

Reachability Survey Results

James Yoo

Software Practices Lab

University of British Columbia



Reachability Questions

“search across feasible paths through a program for target statements matching search criteria.” [1]

find SC in TR

compare(TR_a , TR_b)

Reachability Questions

- 13 developers in a research lab [1]
 - ½ bugs inserted were associated with reachability questions
- 460 professional developers [1]
 - Asked reachability questions > 9 times a day
 - 82% rated 1 or more as difficult to answer
- 17 developers in the field [1]
 - 9 of the 10 longest activities were associated with reachability questions.

Overview

- Live date: November 30, 2021
- Focused on developers who use statically-typed languages
- 9 Likert-scale questions related to:
 - [Reachability questions](#) (LaToza et al.)
 - [Hard-to-answer questions about code](#) (LaToza et al.)

Overview

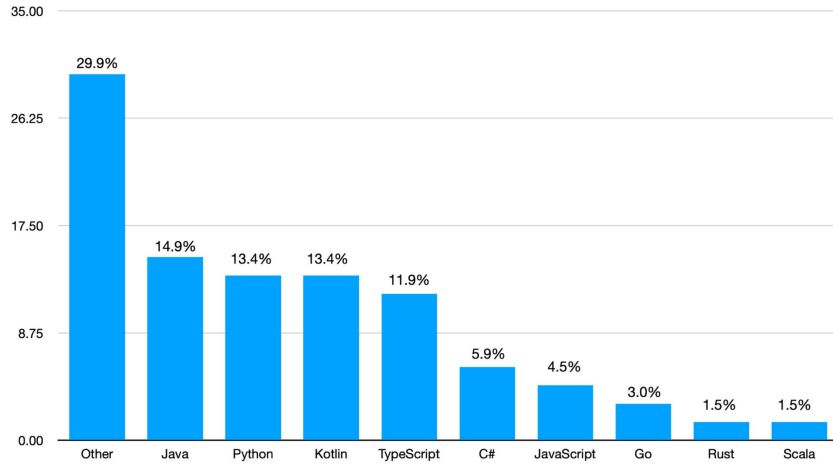
- Live date: November 30, 2021
- Focused on developers who use statically-typed languages
- 9 Likert-scale questions related to:
 - [Reachability questions](#) (LaToza et al.)
 - [Hard-to-answer questions about code](#) (LaToza et al.)
- Survey started 101 times, finished 67 times:
 - **Completion rate: 66%**

Popular Reachability Questions

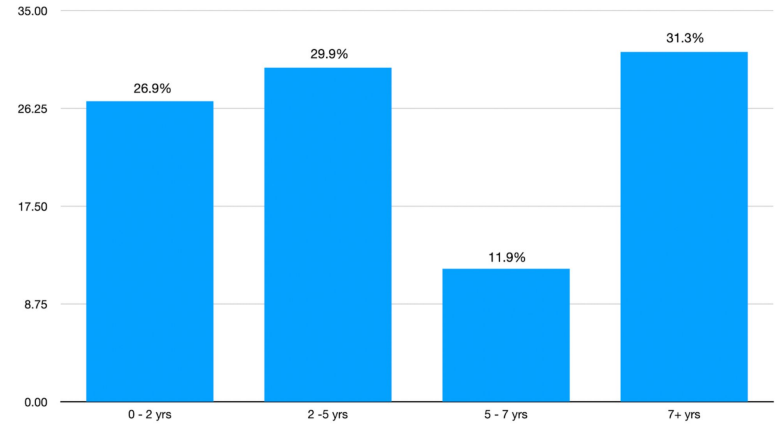
1. Where does this value come from, and how is it formed?
2. Given some data, which parts of it are modified downstream?
3. What does the control-flow look like between two locations in a program?

Demographics

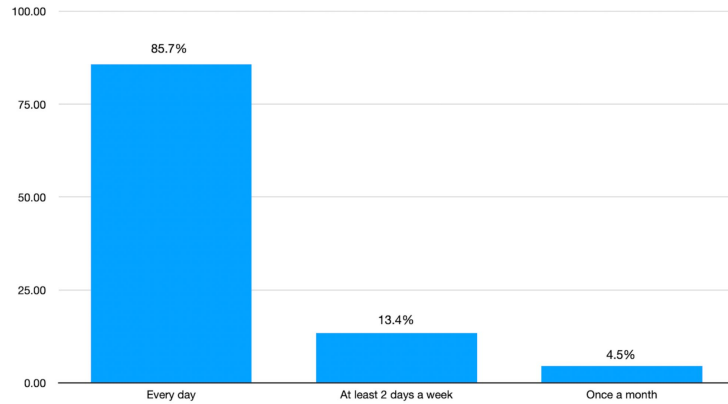
Primary Development Language



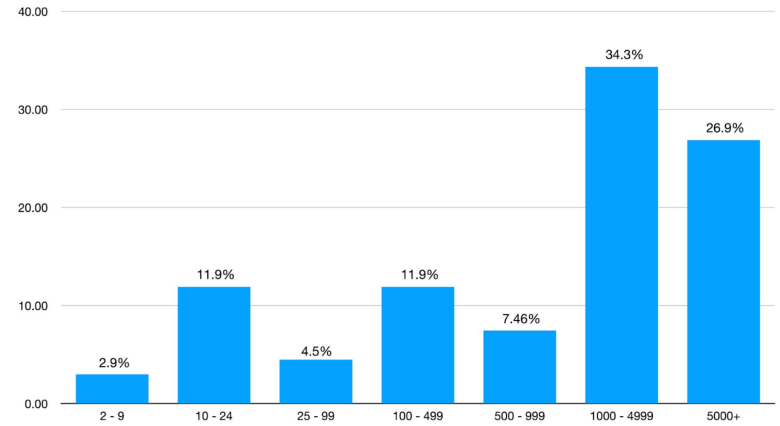
Years of Paid Software Development Experience



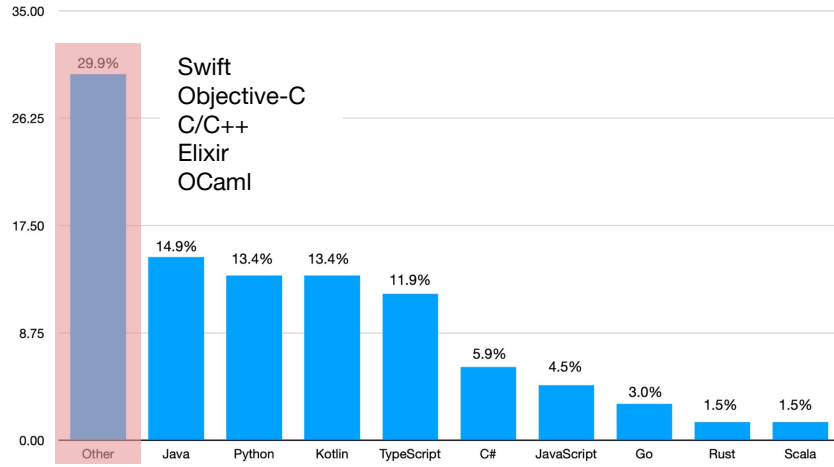
How Often Do Our Respondents Write Code?



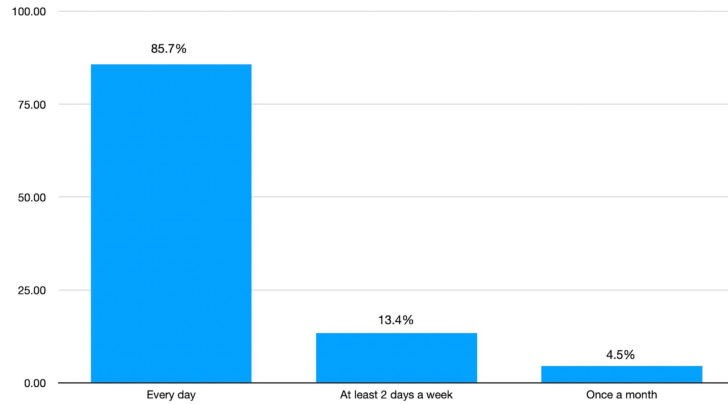
Number of People at Workplace



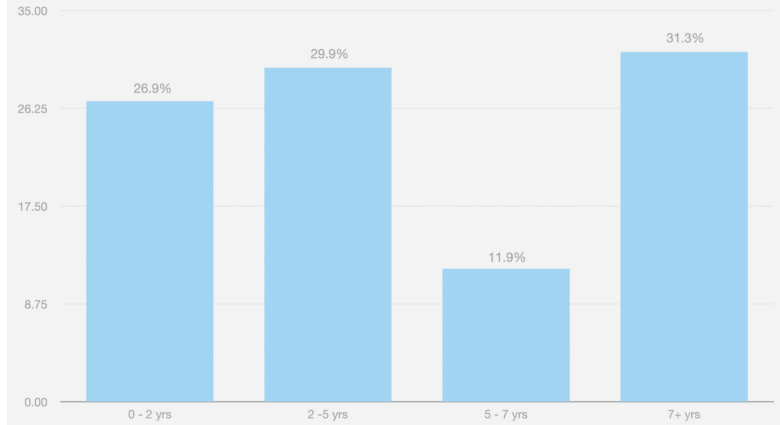
Primary Development Language



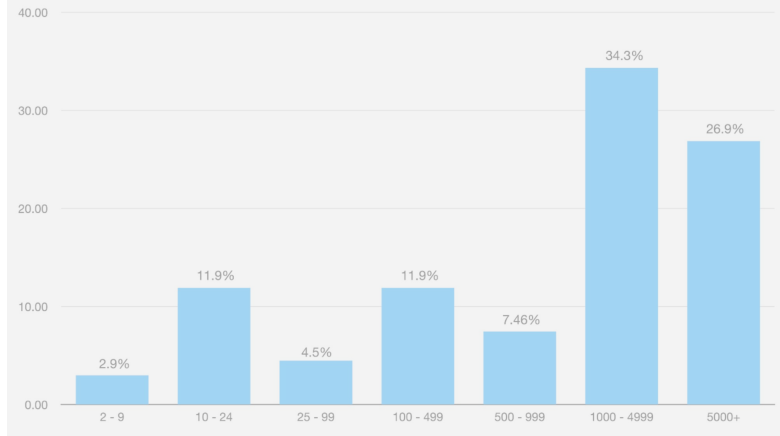
How Often Do Our Respondents Write Code?



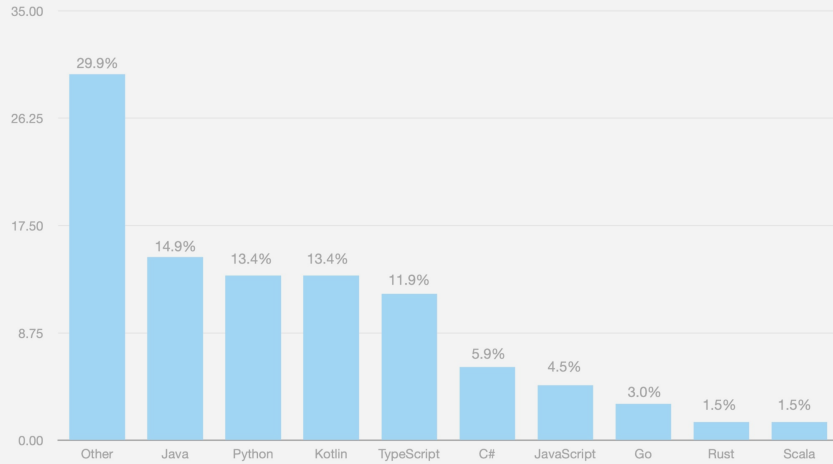
Years of Paid Software Development Experience



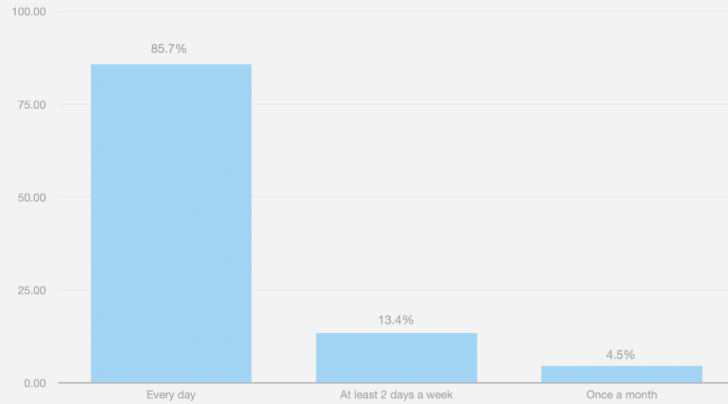
Number of People at Workplace



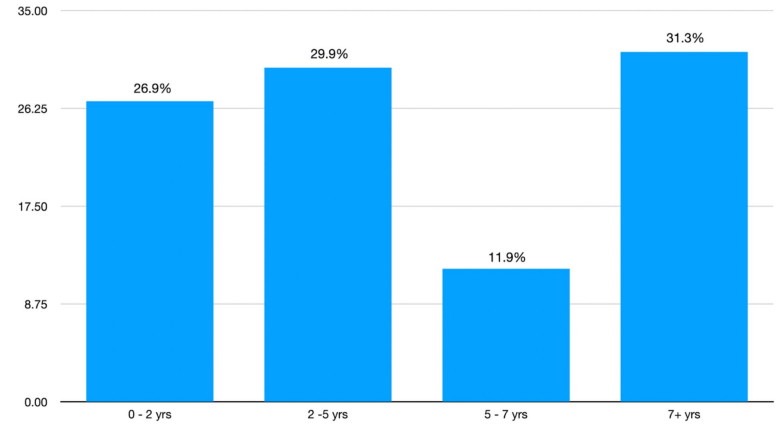
Primary Development Language



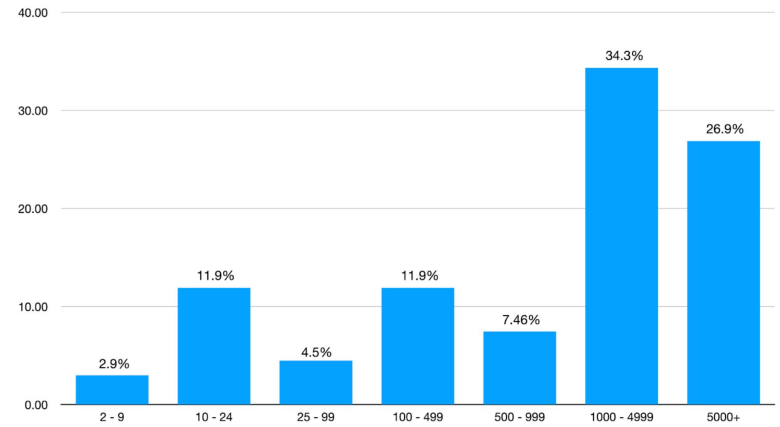
How Often Do Our Respondents Write Code?



Years of Paid Software Development Experience




Number of People at Workplace



Results

Where does this value come from, and how is it formed?

Are you interested in how this parameter is formed? E.g., the method calls and parts of the program that contribute to its creation.



```
public Map<Category, Double> computeTotalsByCategory(Map<ShopItem, Integer> items) {  
    Map<Category, Double> totalsByCategory = new HashMap<>();  
    items.forEach((item, quantity) → {  
        double cost = item.getCost() * quantity;  
        totalsByCategory.put(item.getCategory(), cost);  
    });  
    return totalsByCategory;  
}
```

find SC in TR

Given some data, which parts of it are modified downstream?

```
public ShoppingCart provision() throws CartValidationException {  
    logger.info(msg: "Provisioning new cart");  
    ShoppingCart cart = CartBuilder.build();  
    if (this.hasValidState(cart)) {  
        this.register(cart);  
        return cart;  
    }  
    throw new CartValidationException("Provisioning a new cart failed");  
}
```

Are you interested in how **cart** could be **modified** in these method calls?

find SC in TR

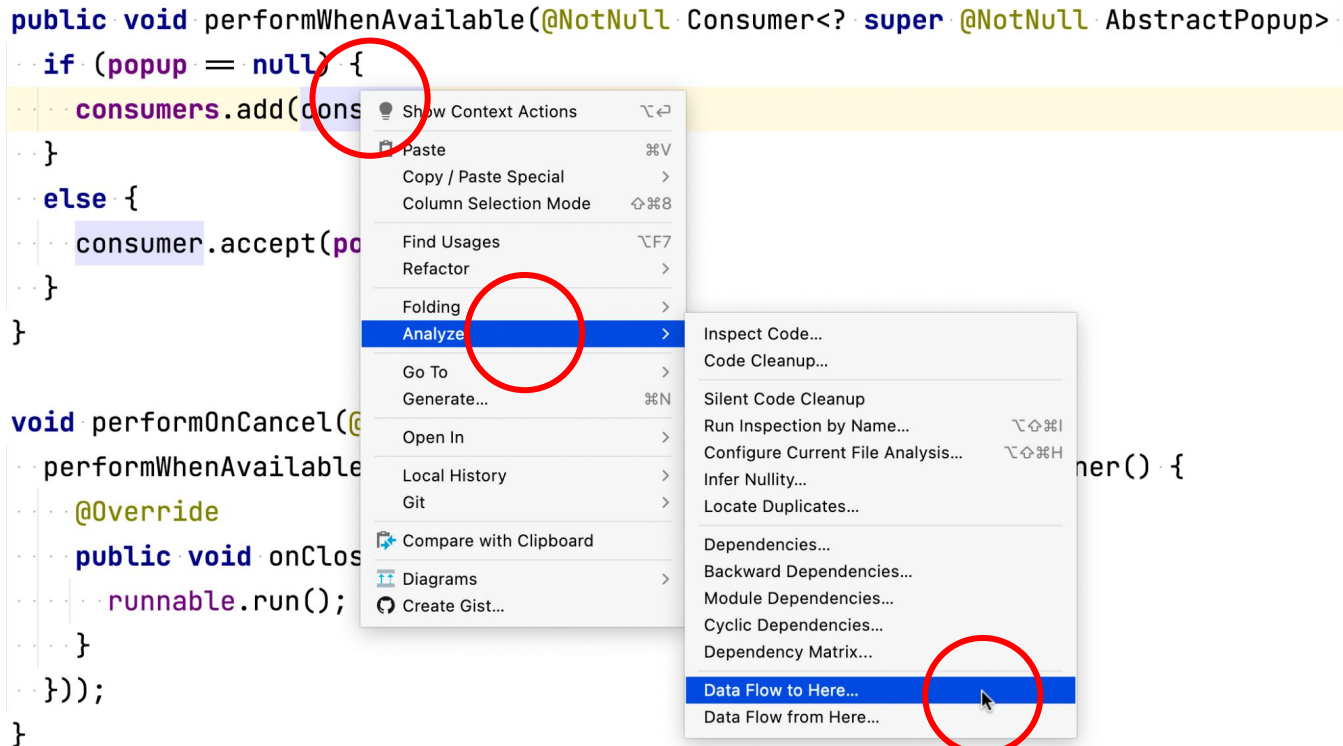
What does the control-flow look like between two locations in a program?

A developer is inspecting a method and its call hierarchy. She begins to read it line-by-line, and click through other method calls that are within it (e.g., jump to method implementation). She continues this process until she eventually arrives at a location where she wants to stop.

Having navigated through a number of methods to arrive at this location, she wants to trace back to where she started to synthesize a high-level overview of the statements between the method where she started, and the location in code where she ended.

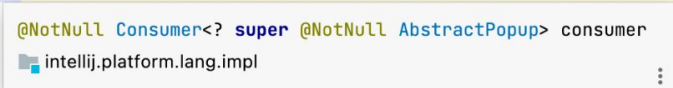
Prototype

Current Tool Discovery



Tool Discovery

```
public void performWhenAvailable(@NotNull Consumer<? super @NotNull AbstractPopup> consumer) {  
    if (popup == null) {  
        consumers.add(consumer);  
    }  
    else {  
        consumer.accept(popup);  
    }  
}
```



Proposed Tool Discovery

```
public void performWhenAvailable(@NotNull Consumer<? super @NotNull AbstractPopup> consumer) {  
    if (popup == null) {  
        consumers.add(consumer);  
    }  
    else {  
        consumer.accept(popup)  
    }  
}
```

How was 'consumer' created?

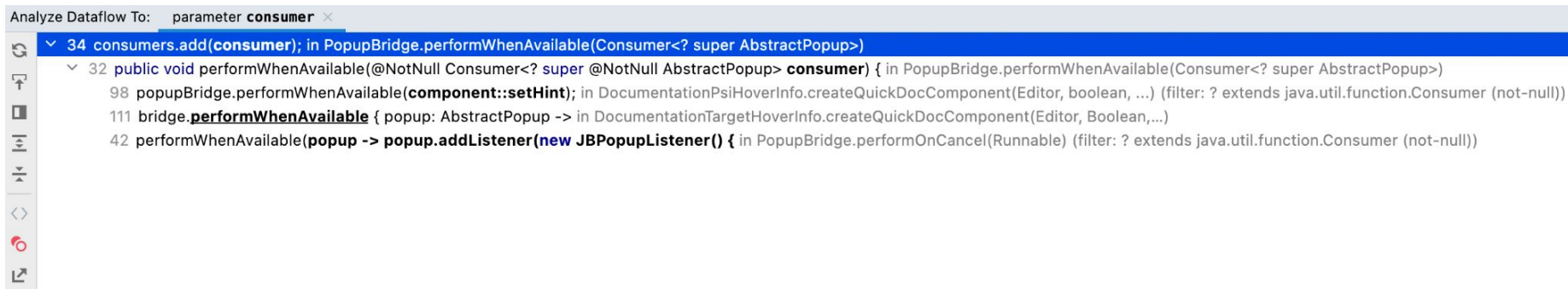
@NotNull Consumer<? super @NotNull AbstractPopup> consumer
intellij.platform.lang.impl

Current Result Visualization

Analyze Dataflow To: parameter **consumer** x

```
34 consumers.add(consumer); in PopupBridge.performWhenAvailable(Consumer<? super AbstractPopup>)
32 public void performWhenAvailable(@NotNull Consumer<? super @NotNull AbstractPopup> consumer) { in PopupBridge.performWhenAvailable(Consumer<? super AbstractPopup>)
    98 popupBridge.performWhenAvailable(component::setHint); in DocumentationPsiHoverInfo.createQuickDocComponent(Editor, boolean, ...) (filter: ? extends java.util.function.Consumer (not-null))
    111 bridge.performWhenAvailable { popup: AbstractPopup -> in DocumentationTargetHoverInfo.createQuickDocComponent(Editor, Boolean,...)
    42 performWhenAvailable(popup -> popup.addListener(new JBPopupListener()) { in PopupBridge.performOnCancel(Runnable) (filter: ? extends java.util.function.Consumer (not-null))
```

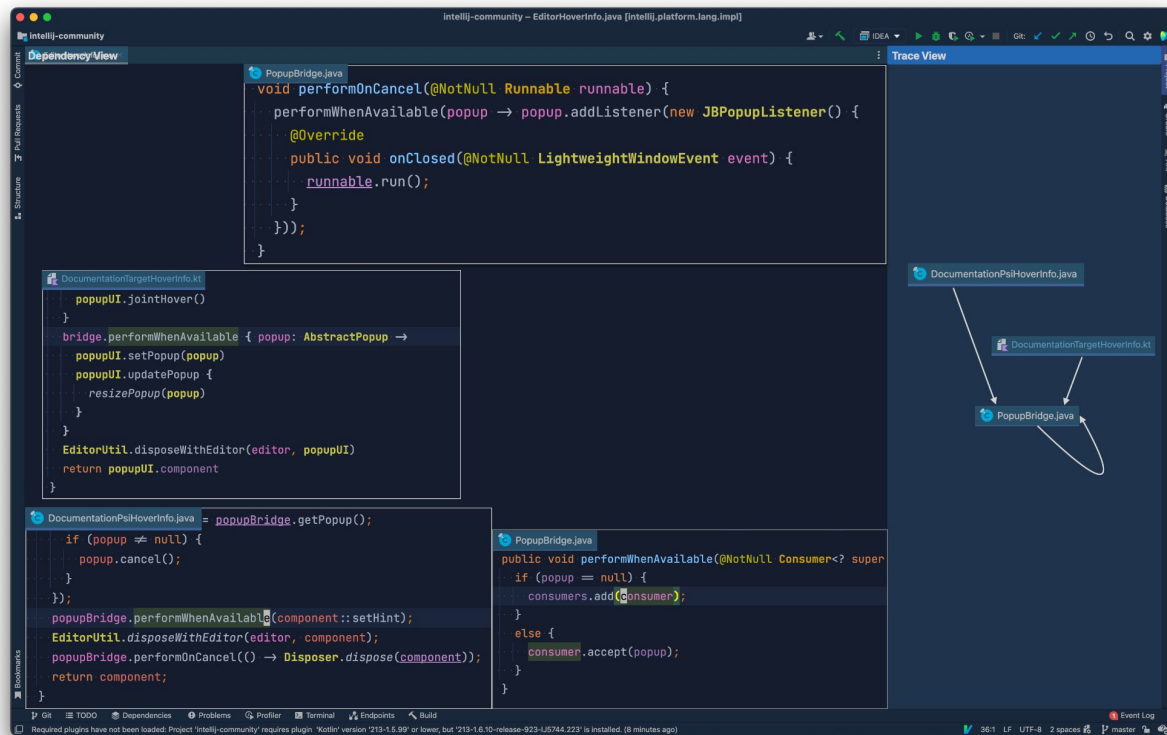
Current Result Visualization



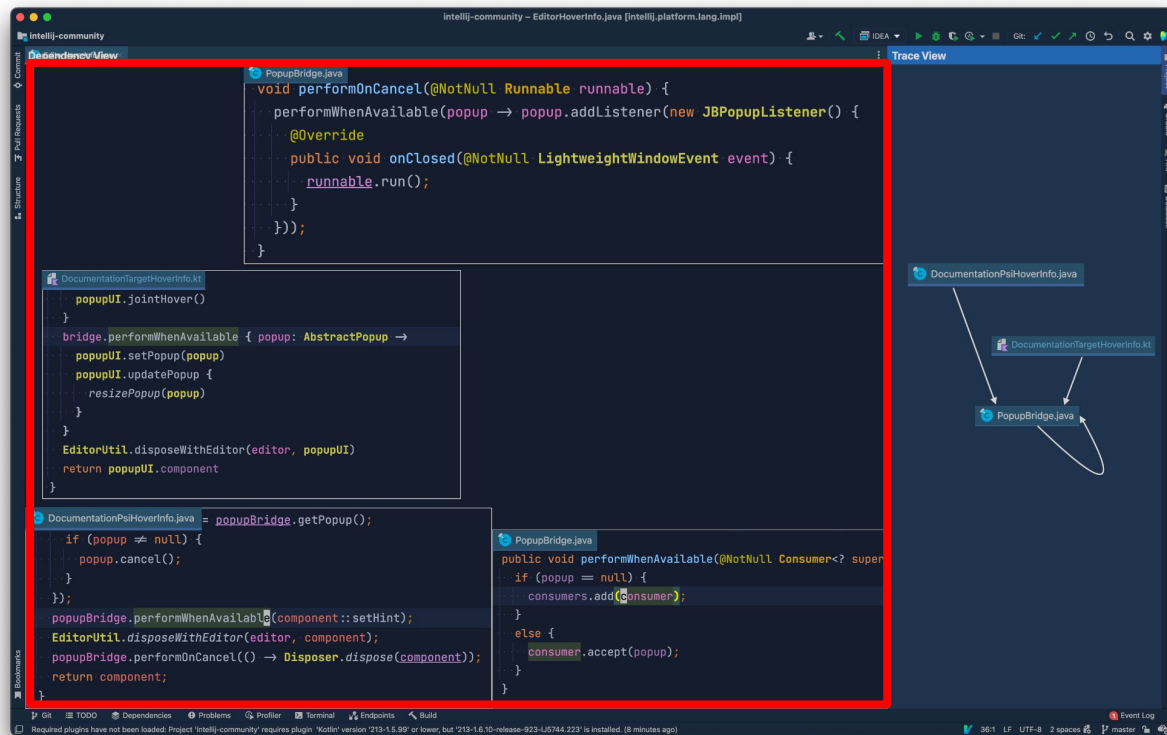
```
Analyze Dataflow To: parameter consumer x
34 consumers.add(consumer); in PopupBridge.performWhenAvailable(Consumer<? super AbstractPopup>)
32 public void performWhenAvailable(@NotNull Consumer<? super @NotNull AbstractPopup> consumer) { in PopupBridge.performWhenAvailable(Consumer<? super AbstractPopup>)
98 popupBridge.performWhenAvailable(component::setHint); in DocumentationPsiHoverInfo.createQuickDocComponent(Editor, boolean, ...) (filter: ? extends java.util.function.Consumer (not-null))
111 bridge.performWhenAvailable { popup: AbstractPopup -> in DocumentationTargetHoverInfo.createQuickDocComponent(Editor, Boolean,...)
42 performWhenAvailable(popup -> popup.addListener(new JBPopupListener()) { in PopupBridge.performOnCancel(Runnable) (filter: ? extends java.util.function.Consumer (not-null))
```

“Developers using Eclipse’s call graph exploration tool to traverse callers found it difficult both to identify feasible paths and those leading to their target.” [1]

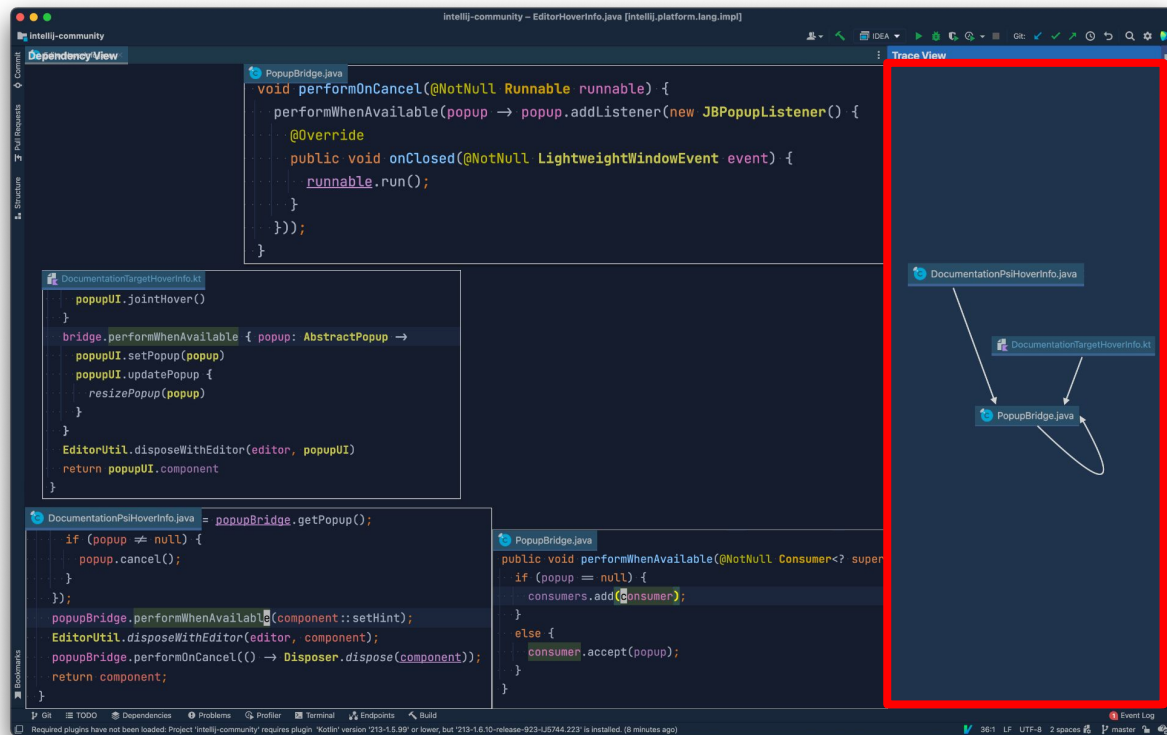
Proposed Result Visualization



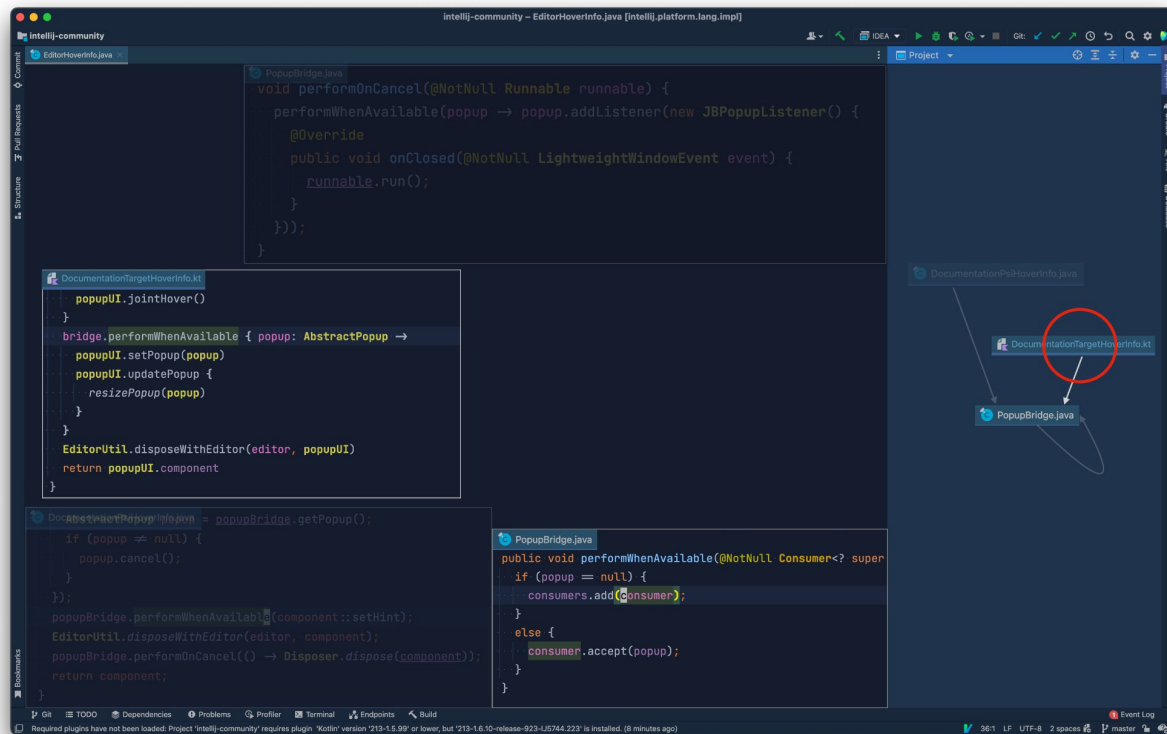
Proposed Result Visualization



Proposed Result Visualization



Proposed Result Visualization



Questions?

Links

- Survey copy:
 - www.cs.ubc.ca/~yoo/research/reachability-survey.pdf

Likert-scale Responses to Survey Questions

