Usability-Oriented Design of Liquid Types for Java

Catarina Gamboa

Paulo Canelas

Christopher Timperley

Alcides Fonseca

FCT Fundação para a Ciência e a Tecnologia

Carnegie Mellon Portugal

S3D

U LISBOA

Ciências ULisboa

LASIGE

Software Verification for finding bugs

```
int x = "hello world";
```

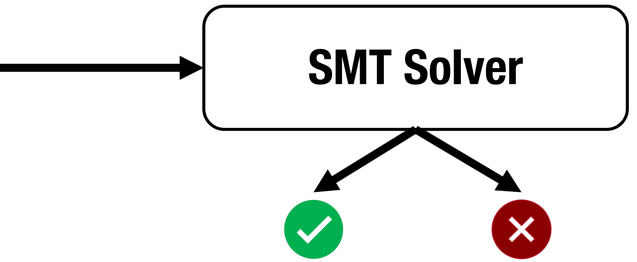
Refinement Types

```
@Refinement("0 <= red && red <= 255")
int red;
red = 200;
red = 200 + 90;
```

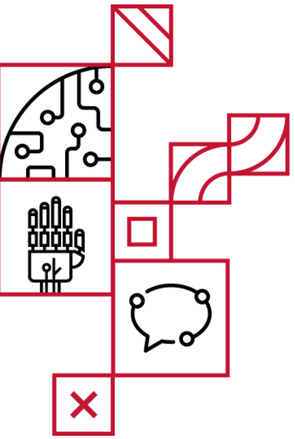
Logical predicate

Liquid Types

```
red == 200 + 90 ->
0 <= red && red <= 255
```



Refinement Types and Liquid Types



ML(1991) **C (2012)** **Haskell (2014)** **Scala (2016)** **Rust (2022)**

Javascript (2012) **Typescript (2016)**

Classes of Errors

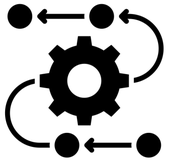
- Division by zero
- Array accesses
- Protocol violations
- Security Issues

Our Contributions

- Design 
- User Study 

Design

Three requirements for the language



Refinements must be optional

`@Refinement`

Refinements should be expressive and idiomatic

`"0 <= red"`

Type-checking should be decidable

QF-UFLA

SMT-Solvers

Syntax Survey with 50 collected answers

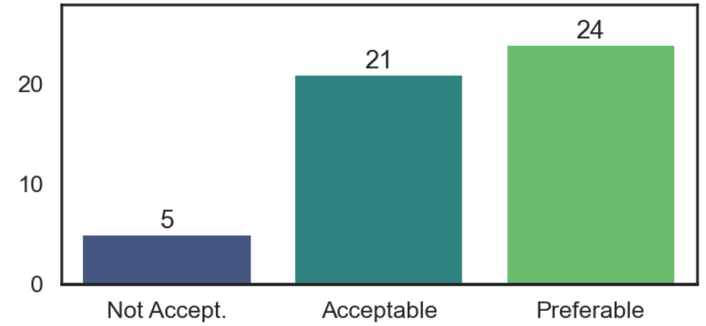
A

```
@Refinement("_ >= 0 && _ <= 100")
public static int percentageFromGrade(@Refinement("grade >= 0") int grade,
                                      @Refinement("scale > 0")  int scale)
```

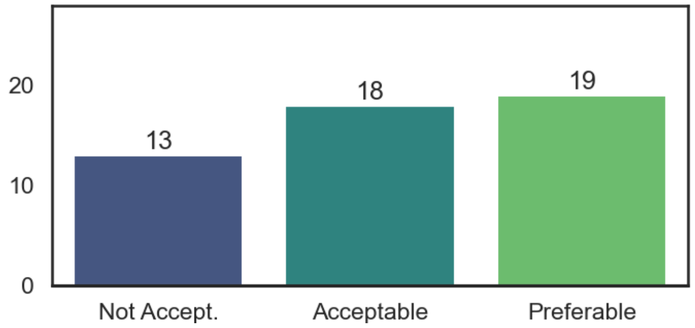
B

```
@Refinement("{grade >= 0} -> {scale > 0} -> {_ >= 0 && scale > 0}")
public static int percentageFromGrade(int grade, int scale)
```

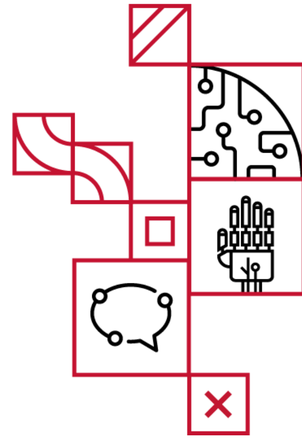
A



B

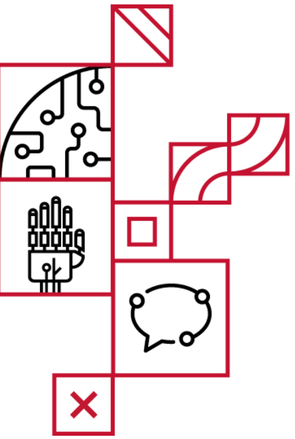


LiquidJava Prototype



```
@Refinement("0 <= red && red <= 255")
int red = 290;
```

```
int inRange(int a, @Refinement("b > a") int b ){…}
inRange(10, 5);
```



LiquidJava Prototype

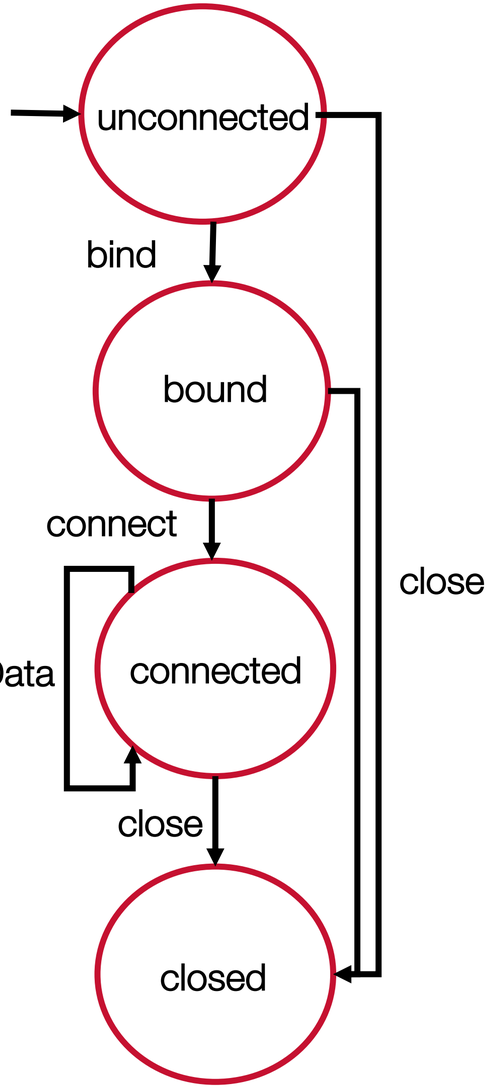
Overview Package Class Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.net

Class Socket



LiquidJava Prototype

Overview Package Class Use Tree Deprecated Index Help

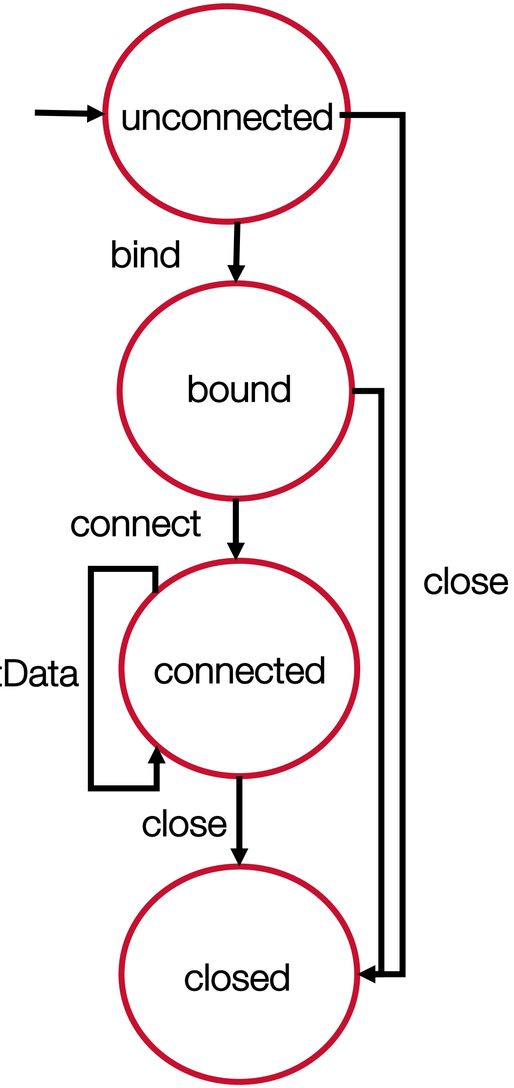
Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

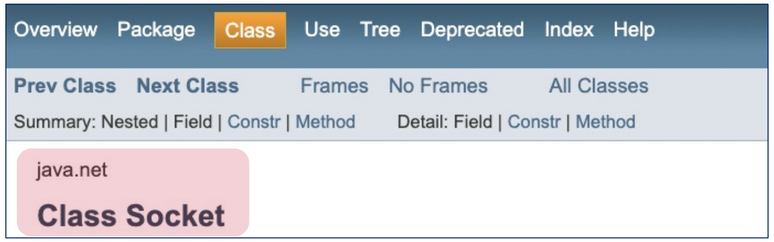
java.net

Class Socket

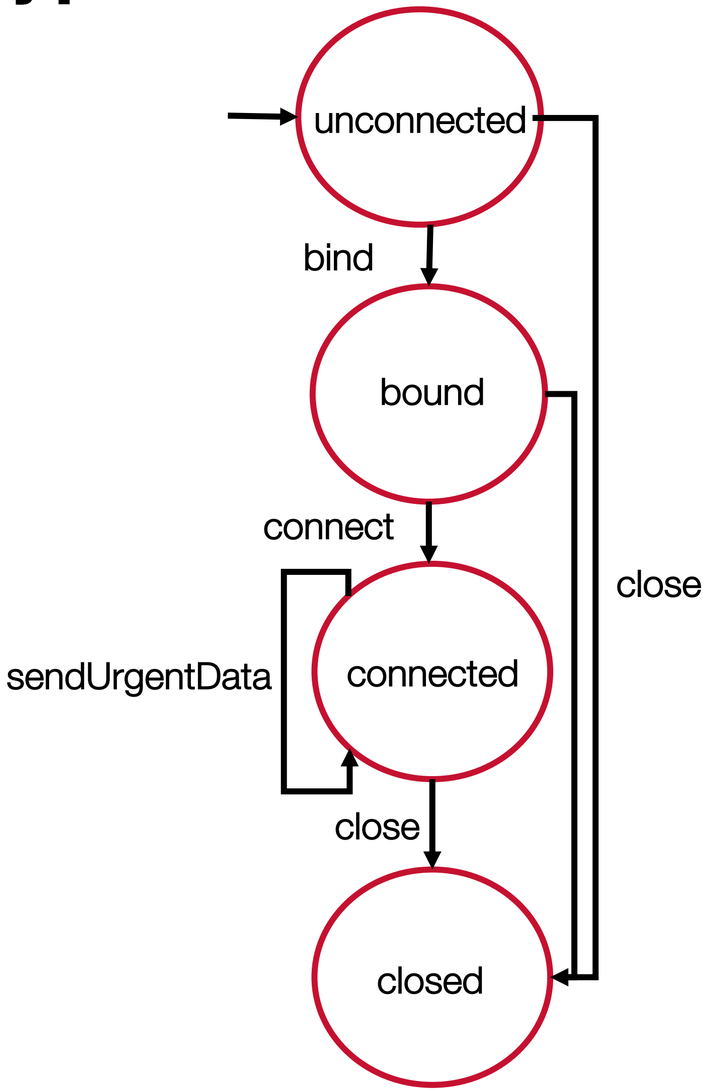
```
@StateSet({"unconnected", "bound", "connected", "closed"})
@ExternalRefinements("java.net.Socket")
interface SocketRefinements{

    @StateRefinement(from = "bound(this)",
                     to = "connected(this)")
    void connect(SocketAddress addr);
    ...
}
```



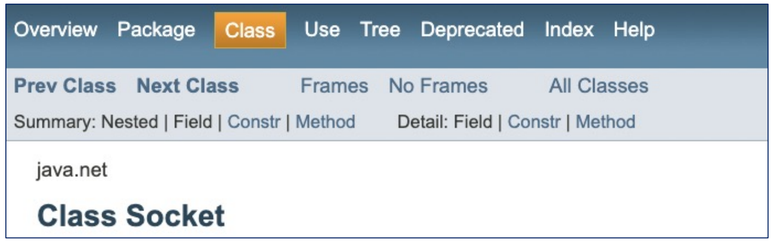
LiquidJava Prototype



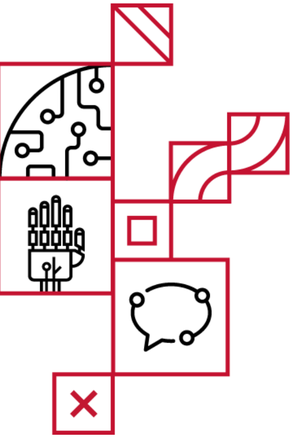

```
@StateSet({"unconnected", "bound", "connected", "closed"})
@ExternalRefinements("java.net.Socket")
interface SocketRefinements{

    @StateRefinement(from = "bound(this)",
                     to = "connected(this)")
    void connect(SocketAddress addr);


    @StateRefinement(from = "connected(this)")
    void sendUrgentData(int n);    ...
}
```

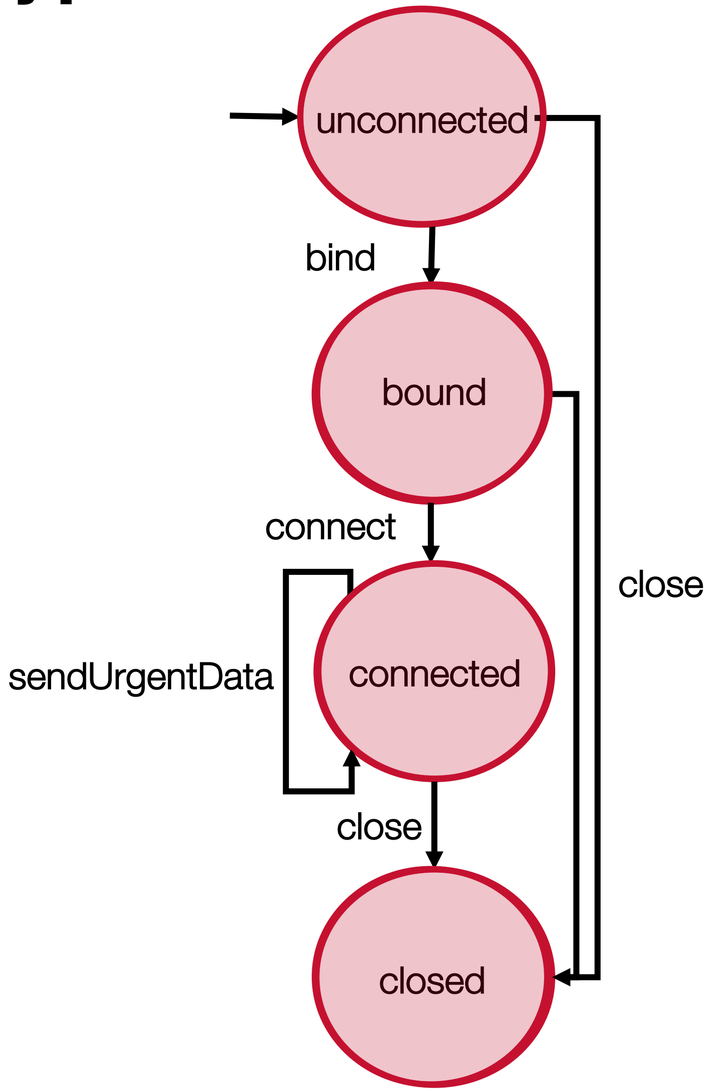


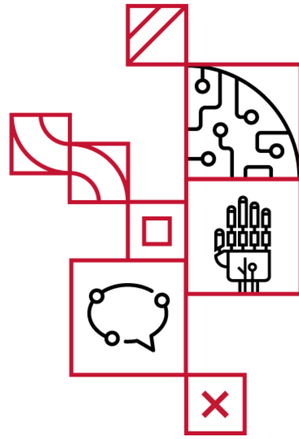
LiquidJava Prototype




```
@StateSet({"unconnected", "bound", "connected", "closed"})
@ExternalRefinements("java.net.Socket")
interface SocketRefinements{

    @StateRefinement(from = "bound(this)",
                     to = "connected(this)")
    void connect(SocketAddress addr);


    @StateRefinement(from = "connected(this)")
    void sendUrgentData(int n);    ...
}
```







LiquidJava Prototype

Overview Package Class Use Tree Deprecated Index Help

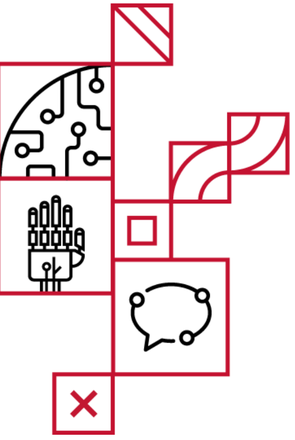
Prev Class Next Class Frames No Frames All Classes

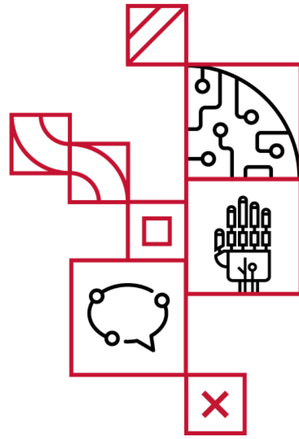
Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

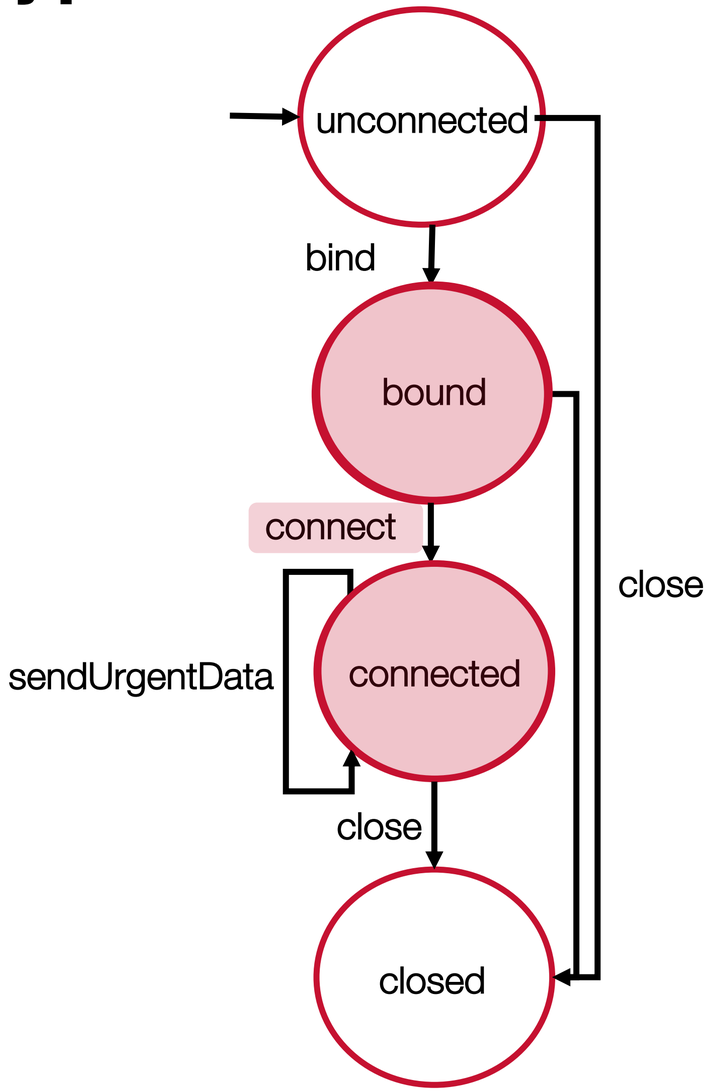
java.net

Class Socket

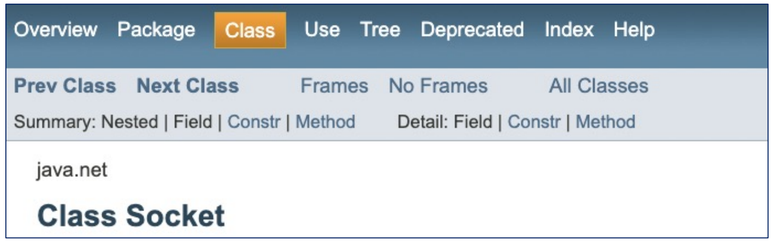
```
@StateSet({"unconnected", "bound", "connected", "closed"})
@ExternalRefinements("java.net.Socket")
interface SocketRefinements{

    @StateRefinement(from = "bound(this)",
                     to = "connected(this)")
    void connect(SocketAddress addr);


    @StateRefinement(from = "connected(this)")
    void sendUrgentData(int n);    ...
}
```



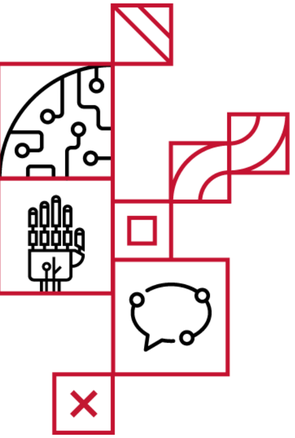


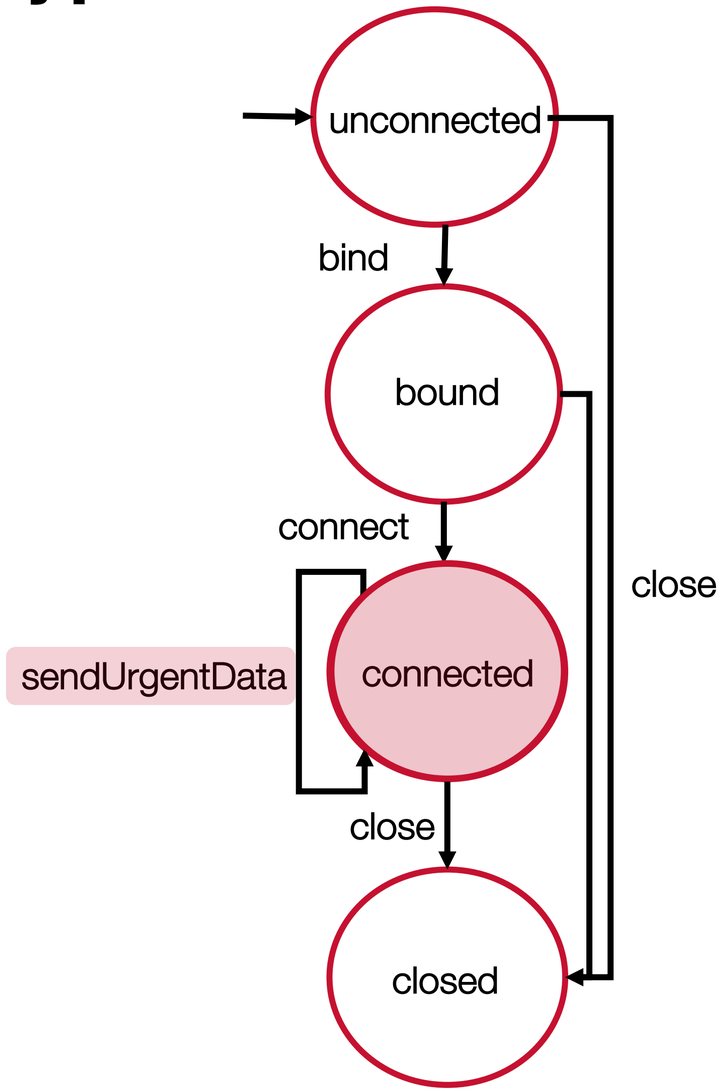


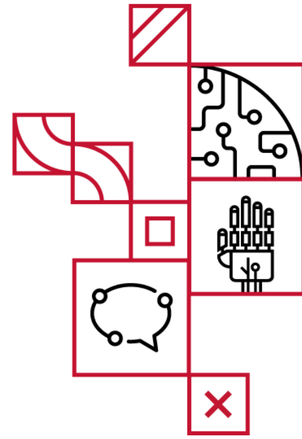
LiquidJava Prototype



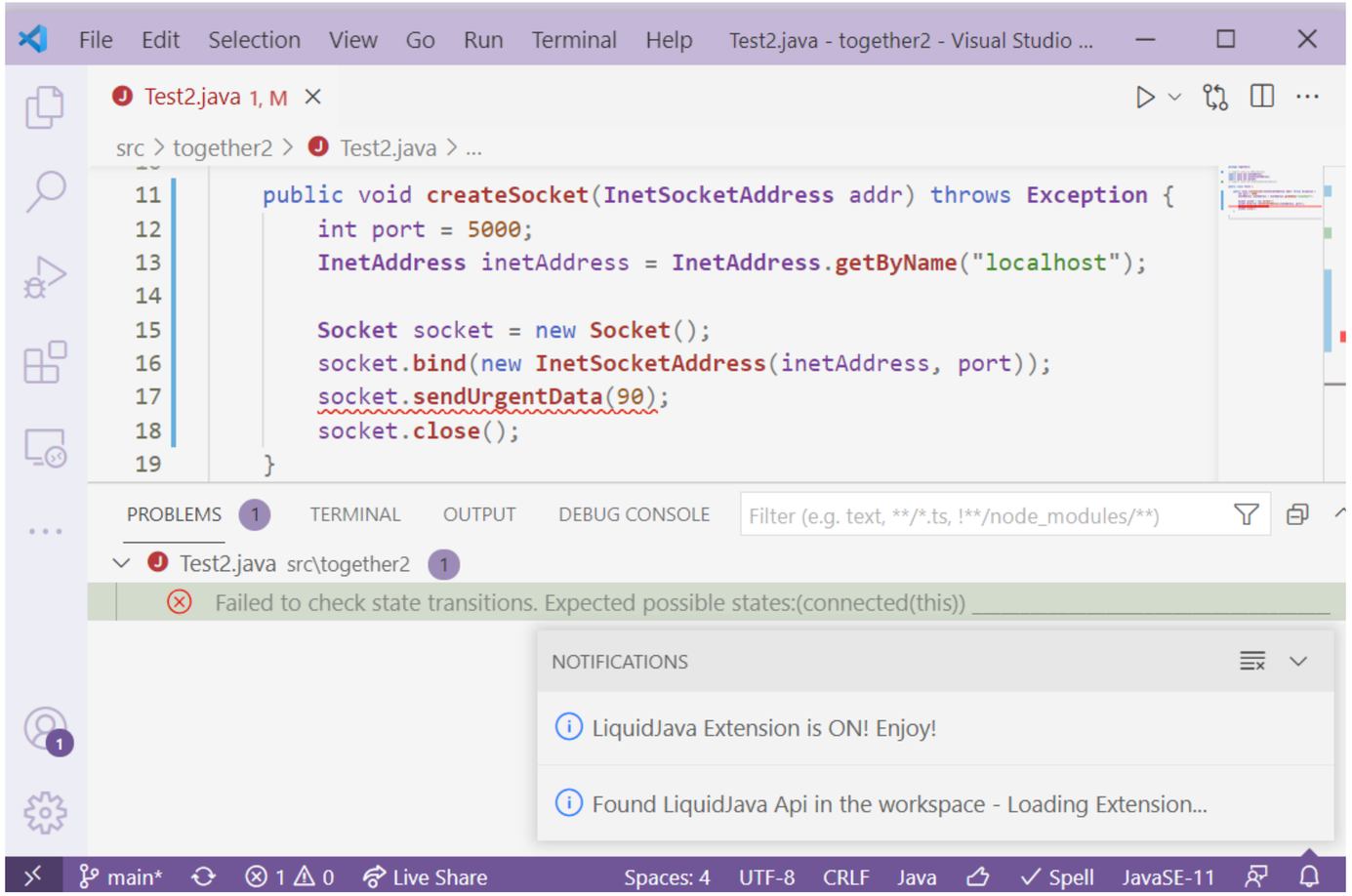

```
@StateSet({"unconnected", "bound", "connected", "closed"})
@ExternalRefinements("java.net.Socket")
interface SocketRefinements{

    @StateRefinement(from = "bound(this)",
                     to = "connected(this)")
    void connect(SocketAddress addr);


    @StateRefinement(from = "connected(this)")
    void sendUrgentData(int n);    ...
}
```







LiquidJava Prototype



User Study

Four research questions

1 Are refinements easy to understand?

2 Is it easier and faster to find implementation errors using LiquidJava than with plain Java?

3 How hard is it to annotate a program with refinements?

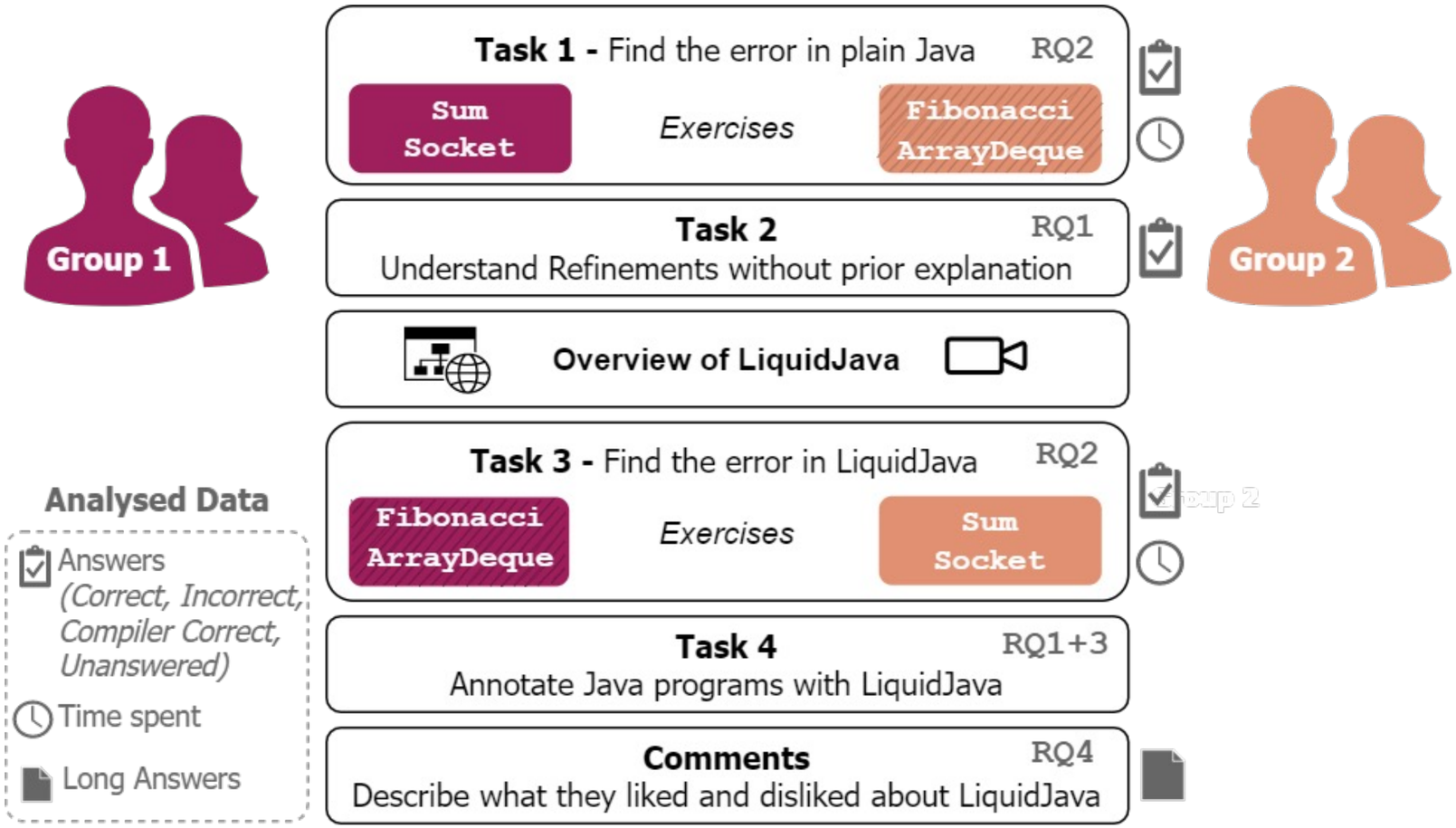
4 Are developers open to using LiquidJava in their projects?

30

56% ***Very Familiar*** **with Java**

80% ***Vaguely*** **or** ***Not Familiar*** **with Refinement Types**

Study configuration



Study configuration

Task 1 - Find the error in plain Java RQ2

`Sum` `Socket` *Exercises* `Fibonacci` `ArrayDeque`

Task 2 RQ1

Understand Refinements without prior explanation

Overview of LiquidJava

Task 3 - Find the error in LiquidJava RQ2

`Fibonacci` `ArrayDeque` *Exercises* `Sum` `Socket`

Task 4 RQ1+3

Annotate Java programs with LiquidJava

Comments RQ4

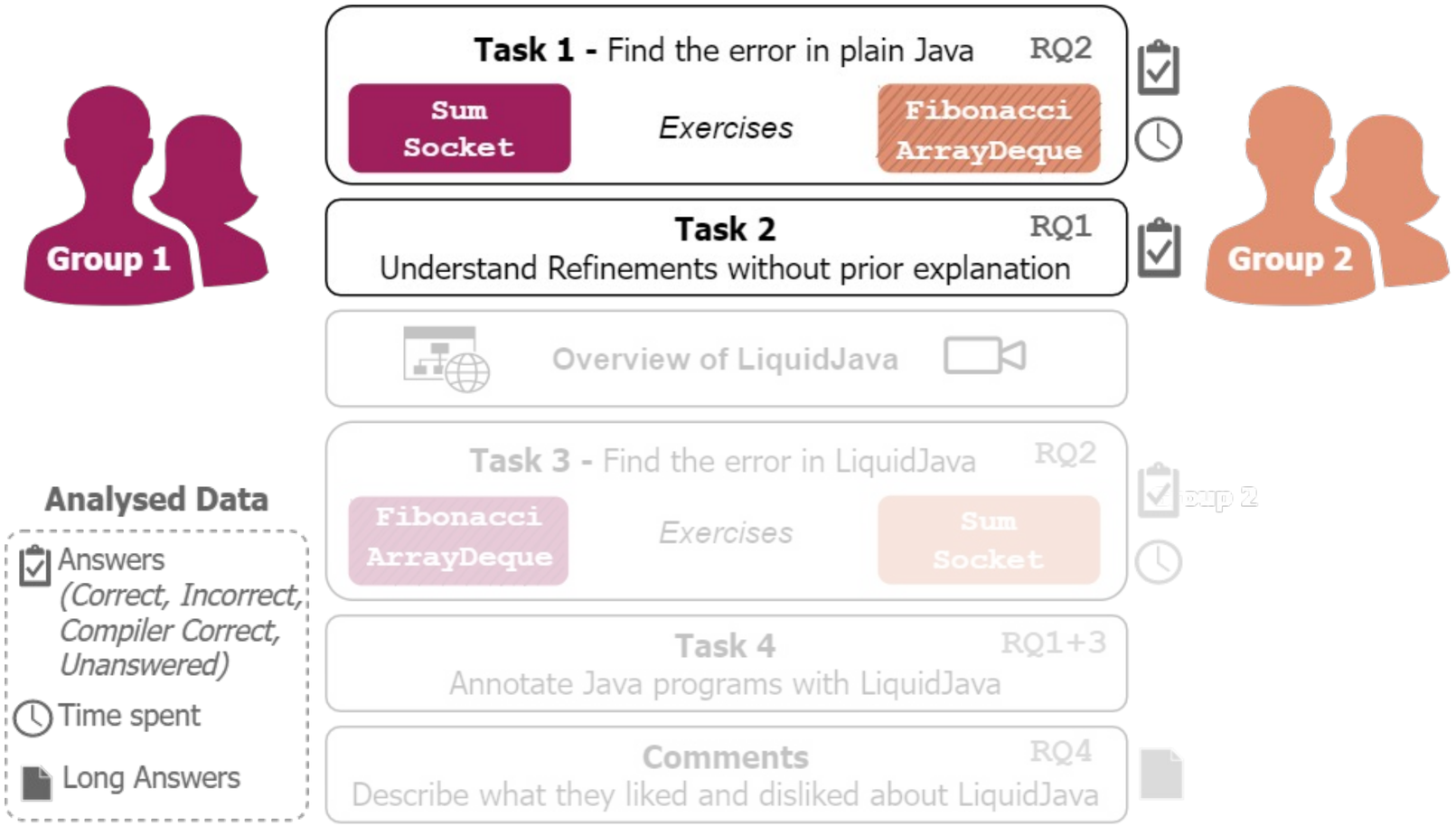
Describe what they liked and disliked about LiquidJava

Analysed Data

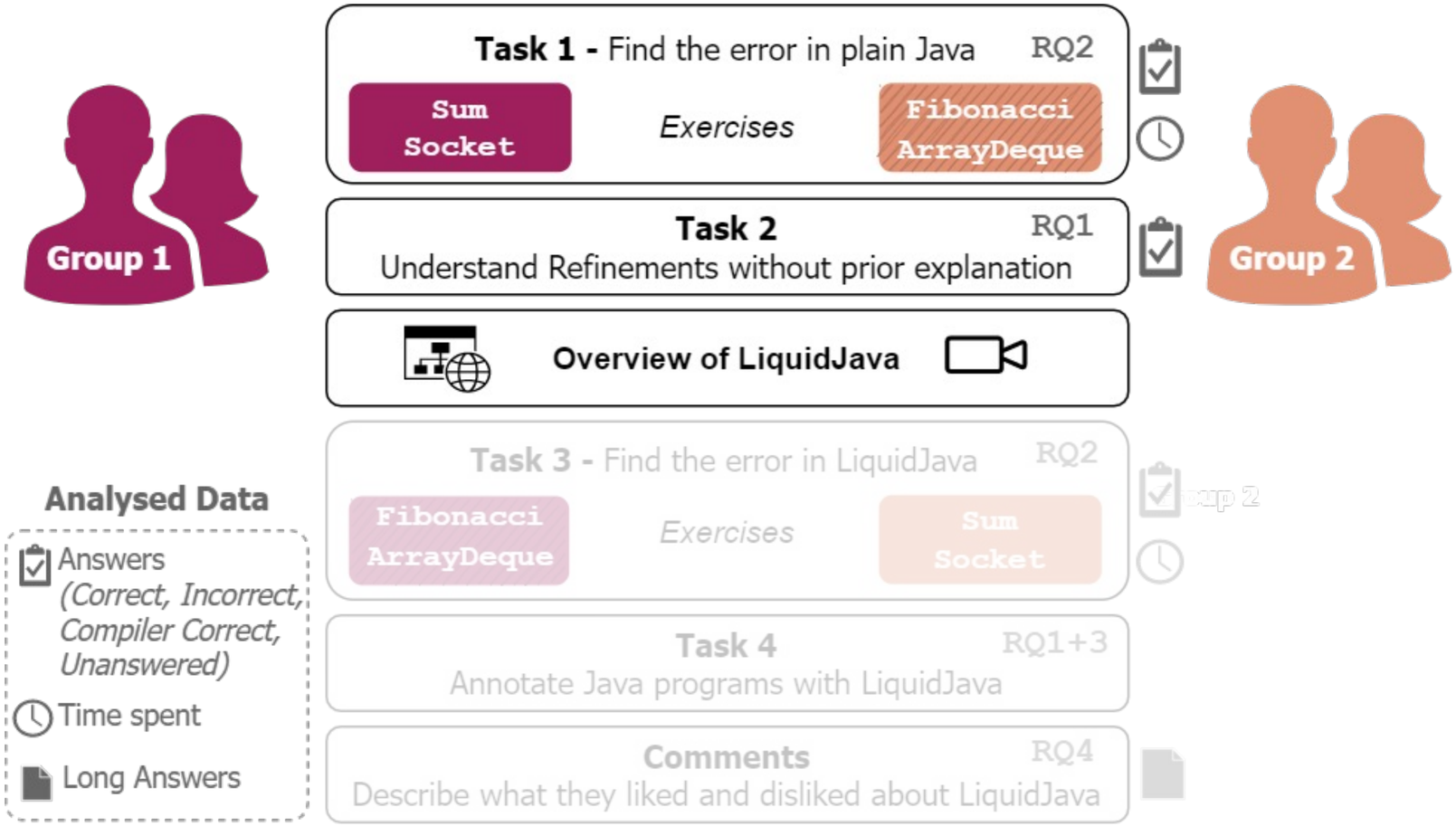
- Answers *(Correct, Incorrect, Compiler Correct, Unanswered)*
- Time spent
- Long Answers

Study configuration

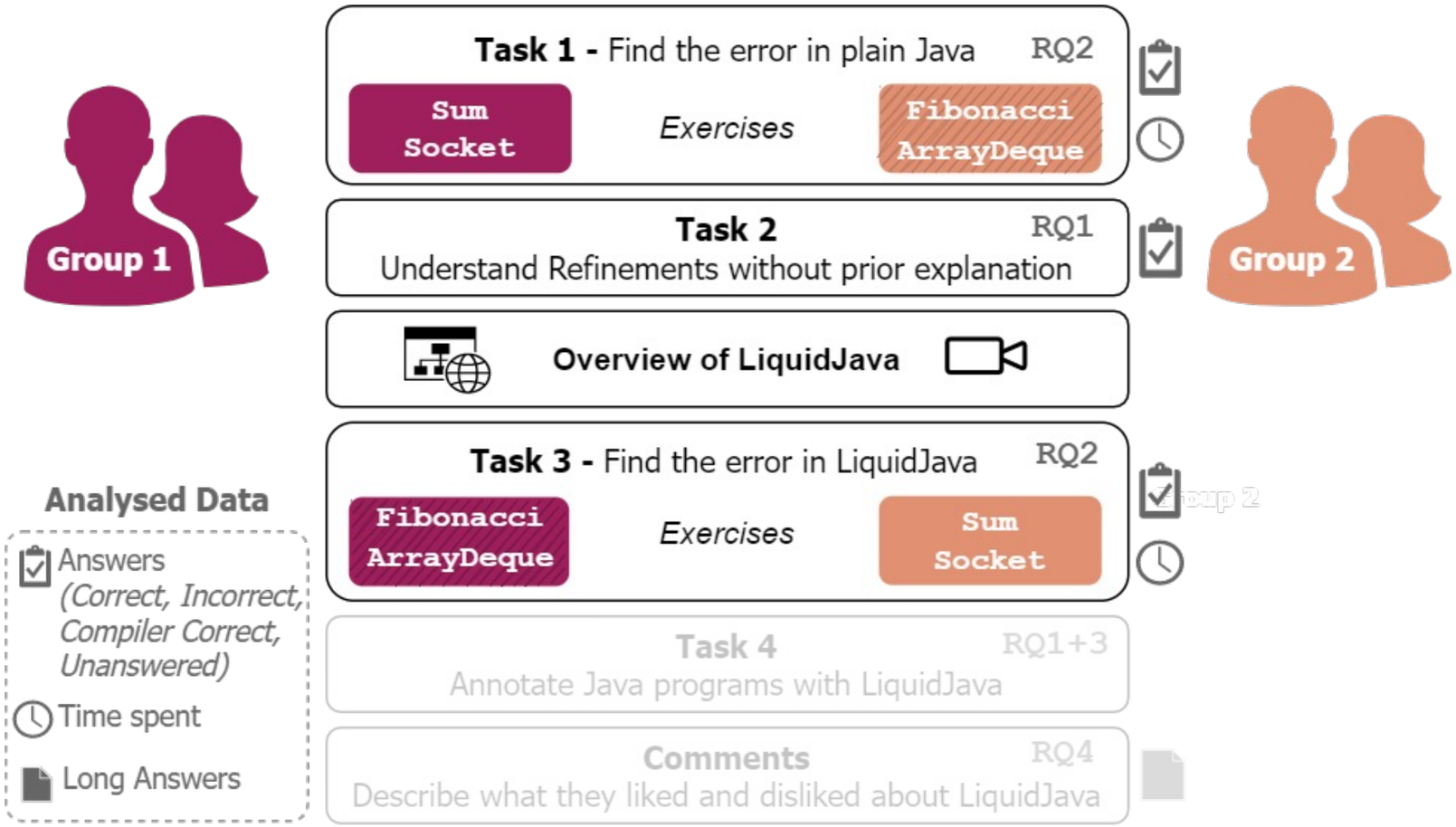
Study configuration



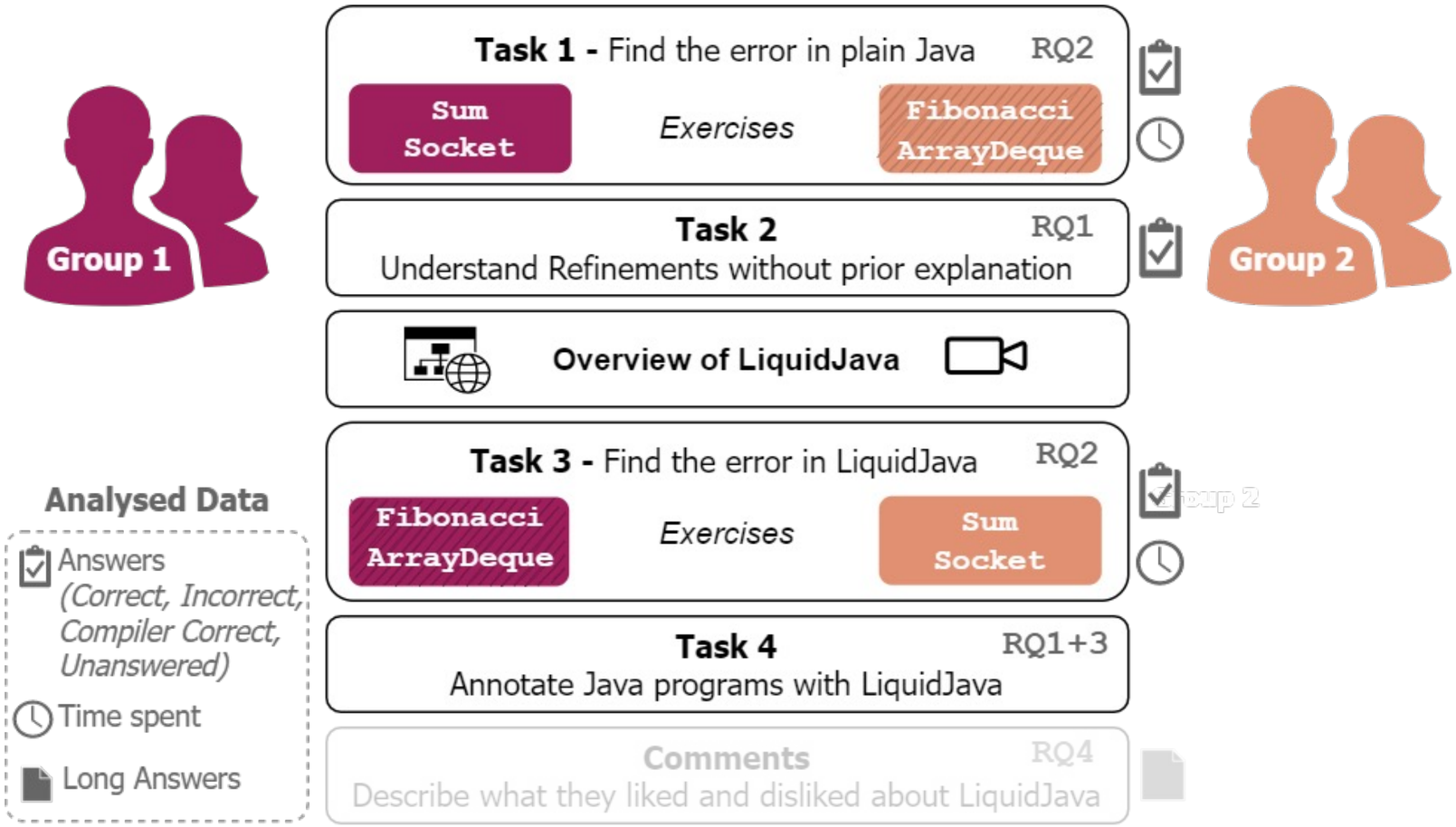
Study configuration



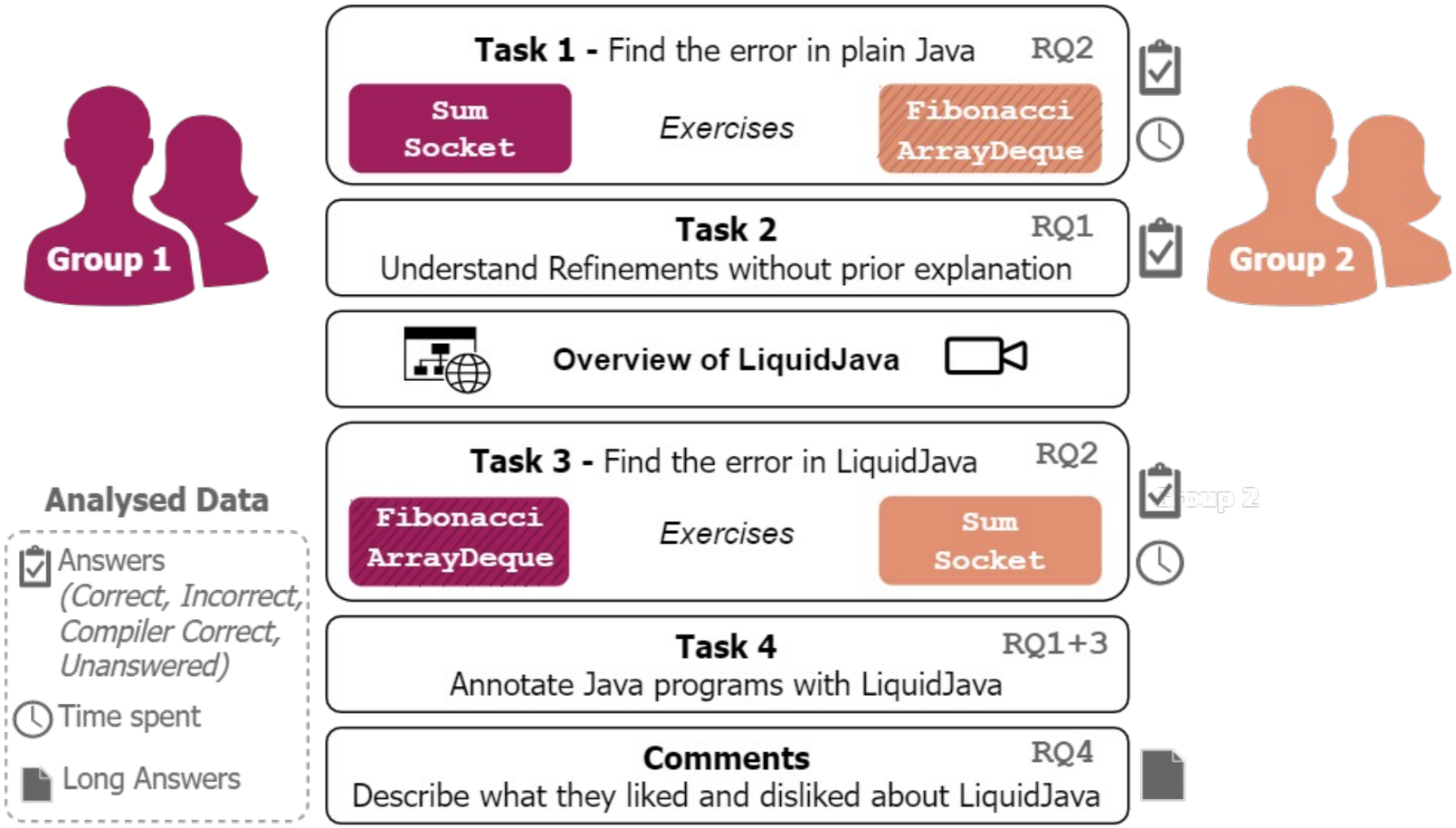
Study configuration



Study configuration



Study configuration



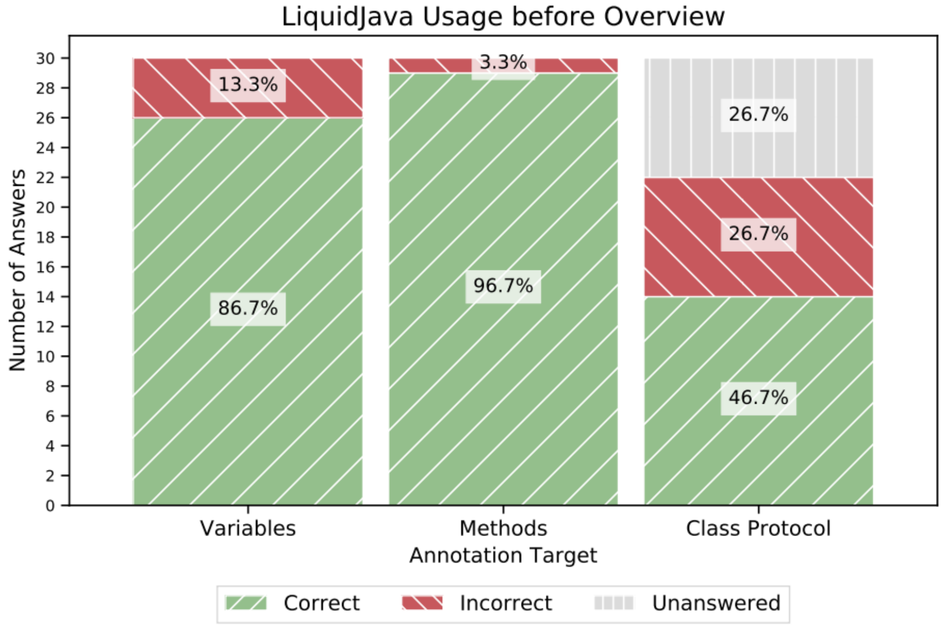
Study Conclusions

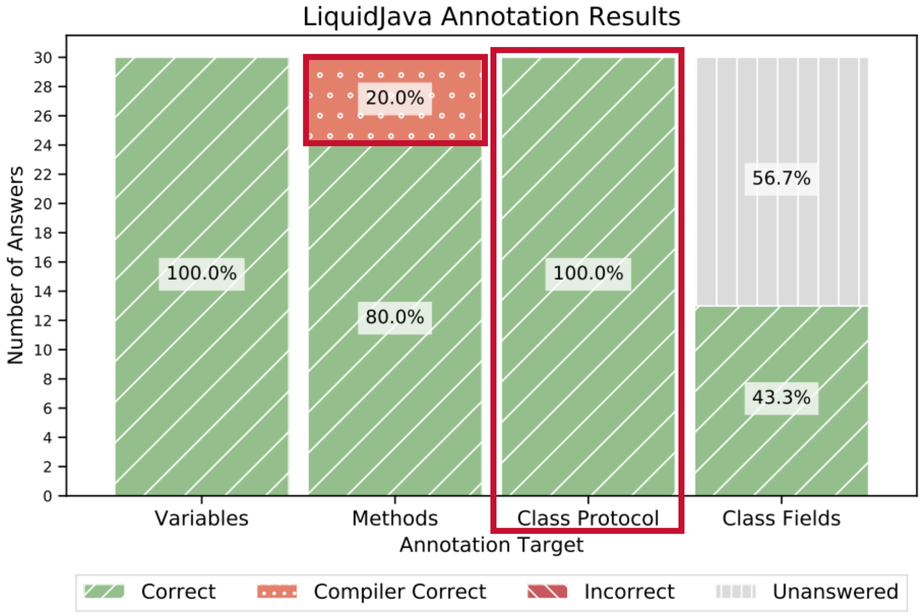
Intuitive refinements

```
@Refinement("-25 <= x && x <= 45")
int x;
//Correct:
x = 0;
//Incorrect:
x = 46;
```

Overview
Video and website

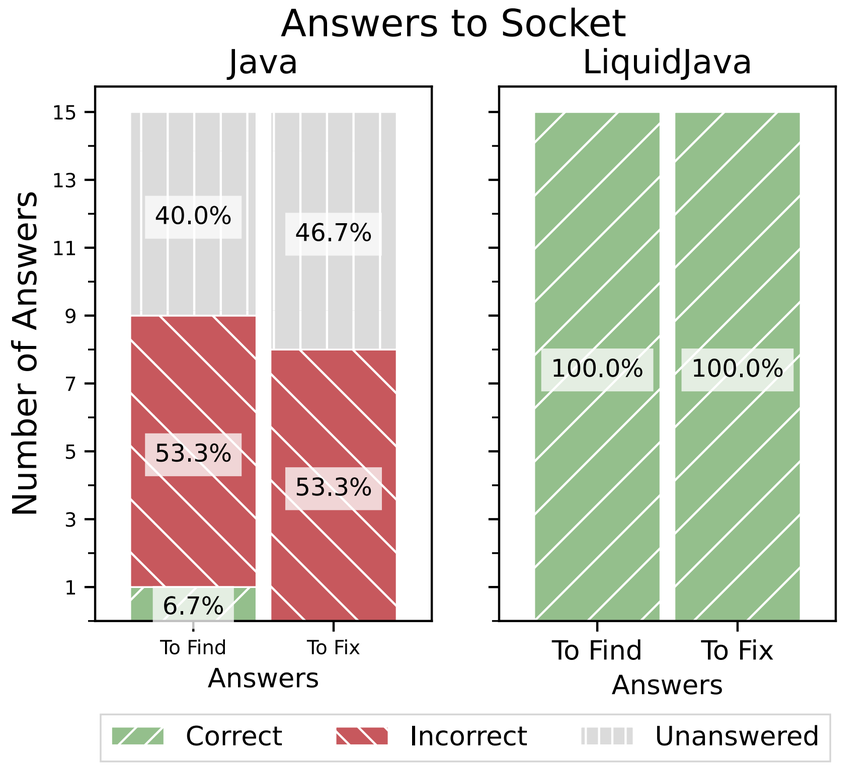
Add annotations





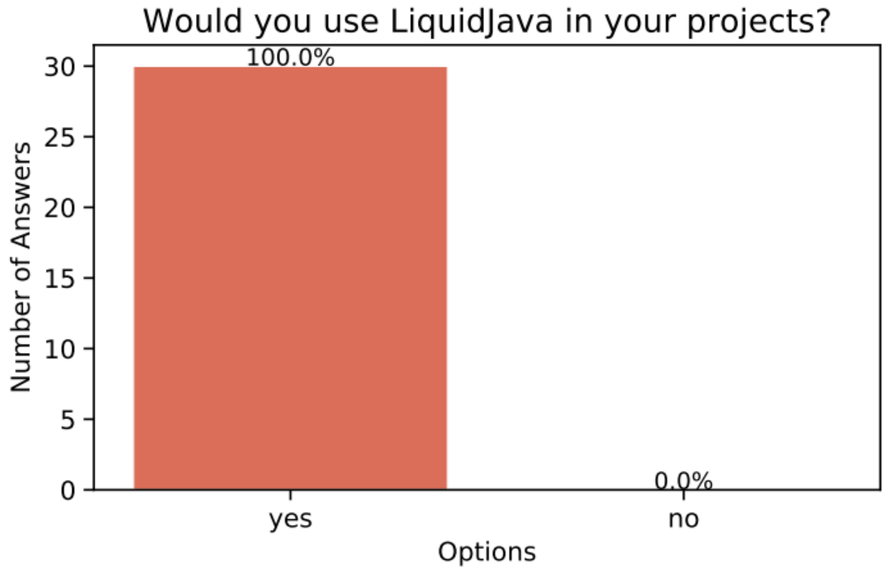
Study Conclusions

- LiquidJava **helped** developers find the bugs in code
  - Best results for **lesser-known** classes

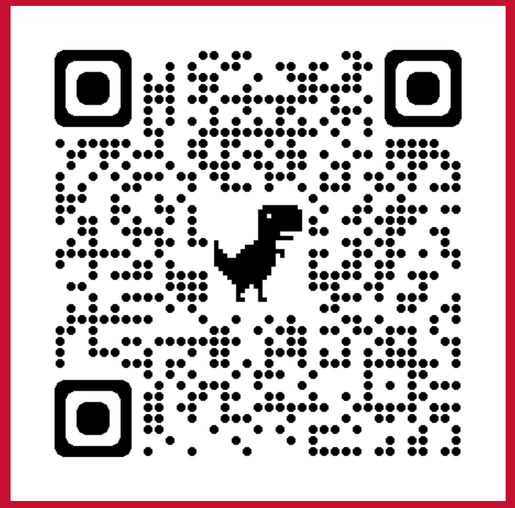


Study Conclusions

- Developers liked the error reporting and state refinements but want better better plugin integration and error messages
- Participants are interested in using LiquidJava



Usability-Oriented Design of Liquid Types for Java



cgamboa@andrew.cmu.edu

FCT Fundação para a Ciência e a Tecnologia
Carnegie Mellon Portugal
S3D
U LISBOA
Ciências ULisboa
LASIGE