Automatic Unit Test Amplification for DevOps

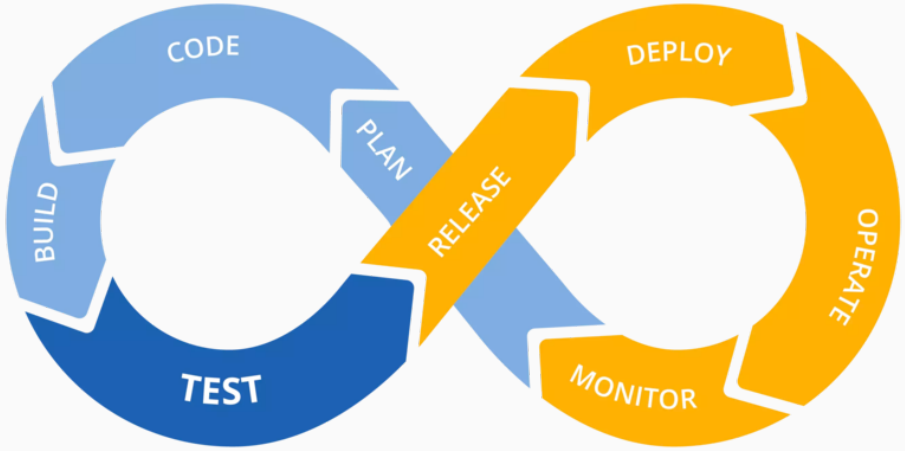
Benjamin Danglot

Thursday, November 14, 2019

Supervisors:	Martin Monperrus	-	KTH Royal Institute of Technology
	Lionel Seinturier	-	University of Lille
Reviewers:	Lydie du Bousquet	-	Université Grenoble Alpes
	Jordi Cabot	-	University of Catalonia
Examiner:	Benoit Baudry	-	KTH Royal Institute of Technology
Chair:	Daniel Le Berre	-	University of Artois
Invited:	Vincent Massol	-	XWiki SAS

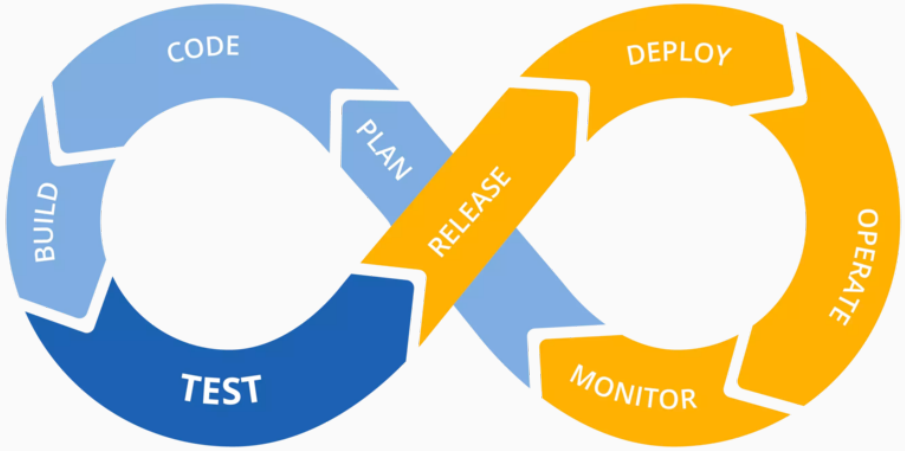
Introduction

Introduction



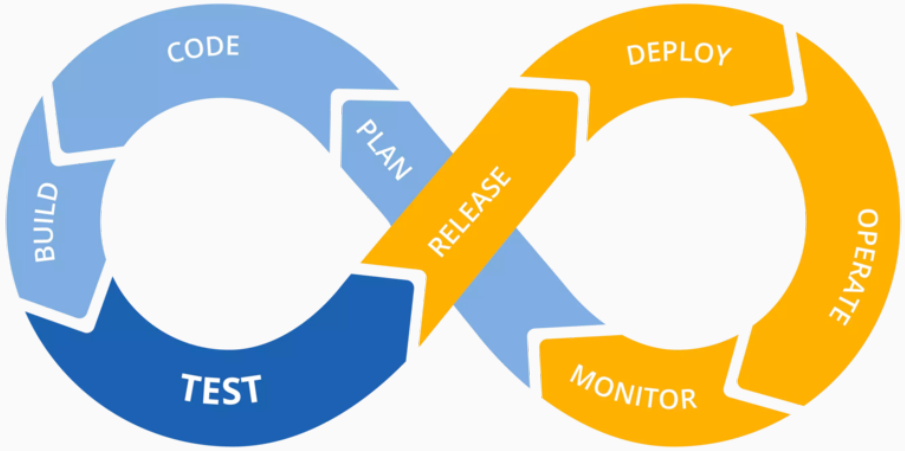
- DevOps

Introduction



- DevOps
- Automated regression testing

Introduction



- DevOps
- Automated regression testing
- Continuous integration

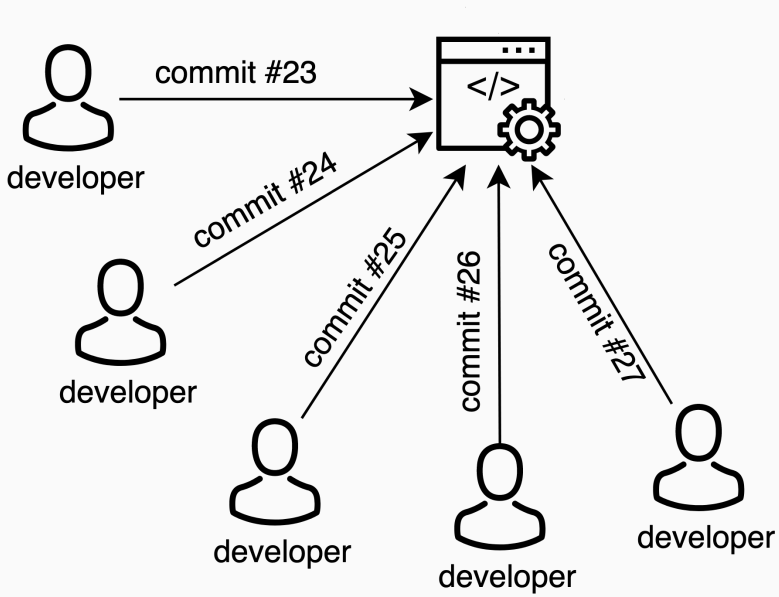
Introduction

A program

Introduction

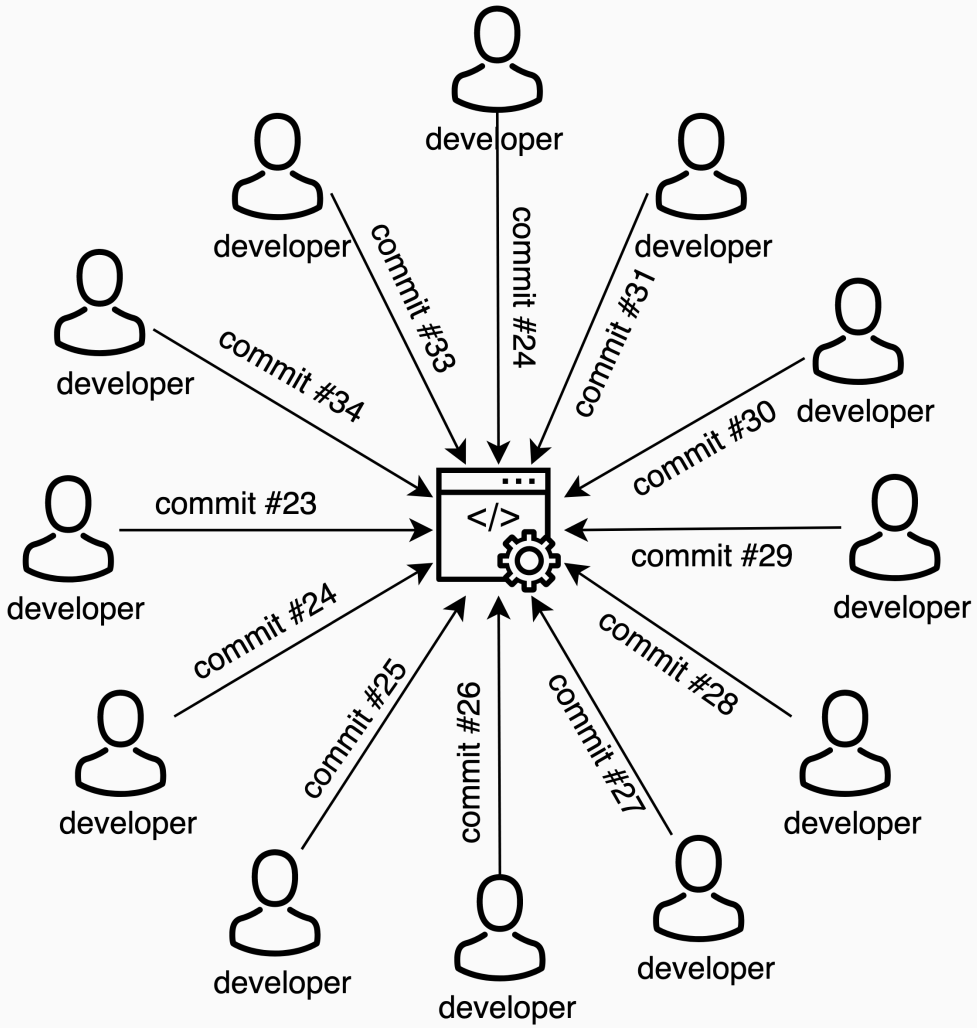
A developer commits to the program

Introduction



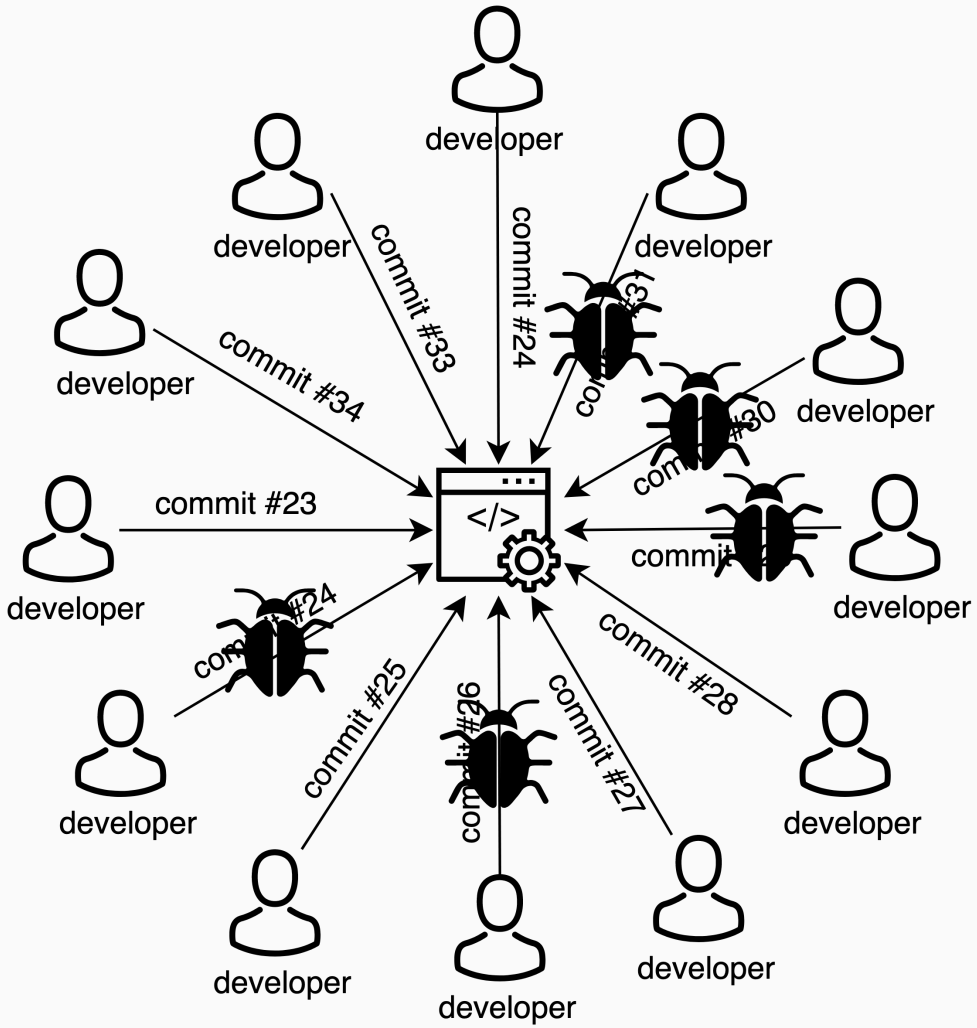
Some developers commit to the program

Introduction



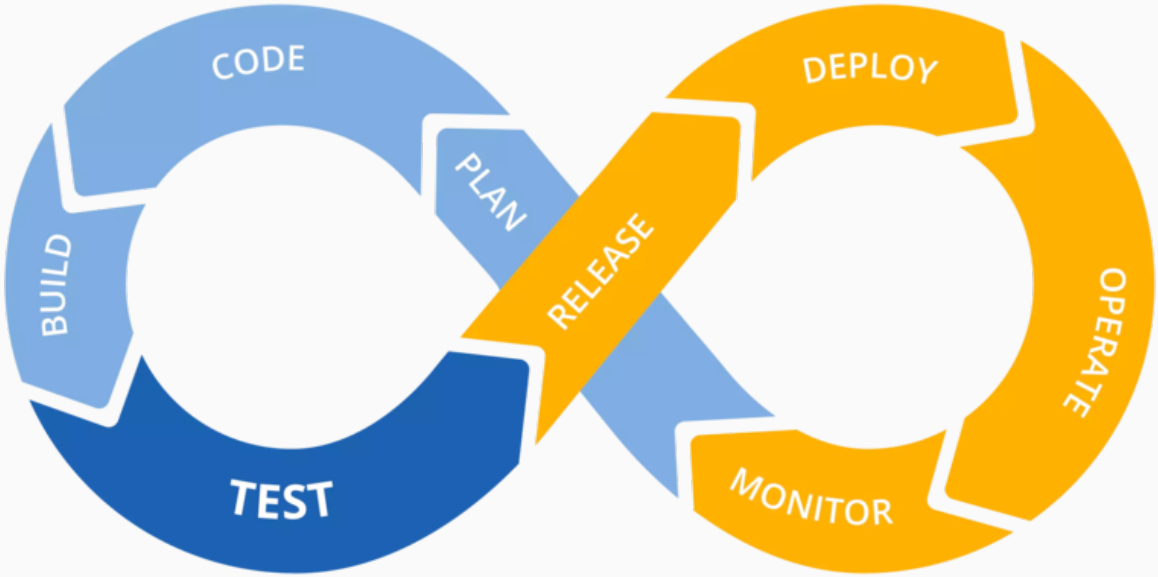
A lot of developers commit to the program

Introduction

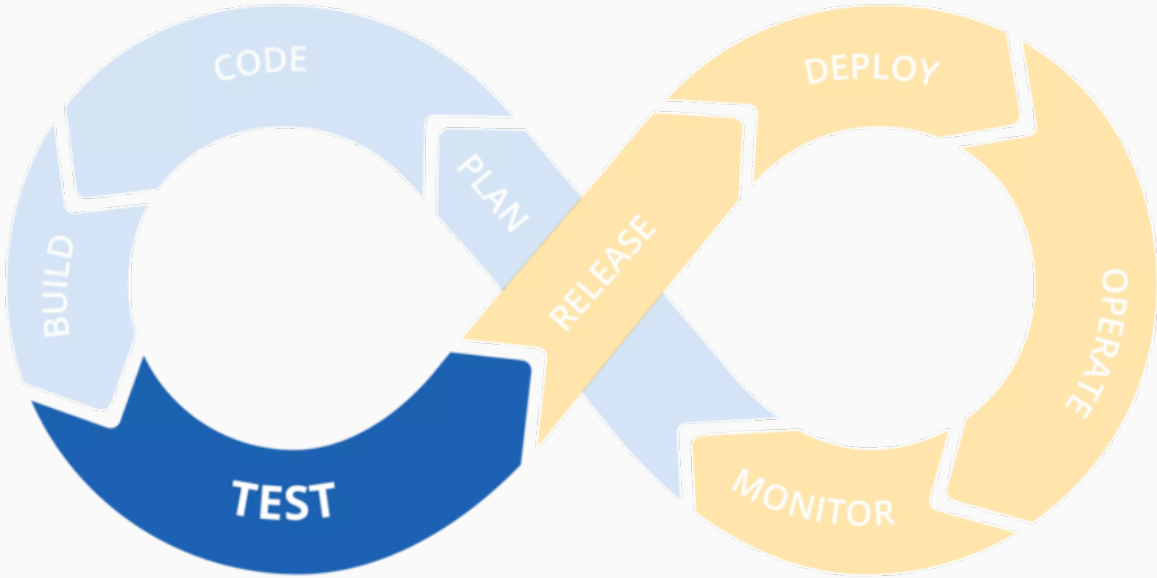


Some of them can introduce bugs

Introduction



Introduction



Automatic Test Generation[1]

[1] Fraser and Arcuri. "Evolutionary Generation of Whole Test Suites". *QSIC'11*.

Automatic Test Generation[1]

Generate tests that cover 100% of the program

[1] Fraser and Arcuri. "Evolutionary Generation of Whole Test Suites". *QSIC'11*.

Automatic Test Generation

Generate tests that cover 100% of the program

Do generated tests help developers?[2]

[2]Fraser et al. "Does Automated Unit Test Generation Really Help Software Testers? A Controlled Empirical Study". *TOSEM'15*.

Automatic Test Generation

Generate tests that cover 100% of the program

Do generated tests help developers?[3]

- Generated tests are too artificial
- Generated tests do not have clear intent

[3] Fraser et al. "Does Automated Unit Test Generation Really Help Software Testers? A Controlled Empirical Study". *TOSEM'15*.

Thesis: modifying **human tests** can provide actionable tests for developers in regression testing context

Thesis: modifying **human tests** can provide actionable tests for developers in regression testing context

Why modifying **human tests** is a good candidate ?

Thesis: modifying **human tests** can provide actionable tests for developers in regression testing context

Why modifying **human tests** is a good candidate ?

1. Feasible: Developers do write tests

Thesis: modifying **human tests** can provide actionable tests for developers in regression testing context

Why modifying **human tests** is a good candidate ?

1. Feasible: Developers do write tests
2. Better: Starting point is not zero, the result is better than generation

Thesis: modifying **human tests** can provide actionable tests for developers in regression testing context

Why modifying **human tests** is a good candidate ?

1. Feasible: Developers do write tests
2. Better: Starting point is not zero, the result is better than generation
3. More actionable: The output would be real tests

Thesis: modifying **human tests** can provide actionable tests for developers in regression testing context

Why modifying **human tests** is a good candidate ?

1. Feasible: Developers do write tests
2. Better: Starting point is not zero, the result is better than generation
3. More actionable: The output would be real tests

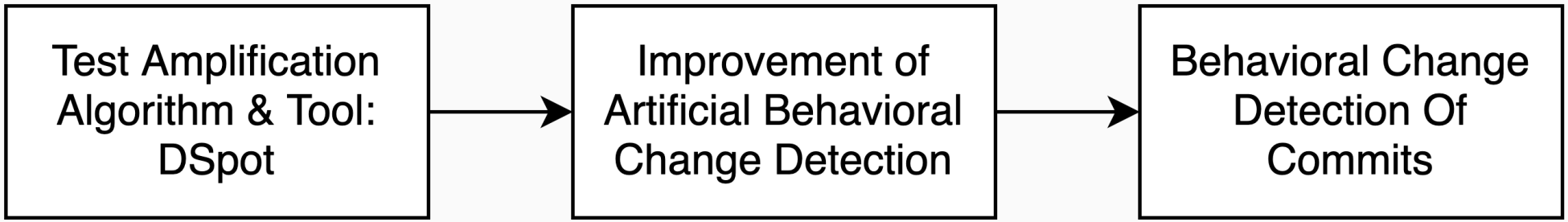
I coin this: **Test Amplification**

Thesis Goal & Outline

Goal: Improve **regression detection** by using test amplification

Thesis Goal & Outline

Goal: Improve **regression detection** by using test amplification



State Of The Art

Test amplification

Definition: test amplification consists of exploiting the **knowledge of existing test methods**, in which developers embed meaningful input data and expected properties in the form of oracles, in order to enhance these manually written tests with respect to an engineering goal.

State Of The Art: Review

Snowballing review[4] of the literature

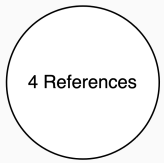
Starting set: 4 references

[4] Wohlin, Claes. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering". *EASE'14*.

State Of The Art: Review

Snowballing review[4] of the literature

Starting set: 4 references

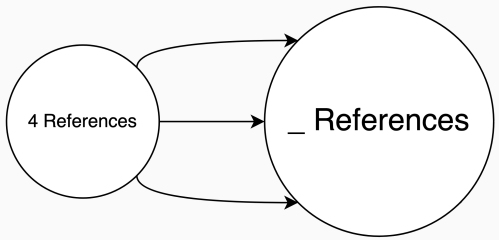


[4] Wohlin, Claes. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering". *EASE'14*.

State Of The Art: Review

Snowballing review[5] of the literature

Starting set: 4 references

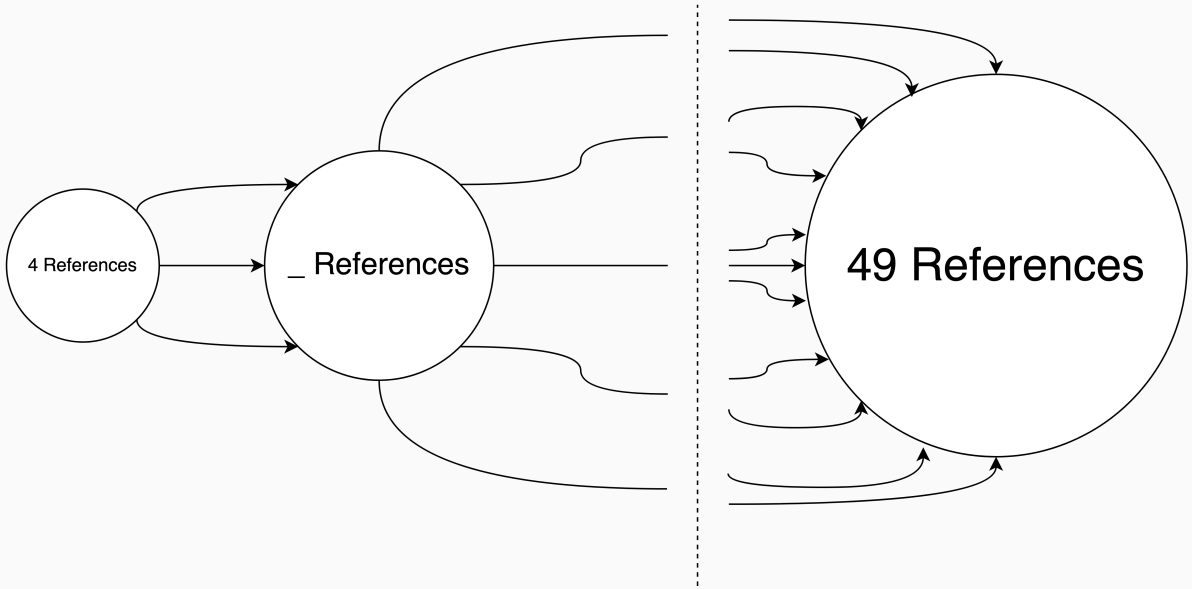


[5] Wohlin, Claes. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering". *EASE'14*.

State Of The Art: Review

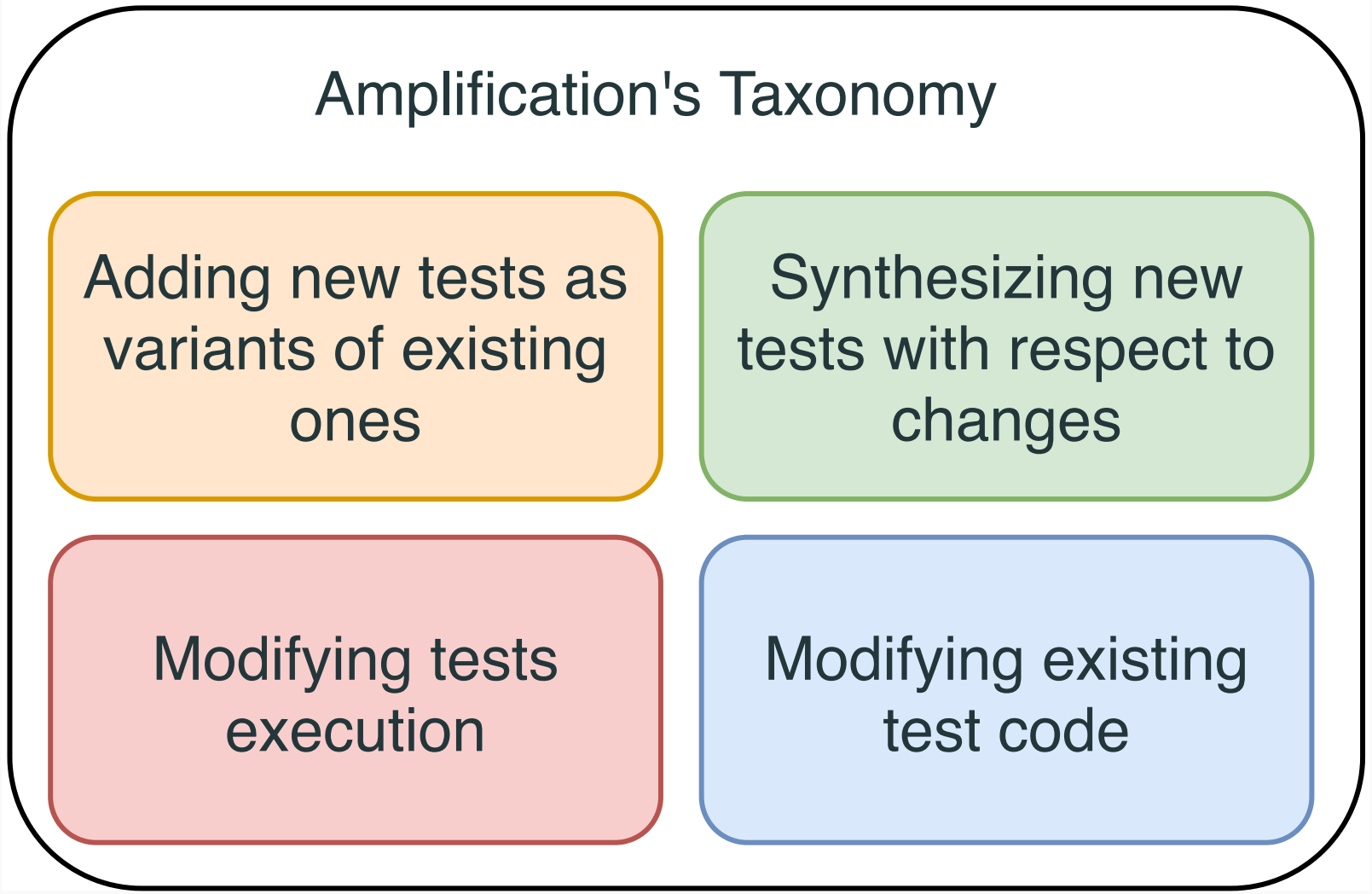
Snowballing review[6] of the literature

Starting set: 4 references



[6] Wohlin, Claes. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering". *EASE'14*.

State of the art: Review



Adding new tests as variants of existing ones

Example: random modification of integer vector of existing test methods to trigger an untested behavior.

Adding new tests as variants of existing ones

Example: random modification of integer vector of existing test methods to trigger an untested behavior.

Yoo, Shin and Harman, Mark. “Test data regeneration: generating new test data from existing test data”. STVR’12

Adding new tests as variants of existing ones

Example: random modification of integer vector of existing test methods to trigger an untested behavior.

Yoo, Shin and Harman, Mark. "Test data regeneration: generating new test data from existing test data". STVR'12

```
@Test
public void test() {
    MyClass myClass = new MyClass();
    myClass.myMethod( x: 1,  y: 1,  z: 1);
}
```

Adding new tests as variants of existing ones

Example: random modification of integer vector of existing test methods to trigger an untested behavior.

Yoo, Shin and Harman, Mark. "Test data regeneration: generating new test data from existing test data". STVR'12

```
@Test
public void test() {
    MyClass myClass = new MyClass();
    myClass.myMethod( x: 1,  y: 1,  z: 1);
}
```

original:

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Adding new tests as variants of existing ones

Example: random modification of integer vector of existing test methods to trigger an untested behavior.

Yoo, Shin and Harman, Mark. "Test data regeneration: generating new test data from existing test data". STVR'12

```
@Test
public void test() {
    MyClass myClass = new MyClass();
    myClass.myMethod( x: 1,  y: 1,  z: 1);
}
```

original: ampl1:

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

Adding new tests as variants of existing ones

Example: random modification of integer vector of existing test methods to trigger an untested behavior.

Yoo, Shin and Harman, Mark. "Test data regeneration: generating new test data from existing test data". STVR'12

```
@Test
public void test() {
    MyClass myClass = new MyClass();
    myClass.myMethod( x: 1,  y: 1,  z: 1);
}
```

original: $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ ampl1: $\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$ ampl2: $\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$ ... ampl n: $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

Synthetizing new tests with respect to change

Example: create a new test that specify the new behavior.

Synthetizing new tests with respect to change

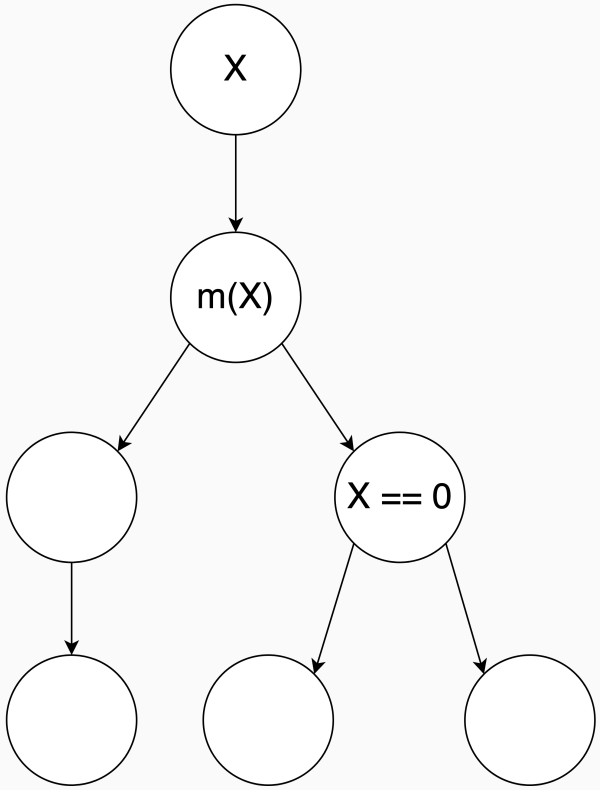
Example: create a new test that specify the new behavior.

Palikareva, Hristina, Kuchta, Tomasz, and Cadar, Cristian. "Shadow of a Doubt: Testing for Divergences Between Software Versions". ICSE'16

Synthetizing new tests with respect to change

Example: create a new test that specify the new behavior.

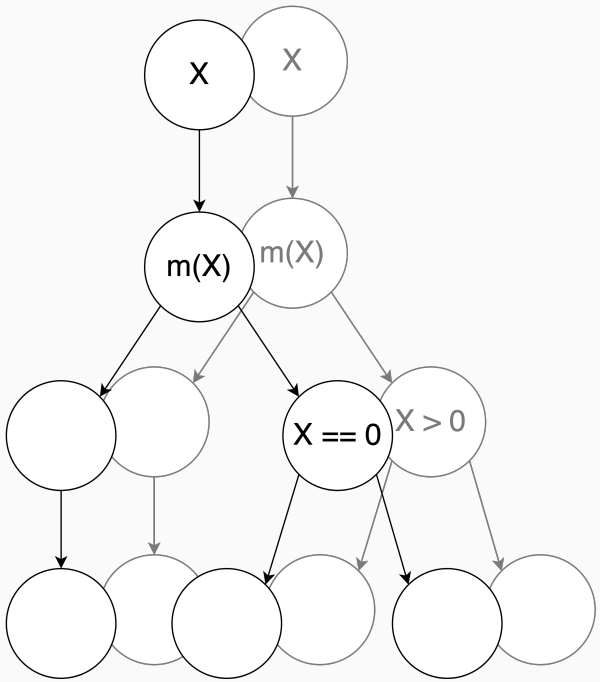
Palikareva, Hristina, Kuchta, Tomasz, and Cadar, Cristian. "Shadow of a Doubt: Testing for Divergences Between Software Versions". ICSE'16



Synthetizing new tests with respect to change

Example: create a new test that specify the new behavior.

Palikareva, Hristina, Kuchta, Tomasz, and Cadar, Cristian. "Shadow of a Doubt: Testing for Divergences Between Software Versions". ICSE'16



Modifying tests execution

Example: Randomize the order of test execution.

Modifying tests execution

Example: Randomize the order of test execution.

Cornu, Benoit, Seinturier, Lionel, and Monperrus, Martin. “Exception handling analysis and transformation using fault injection: Study of resilience against unanticipated exceptions”. IST’15

Modifying tests execution

Example: Randomize the order of test execution.

Cornu, Benoit, Seinturier, Lionel, and Monperrus, Martin. "Exception handling analysis and transformation using fault injection: Study of resilience against unanticipated exceptions". IST'15

```
public void myMethod() {
    try {
        FileUtils.deleteDirectory(new File(path));
    } catch (IOException e) {
        // do something
    }
}
```

Modifying tests execution

Example: Randomize the order of test execution.

Cornu, Benoit, Seinturier, Lionel, and Monperrus, Martin. "Exception handling analysis and transformation using fault injection: Study of resilience against unanticipated exceptions". IST'15

```
public void myMethod() {
    try {
        throwCornuEtAlException();
        FileUtils.deleteDirectory(new File(path));
    } catch (IOException e) {
        // do something
    }
}
```

Modifying existing test code

Example: generate assertions on existing test methods.

Modifying existing test code

Example: generate assertions on existing test methods.

Xie, Tao. “Augmenting Automatically Generated Unit-test Suites with Regression Oracle Checking”. ECOOP’06

Modifying existing test code

Example: generate assertions on existing test methods.

Xie, Tao. "Augmenting Automatically Generated Unit-test Suites with Regression Oracle Checking". ECOOP'06

```
@Test
public void test(){
    MyClass myClass = new MyClass();
    myClass.myMethod();
    assertEquals( expected: 1, myClass.getX());
}
```

Modifying existing test code

Example: generate assertions on existing test methods.

Xie, Tao. "Augmenting Automatically Generated Unit-test Suites with Regression Oracle Checking". ECOOP'06

```java
@Test
public void test(){
    MyClass myClass = new MyClass();
    myClass.myMethod();
    assertEquals( expected: 1, myClass.getX());
}
```

```java
@Test
public void test(){
    MyClass myClass = new MyClass();
    myClass.myMethod();
    assertEquals( expected: 1, myClass.getX());
    assertEquals( expected: true, myClass.getY());
    assertEquals( expected: 23, myClass.getZ());
}
```

Modifying existing test code

Example: generate assertions on existing test methods.

Xie, Tao. "Augmenting Automatically Generated Unit-test Suites with Regression Oracle Checking". ECOOP'06

```java
@Test
public void test(){
    MyClass myClass = new MyClass();
    myClass.myMethod();
    assertEquals( expected: 1, myClass.getX());
}
```

```java
@Test
public void test(){
    MyClass myClass = new MyClass();
    myClass.myMethod();
    assertEquals( expected: 1, myClass.getX());
    assertEquals( expected: true, myClass.getY());
    assertEquals( expected: 23, myClass.getZ());
}
```

Lack in state of the art

Lack in state of the art

- No evaluation including developers has been carried out

Lack in state of the art

- No evaluation including developers has been carried out

  ✗ need for assessment from real **developers** assessment

Lack in state of the art

- No evaluation including developers has been carried out
  ✗ need for assessment from real **developers** assessment
- Lack of study on regression detection improvement

Lack in state of the art

- No evaluation including developers has been carried out

  ✗ need for assessment from real **developers** assessment

- Lack of study on regression detection improvement

  ✗ need for an approach to improve **detection of regressions**

Lack in state of the art

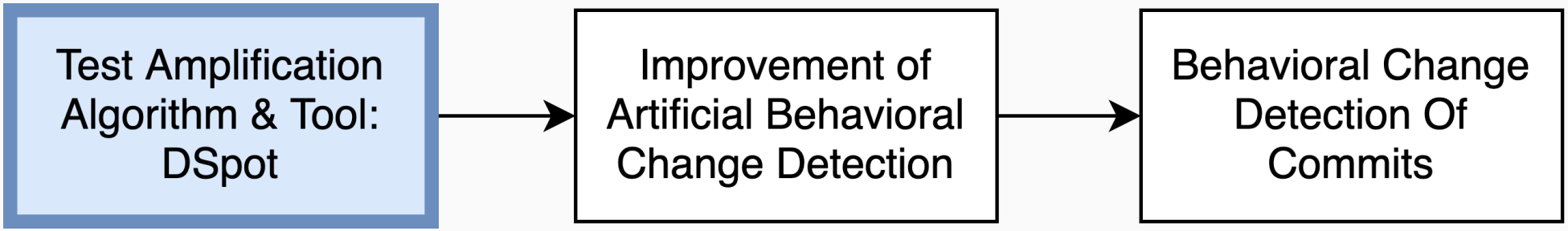
- No evaluation including developers has been carried out
  ✗ need for assessment from real **developers** assessment
- Lack of study on regression detection improvement
  ✗ need for an approach to improve **detection of regressions**
- Lack of generalizable results

Lack in state of the art

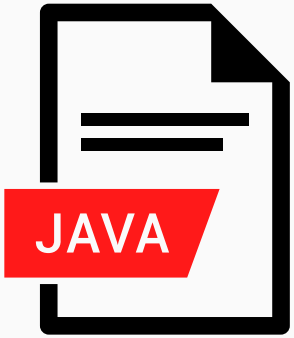
- No evaluation including developers has been carried out
  ✗ need for assessment from real **developers** assessment
- Lack of study on regression detection improvement
  ✗ need for an approach to improve **detection of regressions**
- Lack of generalizable results
  ✗ need for **benchmarks of real programs**

Test Amplification Algorithm & Tool: DSpot

Outline


Test Amplification Algorithm & Tool: DSpot
Improvement of Artificial Behavioral Change Detection
Behavioral Change Detection Of Commits

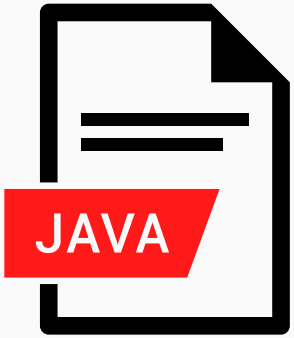
DSpot: Principle



Program

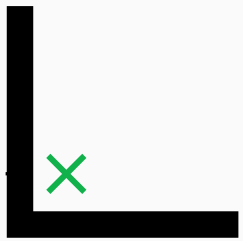

Test suite

DSpot: Principle



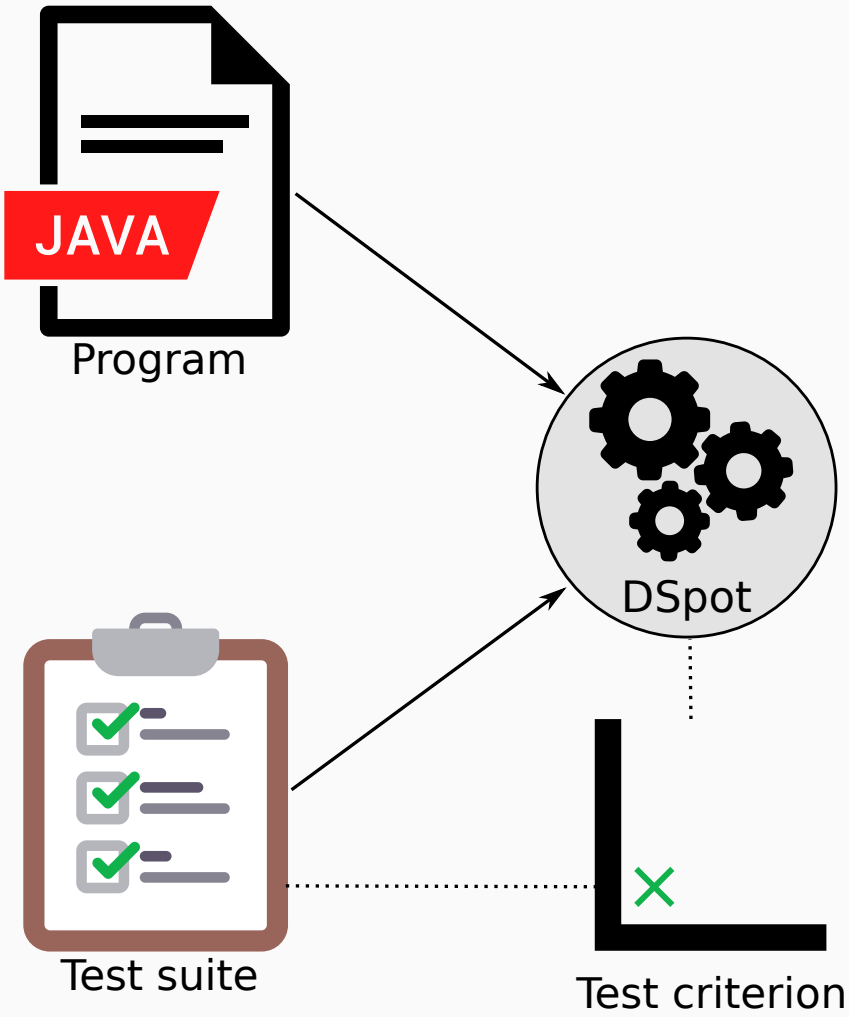
Program

Test suite

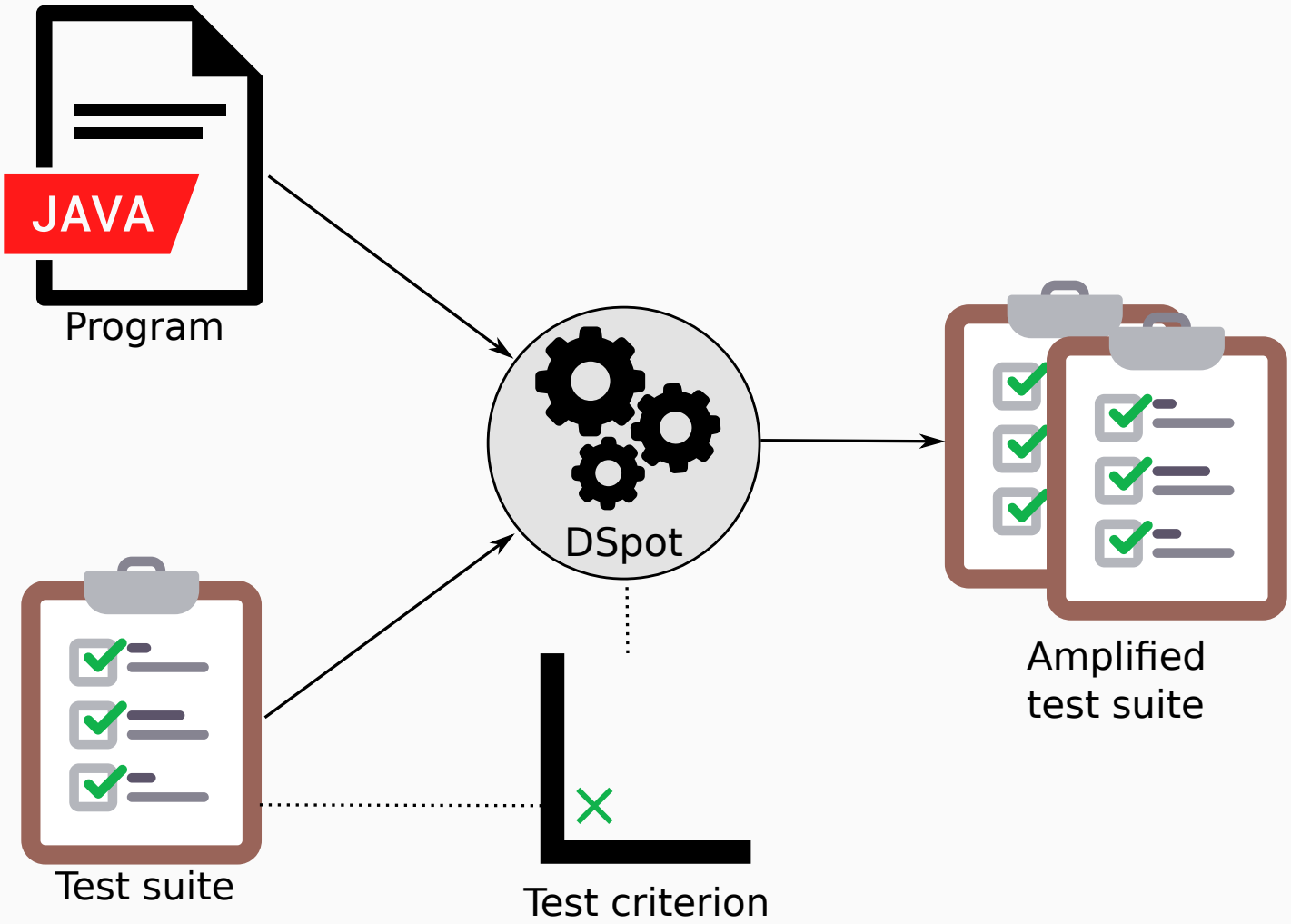


Test criterion

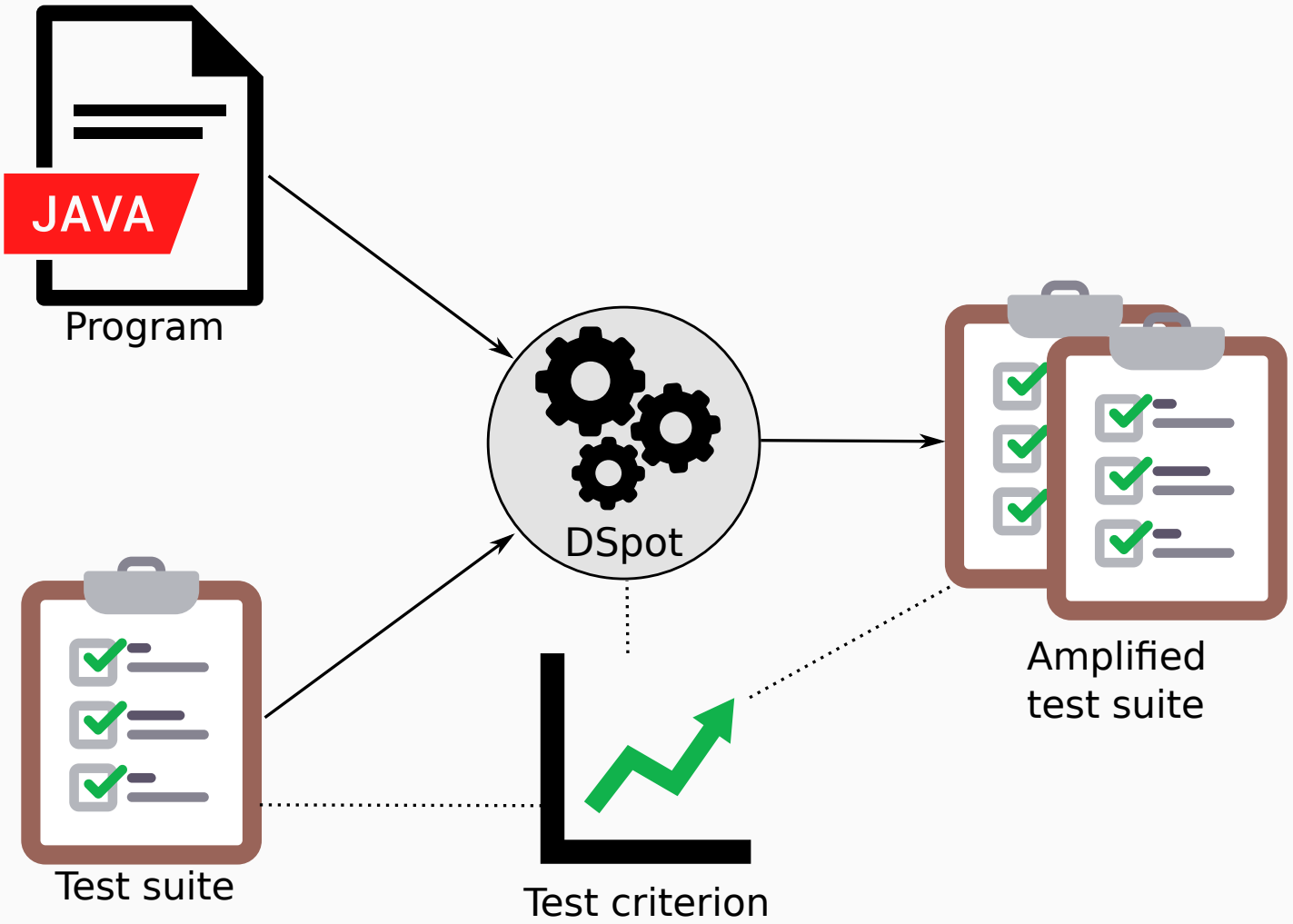
DSpot: Principle



DSpot: Principle



DSpot: Principle



Test Example

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Test Example

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

In blue, the input of the test

Test Example

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

In blue, the input of the test

In green, an oracle of the test to verify the current behavior

Test Example

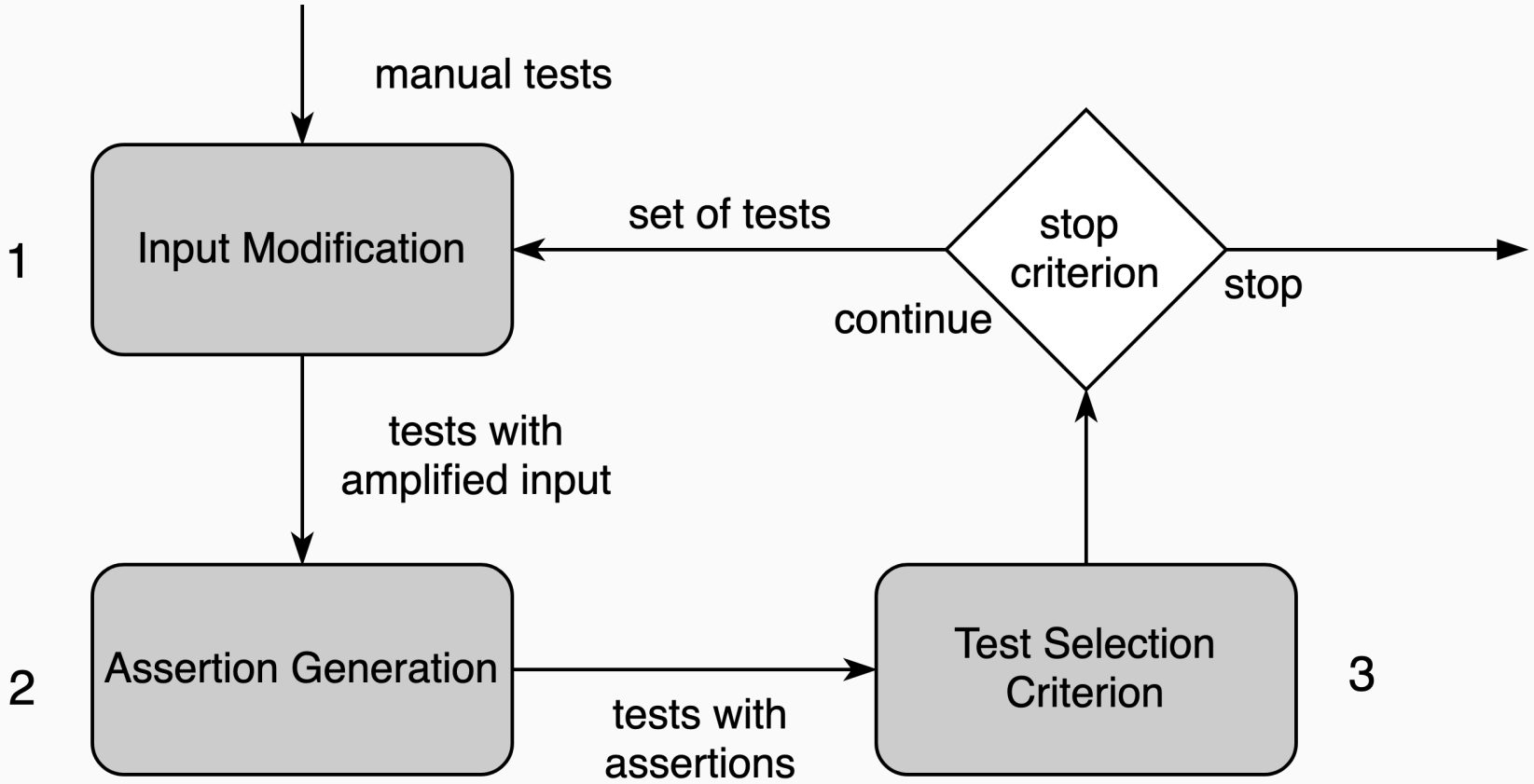
```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

In blue, the input of the test

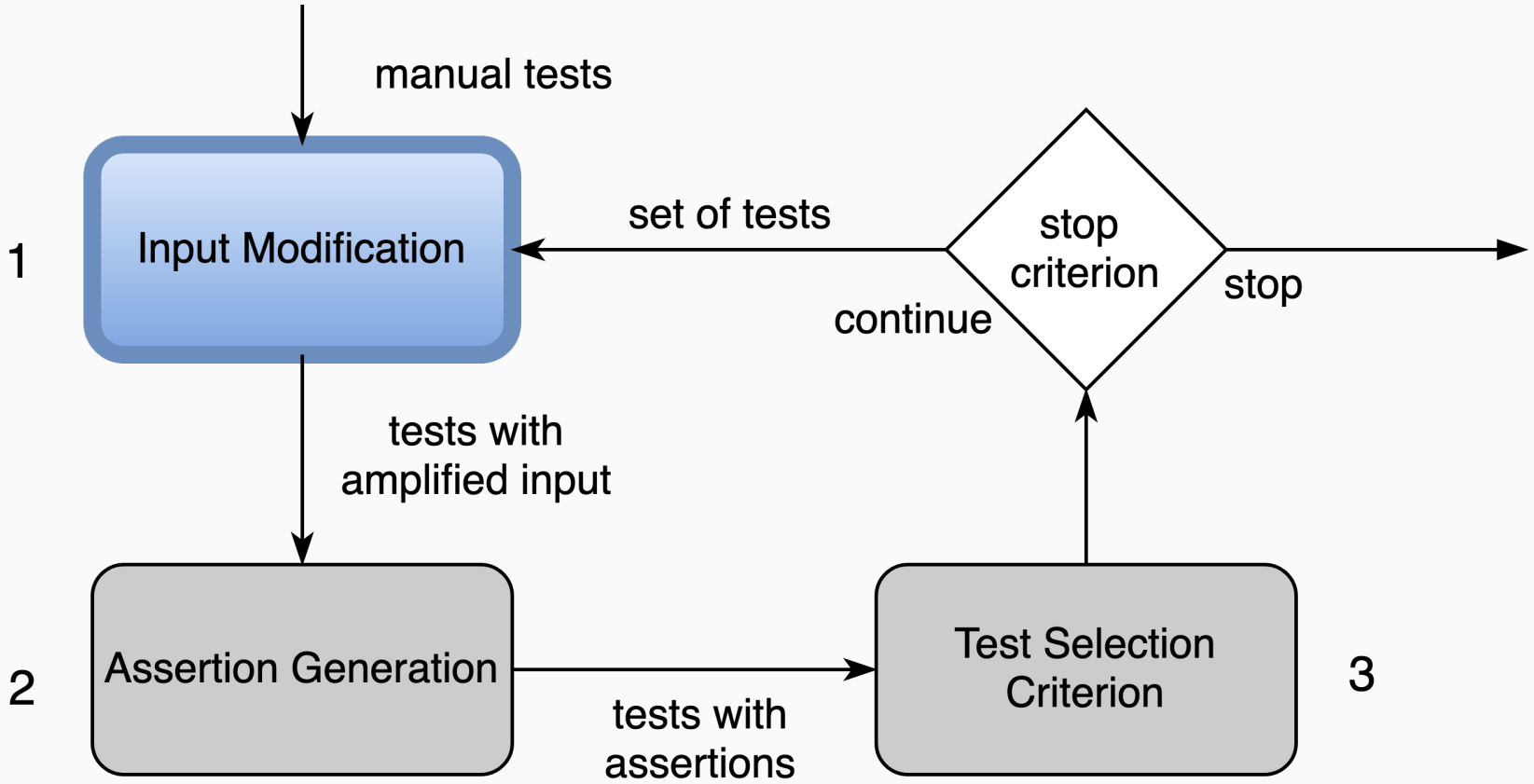
In green, the oracle of the test to verify the current behavior

In orange, the observable state of the program through getters

How DSpot works?



DSpot: 1. Input Modification



DSpot: 1. Input Modification

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Goal: create a new observable state

DSpot: 1. Input Modification

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Goal: create a new observable state

Original observable state:

benjamin.isHungry() → *false*

benjamin.isHappy() → *true*

DSpot: 1. Input Modification

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Goal: create a new observable state

Original observable state:

benjamin.isHungry() → *false*

benjamin.isHappy() → *true*

DSpot: 1. Input Modification

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    [                              ]  ← remove method call
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Goal: create a new observable state

Original observable state:

benjamin.isHungry() → *false*

benjamin.isHappy() → *true*

DSpot: 1. Input Modification

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Goal: create a new observable state

Original observable state:

benjamin.isHungry() → *false*

benjamin.isHappy() → *true*

DSpot: 1. Input Modification

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Goal: create a new observable state

Original observable state:

benjamin.isHungry() → *false*

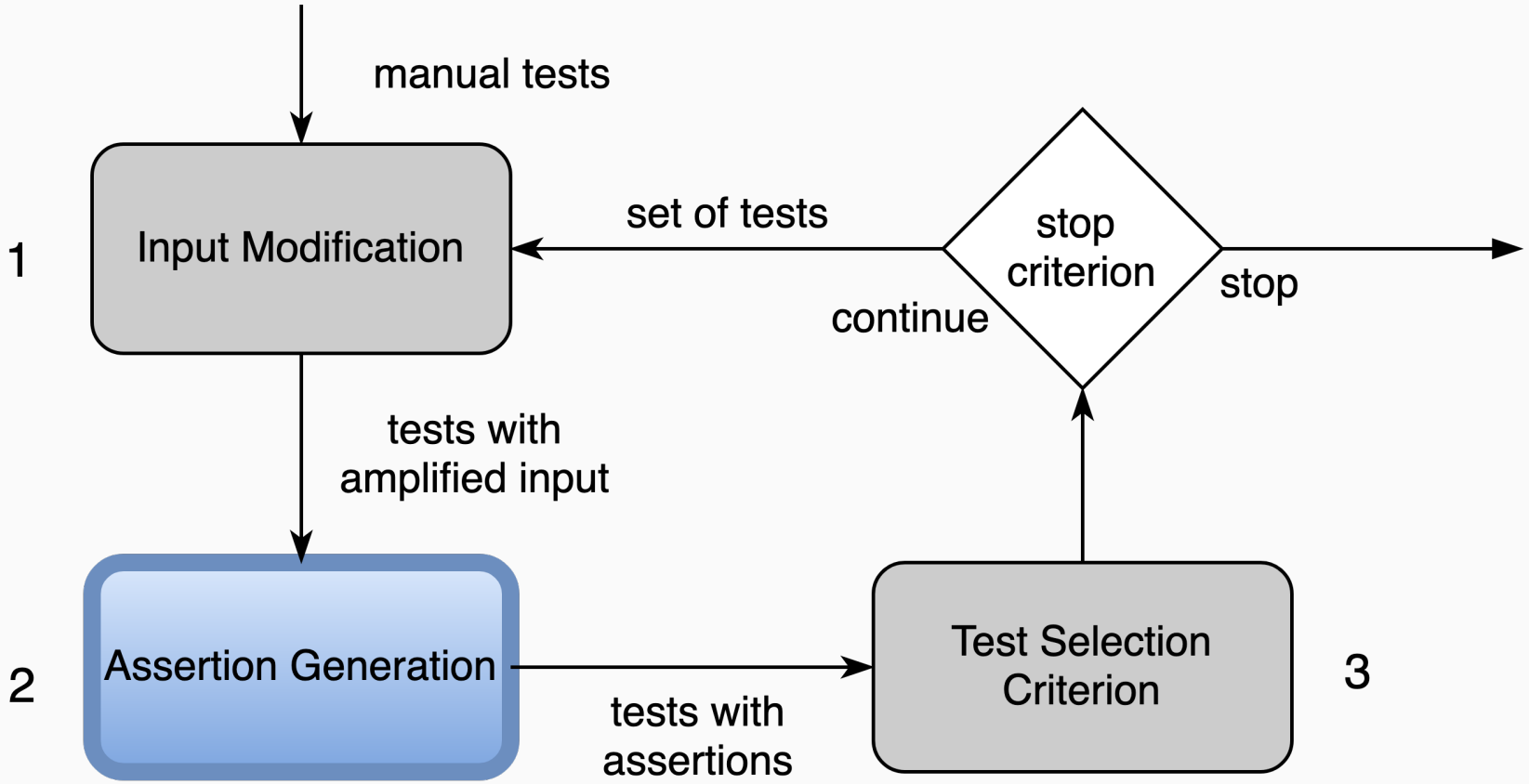
benjamin.isHappy() → *true*

Amplified observable state:

benjamin.isHungry() → *true*

benjamin.isHappy() → *false*

DSpot: 2. Assertion Generation



DSpot: 2. Assertion Generation

1. Remove existing assertions
2. Instrument the test
3. Run instrumented test to collect values
4. Generate assertions based on values collected

DSpot: 2. Assertion Generation

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

1. Remove existing assertions

DSpot: 2. Assertion Generation

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

1. Remove existing assertions

DSpot: 2. Assertion Generation

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

1. Remove existing assertions

DSpot: 2. Assertion Generation

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    Log.log(benjamin, id:"benjamin");
}
```

2. Instrument the test to collect observable values

DSpot: 2. Assertion Generation

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    Log.log(benjamin, id:"benjamin");
}
```

3. Execute the test and collect observable values

DSpot: 2. Assertion Generation

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    Log.log(benjamin, id:"benjamin");
}
```

3. Execute the test and collect observable values

Observable state:

benjamin.isHungry() → *true*

benjamin.isHappy() → *false*

DSpot: 2. Assertion Generation

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: true, benjamin.isHungry()
    );
    assertEquals(
            expected: false, benjamin.isHappy()
    );
}
```

4. Generate assertions based on the observed values

DSpot: 2. Assertion Generation

```java
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: true, benjamin.isHungry()
    );
    assertEquals(
            expected: false, benjamin.isHappy()
    );
}
```

Sum-up: input modification and assertion generation

Original test

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Amplified test

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: true, benjamin.isHungry()
    );
    assertEquals(
            expected: false, benjamin.isHappy()
    );
}
```

Sum-up: input modification and assertion generation

Original test

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Amplified test

```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    // Method call removed
    assertEquals(
            expected: true, benjamin.isHungry()
    );
    assertEquals(
            expected: false, benjamin.isHappy()
    );
}
```

→ 2 non-regression assertions

Discussion on assertions

Original test

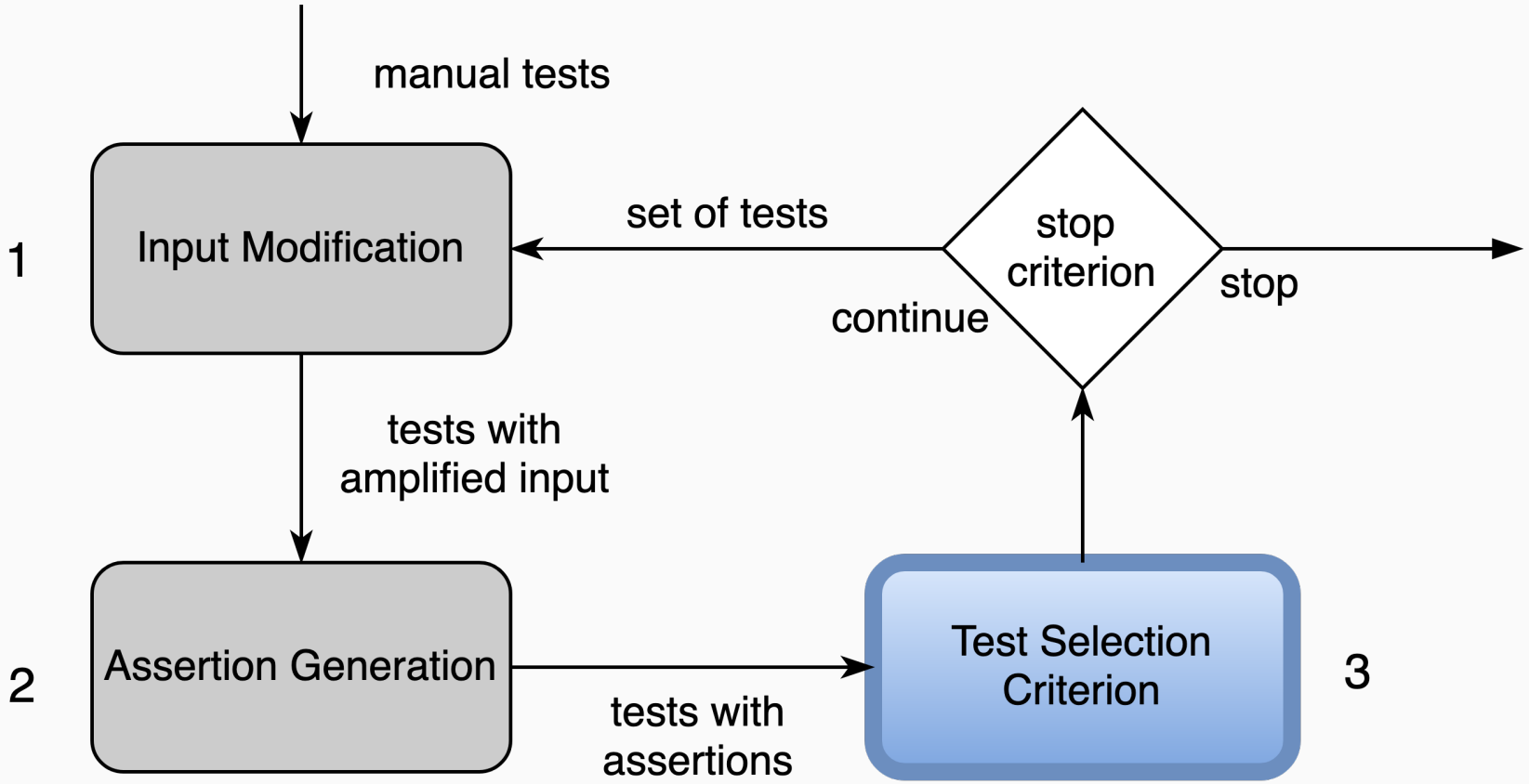
```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();
    benjamin.eat(tacos);
    assertEquals(
            expected: false, benjamin.isHungry()
    );
}
```

Amplified test

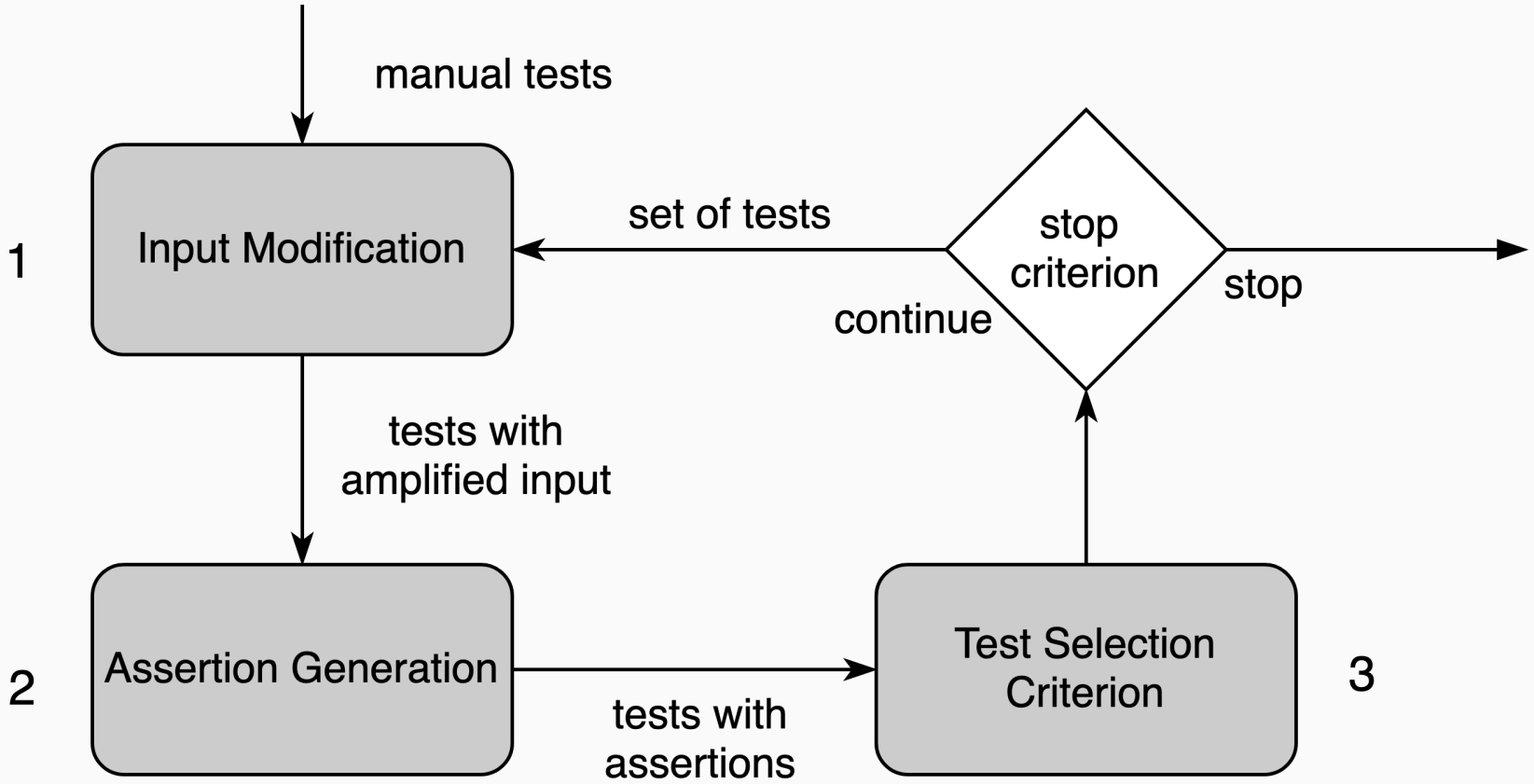
```
@Test
public void testEatTacos() {
    Tacos tacos = new Tacos();
    Benjamin benjamin = new Benjamin();

    assertEquals(
            expected: true, benjamin.isHungry()
    );
    assertEquals(
            expected: false, benjamin.isHappy()
    );
}
```

DSpot: 3. Test Selection

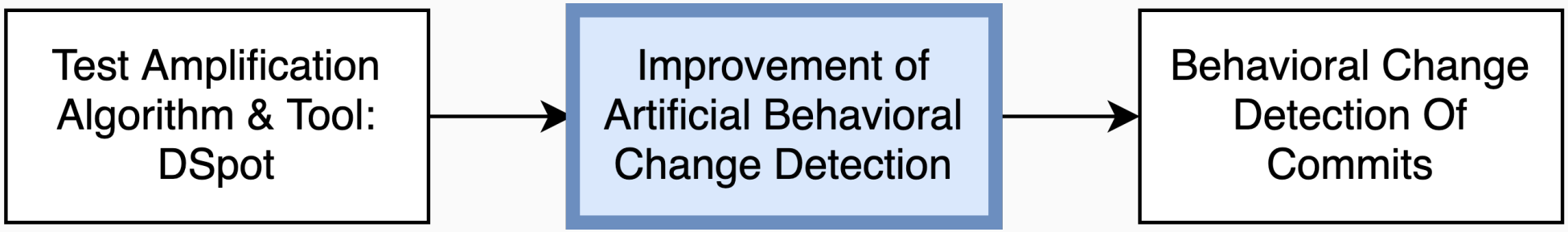


DSpot: iteration

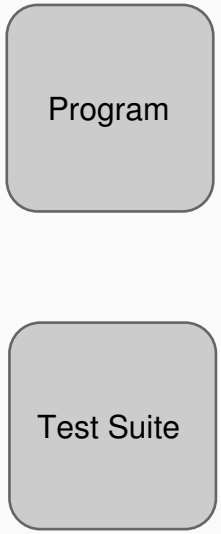


Improvement of Artificial Behavioral Change Detection

Outline

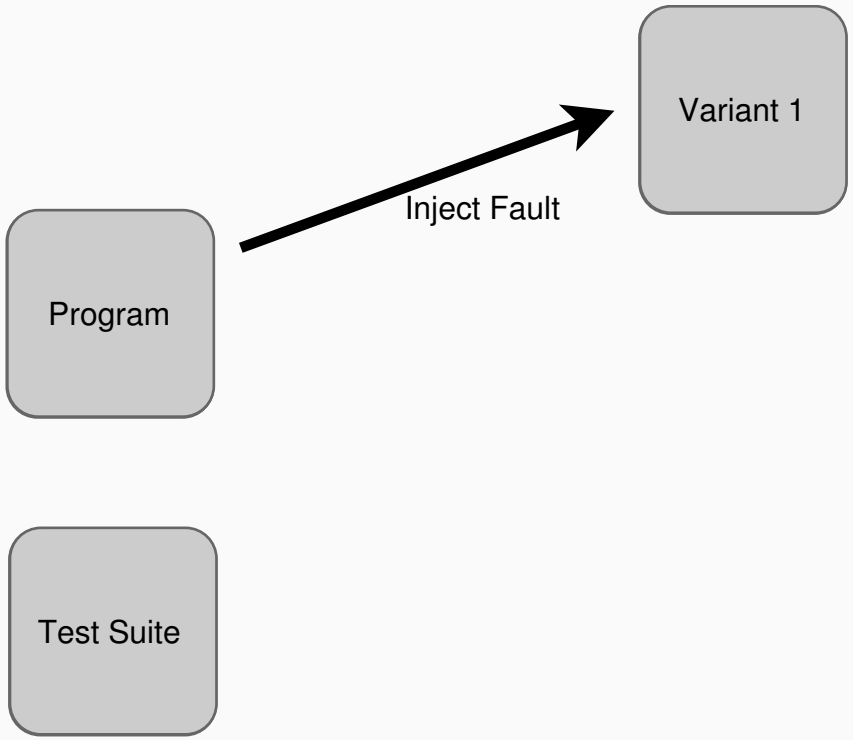

Test Amplification Algorithm & Tool: DSpot
Improvement of Artificial Behavioral Change Detection
Behavioral Change Detection Of Commits

Pre-Requisite: Mutation score



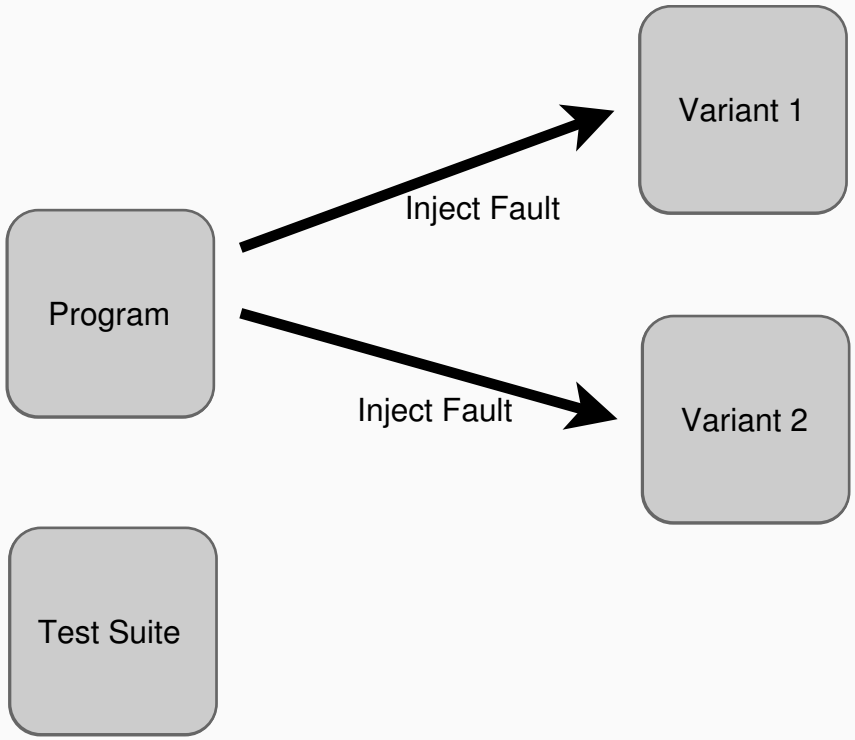
The mutation score is a measure of the efficiency of tests

Mutation score



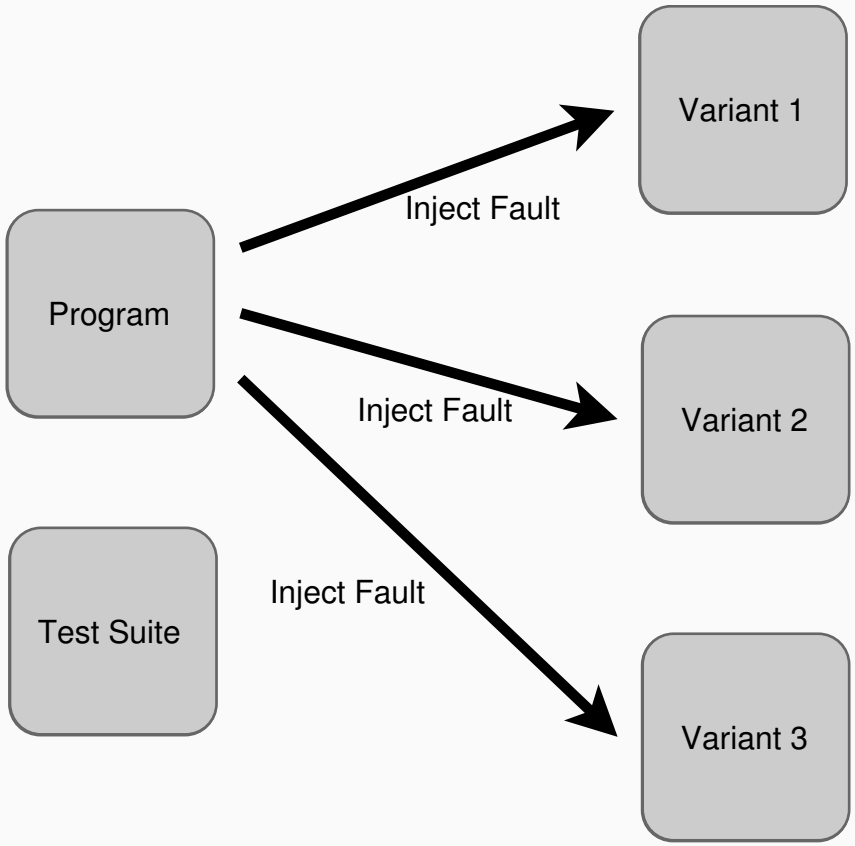
First, creates variants of program by injecting faults

Mutation score



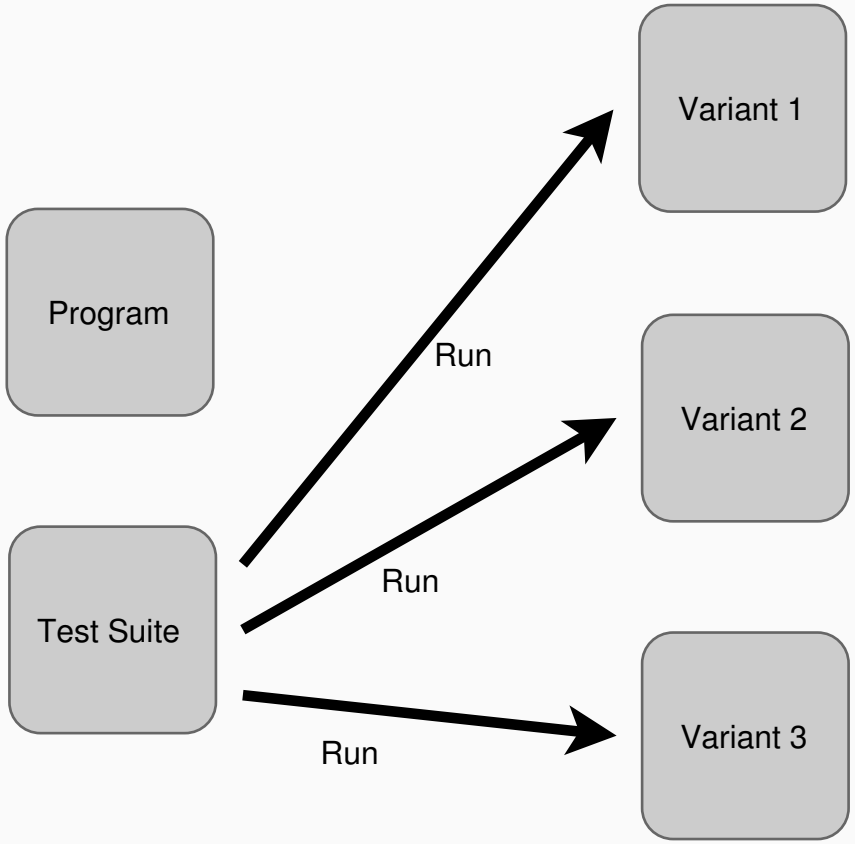
First, creates variants of program by injecting faults

Mutation score



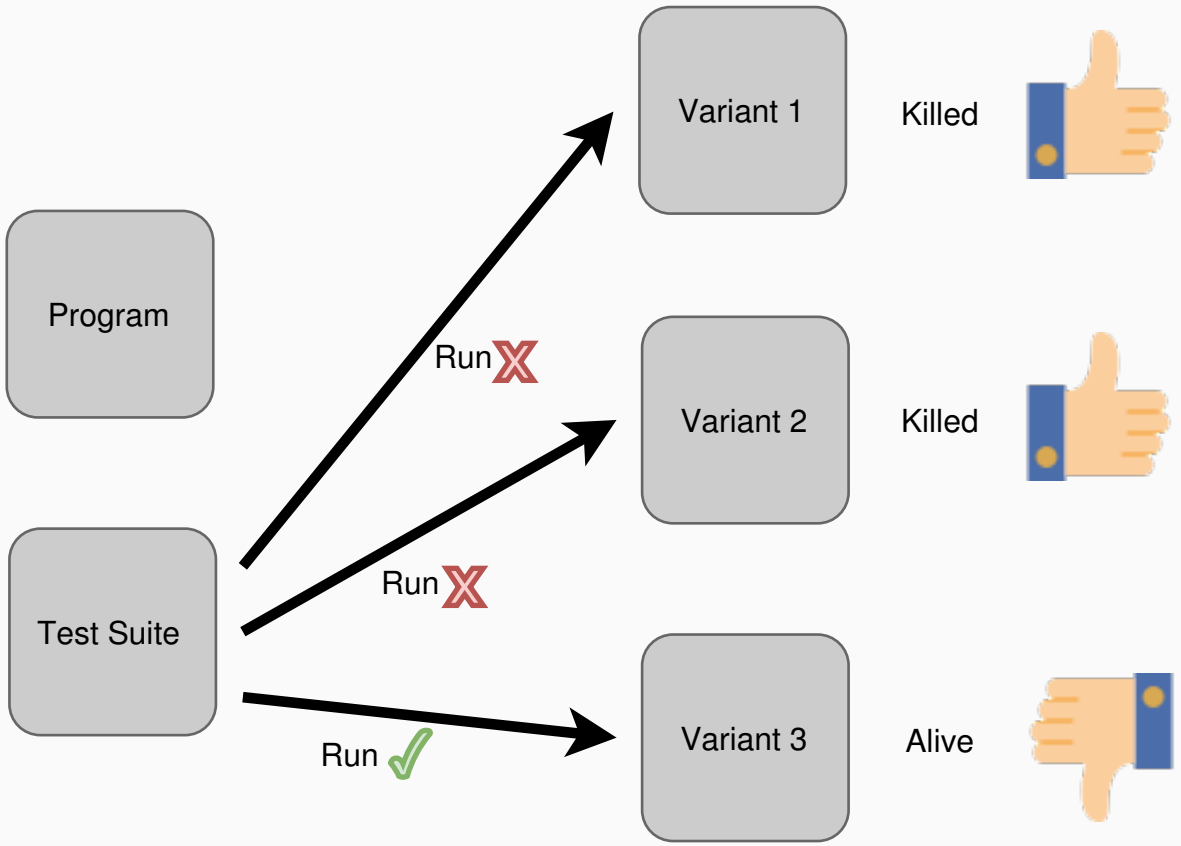
First, creates variants of program by injecting faults

Mutation score



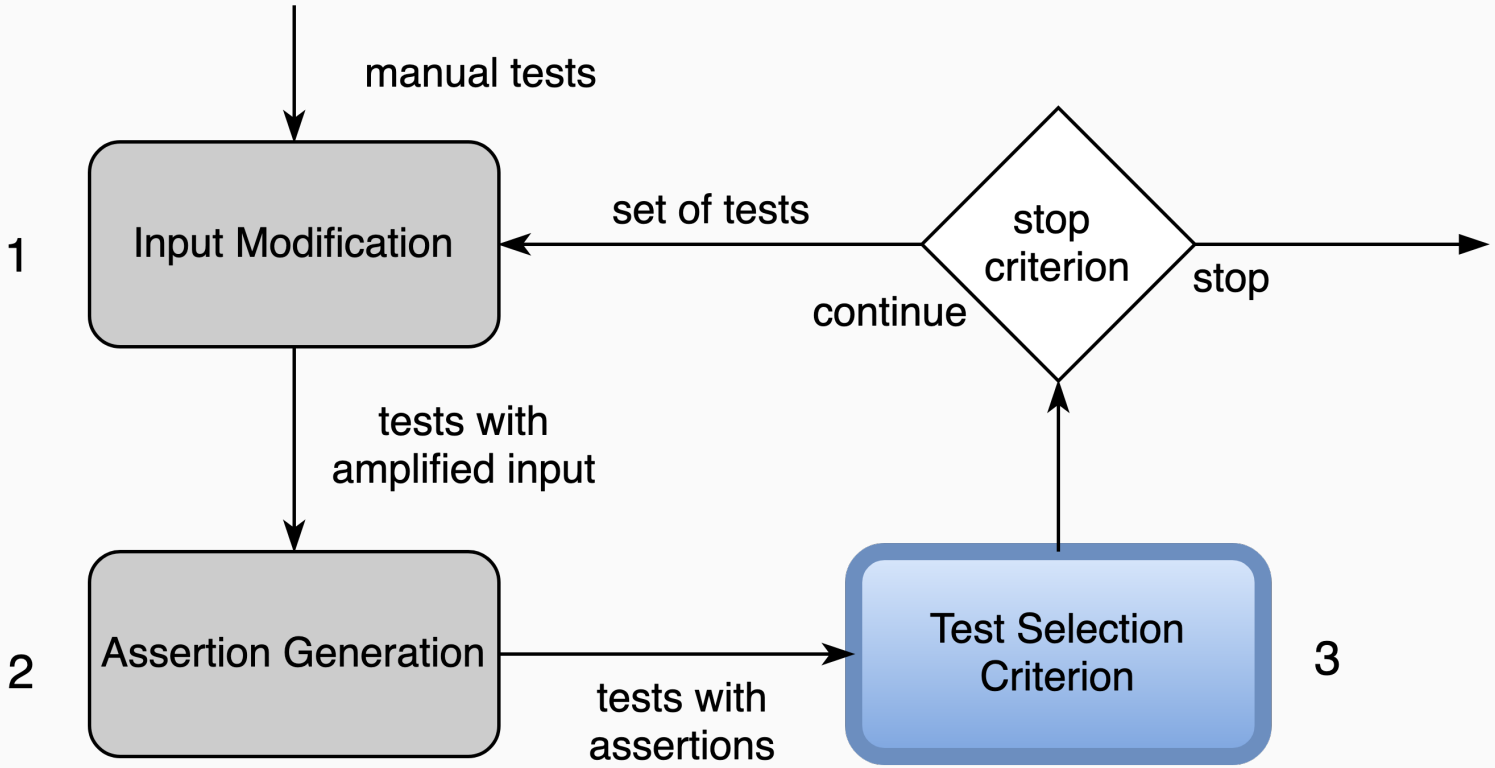
Then, runs tests against each variant

Mutation score



Failing test $\Leftrightarrow$ mutant killed $\Leftrightarrow$ behavior variation detected

DSpot: Test Selection



Mutation score as test criterion

- Coverage relies on execution
- Mutation score relies on values and oracles[7]
- Oracles are the key components to characterize behavior
- Increasing mutation score increase the coverage

[7] Offutt, A. Jefferson. "Investigations of the Software Testing Coupling Effect". *TOSEM'92*.

Mutation score as test criterion

- Coverage relies on execution
- Mutation score relies on values and oracles[7]
- Oracles are the key components to characterize behavior
- Increasing mutation score increase the coverage

Mutants emulate changes made by developers

[7] Offutt, A. Jefferson. "Investigations of the Software Testing Coupling Effect". *TOSEM'92*.

Mutation score as test criterion

- Coverage relies on execution
- Mutation score relies on values and oracles[7]
- Oracles are the key components to characterize behavior
- Increasing mutation score increase the coverage

Mutants emulate changes made by developers

Mutants are behavioral changes, such as regression

[7] Offutt, A. Jefferson. "Investigations of the Software Testing Coupling Effect". *TOSEM'92*.

Mutation score as test criterion

- Coverage relies on execution
- Mutation score relies on values and oracles[7]
- Oracles are the key components to characterize behavior
- Increasing mutation score increase the coverage

Mutants emulate changes made by developers

Mutants are behavioral changes, such as regression

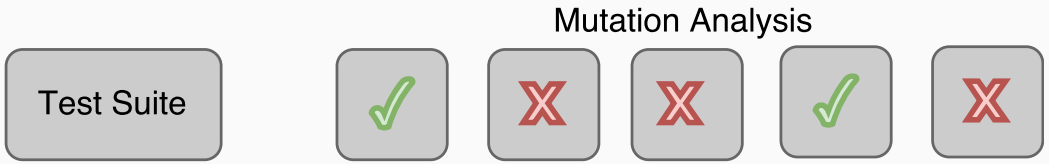
$\Rightarrow$ Approximate regressions with mutants

[7] Offutt, A. Jefferson. "Investigations of the Software Testing Coupling Effect". *TOSEM'92*.

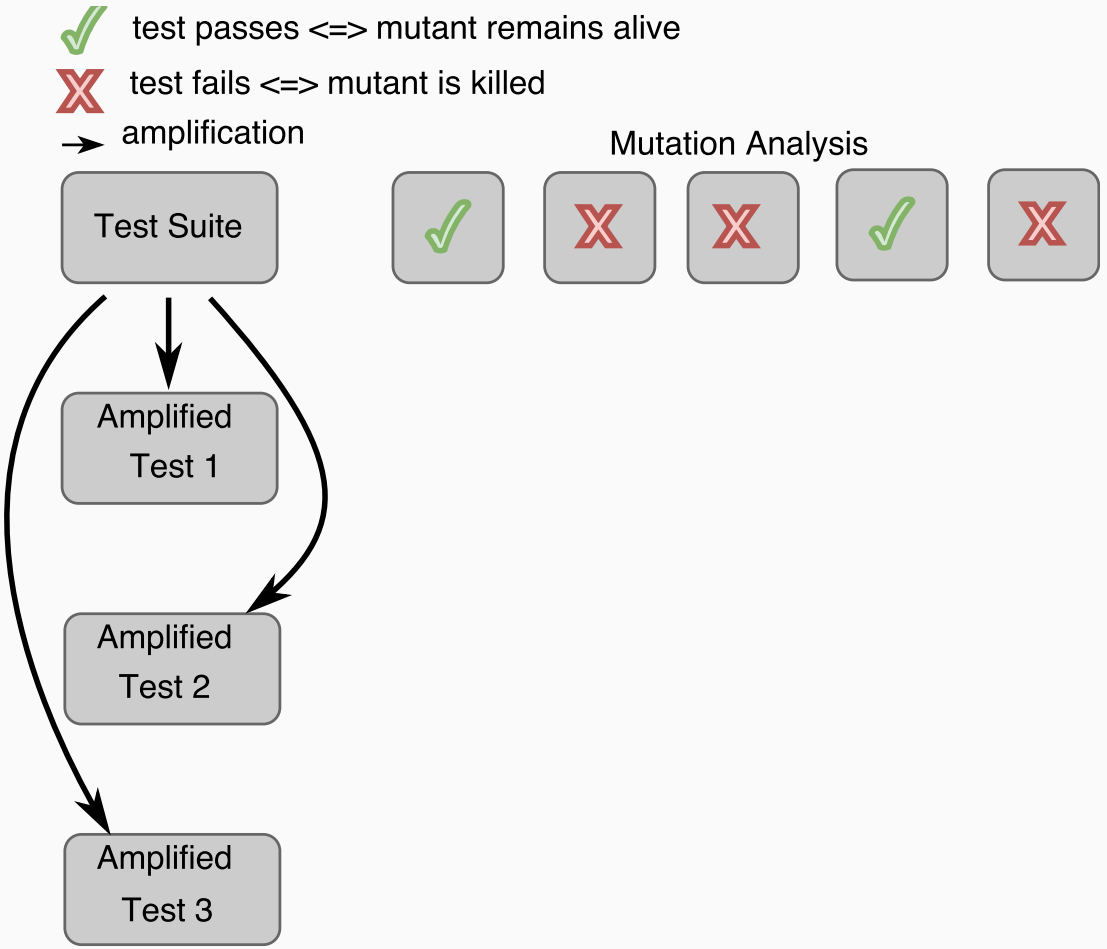
Selection Of Test Using Mutation Analysis

✓ test passes <=> mutant remains alive

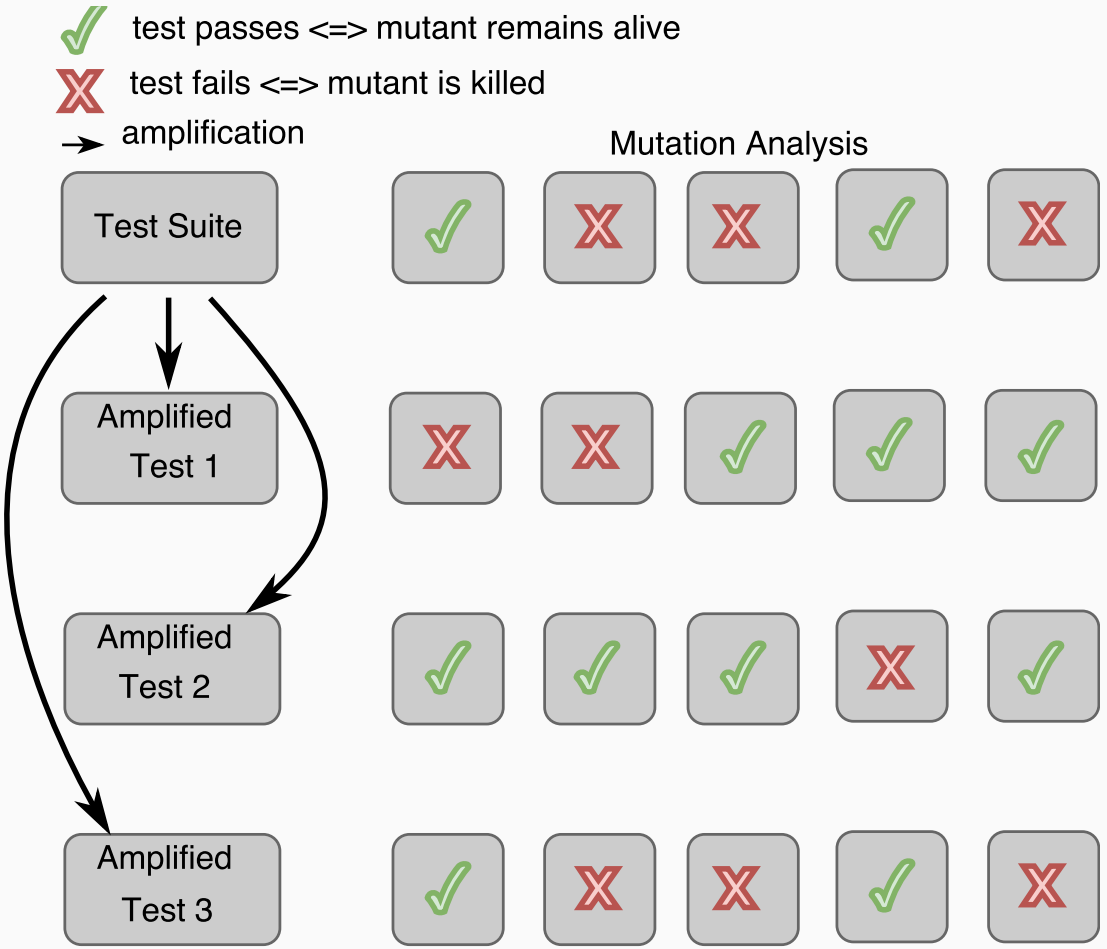
✗ test fails <=> mutant is killed



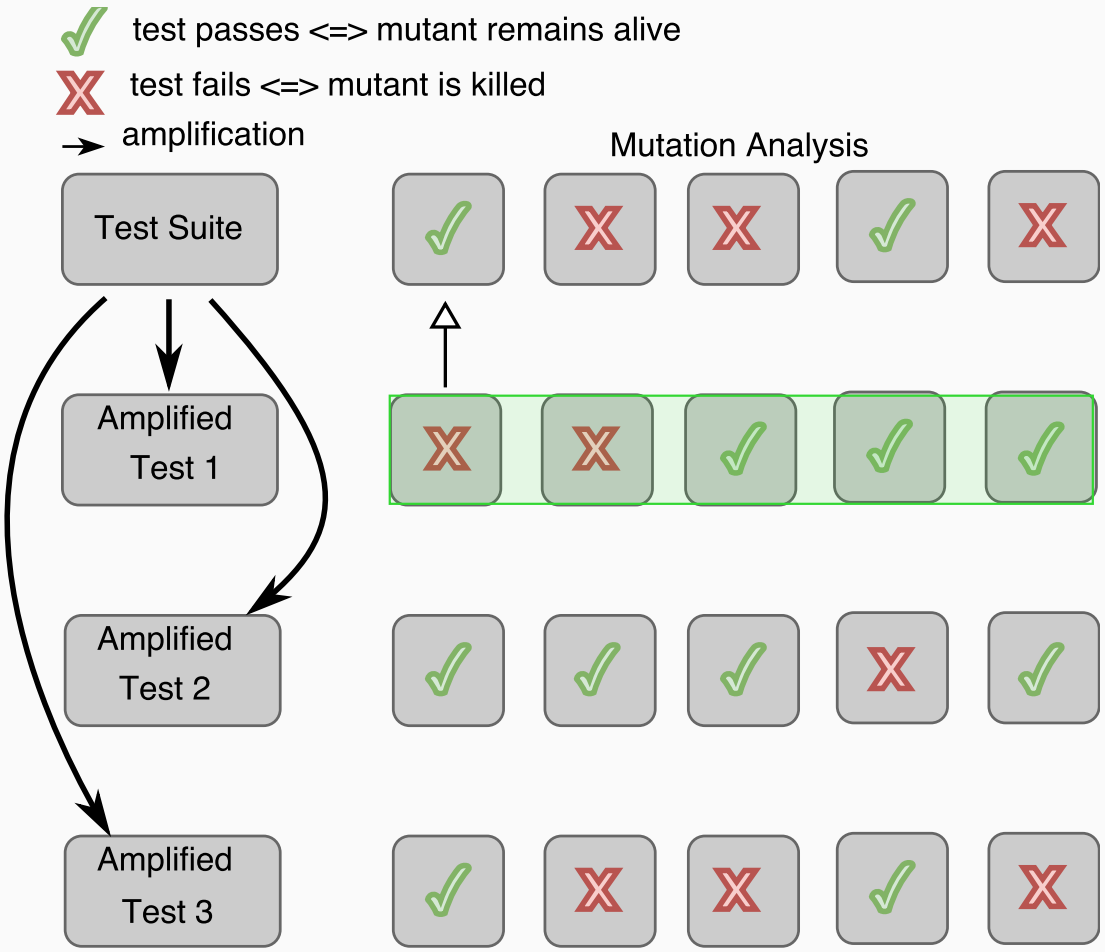
Selection Of Test Using Mutation Analysis



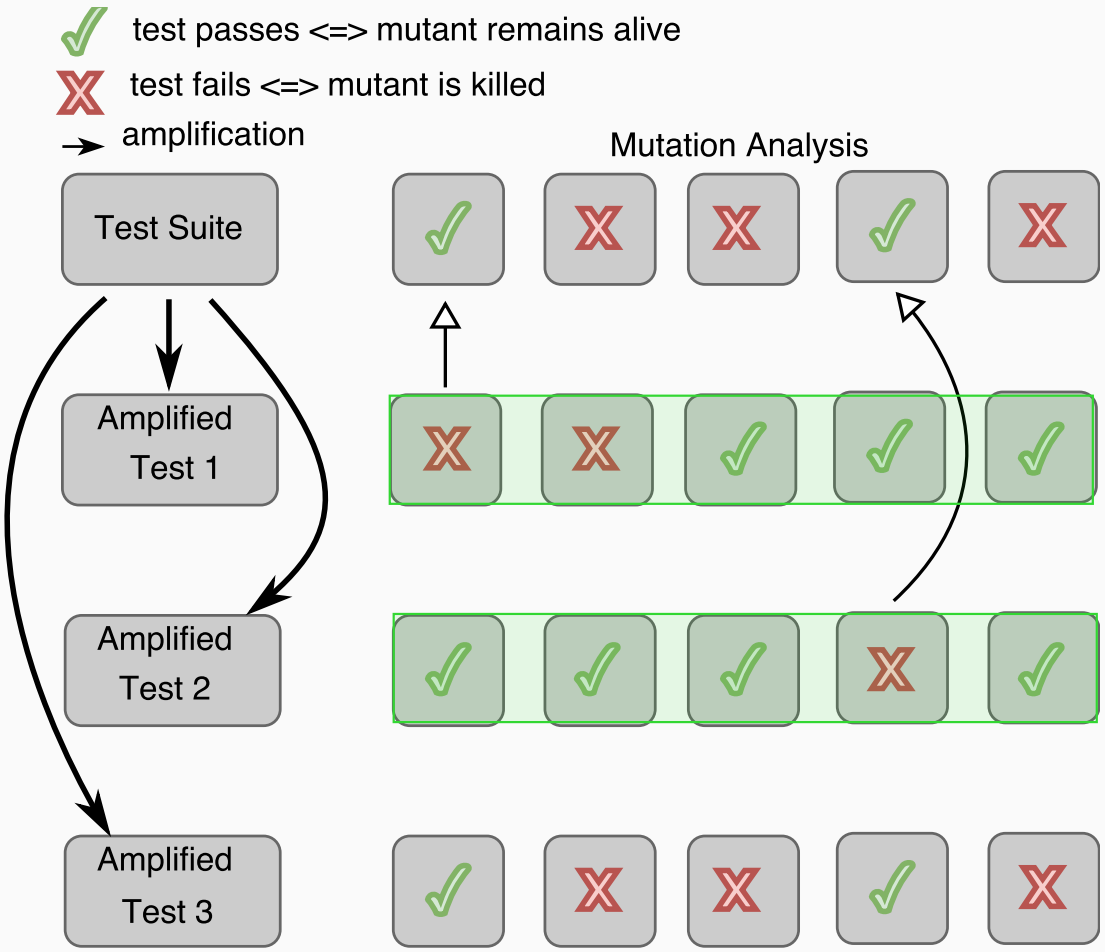
Selection Of Test Using Mutation Analysis



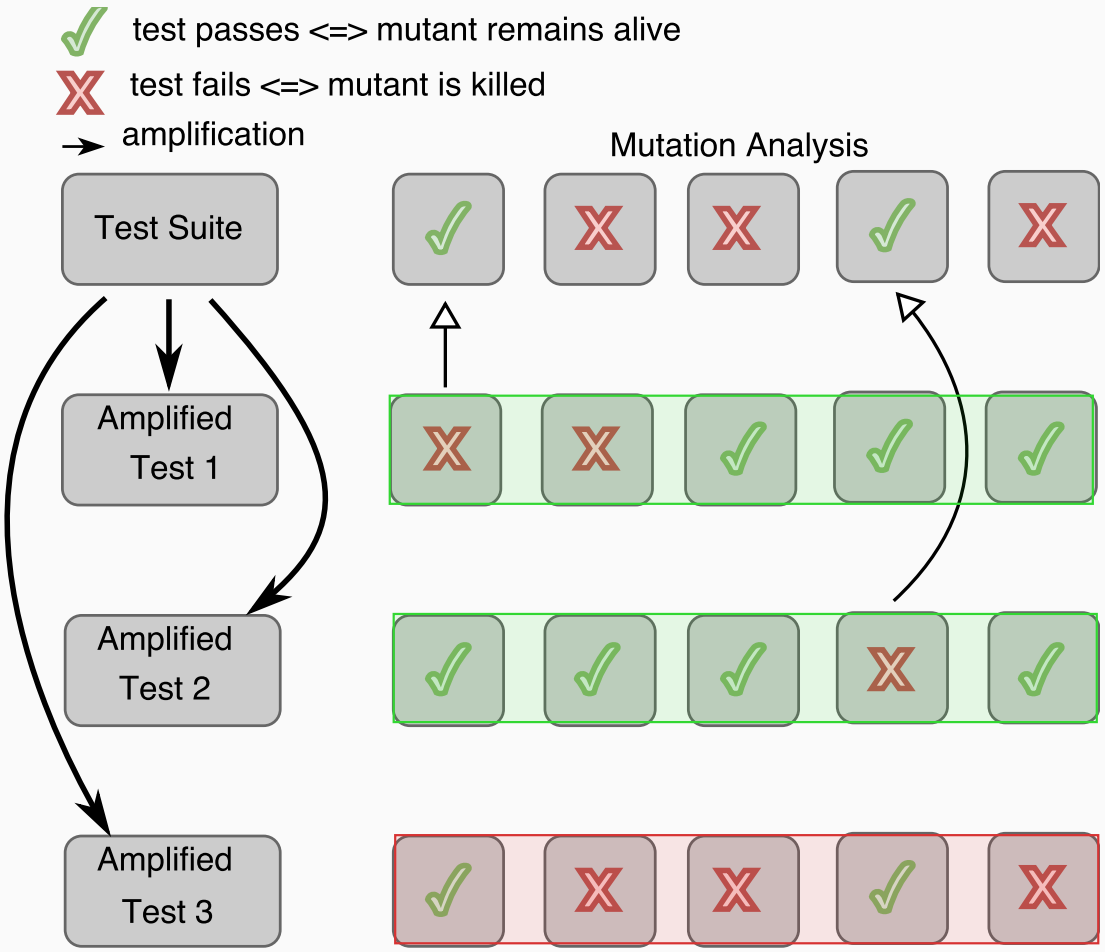
Selection Of Test Using Mutation Analysis



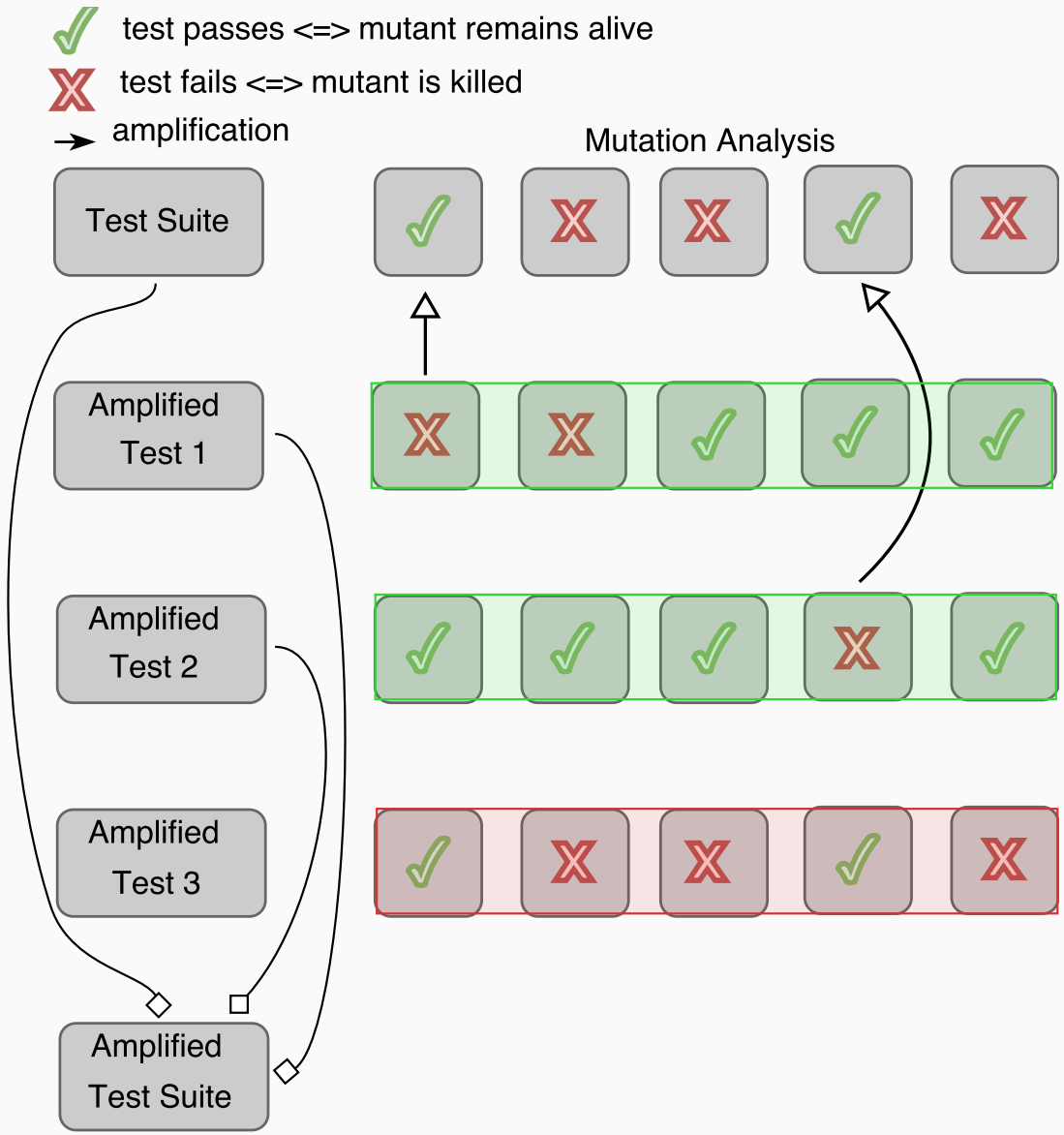
Selection Of Test Using Mutation Analysis



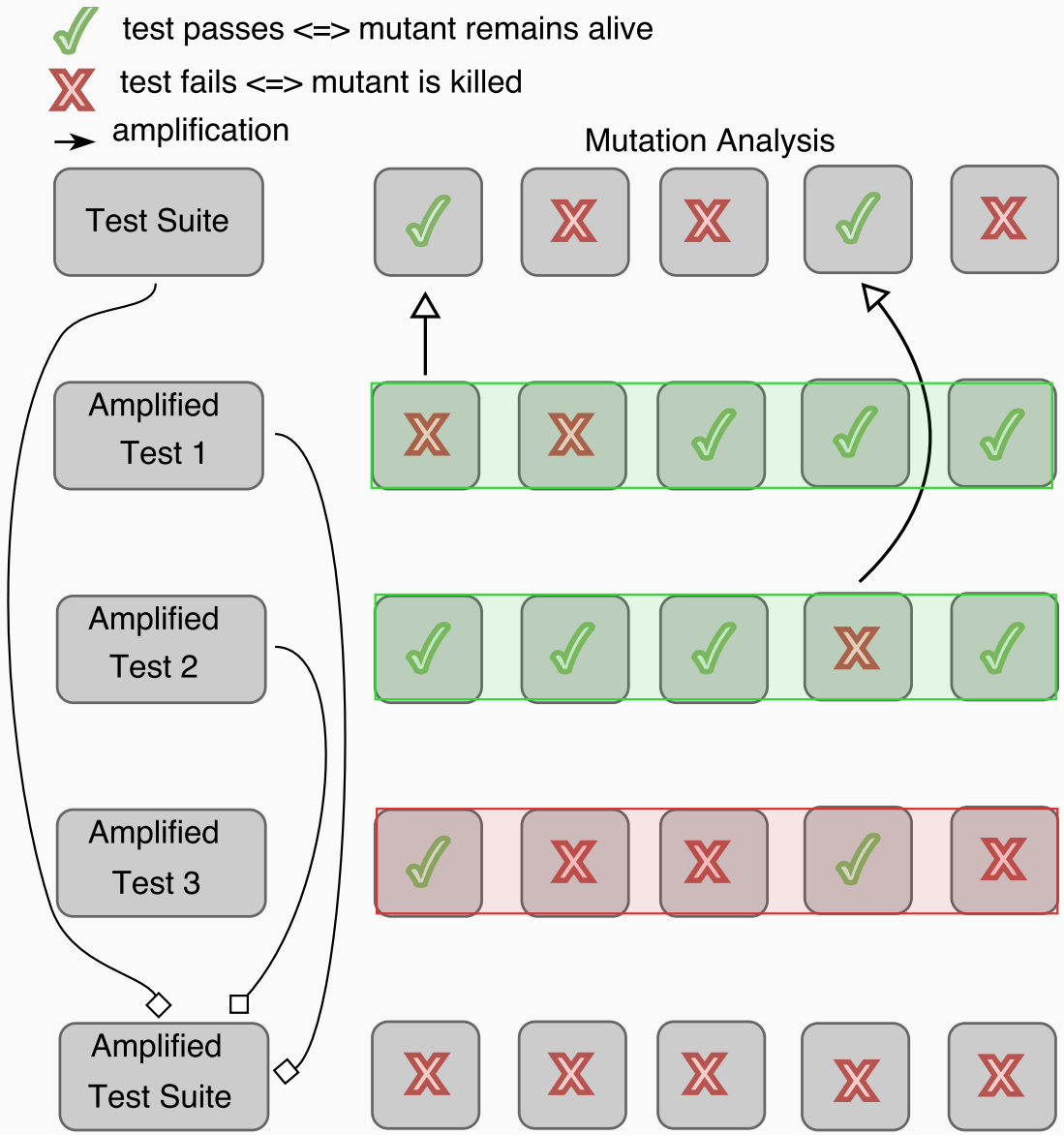
Selection Of Test Using Mutation Analysis



Selection Of Test Using Mutation Analysis



Selection Of Test Using Mutation Analysis



Evaluation: dataset

project	# LOC	# PR	# tests	mutation score (%)
javapoet	3150	93	295	84.00
mybatis-3	20683	288	1269	91.26
traccar	32648	373	229	56.94
stream-lib	4767	21	147	77.64
mustache.java	3166	11	193	81.71
twilio-java	54423	87	914	70.30
jsoup	10925	72	529	78.08
protostuff	4700	35	245	78.45
logback	15490	104	524	76.58
retrofit	2743	249	275	88.66

Evaluation: research questions

- **RQ**: Are the amplified test methods produced by DSpot relevant for developers?

[8] Fraser et al. "Does Automated Unit Test Generation Really Help Software Testers? A Controlled Empirical Study". *TOSEM'15*.

Evaluation: research questions

- **RQ**: Are the amplified test methods produced by DSpot relevant for developers?
- Do generated tests by **DSpot** help developers?[8]

[8] Fraser et al. "Does Automated Unit Test Generation Really Help Software Testers? A Controlled Empirical Study". *TOSEM'15*.

Evaluation: pull requests

project	# opened	# merged
javapoet	4	4
mybatis-3	2	2
traccar	2	1
stream-lib	1	1
mustache	2	2
twilio	2	1
jsoup	2	1
prostostuff	2	2
logback	2	0
retrofit	0	0
total	19	14

Pull-request example 1

Merged **test: specify more toJavaIdentifier** #669

Changes from **all commits** ▾ **File filter...** ▾ **Jump to...** ▾ ⚙ ▾

3 src/test/java/com/squareup/javapoet/NameAllocatorTest.java

```
25  27      NameAllocator nameAllocator = new NameAllocator();
26  28      assertThat(nameAllocator.newName("foo", 1)).isEqualTo("foo");
        @@ -59,6 +61,7 @@
59  61    @Test public void characterMappingInvalidStartButValidPart() throws Exception {
60  62      NameAllocator nameAllocator = new NameAllocator();
61  63      assertThat(nameAllocator.newName("1ab", 1)).isEqualTo("_1ab");
    64  +   assertThat(nameAllocator.newName("a-1", 2)).isEqualTo("a_1");
62  65    }
```

Pull-request example 2

Merged **test: test writeVarInt32 with large Integer** #250

Changes from **all commits** ▾ **File filter...** ▾ **Jump to...** ▾ ⚙ ▾

▾ 9 protostuff-core/src/test/java/io/protostuff/LinkBufferTest.java

```
12  12
        13  +     @Test
        14  +     public void testWriteLargeVar32() throws Exception
        15  +     {
        16  +         LinkBuffer b = new LinkBuffer(8);
        17  +         b.writeVarInt32(Integer.MAX_VALUE);
        18  +         assertEquals(1, b.getBuffers().size());
        19  +         assertEquals(5, b.getBuffers().get(0).remaining());
        20  +     }
        21  +
```

Sum-up & Take away

- Devise an algorithm to improve existing test methods according to a test criterion

Sum-up & Take away

- Devise an algorithm to improve existing test methods according to a test criterion
- Implement it in a tool called DSpot

Sum-up & Take away

- Devise an algorithm to improve existing test methods according to a test criterion
- Implement it in a tool called DSpot
- Evaluate DSpot on open-source projects using mutation score

Sum-up & Take away

- Devise an algorithm to improve existing test methods according to a test criterion
- Implement it in a tool called DSpot
- Evaluate DSpot on open-source projects using mutation score

DSpot is able to improve the mutation score of the 9 selected projects

Sum-up & Take away

- Devise an algorithm to improve existing test methods according to a test criterion
- Implement it in a tool called DSpot
- Evaluate DSpot on open-source projects using mutation score

DSpot is able to improve the mutation score of the 9 selected projects

Resulting test methods has been proposed and accepted as pull request on these projects

Recall: Lack in state of the art

- No evaluation including developers has been carried out
  ✗ need for assessment from real **developers** assessment
- Lack of study on regression detection improvement
  ✗ need for an approach to improve **detection of regressions**
- Lack of generalizable results
  ✗ need for **benchmarks of real programs**

Recall: Lack in state of the art

- ~~No evaluation including developers has been carried out~~
  ✓ need for assessment from real **developers** assessment
- Lack of study on regression detection improvement
  ✗ need for an approach to improve **detection of regressions**
- Lack of generalizable results
  ✗ need for **benchmarks of real programs**

Lessons learnt form this first evaluation

[9] DeMillo, R. A., Lipton, R. J., and Sayward, F. G. "Hints on Test Data Selection: Help for the Practicing Programmer". *Computer'78*.

[10] Offutt, A. Jefferson. "Investigations of the Software Testing Coupling Effect". *TOSEM'92*.

Lessons learnt form this first evaluation

Mutants are temporary, small and artificial behavioral changes[9,10]

[9] DeMillo, R. A., Lipton, R. J., and Sayward, F. G. "Hints on Test Data Selection: Help for the Practicing Programmer". *Computer'78*.

[10] Offutt, A. Jefferson. "Investigations of the Software Testing Coupling Effect". *TOSEM'92*.

Lessons learnt form this first evaluation

Mutants are temporary, small and artificial behavioral changes[9,10]

Problem: Is mutation score enough to improve regression detection?

[9] DeMillo, R. A., Lipton, R. J., and Sayward, F. G. "Hints on Test Data Selection: Help for the Practicing Programmer". *Computer'78*.
[10] Offutt, A. Jefferson. "Investigations of the Software Testing Coupling Effect". *TOSEM'92*.

Lessons learnt form this first evaluation

Mutants are temporary, small and artificial behavioral changes[9,10]

Problem: Is mutation score enough to improve regression detection?

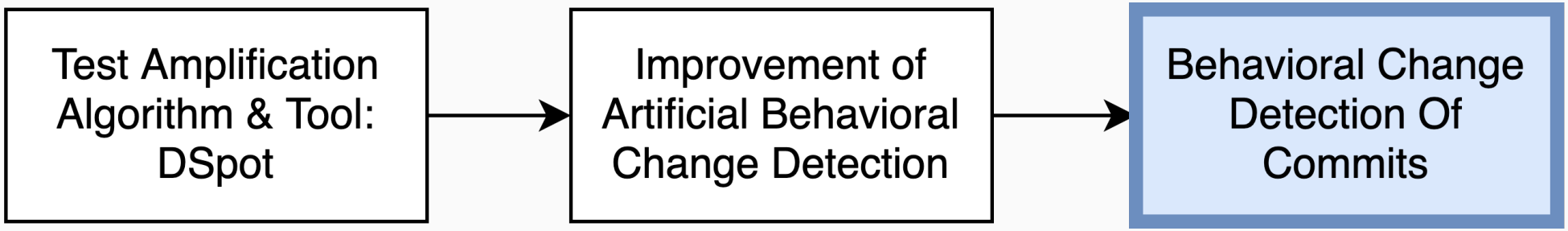
What is the performance of DSpot on more complex and real behavioral changes?

[9] DeMillo, R. A., Lipton, R. J., and Sayward, F. G. "Hints on Test Data Selection: Help for the Practicing Programmer". *Computer'78*.

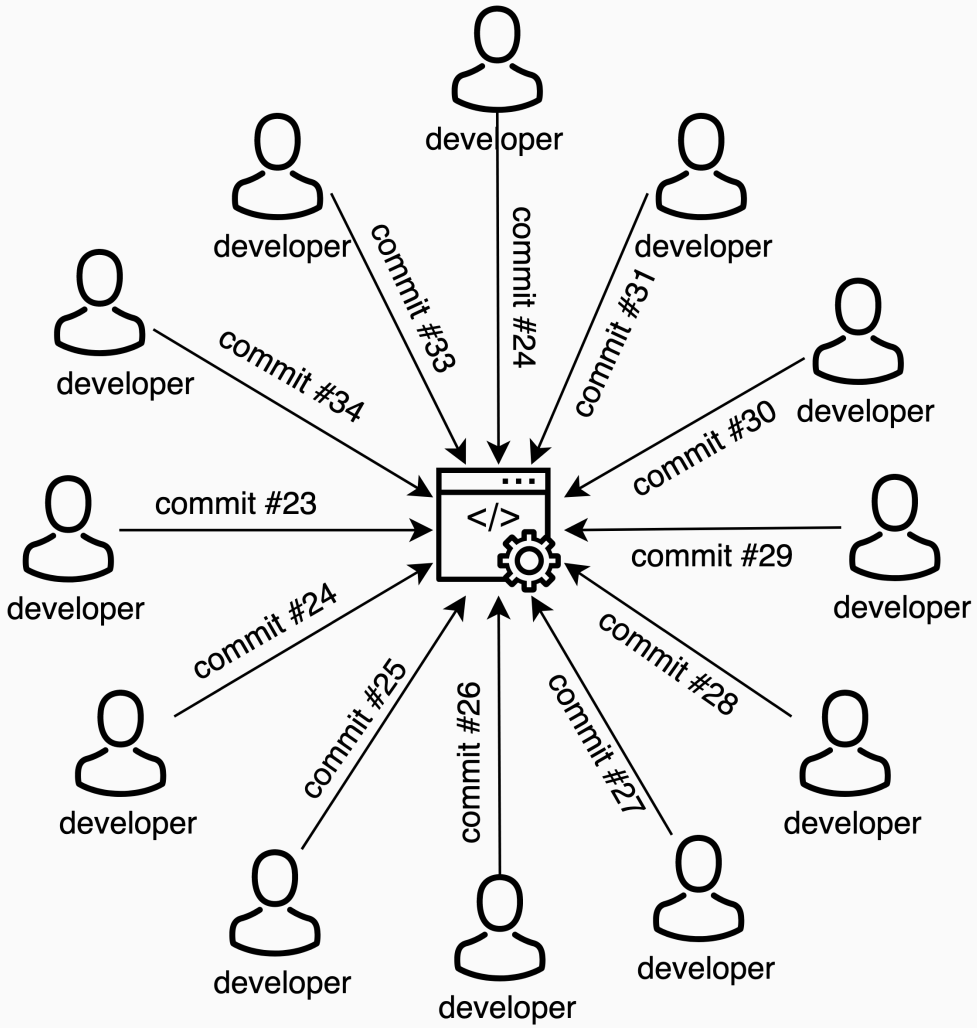
[10] Offutt, A. Jefferson. "Investigations of the Software Testing Coupling Effect". *TOSEM'92*.

Behavioral Change Detection Of Commits

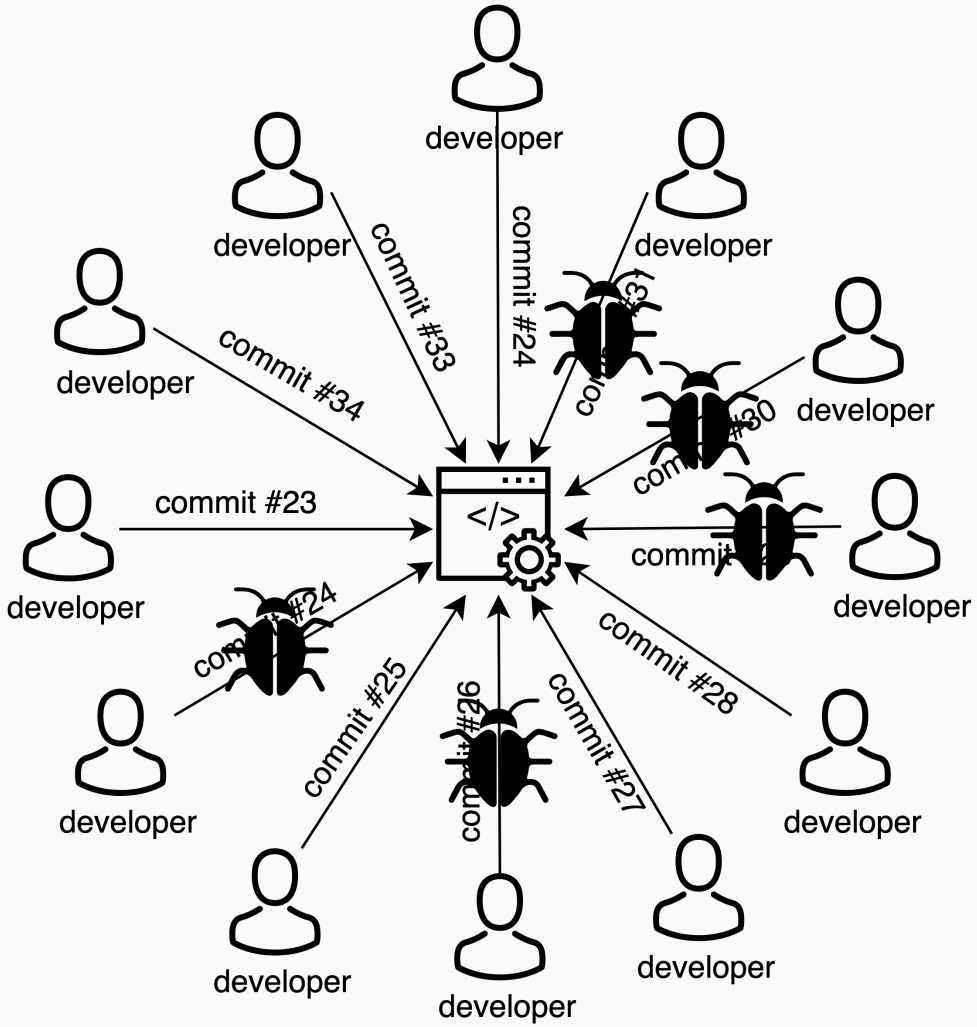
Outline


Test Amplification Algorithm & Tool: DSpot
Improvement of Artificial Behavioral Change Detection
Behavioral Change Detection Of Commits

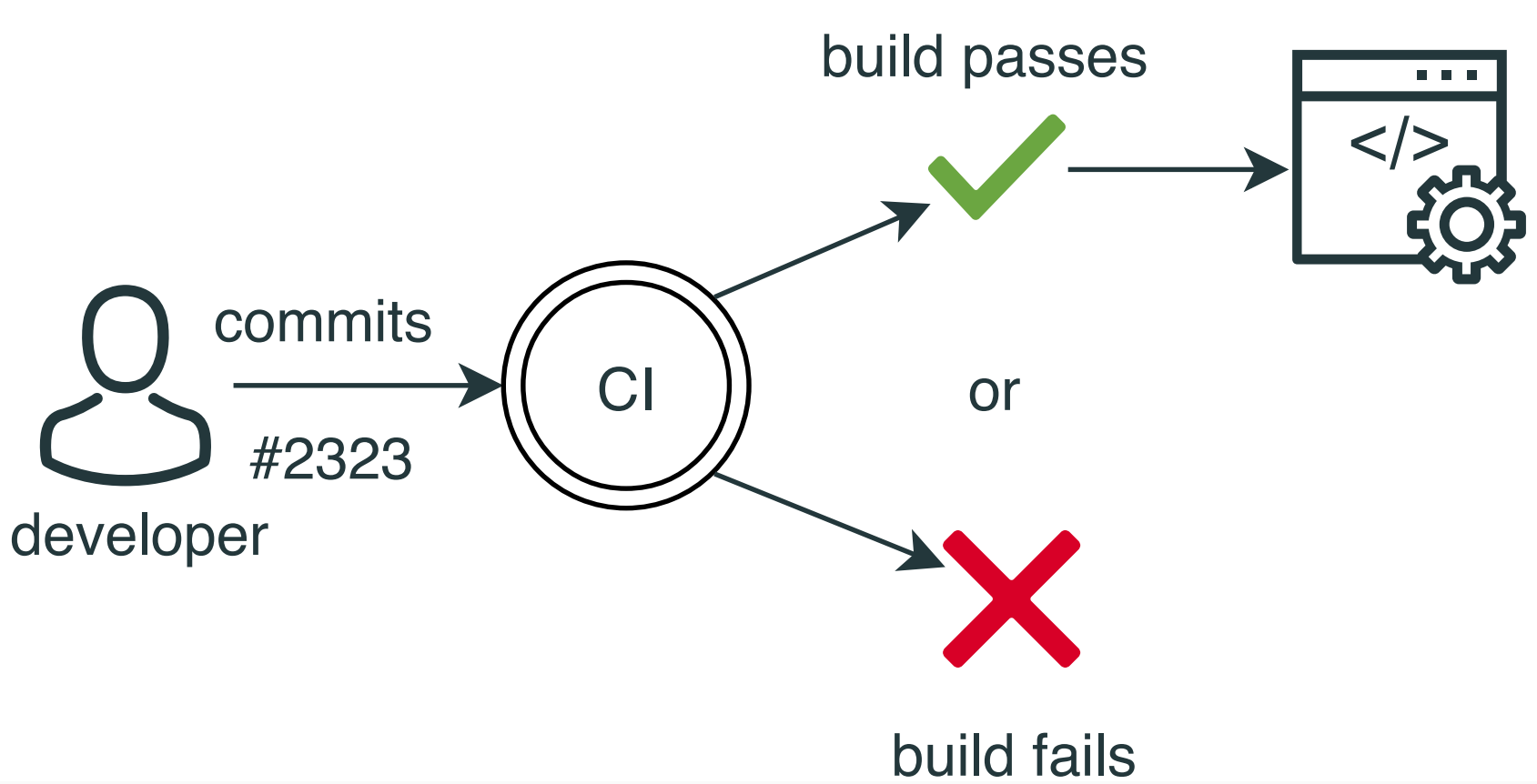
Introduction: Use case example



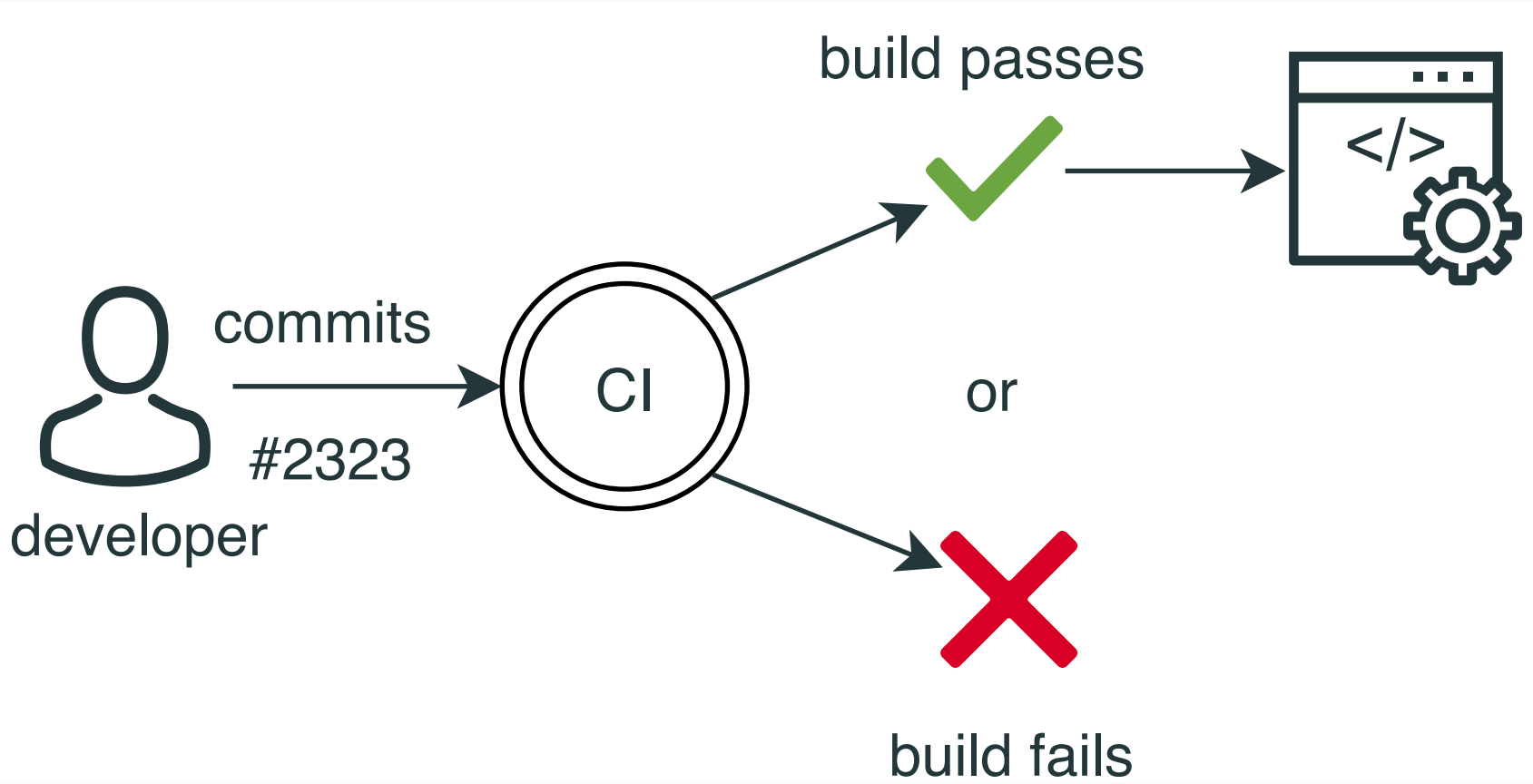
Introduction: Use case example



Introduction: Use case example



Introduction: Use case example



Problems:

- Is the test suite capable of detecting the behavioral change?
- Do developers offer a test that encode the behavioral change?

Introduction: real example

tmortagne committed on 10 Aug 2018 1 parent d10daeb commit 7e79f774a58f15897e49600

Showing **1 changed file** with **2 additions** and **1 deletion**.

3 ...filter/xwiki-commons-filter-api/src/main/java/org/xwiki/filter/type/FilterStreamType.java

```
@@ -260,7 +260,8 @@ public boolean equals(Object object)
260 260             } else {
261 261                 if (object instanceof FilterStreamType) {
262 262                     result = Objects.equals(getType(), ((FilterStreamType) object).getType())
263     -                       && Objects.equals(getDataFormat(), ((FilterStreamType) object).getDataFormat());
    263 +                       && Objects.equals(getDataFormat(), ((FilterStreamType) object).getDataFormat())
    264 +                       && Objects.equals(getVersion(), ((FilterStreamType) object).getVersion());
264 265                 } else {
265 266                     result = false;
266 267                 }
```

Introduction: real example

tmortagne committed on 10 Aug 2018 1 parent d10daeb commit 7e79f774a58f15897e49600

Showing **1 changed file** with **2 additions** and **1 deletion**.

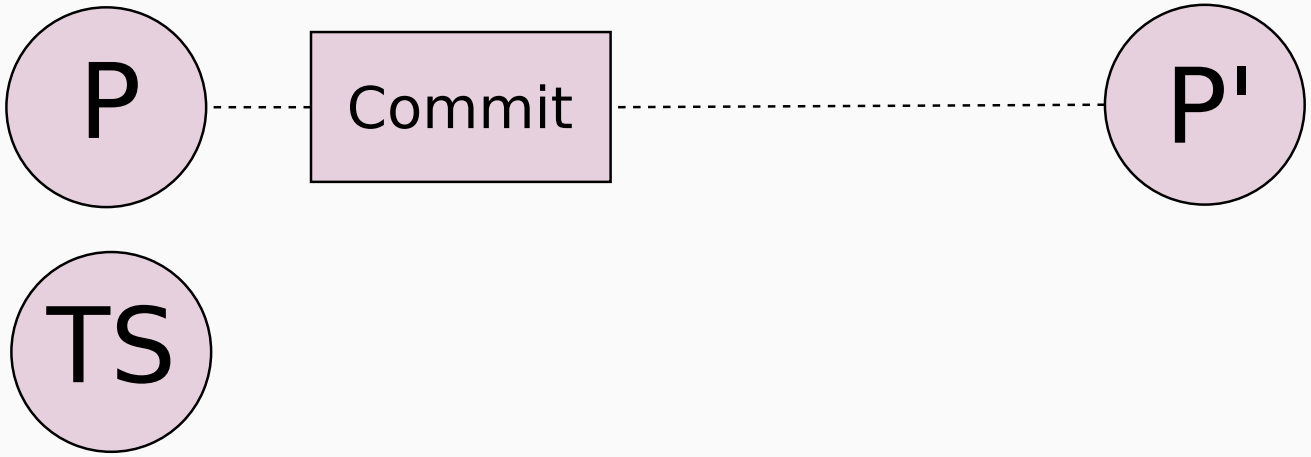
3 ...filter/xwiki-commons-filter-api/src/main/java/org/xwiki/filter/type/FilterStreamType.java

```
@@ -260,7 +260,8 @@ public boolean equals(Object object)
260 260             } else {
261 261                 if (object instanceof FilterStreamType) {
262 262                     result = Objects.equals(getType(), ((FilterStreamType) object).getType())
263     -                        && Objects.equals(getDataFormat(), ((FilterStreamType) object).getDataFormat());
    263 +                        && Objects.equals(getDataFormat(), ((FilterStreamType) object).getDataFormat())
    264 +                        && Objects.equals(getVersion(), ((FilterStreamType) object).getVersion());
264 265                 } else {
265 266                     result = false;
266 267                 }
```

What if this is reverted in the future?

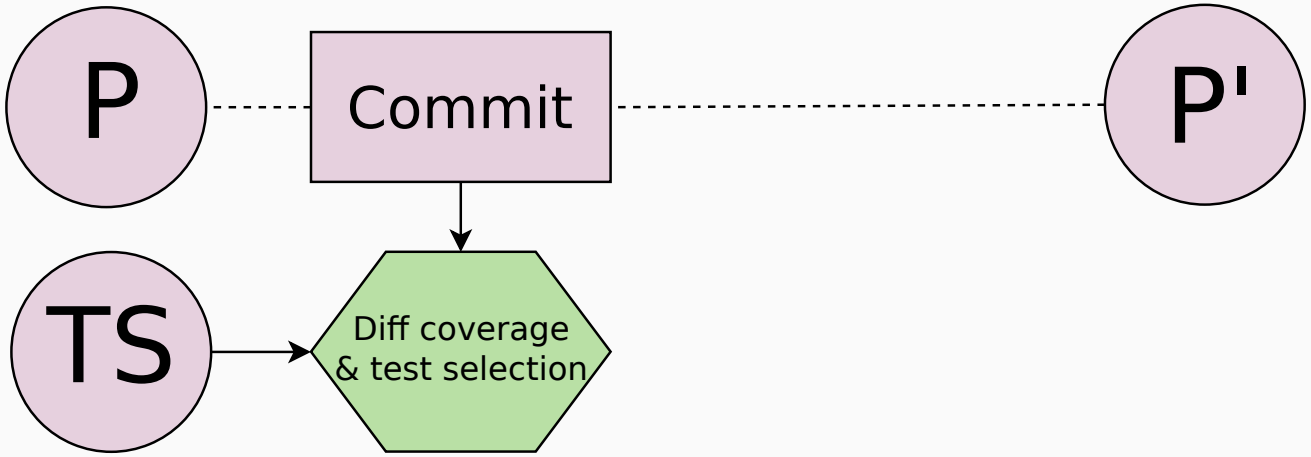
Proposed approach

Goal: Amplify test methods to detect behavioral changes



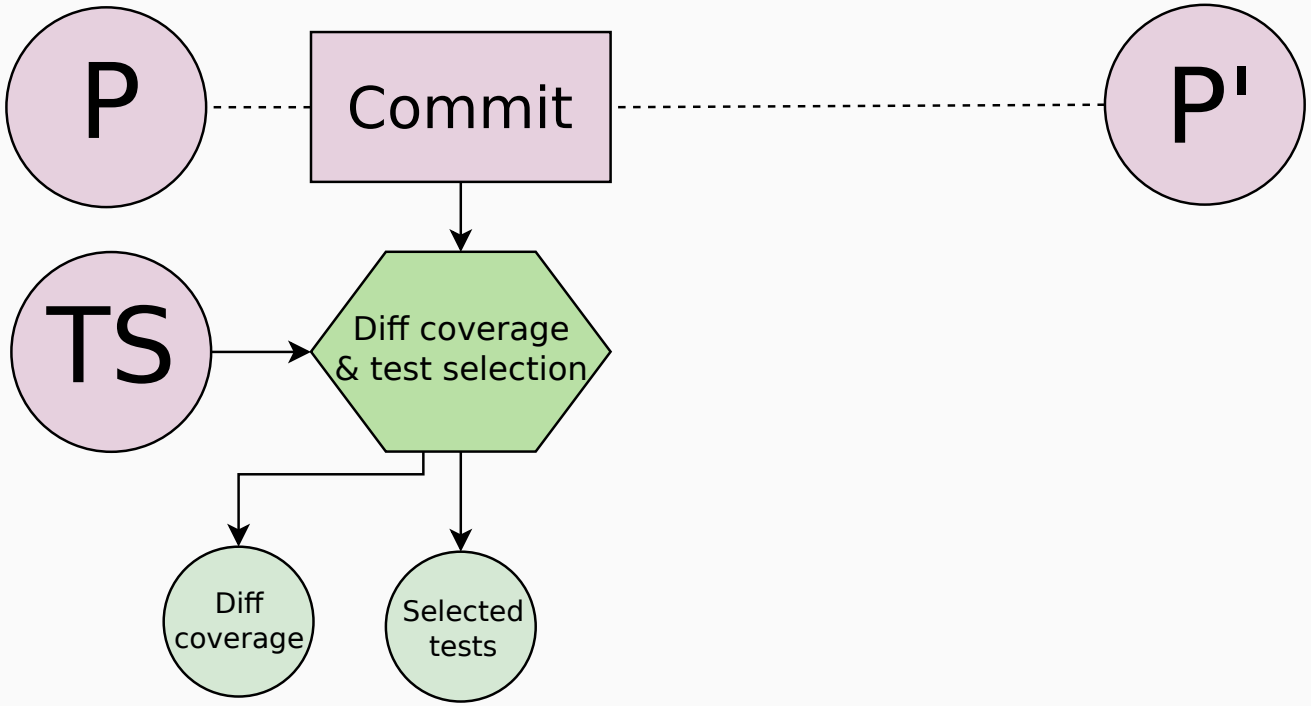
Proposed approach

Goal: Amplify test methods to detect behavioral changes



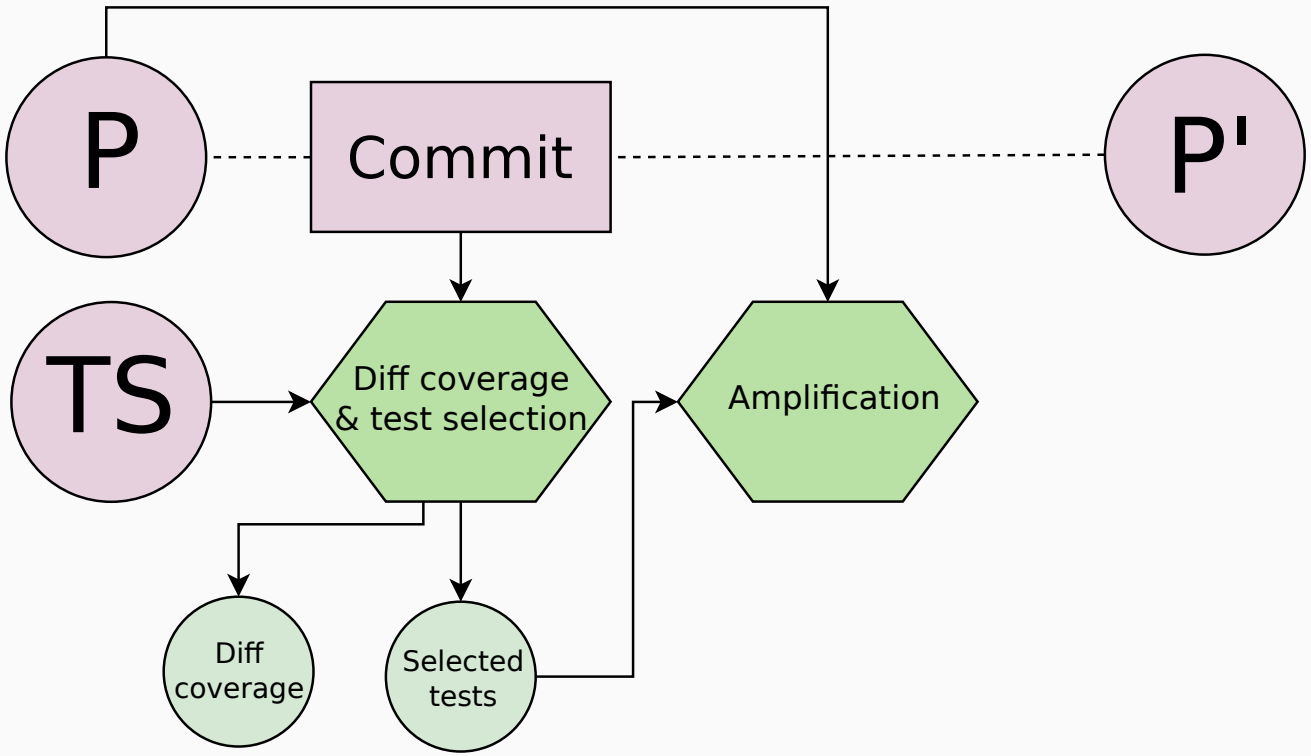
Proposed approach

Goal: Amplify test methods to detect behavioral changes



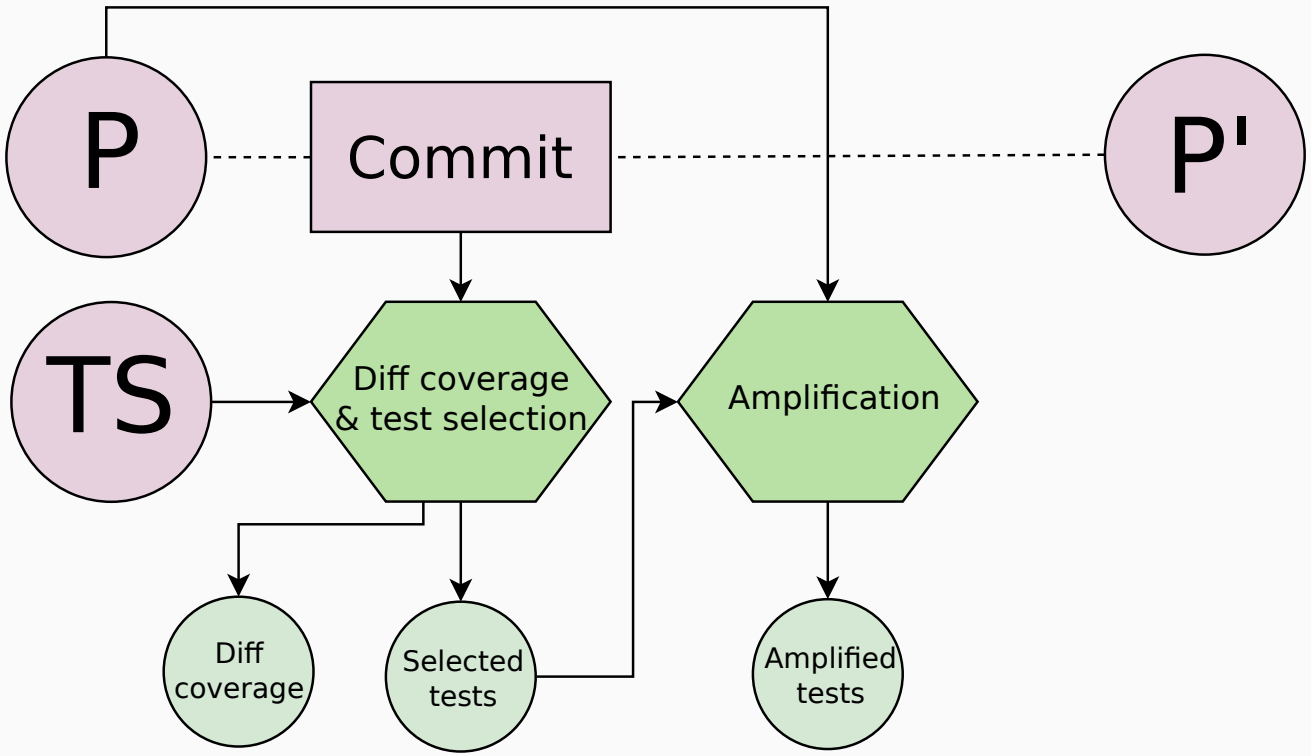
Proposed approach

Goal: Amplify test methods to detect behavioral changes



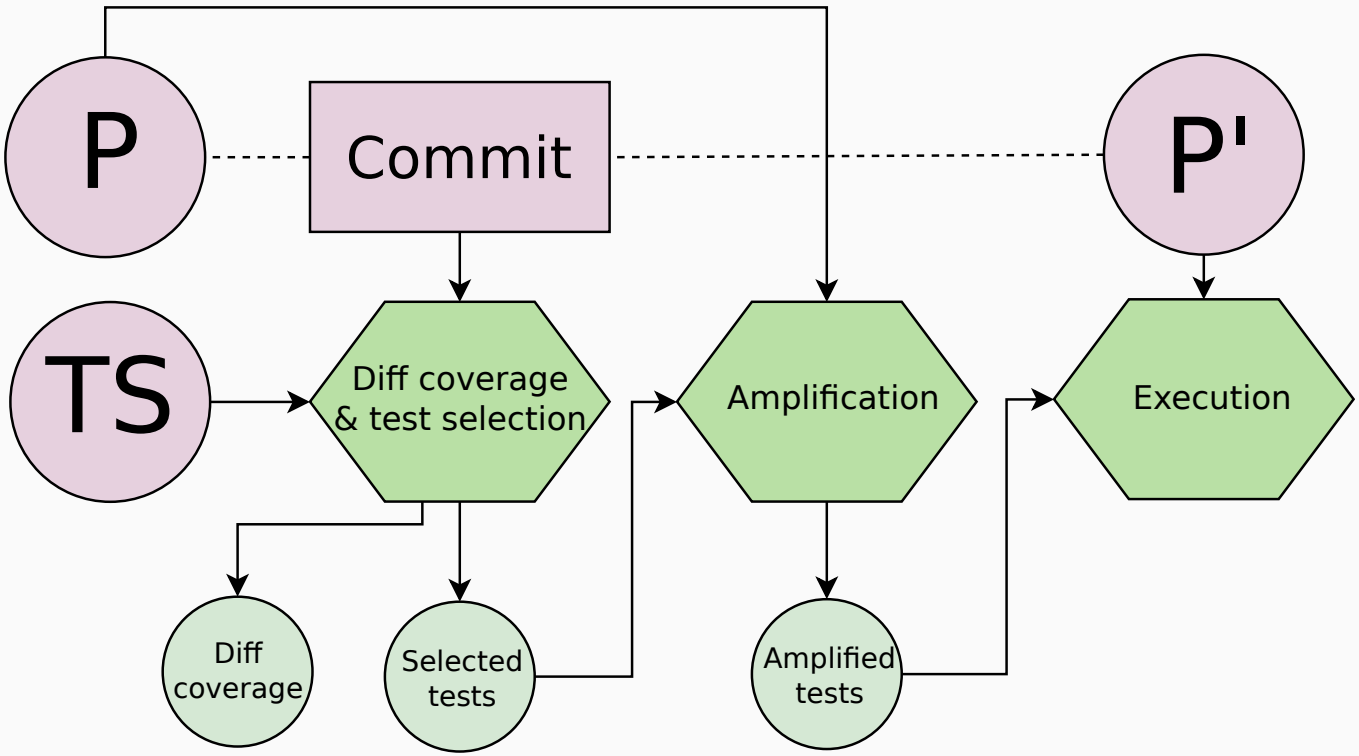
Proposed approach

Goal: Amplify test methods to detect behavioral changes



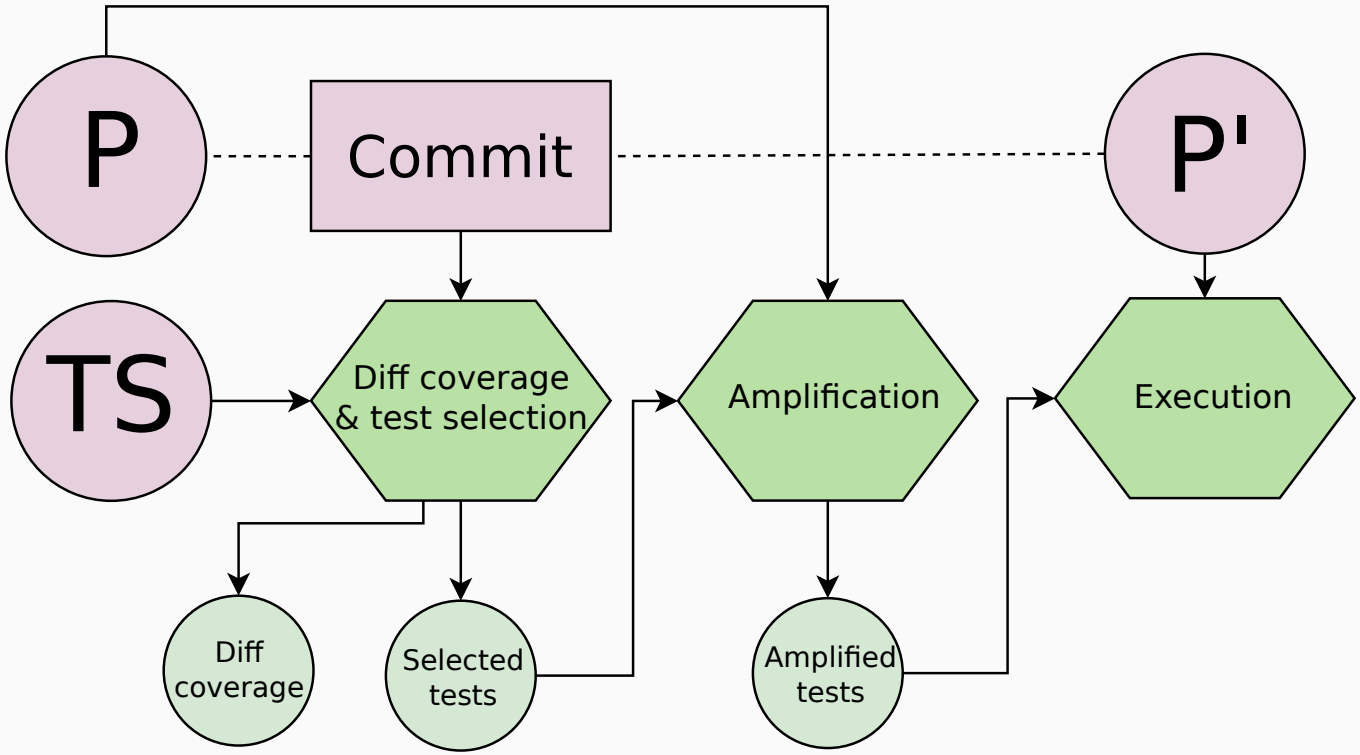
Proposed approach

Goal: Amplify test methods to detect behavioral changes



Proposed approach

Goal: Amplify test methods to detect behavioral changes



✓ all amplified tests pass = behavioral change **un**detected

✗ at least one amplified test fails = behavioral change detected

Proposed approach: output usage

✗ at least one amplified test fails = behavioral change detected

Proposed approach: output usage

✗ at least one amplified test fails = behavioral change detected

1. The amplified test method detects the desired behavioral change

Proposed approach: output usage

✗ at least one amplified test fails = behavioral change detected

1. The amplified test method detects the desired behavioral change
   ⇒ negate the assertions to have a test that specify the new behavior

Proposed approach: output usage

✗ at least one amplified test fails = behavioral change detected

1. The amplified test method detects the desired behavioral change
   ⇒ negate the assertions to have a test that specify the new behavior
2. The amplified test method detects an **un**desired behavioral change

Proposed approach: output usage

✗ at least one amplified test fails = behavioral change detected

1. The amplified test method detects the desired behavioral change
   ⇒ negate the assertions to have a test that specify the new behavior
2. The amplified test method detects an **un**desired behavioral change
   ⇒ the developer has now a test to fix a potential regression or a new bug

Evaluation: research question

RQ: To what extent DSpot is able to obtain amplified test methods that detect the behavioral changes introduced by commits?

Benchmark

project	LOC	#total commits	#matching commits	#selected commits
commons-io	59607	385	49(12%)	10
commons-lang	77410	227	40(17%)	10
gson	49766	159	56(35%)	10
jsoup	20088	50	42(84%)	10
mustache.java	10289	68	28(41%)	10
xwiki-commons	87289	687	26(3%)	10

Commits selection criretia

#Matching commit $\Leftrightarrow$ Commit on which the approach can be applied

Commits selection criretia

#Matching commit $\Leftrightarrow$ Commit on which the approach can be applied

- The change modifies a java file

Commits selection criretia

#Matching commit $\Leftrightarrow$ Commit on which the approach can be applied

- The change modifies a java file
- There is at least one test method that executes the change

Commits selection criretia

#Matching commit $\Leftrightarrow$ Commit on which the approach can be applied

- The change modifies a java file
- There is at least one test method that executes the change

#Selected commit: 10 first commits that match a third criterion:

Commits selection criretia

#Matching commit $\Leftrightarrow$ Commit on which the approach can be applied

- The change modifies a java file
- There is at least one test method that executes the change

#Selected commit: 10 first commits that match a third criterion:

- A test of P' is failing when executed on P

Benchmark

project	LOC	#total commits	#matching commits	#selected commits
commons-io	59607	385	49(12%)	10
commons-lang	77410	227	40(17%)	10
gson	49766	159	56(35%)	10
jsoup	20088	50	42(84%)	10
mustache.java	10289	68	28(41%)	10
xwiki-commons	87289	687	26(3%)	10

Result

project	#Detection	Average Time
commons-io	5/10	38.8m
commons-lang	3/10	74.7m
gson	4/10	86.5m
jsoup	3/10	2.6h
mustache.java	6/10	4.2h
xwiki-commons	4/10	49.5m
total	25/60	-

Sum-up & Take away

- Set up an approach to amplify the tests to detect behavioral change

Sum-up & Take away

- Set up an approach to amplify the tests to detect behavioral change
- Implement it using DSpot

Sum-up & Take away

- Set up an approach to amplify the tests to detect behavioral change
- Implement it using DSpot
- Build a benchmark of 6 projects and 60 commits

Sum-up & Take away

- Set up an approach to amplify the tests to detect behavioral change
- Implement it using DSpot
- Build a benchmark of 6 projects and 60 commits
- Apply the approach on this benchmark

Sum-up & Take away

- Set up an approach to amplify the tests to detect behavioral change
- Implement it using DSpot
- Build a benchmark of 6 projects and 60 commits
- Apply the approach on this benchmark

DSpot is able to amplified tests to detect behavioral change

Sum-up & Take away

- Set up an approach to amplify the tests to detect behavioral change
- Implement it using DSpot
- Build a benchmark of 6 projects and 60 commits
- Apply the approach on this benchmark

DSpot is able to amplified tests to detect behavioral change

This is meant to be used in the CI to:

Sum-up & Take away

- Set up an approach to amplify the tests to detect behavioral change
- Implement it using DSpot
- Build a benchmark of 6 projects and 60 commits
- Apply the approach on this benchmark

DSpot is able to amplified tests to detect behavioral change

This is meant to be used in the CI to:

1. **Specify automatically a new behavior**

Sum-up & Take away

- Set up an approach to amplify the tests to detect behavioral change
- Implement it using DSpot
- Build a benchmark of 6 projects and 60 commits
- Apply the approach on this benchmark

DSpot is able to amplified tests to detect behavioral change

This is meant to be used in the CI to:

1. **Specify automatically a new behavior**
2. **Improve the detection of regression**

Lack in state of the art

- ~~No evaluation including developers has been carried out~~
  ✓ need for assessment from real **developers** assessment
- Lack of study on regression detection improvement
  ✗ need for an approach to improve **detection of regressions**
- Lack of generalizable results
  ✗ need for **benchmarks of real programs**

Lack in state of the art

- ~~No evaluation including developers has been carried out~~
  ✓ need for assessment from real **developers** assessment
- ~~Lack of study on regression detection improvement~~
  ✓ need for an approach to improve **detection of regressions**
- Lack of generalizable results
  ✗ need for **benchmarks of real programs**

Lack in state of the art

- ~~No evaluation including developers has been carried out~~
  ✓ need for assessment from real **developers** assessment
- ~~Lack of study on regression detection improvement~~
  ✓ need for an approach to improve **detection of regressions**
- ~~Lack of generalizable results~~
  ✓ need for **benchmarks of real programs**

Conclusion

Publications

Main contributions:

Benjamin Danglot et al. “A snowballing literature study on test amplification”. In: *Journal of Systems and Software* (2019)

Benjamin Danglot et al. “Automatic test improvement with DSpot: a study with ten mature open-source projects”. In: *Empirical Software Engineering* (2019)

Benjamin Danglot et al. “An Approach and Benchmark to Detect Behavioral Changes of Commits in Continuous Integration”. In: *minor revision @ EMSE* (2019)

Transversal contributions:

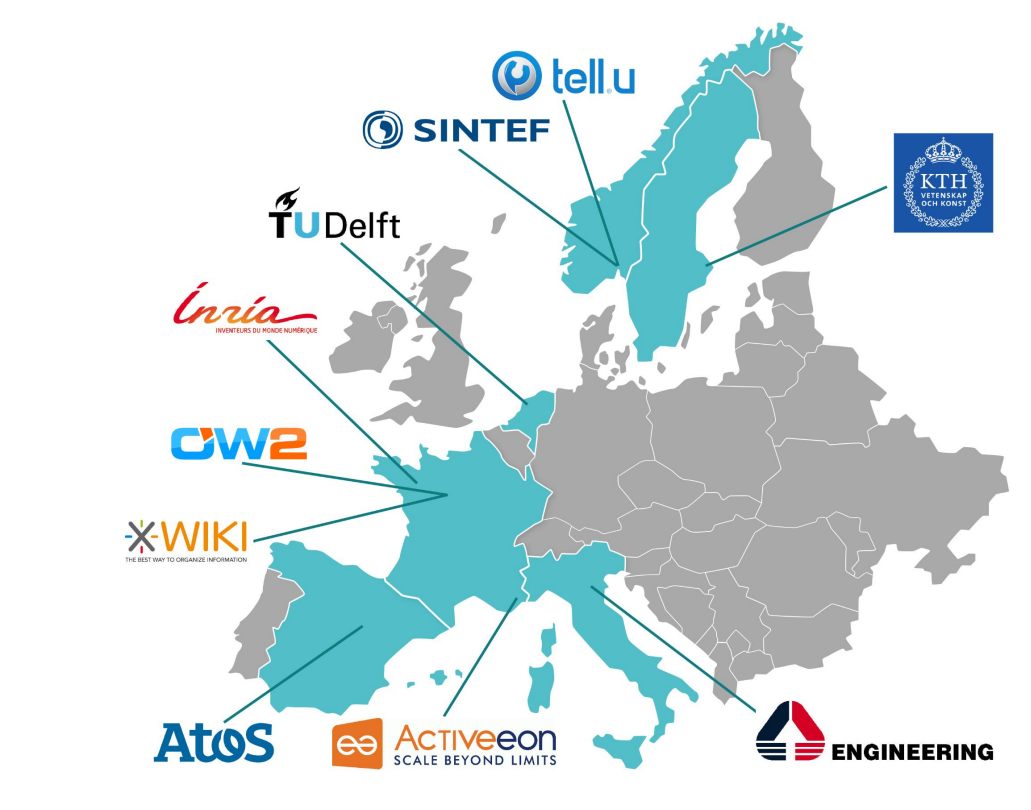
Benjamin Danglot et al. “Correctness attraction: a study of stability of software behavior under runtime perturbation”. In: *Empirical Software Engineering* (2018)

Oscar Luis Vera-Pérez et al. “A comprehensive study of pseudo-tested methods”. In: *Empirical Software Engineering* (2019)

Zhongxing Yu et al. “Alleviating patch overfitting with automatic test generation: a study of feasibility and effectiveness for the Nopol repair system”. In: *Empirical Software Engineering* (2019)

Context: Stamp-project

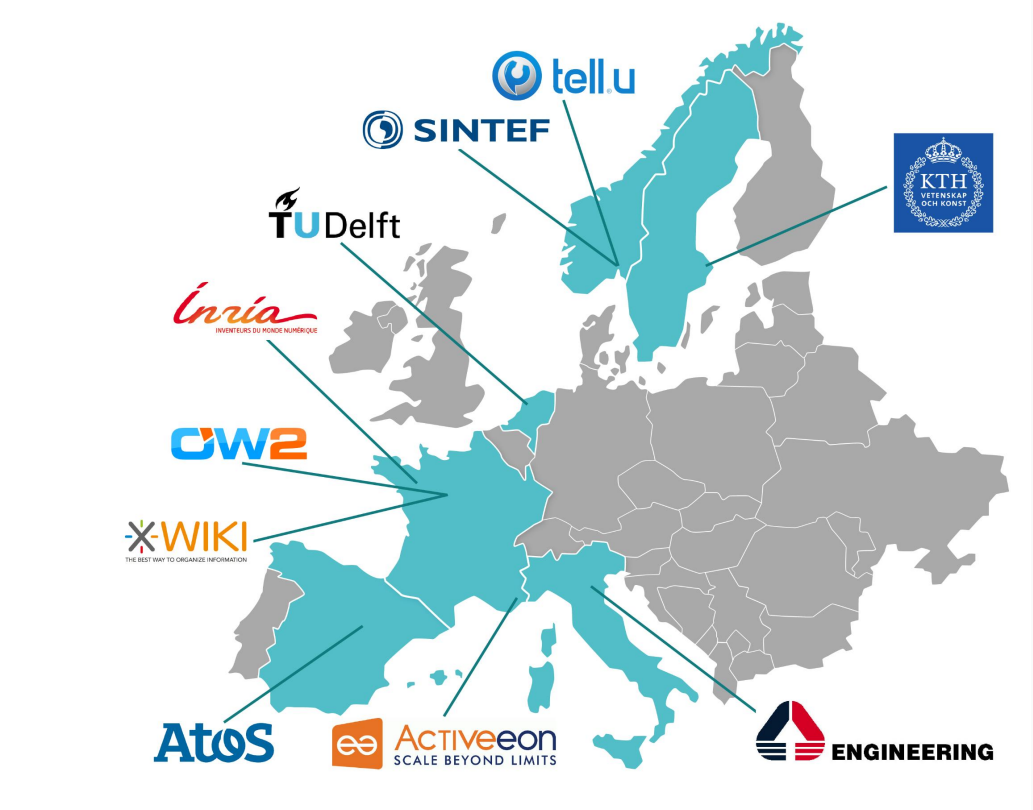
- 4 res. institutions
- 5 industrials partners
- 1 consortium



Context: Stamp-project

- 4 res. institutions
- 5 industrials partners
- 1 consortium

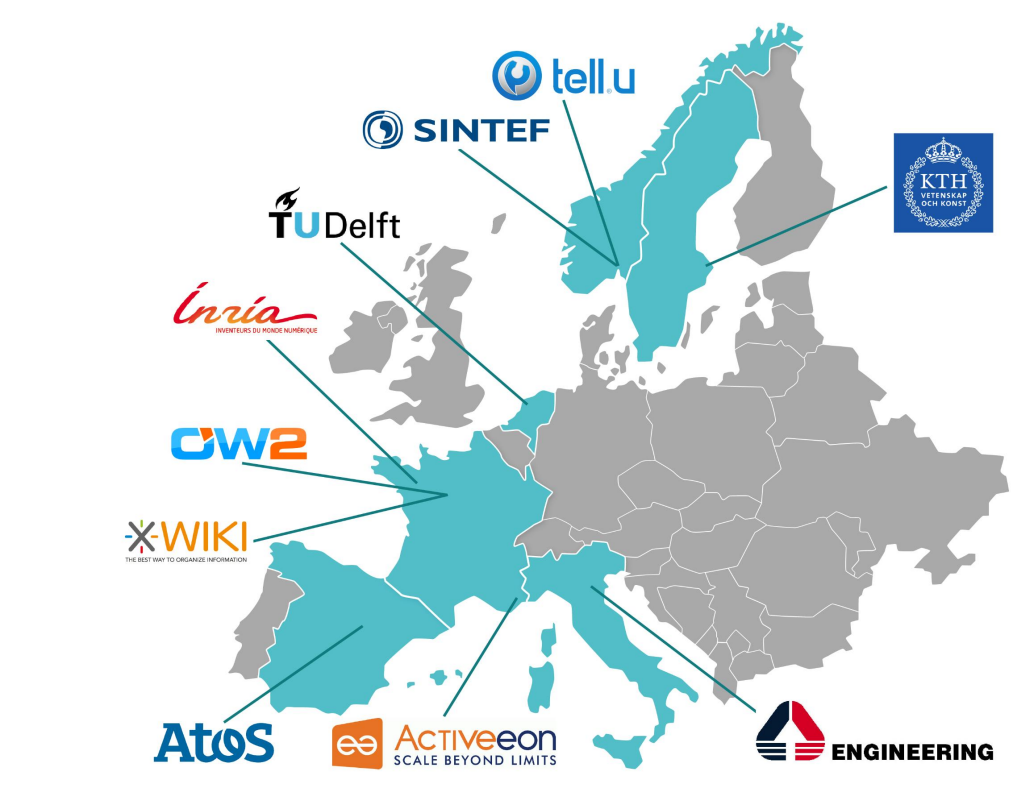
$\Rightarrow$ 8979



Context: Stamp-project

- 4 res. institutions
- 5 industrials partners
- 1 consortium

$\Rightarrow$ 8979
amplified test methods
generated by DSpot



Conclusion

Thesis: **test amplification** can provide actionable tests for developers in regression testing context

Conclusion

Thesis: **test amplification** can provide actionable tests for developers in regression testing context

- **Concept**: Definition of test amplification

Conclusion

Thesis: **test amplification** can provide actionable tests for developers in regression testing context

- **Concept**: Definition of test amplification
- **Survey**: Review of test amplification literature

Conclusion

Thesis: **test amplification** can provide actionable tests for developers in regression testing context

- **Concept**: Definition of test amplification
- **Survey**: Review of test amplification literature
- **Algorithm**: Devising a test amplification algorithm

Conclusion

Thesis: **test amplification** can provide actionable tests for developers in regression testing context

- **Concept**: Definition of test amplification
- **Survey**: Review of test amplification literature
- **Algorithm**: Devising a test amplification algorithm
- **Tool**: Implementation of an open-source tool

Conclusion

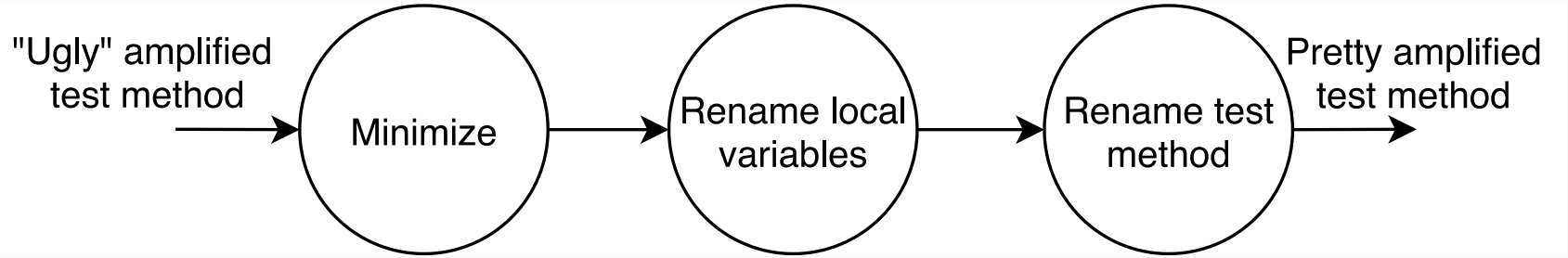
Thesis: **test amplification** can provide actionable tests for developers in regression testing context

- **Concept**: Definition of test amplification
- **Survey**: Review of test amplification literature
- **Algorithm**: Devising a test amplification algorithm
- **Tool**: Implementation of an open-source tool
- **Benchmark**: Selection of projects and commits

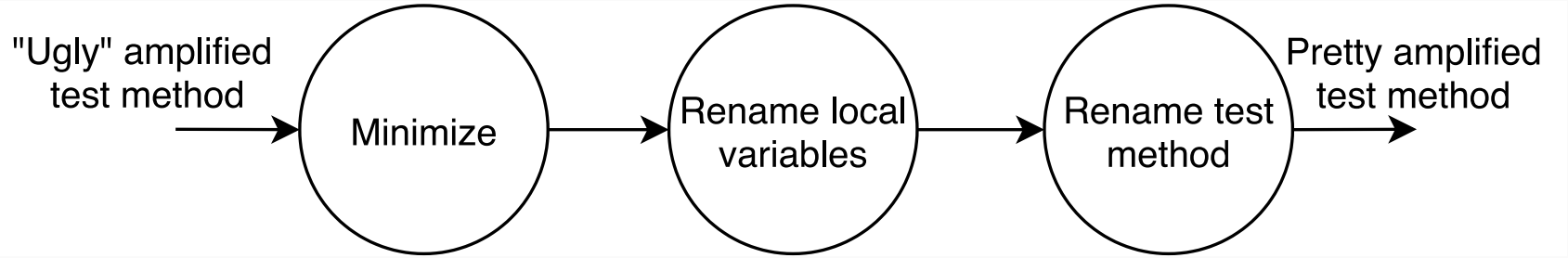
Short-term perspective

```
@Test(timeout = 120000)
public void testCreateAndDeleteMonitorConfiguration_add1() throws Exception {
    AppStoreMonitorConfiguration o_testCreateAndDeleteMonitorConfiguration_add1__1 = createMonitorConfiguration();
    Assert.assertEquals("localhost", ((java.net.URI)((java.net.URL)((eu.supersede.integration.api.monitoring.manager.type
    Assert.assertEquals("//localhost:9092", ((java.net.URI)((java.net.URL)((eu.supersede.integration.api.monitoring.manag
    Assert.assertNull(((eu.supersede.integration.api.monitoring.manager.types.AppStoreMonitorConfiguration)o_testCreateAn
    Assert.assertEquals(0, ((int) (((java.net.URLConnection)((java.net.URL)((eu.supersede.integration.api.monitoring.mana
    Assert.assertEquals("//localhost:9092", ((java.net.URI)((java.net.URL)((eu.supersede.integration.api.monitoring.manag
    Assert.assertTrue(((java.net.URLConnection)((java.net.URL)((eu.supersede.integration.api.monitoring.manager.types.App
    Assert.assertEquals(2133928973, ((int) (((java.net.URL)((eu.supersede.integration.api.monitoring.manager.types.AppSto
    Assert.assertNull(((java.net.URL)((eu.supersede.integration.api.monitoring.manager.types.AppStoreMonitorConfiguration
    Assert.assertFalse(((java.net.URLConnection)((java.net.URL)((eu.supersede.integration.api.monitoring.manager.types.Ap
    Assert.assertEquals("567630281", ((eu.supersede.integration.api.monitoring.manager.types.AppStoreMonitorConfiguration
    Assert.assertNull(((java.net.URL)((eu.supersede.integration.api.monitoring.manager.types.AppStoreMonitorConfiguration
    Assert.assertEquals("MarketPlace", ((eu.supersede.integration.api.monitoring.manager.types.AppStoreMonitorConfigurati
    Assert.assertEquals("localhost", ((java.net.URL)((eu.supersede.integration.api.monitoring.manager.types.AppStoreMonit
    Assert.assertEquals(9092, ((int) (((java.net.URL)((eu.supersede.integration.api.monitoring.manager.types.AppStoreMoni
    Assert.assertEquals(0L, ((long) (((java.net.URLConnection)((java.net.URL)((eu.supersede.integration.api.monitoring.ma
    ...
}
```

Short-term perspective

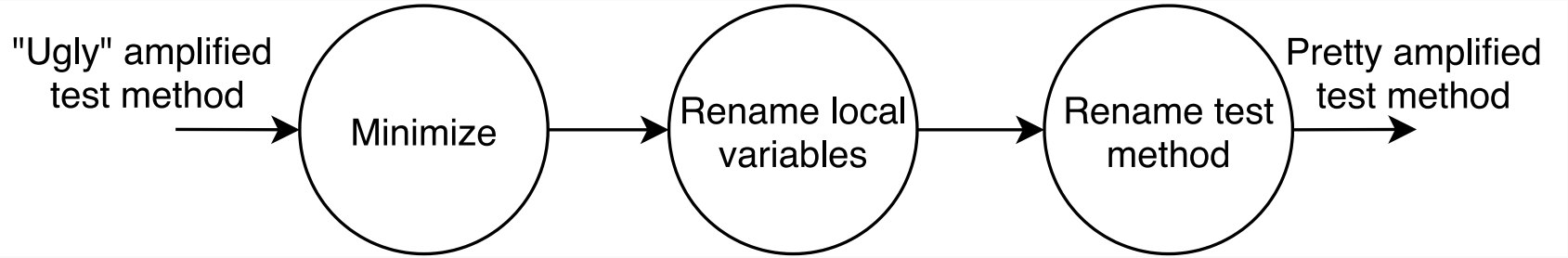


Short-term perspective



Minimization: remove duplicated assertions, use local variables, etc.

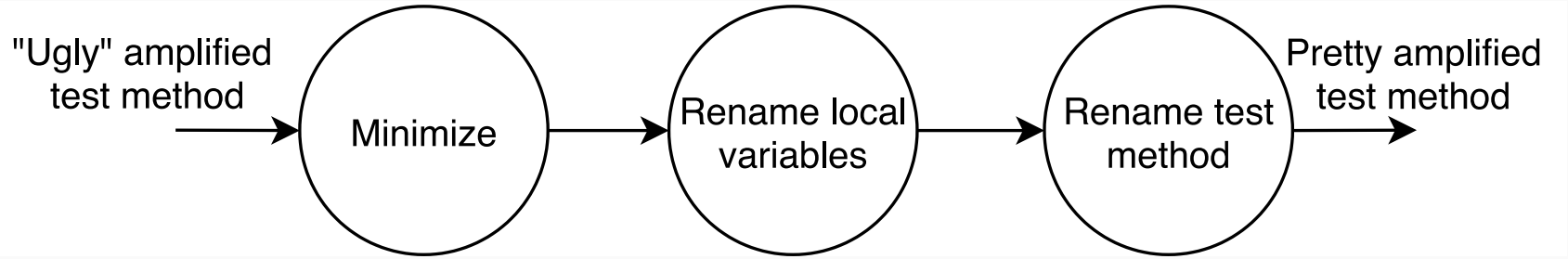
Short-term perspective



Rename local variables: context2name[11]

[11] Bavishi, Rohan, Pradel, Michael, and Sen, Koushik. "Context2Name: A Deep Learning-Based Approach to Infer Natural Variable Names from Usage Contexts". *arXiv*.

Short-term perspective



Rename test method: Code2Vec[12]

[12] Alon, Uri et al. "Code2Vec: Learning Distributed Representations of Code". *Proc. ACM Program. Lang.*

Long-term perspective

Closed **Add Test: improperly closed variable** #186

danglotb wants to merge 1 commit into `spullara:master` from `danglotb:ampl-abstractClassTest`

spullara commented on 27 Feb 2017 — **Owner**

I'll add this test but it should really be in one of the concrete classes like InterpreterTest.

Long-term perspective

Closed **Add Test: improperly closed variable** #186

danglotb wants to merge 1 commit into `spullara:master` from `danglotb:ampl-abstractClassTest`

spullara commented on 27 Feb 2017 — Owner

I'll add this test but it should really be in one of the concrete classes like InterpreterTest.

- What would be the best location for amplified test methods?

Long-term perspective

Merged **test: specify more toJavaIdentifier** #669

Changes from **all commits** File filter... Jump to...

3 src/test/java/com/squareup/javapoet/NameAllocatorTest.java

```
25 27     NameAllocator nameAllocator = new NameAllocator();
26 28     assertThat(nameAllocator.newName("foo", 1)).isEqualTo("foo");
   @@ -59,6 +61,7 @@
59 61   @Test public void characterMappingInvalidStartButValidPart() throws Exception {
60 62     NameAllocator nameAllocator = new NameAllocator();
61 63     assertThat(nameAllocator.newName("1ab", 1)).isEqualTo("_1ab");
   64 +   assertThat(nameAllocator.newName("a-1", 2)).isEqualTo("a_1");
62 65   }
```

Merged **test: test writeVarInt32 with large Integer** #250

Changes from **all commits** File filter... Jump to...

9 protostuff-core/src/test/java/io/protostuff/LinkBufferTest.java

```
12 12
   13 +   @Test
   14 +   public void testWriteLargeVar32() throws Exception
   15 +   {
   16 +       LinkBuffer b = new LinkBuffer(8);
   17 +       b.writeVarInt32(Integer.MAX_VALUE);
   18 +       assertEquals(1, b.getBuffers().size());
   19 +       assertEquals(5, b.getBuffers().get(0).remaining());
   20 +   }
   21 +
```

Long-term perspective

Merged **test: specify more toJavaIdentifier** #669

Changes from **all commits** File filter... Jump to...

3 src/test/java/com/squareup/javapoet/NameAllocatorTest.java

```
25 27     NameAllocator nameAllocator = new NameAllocator();
26 28     assertThat(nameAllocator.newName("foo", 1)).isEqualTo("foo");
   @@ -59,6 +61,7 @@
59 61   @Test public void characterMappingInvalidStartButValidPart() throws Exception {
60 62     NameAllocator nameAllocator = new NameAllocator();
61 63     assertThat(nameAllocator.newName("1ab", 1)).isEqualTo("_1ab");
   64 +   assertThat(nameAllocator.newName("a-1", 2)).isEqualTo("a_1");
62 65   }
```

Should we modify an existing test or should we create a new one?

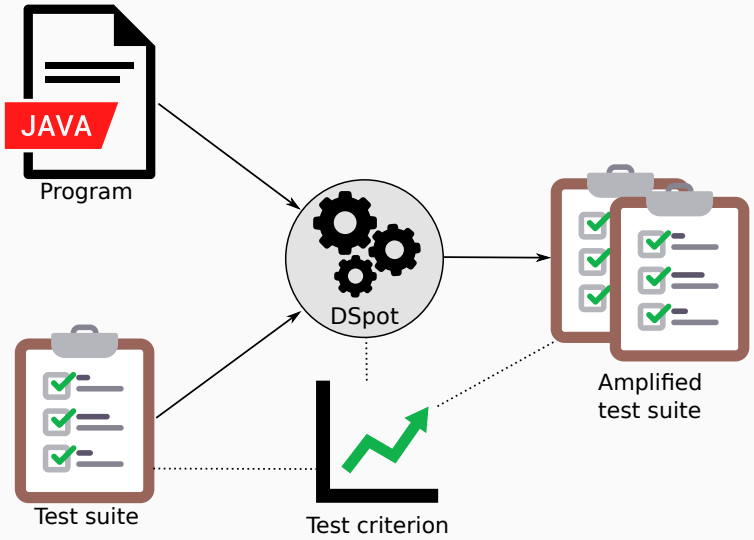
Merged **test: test writeVarInt32 with large Integer** #250

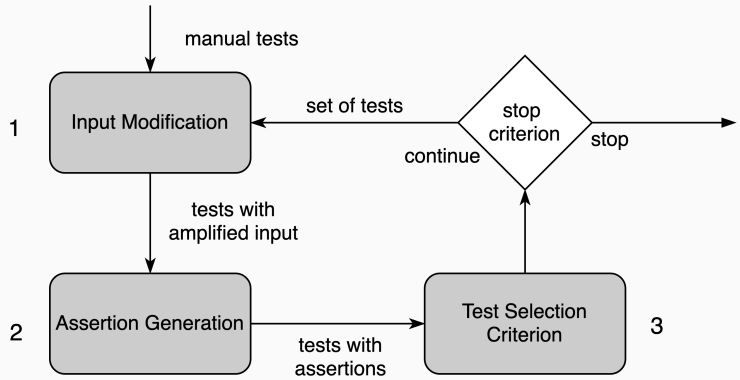
Changes from **all commits** File filter... Jump to...

9 protostuff-core/src/test/java/io/protostuff/LinkBufferTest.java

```
12 12
   13 +   @Test
   14 +   public void testWriteLargeVar32() throws Exception
   15 +   {
   16 +       LinkBuffer b = new LinkBuffer(8);
   17 +       b.writeVarInt32(Integer.MAX_VALUE);
   18 +       assertEquals(1, b.getBuffers().size());
   19 +       assertEquals(5, b.getBuffers().get(0).remaining());
   20 +   }
   21 +
```

Conclusion





project	# opened	# merged
javapoet	4	4
mybatis-3	2	2
traccar	2	1
stream-lib	1	1
mustache	2	2
twilio	2	1
jsoup	2	1
prostostuff	2	2
logback	2	0
retrofit	0	0
total	19	14

project	#Detection	Average Time
commons-io	5/10	38.8m
commons-lang	3/10	74.7m
gson	4/10	86.5m
jsoup	3/10	2.6h
mustache.java	6/10	4.2h
xwiki-commons	4/10	49.5m
total	25/60	-