Predicting Co-Changes between Functionality Specifications and Source Code in Behavior Driven Development

Aidan Yang

Queen's University

Daniel A. da Costa

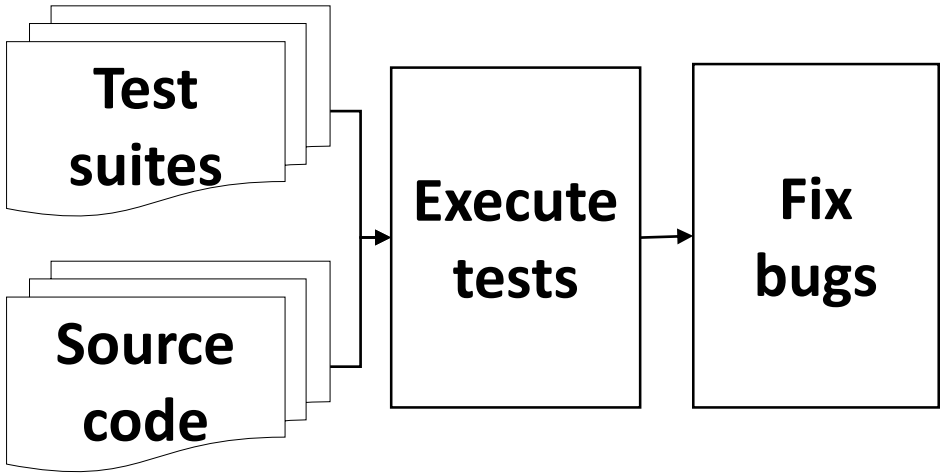
University of Otago

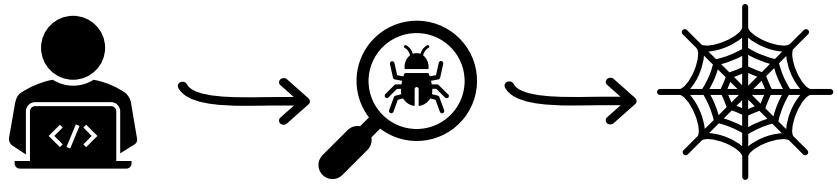
Ying (Jenny) Zou

Queen's University

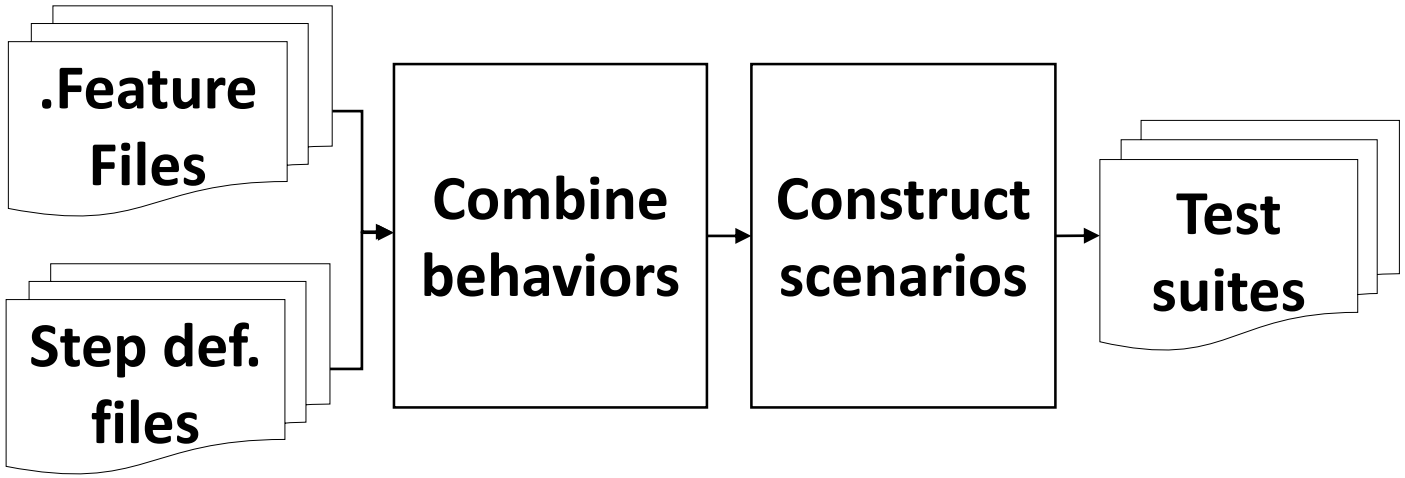
Behavior Driven Development (BDD) is a New Style of Testing Strategy Based on Test Driven Development

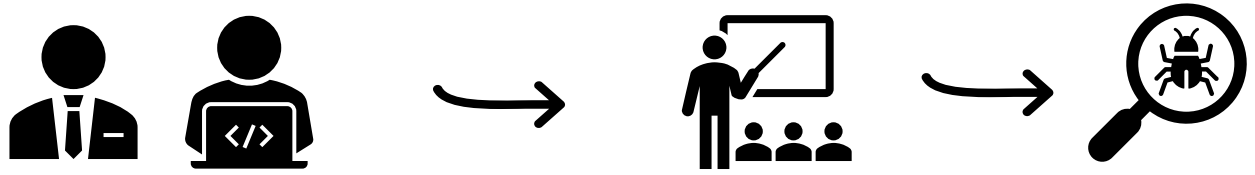
Traditional Testing Strategies





Behavior Driven Development





Example .Feature File

.Feature

```
@players
Feature: Trivialt Players and Teams
Scenario: Register as a new player
When you register "Tobias" with handle
    "@thobe"
Then trivialt knows "@thobe" is "Tobias"
And "@thobe" should be the current player
```

Example Step Definition File

.Feature

```
@players
Feature: Trivialt Players and Teams
Scenario: Register as a new player
When you register "Tobias" with handle
    "@thobe"
Then trivialt knows "@thobe" is "Tobias"
And "@thobe" should be the current player
```

Step Definition

```
@players
@When("^you register \"([^\"]*)\" with
    handle \"([^\"]*)\"$")
public void youRegisterUser(String name,
    String handle){
  currentPlayer =
      trivialtWorld.register(handle, name);
  assertThat(currentPlayer,
      is(not(nullValue())));
}
@Then("^trivialt knows \"([^\"]*)\" is
    \"([^\"]*)\"$")
```

Step Definition File Linkage to Source Code

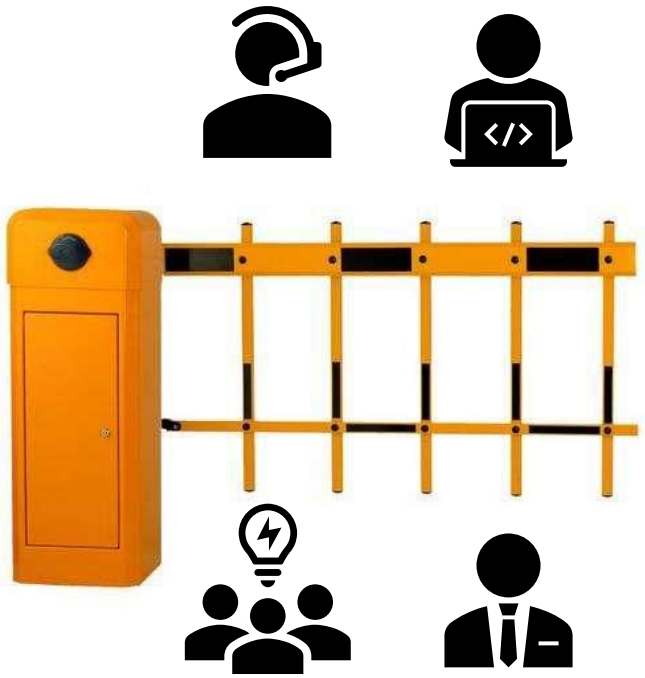
Step Definition

```
@players
@When("^you register \"([^\"]*)\" with
    handle \"([^\"]*)\"$")
public void youRegisterUser(String name,
    String handle){
  currentPlayer =
      trivialtWorld.register(handle, name);
  assertThat(currentPlayer,
      is(not(nullValue())));
}
@Then("^trivialt knows \"([^\"]*)\" is
    \"([^\"]*)\"$")
```

Source Code

```
public Player register( String handle, String name )
{
    Node node = playerMap.get( handle );
    Player player = null;
    if (node == null) {
        node = playerMap.put( handle, graphdb.createNode() );
        player = new Player(node);
        player.setName(name);
        player.setHandle( handle );
    } else {
        player = new Player(node);
    }
```

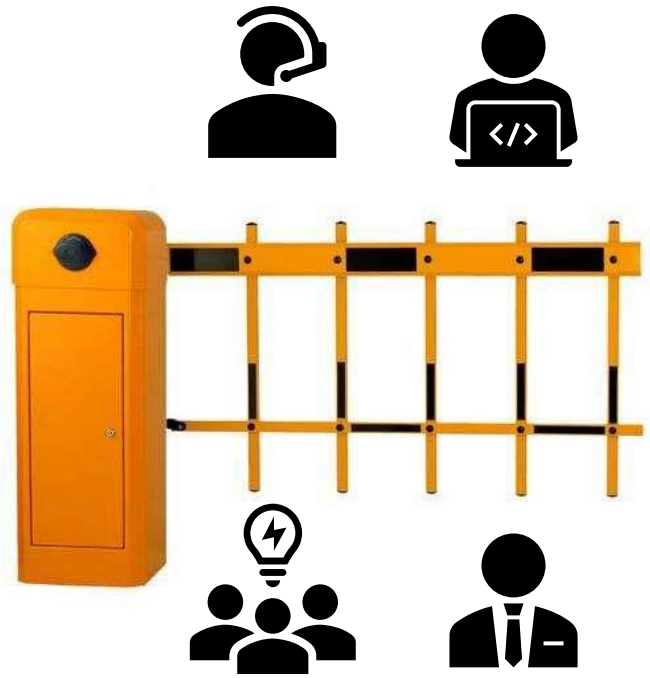
Motivation



BDD

- Out-of-synch co-change reduces benefits of BDD
- New developers entering a project struggle to understand requirements

Motivation

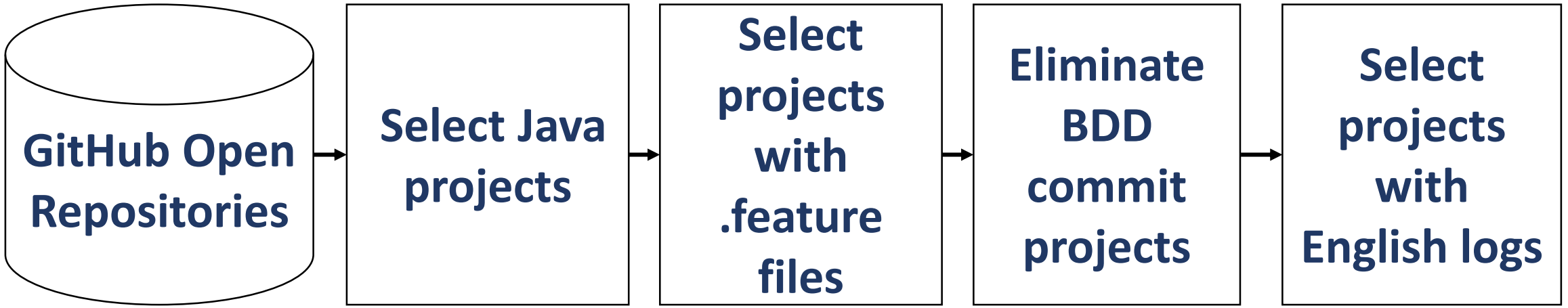


BDD

- Out-of-synch co-change reduces benefits of BDD
- New developers entering a project struggle to understand requirements

Identify code characteristics that can predict co-changes between .feature files and source code files (BDD co-changes) to help developers reduce out-of-sync BDD co-changes

We Selected 133 BDD Projects for Analysis



Research Questions

- **(RQ1): Can we accurately identify co-changes between .feature files and source code files?**

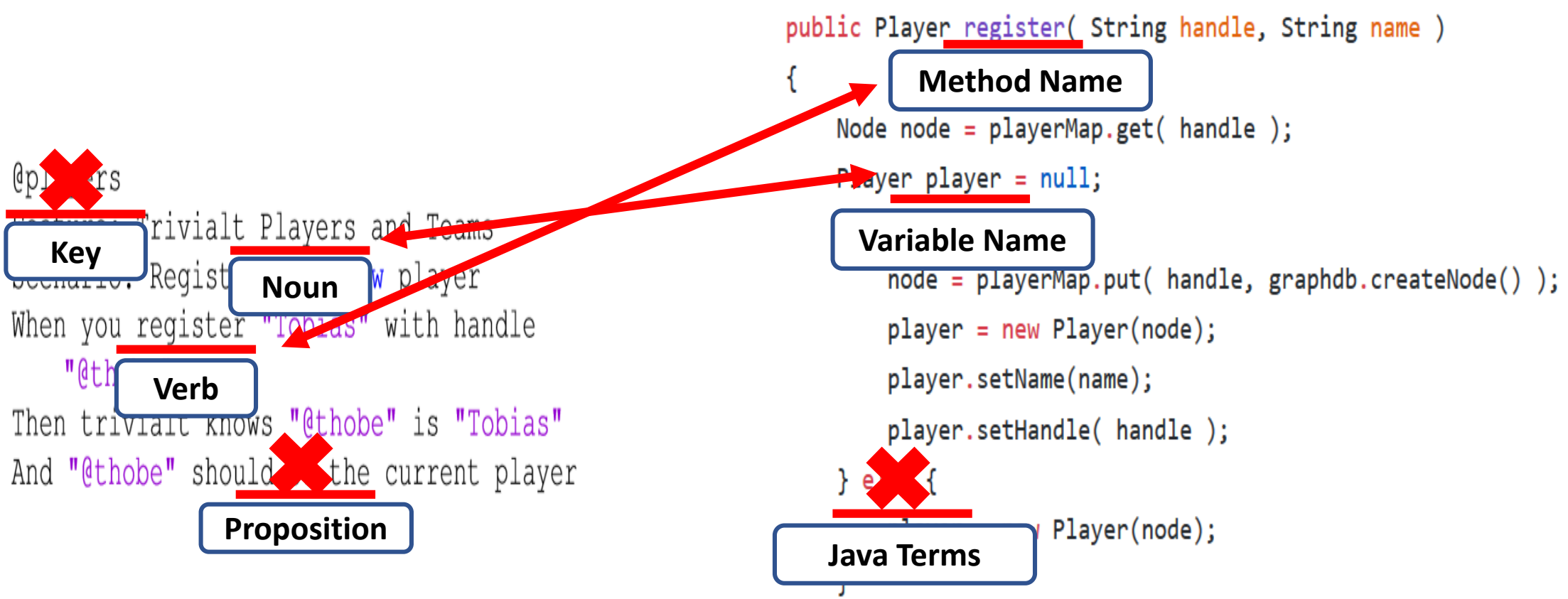
- **(RQ2): Can we accurately predict when co-changes between .feature files and source code files are necessary?**

- **(RQ3): What are the most significant characteristics for predicting co-changes between .feature files and source code files?**

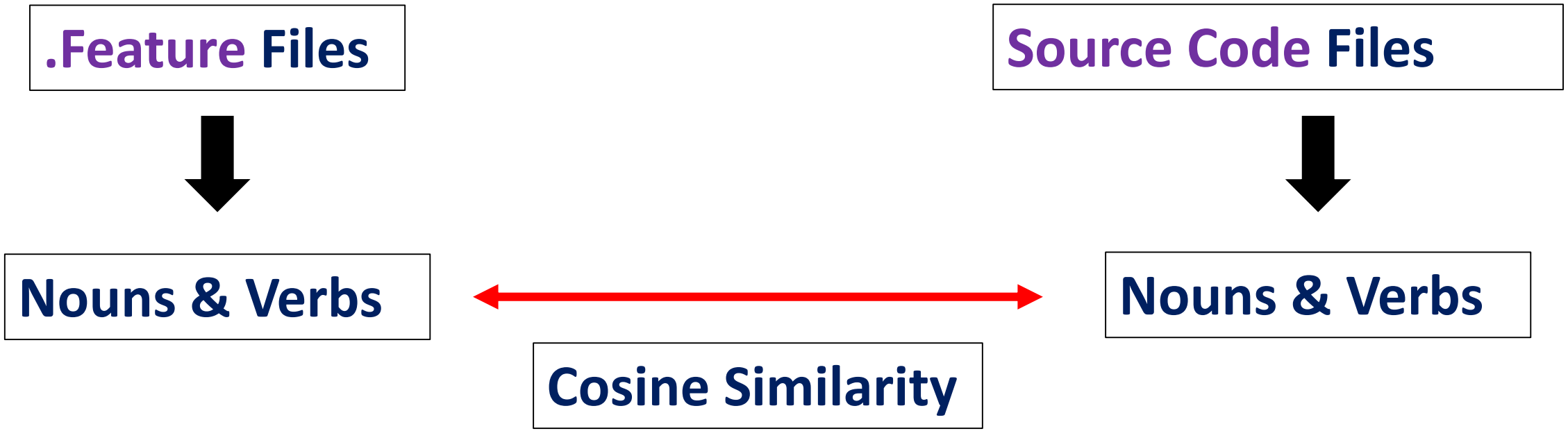
Link .Feature Files and Source Code Using Semantic Similarity

Characteristics of .Feature Files

Characteristics of Source Code Files



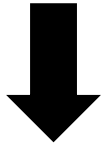
Our Analysis Obtained Over 60,000 Links Within 133 BDD Projects



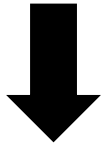
NLP analysis obtained 60,203 links within the same commit and 1,815 cross commit links.

We then Perform Manual Analysis to Check the Accuracy of Our Approach

1,815 cross commit links. Too many to check in detail



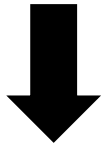
Confidence level of 95% and confidence interval of 5%



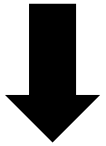
Sample size of 451

Analysis of Links in the Sample

Sample size of 451.



360 co-changing work items actually linked together.



80% agreement rate after inspection by another author.

Our Approach Yields

79% Accuracy

Episode 2: The RQs Strike Back

- (RQ1): Can we accurately identify co-changes between .feature files and source code files?

- **(RQ2): Can we accurately predict when co-changes between .feature files and source code files are necessary?**

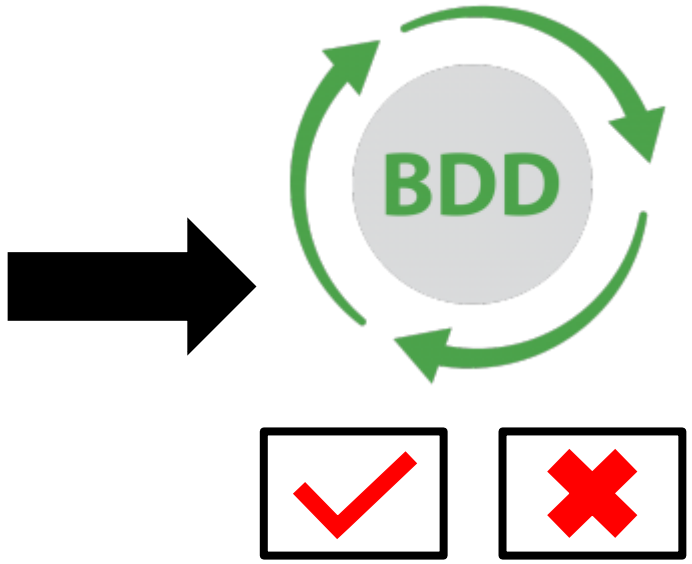
- (RQ3): What are the most significant characteristics for predicting co-changes between .feature files and source code files?

19 Characteristics for Prediction

- Source files added
- Author Experience with BDD
- Test files renamed
- Other files deleted
- Dependencies added (libraries imported)
- Source files renamed
- …

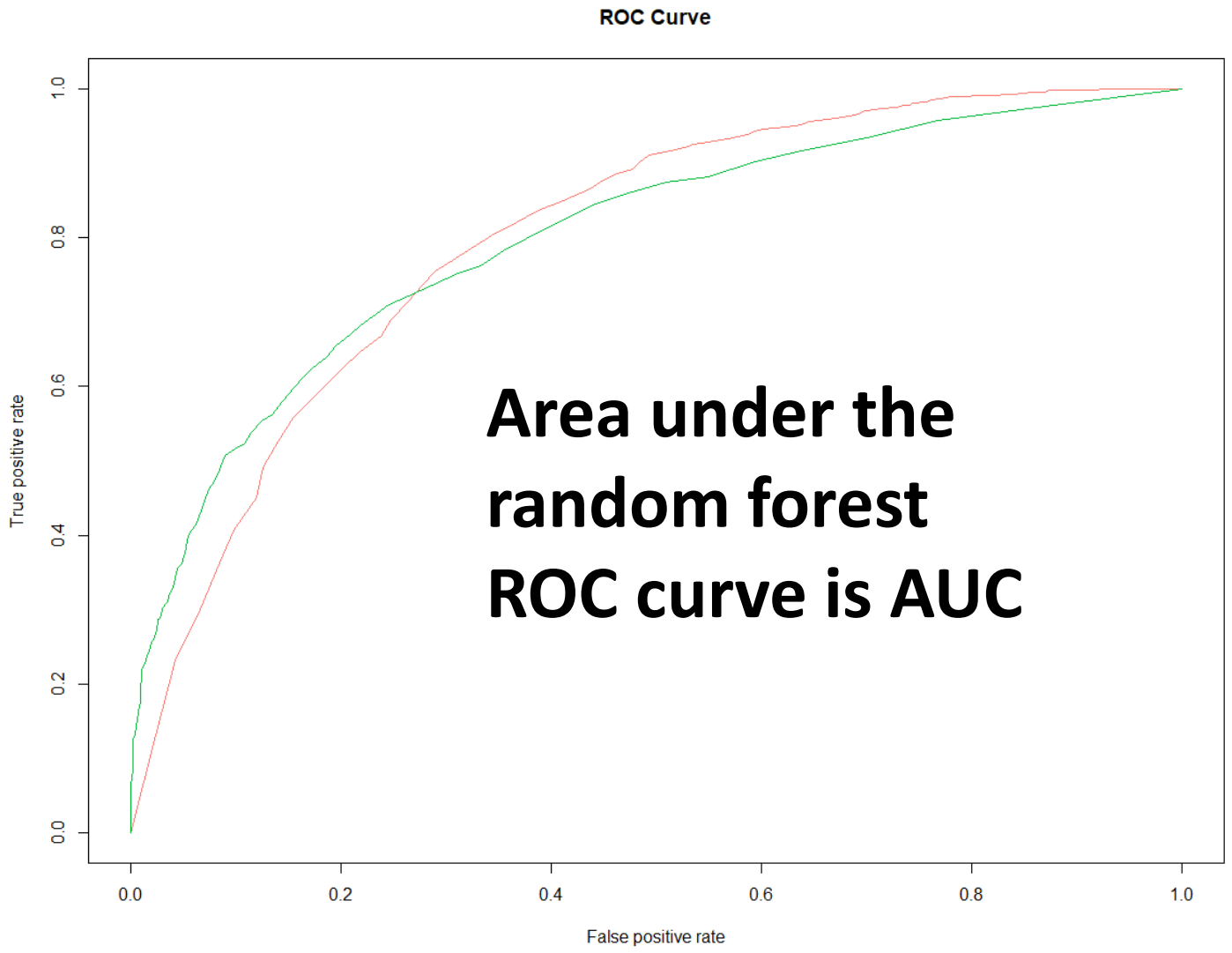
We Approach Our Question as a Binary Classification Problem

- Source files added
- Test files changed
- Dependencies added
- Source files renamed
- Author Experience with BDD
- ⋮



Our Top Performing Model Has 0.76 AUC

- Random Forest (0.76)
- Naïve Bayes (0.74)
- Logistic Regression (0.70)

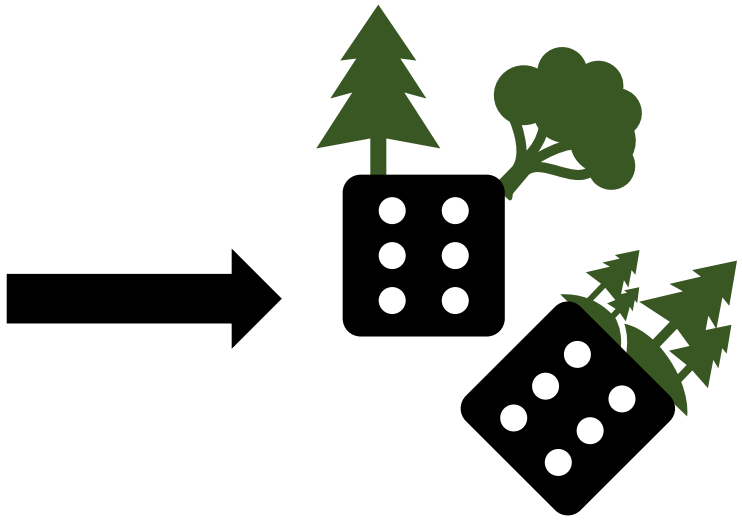


Episode 3: Return of the RQs

- (RQ1): Can we accurately identify co-changes between .feature files and source code files?
- (RQ2): Can we accurately predict when co-changes between .feature files and source code files are necessary?
- **(RQ3): What are the most significant characteristics for predicting co-changes between .feature files and source code files?**

Obtain an AUC Value after Eliminating an Attribute

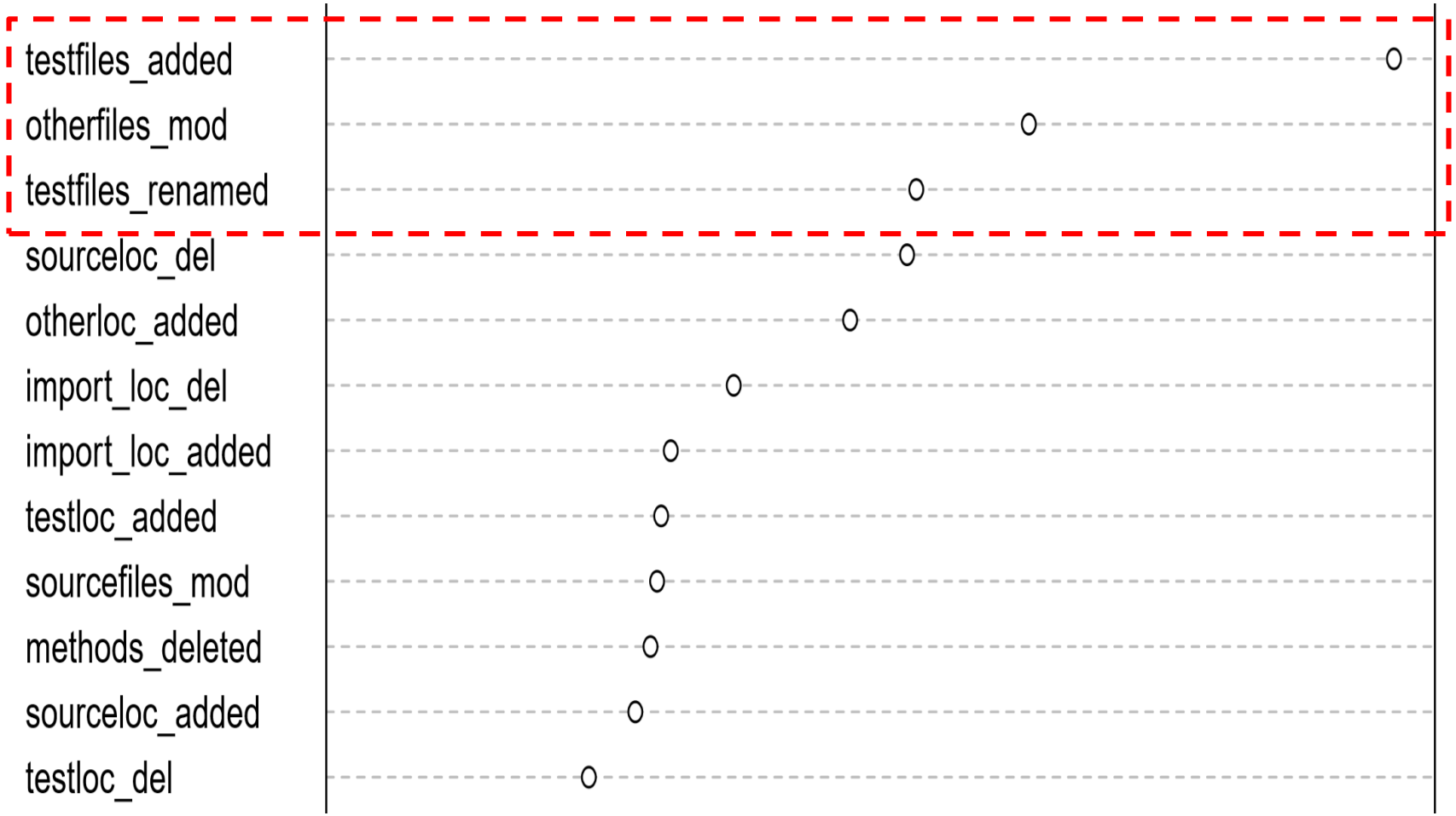
- ~~Source file added~~
- Test file added
- Other file added
- Source file modified
- Test file modified



AUC Decrease?

Test Files and All Other Files Changes Are the Strongest Predictors

- **Test files added**
- **Other files modified**
- **Test files re-named**



Co-changes Can Be Detected and Predicted!

RQ1: We can detect BDD co-changes with 79% accuracy

RQ2: Our top classification technique yields a 0.76 AUC

RQ3: Test files added and renamed, and other files modified are strongest predictors

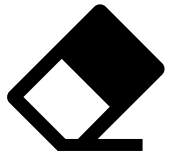
Modify .Feature Files to Maximize BDD Advantages When…

Adding and renaming test files

☐

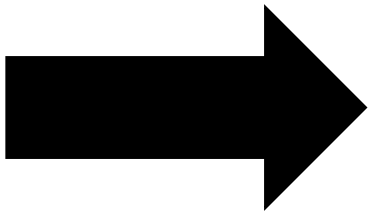
Modifying all other files

☐



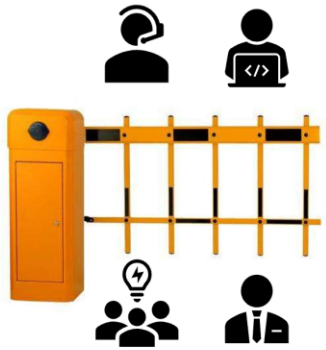
Deleting source code

☐



Modify .feature files!

Motivation



BDD

- Out-of-synch co-change reduces benefits of BDD
- New developers entering a project struggle to understand requirements

Identify code characteristics that can predict co-changes between .feature files and source code files (BDD co-changes) to help developers reduce out-of-sync BDD co-changes

Analysis of Links in the Sample

Sample size of 451.

↓

360 co-changing work items actually linked together.

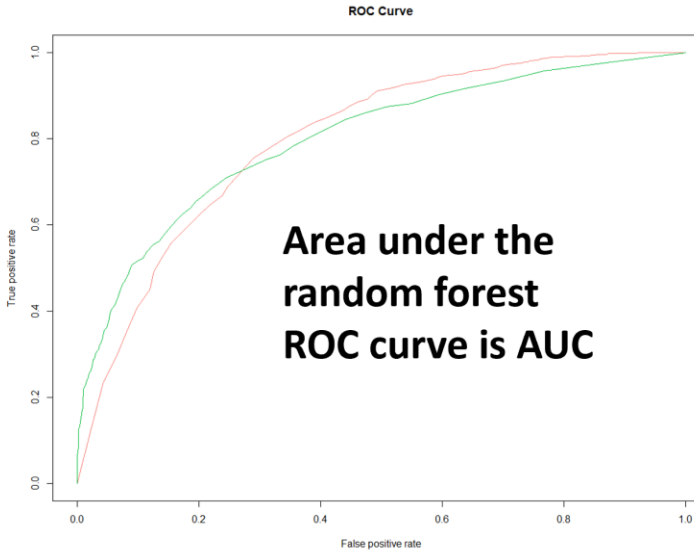
↓

80% agreement rate after inspection by another author.

Our Approach Yields 79% Accuracy

Our Top Performing Model Has 0.76 AUC

- Random Forest (0.76)
- Naïve Bayes (0.74)
- Logistic Regression (0.70)



Area under the random forest ROC curve is AUC

Test Files and All Other Files Changes Are the Strongest Predictors

- Test files added
- Other files modified
- Test files re-named