Improving the Usability of LiquidJava

Ricardo Costa[1], Catarina Gamboa[1,2], Alcides Fonseca[1]

[1] LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
[2] S3D, Carnegie Mellon University, Pittsburgh, PA, USA

reliable software systems

What is LiquidJava?

LiquidJava is a type system for Java that uses **liquid types** and **typestates** to verify value properties and object protocols, catching common bugs at compile time.


```java
ArrayList<String> l = new ArrayList<>();
l.add("lasige");
l.get(0);
l.get(1);
```

Its **VS Code extension** automatically performs the verification and reports diagnostics in the editor.

```java
@ExternalRefinementsFor("java.util.ArrayList")
@Ghost("int size")
public interface ArrayListRefinements<E> {
  @StateRefinement(to="size(this) == 0")
  public void ArrayList();

  @StateRefinement(to="size(this) == size(old(this)) + 1")
  public boolean add(E elem);

  public E get(@Refinement("0 <= _ && _ < size(this)") int index);
}
```

It's Great, But...

Unhelpful diagnostics and **limited IDE support** negatively impact developer experience, limiting adoption in mainstream software development.

Why did the verification fail?

What's wrong?

```java
@Refinement("_ > 0")
public int half(@Refinement("_ > 0") int x) {
  return x / 2;
}
```

```
Refinement Error:
Failed to check refinement at:

return x / 2

Type expected:(#ret_1 > 0)
Refinement found:true && #ret_1 == #x_0 / (2)
&& #x_0 == x && (x > 0)
...
```

- **Poor Readability**
- **Limited Context**
- **Plain Text**
- **Redundant Expressions**
- **Internal Variables**
- **No Explanation**

To address the usability barriers of LiquidJava, we adopted a user-centered approach focused on **improving verification feedback** and **IDE support**

IDE Integration

Enhanced the **VS Code extension** with practical features that help developers read, write, and debug refinements in the editor.

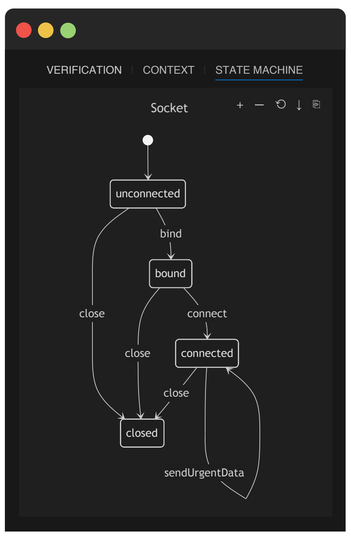
```java
@ExternalRefinementsFor("java.util.ArrayList")
@Ghost("int size")
public interface ArrayListRefinements<E> {
  @StateRefinement(to="s")
  public void ArrayList(
    size(ArrayList)  ghost

  @StateRefinement(to="size(this) == size(old(this)) + 1")
  public boolean add(E elem);

  public E get(@Refinement("0 <= _ && _ < size(this)") int index);
}
```

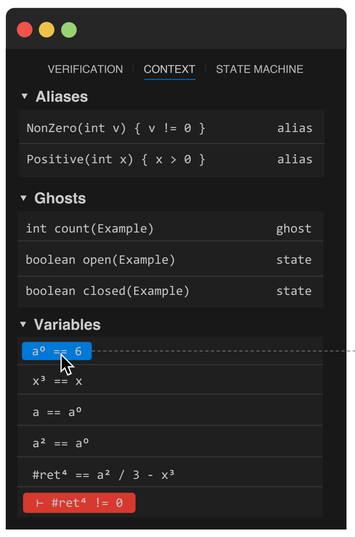
```
int index — ArrayListRefinements.get(int)
@Refinement("0 <= _ && _ < size(this)")
```

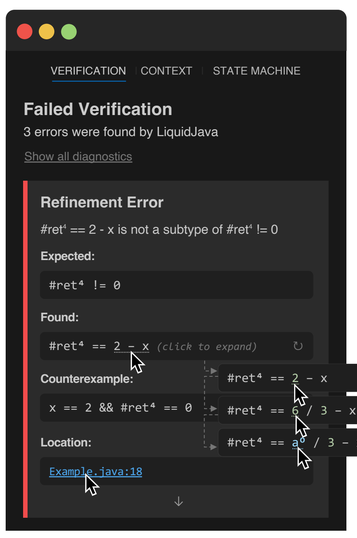
- **Syntax Highlighting**
- **Autocomplete**
- **Hover Information**

Interactive View

Custom VS Code webview where developers can explore diagnostics in more detail, inspect the verification context at the cursor's position, and visualize typestate protocols through state machine diagrams.



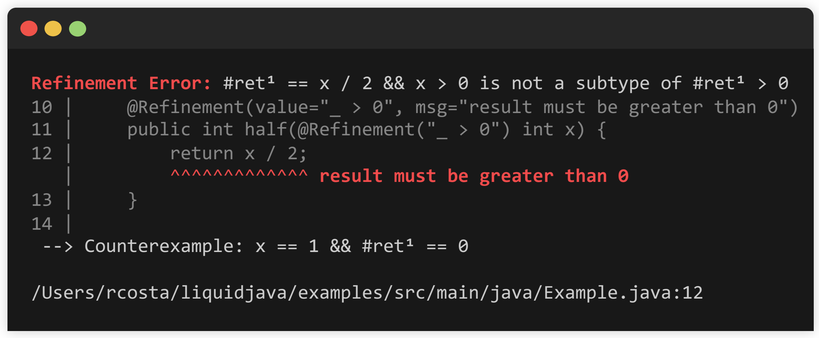




- **Typestate Visualizer**
- **Context Debugger**
- **Diagnostic Explorer**

Readability & Usability of Diagnostics

Improved **diagnostic messages** with clearer structure, better visual presentation, simplified expressions, and actionable feedback.

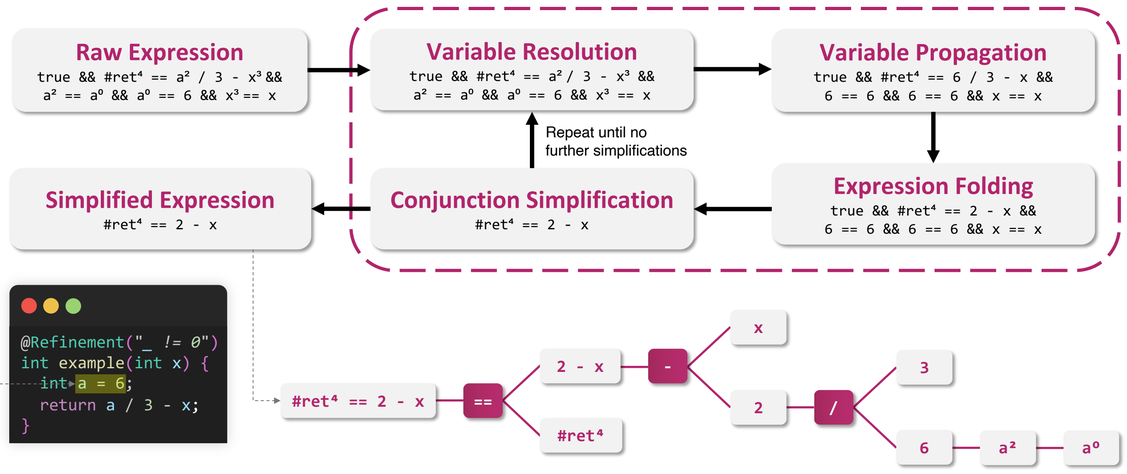


- **x > 0** (refinement)
- **x = 1** (counterexample)
- **1 / 2 = 0** (integer division)
- **0 > 0** (refinement violation)

- **New Design**
- **Counterexample**
- **Custom Message**

Expression Simplification

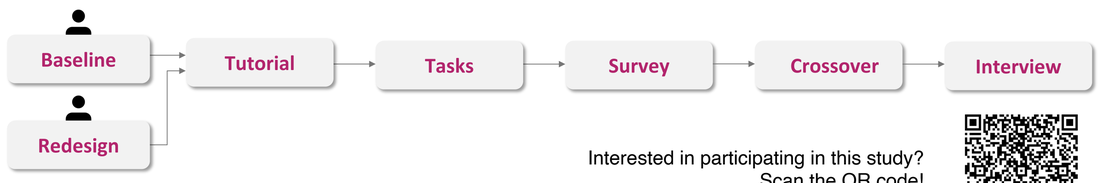
Simplification pipeline that saves each step as a node in a tree data structure, enabling the interactive exploration of simplifications in the diagnostic explorer of the VS Code extension.




```java
@Refinement("_ != 0")
int example(int x) {
  int a = 6;
  return a / 3 - x;
}
```


Planned User Study

We aim to recruit 30 Java developers and split them into **baseline** and **redesign** groups to evaluate the impact of these improvements.



Interested in participating in this study? Scan the QR code!

Research Questions

RQ1. How does improved verification feedback affect developers' ability to **resolve** different types of verification errors?
RQ2. How does improved verification feedback affect developers' **understanding** of verification state and outcomes?
RQ3. What aspects of verification feedback do developers find **most valuable** for diagnosing and resolving errors?

Acknowledgements:
This work was funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the project CMU-Portugal, ref. PRT/BD/154254/2021, and under the LASIGE Research Unit, ref. UID/00408/2025, DOI https://doi.org/10.54499/UID/00408/2025, which includes the LASIGE Seed project ref. LASIGE-SP-2025.07.
This work was also funded by the European Union - NextGenerationEU through the PRR - Recovery and Resilience Plan under the LASIGE Research Unit, ref. UID/PRR/00408/2025, DOI https://doi.org/10.54499/UID/PRR/00408/2025 and ref. UID/PRR2/00408/2025.