Analyzing Conflict Predictors in Open-Source Java Projects

Paola Accioly - prga@cin.ufpe.br
Paulo Borba - phmb@cin.ufpe.br
Leuson Silva - lmps2@cin.ufpe.br
Guilherme Cavalcanti - gjcc@cin.ufpe.br

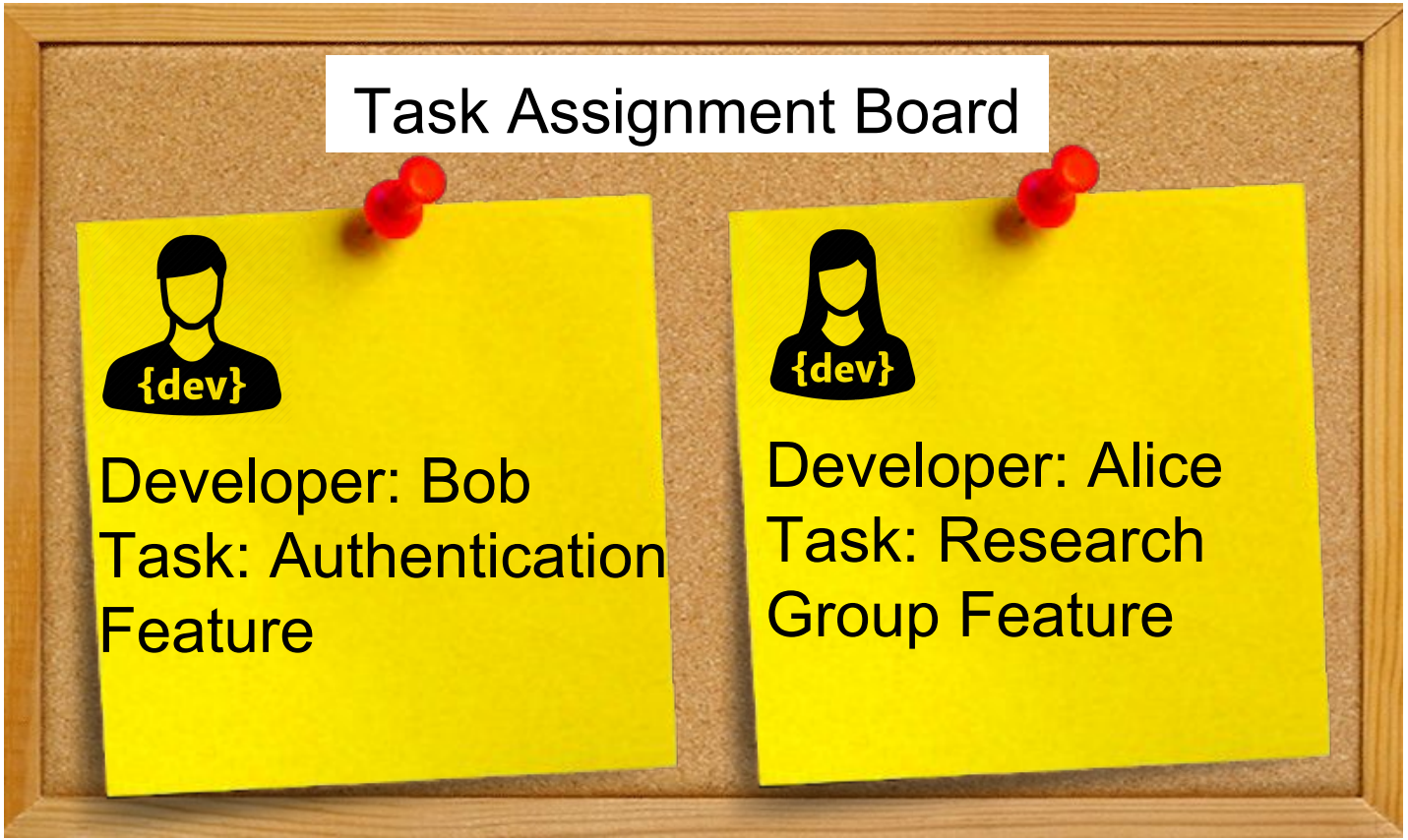
SPG
SOFTWARE·PRODUCTIVITY·GROUP

ines

FACEPE
Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

VIRTUS IMPAVIDA
UNIVERSIDADE FEDERAL DE PERNAMBUCO

Collaborative development environment



```
public class Member{
    String name;
    ...
}
```

Base

Authentication

```
public class Member{
    String name;
    String username;
    public String toString(){
        return this.username;
    }
    ...
}
```

Research Group

```
public class Member{
    String name;
    public String toString(){
        return this.name;
    }
    ...
}
```

While merging, conflicts might occur...

Merge conflict

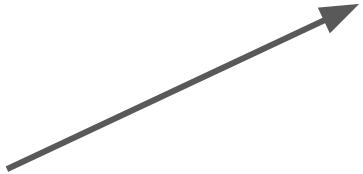
```
<<<<<<< Authentication
	String username;
	public String toString(){
		return this.username;
	}
=======
	public String toString(){
		return this.name;
	}
>>>>>>> Research Group
```

Build conflict

```
public String toString(){
    return this.name;
}
...
public String toString(){
    return this.username;
}
```

Test conflict

```
public String toString(){
    return this.name;
}
...
```



One test from the Authentication feature failed after code integration

Conflicts occur frequently and resolving them is a time consuming and error prone task

Which might impact both development’s productivity and the resulting product’s quality

Most merge conflicts happen when developers edit the same lines from the same method [Accioly et al, 2017]

Most build and test conflicts occur when developers edit directly dependent methods [Lima, 2014]

Are those good conflict predictors?

Goal

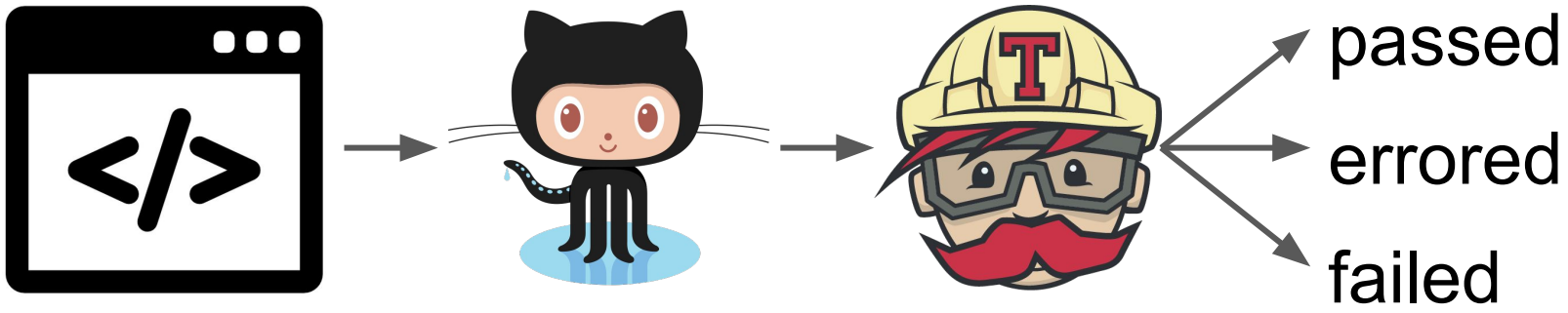
Analyze editions to the same method— **EditSameMC**— and editions to directly dependent methods— **EditDepMC**— effectiveness as conflict predictors during the development history of different Java projects hosted on GitHub

Strategy

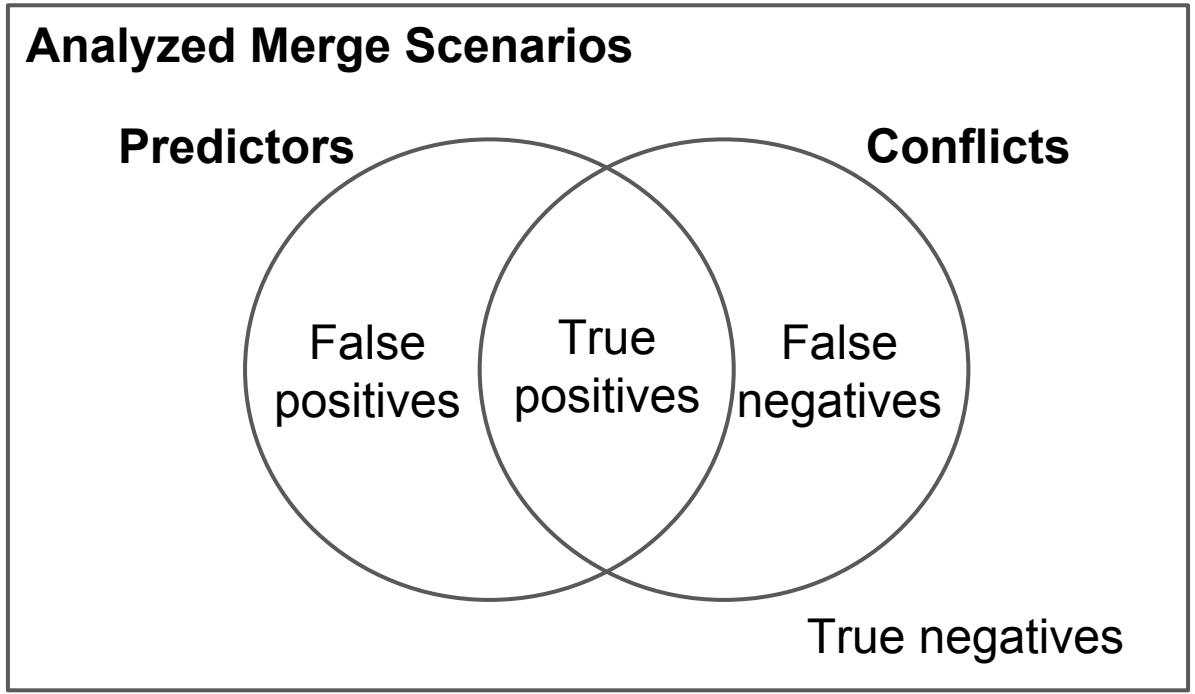
Reproduce merge scenarios while collecting conflict predictor instances together with merge, build, and test conflict instances to compute how often the predictors are associated to conflicts

To collect EditSameMC instances we used an adapted version of FSTMerge, a semistructured merge tool [Apel et al, 2011]

For establishing build and test conflicts ground truth, we rely on the status of building and testing processes executed by the Travis CI service

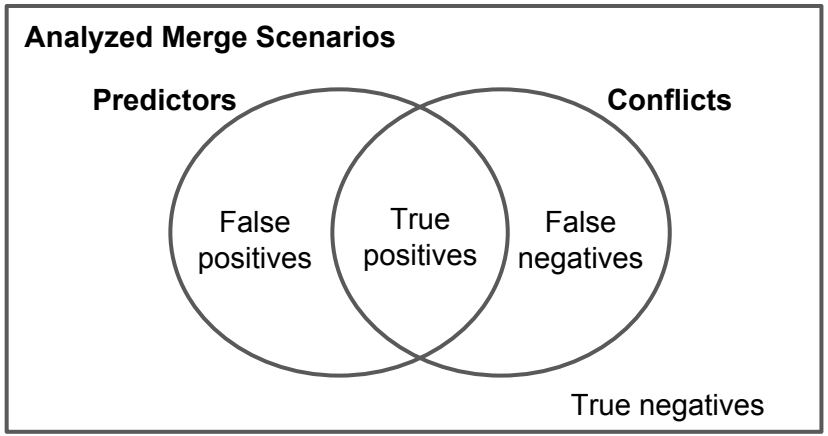


How frequently predictors are associated to conflicts



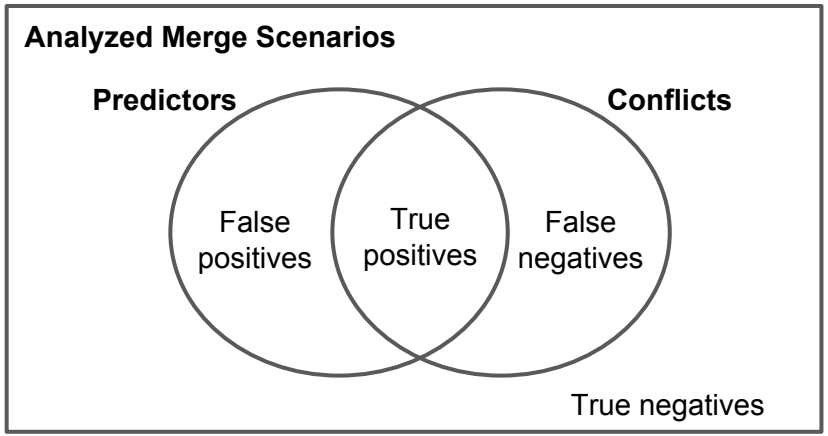
RQ1: How precise are EditSameMC and EditDepMC predictors?

$$Precision = \frac{true\ \ positives}{true\ \ positives + false\ \ positives}$$



RQ2: How many conflicts can we avoid by detecting EditSameMC and EditDepMC predictors?

$$Recall = \frac{true\ \ positives}{true\ \ positives\ +\ false\ \ negatives}$$

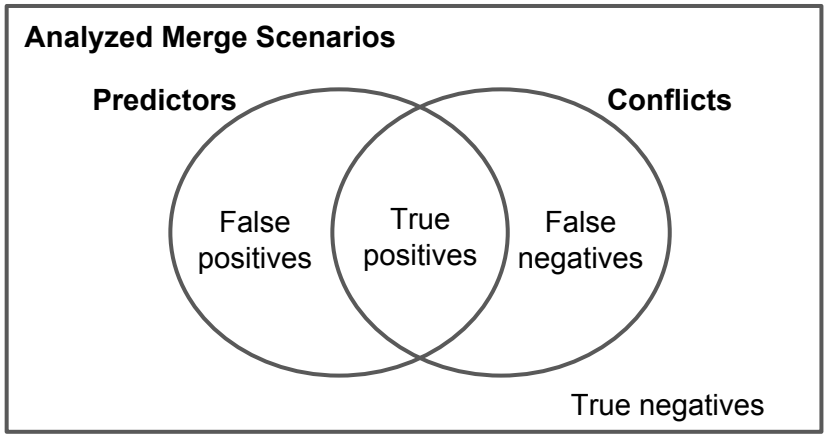


RQ3: Why EditSameMC and EditDepMC instances are not associated with merge, build, or test conflicts?

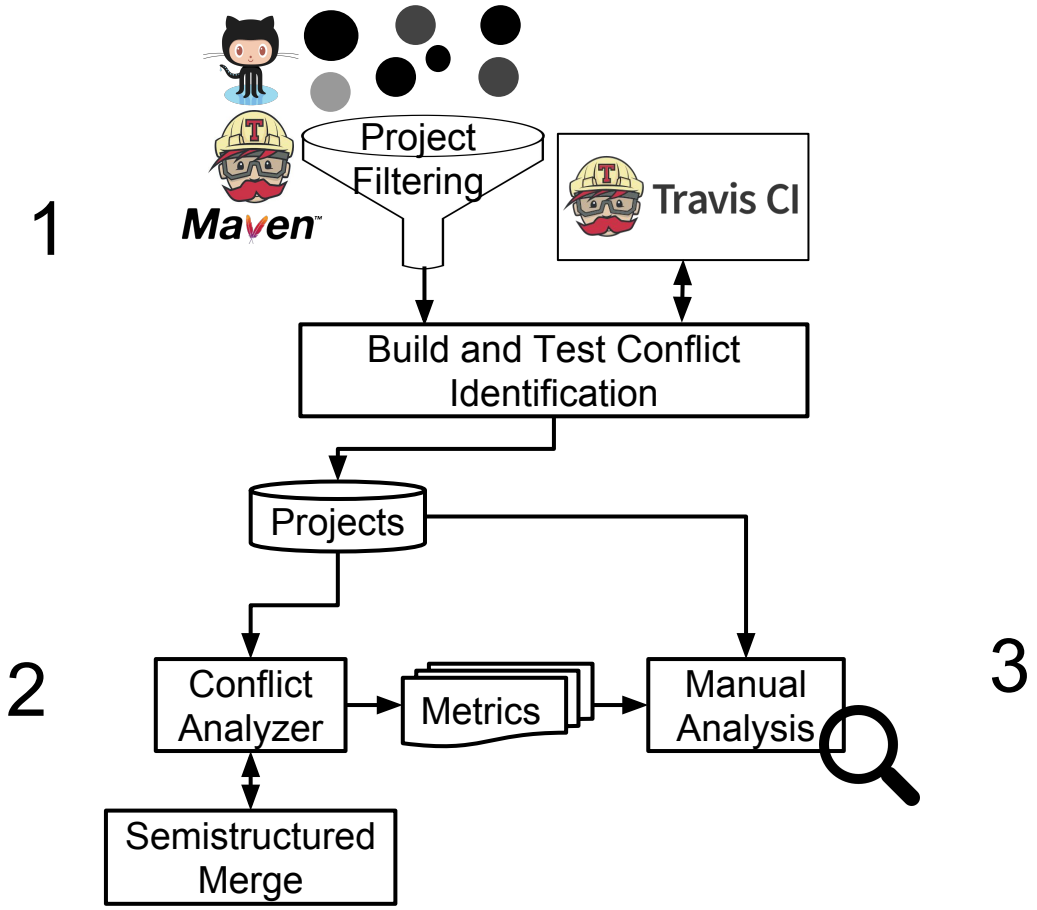
Considering a sub-sample of the false positives we conduct a manual analysis to check if there are contributions' interferences missed by our oracle (Travis CI testing process)

RQ4: What other change patterns are associated with conflicts?

We identify the false negatives' conflict cause



Study Setup


1
Project Filtering
Travis CI
Maven
Build and Test Conflict Identification
Projects
2
Conflict Analyzer
Metrics
Manual Analysis
3
Semistructured Merge

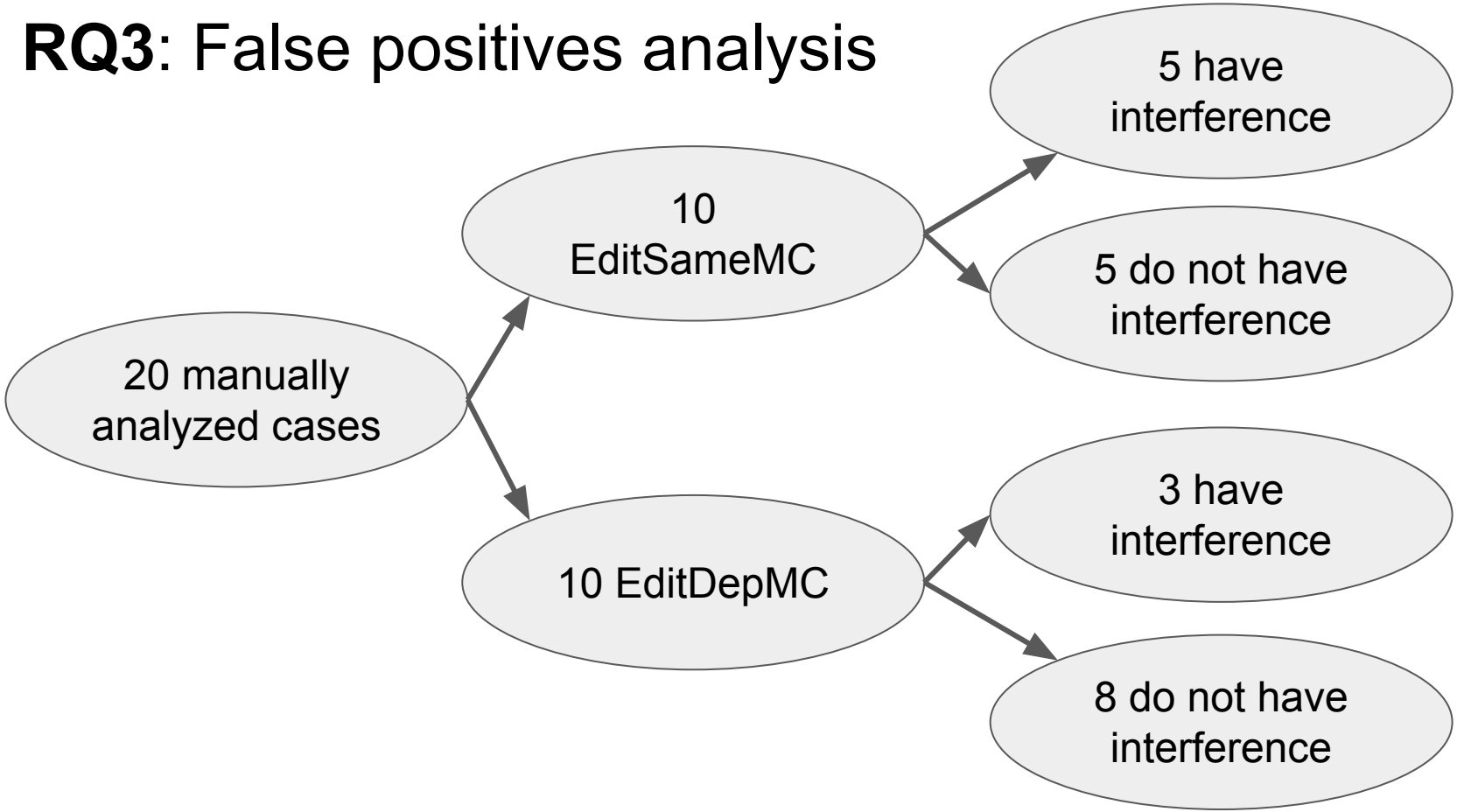
Sample

- 45 analyzed projects
- 5,647 merge scenarios
- 290 merge conflicts
- 84 build conflicts
- 5 test conflicts

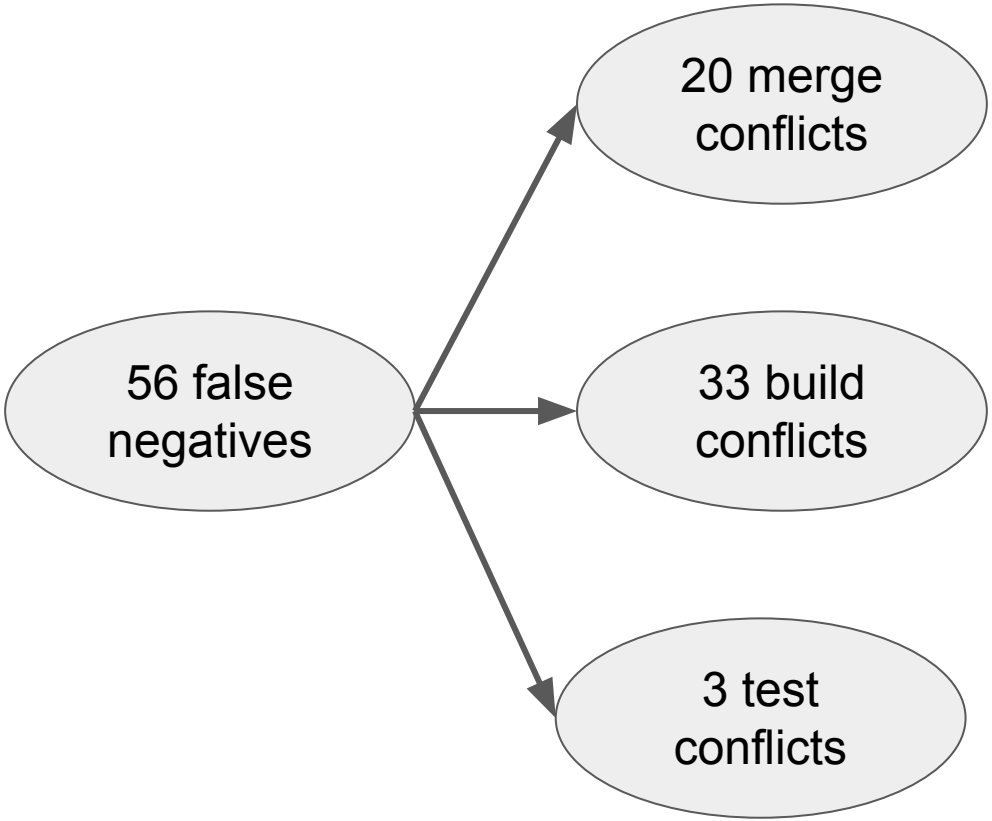
RQ1 and **RQ2** - conflict predictors precision and recall

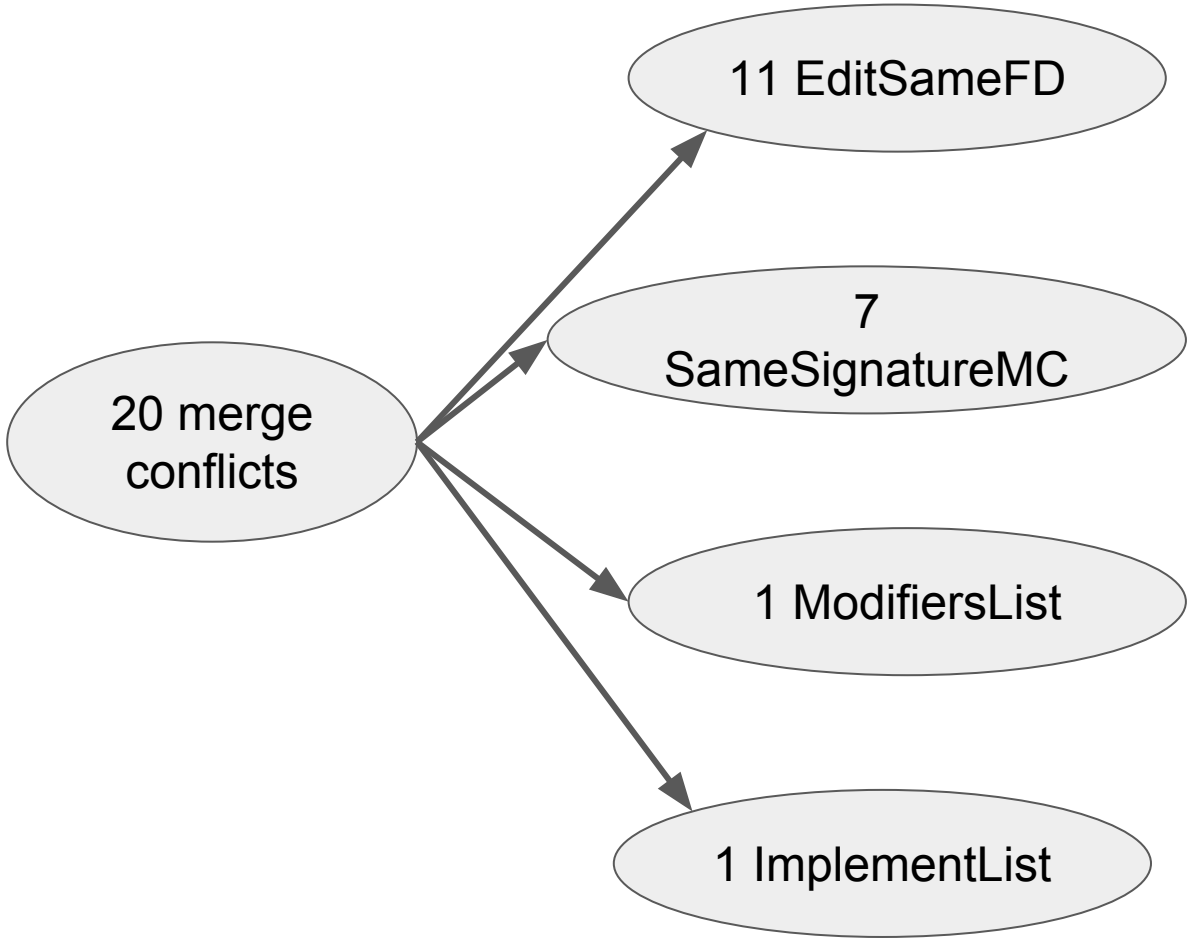
	Both predictors	EditSameMC	EditDepMC
Precision	56.29%	55.51%	8.85%
Recall	83.62%	82.45%	13.15%

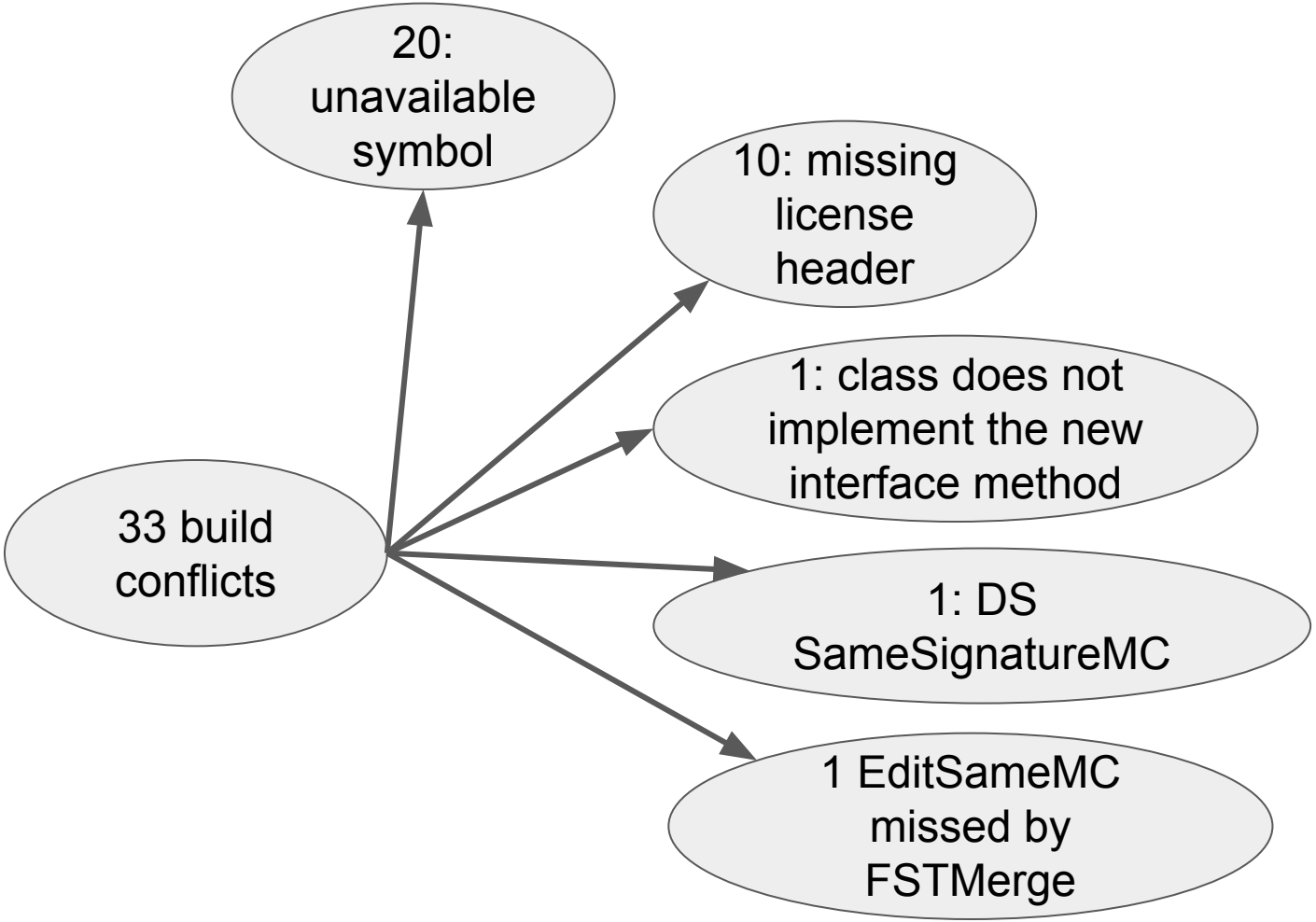
RQ3: False positives analysis

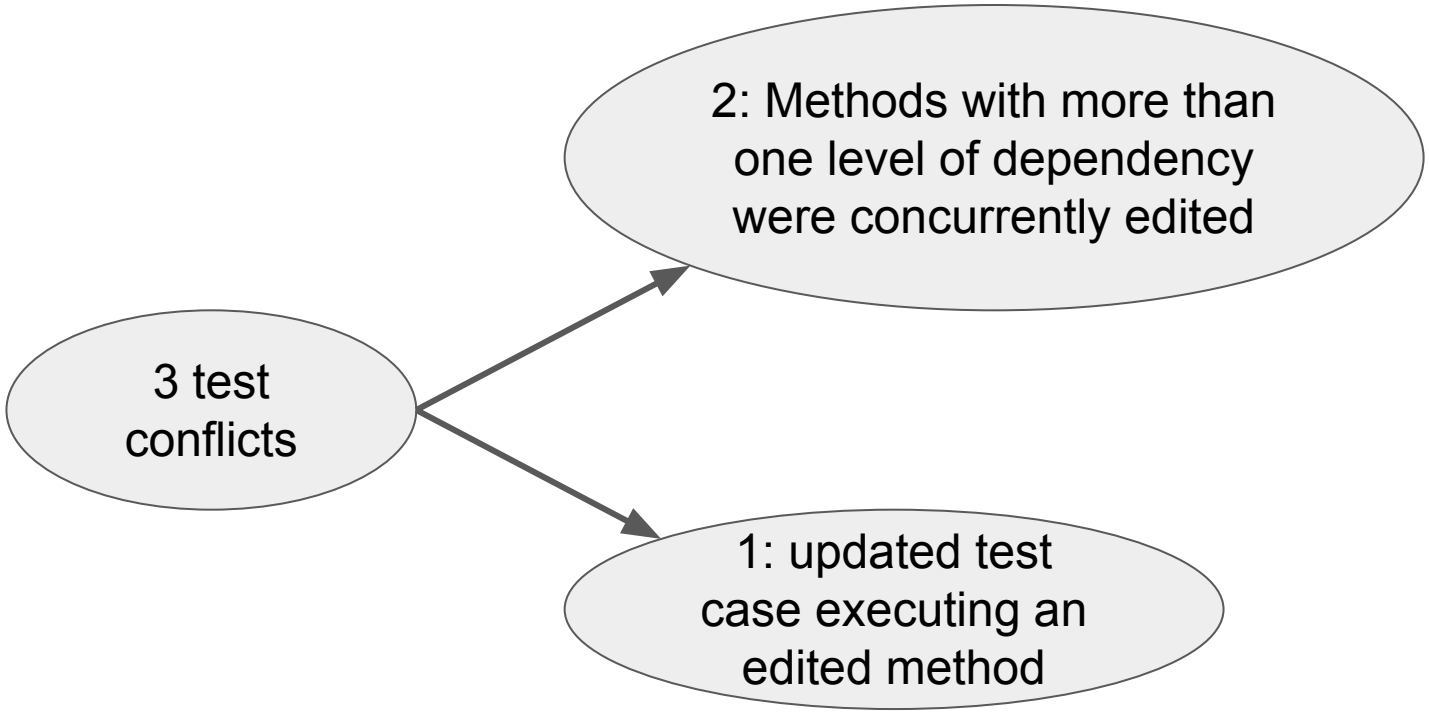


RQ4: False negatives analysis




11 EditSameFD
7
SameSignatureMC
20 merge
conflicts
1 ModifiersList
1 ImplementList


20: unavailable symbol
10: missing license header
1: class does not implement the new interface method
33 build conflicts
1: DS SameSignatureMC
1 EditSameMC missed by FSTMerge


2: Methods with more than one level of dependency were concurrently edited
3 test conflicts
1: updated test case executing an edited method

How effective are EditSameMC and EditDepMC?

These results can be used to guide different conflict awareness strategies depending on each team preferences:

- Conservative vs. Precise

Precision	Recall
56.29%	83.62%

Strategies to improve precision and recall

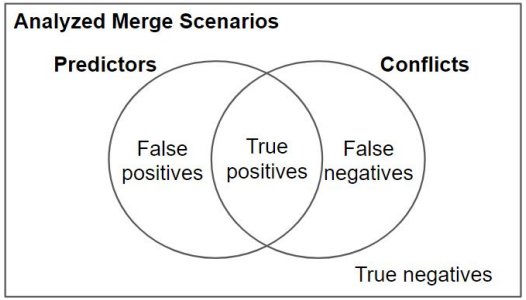
1. Implement strategies to identify and ignore cases where clearly there is no interference
   a. Detect different spacing changes or comments
   b. Detect refactoring changes
2. Implement strategies to identify possible interferences
   a. Identify information flow control between contributions [Filho, 2017]
   b. Generate test cases to explore contributions interaction [Böhme, 2013]

Conflicts occur frequently and resolving them is a time consuming and error prone task

Which might impact both development's productivity and the resulting product's quality

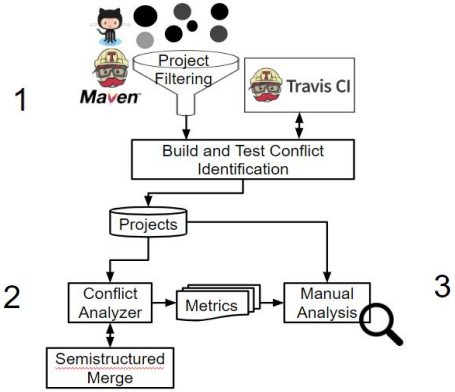
8

How frequently predictors are associated to conflicts



15

Study Setup



20

RQ1 and **RQ2** - conflict predictors precision and recall

	Both predictors	EditSameMC	EditDepMC
Precision	56.29%	55.51%	8.85%
Recall	83.62%	82.45%	13.15%

22

@paola_accioly