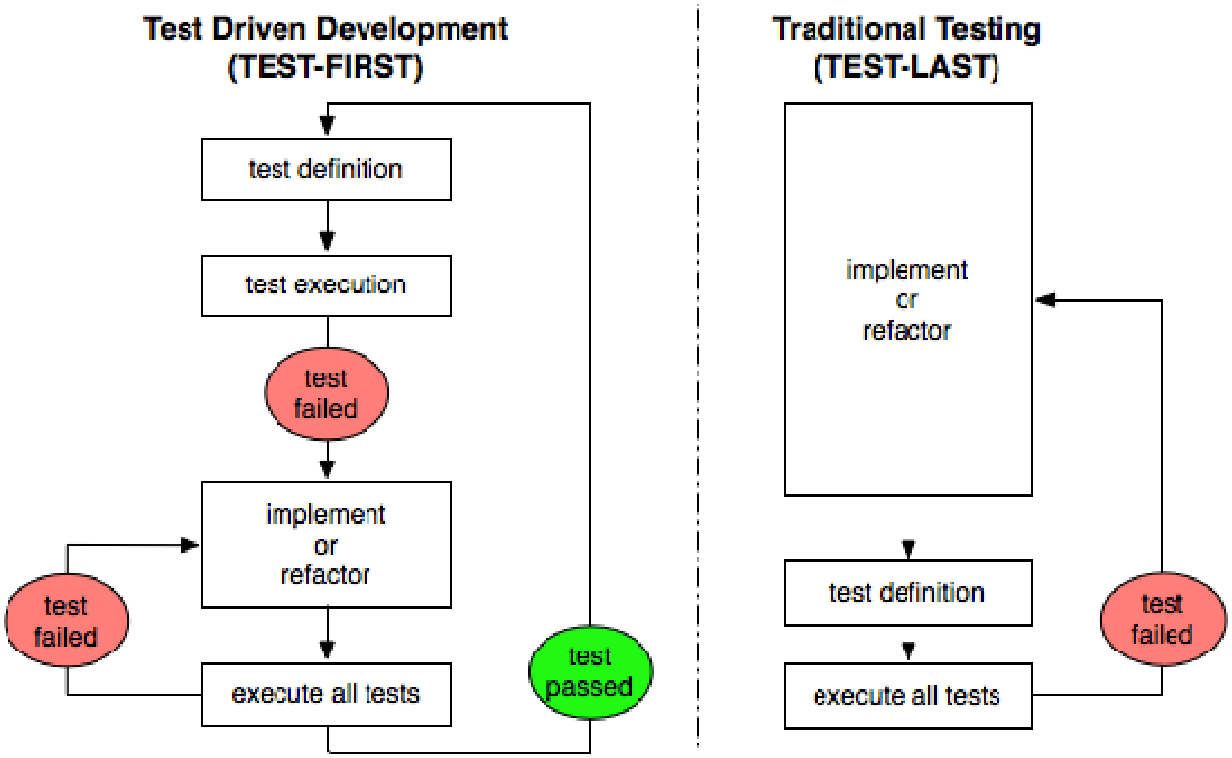
- Introducing unit tests after deployment is a promising approach for
  (a) Enabling systematic and automation-supported tests after deployment.
  (b) Increasing product quality significantly.

Key research questions focus on:

- How to introduce test cases in “old” and even “unknown code”? Manually by experts? Supported by tools?

Test-First & Test-Last Test Strategies

- Test-First Development (based on agile concepts)
  - Defining test cases prior to software construction.
- Test-Last Development (traditional software processes)
  - Writing/execution tests after the construction phase.



Human-Based & Tool-Supported Testing

- Changing product requirements, enhancements, and evolution of software products could require testing after deployment.
- New and missing test cases need to be (re-)written to ensure proper maintenance → “Test Last Approach”

Strategies to introduce test cases after deployment

- Human-Based Test Case Construction (manually)
  - Introducing test cases manually.
  - Requires deep understanding of requirements and source code.
  - Additional effort when creating test cases.

- Tool-Supported Random Test Case Generation (automation supported)
  - Automated generation of test cases.
  - Based on specification, models, or source code.
  - Additional effort required when integrating tests.

Research Questions & Experiment Setup

Research Questions focus on

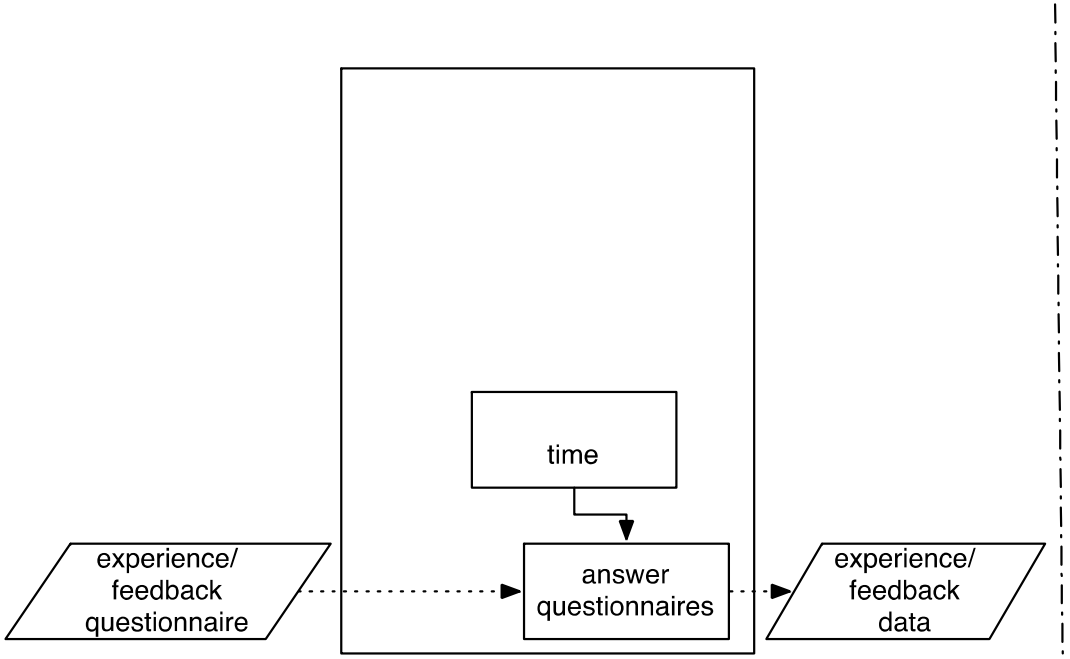
- Defect Detection Effectiveness (EFF)
- False Positives (FP)
- Method Coverage (MC)

Controlled Experiment with seeded defects.

- Subjects: 48 human participants (master students with software engineering & testing background) vs. Randoop.
- Time duration: 60 min for human participants and 2 min for Randoop.
- Study Material consists of Java Collection Classes
  - Software package with 2800 LOCs, 34 interfaces and classes, 164 methods.
  - Javadoc API and class files were provided to force black box testing.
  - 35 seeded defects with 4 defect classifications: algorithm, assignment, checking, and data defects.
  - Supporting material: experience and feedback questionnaire.

Experiment Process

- Study preparation: study material, Randoop configuration, briefing.
- Study execution:
  Session 1 (human-based test case construction) and
  Session 2 (tool-supported test case generation with Randoop).



- Data submission and evaluation.

Limitiations

Internal validity

- Experts reviewed the material and experiment package (Reuse of proven experiment package)
- Avoidance of communication between participants during the study execution.
- Experience questionnaire to capture the skills of the participants.
- Classroom setting to monitor and control study variables.

External validity

- Limitation of human-based testing effort(1 hour of test case generation)
- Well-known study objects to avoid domain-specific interpretation problems.
- Participants are semi-professionals in the field of software testing.

Construct validity

- The study is based on related work and previous experiments and addresses common variables in empirical studies.

Conclusion validity

- Application of Statistical Testing

Results: Effort and Delivered Test Cases

Effort

- The effort does not include the individual preparation duration (i.e., 15 min briefing) and the tool configuration effort (i.e., 2 hours).

Test Strategy	Study Effort [min]				
	No.	Min.	Max.	Mean	SD
Randoop	1	2 min	2 min	2 min	0 min
Participants	48	52 min	68 min	59 min	2 min

Reported/Generated Test Cases

- Tool-supported test case generation delivered far more test cases than humans-based testing.

Test Strategy	Delivered Test Cases				
	No.	Min.	Max.	Mean	SD
Randoop	1	5368	5368	5368	0
Participants	48	1	92	27.1	21.23

Results: Defect Detection Effectiveness

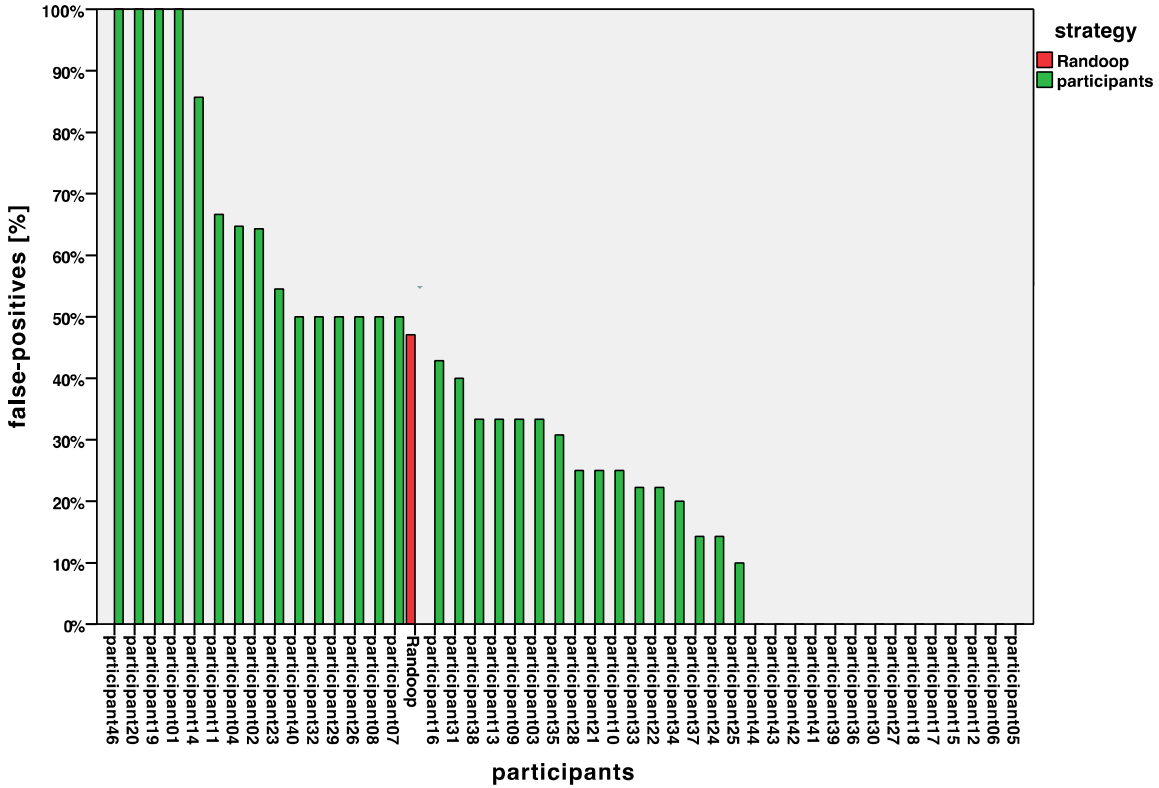
- Defect Detection Capability
- Expectation: Advantages for Randoop.
- Effectiveness: Share of identified defects and seeded defects.
- Results:
  - No significant differences (p: 0.082(-)) for all defect classes.
  - Significant differences for algorithm (p-value: 0.041(s)) and checking (p-value: 0.041(s)) defects.

yp

	Identified Defects (Matched Defects)		Effectiveness [%]	
	Randoop	Participants	Randoop	Participants
Minimum	9	0	25.7%	0%
Maximum	9	**9**	25.7%	**25.7%**
Mean	**9**	3.7	**25.7**%	10.6%
SD	0	2.68	0.0%	7.66

Results: False Positives (FP)

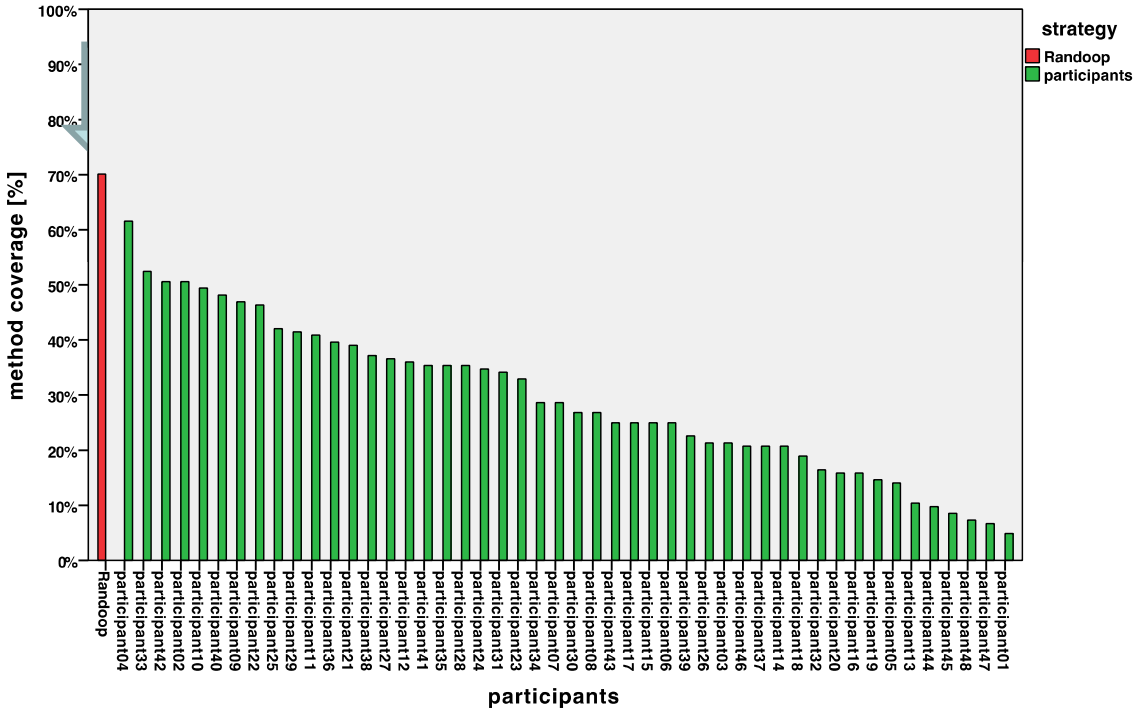
- Expectations: Tool-supported testing reports more false positives than human-based testing.
- No significant differences
  - On average participants (30.4%) deliver a fewer number of false positives than Randoop (47.1%).



- Possible explanation
  - Participants have additional knowledge (i.e., context, requirements and design specification).
  - Participants with 0% FP either had enough experience to avoid them or did not report many tests at all.
  - Participants with 100% FP did not write many proper tests.

Results: Method Coverage

- Expectations: Tool-supported testing achieves a higher method coverage than human-based testing.
- Significant advantage for Randoop (p-value: 0.041(s)).
- Randoop enables testing of all public classes; time limitations for human participants.



Test Strategy	No.	Number of Covered Methods Min.	Max.	Mean	SD	Method Coverage [%] Min	Max	Mean	SD	P-Value
Randoop	1	115	115	115	0.0	70	70	70	0.0	0.041(s)
Participants	48	8	101	48.1	22.89	5	62	29.3	13.96	

Summary and Further Work

Summary

- Existing software solutions often suffer from a lack of unit tests due time restrictions and/or resource limitations.
- The question is whether tests cases can be introduced into old and unknown code manually by experts or automation supported by tools.
- Main results:
  - Different testing approaches support various defect classes.
  - Application of domain knowledge and context information by humans.
  - High number of test cases by Randoop.
  - A mix of the testing strategies should be chosen in order to receive the unified benefits of both.

Further Work

- More detailed investigation of participants test cases and test case quality.
- Investigation of tester qualification focusing on written test cases and defect detection capability.
- Alternative configurations of Randoop (Optimization).

Thank you ...

Improving Unfamiliar Code with Unit-Tests: An Empirical Investigation on Tool-Supported and Human-Based Testing

Dietmar Winkler[1], Martina Schmidt[1], Rudolf Ramler[2], Stefan Biffl[1]

[1]Vienna University of Technology
Institute of Software Technology, Christian Doppler Laboratory
“Software Engineering Integration for Flexible Automation Systems” (CDL-Flex)

[2]Software Competence Center Hagenberg

Dietmar.Winkler@tuwien.ac.at