

# Understanding how developers use the debugger

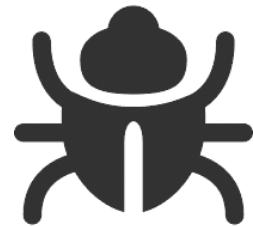
Presentation for SCG seminar

Roger Stebler

Supervisor: Andrei Chiș



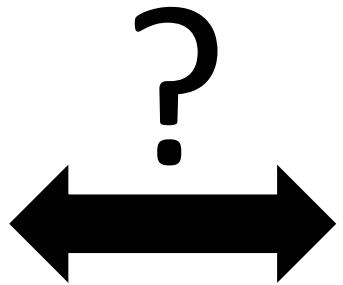
Applications are complex;  
debugging is difficult



One tool used for debugging is the  
debugger



How to improve the debugger?



Stack

```
SmallInteger      /
UndefinedObject   Dolt
OpalCompiler     evaluate
RubSmalltalkEditor evaluate:andDo:
RubSmalltalkEditor highlightEvaluateAndDo:
GLMMorphicPharoScriptRenderer(GLMMoi actOnHighlightAndEvaluate: [ textMorph textArea editor highlightEvaluateAndDo: ann action. textMorph shoutStyler style: textMorph|
```

Source

```
/ aNumber
"Primitive. This primitive (for /) divides the receiver by the argument
and returns the result if the division is exact. Fail if the result is not a
whole integer. Fail if the argument is 0 or is not a SmallInteger. Optional.
No Lookup. See Object documentation whatIsAPrimitive."
```

<primitive: 10>
aNumber isZero ifTrue: [^(ZeroDivide dividend: self) signal].
^(aNumber isMemberOf: SmallInteger)
 ifTrue: [(Fraction numerator: self denominator: aNumber) reduced]
 ifFalse: [super / aNumber]

Variables

Type	Variable	Value
implicit	self	1
parameter	aNumber	0
implicit	thisContext	SmallInteger>/
implicit	stack top	0

Understand how do developers use the debugger?

# Demo



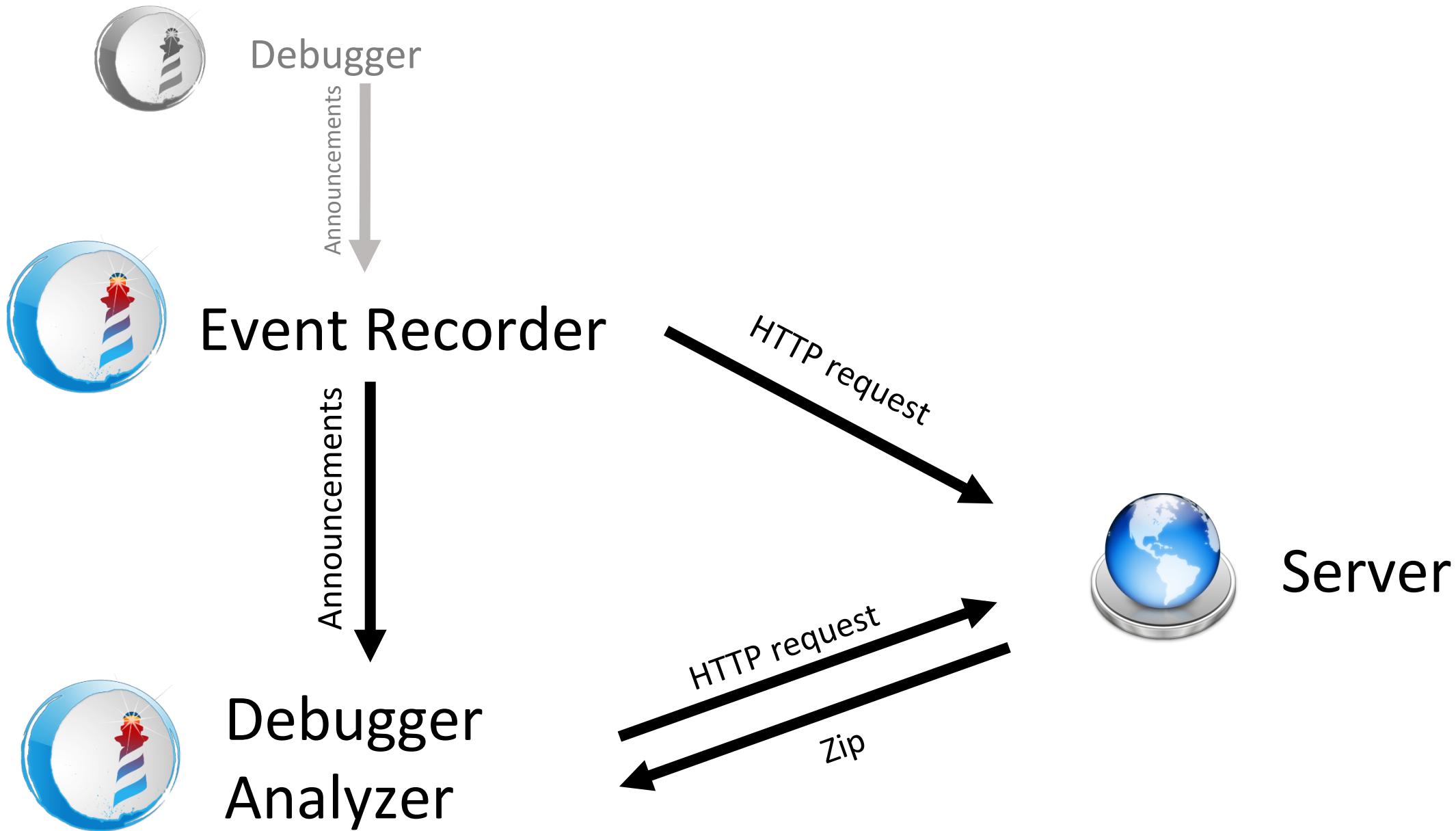
Debugger



Event Recorder



Server



# Summarization

# Statistical analysis

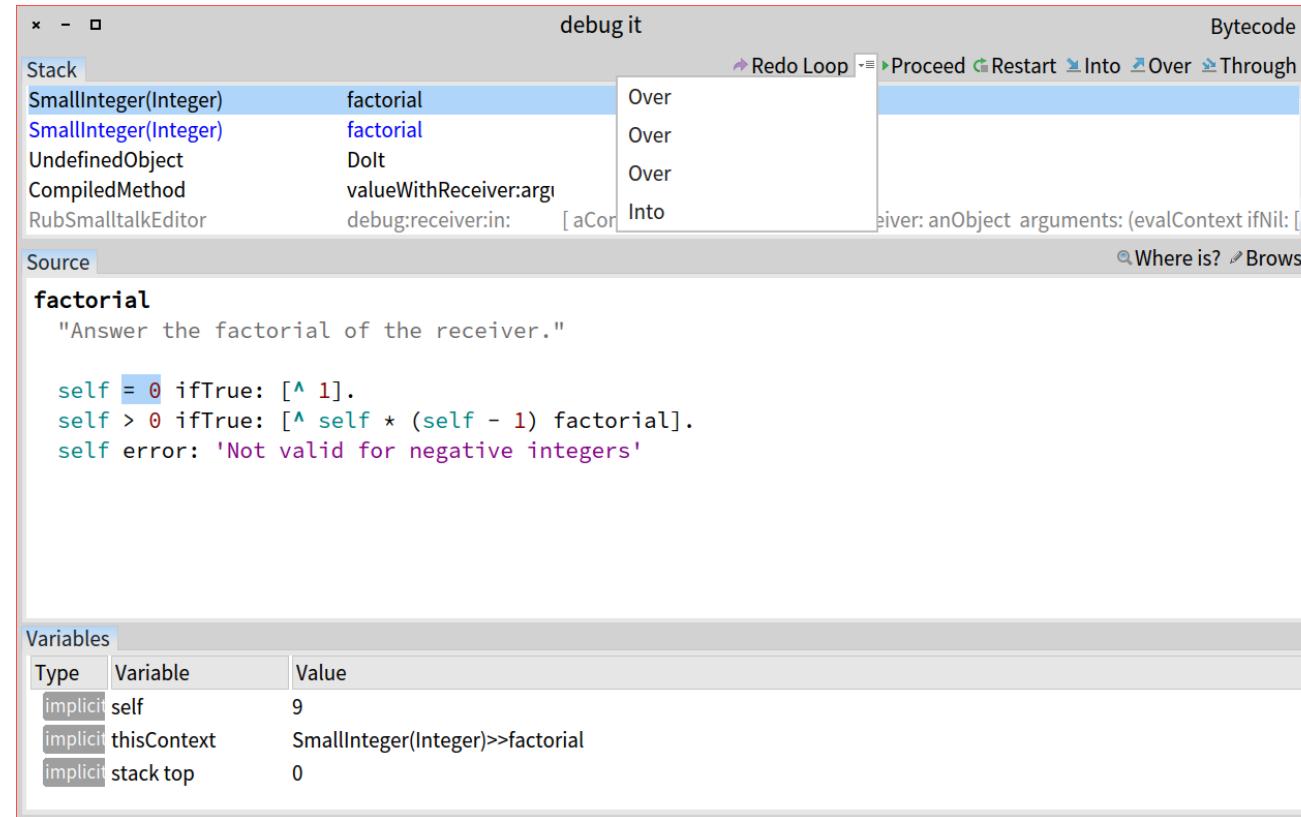


# In what context is a debugging action used the most?

Key	Value
'Senders of it'	3
'Proceed'	11
'Run to here'	2
'Fuel out Stack'	1
'Change to GTGenericStackDebugger'	21
'Peel to first like this'	2
'Copy'	23
'Close debugger'	145
'Step to next parser'	11
'Change to GTPPDebugger'	8
'PetitParser Debugger'	3
'Step to bytecode'	2
'Cancel'	1
'Browse'	3
'Debug it'	6
'Restart'	2
'Do it and go'	33

Give  
recommendations  
what to do next

# Are there some debugging patterns?



Automatization

# Redo Loop

Same position in the code earlier  
in the current session

The screenshot shows the Smalltalk debugger interface with the following components:

- Stack:** Displays the call stack with the following frames:
  - SmallInteger(Integer) factorial (highlighted)
  - SmallInteger(Integer) factorial
  - UndefinedObject Dolt
  - CompiledMethod valueWithReceiver:arg: [aCor]
  - RubSmalltalkEditor debug:receiver:in: [aCor]
- Source:** Shows the source code for the factorial method:

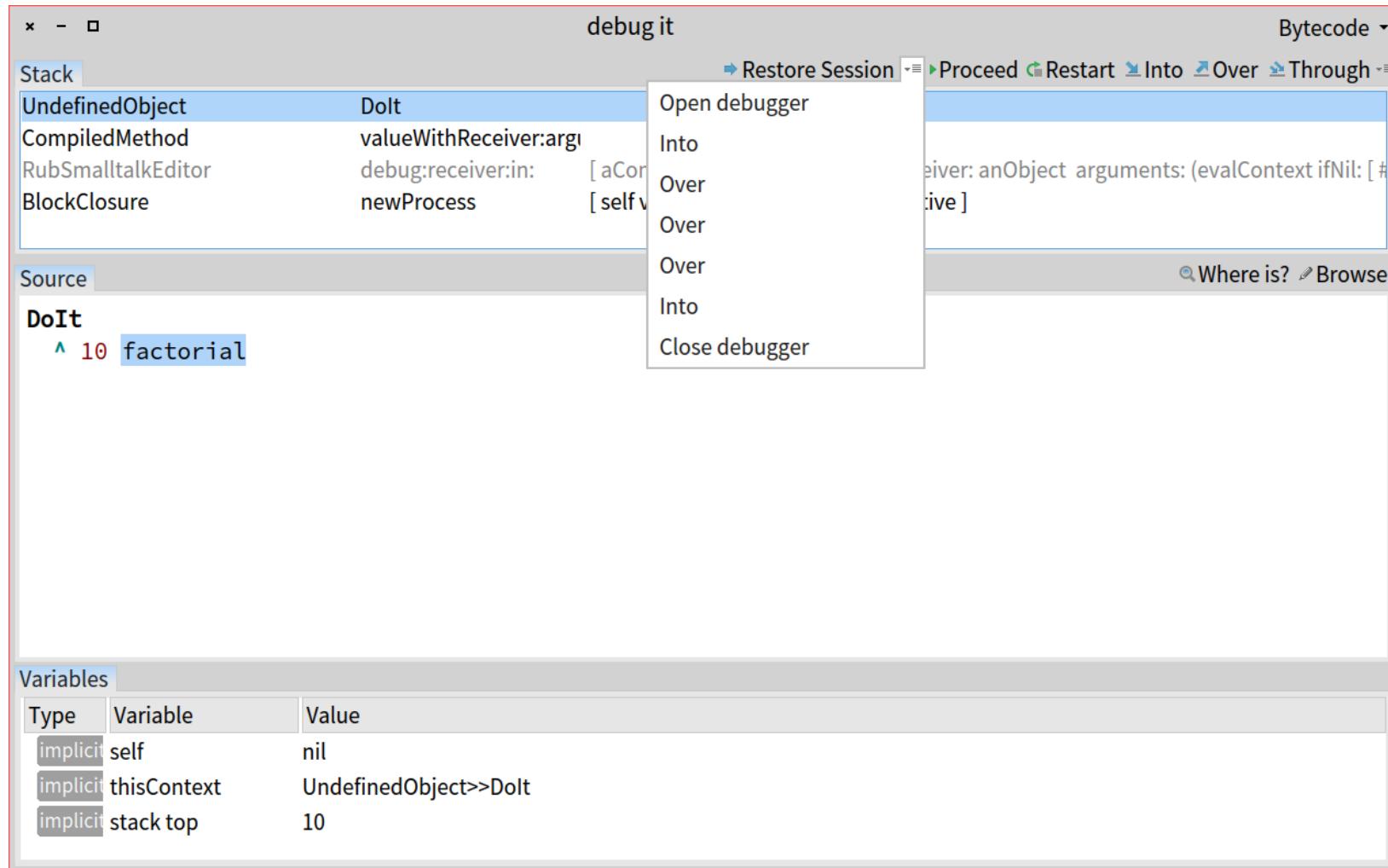
```
factorial
    "Answer the factorial of the receiver.

    self = 0 ifTrue: [^ 1].
    self > 0 ifTrue: [^ self * (self - 1) factorial].
    self error: 'Not valid for negative integers'
```
- Variables:** Shows the current variables:

Type	Variable	Value
implicit	self	9
implicit	thisContext	SmallInteger(Integer)>>factorial
implicit	stack top	0
- Toolbar:** Includes buttons for Redo Loop, Proceed, Restart, Into, Over, and Through.
- Bytecode:** A dropdown menu currently set to "Redo Loop".

Same context in a previous session

# Restore Session





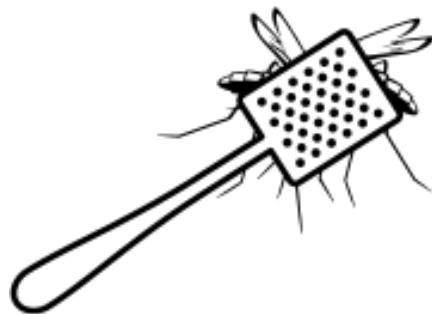
Find a categorization?



## Debugging Actions

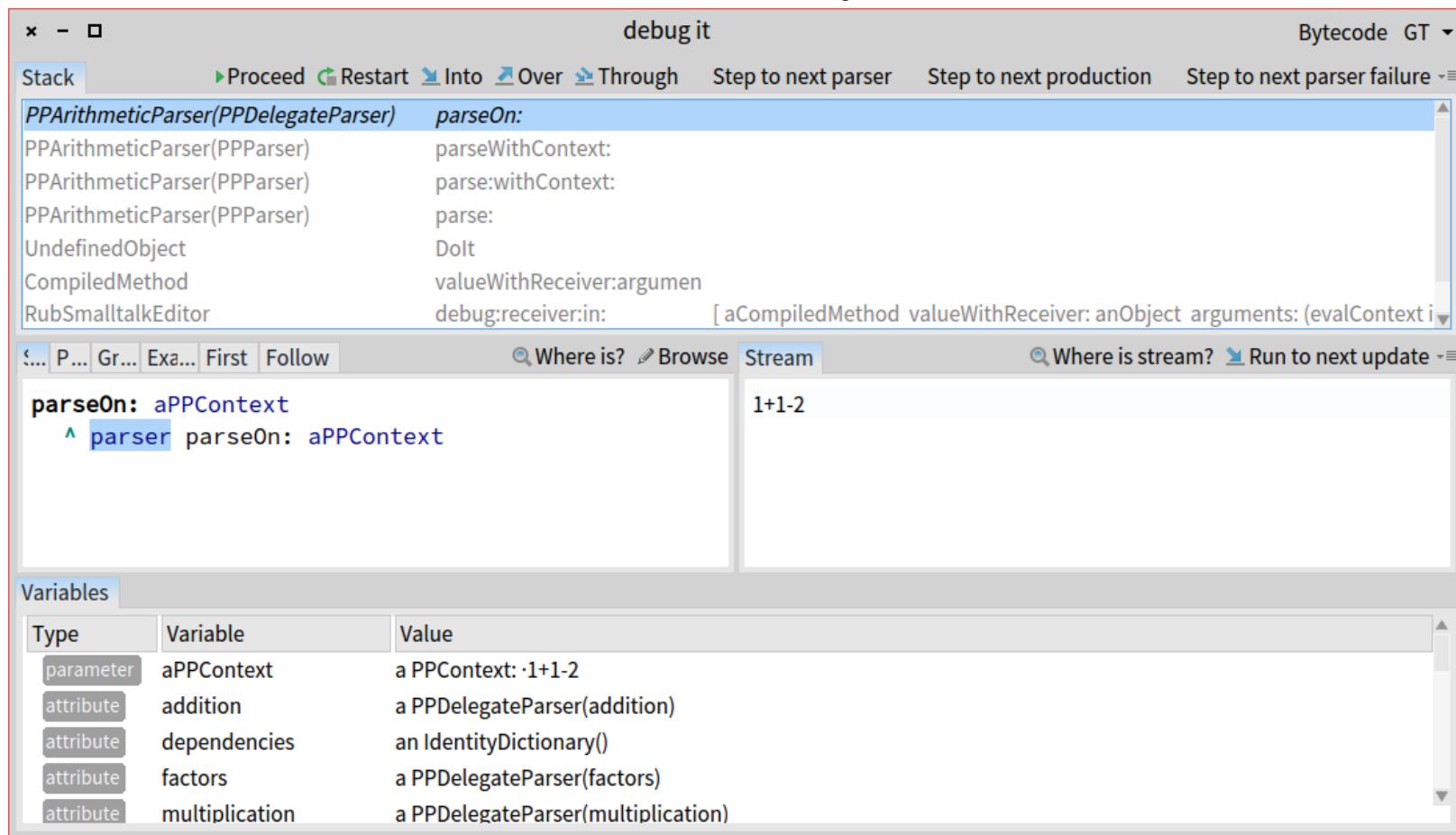


# Are different types of bugs handled differently?



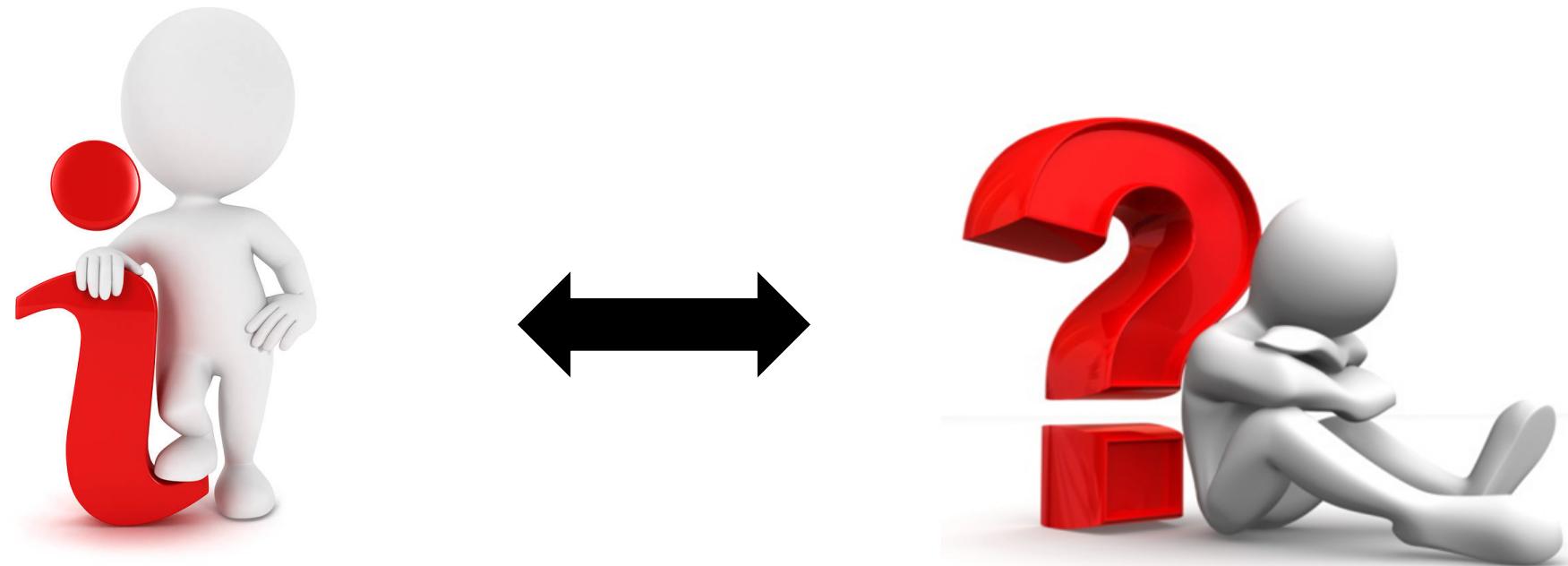
Custom debugger extensions

# Do custom extensions help?



Compare efficiency with and  
without extension

# Experienced vs inexperienced developers?

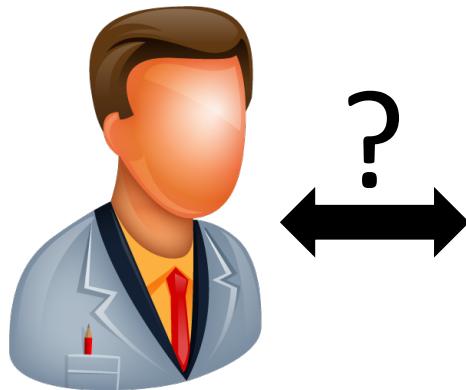


Learning from experienced developers

# Debugging is difficult



# Understand how developers use the debugger



The screenshot shows a debugger window titled "ZeroDivide". The "Bytecode" tab displays assembly-like code. The "Source" tab shows a Smalltalk method for division. The "Variables" tab lists:

Type	Variable	Value
Implicit	self	1
Implicit	aNumber	0
Implicit	thisContext	SmallInteger>>factorial
Implicit	stack top	0

# Provide better alternatives?

The screenshot shows a debugger window titled "debug it". The "Stack" tab shows a call stack with "factorial" at the top. The "Source" tab shows the factorial method implementation. The "Variables" tab lists:

Type	Variable	Value
Implicit	self	9
Implicit	thisContext	SmallInteger(Integer)>>factorial
Implicit	stack top	0

# Many possible ways to continue?

