An Empirical Investigation into the Impact of Refactoring on Regression Testing

Napol Rachatasumrit and Miryung Kim
The University of Texas at Austin

Motivation

- It is believed that refactoring improves software quality and maintainability
- Refactoring has the risk of functionality regression and increased testing cost
- The impact of refactoring edits on regression tests has not been investigated using version history.

Study Findings

- We use **refactoring reconstruction** analysis and **change impact analysis** in tandem.
  - Only 22% of refactored code is tested by existing regression tests.
  - While refactoring edits constitute only 8% of changes, 38% of affected tests are relevant to refactorings.
  - Refactoring edits appear in almost half of the failed test case execution.

Outline

- Motivation & Related Work
- Study Approach Overview
- Research Questions and Results
- Limitations
- Conclusions

Conventional Wisdom

- Refactoring improves software quality and maintainability
- A lack of refactoring incurs ***technical debt***. [Cunningham, Lehman]
- Refactor ***mercilessly*** [Beck, eXtreme Programming]

Refactoring Realities

- The number of bug reports increases after refactorings [Weißgerber & Diehl, Kim et al.]
- Refactoring tools are buggy [Daniel et al.]
- Programmers do not leverage refactoring tools effectively [Murphy-Hill et al. Vakilian et al.]
- Refactoring comes with a risk of introducing subtle bugs and functionality regression [Kim et al.]

Outline

- Motivation
- **Study Approach Overview**
- Research Questions and Results
- Limitations
- Related Work
- Conclusions

Approach Overview

1 **RefFinder: Refactoring Reconstruction**
[ICSM 2010, Prete et al.]

Identify Refactoring Edits

2 **FaultTracer: Change Impact Analysis**
[ICSM 2011, Zhang et al.]

Identify Change Impact on Regression Tests

3 **Investigate Refactoring Change Impact on Tests**

Step 1. Reconstruction of Refactoring Edits

RefFinder: Refactoring Reconstruction
[ICSM 2010, Prete et al.]

Identify edits that fit the program structure before and after each refactoring type

Inferred Refactoring Edits: Type and Location
move_field("color", "PieChart", "Chart")
pull_up_field("color", "PieChart", "Chart")
collapse_hierarchy("Chart", "PieChart")
introduce_explaining_var("val","EXPR…" , "get()")

Step 1. Reconstruction of Refactoring Edits

We use our **logical program differencing framework,** LSdiff [ICSE 2009, Kim & Notkin] to compute change facts at the level of code elements, control and data dependences, etc.

Original Fact-Base
past_subtype("Chart","PieChart"), deleted_subtype("Chart","PieChart")
deleted_field("PieChart.color", "color", "PieChart"),
added_field("Chart.color", "color", "Chart")
deleted_access("PieChart.color", "Chart.draw"), added_access("Chart.color", "Chart.draw"). . .

Step 1. Reconstruction of Refactoring Edits

We encode each refactoring type in a **template logic rule**.

Refactoring Reconstruction Rules
1. collapse hierarchy: A superclass and its subclass is not very different. Merge them together.
(deleted_subtype(t1,t2) $\wedge$ (pull_up_field(f,t2,t1) $\vee$ pull_up_method(m,t2,t1))) $\vee$ (past_subtype(t1,t2) $\wedge$ deleted_type(t1,n,p) $\wedge$ (push_down_field(f,t1,t2) $\vee$ push_down_method(m,t1,t2)))=> collapse_hierarchy(t1,t2)
2. pull up method: A method is moved from a class to its superclass.
move_method(f, t, t1) ^ past_subtype(t1, t)=>pull_up_method(f, t, t1)

Step 1. Reconstruction of Refactoring Edits

RefFinder converts the antecedent of a rule to a logic query and **invokes the query** on the change-fact database.

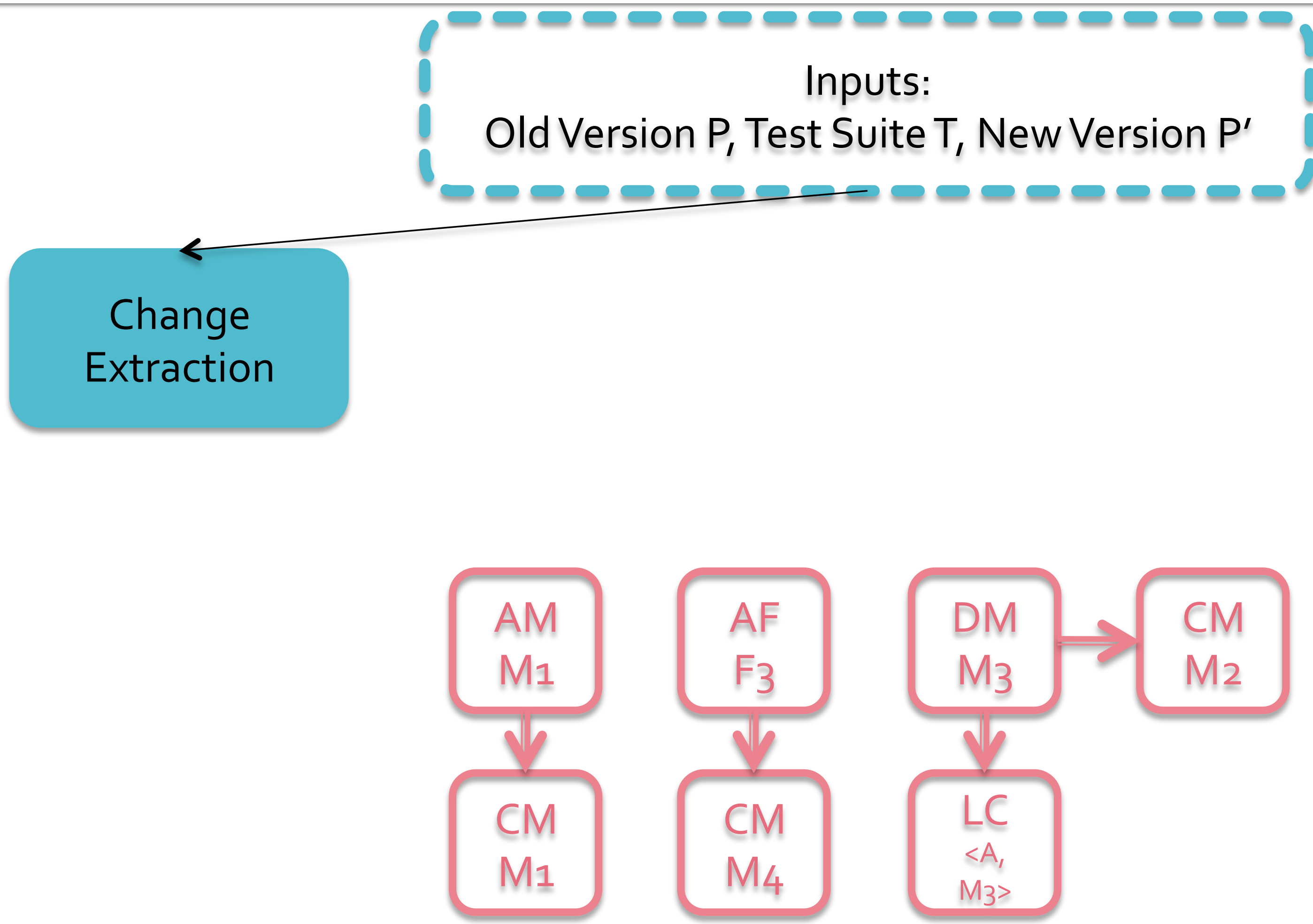
Logic-Query Invocation	Added Facts
deleted_field(f1, f, t1) ^ added_field(f2, f, t2) ^ deleted _access(f1, m1) ^ added_access(f2, m1) ?	+ move_field("color", "PieChart", "Chart")
move_field(f, t1, t2) ^ past_subtype(t2, t1) ?	+ pull_up_field("color", "PieChart", "Chart")
invoking a collapse hierarchy query. . .	+ collapse_hierarchy("Chart", "PieChart")

Step 2. Fault Tracer Change Impact Analysis

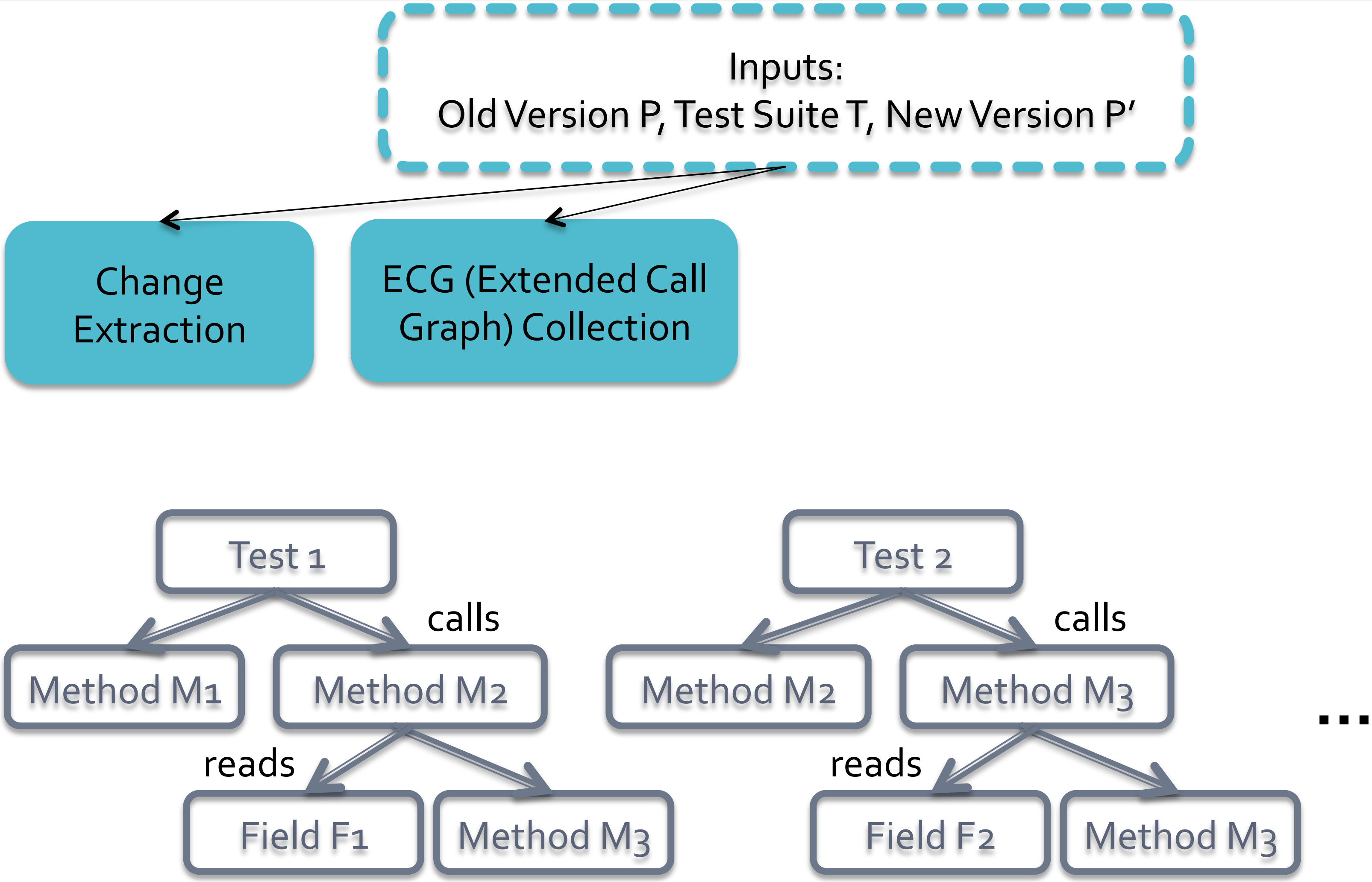
FaultTracer: Change Impact Analysis
[ICSM 2011, Zhang et al.]

affected tests—*a set of regression tests relevant to atomic changes*
affecting changes—*a subset of atomic changes relevant to each affected test*

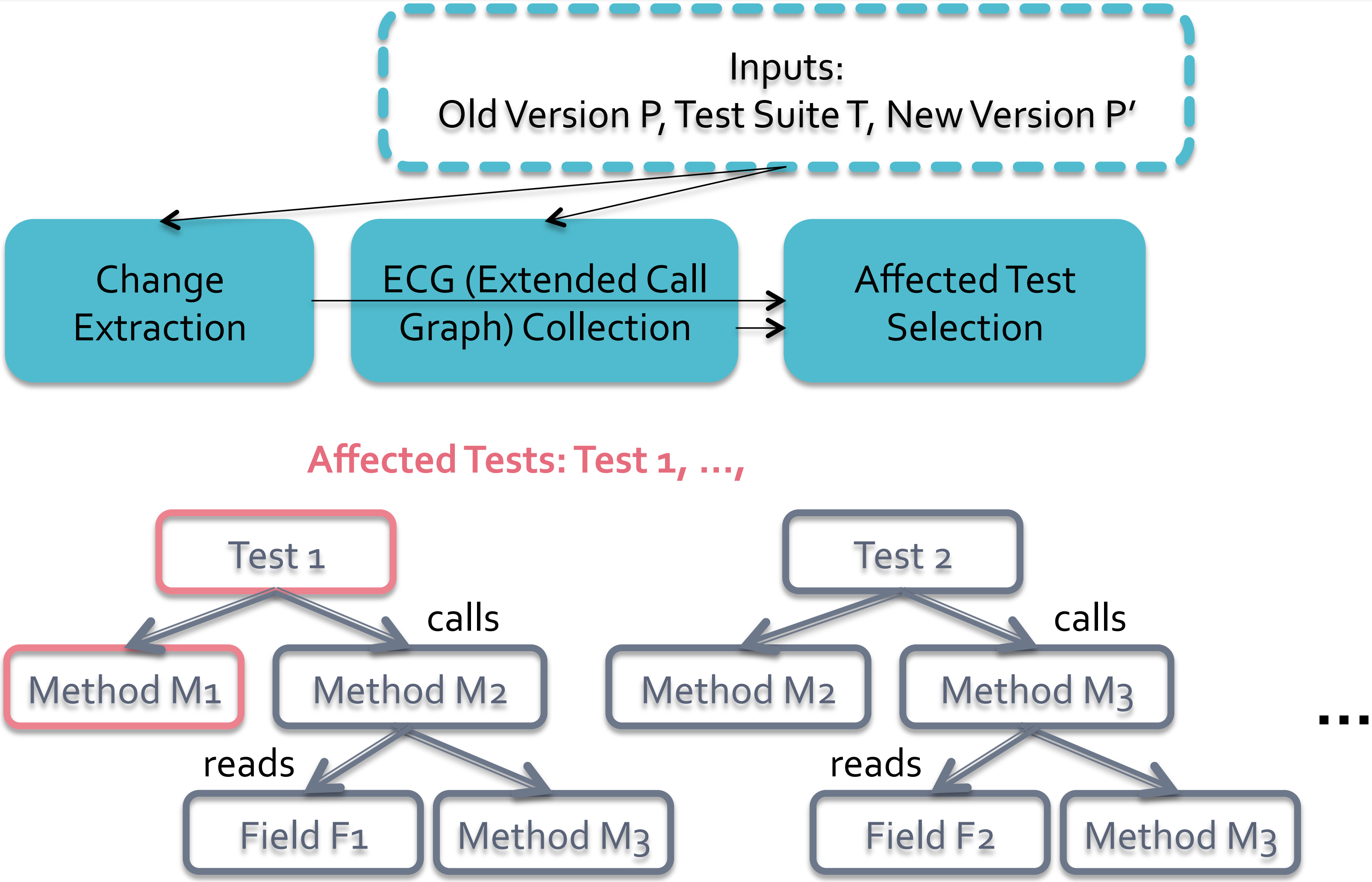
Step 2. Fault Tracer Change Impact Analysis



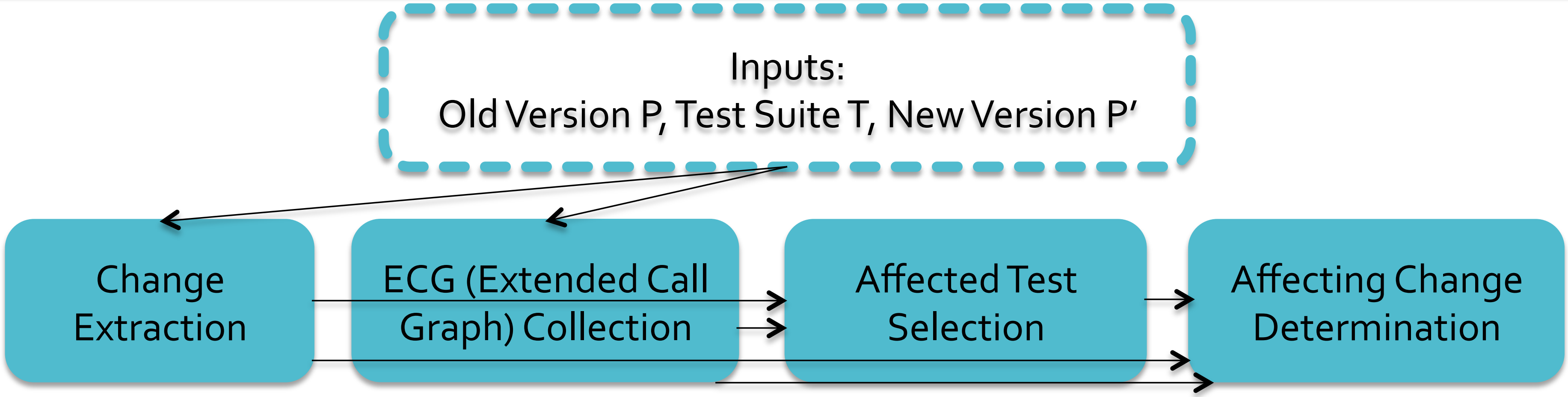
Step 2. Fault Tracer Change Impact Analysis

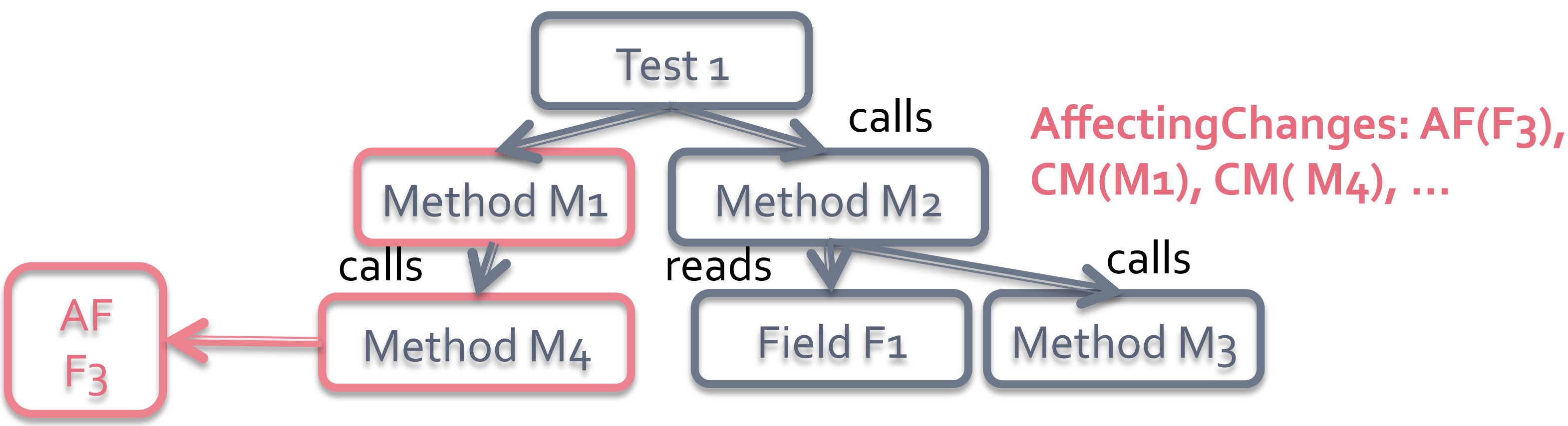


Step 2. Fault Tracer Change Impact Analysis



Step 2. Fault Tracer Change Impact Analysis

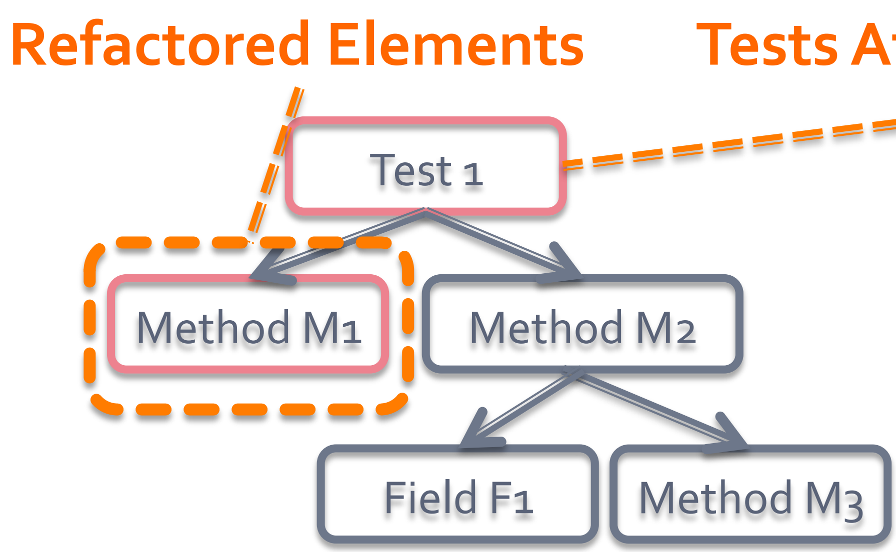




Step 3. Refactoring Change Impact Assessment

Investigate Refactoring Change Impact on Tests

Identify Tests Affected by Refactoring Edits
Identify Refactoring Edits Affecting Tests



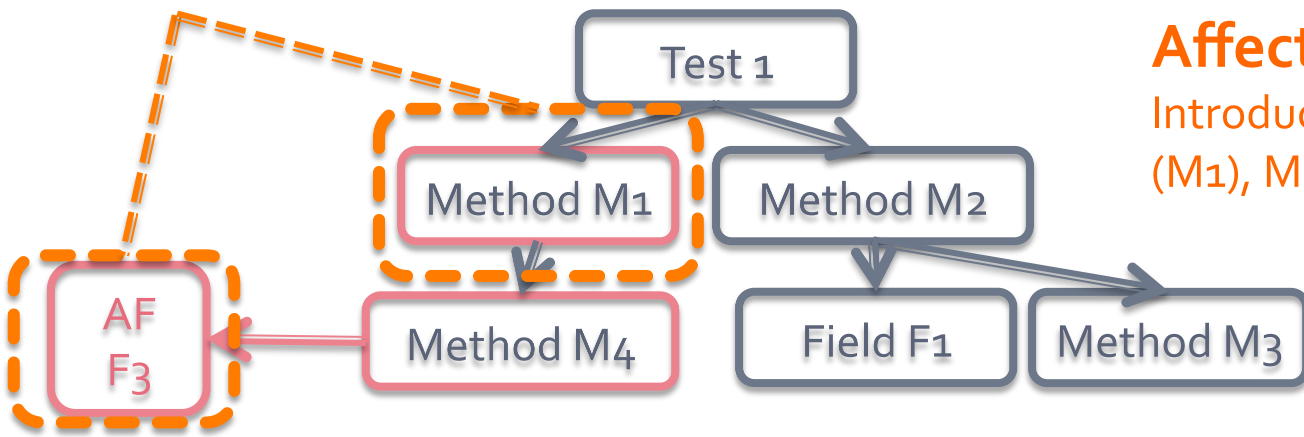
Step 3. Refactoring Change Impact Assessment

Investigate Refactoring Change Impact on Tests

Identify Tests Affected by Refactoring Edits

Identify Refactoring Edits Affecting Tests

Refactored Elements



Refactoring Edits Affecting Tests:

Introduce_Local_Variable (M1), Move Field (F3), …

Data Sets

	JMeter	XMLSecurity	Ant
# Versions	6	4	9
Releases	R0.0 to R5.0	R0.0 to R3.0	R0.0 to R8.0
LOC	31005~40695	17435~22863	17201~80444
Classes	313~402	181~154	172~650
Methods	2501~3237	1244~1023	1581~7190
Fields	830~970	129~151	440~3212
Refactoring Types	4 ~ 12	6~10	0~14
Total Correct Refactorings	349	161	511
Atomic Changes	307	214	1155

Outline

- Motivation and Related Work
- Study Approach Overview
- **Research Questions and Results**
- Limitations
- Conclusions

Research Questions

- Q1: Are there adequate tests for refactoring edits in practice?
- Q2: How many of existing regression tests are relevant to refactoring edits and thus need to be re-run for the new version?
- Q3: What proportion of failure-inducing changes are relevant to refactorings?

Q1. Are there adequate tests for refactoring edits in practice?

- Test Coverage $\frac{|T|}{|A|}$
  - The percentage of tested elements $|T|$ out of all code elements $|A|$
- Change Test Coverage $\frac{|T \cap C|}{|C|}$
  - The percentage of changed elements exercised by tests $|T \cap C|$ out of all changed elements $|C|$
- Refactoring Test Coverage $\frac{|T \cap R|}{|R|}$
  - The percentage of refactored elements exercised by tests $|T \cap R|$ out of all refactored elements $|R|$

Q1. Are there adequate tests for refactoring edits in practice?

	Refactored Elements $\lvert R \rvert$	Changed Elements $\lvert C \rvert$	Tested Elements $\lvert T \rvert$	Change Test Coverage $\frac{\lvert T \cap C \rvert}{\lvert C \rvert}$	Refactoring Test Coverage $\frac{\lvert T \cap R \rvert}{\lvert R \rvert}$	Test Coverage $\frac{\lvert T \rvert}{\lvert A \rvert}$
JMeter	352	4040	5776	23.8%	16.5%	29.8%
XML	60	1101	1719	25.1%	61.7%	41.2%
Ant	326	4375	10588	15.1%	19.9%	25.7%
Total	738	9516	18038	19.9%	22.1%	27.9%

Only 22% of refactored methods and fields are tested by existing regression tests.

Q1. Are there adequate tests for refactoring edits in practice?

	Refactored Elements $\lvert R \rvert$	Changed Elements $\lvert C \rvert$	Tested Elements $\lvert T \rvert$	Change Test Coverage $\frac{\lvert T \cap C \rvert}{\lvert C \rvert}$	Refactoring Test Coverage $\frac{\lvert T \cap R \rvert}{\lvert R \rvert}$	Test Coverage $\frac{\lvert T \rvert}{\lvert A \rvert}$
JMeter	352	4040	5776	23.8%	16.5%	29.8%
XML	60	1101	1719	25.1%	61.7%	41.2%
Ant	326	4375	10588	15.1%	19.9%	25.7%
Total	738	9516	18038	19.9%	22.1%	27.9%

: If the refactoring edits are impure, more tests need to cover refactoring edits

Q2. How many of existing tests are relevant to refactoring edits?

- **AT:** affected tests
- **AT_R:** the ratio of affected tests that exercise at least one refactoring edit location
- **AC:** affecting changes
- **AC_R:** the ratio of affecting changes whose location overlaps with at least one refactoring edit

Q2. How many of existing tests are relevant to refactoring edits?

Pair	Affected Tests $\lvert AT \rvert$	Tests Affected By Refactoring $\lvert AT_R \rvert$	Affecting Refactorings $\lvert AC_R \rvert$	Refactoring to Change Ratio $\frac{\lvert R \rvert}{\lvert C \rvert}$
JMeter	284	120 (42.2%)	70	8.7%
XML	180	133 (73.8%)	35	5.4%
Ant	1100	311(28.2%)	85	7.4%
Total	1564	594(38.0%)	190	7.8%

While refactoring edits constitute only 8% of atomic changes, 38% of affected tests are relevant to refactoring edits

Q2. How many of existing tests are relevant to refactoring edits?

Pair	Affected Tests $\|AT\|$	Tests Affected By Refactoring $\|AT_R\|$	Affecting Refactorings $\|AC_R\|$	Refactoring to Change Ratio $\frac{\|R\|}{\|C\|}$
JMeter	284	120 (42.2%)	70	8.7%
XML	180	133 (73.8%)	35	5.4%
Ant	1100	311(28.2%)	85	7.4%
Total	1564	594(38.0%)	190	7.8%

If the refactorings are pure and can be isolated, then there's an opportunity of saving testing cost.

Q3. How much of failure-inducing edits are related to refactorings?

- **AT_F:** affected tests that succeeded in the old version but failed in the new version
- **AT_{RF}:** a subset of **AT_F** that exercise refactoring edits
- **AC_F:** failure-inducing changes, i.e., a set of affecting changes for the failed tests
- **AC_{RF}:** a subset of **AC_F** that exercise the location of refactoring edits

Q3.How much of failure-inducing edits are related to refactorings?

Pair	Failed Affected Tests $\lvert AT_F \rvert$	Tests Affected By Refactoring $\lvert AT_{RF} \rvert$	Failure-Inducing Changes $\lvert AC_F \rvert$	Failure-Inducing Refactorings $\lvert AC_{RF} \rvert$
JMeter	19	14	43	3
XML	5	5	12	7
Ant	61	20	607	57
Total	80	39	662	67

Half of the failed affected tests include refactoring edits

Q3. How much of failure-inducing edits are related to refactorings?

Pair	Failed Affected Tests $\lvert AT_F \rvert$	Tests Affected By Refactoring $\lvert AT_{RF} \rvert$	Failure-Inducing Changes $\lvert AC_F \rvert$	Failure-Inducing Refactorings $\lvert AC_{RF} \rvert$
JMeter	19	14	43	3
XML	5	5	12	7
Ant	61	20	607	57
Total	80	39	662	67

Refactorings seem to appear on the execution traces of failed tests without being root failure causes.

Study Limitations and Future Work

- False negatives of refactoring reconstruction
- Our broader definition of refactoring edits—tolerating behavior modifications during our manual inspection
- Only three projects in SIR
- Our data is available in public:
  - http://users.ece.utexas.edu/~miryung/inspected_dataset.zip

Summary

- We study the impact of refactoring edits on regression tests
  - Refactoring test coverage is insufficient
  - Though only a small portion of edits consists of refactoring edits, many tests are impacted by them
- We need an automated regression test augmentation and validation approach targeting refactoring edits

Acknowledgment

- This research is in part supported by National Science Foundation, CCF-1117902, CCF-1149391, and CCF-1043810 and Microsoft SEIF award.