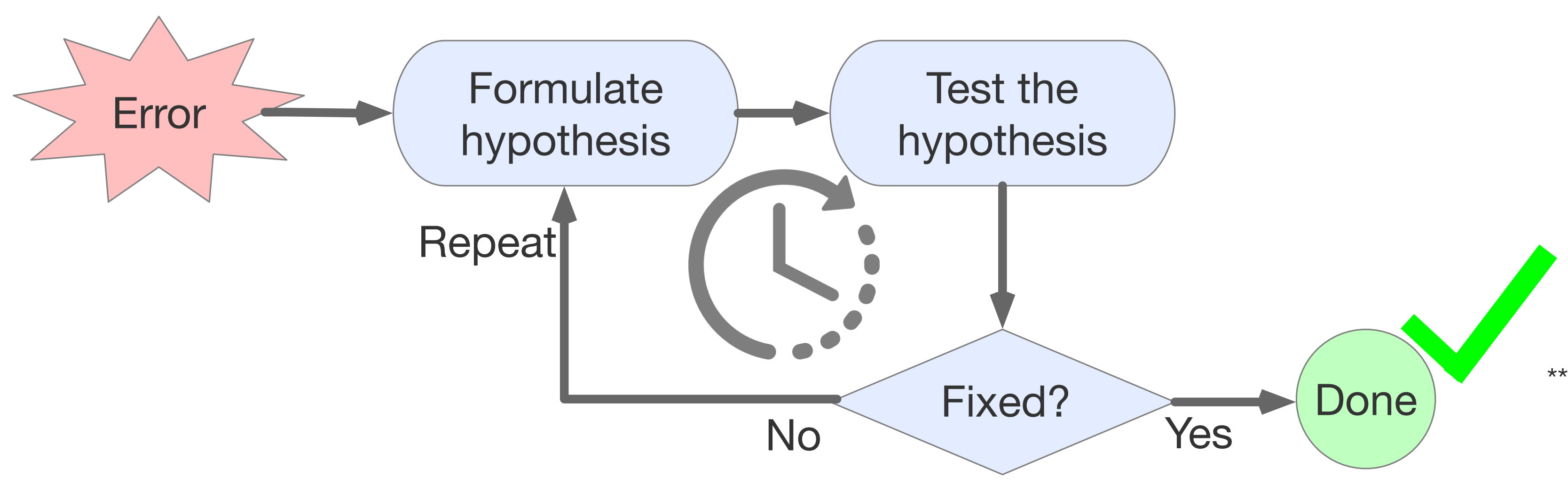
TIME-TRAVELING QUERIES
FOR FASTER DEBUGGING
AND PROGRAM COMPREHENSION

Maximilian Ignacio Willembrinck Santander Steven Costiou
Anne Etien Stéphane Ducasse

Université de Lille · Inria · CRIStAL Centre de Recherche en Informatique, Signal et Automatique de Lille · RMod

Context: The debugging process

Debugging is a time-consuming iterative process:


Error → Formulate hypothesis → Test the hypothesis → Fixed?
No → Repeat
Yes → Done

What is the cause of the error?

Trying to understand the program emoji

To understand the behavior of a program, developers ask program comprehension questions

- *When during the execution is this method called?*
- *Where are instances of this class created?*
- *Where is this variable or data being accessed?*
- *Etc.*

BETTER PROGRAM COMPREHENSION => SHORTER DEBUGGING PROCESS

Problem: Understanding programs for debugging is difficult

To find answers, developers explore their program executions using debugging tools

Debugging question: What is the value of this variable during the execution?

Step Over + Step Over + Step Into + Step Over + Step Into + Step Over + + + + + + …

Tedious!

Breakpoint hit + Breakpoint hit + Breakpoint hit + Step Over + Step Into + Step Over + Stepped too far!

Missed the target!

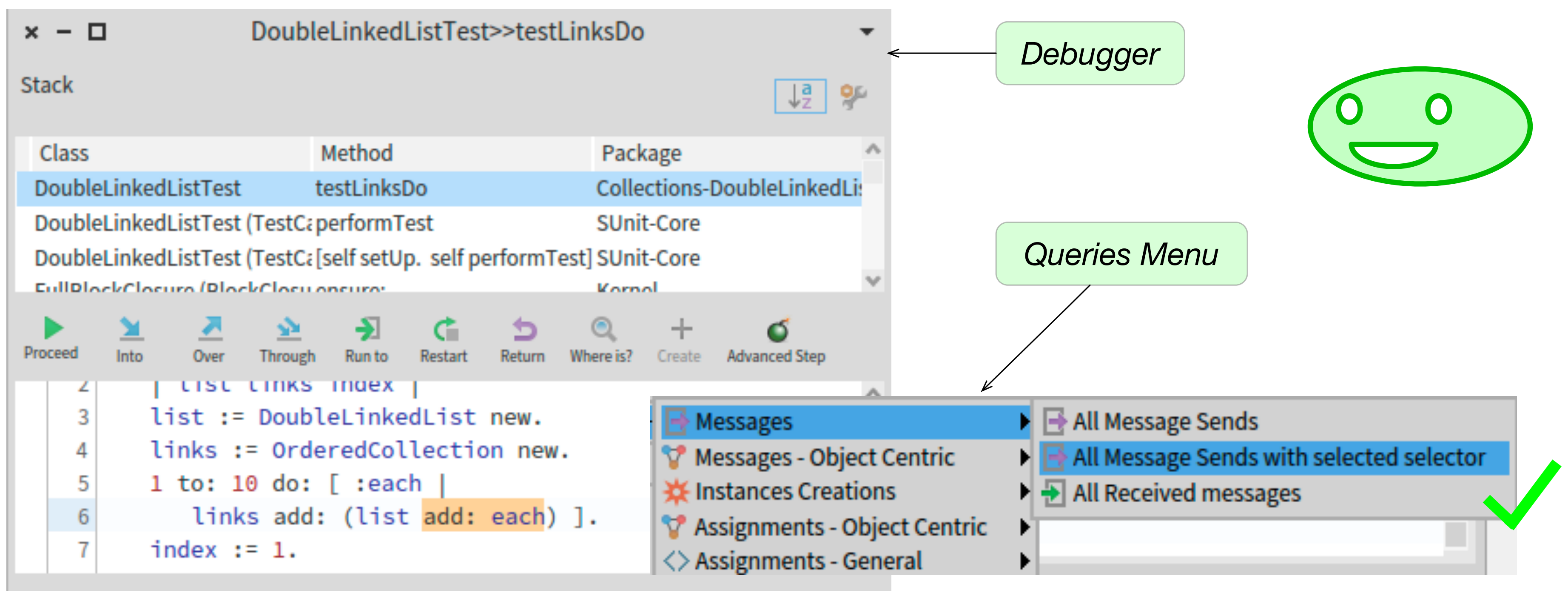
Can we do better?

Solution: Time-Traveling Queries

Time-Traveling Queries (TTQs)

*Do you have a **debugging question?***

Select a Time-Traveling Query from the Queries Menu!



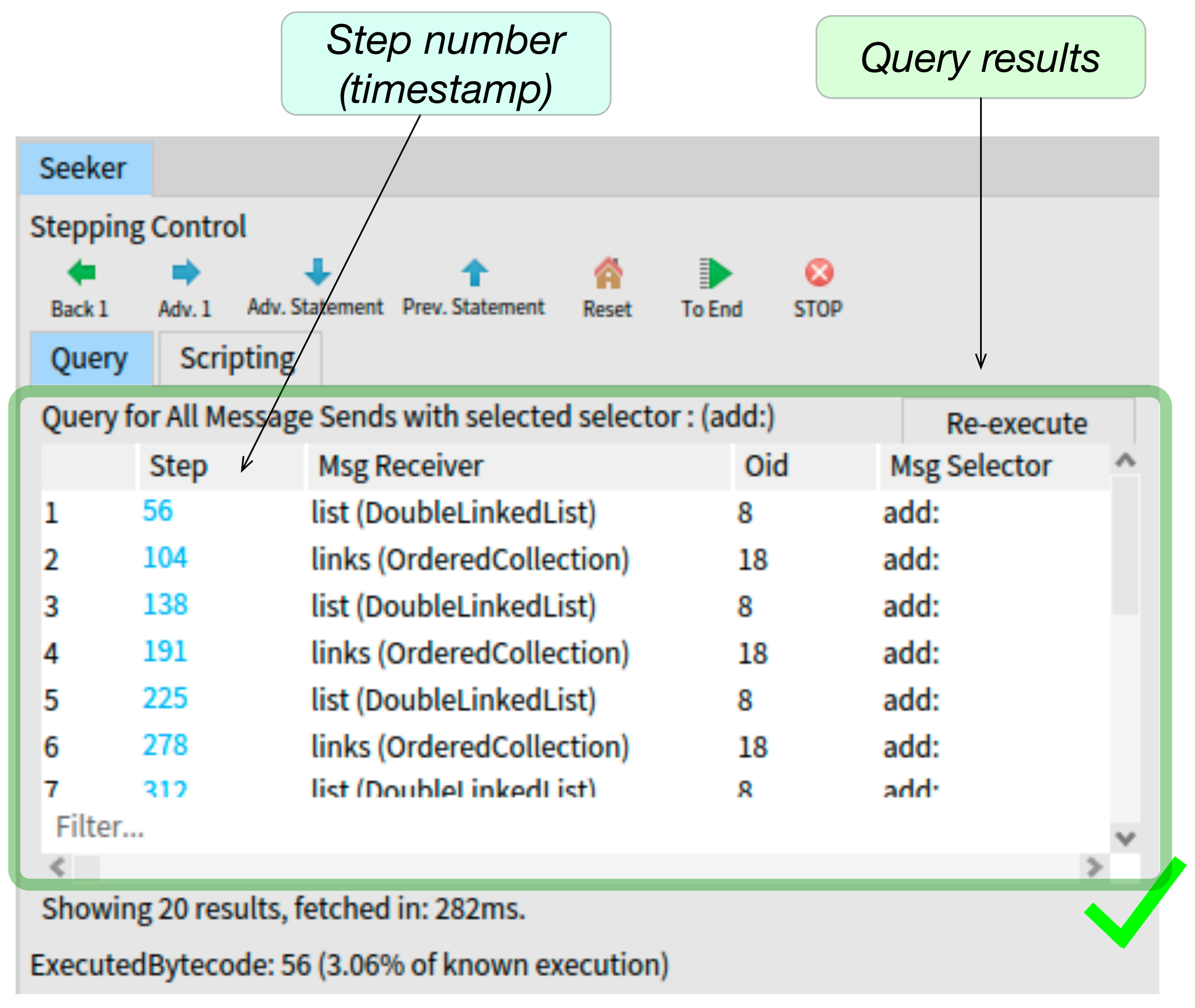
DoubleLinkedListTest>>testLinksDo

Class	Method	Package
DoubleLinkedListTest	testLinksDo	Collections-DoubleLinkedLi
DoubleLinkedListTest (TestCa	performTest	SUnit-Core
DoubleLinkedListTest (TestCa	[self setUp. self performTest]	SUnit-Core
FullBlockClosure (BlockClosu	ensure:	Kernel

```
    | list links index |
    list := DoubleLinkedList new.
    links := OrderedCollection new.
    1 to: 10 do: [ :each |
        links add: (list add: each) ].
    index := 1.
```

- *TTQs request information of an execution related to **common debugging questions.***

- ***Find answers directly** in your query result (Don't miss target!).*
- ***"Click & Time-Travel"***
  *Reverse or advance the execution **jumping** directly to any of the results timestamp (Less tedious!).*
- ***Explore** your execution states **forward or backward**.*



Query for All Message Sends with selected selector : (add:)

	Step	Msg Receiver	Oid	Msg Selector
1	56	list (DoubleLinkedList)	8	add:
2	104	links (OrderedCollection)	18	add:
3	138	list (DoubleLinkedList)	8	add:
4	191	links (OrderedCollection)	18	add:
5	225	list (DoubleLinkedList)	8	add:
6	278	links (OrderedCollection)	18	add:
7	312	list (DoubleLinkedList)	8	add:

Showing 20 results, fetched in: 282ms.

ExecutedBytecode: 56 (3.06% of known execution)

*Do you have another **debugging question?*** *Just select another query!*

*There is no query for your **debugging question?*** *Write your own TTQ!*

Example:

```
"Lists global variables assignments"
^ UserTTQ from: programStates
    select: [ :state | state isAssignment and: [ state node variable isGlobalVariable ] ]
    collect: [ :state |
        ResultItem new
            bytecodeIndex: state bytecodeIndex;
            variableName: state assignmentVariableName;
            yourself ]
```

From where to extract the data?

What program states are relevant?

What should be included in the results?

Time-Traveling Queries Evaluation

With TTQs, developers perform program comprehension tasks more accurately, faster, and with less effort than with standard debugging tools*.

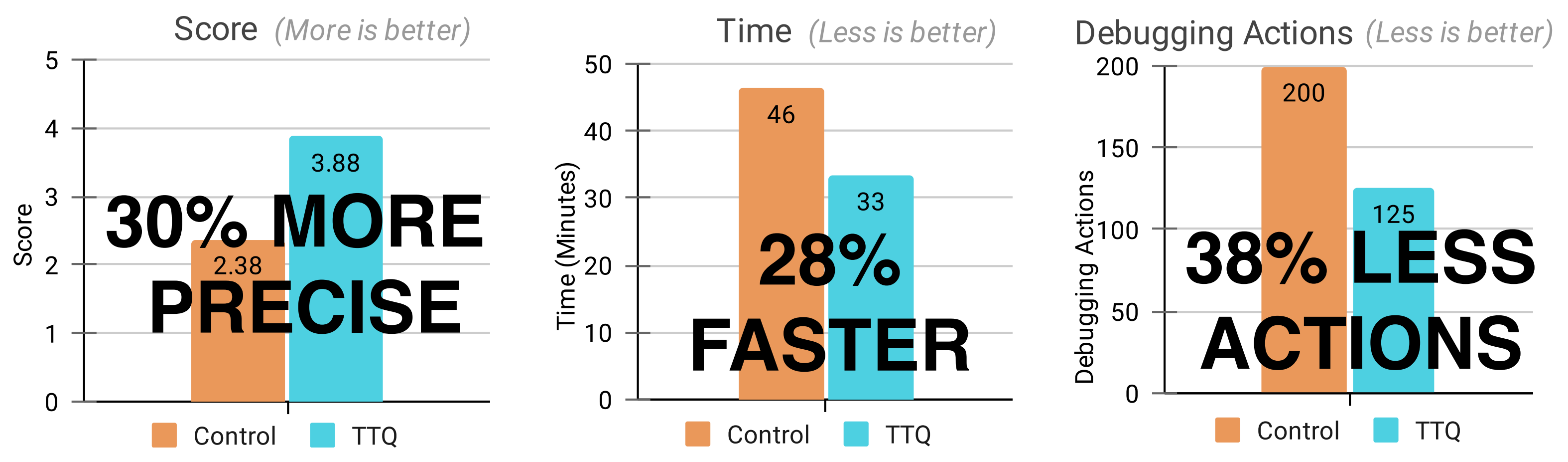
Controlled Experiment

- Repeated Measures Design (Within-subject)
- **34 Participants**.

Research Question

Do TTQs improve program comprehension tasks of participants regarding precision, time spent and efforts? *(vs. using standard debugging tools)*

Results



Control: Without Time-Traveling Queries **TTQ:** Using Time-Traveling Queries

(*) M. Willembrinck, S. Costiou, A. Etien and S. Ducasse, "Time-Traveling Debugging Queries: Faster Program Exploration," *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, 2021, pp. 642-653.

Future work

- Scaling the solution to daily debugging scenarios.
- Studying new relevant queries.
- Research TTQs generalization to different programming languages.
- Designing new TTQs-based debugging tools.

Code and video demonstration available at: github.com/Willembrinck/TTQ-Debugging