ONWARD!
PAPERS

Time-Awareness in Object Exploration Tools

Toward In Situ Omniscient Debugging

Christoph Thiede, Marcel Taeumel, and Robert Hirschfeld
Software Architecture Group
Hasso Plattner Institute, Potsdam, Germany
https://hpi.de/swa

Image: "File:Time-Travel.jpg." Wikimedia Commons. 2023-10-14. CC BY-SA 4.0.. URL: https://commons.wikimedia.org/w/index.php?title=File:Time-Travel.jpg&oldid=732421938 Scaled up with bigjpg.com AI Image Enlarger.

Motivation

- Programmers explore running systems by their space and time
  - What does the state of this object look like?
  - What is this object doing?
  - Why does this variable have changed?
- Different tools and workflows for different questions
  - Spatial questions: inspection tools
  - Temporal questions: debuggers, omniscient debuggers
- Omniscient debuggers require upfront commitment
  - Will I need to go back in history?

Research Question

How can we design tools for program exploration that support both space-related and time-related questions and thus combine historical information about program execution (and object evolution) in a single workflow?

Contributions

Approach

Practical universal tracing mode for exploratory programming systems

Concept

The spacetime exploration model

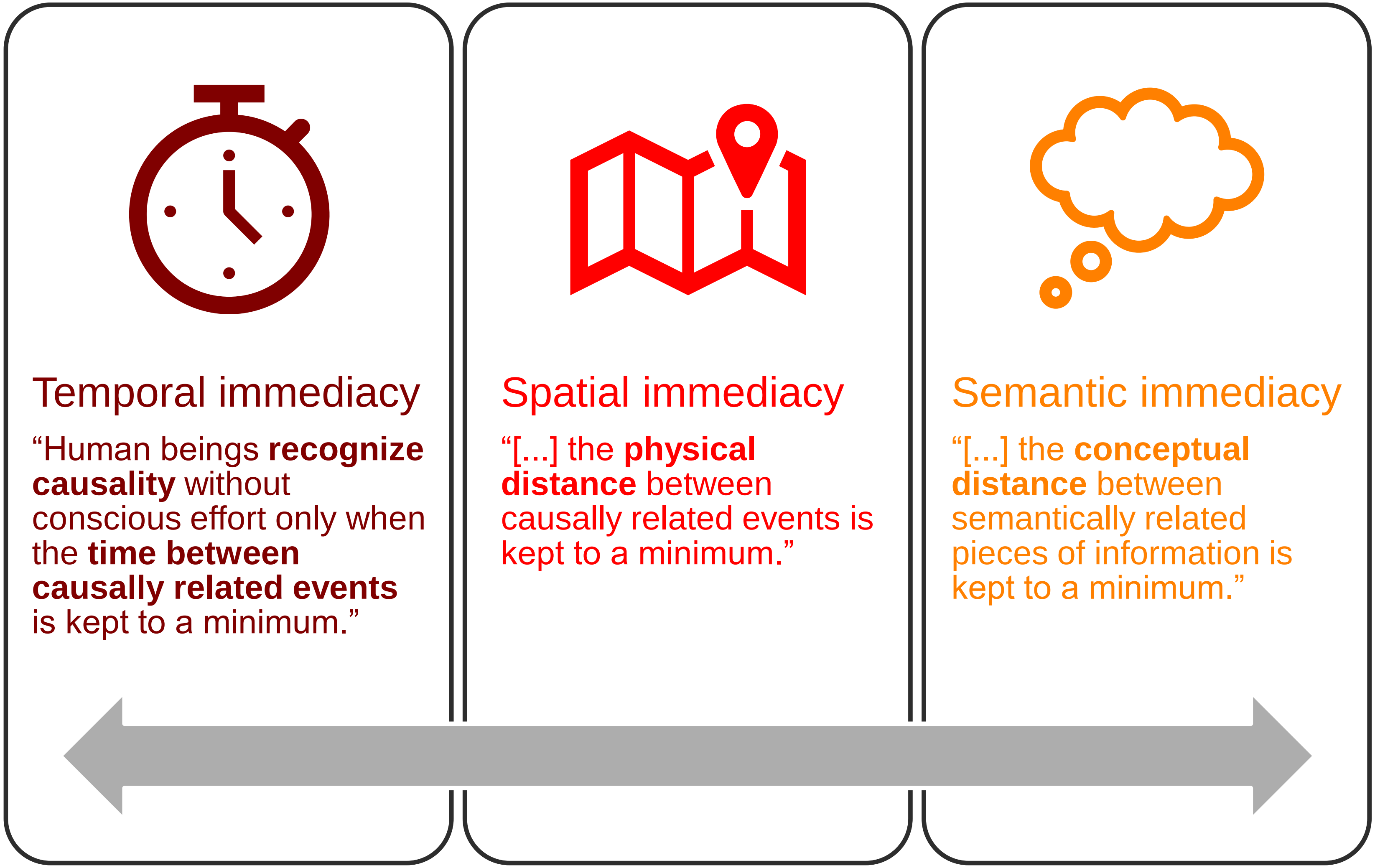
Application

The spacetime inspector prototype for Squeak/Smalltalk

Background: Program Exploration

- Exploratory programming [KER2017, REI2019, SAN1998, TAE2022]
  - Working on a software system where the system or the requirements are not fully understood
  - Iteratively acquire knowledge and prototype solutions
  - Theory building: ask questions, run experiments, repeat
- Aspects of questions
  - System space (state): meaning and structure of data
  - System time (behavior): inner functioning, construction and manipulation of data
- Object-oriented programming systems [GOL1983, THI2023b]
  - Everything is an object (with identity, state, and behavior)
  - Systems of objects
  - Programmers can access and manipulate all objects

Background: The Experience of Immediacy



Temporal immediacy

“Human beings **recognize causality** without conscious effort only when the **time between causally related events** is kept to a minimum.”

Spatial immediacy

“[...] the **physical distance** between causally related events is kept to a minimum.”

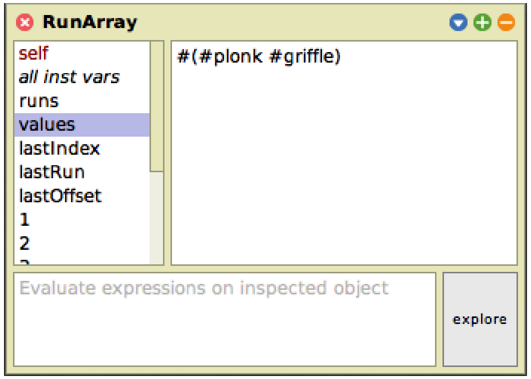
Semantic immediacy

“[...] the **conceptual distance** between semantically related pieces of information is kept to a minimum.”

[UNG1997]

Program Exploration Tools

Object inspection

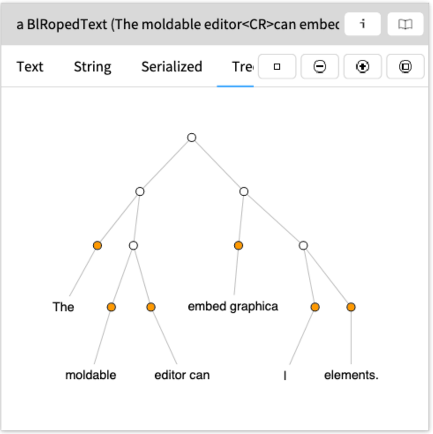

RunArray

self
all inst vars
runs
values
lastIndex
lastRun
lastOffset
1
2

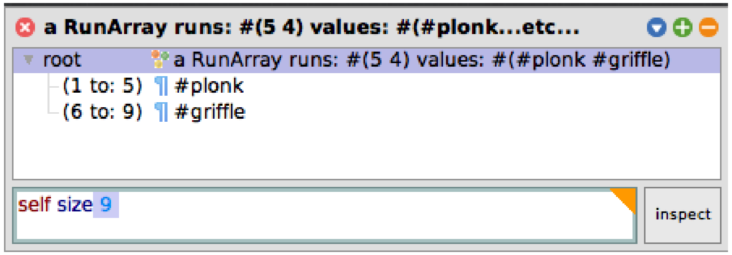
#(#plonk #griffle)

Evaluate expressions on inspected object

[Squeak]


a BlRopedText (The moldable editor<CR>can embec

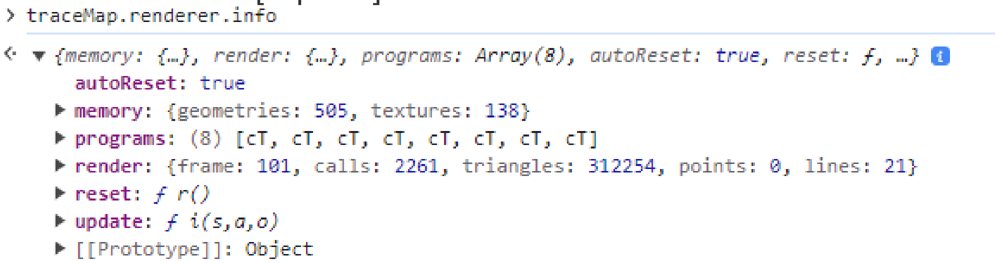
[Glamorous Toolkit]


a RunArray runs: #(5 4) values: #(#plonk...etc...

```
root          a RunArray runs: #(5 4) values: #(#plonk #griffle)
  (1 to: 5)  ¶ #plonk
  (6 to: 9)  ¶ #griffle
```

```
self size 9
```

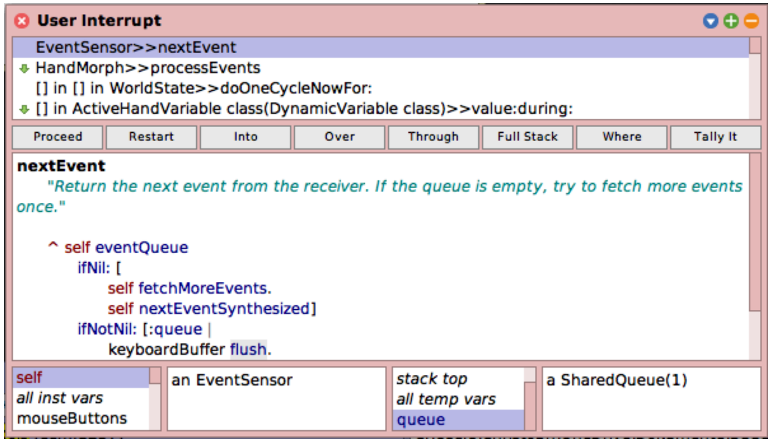
[Squeak]



```
> traceMap.renderer.info
<· ▼ {memory: {…}, render: {…}, programs: Array(8), autoReset: true, reset: ƒ, …}
      autoReset: true
    ▶ memory: {geometries: 505, textures: 138}
    ▶ programs: (8) [cT, cT, cT, cT, cT, cT, cT, cT]
    ▶ render: {frame: 101, calls: 2261, triangles: 312254, points: 0, lines: 21}
    ▶ reset: ƒ r()
    ▶ update: ƒ i(s,a,o)
    ▶ [[Prototype]]: Object
```

[Chrome Dev Tools]

Process debugging


User Interrupt

EventSensor>>nextEvent
HandMorph>>processEvents
[] in [] in WorldState>>doOneCycleNowFor:
[] in ActiveHandVariable class(DynamicVariable class)>>value:during:

```
nextEvent
	"Return the next event from the receiver. If the queue is empty, try to fetch more events once."

	^ self eventQueue
		ifNil: [
			self fetchMoreEvents.
			self nextEventSynthesized]
		ifNotNil: [:queue |
			keyboardBuffer flush.
```

self
all inst vars
mouseButtons

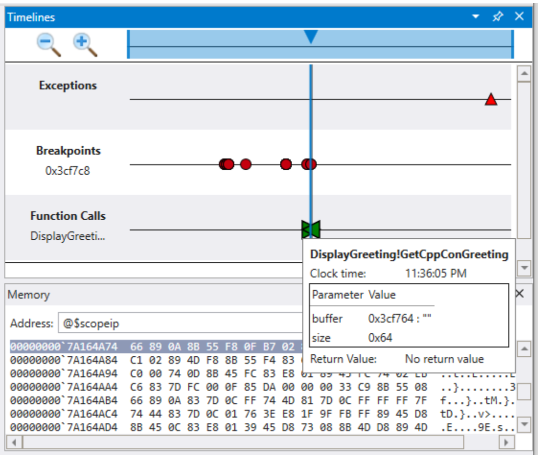
an EventSensor

stack top
all temp vars
queue

a SharedQueue(1)

[Squeak]

Omniscient debugging


Timelines

Exceptions

Breakpoints
0x3cf7c8

Function Calls
DisplayGreeti...

DisplayGreeting!GetCppConGreeting
Clock time: 11:36:05 PM

Parameter	Value
buffer	0x3cf764 : ""
size	0x64

Return Value: No return value

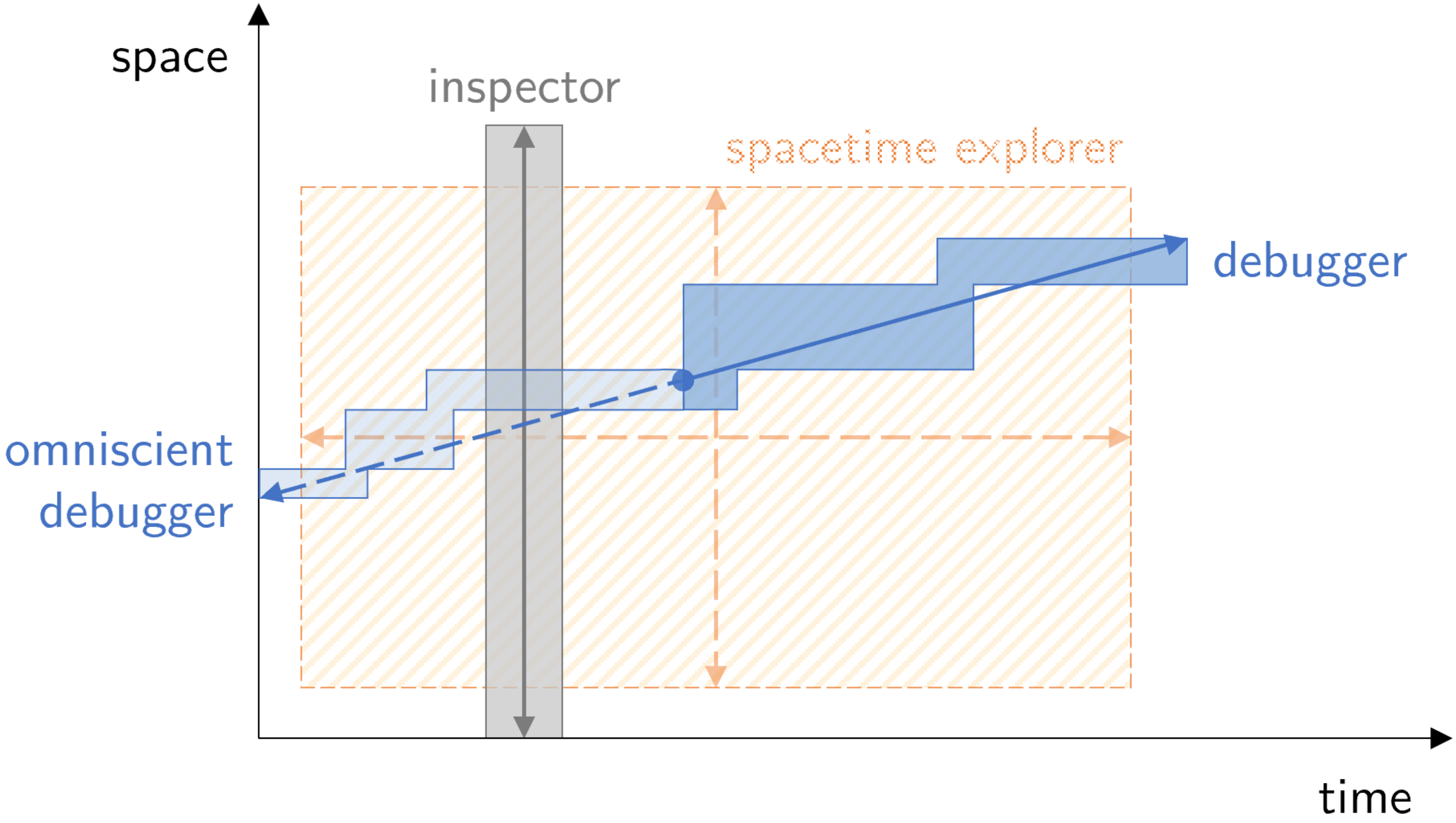
Memory

Address: @$scopeip

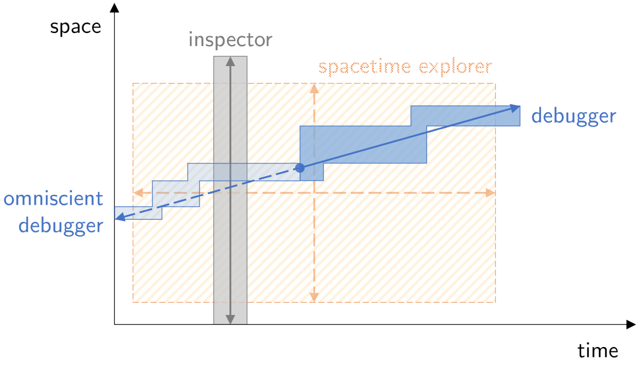
```
00000000`7A164A74  66 89 0A 8B 55 F8 0F B7 02
00000000`7A164A84  C1 02 89 4D F8 8B 55 F4 83
00000000`7A164A94  C0 00 74 0D 8B 45 FC 83 E8
00000000`7A164AA4  C6 83 7D FC 00 0F 85 DA 00 00 00 33 C9 8B 55 08  ..}........3
00000000`7A164AB4  66 89 0A 83 7D 0C FF 74 4D 81 7D 0C FF FF FF 7F  f...}..tM.}.
00000000`7A164AC4  74 44 83 7D 0C 01 76 3E E8 1F 9F FB FF 89 45 D8  tD.}..v>....
00000000`7A164AD4  8B 45 0C 83 E8 01 39 45 D8 73 08 8B 4D D8 89 4D  .E....9E.s..
```

[WinDbg]

Toward Immediacy Across Space and Time



Toward Immediacy Across Space and Time



Temporal immediacy

Spatial immediacy

Semantic immediacy

Universal tracing mode

Spacetime exploration model

Universal Tracing Mode

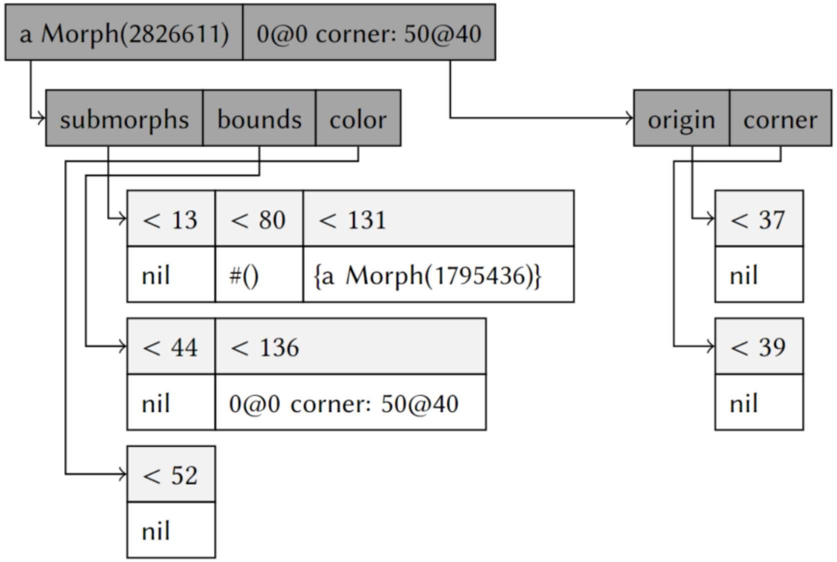
- Program tracing involves significant overheads of runtime/memory consumption
- Strategies:

EFFICIENT TRACE MODEL

EXPLICIT EXPLORATORY INTERFACES

PROGRAM REPRODUCTION

Universal Tracing Mode: Efficient Trace Model



- Incremental historic memory
  - Detect fine-grained side effects in bytecode and store previous values
  - Reduced memory consumption
  - Efficient read/append access

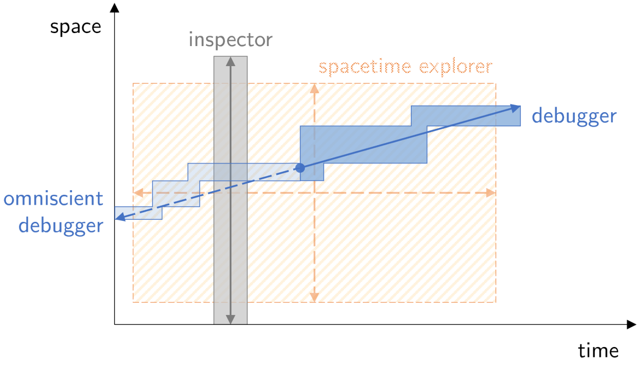
Universal Tracing Mode: Explicit Exploratory Interfaces

- Approach: only trace relevant behavior
  - GUI: DO trace user interactions, DO NOT trace rendering
  - Enterprise: DO trace business logic, DO NOT trace ORM
  - Unit tests: DO trace test case, DO NOT trace error reporting
- Define system boundaries for enabling/disabling program tracer
- Examples:
  - MVC/MVVM: model accesses
  - exploratory programming systems: custom expression evaluation, debugger invocations, direct manipulations

Universal Tracing Mode: Program Reproduction

- Approach: re-run program on demand to collect information [PER2010]
  - Requires reproducible entry point and deterministic behavior
- Reproducible entry points
  - Log invocations of explicit exploratory interfaces
  - Exploit existing log sources (e.g., changes files, command histories, database logs, …) [THI2023b, chap. 7, sec. 4, BIN2022]
- Deterministic behavior
  - Often cannot be guaranteed
  - Use heuristics to detect deviations and ask programmers
  - Prioritize upfront tracing

Toward Immediacy Across Space and Time



Temporal immediacy

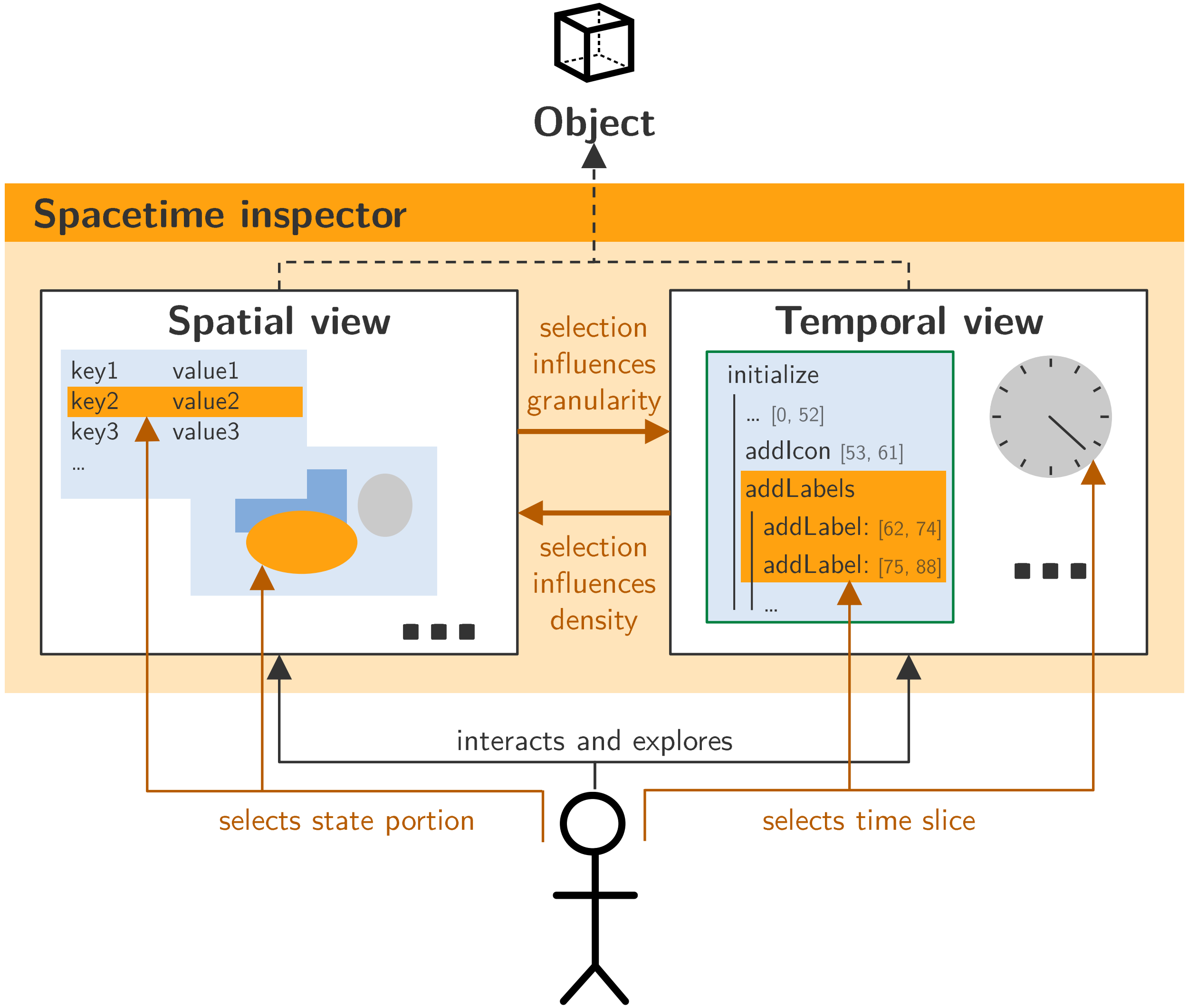
Spatial immediacy

Semantic immediacy

Universal tracing mode

Spacetime exploration model

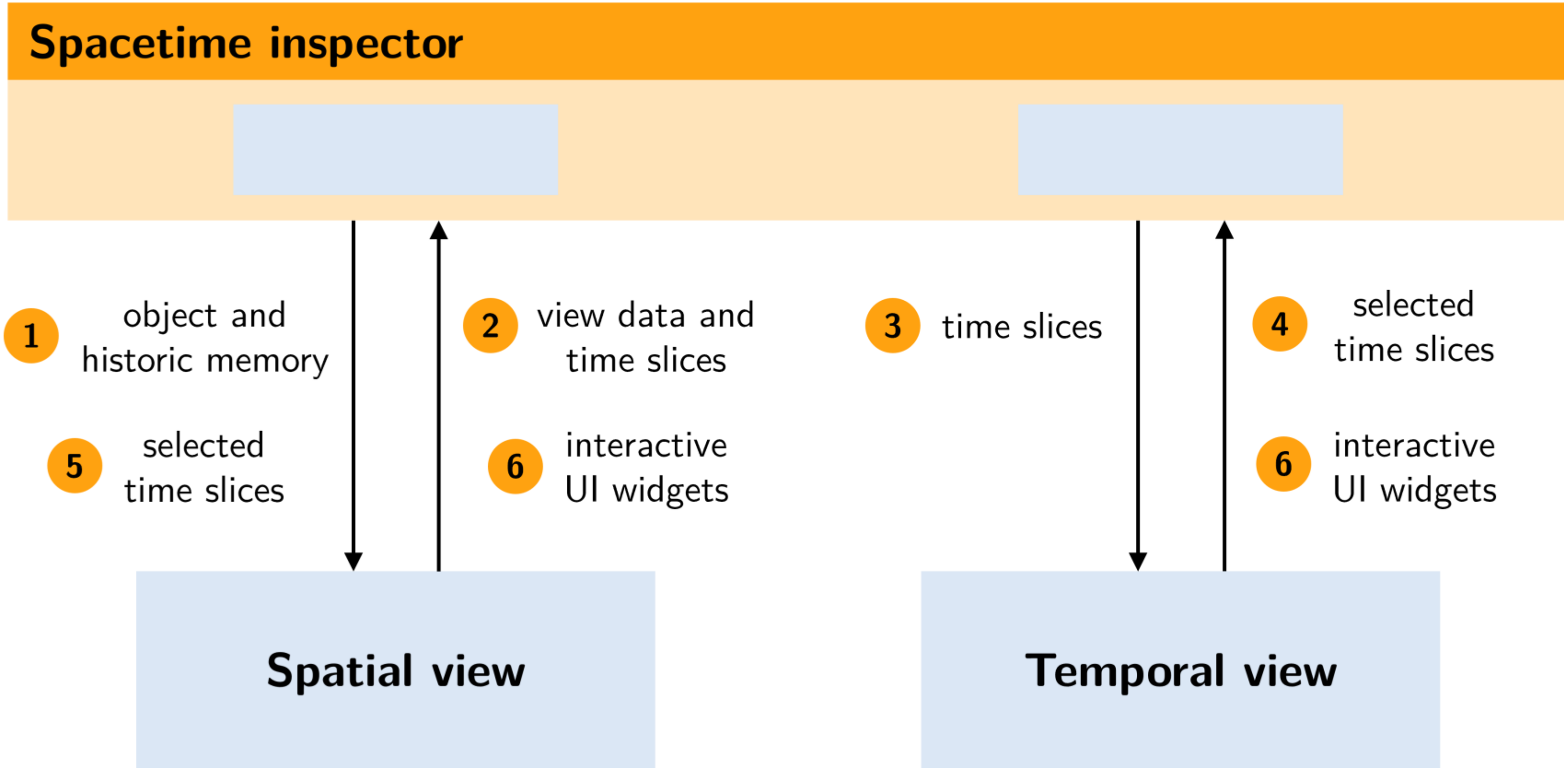
Spacetime Exploration



Spacetime Inspector: Demo

RunArray: a sparse collection object in Squeak

Spacetime Inspector: Implementation

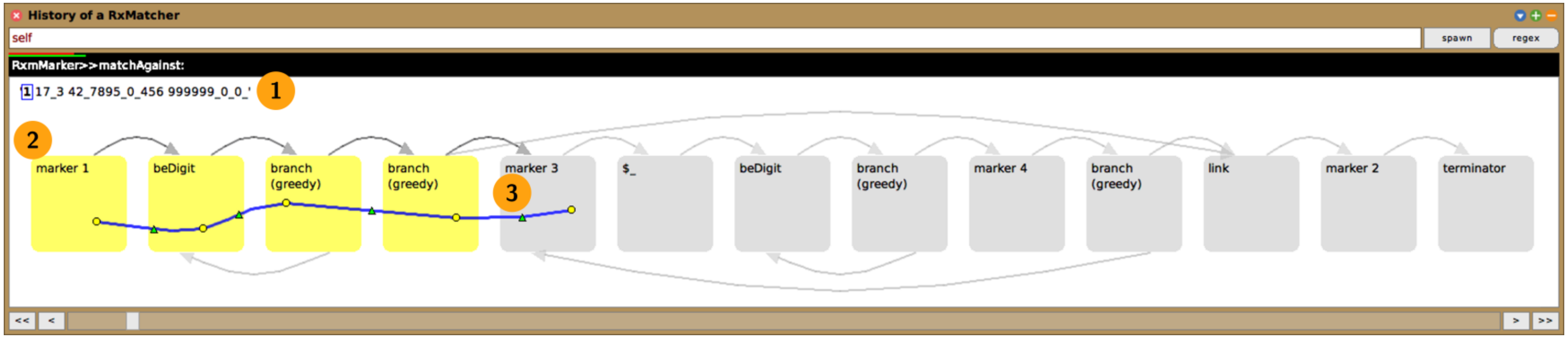


[THI2023a]

Spacetime Inspector: Demo #2

How does Squeak’s regular expression matcher work?

Spacetime Inspector: Demo #2

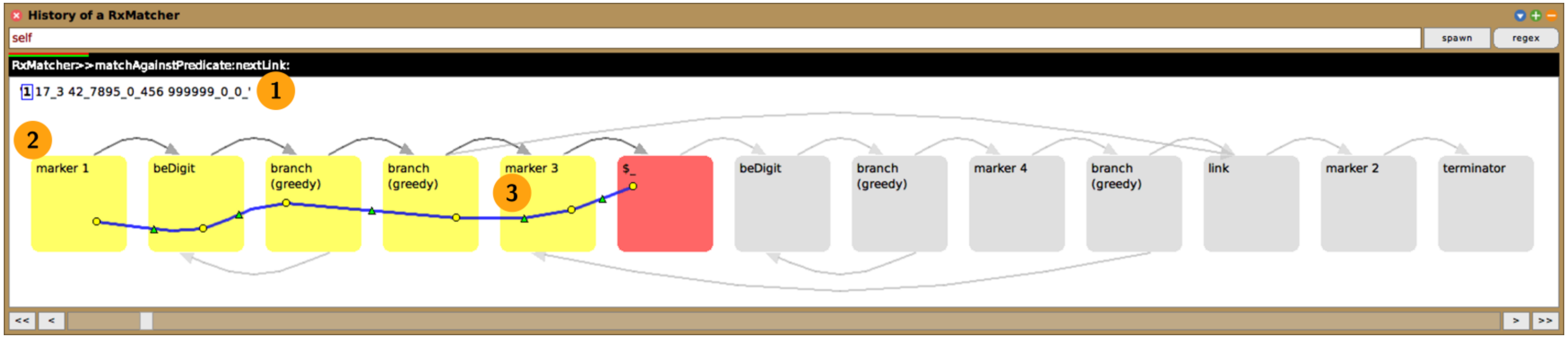


History of a RxMatcher

self

RxmMarker>>matchAgainst:

'1 17_3 42_7895_0_456 999999_0_0_'

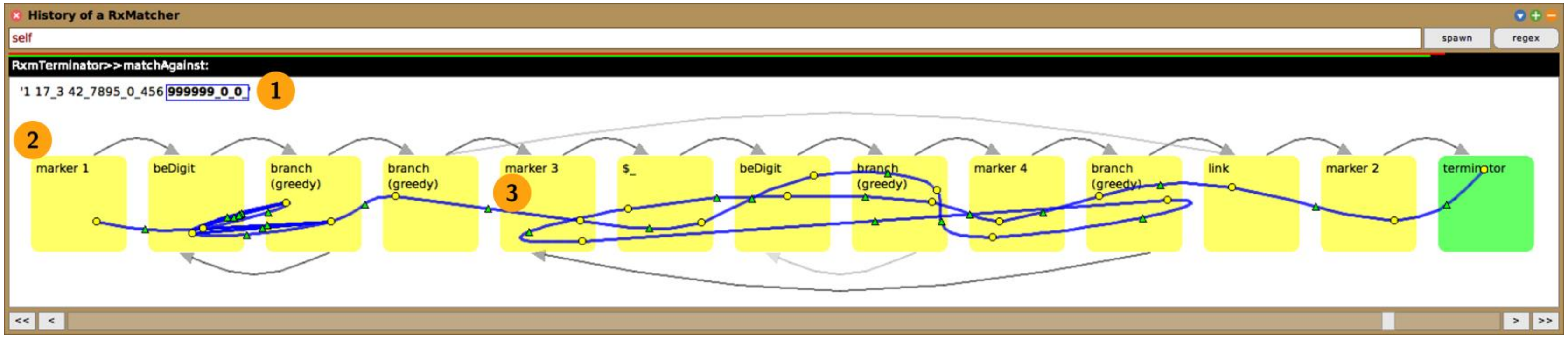


History of a RxMatcher

self

RxMatcher>>matchAgainstPredicate:nextLink:

'1 17_3 42_7895_0_456 999999_0_0_'



History of a RxMatcher

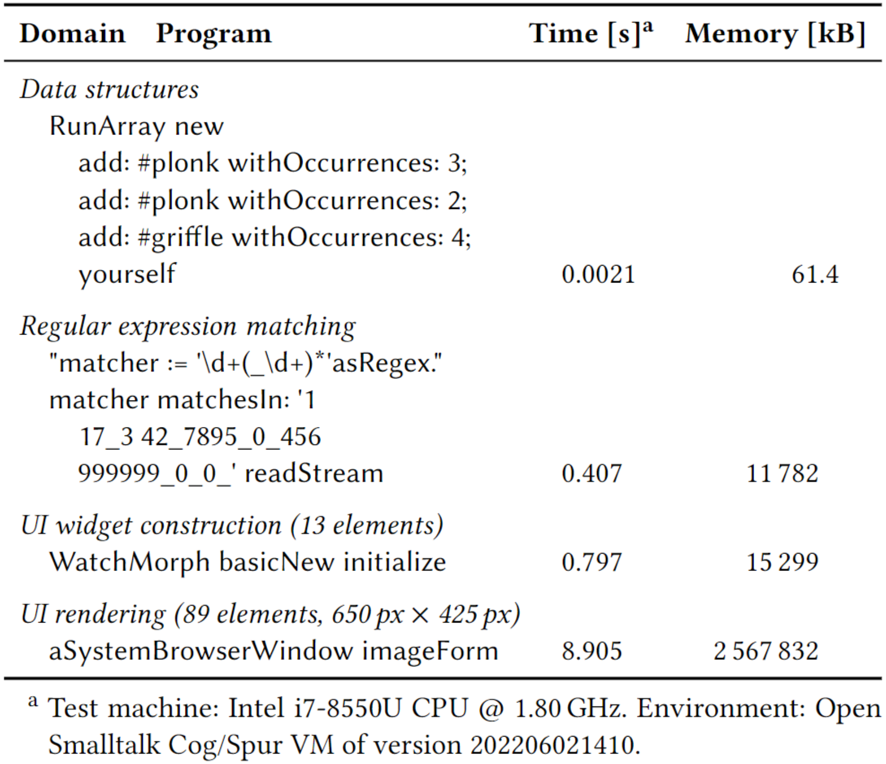
self

RxmTerminator>>matchAgainst:

'1 17_3 42_7895_0_456 999999_0_0_'

Evaluation of Performance

Evaluation of Performance: Universal Tracing Mode



Domain	Program	Time [s][a]	Memory [kB]
Data structures			

	RunArray new
	add: #plonk withOccurrences: 3;
	add: #plonk withOccurrences: 2;
	add: #griffle withOccurrences: 4;
	yourself
		0.0021	61.4

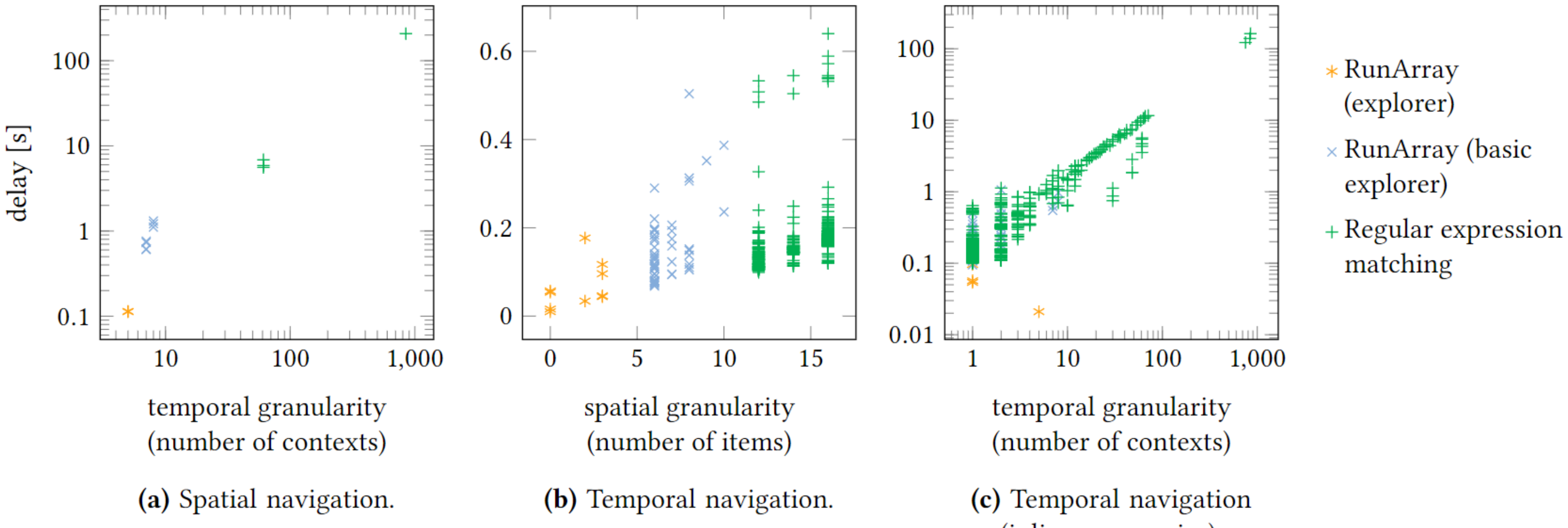
Regular expression matching			
	"matcher := '\d+(_\d+)*'asRegex." matcher matchesIn: '1 17_3 42_7895_0_456 999999_0_0_' readStream	0.407	11 782
UI widget construction (13 elements)			
	WatchMorph basicNew initialize	0.797	15 299
UI rendering (89 elements, 650 px × 425 px)			
	aSystemBrowserWindow imageForm	8.905	2 567 832

[a] Test machine: Intel i7-8550U CPU @ 1.80 GHz. Environment: Open Smalltalk Cog/Spur VM of version 202206021410.

- Naïve implementation: runtime overhead up to 1M%
- Incremental historic memory: low memory footprint
- But: responsive up to medium-sized workloads

[SHN2005]

Evaluation of Performance: Spacetime Exploration



(a) Spatial navigation.

(b) Temporal navigation.

(c) Temporal navigation (inline summaries).

- Temporal navigation: responsive for all use cases
- Spatial navigation and inline summaries (optional feature!): temporal granularity matters
- Domain-specific optimizations are possible and significant

[SHN2005]

Discussion

Discussion: Programming Experience

- Streamlined model for program exploration
  - Answer questions that relate to both the space and time of a system within a consolidated tool
  - Smaller gulf of execution
  - Reflect on hypotheses using a higher-level, more natural meta-vocabulary
- Fewer interruptions and inconsistencies: higher experience of immediacy
- Rich contextual information: e.g., understand structure of state by its evolution

Discussion: Tool Building

- How can spacetime exploration frameworks assist tool builders in developing better tools in shorter time?

+

- Reuse existing, spacetime-agnostic tools
- Combine several state-centric and time-centric views

−

- Intrinsic complexity of tool building/often limited offer of existing tools
- Manual adjustments required for providing aggregated summaries or scaling views for larger spacetime workloads

Future Work

- How can we integrate dataflow into the spacetime exploration model? [KO2008]
- Can we use spacetime exploration as an overarching concept for all kinds of exploratory programming activities? [TAE2020]
  - Dynamic composition of views
  - Custom means for filtering space and time
  - Symbolic debuggers as configurable spacetime views

Conclusion

Approach

Practical universal tracing mode
for exploratory
programming systems

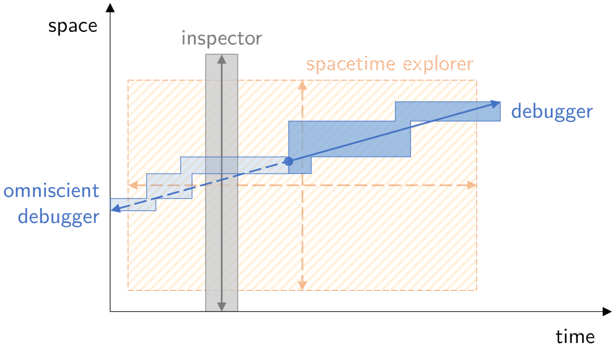
EFFICIENT TRACE MODEL

EXPLICIT EXPLORATORY INTERFACES

PROGRAM REPRODUCTION

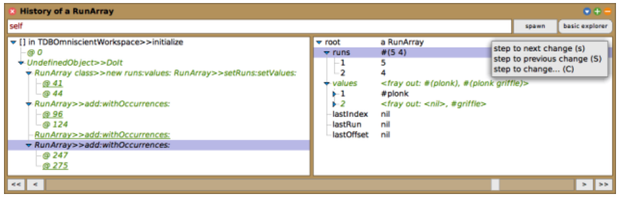
Concept

The spacetime exploration model

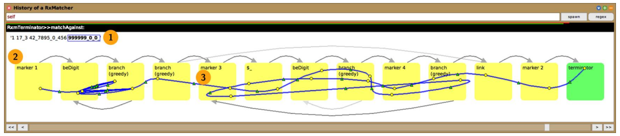


Application

The spacetime inspector prototype
for Squeak/Smalltalk



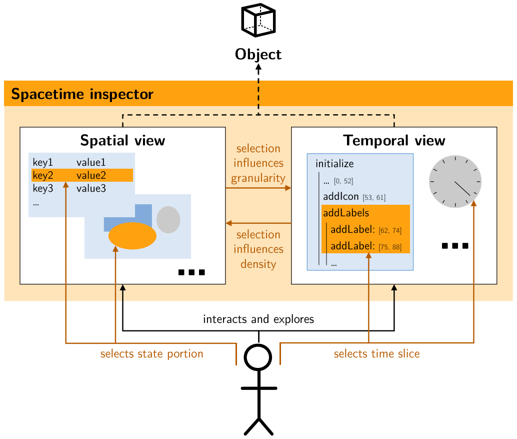
```
self

▼ [] in TDBOmniscientWorkspace>>initialize
   @ 0
  ▼ UndefinedObject>>DoIt
    ▼ RunArray class>>new runs:values: RunArray>>setRuns:setValues:
        @ 41
        @ 44
    ▼ RunArray>>add:withOccurrences:
        @ 96
        @ 124
      RunArray>>add:withOccurrences:
    ▼ RunArray>>add:withOccurrences:
        @ 247
        @ 275

▼ root        a RunArray
  ▼ runs      #(5 4)
      1       5
      2       4
  ▼ values    <fray out: #(plonk), #(plonk griffle)>
    ▸ 1       #plonk
    ▸ 2       <fray out: <nil>, #griffle>
    lastIndex   nil
    lastRun     nil
    lastOffset  nil

step to next change (s)
step to previous change (S)
step to change... (C)
```



```
self
RxmTerminator>>matchAgainst:
'1 17_3 42_7895_0_456 999999_0_0
```

Performance:

- ➢ Low memory footprint
- ➢ Tracing responsive up to modest workloads
- ➢ Spatial navigation requires optimizations for modest workloads



Programming experience:

- ➢ Streamlined program exploration model
- ➢ Increased immediacy
- ➢ Rich contextual information
- ➢ Framework to combine existing domain-specific views

Further Information

- Christoph Thiede, Marcel Taeumel, and Robert Hirschfeld. 2023. **Time-Awareness in Object Exploration Tools: Toward In Situ Omniscient Debugging.** In *Proceedings of the 2023 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (Onward! ’23), October 25–27, 2023, Cascais, Portugal. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3622758.3622892
- Artifacts:
  - https://github.com/hpi-swa-lab/squeak-tracedebugger
  - https://github.com/LinqLover/Regex-Tools

Literature

- [BIN2022] L. Thomas van Binsbergen, Mauricio Verano Merino, Pierre Jeanjean, Tijs van der Storm, Benoit Combemale, and Olivier Barais. 2020. A Principled Approach to REPL Interpreters. In *Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (Virtual, USA) *(Onward! 2020)*. Association for Computing Machinery, New York, NY, USA,84–100. https://doi.org/10.1145/3426428.3426917
- [GOL1983] Adele Goldberg and David Robson. 1983. *Smalltalk-80: The Language and Its Implementation.* Addison-Wesley Longman Publishing Co., Inc., USA. https://dl.acm.org/doi/10.5555/273
- [KER2017] Mary Beth Kery and Brad A. Myers. 2017. Exploring Exploratory Programming. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 25–29. https://doi.org/10.1109/VLHCC.2017.8103446
- [KO2008] Amy J. Ko and Brad A. Myers. 2008. Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior. In *Proceedings of the 30th International Conference on Software Engineering* (Leipzig, Germany) *(ICSE '08)*. Association for Computing Machinery, New York, NY, USA, 301–310. https://doi.org/10.1145/1368088.1368130
- [PER2010] Michael Perscheid, Bastian Steinert, Robert Hirschfeld, Felix Geller, and Michael Haupt. 2010. Immediacy through Interactivity: Online Analysis of Run-Time Behavior. In *2010 17th Working Conference on Reverse Engineering*. 77–86. https://doi.org/10.1109/WCRE.2010.17
- [REI2019] Patrick Rein, Stefan Ramson, Jens Lincke, Robert Hirschfeld, and Tobias Pape. 2019. Exploratory and Live, Programming and Coding: A Literature Study Comparing Perspectives on Liveness. *The Art, Science, and Engineering of Programming* 3, 1 (07 2019), 33 pages. https://doi.org/10.22152/programming-journal.org/2019/3/1
- [SAN1998] David W. Sandberg. 1988. Smalltalk and Exploratory Programming. SIGPLAN Not. 23, 10 (1988), 85–92. https://doi.org/10.1145/51607.51614
- [SHN2005] Ben Shneiderman and Catherine Plaisant. 2005. *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (4th ed.). Pearson Education, India. http://seu1.org/files/level5/IT201/Book%20-%20Ben%20Shneiderman-Designing%20the%20User%20Interface-4th%20Edition.pdf
- [TAE2020] Marcel Taeumel. 2020. *Data-Driven Tool Construction in Exploratory Programming Environments.* Ph. D. Dissertation. University of Potsdam, Digital Engineering Faculty, Hasso Plattner Institute. https://doi.org/10.25932/publishup-44428
- [TAE2022] Marcel Taeumel, Jens Lincke, Patrick Rein, and Robert Hirschfeld. 2022.A Pattern Language of an Exploratory Programming Workspace. In *Design Thinking Research: Achieving Real Innovation*, Christoph Meinel and Larry Leifer (Eds.). Springer International Publishing, Cham, 111–145. https://doi.org/10.1007/978-3-031-09297-8_7
- [THI2023a] Christoph Thiede, Marcel Taeumel, and Robert Hirschfeld. 2023. Object-Centric Time-Travel Debugging: Exploring Traces of Objects. In *Companion Proceedings of the 7th International Conference on the Art, Science, and Engineering of Programming* (Tokyo, Japan) *(<Programming> '23)*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3594671.3594678
- [THI2023b] Christoph Thiede and Patrick Rein. 2023. Squeak by Example. Vol. 6.0. Lulu. https://www.lulu.com/shop/patrick-rein-and-christoph-thiede/squeak-by-example-60/paperback/product-8vr2j2.html ISBN 978-1-4476-2948-1.
- [UNG1997] David Ungar, Henry Lieberman, and Christopher Fry. 1997. Debugging and the Experience of Immediacy. *Commun. ACM* 40, 4 (apr 1997), 38–43. https://doi.org/10.1145/248448.248457

Literature (ctd.)

- Squeak: https://squeak.org
- Chrome Dev Tools: https://developer.chrome.com/docs/devtools/
- Glamorous Toolkit: https://gtoolkit.com/
- WinDbg: https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/