Exploring the detection of semantic conflicts in code integrations involving multiple methods

Toni Maciel
Universidade Federal de Pernambuco
jaam@cin.ufpe.br

Paulo Borba
Universidade Federal de Pernambuco
phmb@cin.ufpe.br

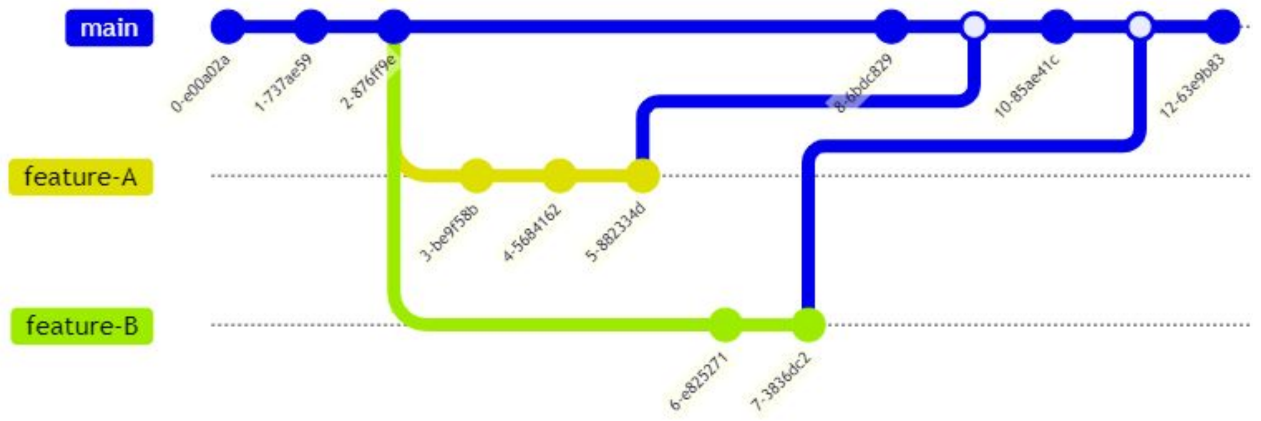
Léuson Da Silva
Polytechnique Montreal
leuson-mario-pedro.da-silva@polymtl.ca

Thaís Burity
Universidade Federal do Agreste de Pernambuco
thais.burity@ufape.edu.br

Centro de Informática UFPE

Universidade Federal de Pernambuco

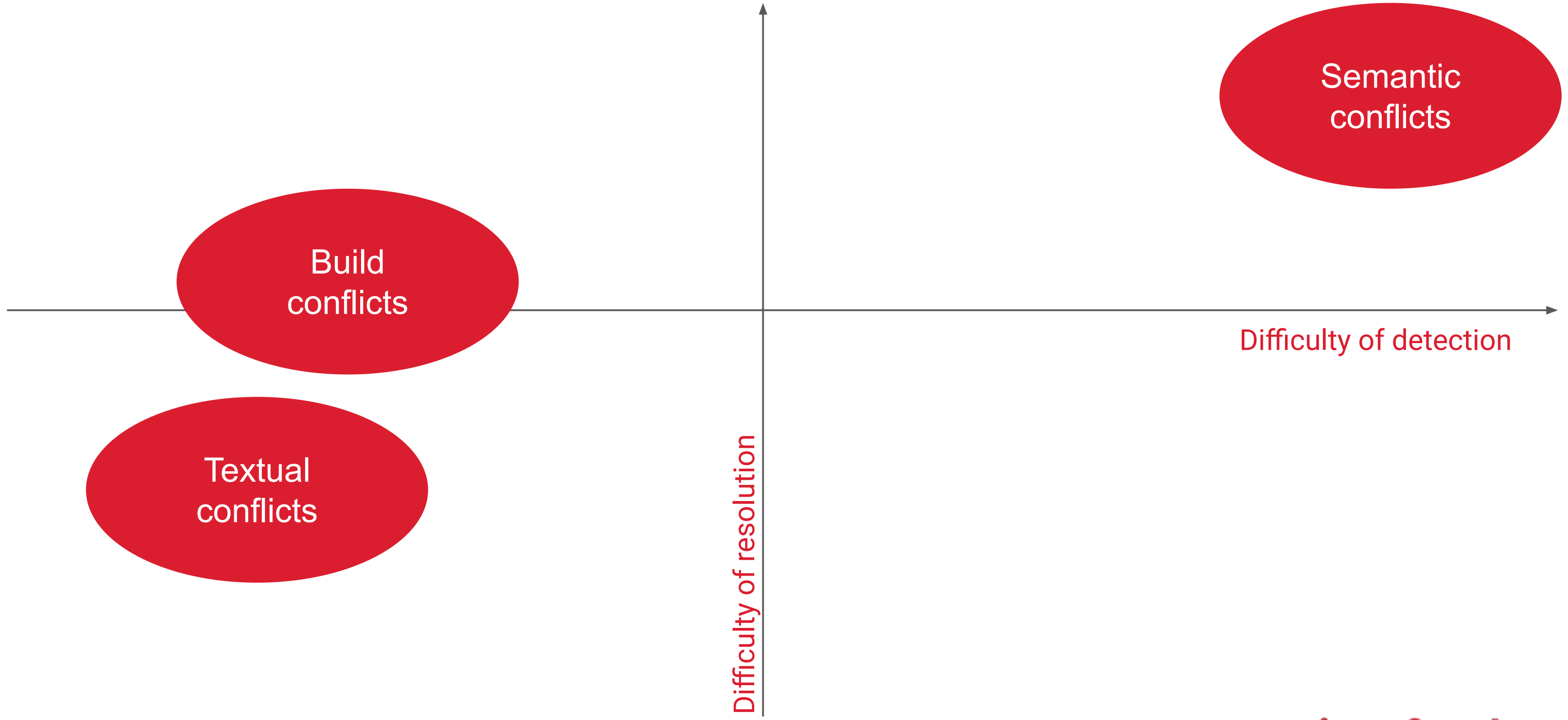
Software development is an **collaborative process.**

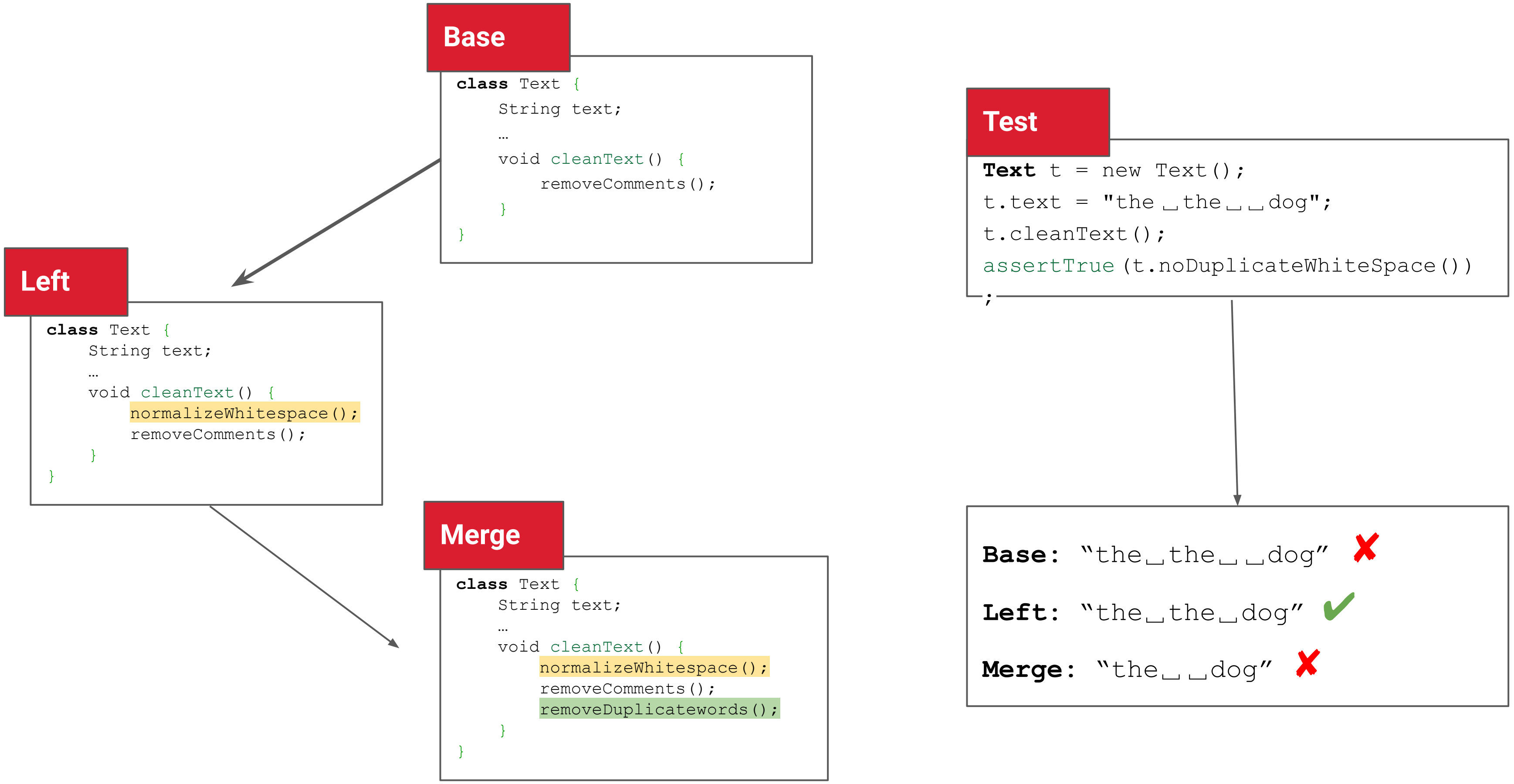


Collaboration → Integration

Parallel development aims to enhance team performance by enabling faster deliveries. This often requires **merging** changes to integrate updates.

However, conflicts can arise in various forms...





Base

```
class Text {
    String text;
    …
    void cleanText() {
        removeComments();
    }
}
```

Left

```
class Text {
    String text;
    …
    void cleanText() {
        normalizeWhitespace();
        removeComments();
    }
}
```

Merge

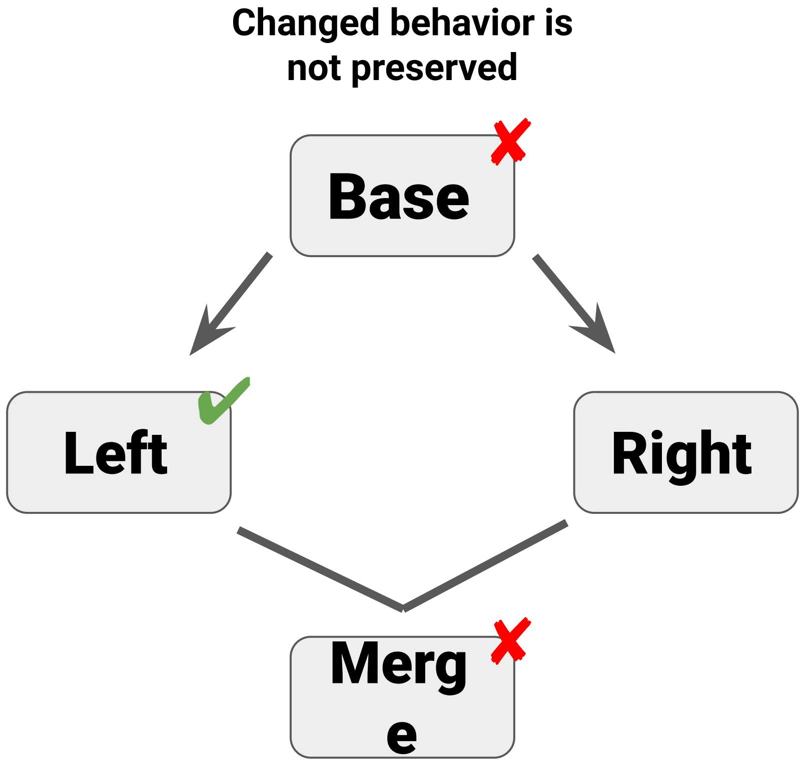
```
class Text {
    String text;
    …
    void cleanText() {
        normalizeWhitespace();
        removeComments();
        removeDuplicatewords();
    }
}
```

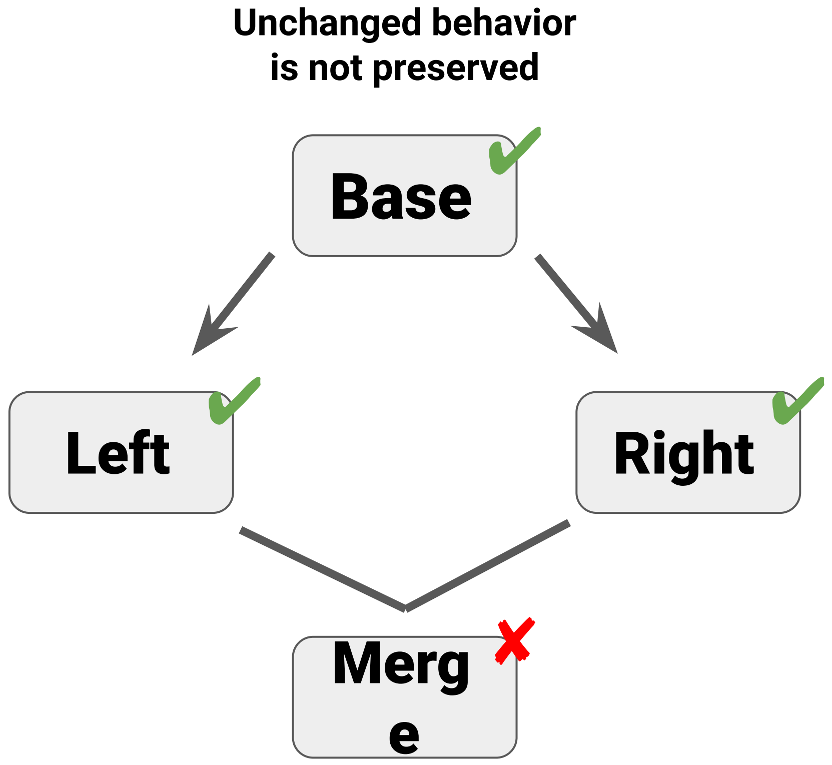
Test

```
Text t = new Text();
t.text = "the␣the␣␣dog";
t.cleanText();
assertTrue(t.noDuplicateWhiteSpace())
;
```

```
Base:  "the␣the␣␣dog" ✗
Left:  "the␣the␣dog" ✓
Merge:  "the␣␣dog" ✗
```

Identification of semantic conflicts through testing





Exploring the detection of semantic conflicts

01

A larger and more complex sample

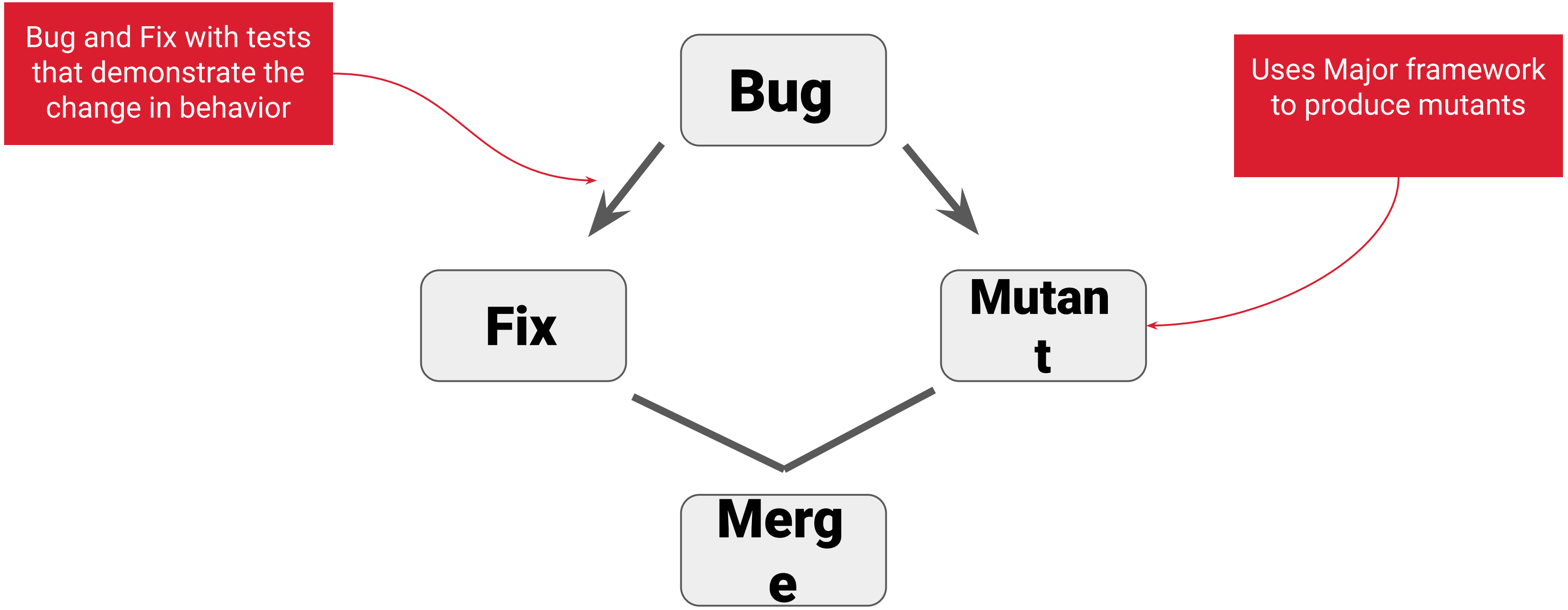
02

Amplification of test generation tools

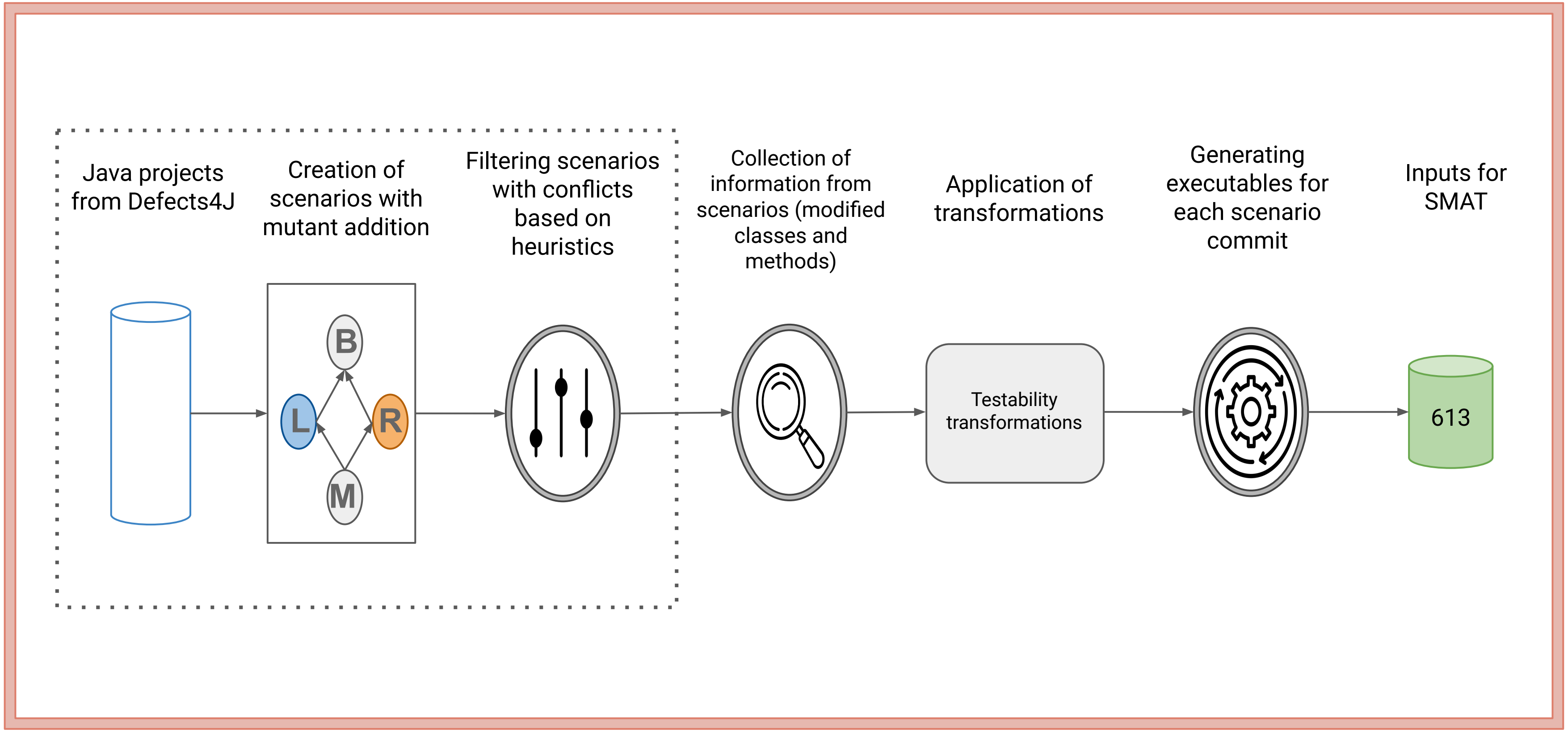
03

Orchestrating with SMAT

Built with Defects4J


Bug and Fix with tests that demonstrate the change in behavior
Bug
Uses Major framework to produce mutants
Fix
Mutant
Merge

Sample generation workflow



Java projects from Defects4J
Creation of scenarios with mutant addition
Filtering scenarios with conflicts based on heuristics
Collection of information from scenarios (modified classes and methods)
Application of transformations
Generating executables for each scenario commit
Inputs for SMAT
B
L
R
M
Testability transformations
613

Exploring the detection of semantic conflicts

01
A larger and more complex sample

02
Amplification of test generation tools

03
Orchestrating with SMAT

EvoSuite looks at the entire target class...

EvoSuite is based on objective coverage

```
public List<LineCoverageTestFitness> getCoverageGoals() {
   List<LineCoverageTestFitness> goals = new ArrayList<>();
   for (String className : LinePool.getKnownClasses()) {
     if (!isCUT(className))
       continue;
     for (String methodName : LinePool.getKnownMethodsFor(className)) {
       if (isEnumDefaultConstructor(className, methodName))
         continue;
       Set<Integer> lines = LinePool.getLines(className, methodName);
       for (Integer line : lines) {
         (…)
         goals.add(new LineCoverageTestFitness(className, methodName, line));
       }
     }
   }
   return goals;
 }
```

Only lines of the class under test

Adds the lines of **all** methods in the class

… Focused EvoSuite looks only at the target method

```
public List<LineCoverageTestFitness> getCoverageGoals() {
  List<LineCoverageTestFitness> goals = new ArrayList<>();
  for (String className : LinePool.getKnownClasses()) {
    if (!isCUT(className))
      continue;
    for (String methodName : LinePool.getKnownMethodsFor(className)) {
      if (isEnumDefaultConstructor(className, methodName))
        continue;
      if (!matcher.methodMatches(methodName)) {
        continue;
      }
      Set<Integer> lines = LinePool.getLines(className, methodName);
      for (Integer line : lines) {
        (…)
        goals.add(new LineCoverageTestFitness(className, methodName, line));
      }
    }
  }
  return goals;
}
```

```
if (methodName.matches(targetMethodRegex))
    return true;
```

Uses regular expressions for validation due to ASM signatures

Exploring the detection of semantic conflicts

01

A larger and more complex sample

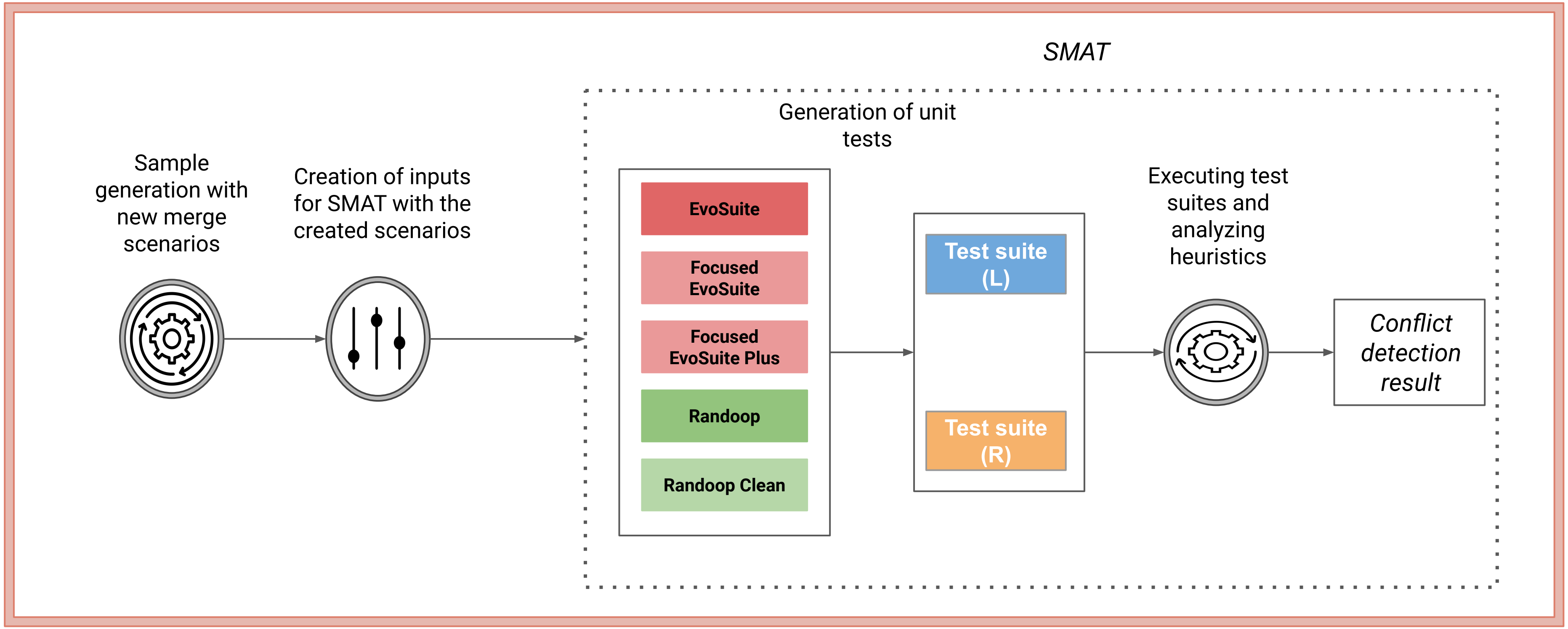
02

Amplification of test generation tools

03

Orchestrating with SMAT

Experiment workflow


SMAT
Sample generation with new merge scenarios
Creation of inputs for SMAT with the created scenarios
Generation of unit tests
EvoSuite
Focused EvoSuite
Focused EvoSuite Plus
Randoop
Randoop Clean
Test suite (L)
Test suite (R)
Executing test suites and analyzing heuristics
Conflict detection result

Analysis of executions on the servers

Server 1: 200 detected conflicts;

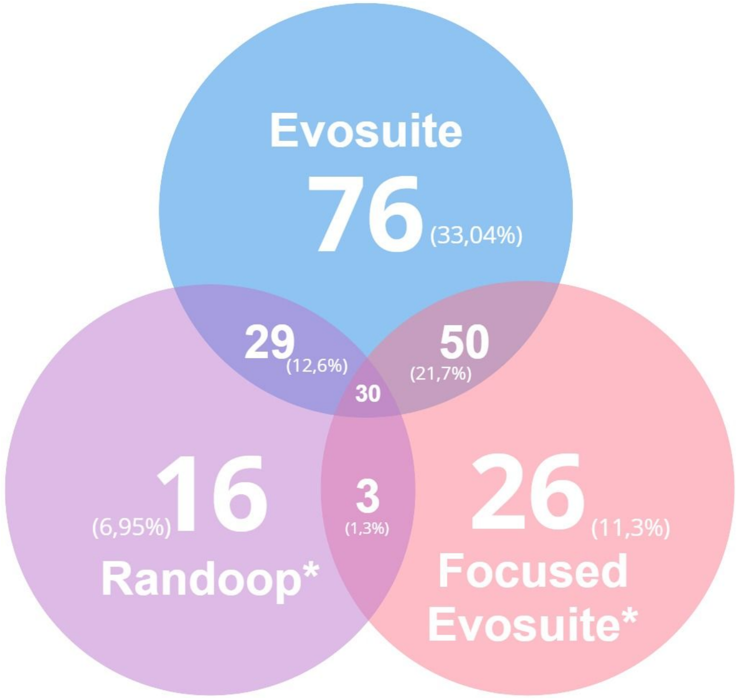
Server 2: 202 detected conflicts;

Server 1 ∩ Server 2: 172 detected conflicts;

Server 1 ∪ Server 2: **230 (37,52%)** detected conflicts;

Complementarity shows a good result

- Dominance of EvoSuite
- Contributions of Focused EvoSuite
- Limitations of Randoop



Benchmarking Against Silva (2022)

	Silva (2022)	This study
Number of scenarios	85	613
Existing conflicts	28	613
Detected conflicts (%)	9 (10,6%)	230 (37,5%)
Recall	0,321	0,375
Tools	EvoSuite, Randoop, Randoop Clean e Differential EvoSuite	EvoSuite, Randoop, Randoop Clean e Focused EvoSuite*

SILVA, L. M. P. d. (2022). **Detecting, understanding, and resolving build and test conflicts.** PhD thesis, Recife, PE, Brazil.

Exploring the detection of semantic conflicts

01

A larger and more complex sample

02

Amplification of test generation tools

03

Orchestrating with SMAT

Future work

Refine automated test generation techniques: Explore EvoSuite's fitness to produce tests more aligned with conflict detection heuristics.

Incorporation of Large Language Models: Develop complex tests with detailed assertions that inspect the maintenance or alteration of expected behaviors across the different tested versions.

Centro de Informática UFPE

Universidade Federal de Pernambuco

Acknowledgments!

SPG

SOFTWARE·PRODUCTIVITY·GROUP

ines

National Institute
of Science and Technology
in Software Engineering

50 anos

Centro de Informática
UFPE

CBSOFT'25

XVI CONGRESSO BRASILEIRO DE SOFTWARE: TEORIA E PRÁTICA
22 A 26 DE SETEMBRO | RECIFE/PE

Exploring the detection of semantic conflicts in code integrations involving multiple methods

Toni Maciel
Universidade Federal de Pernambuco
jaam@cin.ufpe.br

Paulo Borba
Universidade Federal de Pernambuco
phmb@cin.ufpe.br

Léuson Da Silva
Polytechnique Montreal
leuson-mario-pedro.da-silva@polymtl.ca

Thaís Burity
Universidade Federal do Agreste de Pernambuco
thais.burity@ufape.edu.br

Centro de Informática UFPE

Universidade Federal de Pernambuco