Time Traveling Queries for Faster Program Exploration

Maximilian Willembrinck

Université de Lille
CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille
Inria
INVENTEURS DU MONDE NUMÉRIQUE
RMod

Presentation Agenda

Context

- **Debugging, Program Comprehension and Exploration** ← **Next**
- Research Question

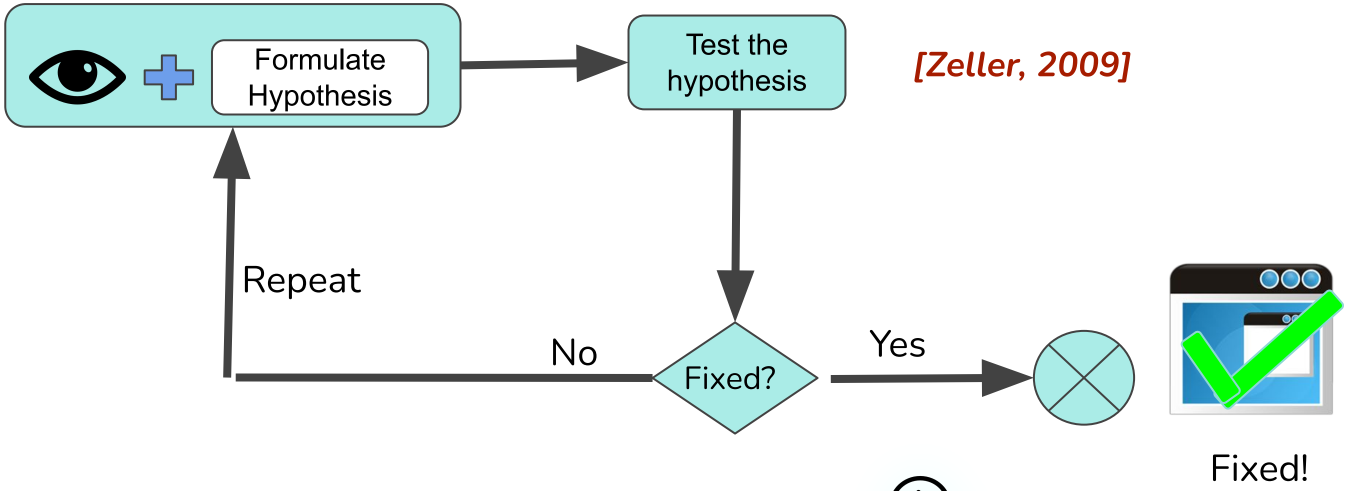
Time-Traveling Queries

- What are they?
- Queries for Program Comprehension

Experiment

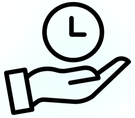
- User Study Description
- Results

Context - The Debugging Process



[Zeller, 2009]

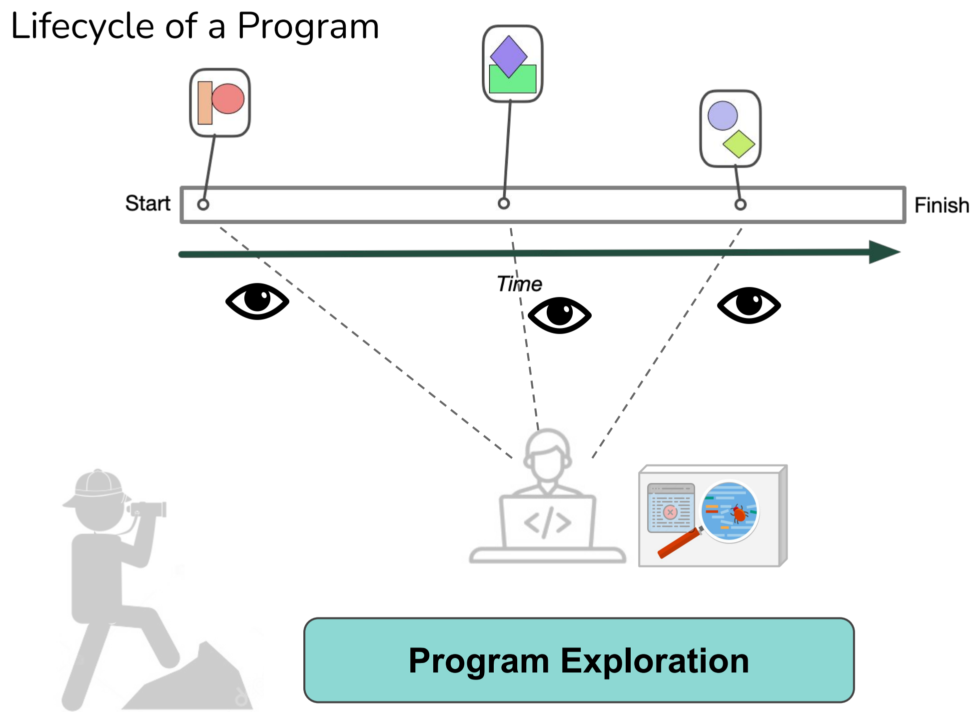
Good Hypothesis → Less Iterations



Good Hypothesis ← **Understand program behavior**

Zeller, 2009: "Why programs fail: a guide to systematic debugging"

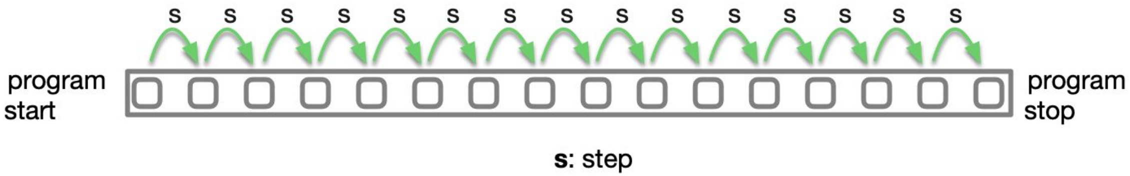
Context - Understanding a program behavior



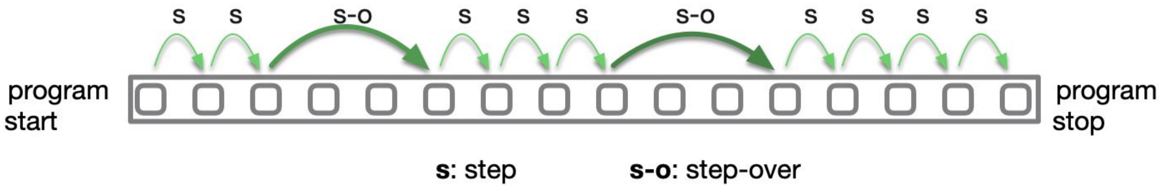
Manually exploring a program execution

Stepping

Step Into



Step Over

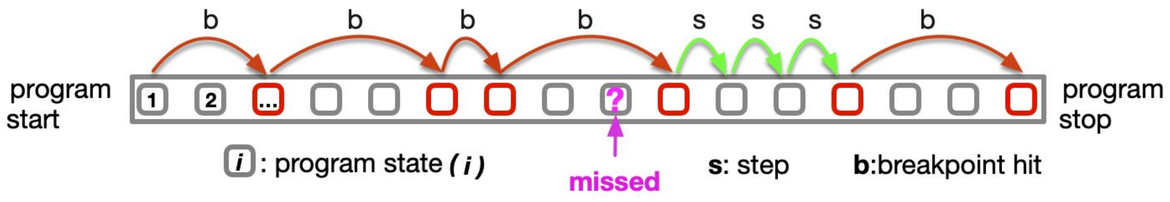


Manual and tedious

Traverse States Forward in Time

Manually exploring a program execution

Breakpoints

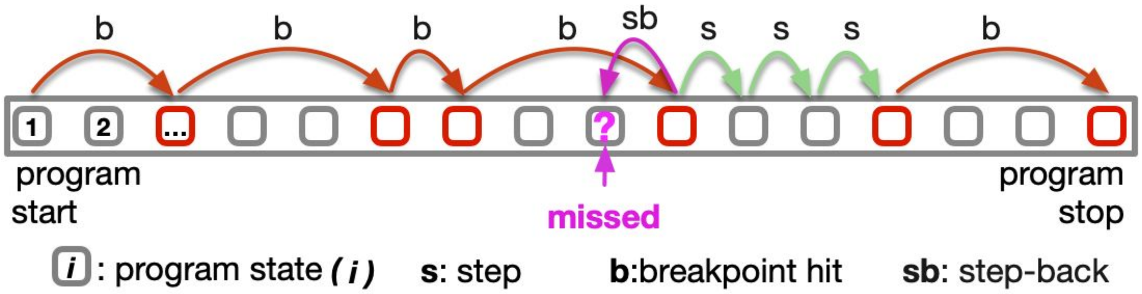


“Missing critical information” problem

Traverse States Forward in Time

Manually exploring a program execution

Time-Traveling Debuggers



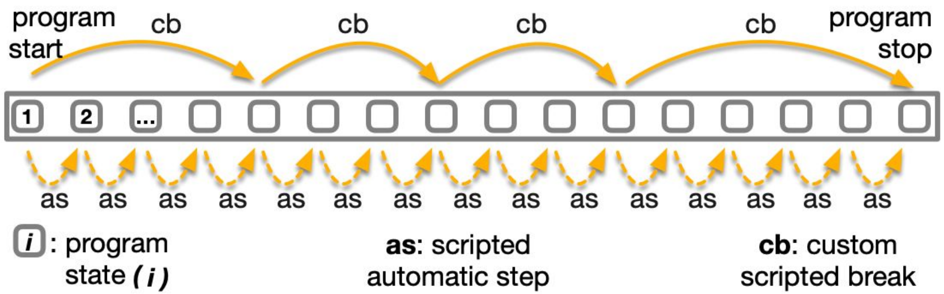
Helps with the "Missing critical information" problem.

Still tedious

Traverse States Forward in Time

Exploring a program execution

Scriptable Debuggers



Help with tediousness but, requires prior knowledge of the execution

Exploring a program execution

Basic stepping, breakpoints, scriptable debugger, time-traveling, etc.

Debugging Questions

Difficult to Translate Into

Sequence of Steps

Problem Summary

Program Exploration is ...

- Manual/Tedious
- Imprecise and miss critical information
- Translating debugging questions -> debugging actions is difficult

Presentation Agenda

Context
- Debugging, Program Comprehension and Exploration
- **Research Question** ← **Next**

Time-Traveling Queries
- What are they?
- Queries for Program Comprehension

Experiment
- User Study Description
- Results

Research Question

"Can we **express** general program comprehension **questions** as **queries** over programs executions, and does that **improve** program exploration regarding developers' efforts, time spent and precision, **compared to standard debugging tools**?"

Proposed Solution

Time-Traveling Queries

Presentation Agenda

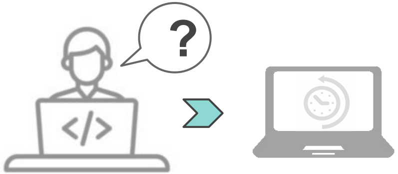
Context
- Debugging, Program Comprehension and Exploration
- Research Question

Time-Traveling Queries
- **What are they?** ← **Next**
- Queries for Program Comprehension

Experiment
- User Study Description
- Results

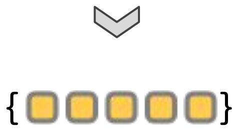
Time Traveling Queries

➔ Programmatic requests for execution data

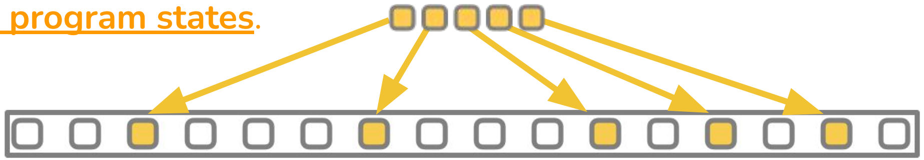


➔ Automatically traverse program states.

➔ Requesting and collecting **relevant data**.



➔ Enabling direct time travel to **relevant program states**.



Time-Traveling Queries

Key supporting components

1. Time-Traveling Debugger

Advances or restores an execution to any point in time

2. ProgramStates

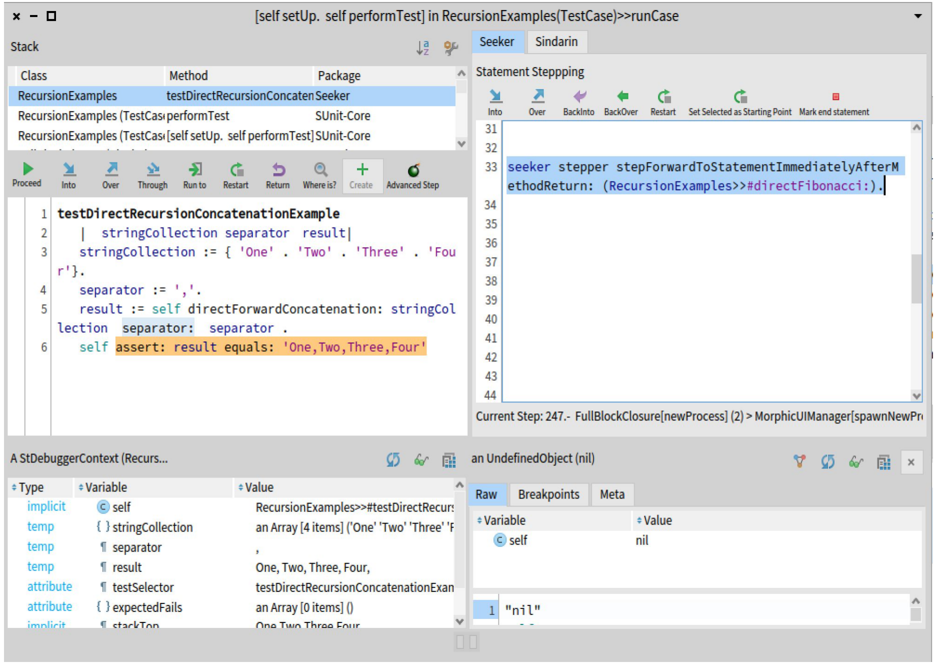
An iterable collection of all the program states

3. Query

A programmatic request of execution data

1. Time-Traveling Debugger

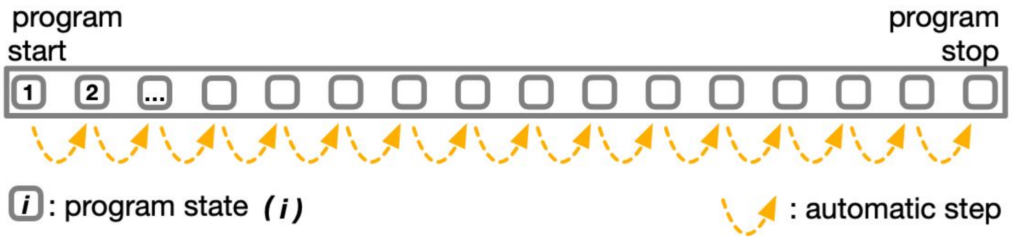
- As an **extension** of Pharo 9.0 **debugger**
- Allows to **reverse** a program's execution (step backwards)
- **Replay**-based Implementation



2. ProgramStates

➔ A generator of ProgramState

- Iterable object that exposes an API to retrieve execution data from every state of the program (during its iteration)
- Every iteration of the loop advances execution by one step
- No trace(recording) is required to answer queries.



3. Query

➔ Declared like this

From where to collect?

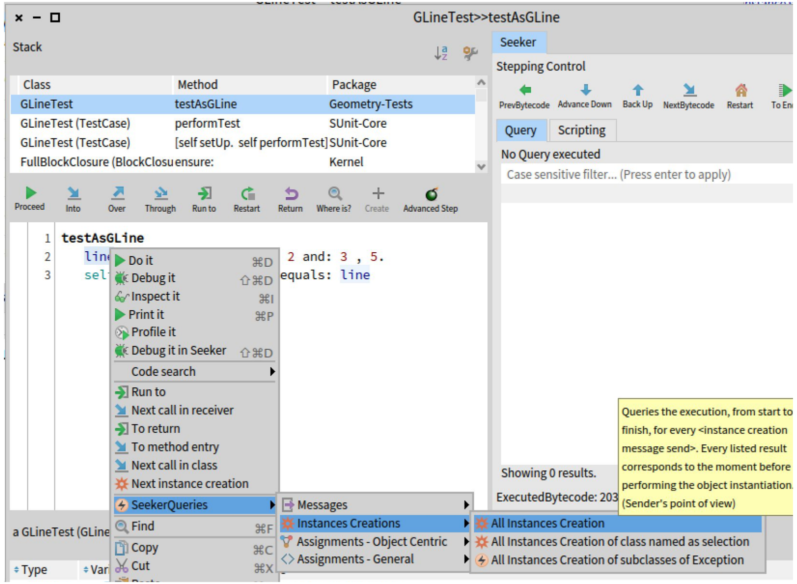
Which states are relevant?

```
allReturnValuesQuery := Query
        from: programStates
        select: [ :cs | cs  isMethodReturn ]
        collect: [ :cs |
  Dictionary newFrom: {
            (#receiverClass -> cs receiverClass).
            (#methodSelector -> cs methodSelector).
            (#returnValue -> cs methodReturnValue) } ].
```

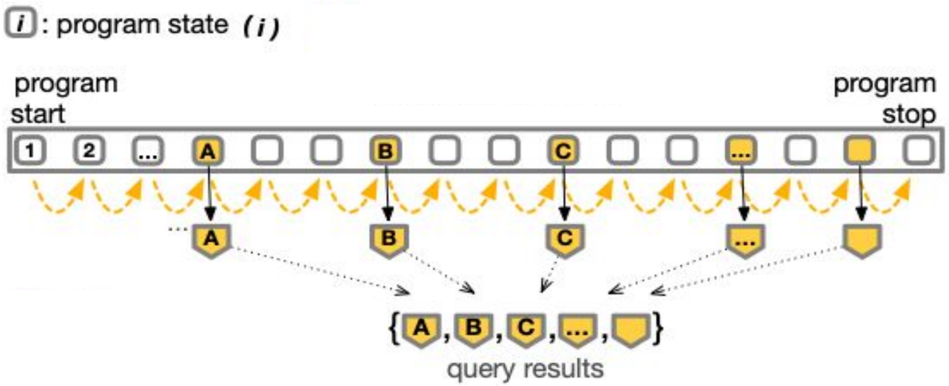
What to Collect?

How to Collect
the data?

Time-Traveling Query Usage

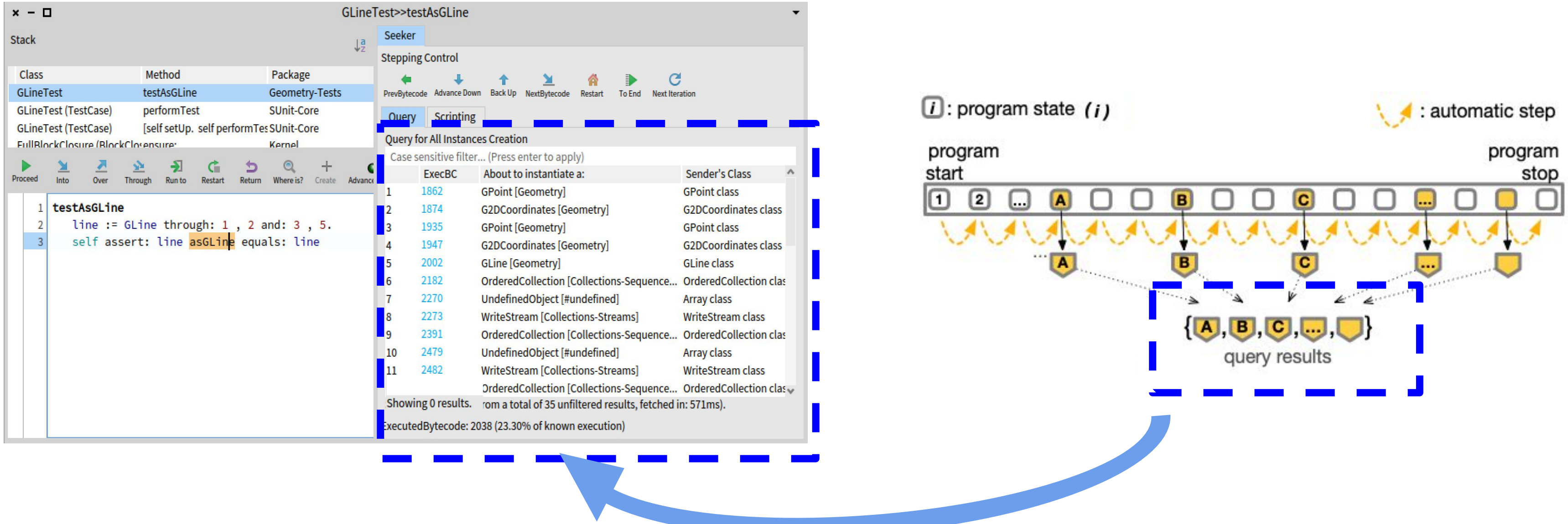


Query from: programStates
select: X
collect: Y

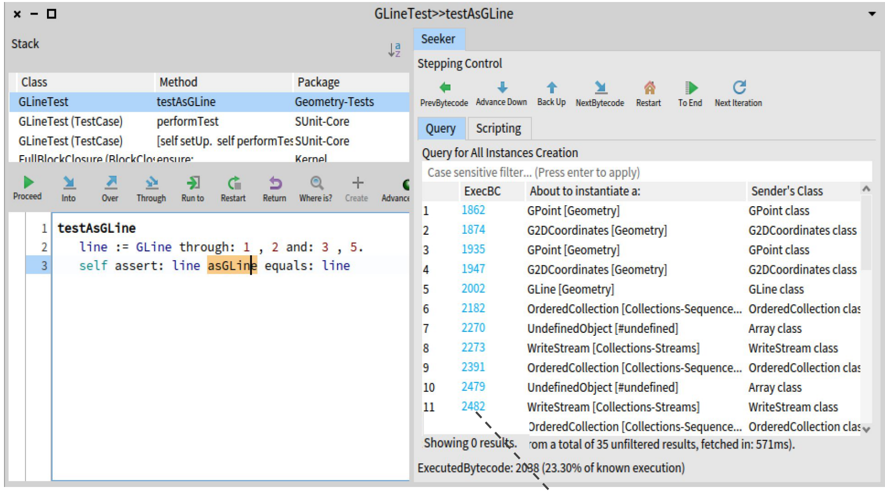


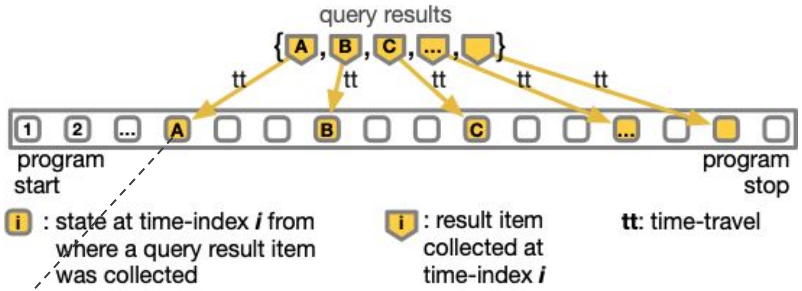
Query results

➔ Shown in UI like this



Time-Traveling from Results





time-index

Presentation Agenda

Context
- Debugging, Program Comprehension and Exploration
- Research Question

Time-Traveling Queries
- What are they?
- **Queries for Program Comprehension** ← **Next**

Experiment
- User Study Description
- Results

Queries for program comprehension

List of Time-Traveling queries

I	**Messages.**
I.1	Find all messages sent during the execution.
I.2	Find all messages, with a given selector, sent during the execution.
I.3	Find all received messages.
II	**Instances Creation.**
II.1	Find all instance creations.
II.2	Find all instance creations of a class with a given name.
II.3	Find all instance creations of exceptions.
III	**Assignments - Object Centric.**
III.1	Find all assignments of instance variables for the receiver of the currently executed method.
III.2	Find all assignments of instance variables for a particular instance.
III.3	Find all assignments of a given instance variable for the receiver of the currently executed method.
IV	**Assignments - General.**
IV.1	Find all assignments of variables with a given name.
IV.2	Find all assignments of any variable.
IV.3	Find all assignments of instance variables for instances of a given class.

Queries for program comprehension

Based on questions from the literature

[Sillito, 2006]
[Kubelka, 2014]

- [13.] When during the execution is this method called?
- [14.] Where are instances of this class created?
- [15.] Where is this variable or data structure being accessed?
- [19.] What are the values of these arguments at run time?
- [20.] What data is being modified in this code?
- [32.] Under what circumstances is this method called or exception thrown?

Sillito, 2006: "Questions Programmers Ask During Software Evolution Tasks"
Kubelka, 2014: "Asking and Answering Questions during a Programming Change Task in Pharo Language"

How they help?

List of queries ➤ Translation of questions into scripts/steps

Time-Traveling from results ➤ Less “manual and tedious” traversal

Presentation Agenda

Context
- Debugging, Program Comprehension and Exploration
- Research Question

Time-Traveling Queries
- What are they?
- Queries for Program Comprehension

Experiment
- **User Study Description** ← **Next**
- Results

User Study - Queries for Debugging

Evaluated our **Queries** approach **vs Standard** Debugging Techniques, for **program comprehension tasks**:

Do Time Traveling Queries …

1. Improve **correctness**?
2. Reduce the **employed time**?
3. Reduce the **number of debugging actions**?

(Versus Standard Debugging Tools)

User Study - Experiment Design

- ➔ Quantitative experiment
- ➔ Repeated Measures Design (Within-subject)
- ➔ **34 Participants**.
- ➔ Session of **90 minutes**, solving program comprehension tasks, using:
  - ◆ Time-Traveling Queries.
  - ◆ Standard Debugging Tools
- ➔ Measure the effect of: “TTQs”
  - ◆ On: Participant **Score**, **Time**, **Debugging Actions**

- ➔ Followed by Qualitative Survey

Experiment Tasks. From "simpler" ...

- How many times is the method #atEnd of the object 'generator' is called? and from which methods?

```
testAtEnd
    | generator |
    generator := self numbersBetween: 1 and: 3.
    self deny: generator atEnd.
    generator next.
    self deny: generator atEnd.
    generator next.
    self deny: generator atEnd.
    generator next.
    self assert: generator atEnd
```

Experiment Tasks.
To less “simple”…

- ➔ What are the different values of the `pc` instance variable of the first `newContext` object during this test?

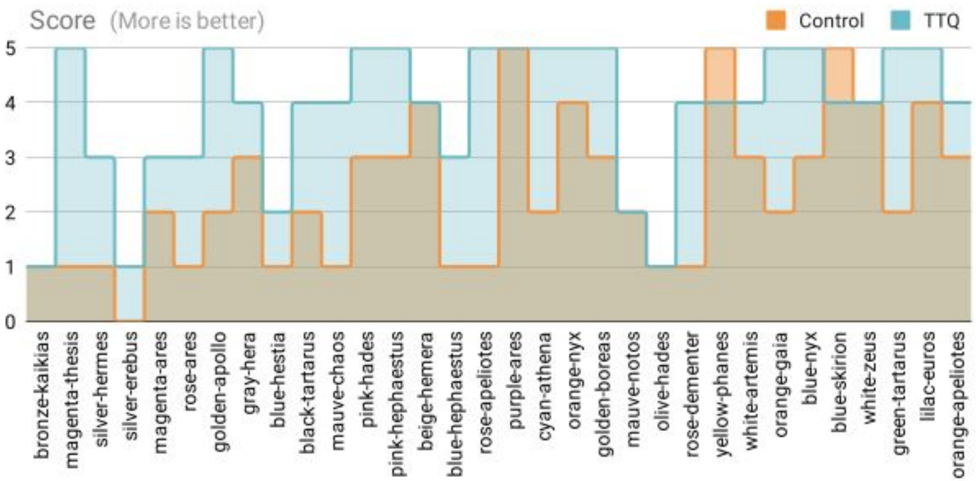
```
testSteppingReturnSelfMethod

  |newContext|
  aMethodContext := Context
     sender: thisContext
     receiver: SimulationMock new
     method: (SimulationMock >>#exampleSelfReturnCall)
     arguments: #().

  aMethodContext step.
  aMethodContext stepIntoQuickMethod: true.
  newContext := aMethodContext step.

  "We're in the quick method now, it should be steppable"
  self assert: newContext sender identicalTo: aMethodContext.
  self assert: newContext willReturn.

  newContext := newContext step.
  self assert: newContext identicalTo: aMethodContext
```

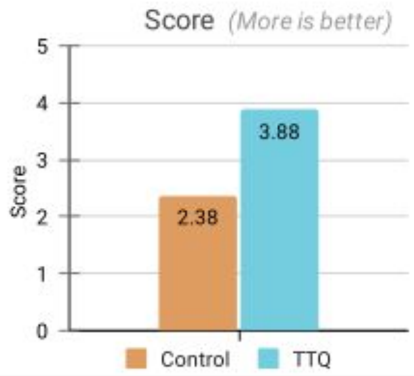
Presentation Agenda

Context	Debugging, Program Comprehension and Exploration
	Research Question
Time-Traveling Queries	What are they?
	Queries for Program Comprehension
Experiment	User Study Description
	Results

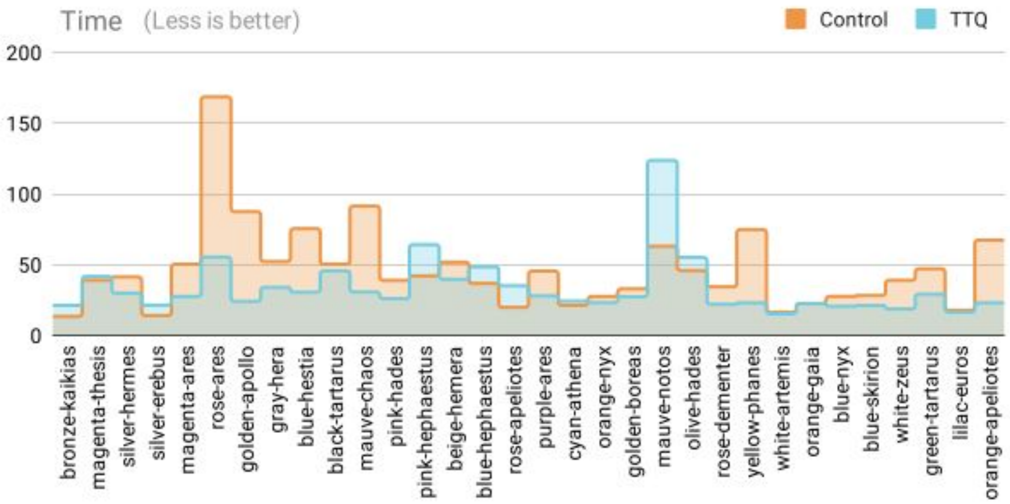
Next

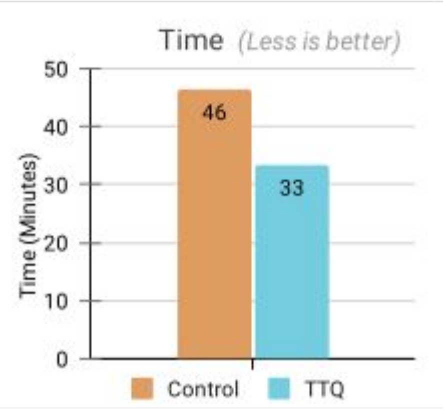
Participants Score





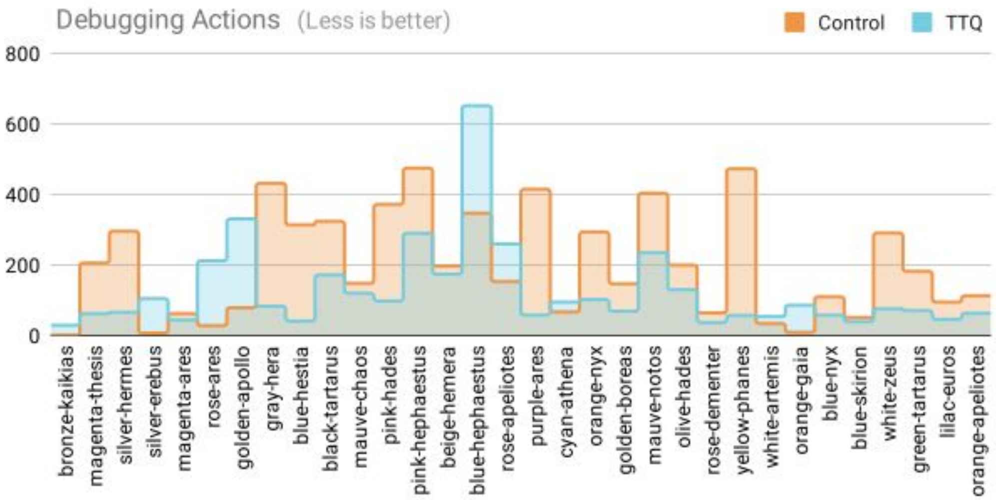
Participants Time

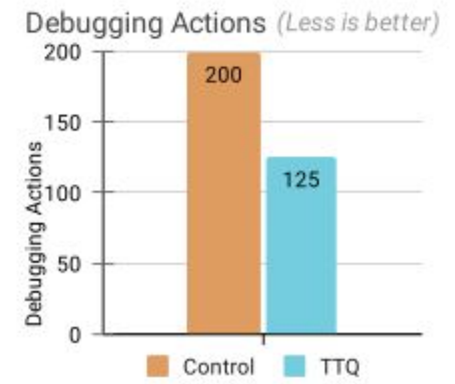




Time to complete all ‘Control’ and ‘TTQs’ tasks.

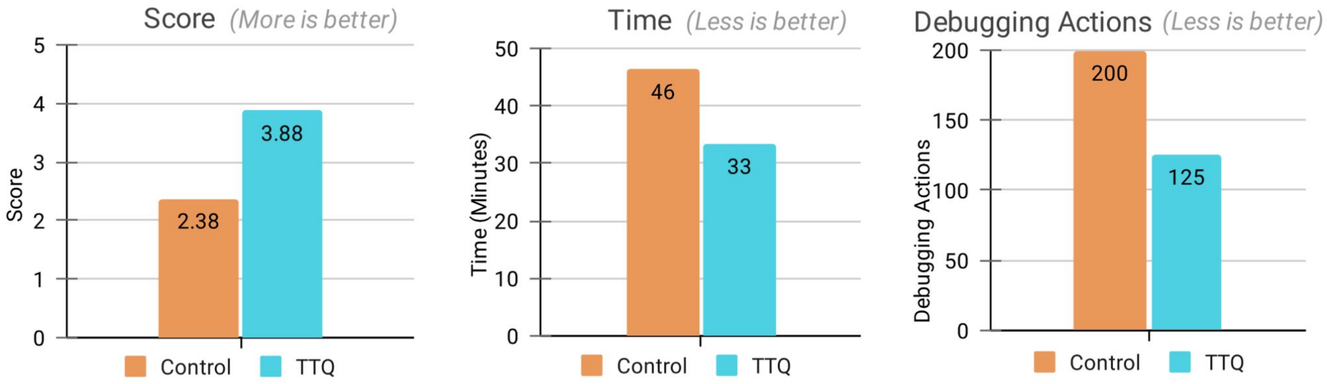
Participants Debugging Actions





Count of debugging actions of participants

Results Summary

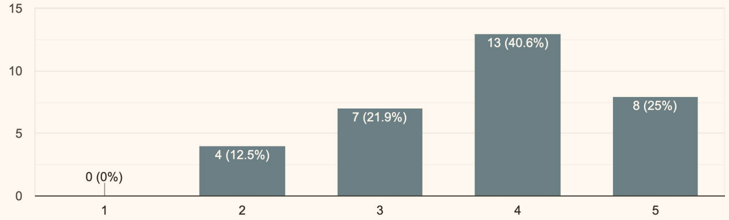


Qualitative Survey

Perceived tasks difficulty

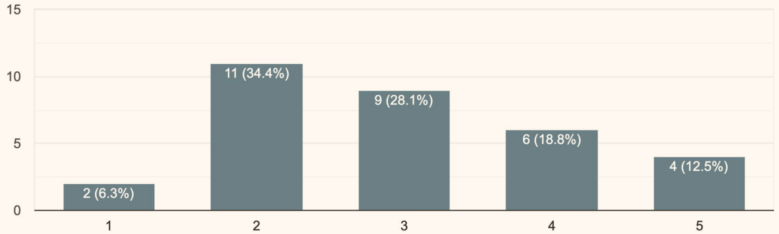
Confidence in answers

Control



Easy

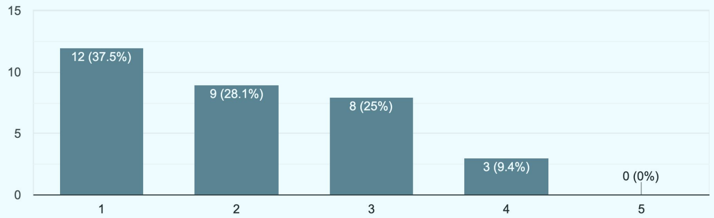
Difficult



Not sure at all

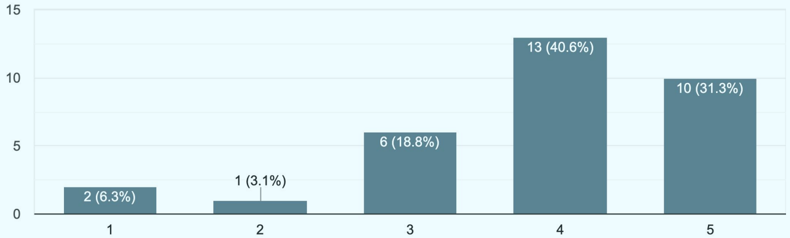
They are for sure the correct answers

TTQs



Easy

Difficult

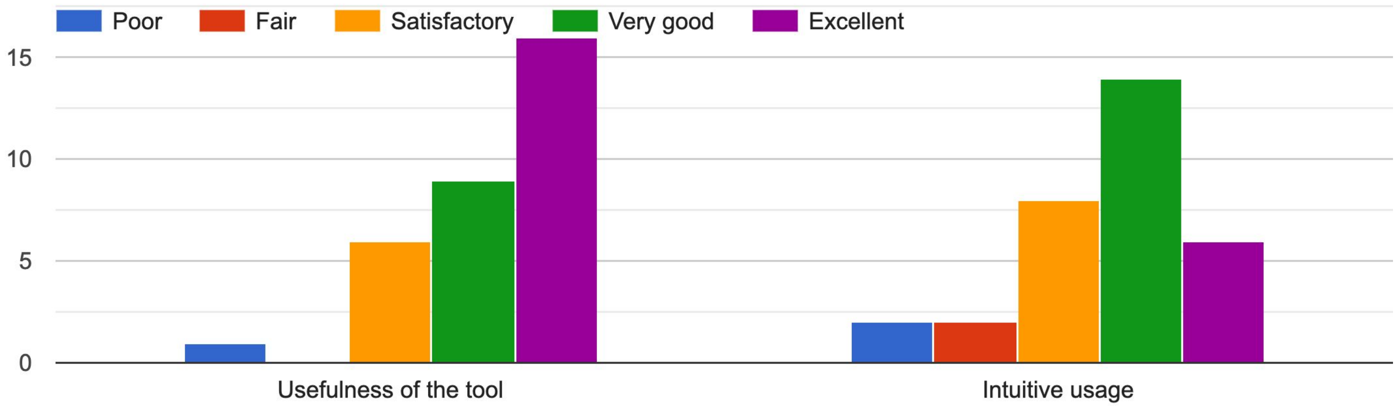


Not sure at all

They are for sure the correct answers

Qualitative Survey

SeekerDebugger with TTQs evaluation



Experiment Conclusion

- We can express general program comprehension questions as queries over programs executions.

- Results show that TTQs improve program exploration regarding developers’ efforts, time spent and precision, compared to standard debugging tools.

- Even with little instruction time for participants, the results were positive.

- Current TTQs don't cover the complete set of problems developers face during their debugging sessions.

Summary

- Different tools and methodologies for program understanding.
- Program exploration using interactive debuggers remains difficult and tedious.
- Proposed TTQs to improve exploration and comprehension.
- Controlled experiment to evaluate our solution.
- With TTQs, developers perform program comprehension tasks more accurately, faster, and with less effort than with standard debugging tools.
- We will continue Time-Traveling Queries research:
  - New relevant queries
  - Improving Time-Traveling Debugger limitations.