Conditional Breakpoints on Top of a Reflective Layer

Thomas Dupriez
ENS Paris-Saclay - Software Language Lab, Vreije Universiteit Brussels
thomas.dupriez@ens-paris-saclay.fr

Abstract

Keywords Debugger, Tool, Stack, Breakpoint

Conditional breakpoints are a very common tool used by developers to acquire knowledge about the execution of a program. Often an IDE feature, the developers place them at strategic points of a program and specify conditions for them. When running the program, the execution will be stopped if it reaches a conditional breakpoint whose condition is fulfilled.

In this paper we focus on a way of implementing these conditional breakpoints: by using the reflective layer of the language itself.

1. Introduction

Contextual Breakpoints. What if a developer had the potential to create more powerful and adaptable breakpoints? What would such breakpoints look like? What kind of runtime information should they have access to to be both useful and efficient? their objects? How can we navigate the

2. Debugging Terminology

Debugging *i.e.,* In Pharo, a program entry point is the first context of a Process, usually executing the Block>>#newProcess method.

3. A Real Complex Debugging Scenario

[1].

```
$ ./pharo Pharo.image test "Pillar.*"
[...]
3182 run, 3182 passes, 0 failures, 0 errors.
```

[1] We setup a repository explaining in details the steps to reproduce the bug in https://github.com/guillep/pillar-bug

Checking the tests, the pillar developers observe that the bug happens in an apparently unrelated piece of code, the PREPubMenuJustHeaderTransformer>>actionOn: method.

```
PREPubMenuJustHeaderTransformer>>actionOn: anInput
  ^ (self class writers
    includes: anInput configuration outputType writerName)
    ifTrue: [ maxHeader := self maxHeaderOf: anInput input.
      super actionOn: anInput ]
    ifFalse: [ anInput ]
```

conditional breakpoints by attaching arbitrary boolean expressions to breakpoints, yielding breakpoints that only trigger if the attached expression evaluates to *true*.

```
self haltIf: [ CurrentExecutionEnvironment value
      isTestEnvironment ].
```

Figure 1: Breakpoint that only triggers while tests are run. In Pharo

References