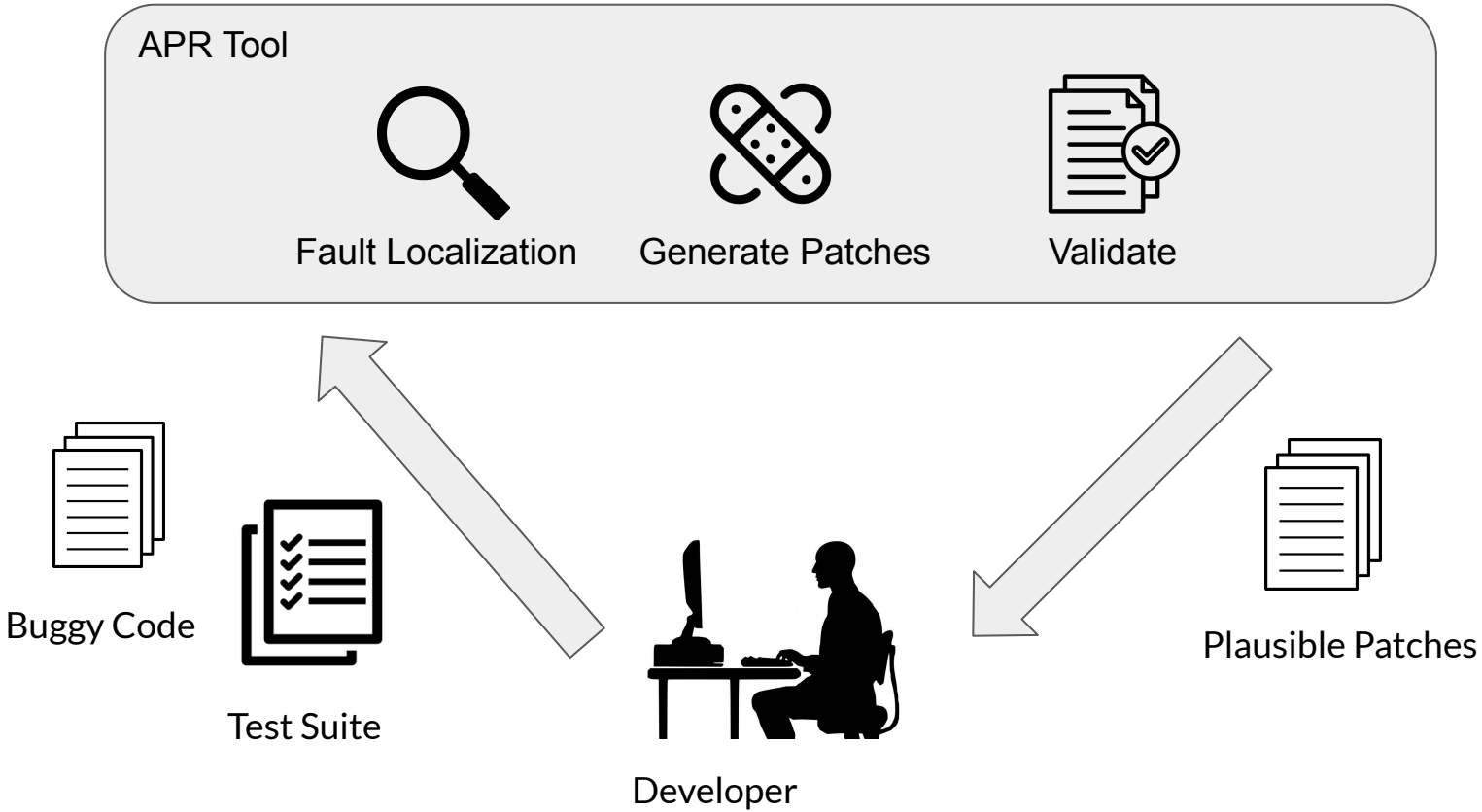
Explainable Automated Program Repair

Team: Nimantha Cooray - 190111B, Nethum Lamahewage - 190349K

Internal Supervisors: Dr. Sandareka Wickramanayake, UoM and Dr. Nisansa de Silva, UoM
External Supervisor: Dr. Ridwan Shariffdeen, NUS

Automated Program Repair (APR) Workflow



Explaining Learning-based APR tools

Black-box APR Model

Generates Patch

Developer

Why did the model generate this patch for this particular bug?

```
int k = 0;
for (Matcher m : matchers) {
-   if (m instanceof CapturesArguments) {
+   if (m instanceof CapturesArguments && i.getArguments().length > k) {
        ((CapturesArguments) m).captureFrom(i.getArguments()[k]);
    }
}
```

Explaining Learning-based APR tools

Explanation:

This is the part that affected the generated patch the most.

```
int k = 0;
for (Matcher m : matchers) {
    if (m instanceof CapturesArguments) {
        ((CapturesArguments) m).captureFrom(i.getArguments()[k]);
    }
}
```

Patch:

```
  int k = 0;
  for (Matcher m : matchers) {
-     if (m instanceof CapturesArguments) {
+     if (m instanceof CapturesArguments && i.getArguments().length > k) {
          ((CapturesArguments) m).captureFrom(i.getArguments()[k]);
      }
  }
```

Problem Statement

Develop a post-hoc feature-based local explanation method for learning-based APR tools

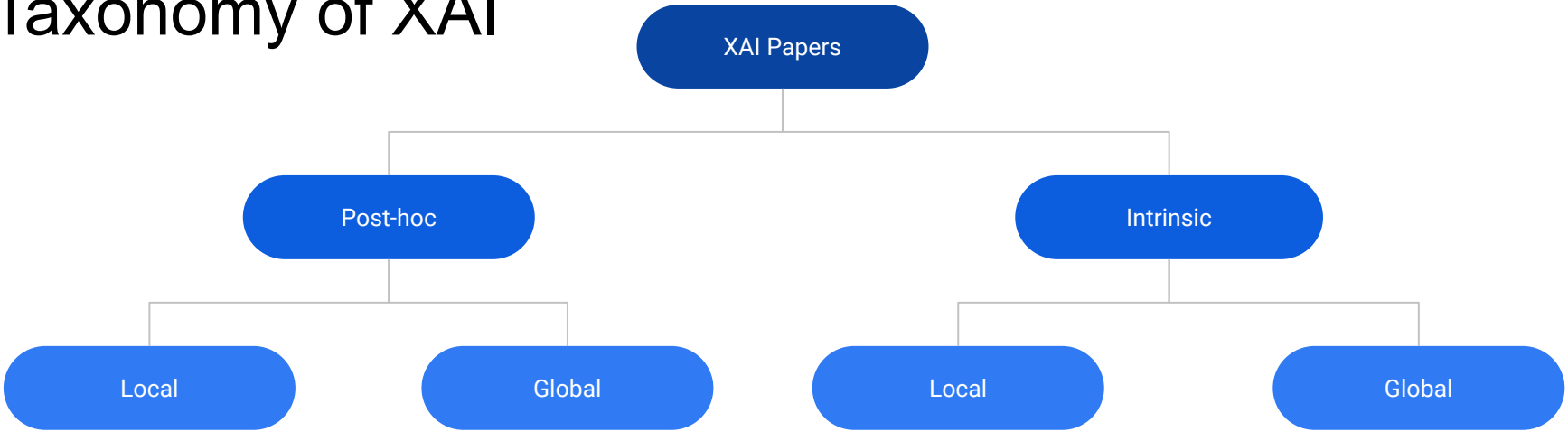
Objectives

- Apply existing XAI techniques on existing APR tools
- Develop a new method to provide local post-hoc explanations for learning-based APR tools
- Evaluate the usefulness of both existing and new method by conducting a user study

RELATED WORK

Taxonomy of Literature Reviewed

Taxonomy of XAI



Shankaranarayana et al. 2019
Hakkoum et al. 2020
Plumb et al. 2020
Garreau et al. 2021
Kokalj et al. 2021
Pornprasit et al. 2021
Slack et al. 2021
Ferreira et al. 2022
Wang et al. 2022
Widyasari et al. 2022

Lundberg et al. 2017
Mosca et al. 2022

Vaughan et al. 2018
Humphreys et al. 2019
Zaeem et al. 2021

Mori 2015

Phases

Phase 1: Apply existing XAI techniques to existing APR tools

- XAI techniques: SHAP, Integrated Gradients

Phase 2: Develop a novel XAI technique for APR, apply it to existing APR models, and evaluate the results with a user study

SHAP Explanation

- SHapley Additive exPlanations
- Game theory-based approach to explain the output of any machine learning model
- When applied to a text-to-text model (e.g. APR), gives an attribution score for each input token for each output token

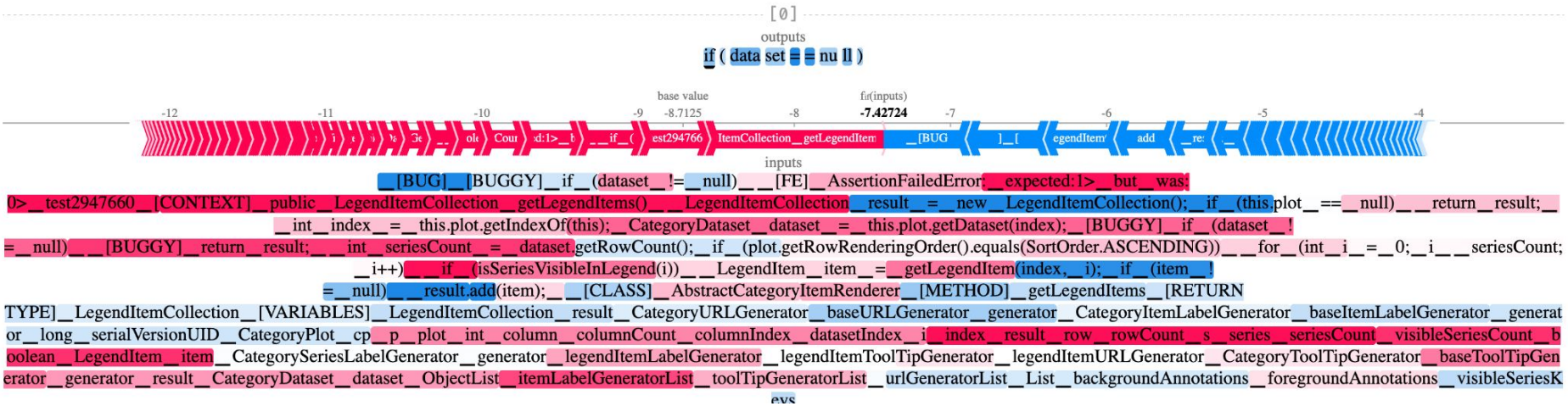
Example bug: Chart 1

```
1  public LegendItemCollection getLegendItems() {
2    LegendItemCollection result = new LegendItemCollection();
3    if (this.plot == null) {
4      return result;
5    }
6    int index = this.plot.getIndexOf(this);
7    CategoryDataset dataset = this.plot.getDataset(index);
8    if (dataset != null) { // buggy line
9      return result;
10   }
11   int seriesCount = dataset.getRowCount();
```

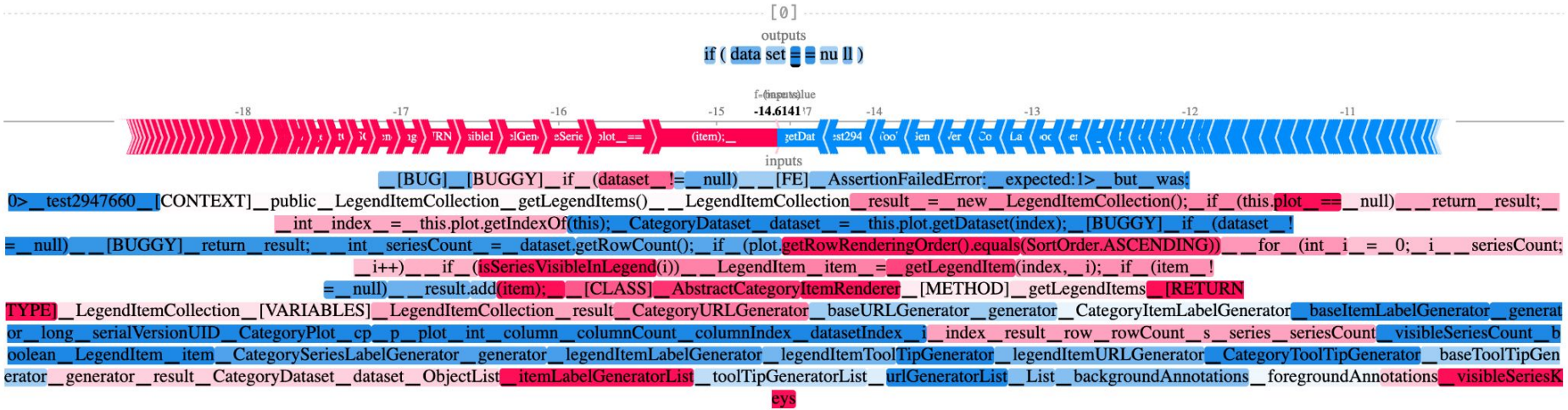
Patch: `if (dataset == null) {`

- From JFreeChart: a 2D chart library for Java applications
- Bug is in class `AbstractCategoryItemRenderer`, method `getLegendItems`
- The method should return a collection of legend items that this renderer is responsible for drawing
- In the case that there aren't any, the returned collection should be empty
- Currently in the, `dataset != null` case, it is returning the empty collection
- Instead, it should be doing that if `dataset == null`

SHAP Explanation: Chart 1, attribution for `if`



SHAP Explanation: Chart 1, attribution for `=`



Current Limitations of SHAP Explanation

- User will need to be guided on how to read the explanation
- Human understandability
  - Complexity of an nxm matrix of attributions
  - Not specific for code input/output

Solutions for current problems: SHAP aggregation

Example SHAP values matrix:

- Each row is an input token
- Each column is an output token
- Approaches to aggregation
  - Sum
  - Absolute sum
  - PCA
  - Softmax
- In experiments, the first method gave better results

	if	(	dataset	==	null	)
[BUG]						
[BUGGY]						
if						
(						
dataset						
!=						
null						
)						
{						

Solutions for current problems: Text explanation

```
1  public LegendItemCollection getLegendItems() {
2    LegendItemCollection result = new LegendItemCollection();
3    if (this.plot == null) {
4      return result;
5    }
6    int index = this.plot.getIndexOf(this);
7    CategoryDataset dataset = this.plot.getDataset(index);
8    if (dataset != null) { // buggy line
9      return result;
10   }
11   int seriesCount = dataset.getRowCount();
```

Patch: `if (dataset == null) {`

- Bug is caused by the `` `if` `` condition in line 8
- SHAP aggregation tells us the most impactful token was the `` `(` `` in line 7
- The cause of the patch is the method invocation
- To a human developer's eyes: the bug is caused by that method invocation potentially returning a `` `null` `` value

Explanation: *This patch was generated because of the method invocation `this.plot.getDataset(...)`*

Solutions for current problems: Text explanation

```
shap_values = shap(model, input_code)
aggregated = aggregate_shap(shap_values)

code_ast = parse(input_code)
most_relevant_node = get_most_relevant_node(code_ast, aggregated)
node_type = most_relevant_node.type

if node_type == "METHOD_INVOCATION":
    method_name = most_relevant_node.name
    print(f"This patch was generated because of the method invocation {method_name}")
elif node_type == "RETURN":
    print("This patch was generated because of this return statement")
elif node_type == "CATCH":
    error = most_relevant_node.exc
    print(f"This patch was generated because of {error} in the catch statement")
...
```

Solutions for current problems: Patch evolution

- Another perspective
- If SHAP gives high attribution to a set of tokens, then removing those tokens from the input should have a significant effect on the patch generated by the model
- We take the original input, remove the most relevant tokens and get the model output
- Then, we add each of those tokens one by one and get the model output at each step
- This will show the patch evolution

Solutions for current problems: Patch evolution

```
1  private void removeUnreferencedFunctionArgs(Scope fnScope) {
2    Node function = fnScope.getRootNode();
3
4    Preconditions.checkState(function.isFunction());
5    if (NodeUtil.isGetOrSetKey(function.getParent())) {
6      return;
7    }
```

Correct patch: `if (!removeGlobals) { return; }`

Actual patch: `if (fnScope.getParent() == null) { return; }`

- SHAP aggregation tells us the most impactful token was the `` `(` `` in line 4 (after `` `checkState` ``)
- In decreasing order of impact, the tokens are
  - `` `(` `` (line 4)
  - `` `checkState` `` (line 4)
  - `` `getRootNode` `` (line 2)
  - `` `fnScope` `` (line 2)

Solutions for current problems: Patch evolution

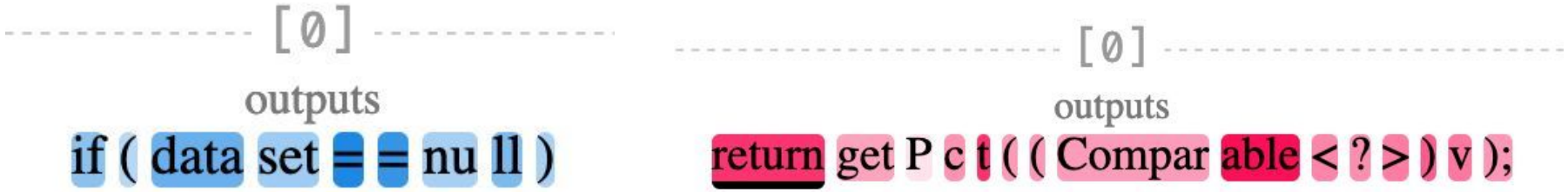
```
1  private void removeUnreferencedFunctionArgs(Scope fnScope) {
2    Node function = fnScope.getRootNode();
3
4    Preconditions.checkState(function.isFunction());
5    if (NodeUtil.isGetOrSetKey(function.getParent())) {
6      return;
7    }
```

```
    if (fnScope.getInstanceof Node) { return; }
```

Solutions for current problems: Patch evolution

Solutions for current problems: Subword Tokenization

The original subword-tokenizer used by RewardRepair APR model causes to assign SHAP values to subwords.



Solutions for current problems: Subword Tokenization

- Approaches to fix the problem:
  1. Train the model again with added Java tokens to the tokenizer
  2. Build a new tokenizer that can fix the problem and use it with SHAP

Approach 1 Results:

- Trained the model again for 1 epoch
- The tokens were still subword tokenized

Solutions for current problems: Subword Tokenization

Approach 2 Results:

- SHAP can be provided a masker.
- SHAP uses the tokenizer in the masker to mask out tokens
- Built a tokenizer that can correctly tokenize valid Java tokens

Original code

```
class Student implements Comparable<Student> {
```

T5 tokenizer

```
['_class', '_Student', '_implement', 's', '_Compar', 'able', '<', '_Student', '>', '{']
```

New tokenizer

```
['class', 'Student', 'implements', 'Comparable', '<', 'Student', '>', '{']
```

Solutions for current problems: Subword Tokenization

T5 Tokenizer can not tokenize Java keywords, operators, separators and class/interface names from standard library correctly.

It split following into subwords:

- 15 / 68 (22.06%) Java keywords
- 19 / 38 (50%) Java operators
- 1 / 12 (8.3%) Java Separators
- 3988 / 4176 (95.5%) Java class/interface names from standard library

Since our tokenizer has all keywords, operators, separators and class/interface names from standard library, it can tokenize them without splitting.

T5 tokenizer (RewardRepair) vocabulary size: 32,104

New Tokenizer vocabulary size: 35,758

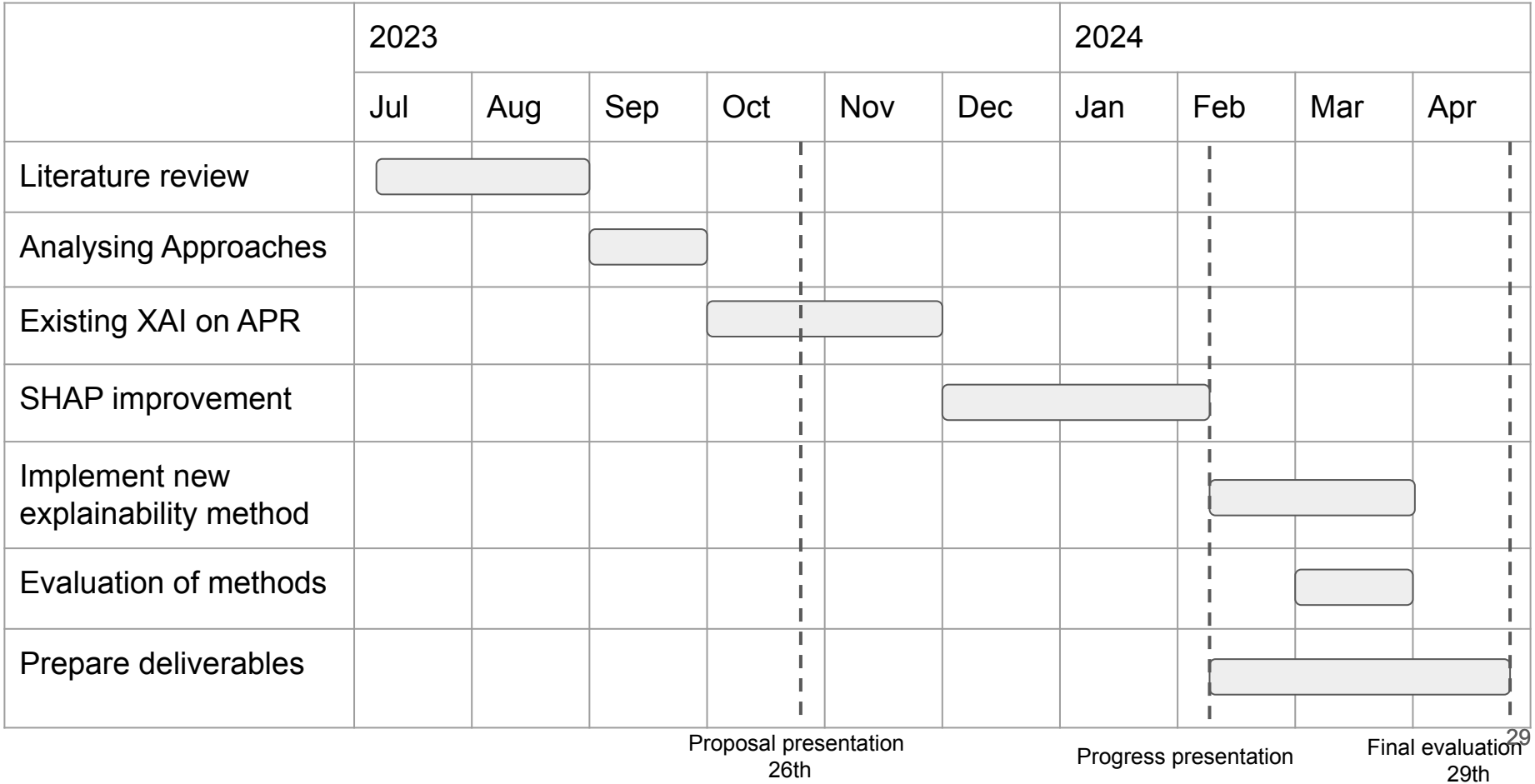
Progress

Task	Status
Literature review	Completed
Analyze approaches	Completed
Applying SHAP on SequenceR, RewardRepair, and SelfAPR	Completed
Improving SHAP's explanation	In Progress
New XAI method implementation	Not Started
Evaluating both methods with a user study	Not Started
XAI method paper	Not Started

Our XAI method for APR

- Our method is to use AST-based perturbations to generate explanations
- Steps
  - Take the input code and get the model output
  - Perturb the code, in terms of AST, and get model outputs
  - Compare input-output pairs and identify most important part of input code

Revised Timeline



THANK YOU

Methodology

```
int k = 0;
for (Matcher m : matchers) {
     if (m instanceof CapturesArguments) {
          ((CapturesArguments) m).captureFrom(i.getArguments()[k]);
     }
}
```

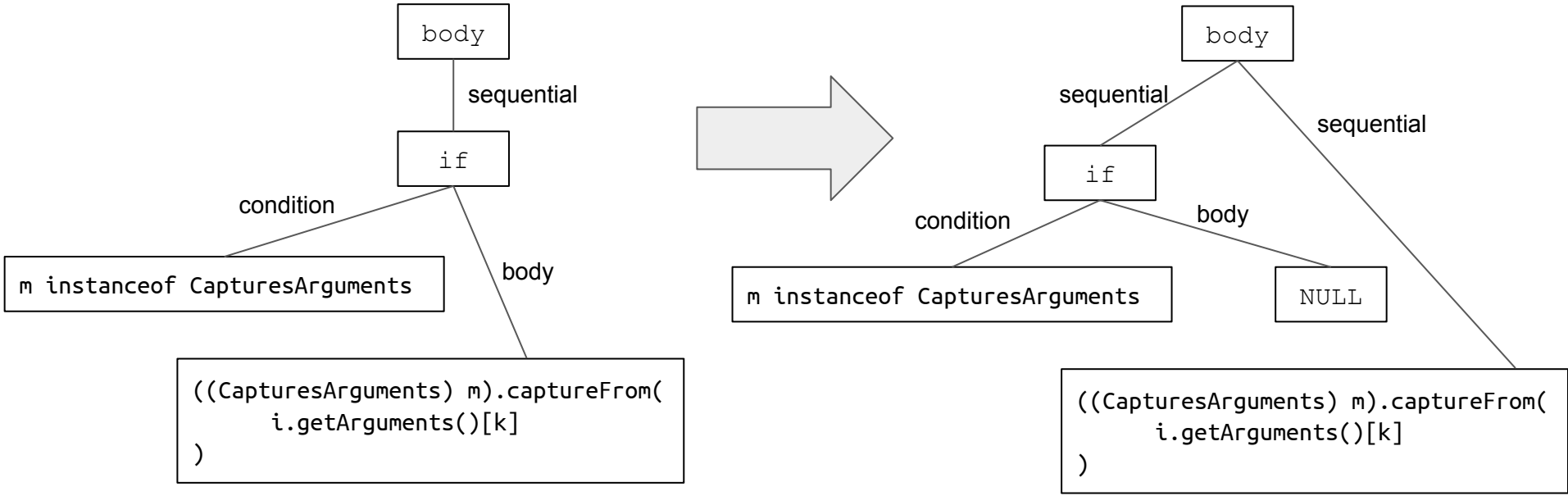
Text-based perturbation

Code-based perturbation

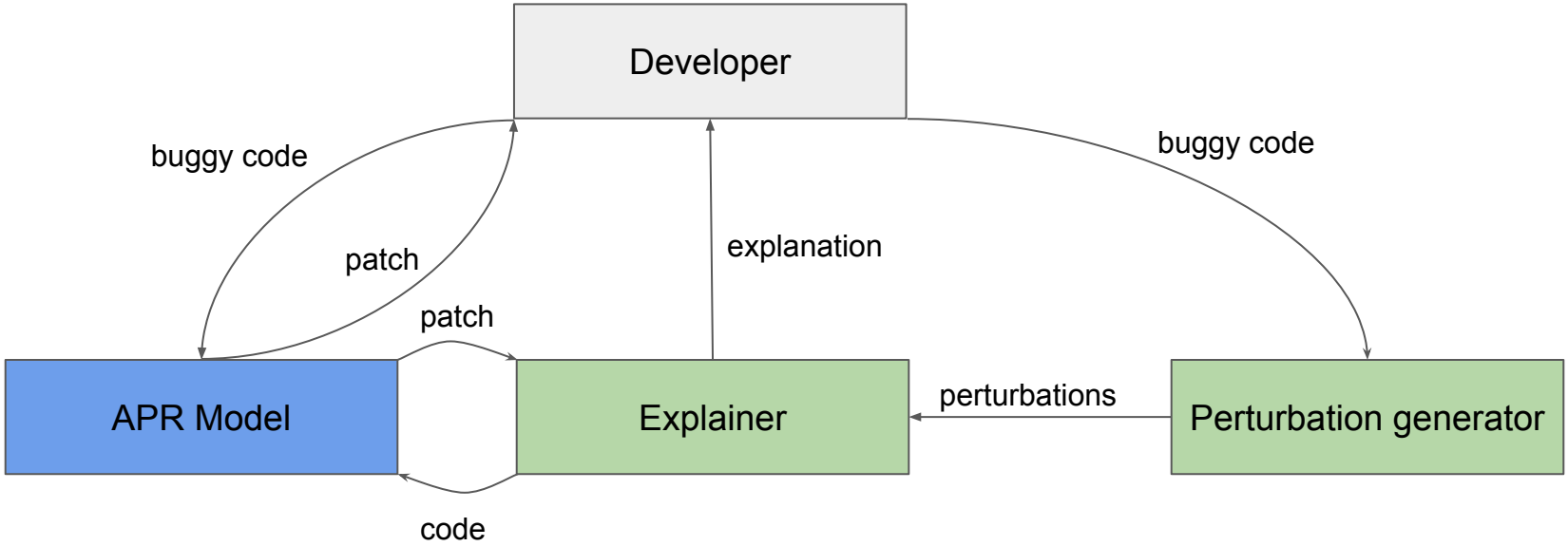
```
for (Matcher : m matchers) {
```

```
if (m instanceof CapturesArguments) {
}
((CapturesArguments) m).captureFrom(i.getArguments()[k]);
```

Methodology – perturbation using AST



Methodology



Previous Timeline