

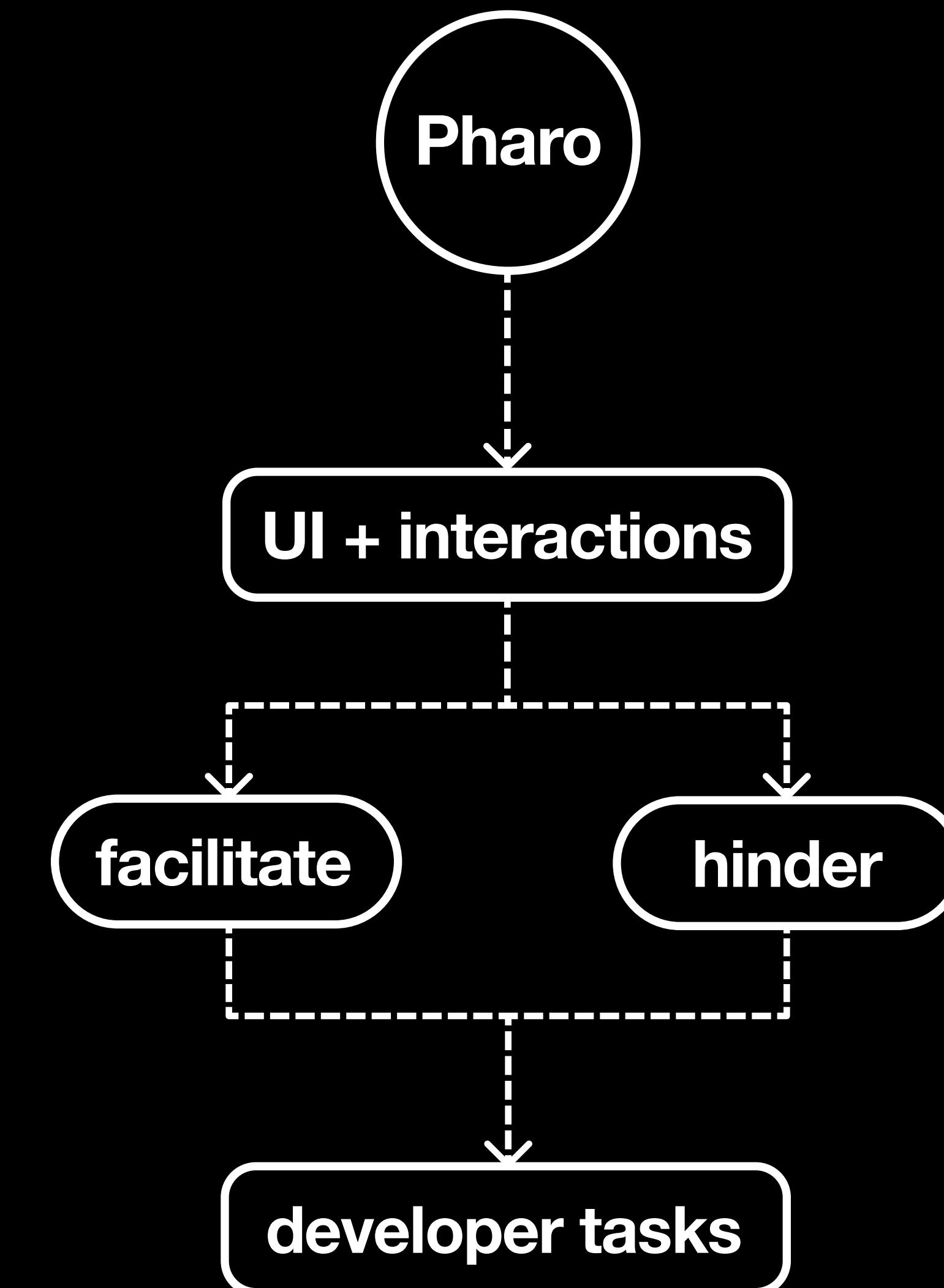
# Pharo Usability Study

Thematic Analysis

Santiago Viana

# Context

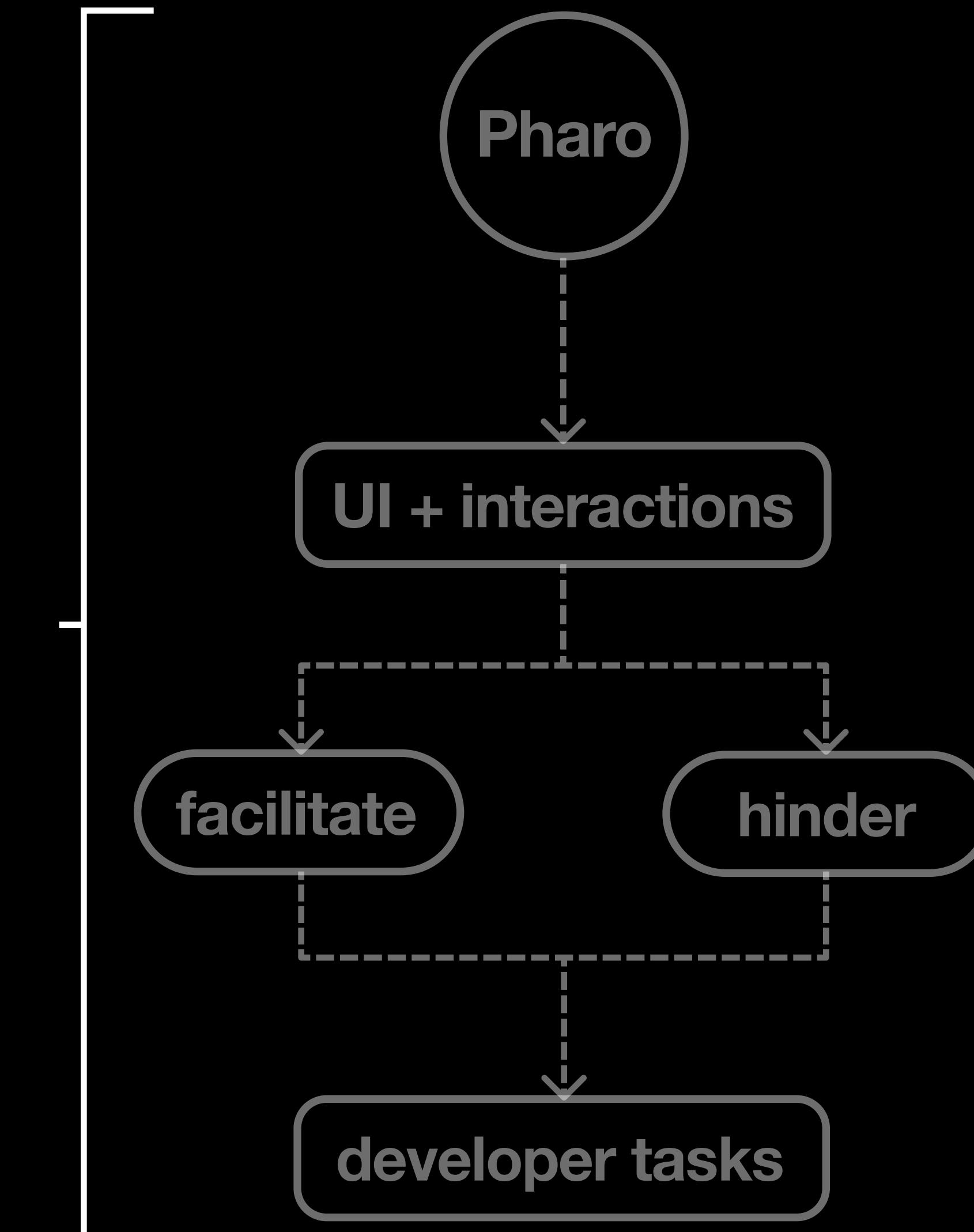
- Publication



# Analysis

## Interviews

8 Participants (1-2 hrs. each)



# Methods

## 1 Observed the Videos

Familiarizing myself with the Data

## 2 Generated Transcriptions

Fixing mistakes along the way

## 3 Created the Codes

Double checking with the research team

## 4 Decide what to Code

Using Thematic analysis tools (Tagette)

## 5 Grouping Codes into Themes

*(Themes are at a provisional stage)*

# Methods

1

# Observed the Video

# Familiarizing myself with the Data

# Generated Transcriptions

# Fixing mistakes along the way

3

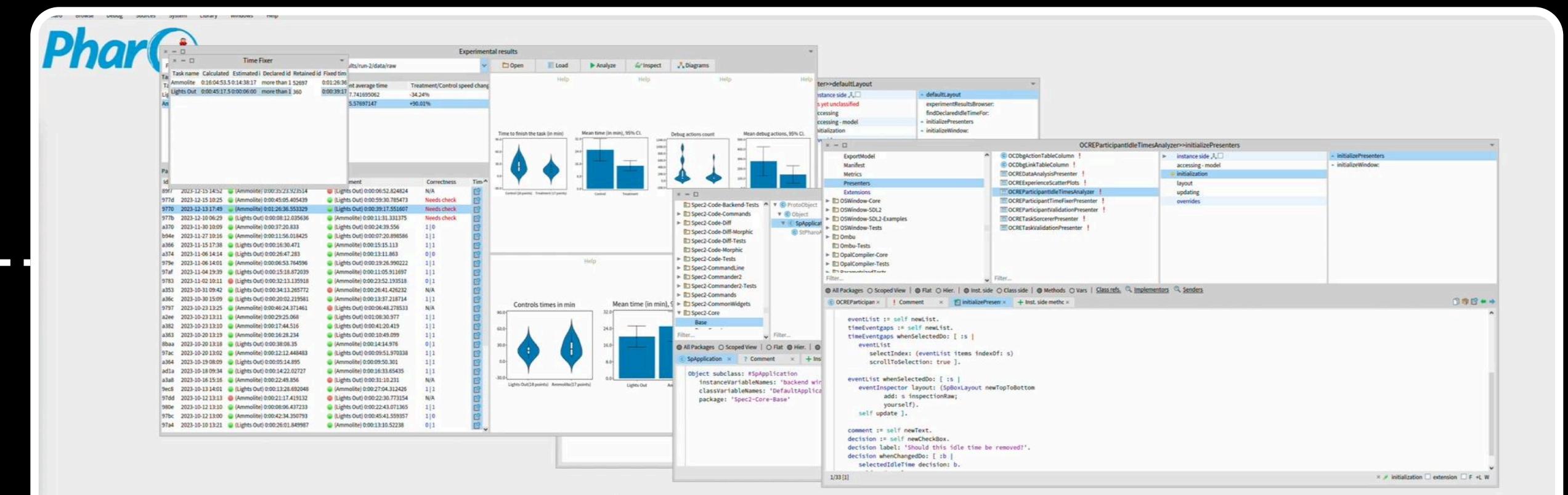
# Created the Codes

4

# Decide what to Code

5

# Grouping Codes into Themes

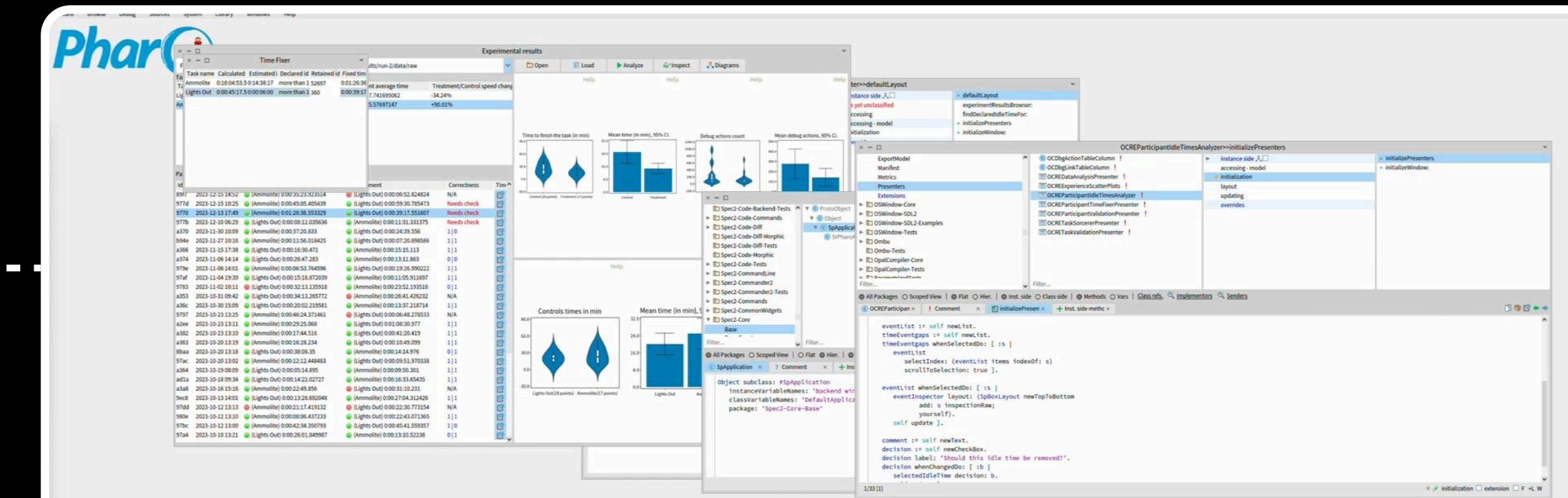


# Methods

1

## Observed the Videos

Familiarizing myself with the Data



2

## Generated Transcriptions

Fixing mistakes along the way

Tasks

3

## Created the Codes

Double checking with the research team

4

## Decide what to Code

Using Thematic analysis tools (Taguette)

5

## Grouping Codes into Themes

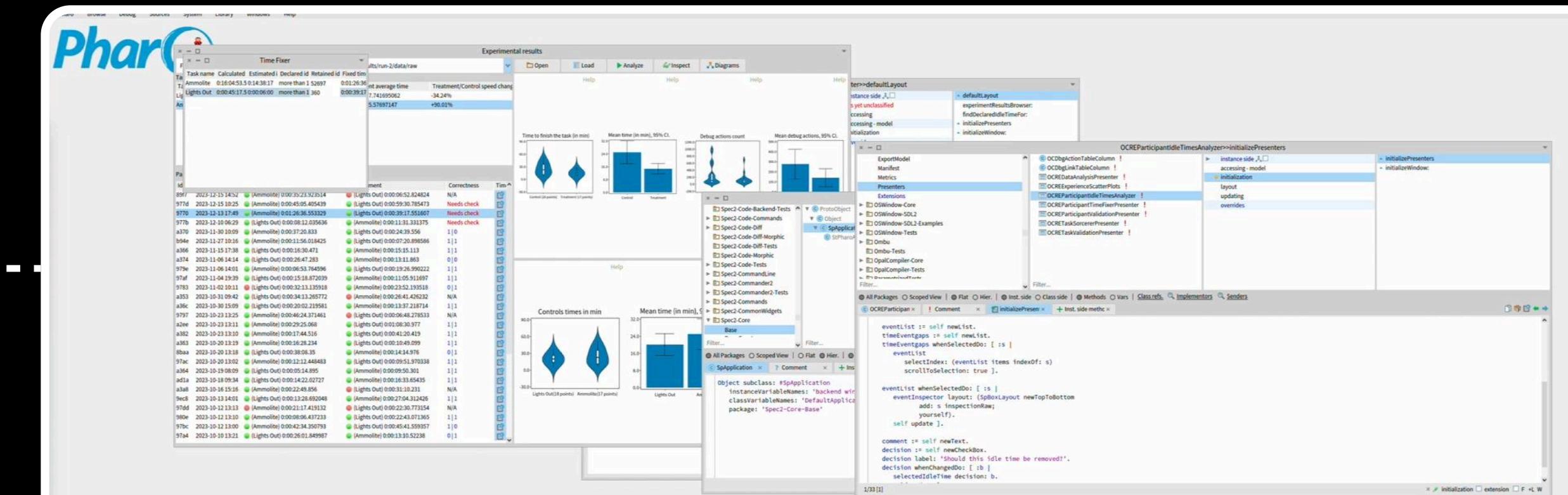
(Themes are at a provisional stage)

# Methods

1

## Observed the Videos

Familiarizing myself with the Data



2

## Generated Transcriptions

Fixing mistakes along the way

Tasks

3

## Created the Codes

Double checking with the research team

4

## Decide what to Code

Using Thematic analysis tools (Taguette)

5

## Grouping Codes into Themes

(Themes are at a provisional stage)

# Methods

1

## Observed the Videos

Familiarizing myself with the Data

2

## Generated Transcriptions

Fixing mistakes along the way

3

## Created the Codes

Double checking with the research team

4

## Decide what to Code

Using Thematic analysis tools (Tagette)

5

## Grouping Codes into Themes

*(Themes are at a provisional stage)*

[10:05.840 --> 10:07.280] Vamos a ver que es lo que está haciendo.  
[10:07.440 --> 10:08.740] Y porque no está stageando.  
[10:10.960 --> 10:11.760] Porque eventualmente.  
[10:11.760 --> 10:13.120] Lo que vamos a hacer es entender el problema.  
[10:13.360 --> 10:14.060] Escribir un test.  
[10:14.780 --> 10:16.140] O ver si podemos escribir un test.  
[10:17.260 --> 10:18.060] Hasta ahora.  
[10:21.960 --> 10:22.360] Perdón.  
[10:22.680 --> 10:24.740] No tengo test del stageando.  
[10:24.860 --> 10:26.480] Lo estaba haciendo de manera.  
[10:27.760 --> 10:28.160] Orgánica.  
[10:29.540 --> 10:30.120] Pero bueno.  
[10:30.140 --> 10:31.100] Como lo estoy haciendo por un video.  
[10:31.160 --> 10:32.300] Ahora me estoy obligado a hacer test.  
[10:33.800 --> 10:34.240] Bueno.  
[10:34.400 --> 10:36.760] Nos frenamos en el R29.  
[10:40.160 --> 10:40.720] Y ahora.  
[10:40.940 --> 10:41.980] Esto debería decir.  
[10:42.120 --> 10:43.680] El R29 debería decir true.  
[10:44.420 --> 10:45.620] Y de hecho lo dice.  
[10:47.120 --> 10:48.380] Vamos a hacer una cosa.

# Methods

## 1 Observed the Videos

Familiarizing myself with the Data

## 2 Generated Transcriptions

Fixing mistakes along the way

## 3 Created the Codes

Double checking with the research team

## 4 Decide what to Code

Using Thematic analysis tools (Tagette)

## 5 Grouping Codes into Themes

*(Themes are at a provisional stage)*

# Main Themes

## Understanding Information Flow

*(Discoverability, Needs, Doubts, Memory/History)*

## Error Correction & Prevention

*(Error Identification - Accuracy - Doubts - Tasks  
- Memory/History)*

## Efficient Window Management

*(Discoverability - Customization - Layout - External  
References - Useful Functionalities - Efficiency - Saturation)*

## Shortcut Learning & Searching

*(Discoverability - Shortcuts - Shortcut Standardization -  
Error Identification - Hesitation)*

# Main Themes

## Understanding Information Flow

(Discoverability, Needs, Doubts, Memory/History)

*"Here you can see well that you couldn't see before. Because the location of records. It was hiding a little bit. The thing. I knew it because I know the code. But the debugger didn't show it to me."*

**"Navigating Pharo browser codes is extremely complex."**

## Error Correction & Prevention

(Error Identification - Accuracy - Doubts - Tasks  
- Memory/History)

## Efficient Window Management

(Discoverability - Customization - Layout - External  
References - Useful Functionalities - Efficiency - Saturation)

## Shortcut Learning & Searching

(Discoverability - Shortcuts - Shortcut Standardization -  
Error Identification - Hesitation)

# Main Themes

## Understanding Information Flow

(Discoverability, Needs, Doubts, Memory/History)

## Error Correction & Prevention

(Error Identification - Accuracy - Doubts - Tasks  
- Memory/History)

**"I don't have a way to go back to where I was debugging just now.  
Because it forgot. I need something with memory. For when I make a  
mistake. Since I'm going fast I make mistakes."**

## Efficient Window Management

(Discoverability - Customization - Layout - External  
References - Useful Functionalities - Efficiency - Saturation)

## Shortcut Learning & Searching

(Discoverability - Shortcuts - Shortcut Standardization -  
Error Identification - Hesitation)

# Main Themes

## Understanding Information Flow

(Discoverability, Needs, Doubts, Memory/History)

## Error Correction & Prevention

(Error Identification - Accuracy - Doubts - Tasks  
- Memory/History)

## Efficient Window Management

(Discoverability - Customization - Layout - External  
References - Useful Functionalities - Efficiency - Saturation)

*"This window goes to the background. You have to keep looking. There it is. **This is how I set it up this time because I needed to see a lot of things at the same time.**"*

## Shortcut Learning & Searching

(Discoverability - Shortcuts - Shortcut Standardization -  
Error Identification - Hesitation)

# Main Themes

## Understanding Information Flow

(Discoverability, Needs, Doubts, Memory/History)

## Error Correction & Prevention

(Error Identification - Accuracy - Doubts - Tasks  
- Memory/History)

## Efficient Window Management

(Discoverability - Customization - Layout - External  
References - Useful Functionalities - Efficiency - Saturation)

## Shortcut Learning & Searching

(Discoverability - Shortcuts - Shortcut Standardization -  
Error Identification - Hesitation)

**“Maybe there is a shortcut. No, there's no shortcut for that. Well, but  
this is the instruction we want.”**

# Shortcut Learning

- *Willingness to learn*
- *Uncertainty*

**"I have used these shortcuts since I programmed in Smalltalk. Especially the debug it. But it was before Pharo. So, I don't remember how when I... how I learned that."**

# Shortcut Learning

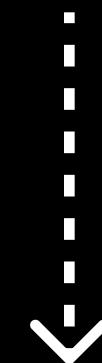
## Background

*"I have used these shortcuts since I programmed in Smalltalk. Especially the debug it. But it was before Pharo. So, I don't remember how when I... how I learned that."*



*"Maybe there is a shortcut. No, there's no shortcut for that. Well, but this is the instruction we want."*

# Background



# Solutions

# Dev Characterization

- Understand UI/Interactions
- 3 archetypes

**1 Specialist**

**2 Explorer**

**3 Stack Organizer**

# Dev Characterization

- Understand UI/Interactions
- 3 archetypes

**1 Specialist**

*(Criteria)*

**Shortcut Use**

**Interface Interaction**

**Layout Customization**

**Tasks and Tools**

**2 Explorer**

**3 Stack Organizer**

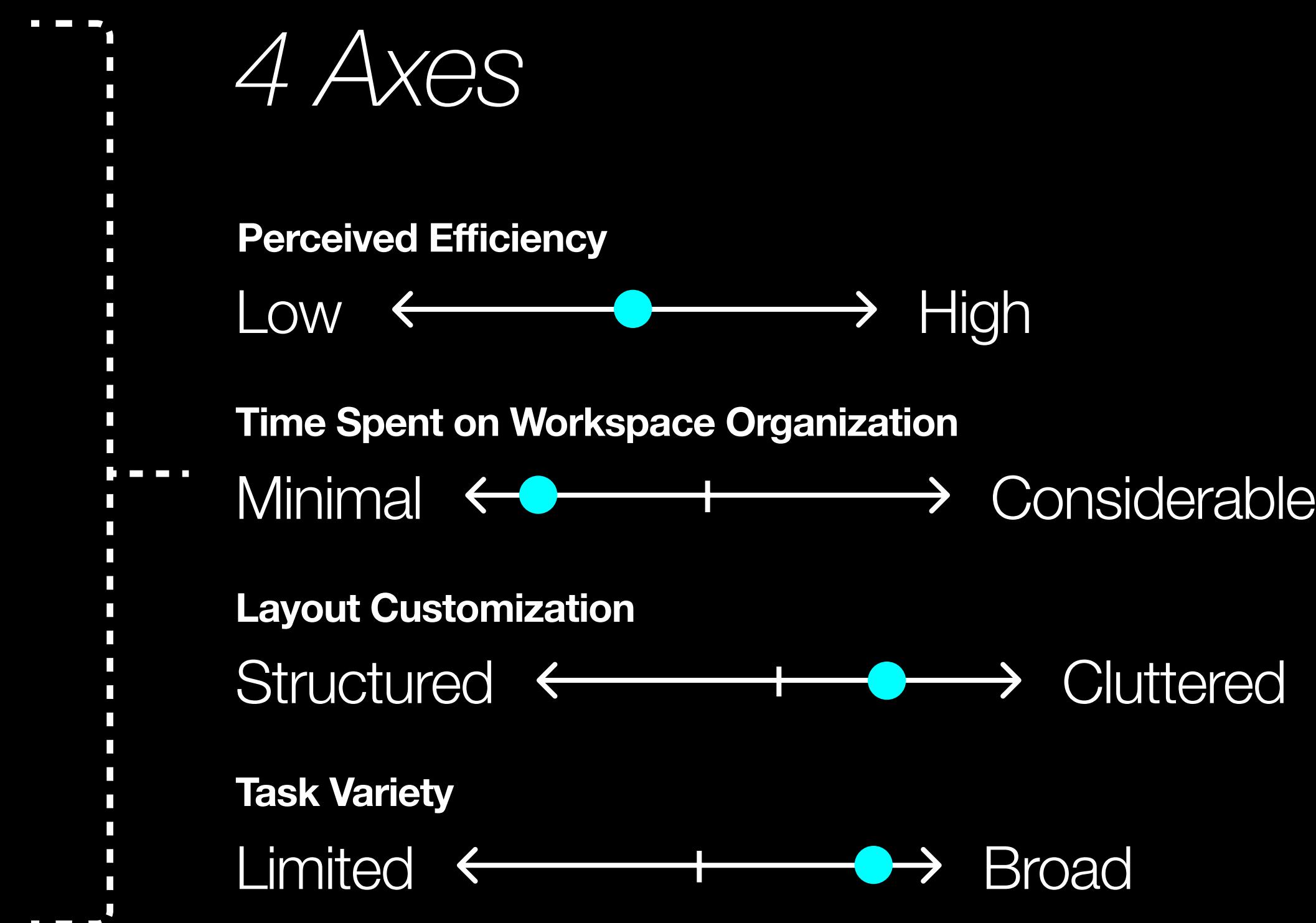
# Dev Characterization

- Understand UI/Interactions
- 3 archetypes

**1 Specialist**

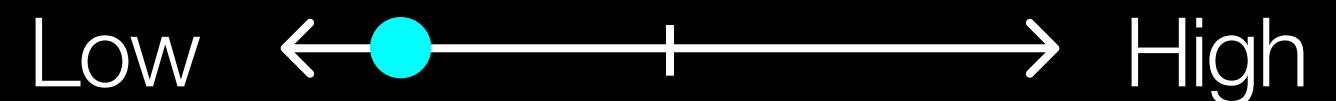
**2 Explorer**

**3 Stack Organizer**



# 1. Specialist

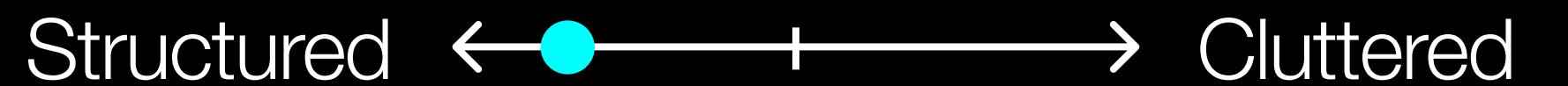
# Perceived Efficiency



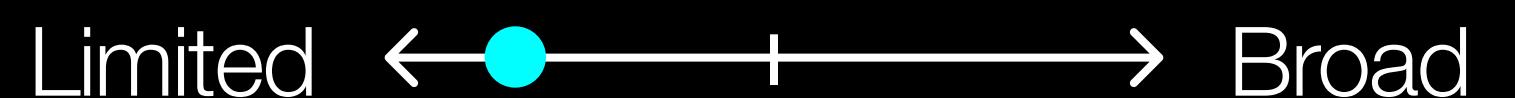
# Time Spent on Workspace Organization



# Layout Customization



# Task Variety



# Interface Interaction:

- Carefully organizes their workspace and avoids having too many windows open in the first place.
  - Prefers to maximize individual windows size to display as much information as possible without overloading the visual space.

# **Shortcut Use:**

- Dives into code manually. Wants to be able to maximize windows with more accuracy and navigate code faster.

*“This is the way I’m organizing it this time because I needed to see a lot of things at the same time.”*

Pharo Browse Debug Sources System Library Windows Help

VMMaker.image

Playground CCodeGenerator>checkClassForNameConflicts:

Dependencies CCodeGenerator Comment checkClassForN Inst. side meths

```
checkClassForNameConflicts: aClass
    "Verify that the given class does not have constant, variable, or method names that conflict with those of previously added classes. Raise an error if a conflict is found, otherwise just return."
    "check for constant name collisions in class pools"
    | tmeth renamedSelector |

    aClass classPool associationsDo: [ :assoc |
        (constants includesKey: assoc key) ifTrue: [
            self error:
                'Constant ', assoc key
                , ' was defined in a previously added class' ] ].

    "and in shared pools"
    (aClass sharedPools reject: [ :pool | pools includes: pool ]) do: [
        :pool |
        pool bindingsDo: [ :assoc |
            (constants includesKey: assoc key) ifTrue: [
                self error:
                    'Constant ', assoc key
                    , ' was defined in a previously added class' ] ] ].

    "check for instance variable name collisions"
    (aClass inheritsFrom: SlangStructType) ifFalse: [
        (self instVarNamesForClass: aClass) do: [ :varName |
            (variables includes: varName) ifTrue: [
                self error:
                    'Instance variable ', varName
                    , ' was defined in a previously added class' ] ] ].

    "check for method name collisions"
    aClass selectors do: [ :sel |

        renamedSelector := aClass renameSelectorIfStaticallyResolved: sel.

        (self isDuplicatedSelector: renamedSelector fromClass: aClass) ifTrue: [
            (aClass >> renamedSelector pragmaAt: #option:)
                ifNil: [
                    self error:
                        'Method ', renamedSelector , ' was defined in a previously added class.' ]
                ifNotNil: [
                    tmeth := methods at: renamedSelector.

                    logger
                        newLine;
                        show: 'warning, method ', aClass name , '>>', renamedSelector storeString
                        , ' overrides ', tmeth definingClass , '>>', renamedSelector storeString;
                        cr ] ] ]
```

48/49 [70]

⚠ Use cascaded nextPutAll's instead of #, in #nextPutAll: ✘ ?

⚠ [renameSelectorIfStaticallyResolved:] Messages sent but not implemented ✘ ?

Stack

| Class  | Method  | Package                  |
|--|---|--------------------------|
| CCodeGeneratorGlobalStructure (CCodeGenerator) | [sel] renamedSelector := aClass renameSelectorIfStaticallyResolved: sel. (self isDuplicatedSelector: Slang) | Collections-Abstract     |
| Array (SequenceableCollection)                 | do:   | Slang                    |
| CCodeGeneratorGlobalStructure (CCodeGenerator) | checkClassForNameConflicts:   | Slang                    |
| CCodeGeneratorGlobalStructure (CCodeGenerator) | addClassSilently:   | Slang                    |
| CCodeGeneratorGlobalStructure (CCodeGenerator) | addClass:   | Slang                    |
| CrossPlatformVMMaker (VMMaker)                 | [sic cg addClass: ic]   | Slang                    |
| OrderedCollection                              | do:   | Collections-Sequenceable |

Error: Method readImageNamed: was defined in a previously added class.

Method

```
27     (variables includes: varName) ifTrue: [
28         self error: 'Instance variable ', varName
29         , ' was defined in a previously added class' ] ].

30
31 "check for method name collisions"
32 aClass selectors do: [ :sel |

33     renamedSelector := aClass renameSelectorIfStaticallyResolved: sel.

34
35     (self isDuplicatedSelector: renamedSelector fromClass: aClass) ifTrue: [
36         (aClass >> renamedSelector pragmaAt: #option:)
37             ifNil: [
38                 self error:
39                     'Method ', renamedSelector , ' was defined in a previously added class.' ]
40             ifNotNil: [
41                 ifNil: [
42                     self error:
43                         'Method ', renamedSelector , ' was defined in a previously added class.' ]
44
45                 tmeth := methods at: renamedSelector.

46
47                 logger
48                     newLine;
49                     show: 'warning, method ', aClass name , '>>', renamedSelector storeString
                     , ' overrides ', tmeth definingClass , '>>', renamedSelector storeString;
                     cr ] ] ]
```

a CCodeGeneratorGlobalStruc...

| Type     | Variable                      | Value   |
|----------|-------------------------------|---|
| implicit | self                          | a CCodeGeneratorGlobalStructure   |
| arg      | aClass                        | SpurImageReader   |
| arg      | sel                           | readImageNamed:   |
| temp.var | tmeth                         | nil   |
| temp.var | renamedSelector               | readImageNamed:   |
| inst.var | ( ) structClasses             | an OrderedCollection [15 items] (SpurScavengeLogRecord SpurSegmentInfo VMMemoryMap VMRememberedS..) |
| inst.var | ( ) translationDict           | a Dictionary [0 items] ()   |
| inst.var | ( ) asArgumentTranslationDict | nil   |
| inst.var | ( ) inlineList                | an Array [0 items] ()   |
| inst.var | ( ) constants                 | a Dictionary [330 items] (size 330)   |
| inst.var | ( ) variables                 | a Set [245 items] (#freeListsMask #remapBuffer #traceSources #profileProcess #maxLiteralCountFor..) |
| inst.var | ( ) variableDeclarations      | a Dictionary [104 items] (size 104)   |
| inst.var | ( ) scopeStack                | an OrderedCollection [0 items] ()   |
| inst.var | ( ) methods                   | a Dictionary [2778 items] (size 2778)   |

# 2. Explorer

## Perceived Efficiency

Low ← → High

## Time Spent on Workspace Organization

Minimal ← → Considerable

## Layout Customization

Structured ← → Cluttered

## Task Variety

Limited ← → Broad

## Interface Interaction:

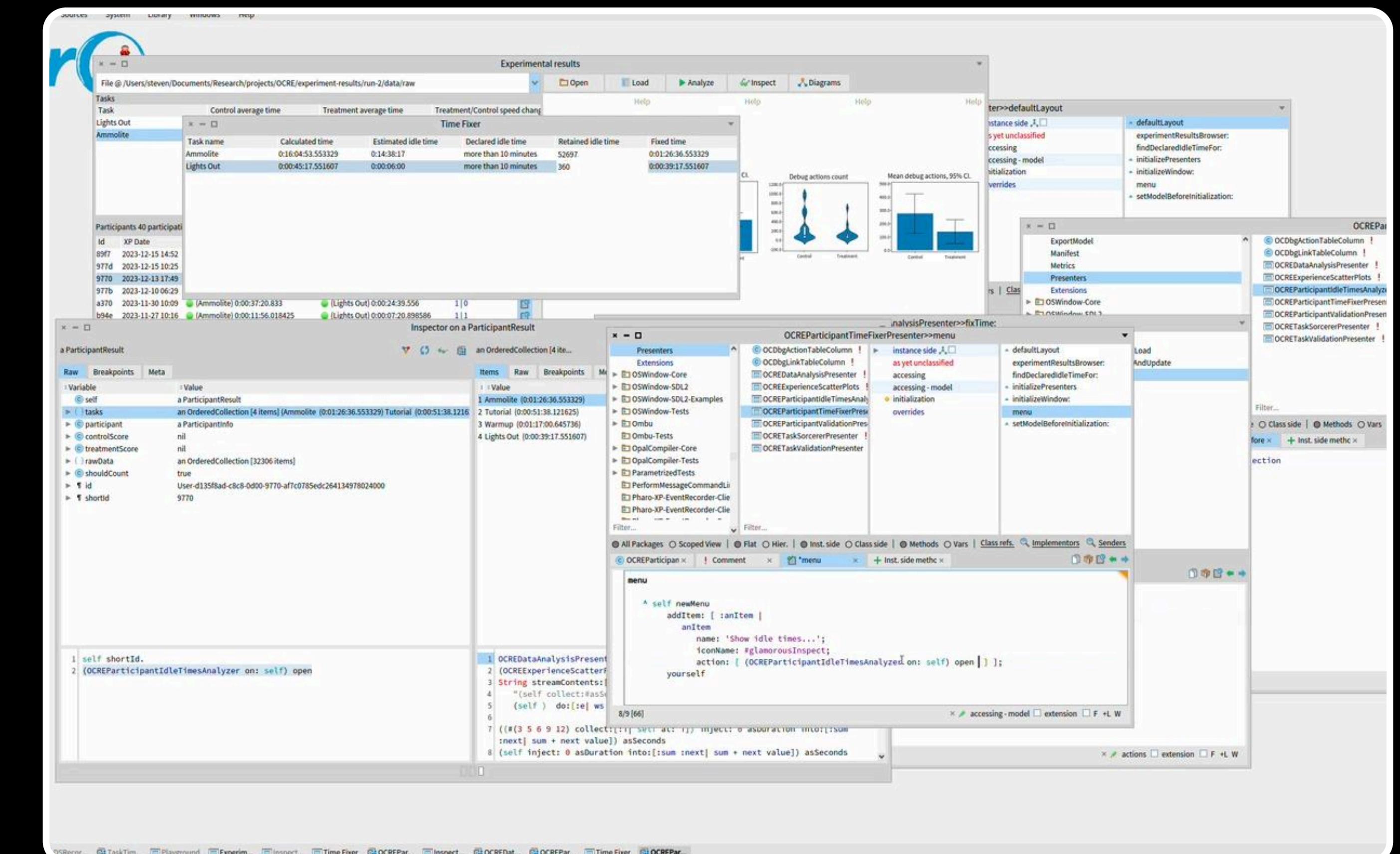
- They open windows as needed and close them once they're no longer useful or when too much information on the screen feels overwhelming.
- Dislikes having to arrange and rearrange windows for task specific layouts, won't do so unless necessary.

## Shortcut Use:

- Uses shortcuts to open and close windows, as well as to copy and paste code. Wishes there were shortcuts to speed up arranging windows.

*"I don't give importance to the code browser. It's going to take 20% of my time. But I prefer to open it. Use it and close it."*

*"So there's too much window. And... I cannot understand what I'm seeing. So I'm closing everything."*



# 3. Stack Organizer

## Perceived Efficiency

Low ← → High

## Time Spent on Workspace Organization

Minimal ← → Considerable

## Layout Customization

Structured ← → Cluttered

## Task Variety

Limited ← → Broad

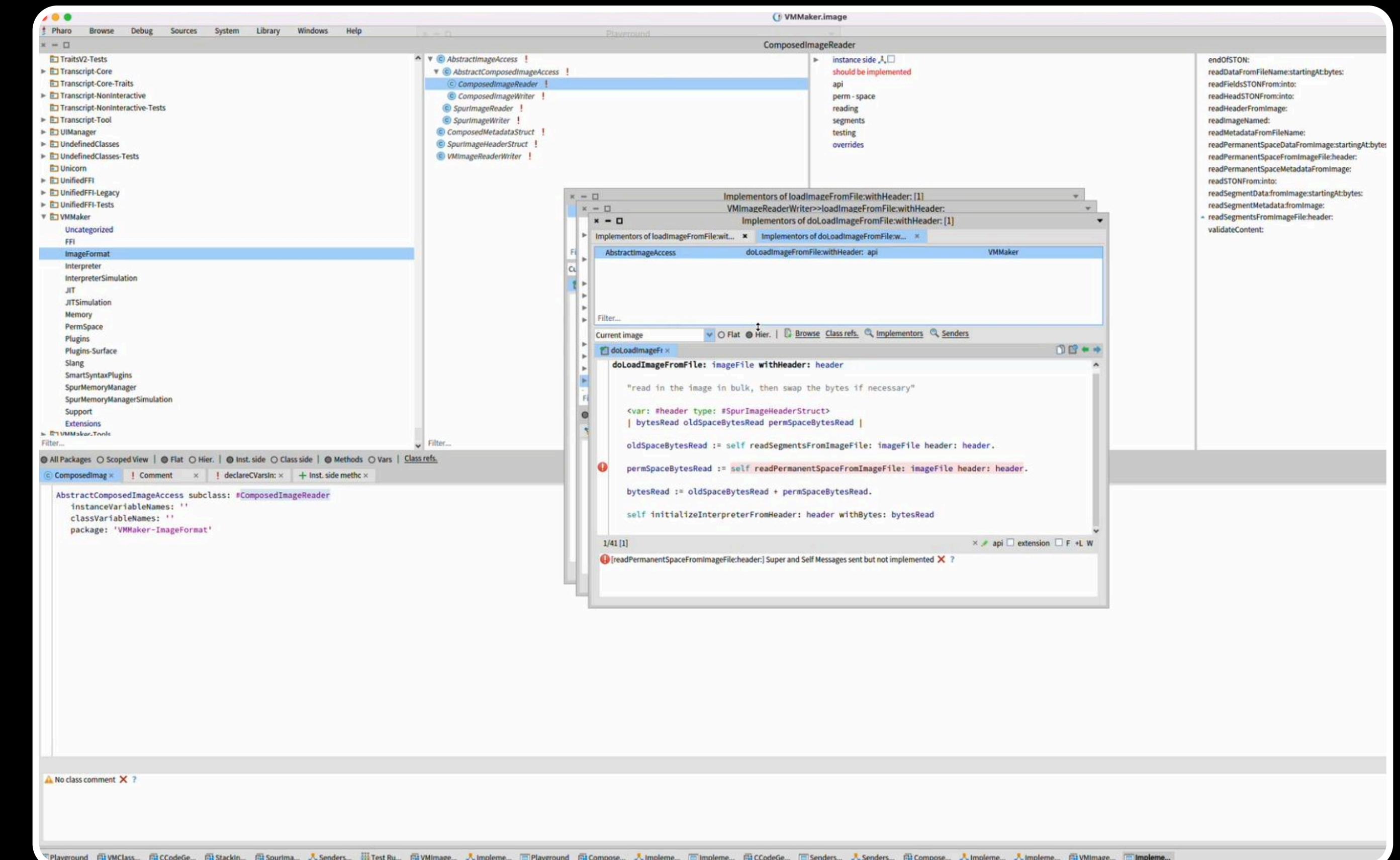
## Interface Interaction:

- Maintains a basic layout with background windows, adding windows according to the flow of each task.
- Sets up stacks of windows related to each task, using a “mind map” to remember their layout and structure.

## Shortcut Use:

- Uses shortcuts to optimize navigation and quick space organization.

*“I use the windows as a stack. Because it's the most comfortable to use with the keyboard, with the current design. That is. I open windows. I close windows. And they open in a stack.”*



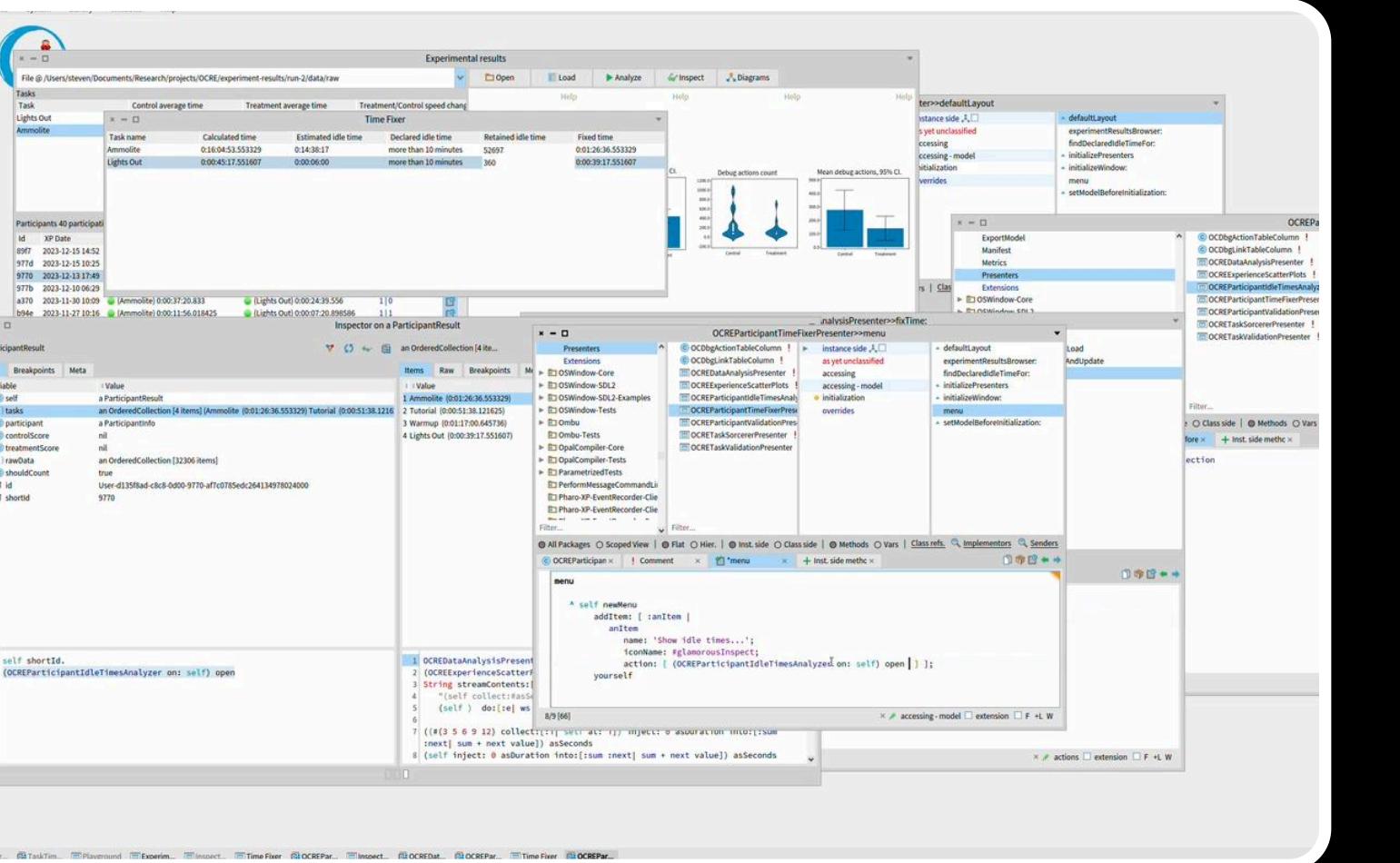
# How does Pharo's UI Impact these Developers?

# How does Pharo's UI Impact these Developers?

Specialist:

This screenshot shows the Pharo Code Generator interface. The main window displays a code editor with a script titled 'CCodeGenerator>>checkClassNameConflicts'. The code is written in Smalltalk and performs various checks for naming conflicts, such as method names and class names. A status bar at the bottom indicates '484012' messages.

Explorer:



Stack Organizer:

This screenshot shows the Pharo Stack Organizer interface. It displays a large tree view of a class hierarchy under 'AbstractImageAccess'. The tree includes numerous subclasses and their implementations, such as 'OCREParticipateTimeAnalyzer', 'OCREParticipantTimeAnalyzer', and 'OCRETaskValidationPresenter'. On the right side, a detailed view of the 'OCRETaskValidationPresenter' class is shown, including its methods, variables, and memory usage statistics.

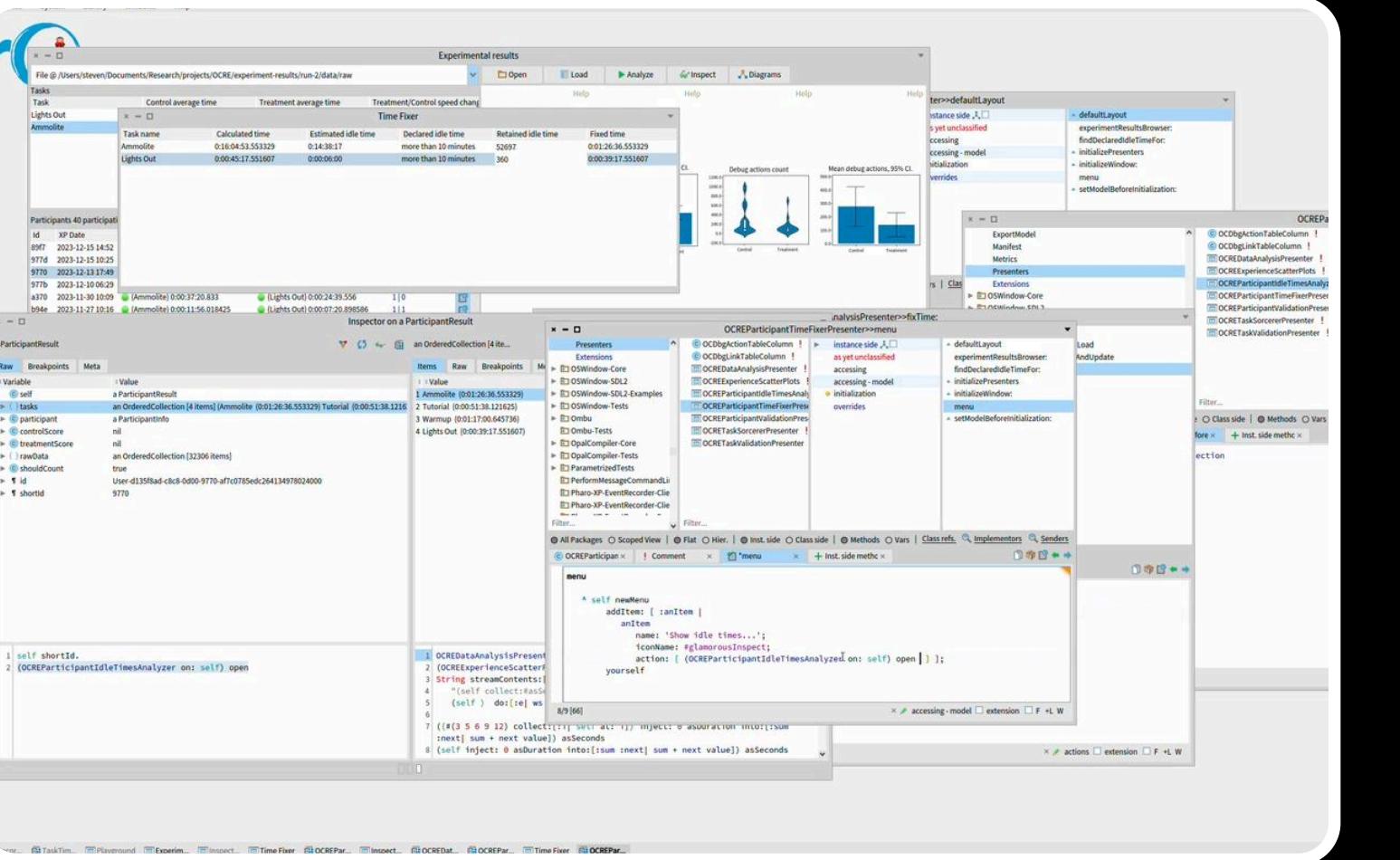
# How does Pharo's UI Impact these Developers?

Specialist:

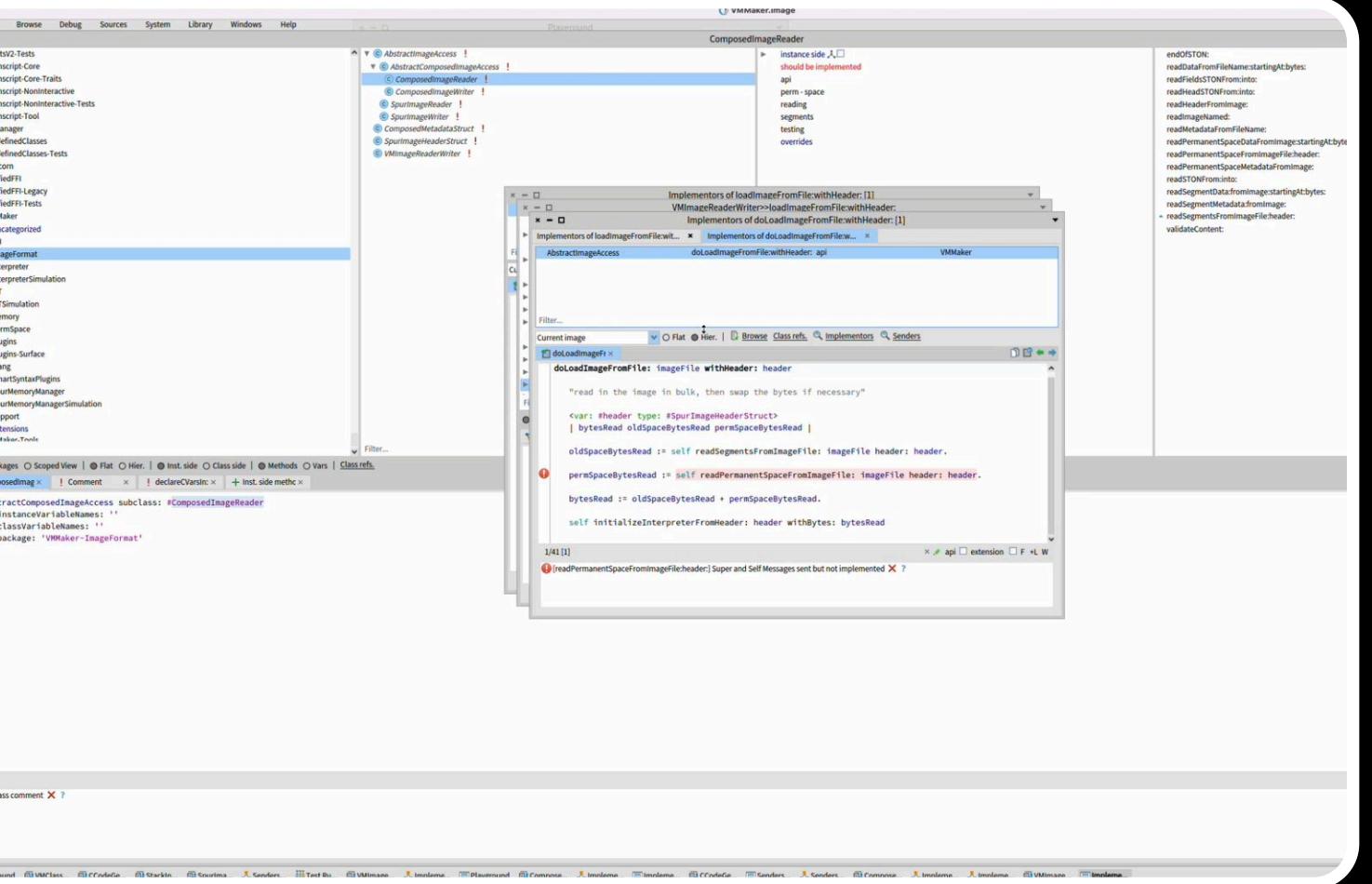
This screenshot shows the Pharo Code Generator interface. It displays a code editor with a snippet of code related to class name conflicts. A floating window titled "checkClassForNameConflicts" provides detailed information about a naming conflict between two classes. The interface includes tabs for Browser, Sources, System, Library, and Help.

```
checkClassForNameConflicts: aClass
    "check for constant name collisions in class pools"
    | tmethod renamedSelector |
    aClass class associationsDo: [ :assoc |
        (constants includesKey: assoc key) ifTrue: [
            'Constant', ' assoc key
            , ' was defined in a previously added class' ]].
    "check for instance variable name collisions"
    aClass class instanceVariablesDo: [ :ivarName |
        (self testVariableForClass: aClass) do: [ :varName |
            (variables includesKey: varName) ifTrue: [
                'Instance variable ', varName
                , ' was defined in a previously added class' ]].
    "check for method name collisions"
    aClass selectors do: [ :sel |
        renamedSelector := aClass renamedSelectorIfStaticallyResolved: sel.
        (self isDuplicatedSelector: renamedSelector fromClass: aClass) ifTrue: [
            self error:
            'Method ', renamedSelector , ' was defined in a previously added class.' ]
        ifNotNil: [
            self error:
            'Method ', renamedSelector , ' was defined in a previously added class.' ]].
    "check for class selectors do: [ :sel |
        renamedSelector := aClass renamedSelectorIfStaticallyResolved: sel.
        (self isDuplicatedSelector: renamedSelector fromClass: aClass) ifTrue: [
            self error:
            'Method ', renamedSelector , ' was defined in a previously added class.' ]
        ifNotNil: [
            self error:
            'Method ', renamedSelector , ' was defined in a previously added class.' ]].
    "check for class pool associations do: [ :pool |
        pool associationsDo: [ :assoc |
            (constants includesKey: assoc key) ifTrue: [
                'Constant', ' assoc key
                , ' was defined in a previously added class' ]].
    "check for class pool associationsDo: [ :pool |
        pool associationsDo: [ :assoc |
            (constants includesKey: assoc key) ifTrue: [
                'Constant', ' assoc key
                , ' was defined in a previously added class' ]].
```

Explorer:



Stack Organizer:



Floating Windows/Tools

# How does Pharo's UI Impact these Developers?

# Specialist:

The screenshot shows the Pharo IDE interface with the following details:

- Top Bar:** Pharo, Browse, Debug, Sources, System, Library, Windows, Help.
- Playground Tab:** VMMaker.image
- Code Editor:** CCodeGenerator>>checkClassNameConflicts: aClass  
"Verify that the given class does not have constant, variable, or method names that conflict with those previously added classes. Raise an error if a conflict is found, otherwise just return."  
"check for constant name collisions in class pools"  
| tmeth renamedSelector |  
  
aClass classPool associationsDo: [:assoc |  
(constants includesKey: assoc key) ifTrue: [  
 self error:  
 'Constant ', assoc key  
 , ' was defined in a previously added class' ].  
  
"and in shared pools"  
(aClass sharedPools reject: [:pool | pools includes: pool]) do: [:pool |  
 pool bindingsDo: [:assoc |  
(constants includesKey: assoc key) ifTrue: [  
 self error:  
 'Constant ', assoc key  
 , ' was defined in a previously added class' ].  
  
"check for instance variable name collisions"  
(aClass inheritsFrom: SlangStructType) ifFalse: [  
 (self instanceVariablesForClass: aClass) do: [:varName |  
 (variables includes: varName) ifTrue: [  
 self error:  
 'Instance variable ', varName  
 , ' was defined in a previously added class' ].  
  
"check for method name collisions"  
aClass selectors do: [:sel |  
  
 renamedSelector := aClass renameSelectorIfStaticallyResolved: sel.  
  
(self isDuplicatedSelector: renamedSelector fromClass: aClass) ifTrue: [  
 (aClass >> renamedSelector pragmaAt: #option)  
 ifNil: [  
 self error:  
 'Method ', renamedSelector , ' was defined in a previously added class.']  
 ifNotNil: [  
  
 tmeth := methods at: renamedSelector.  
  
 logger  
 newline;  
 show: 'warning, method ', aClass name, '>>', renamedSelector storeString  
 , ' overrides ', tmeth definingClass, '>>', renamedSelector storeString;  
 cr ] ] ]

**Debugger Stack Trace:**

| Class  | Method  | Package                  |
|--|---|--------------------------|
| CCodeGeneratorGlobalStructure (CCodeGenerator) | [sel] renamedSelector := aClass renameSelectorIfStaticallyResolved: sel. (self isDuplicatedSelector: Slang) | Collections-Abstract     |
| Array (SequenceableCollection)                 | do:   | Slang                    |
| CCodeGeneratorGlobalStructure (CCodeGenerator) | checkClassNameConflicts:  | Slang                    |
| CCodeGeneratorGlobalStructure (CCodeGenerator) | addClassSiliency:   | Slang                    |
| CCodeGeneratorGlobalStructure (CCodeGenerator) | addClass:   | Slang                    |
| CrossPlatformVMMaker (VMMaker)                 | [sel] cg addClass: ic   | Slang                    |
| OrderedCollection                              | do:   | Collections-Sequenceable |

**Bottom Panel:** Error: Method readImageNamed: was defined in a previously added class.

48/49 [70]

⚠ Use cascaded nextPutAll: instead of #, in nextPutAll: ✘ ❌ ?  
⚠ [renameSelectorIfStaticallyResolved:] Messages sent but not implemented ✘ ?

# Explorer:

# Stack Organizer:

# Benefits

- Layout Customization allows the developer to maintain a clear structure that suits the task at hand.
  - Windows help the developer compare and check different information simultaneously.

# Drawbacks

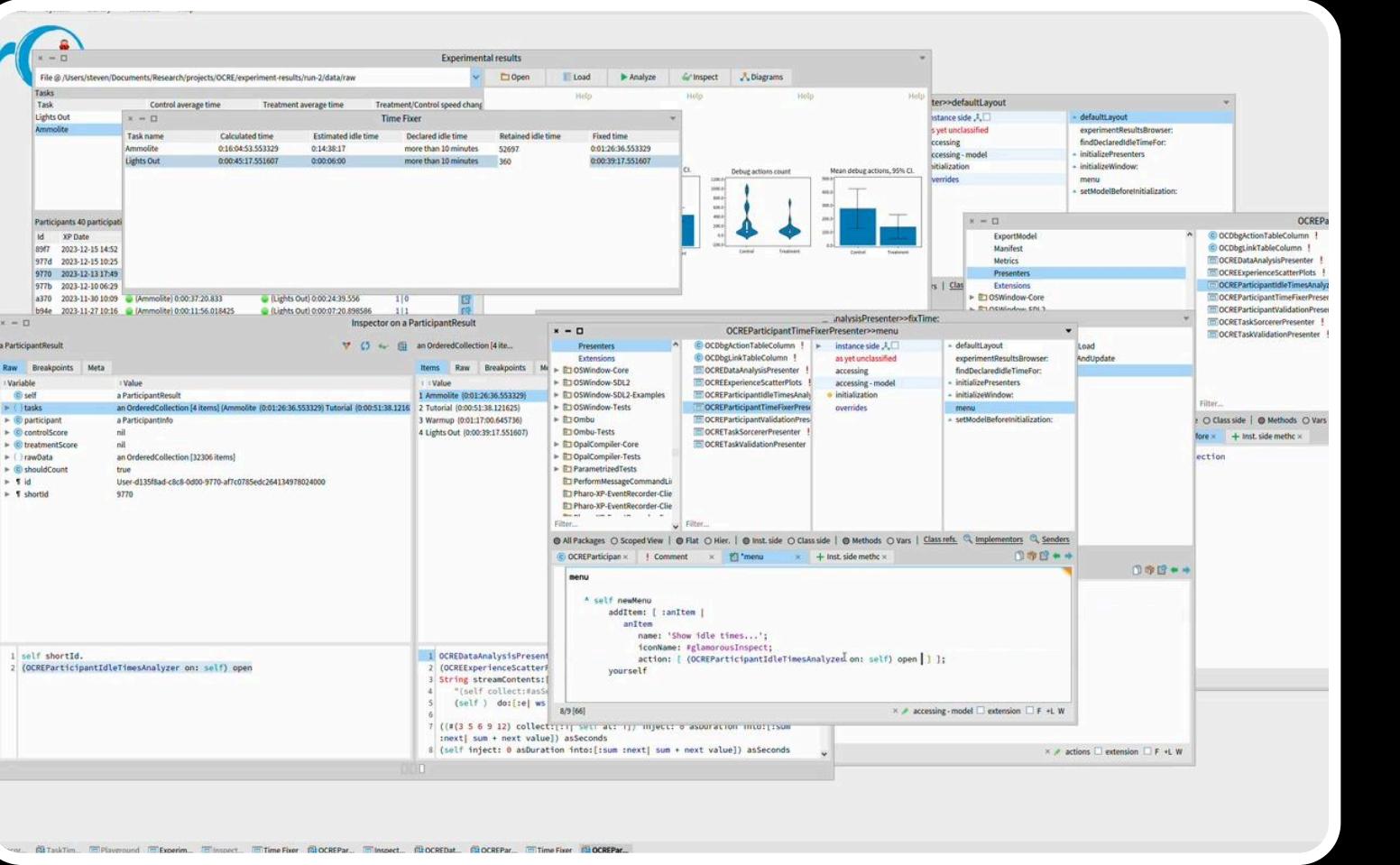
- Spends additional time manually resizing and organizing each window, which can slow down their workflow.
  - Their preference for maximizing windows limits how much information they can see at once.
  - The absence of shortcut options for layout functions makes it harder to work efficiently.

# How does Pharo's UI Impact these Developers?

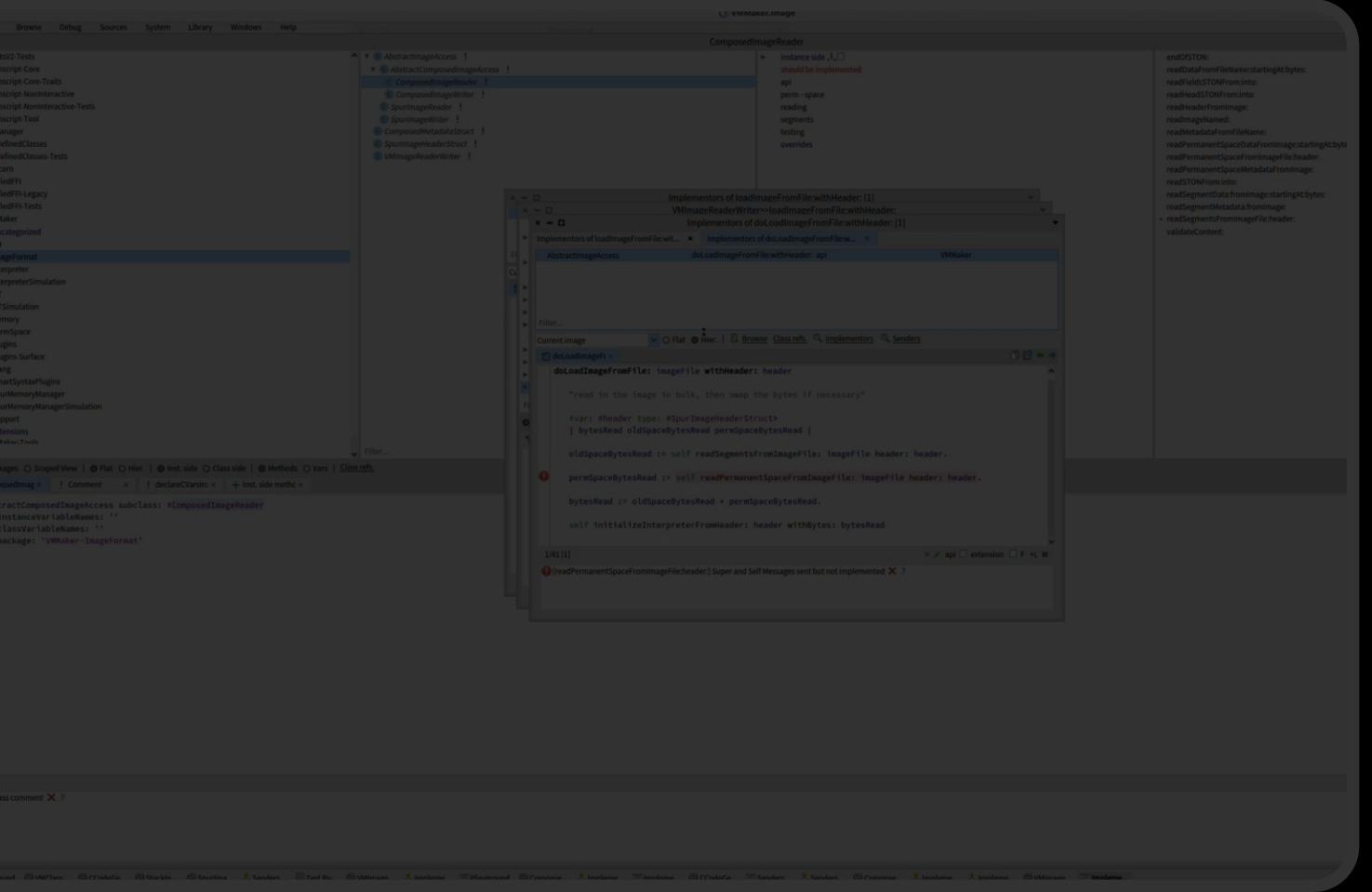
## Specialist:

A screenshot of the Pharo code editor. The main window displays a stack trace for a naming conflict, with multiple frames and error messages. A floating window titled 'Breakpoints' shows a list of breakpoints, including one for 'renamedSelector' at line 41. The interface is dark-themed.

## Explorer:



## Stack Organizer:



## Benefits

- Customizable layout allows the developer to work on complex projects and tasks that need tool diversity.
- Floating windows help users quickly access different tools and layers to complete tasks.

## Drawbacks

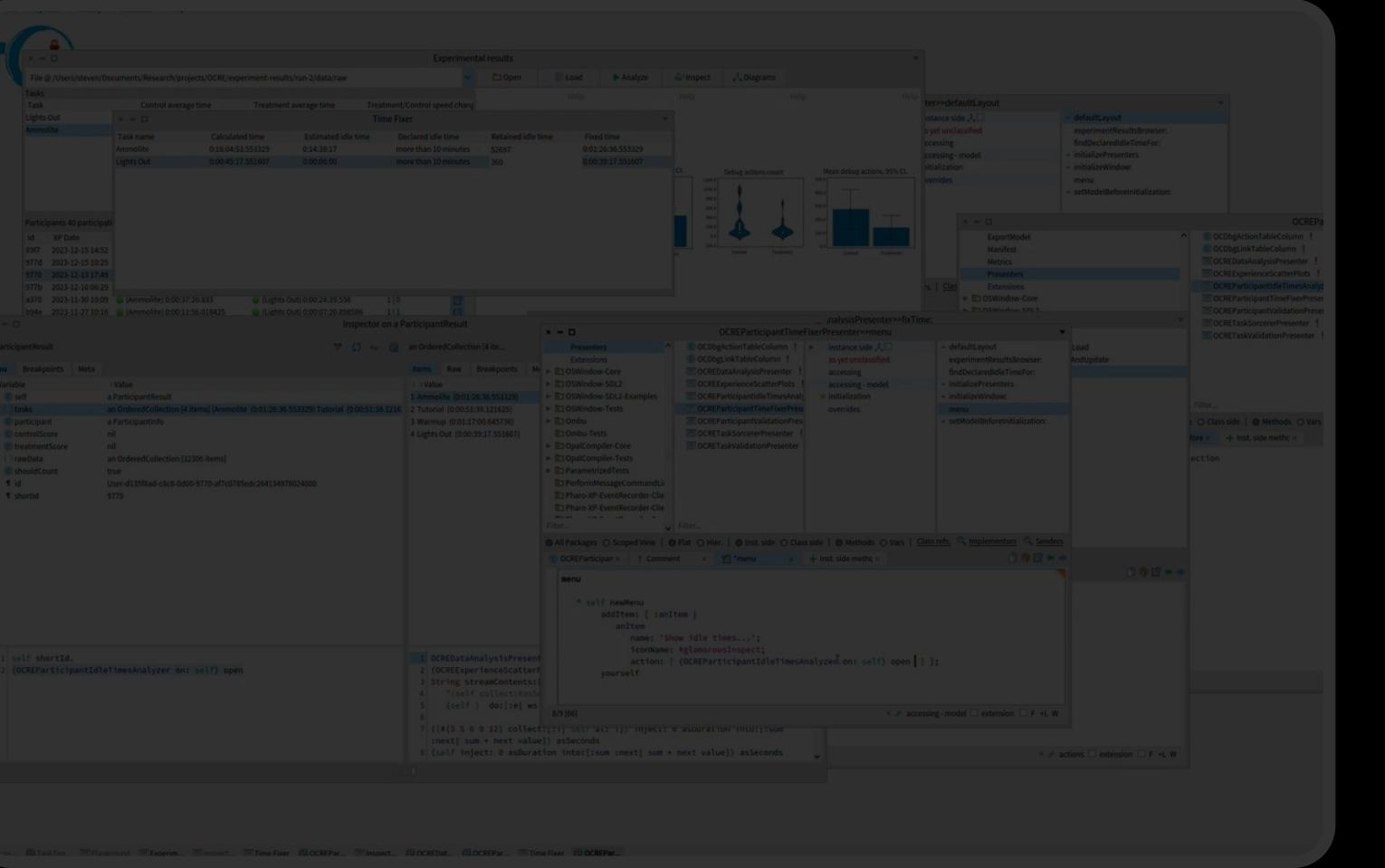
- Once too many windows have been opened, the user's efficiency goes down, forcing them to eventually close everything.
- Inconsistencies in shortcuts disrupt their flow.
- Lack of structure causes developers to lose track of the tasks and tools' layouts.

# How does Pharo's UI Impact these Developers?

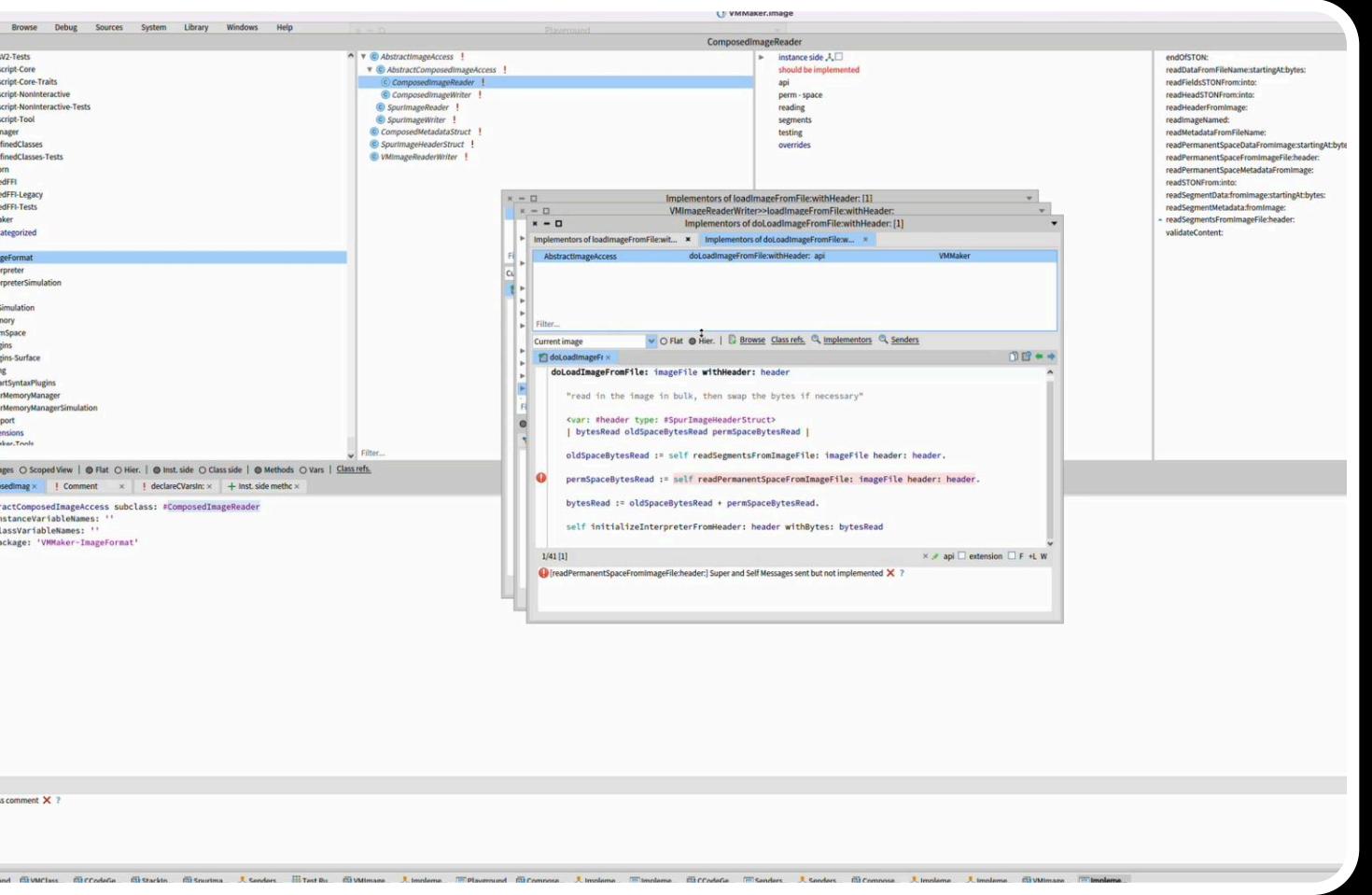
## Specialist:

This screenshot shows the Pharo Code Generation interface. The main window displays a code editor with several error messages related to class name conflicts and renamed selectors. Below the code editor is a stack browser window showing a list of methods and their definitions.

## Explorer:



## Stack Organizer:



## Benefits

- Developers can work on broader tasks without losing their structure.
- Stack system makes navigating more efficient and reduces mouse use.

## Drawbacks

- This layout requires the developer having a mind map/tracking the windows being used.
- Lack of accuracy hinders the developers' workflow.
- Tool layout and window size preferences aren't saved, hindering the developers' speed.

# Specialist:

# Explorer:



# Stack Organizer:

# Solutions

# Benefits

# Drawbacks

# Solutions

## 1 Grouping + Tabs

To avoid Visual Clutter

## 2 Clashing Windows

Faster Window Arrangement

## 3 Customizable Layout Presets

For dedicated tasks

## 4 Window Memory

No need to rearrange window layouts

## 5 Undo Action Shortcut

To correct mistakes

# Solutions

## 1 Grouping + Tabs

To avoid Visual Clutter

## 2 Automatic Window Clashing

Faster Window Arrangement

## 3 Customizable Layout Presets

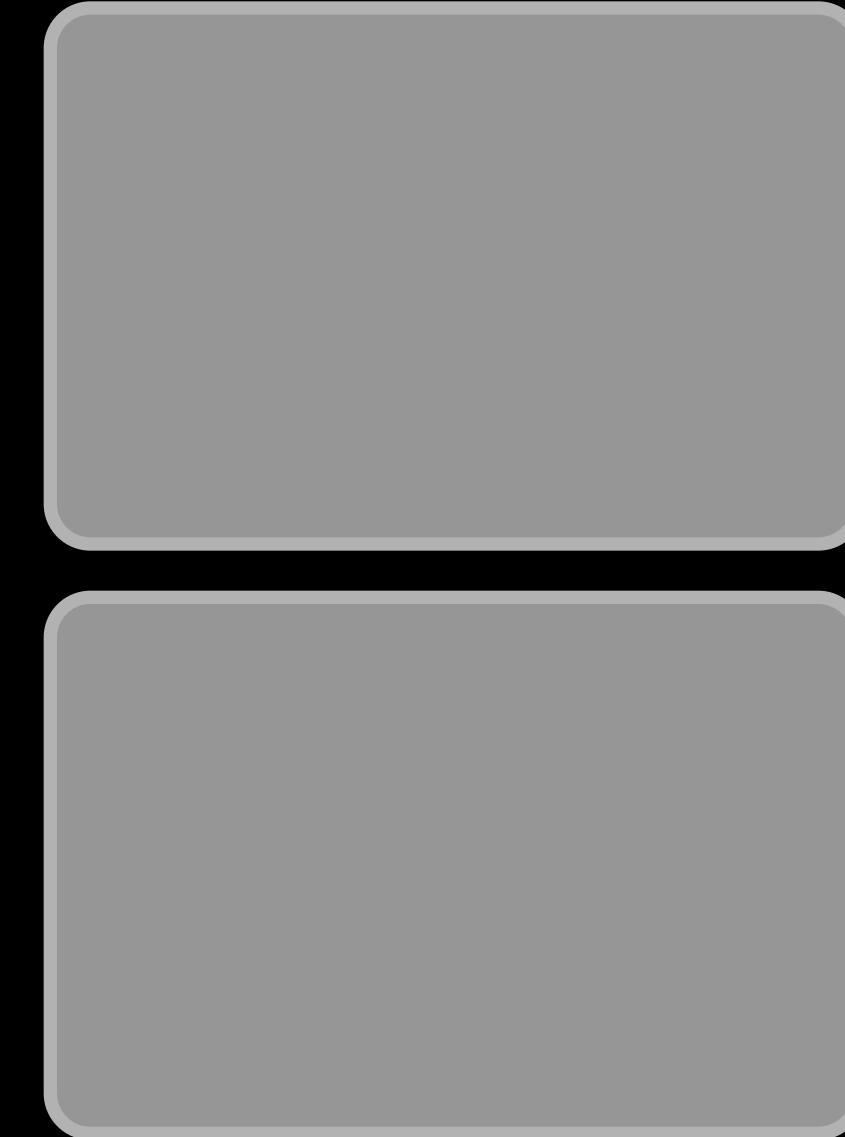
For dedicated tasks

## 4 Window Memory

No need to rearrange window layouts

## 5 Undo Action Shortcut

To correct mistakes



Debuggers

Playgrounds

# Solutions

1

## Grouping + Tabs

To avoid Visual Clutter

2

## Automatic Window Clashing

Faster Window Arrangement

3

## Customizable Layout Presets

For dedicated tasks

4

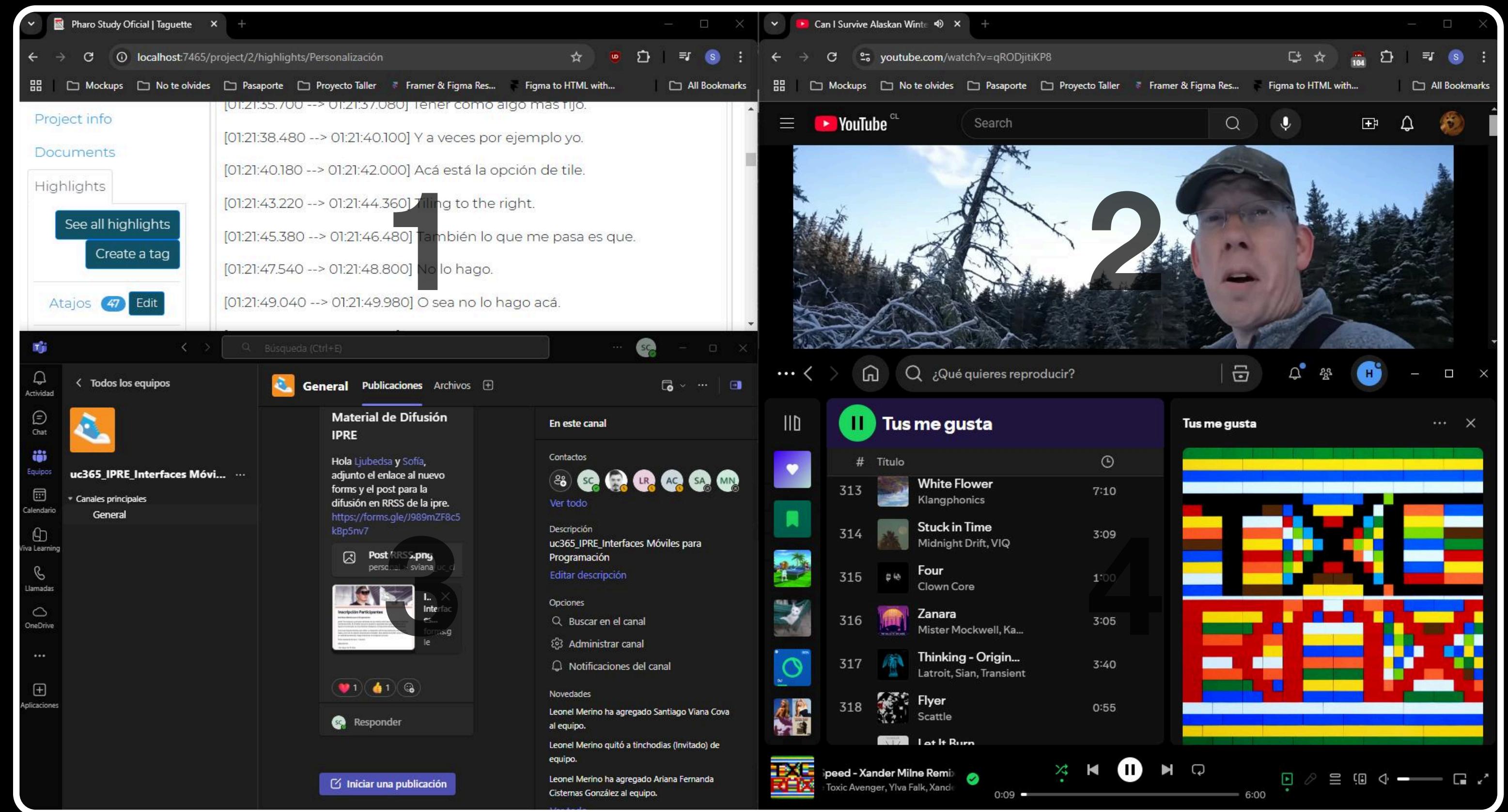
## Window Memory

No need to rearrange window layouts

5

## Undo Action Shortcut

To correct mistakes



# Solutions

1

## Grouping + Tabs

To avoid Visual Clutter

2

## Automatic Window Clashing

Faster Window Arrangement

3

## Customizable Layout Presets

For specific tasks

4

## Window Memory

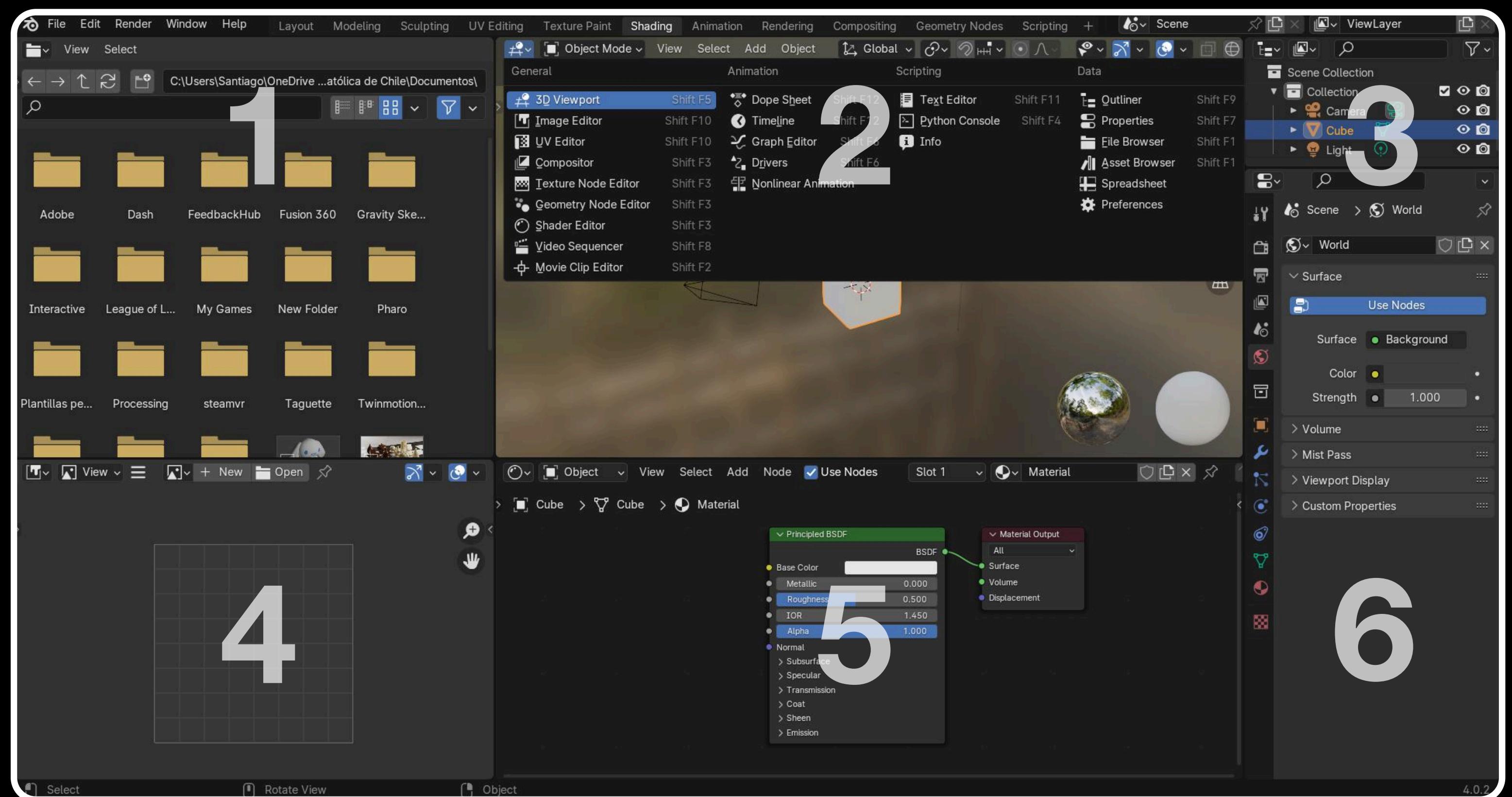
No need to rearrange window layouts

5

## Undo Action Shortcut

To correct mistakes

Presets



# Solutions

1

## Grouping + Tabs

To avoid Visual Clutter

2

## Automatic Window Clashing

Faster Window Arrangement

3

## Customizable Layout Presets

For dedicated tasks

4

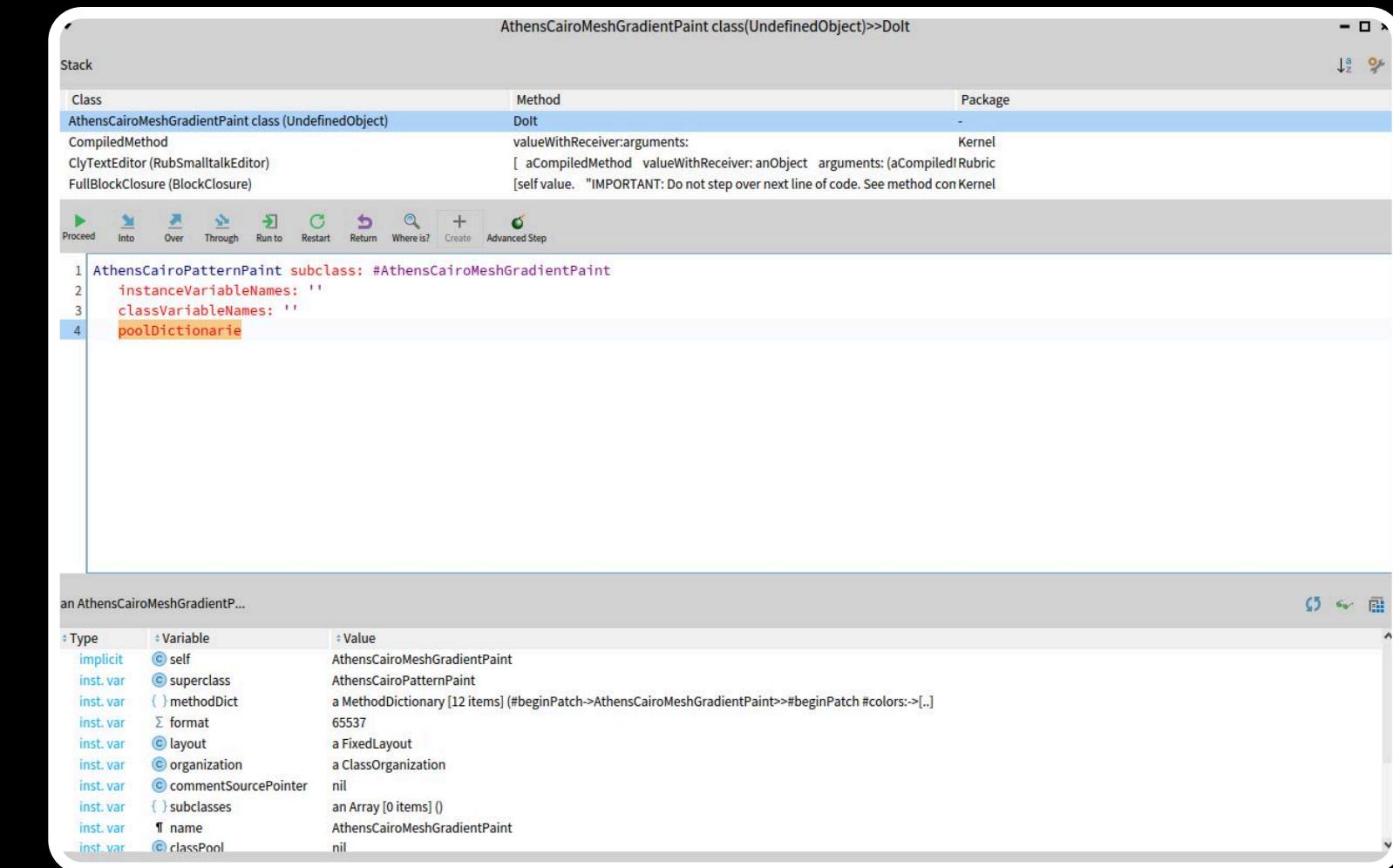
## Window Memory

No need to rearrange window layouts

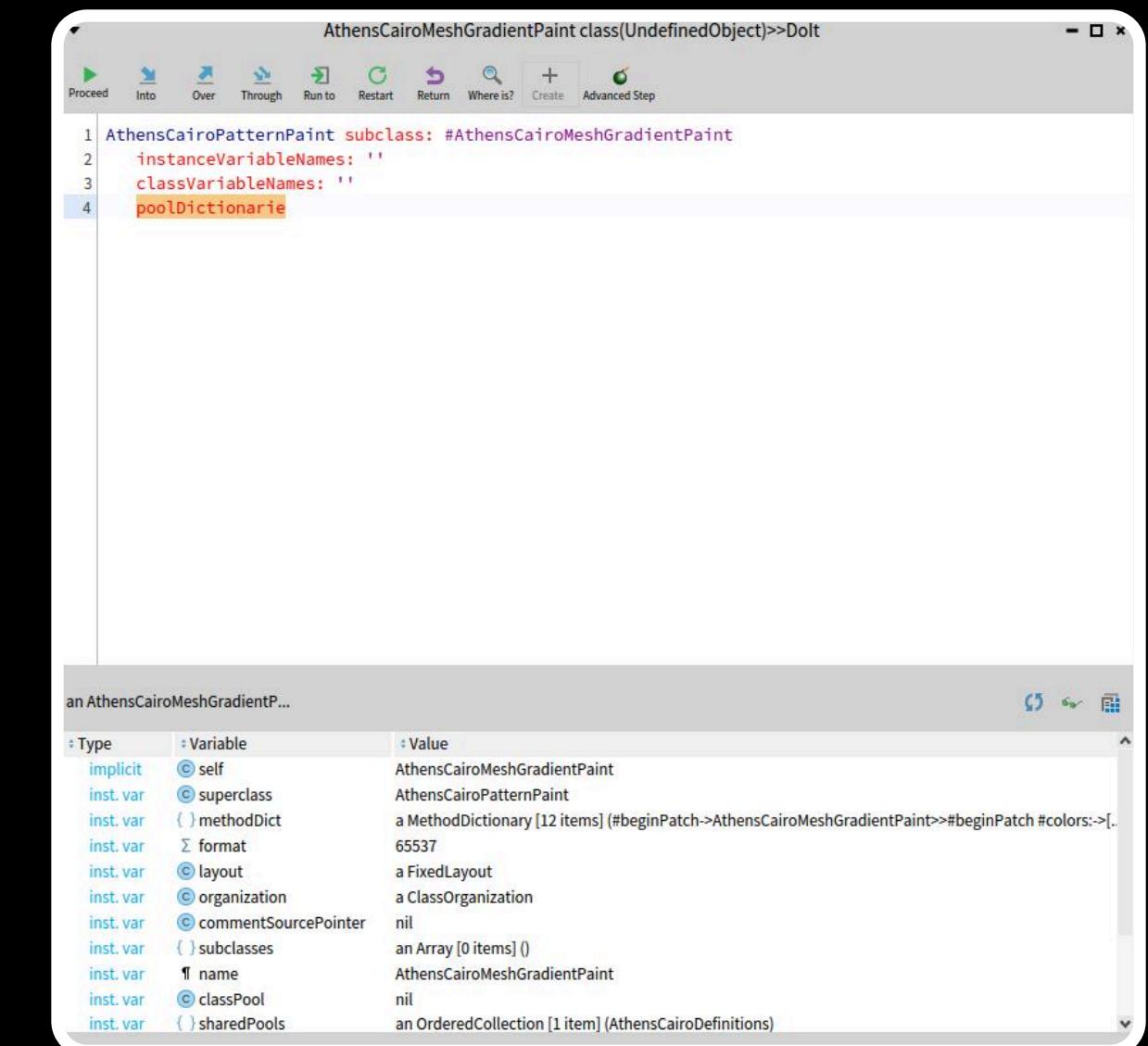
5

## Undo Action Shortcut

To correct mistakes



Default Debugger



Modified Debugger

# Solutions

## 1 Grouping + Tabs

To avoid Visual Clutter

## 2 Automatic Window Clashing

Faster Window Arrangement

## 3 Customizable Layout Presets

For dedicated tasks

## 4 Window Memory

No need to rearrange window layouts

## 5 Undo Action Shortcut

To correct mistakes

