Object-Centric Debugging: a preview

Steven Costiou
RMoD
Inria Lille - Nord Europe

steven.costiou@inria.fr

2019
Debugging Department

Great debugging sorcerers. Beware.

We solve all problems.


NOTE
The door is often closed because of the Pharo Argentinian Baroque Lyric Orchestra in the office on the left. Unless mentioned otherwise on the door, feel free to enter the magical world of debugging. Pay before results. No refunds. No java. Now you can stop reading and get back to work.
Part I
Object-Centric Debugging
Demo
What is object-centric debugging?
Object-centric debugging

- Debugging operations at the level of objects
  - Only target objects are affected
- Examples:
  - A breakpoint active for one object only
  - A method available for one object only
Why object-centric debugging?
Why Object-centric debugging?

- Debugging one object among many:
  - Collections (Hinkle, Jones, Johnson, 1993)
  - Events
  - Graphical objects
Object-Centric Features (preview)
Object-centric breakpoints

- Break when a message is received
  - `haltOnCall` => on every method call
  - `haltOnCall: #selector` => for given selector only
  - `haltOnNextCall` => on next method call
  - `haltOnceOnCall: #selector` => only once for given selector
  - `haltOnCallWhen: condition` => if condition is met
Object-centric breakpoints

• Break on state access
  • `haltOnWriteTo: #instVarName` => when `instVarName` is written
  • `haltOnRead: #instVarName` => when `instVarName` is read
  • `haltOnWrite` => when any instance variable is written
  • `haltOnRead` => when any instance variable is read
Object-centric behavior

• Object-centric methods
  • `compile: sourceCode` => compiles and add new methods
  • `uses: aTrait` => acquires behavior from Trait
  • `acquire: aCompiledMethod` => acquire the method
Object-centric debugging, how is it implemented?
Implementation

Object-centric breakpoints

Object-centric behavior
Implementation

Object-centric breakpoints

Reflectivity

Proxies

Object-centric behavior
Implementation

Object-centric breakpoints

Reflectivity

Proxies

Talents

Object-centric behavior
Implementation

Object-centric breakpoints

Reflectivity

Proxies

Talents

Anonymous Classes

Object-centric behavior
Implementation

- Object-centric breakpoints
- Object-centric behavior
- Reflectivity
- Proxies
- Talents
- Anonymous Classes

...
Implementation goal

Object-centric breakpoints
Object-centric behavior
Object-centric …

Object-centric layer

Reflectivity
Proxies
Talents
Anonymous Classes
Current problems

- Implementation is mixing up different techniques without any clear interaction model
- Requires to migrate the object to an anonymous subclass
- Installation of object-centric instrumentation is not thread-safe
- Sometimes make tools unstable
- Obtaining objects to debug (but work has been done on that…)

20
Part II
Object-Centric Reverse Debugging
Demo
Reverse object-centric debugger

Bytecode execution

<table>
<thead>
<tr>
<th>heading</th>
<th>heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>anOthermethodCall</td>
<td></td>
</tr>
<tr>
<td>methodCall</td>
<td></td>
</tr>
</tbody>
</table>

- self method.
- self a ifTrue: [self b].
- self halt

<table>
<thead>
<tr>
<th>inspector</th>
<th>inspector</th>
</tr>
</thead>
</table>
Reverse object-centric debugger

Bytecode execution

<table>
<thead>
<tr>
<th>heading</th>
<th>anOthermethodCall</th>
<th>methodCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>heading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self method.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self halt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AST execution

<table>
<thead>
<tr>
<th>heading</th>
<th>anOthermethodCall</th>
<th>methodCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>heading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self method.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self halt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

State and execution transfer

inspector | inspector

inspector | inspector
Reverse object-centric debugger

Bytecode execution

<table>
<thead>
<tr>
<th>heading</th>
<th>anOthermethodCall</th>
<th>methodCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>heading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self method.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self a ifTrue: [self b].</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self halt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inspector</td>
<td>inspector</td>
<td></td>
</tr>
</tbody>
</table>

AST execution

<table>
<thead>
<tr>
<th>heading</th>
<th>anOthermethodCall</th>
<th>methodCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>heading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self method.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self a ifTrue: [self b].</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self halt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inspector</td>
<td>inspector</td>
<td></td>
</tr>
</tbody>
</table>

State and execution transfer

Isolation debugging
Reverse object-centric debugger

Bytecode execution

<table>
<thead>
<tr>
<th>heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>anOthermethodCall</td>
</tr>
<tr>
<td>methodCall</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>self method.</td>
</tr>
<tr>
<td>self a ifTrue: [self b].</td>
</tr>
<tr>
<td>self halt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inspector</th>
<th>inspector</th>
</tr>
</thead>
</table>

AST execution

<table>
<thead>
<tr>
<th>heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>anOthermethodCall</td>
</tr>
<tr>
<td>methodCall</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>self method.</td>
</tr>
<tr>
<td>self a ifTrue: [self b].</td>
</tr>
<tr>
<td>self halt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inspector</th>
<th>inspector</th>
</tr>
</thead>
</table>

State and execution transfer

Isolation debugging

stepping
Reverse object-centric debugger
Reverse object-centric debugger
Reverse object-centric debugger

Bytecode execution

<table>
<thead>
<tr>
<th>heading</th>
<th>anOtherMethodCall</th>
<th>methodCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>heading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self method.</td>
<td>self a ifTrue: [self b].</td>
<td></td>
</tr>
<tr>
<td>self halt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AST execution

<table>
<thead>
<tr>
<th>heading</th>
<th>anOtherMethodCall</th>
<th>methodCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>heading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>self method.</td>
<td>self a ifTrue: [self b].</td>
<td></td>
</tr>
<tr>
<td>self halt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

State and execution transfer

Object-centric changes recording

Object-centric queries over the isolated execution

reverse stepping

Isolation debugging

stepping

someMethod
asNumber
new

someMethod

^self asFloat + 1

inspector inspector
Reverse object-centric debugger

<table>
<thead>
<tr>
<th>Bytecode execution</th>
<th>AST execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>heading</td>
<td>heading</td>
</tr>
<tr>
<td>anOtherMethodCall</td>
<td>anOtherMethodCall</td>
</tr>
<tr>
<td>methodCall</td>
<td>methodCall</td>
</tr>
<tr>
<td>heading</td>
<td>heading</td>
</tr>
<tr>
<td>self method.</td>
<td>self method.</td>
</tr>
<tr>
<td>self a ifTrue: [self b].</td>
<td>self a ifTrue: [self b].</td>
</tr>
<tr>
<td>self halt</td>
<td>self halt</td>
</tr>
<tr>
<td>inspector</td>
<td>inspector</td>
</tr>
<tr>
<td>inspector</td>
<td>inspector</td>
</tr>
</tbody>
</table>

State and execution transfer

- Reverse stepping
- Isolation debugging
- Stepping

Object-centric changes recording
Object-centric queries over the isolated execution
Object-centric execution control

```
someMethod
asNumber
new

someMethod
^self asFloat + 1

inspector  inspector
```
Reverse object-centric debugger

<table>
<thead>
<tr>
<th>Bytecode execution</th>
<th>AST execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>heading</td>
<td>heading</td>
</tr>
<tr>
<td>anOthermethodNameCall</td>
<td>anOthermethodNameCall</td>
</tr>
<tr>
<td>methodCall</td>
<td>methodCall</td>
</tr>
<tr>
<td>heading</td>
<td>heading</td>
</tr>
<tr>
<td>self method.</td>
<td>self method.</td>
</tr>
<tr>
<td>self a ifTrue: [self b].</td>
<td>self a ifTrue: [self b].</td>
</tr>
<tr>
<td>self halt</td>
<td>self halt</td>
</tr>
<tr>
<td>inspector</td>
<td>inspector</td>
</tr>
</tbody>
</table>

State and execution transfer

Object-centric changes recording

Object-centric queries over the isolated execution

Object-centric execution control

Isolation debugging

reverse stepping

stepping

someMethod

asNumber

new

someMethod

^self asFloat + 1

inspector | inspector
Reverse object-centric debugger

Bytecode execution

<table>
<thead>
<tr>
<th>heading</th>
<th>anOthermethodCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodCall</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>self method.</td>
</tr>
<tr>
<td>self a ifTrue: [self b].</td>
</tr>
<tr>
<td>self halt</td>
</tr>
</tbody>
</table>

| inspector | inspector |

AST execution

<table>
<thead>
<tr>
<th>heading</th>
<th>anOthermethodCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodCall</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>self method.</td>
</tr>
<tr>
<td>self a ifTrue: [self b].</td>
</tr>
<tr>
<td>self halt</td>
</tr>
</tbody>
</table>

| inspector | inspector |

State and execution transfer

Object-centric changes recording

Object-centric queries over the isolated execution

Object-centric execution control

State and execution transfer

someMethod asNumber
new

| someMethod |
| ^self asFloat + 1 |

| inspector | inspector |

someMethod asNumber
new

| someMethod |
| ^self asFloat + 1 |

| inspector | inspector |
Reverse object-centric debugger

Bytecode execution

AST execution

Object-centric changes recording
Object-centric queries over the isolated execution
Object-centric execution control

WIP
Reverse object-centric debugger

Who’s working on it?

- **AST interpreter** - Carolina Hernandez Phillips
- **Reverse-execution** - Vincent Aranega, Steven Costiou
- **Object-centric debugger** - Steven Costiou
- **Scriptable debugger** - Thomas Dupriez
Thanks! Questions?

Object-Centric Debugging

- haltOnCall
- haltOnCall: #selector
- haltOnNextCall
- haltOnceOnCall: #selector
- haltOnCallWhen: condition
- haltOnWriteTo: #instVarName
- haltOnRead: #instVarName
- haltOnWrite
- haltOnRead

- compile: sourceCode
- uses: aTrait
- acquire: aCompiledMethod

Object-Centric Reverse Debugging

Bytecode execution

AST execution

State and execution transfer

State and execution transfer

reverse stepping

Isolation debugging

stepping

Object-centric changes recording

Object-centric queries over the isolated execution

Object-centric execution control

heading
anOther:methodCall
methodCall

heading

self method.
sel a :true: [self b].
self halt

inspector inspector

Heading

someMethod
asNumber
new

someMethod
^self asFloat + 1

inspector inspector

heading
anOther:methodCall
methodCall

heading

self method.
sel a :true: [self b].
self halt

inspector inspector

heading
anOther:methodCall
methodCall

heading

self method.
sel a :true: [self b].
self halt

inspector inspector

heading
anOther:methodCall
methodCall

heading

self method.
sel a :true: [self b].
self halt

inspector inspector

heading
anOther:methodCall
methodCall

heading

self method.
sel a :true: [self b].
self halt

inspector inspector

heading
anOther:methodCall
methodCall

heading

self method.
sel a :true: [self b].
self halt

inspector inspector

heading
anOther:methodCall
methodCall

heading

self method.
sel a :true: [self b].
self halt

inspector inspector

heading
anOther:methodCall
methodCall

heading

self method.
sel a :true: [self b].
self halt

inspector inspector