# Object-Centric Debugging
for Pharo 8

Steven Costiou
RMoD
Inria Lille - Nord Europe

About me

- **Researcher at RMoD team (Inria) since January 2019**
  - Debugging, object-centric debugging
- **Before that:**
  - Software engineer in the industry for 6 years
  - PhD at the university of Brest (France)
  - Software engineer for the Pharo Consortium
    October 2018 - December 2018

Object-centric debugging

What is object-centric debugging?

What is object-centric debugging?

- Debugging operation or technique which **applies only to one specific instance of a class**:
    - **An object-centric breakpoint**, e.g. halt when the state of that object is written
    - **Object-centric behavior**, e.g. a method only available for that one object

Demo: magic with cards

Object-centric breakpoints API

Send messages to objects to install breakpoint

Target object

```
deck := Deck newWith54Cards.
deck haltOnCall.
deck haltOnCall: #randomCard.
```

halt on all method calls

Halt on calls to #randomCard

Object-centric breakpoints API

Halt on state access:

Target object

```
deck := Deck newWith54Cards.
deck haltOnWrite.
deck haltOnReadTo: #cards.
```

halt when writing in any of the object's instance variables

Halt each time the #cards instance variable is read

Why object-centric debugging?

Why object-centric debugging?

- **Debugging one object among many!**
    - Collections
    - Graphical objects
    - Events
    - Etc.

Example 1

Breakpoints in collections

Example 2

Breakpoints in graphical objects

Example 3

Debugging the debugger

Additional API

Object-centric behavior

Object-centric debugging behavior API

Object-centric debugging behavior:

Target object

```
deck := Deck new.

deck acquire: 'logDeck
                self logCr'.

deck replace: #cards
      with: 'cards
               self logDeck.
               ^cards'
```

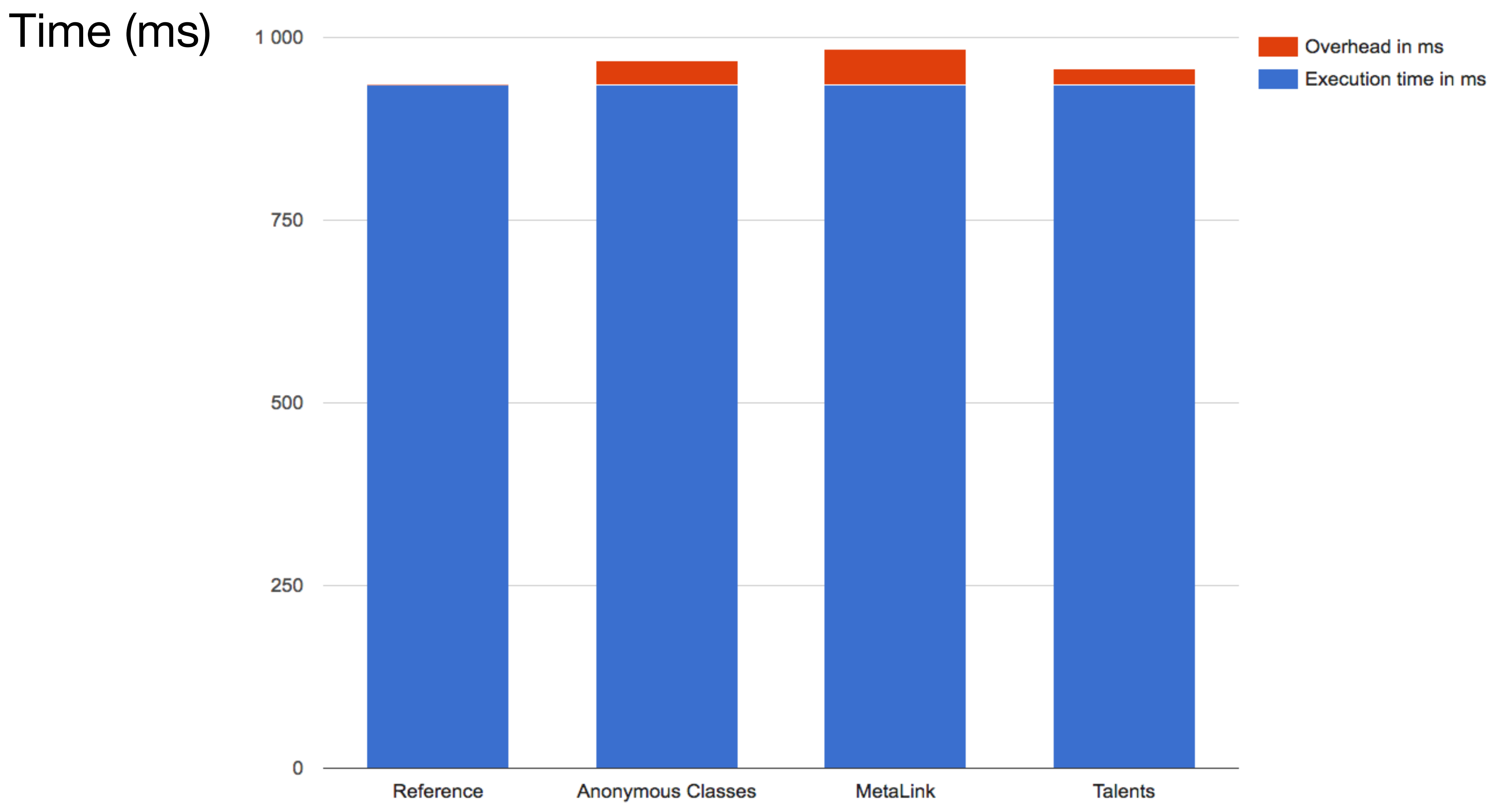
The object acquires a new method

The original method #cards of the object is replaced by a new version

Performance overhead evaluation

(work in progress)

Performance overhead evaluation (WIP)

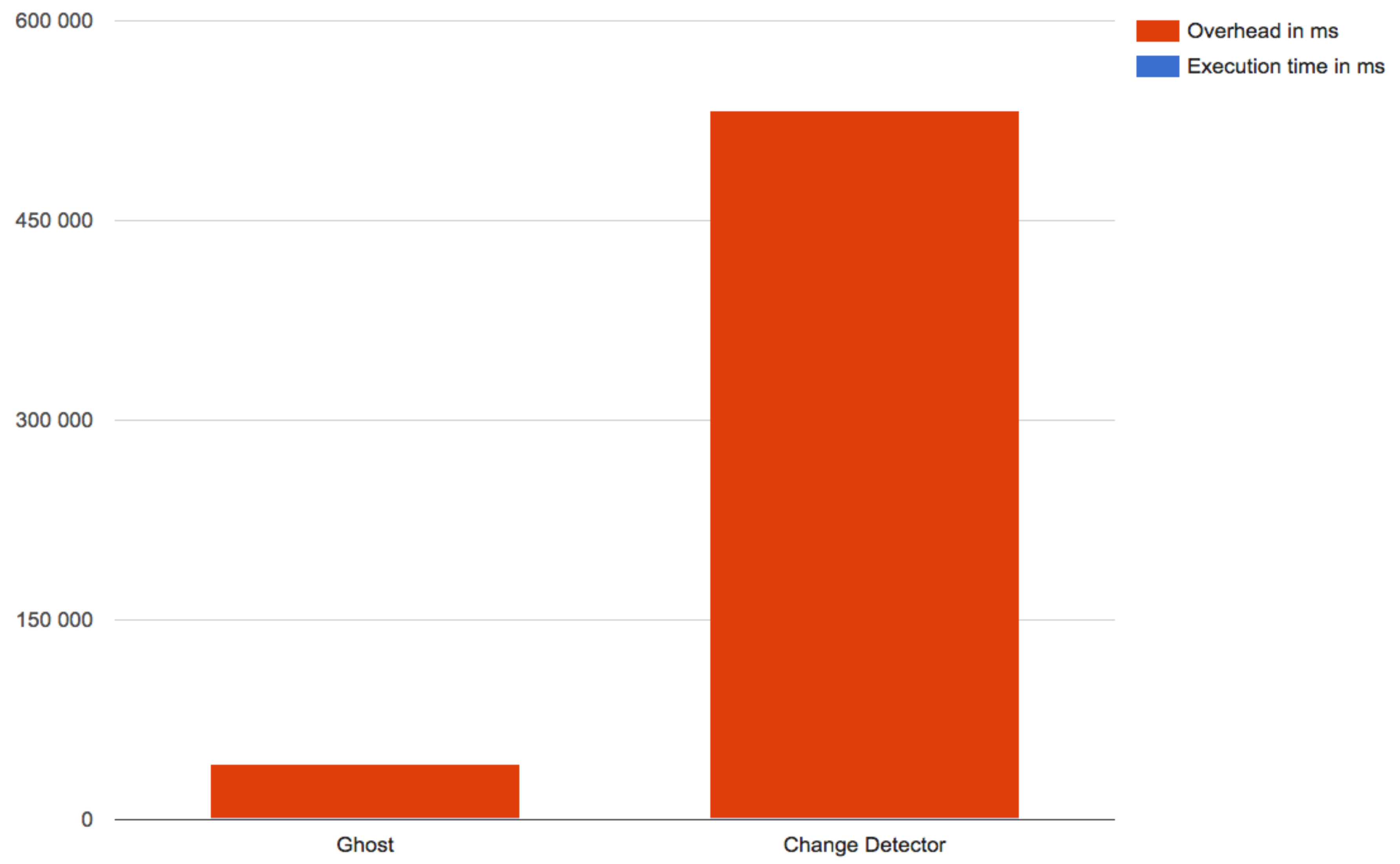


Implementation backends used for:

- Halt on call to specific method
- Halt on state access

Performance overhead evaluation (WIP)

Time (ms)



Implementation backends used for:

- Halt on call (all methods)
- Halt on state access (experimental)

TODO list and technical issues

TODO list

- Implement similar features to traditional breakpoints: **conditions, halt once…**
- Tool support for object-centric behavior
- Test and integrate to Pharo 8

Remaining technical issues

- The implementation relies on **anonymous subclasses**:
  - Different users: Talents, Reflectivity, object-centric breakpoints…
  - These users are **not compatible between them**

    => We need to **common and compatible** way of using anonymous subclasses
- Visibility of instrumentation: **when to see it (or not)?**

Thank you!

Booklet soon available!

Object-Centric Instrumentation with Pharo

Steven Costiou

Square Bracket tutorials

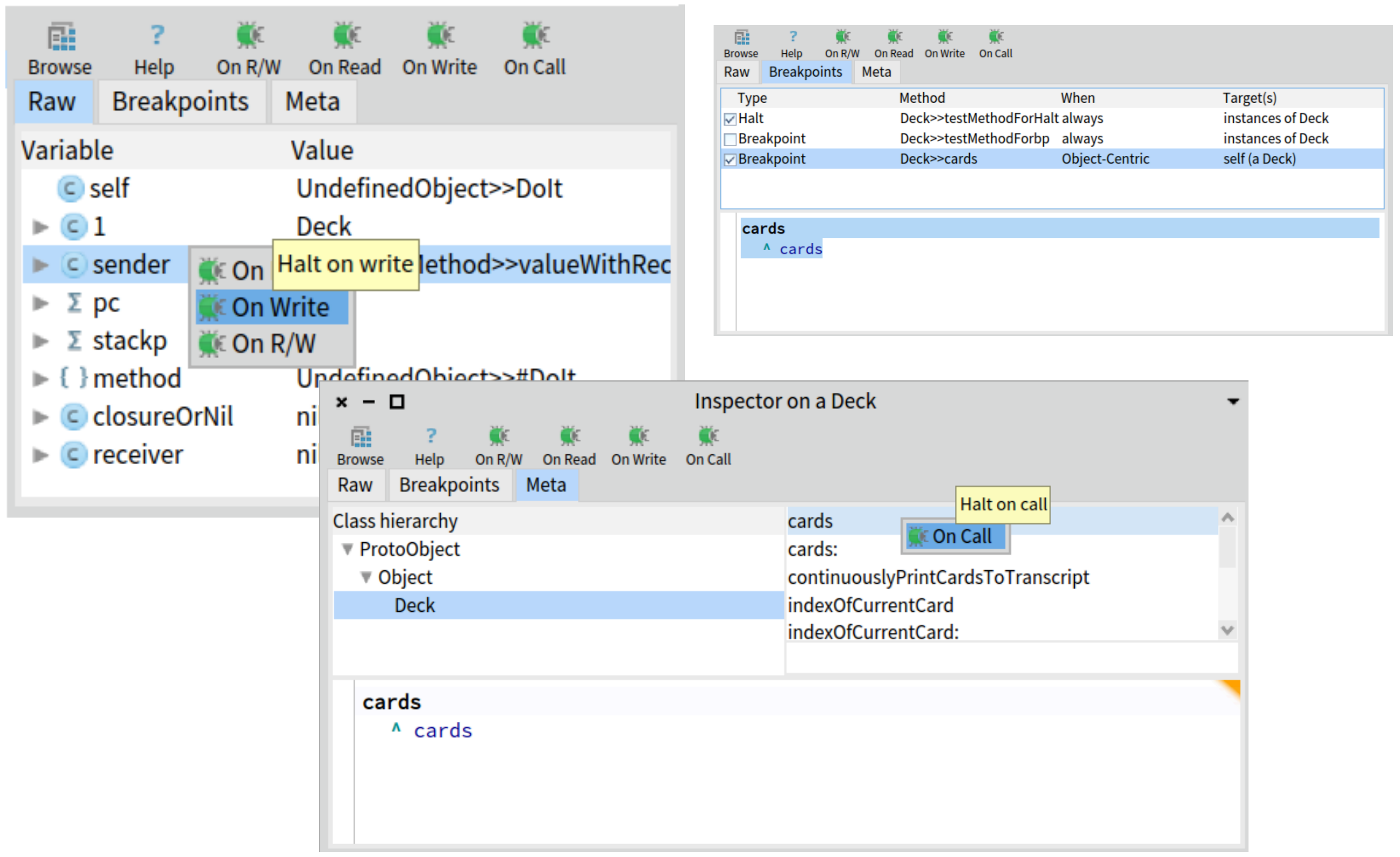
Object-Centric Debugging API

- **haltOnCall**
- **haltOnCall:** #selector
- **haltOnceOnCall:** #selector
- **haltOnCallWhen:** condition

- **haltOnWriteTo:** #instVarName
- **haltOnRead:** #instVarName
- **haltOnWrite**
- **haltOnRead**

- **acquire:** aMethodSourceCode
- **replace:** aSelector
  with: aMethodSourceCode

Object-Centric Debugging tools



Variable	Value
self	UndefinedObject>>DoIt
1	Deck
sender	…lethod>>valueWithRec
pc	
stackp	
method	UndefinedObject>>#DoIt
closureOrNil	ni
receiver	ni

Type	Method	When	Target(s)
✓ Halt	Deck>>testMethodForHalt	always	instances of Deck
☐ Breakpoint	Deck>>testMethodForbp	always	instances of Deck
✓ Breakpoint	Deck>>cards	Object-Centric	self (a Deck)

```
cards
    ^ cards
```

Class hierarchy
- ProtoObject
  - Object
    - Deck

cards
cards:
continuouslyPrintCardsToTranscript
indexOfCurrentCard
indexOfCurrentCard:

```
cards
    ^ cards
```