

Powerful Tools for Live Development with Pharo

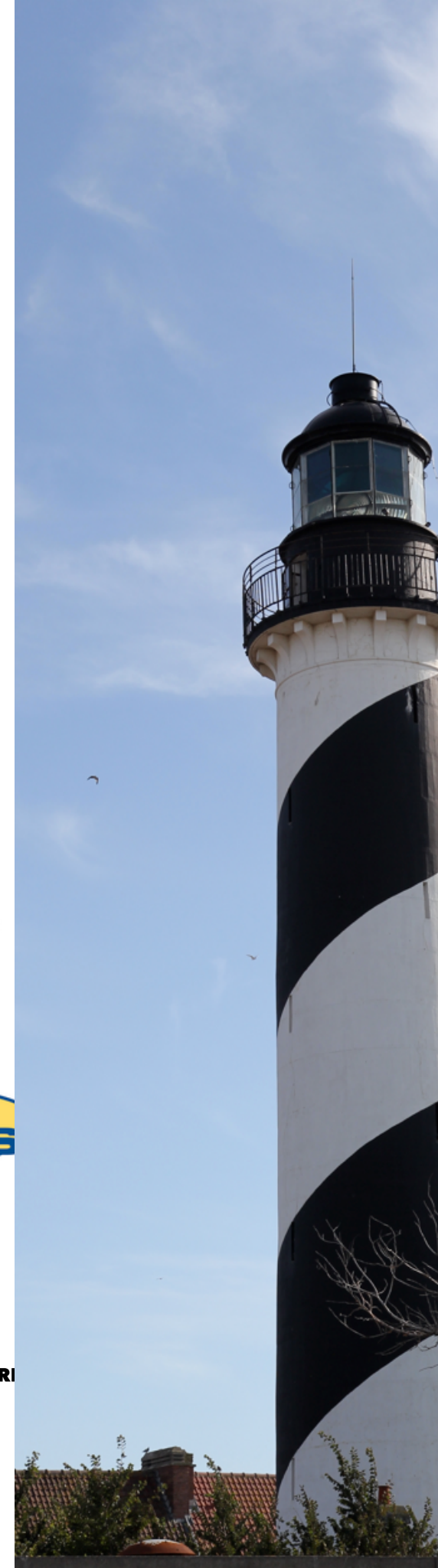
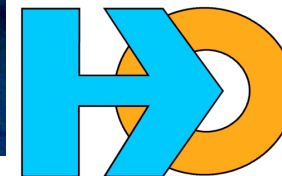
<http://stephane.ducasse.free.fr>
<http://www.pharo.org>

Inria



Université
de Lille





A journey in a live environment and its companion tools

- Pharo in 7 min
- Some advanced features
- Some tools
- Pharo is research friendly

Pharo!

- System: **Pure** object language + full IDE
- **Powerful, elegant** and **fun** to program
- **Living** system under your fingers
- Works on Mac OSX, Linux(es), iOS, Windows, Pi, and “*android*”
- 100% MIT

Pharo in Numbers

13 releases since 2008
Language Core + IDE +
Tools + Frameworks
710 packages (tests
included)

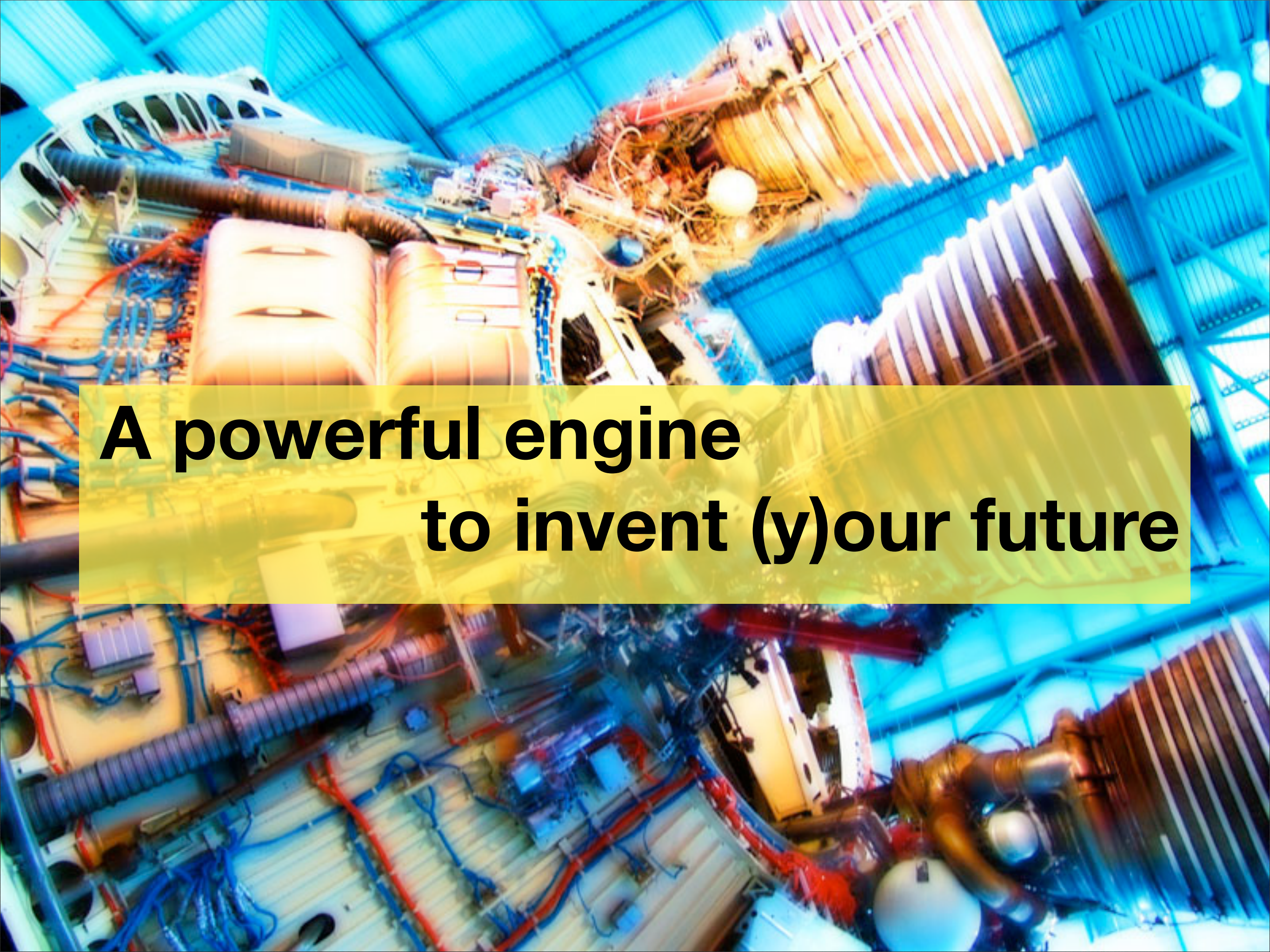
~ **27000 tests**
5 platforms
9 400 classes
130 000 methods
61 Mb (64 bits)

<http://github.com/pharo-project/Pharo> (~220
forks, 15/123
international
contributors)


Growing ecosystem
polymath
pharo-graphics
pharo-gis
pharo-container
pharo-ai



**Pharo is our vehicle
We improve it everyday**



**A powerful engine
to invent (y)our future**



**An ecosystem where
innovation/business bloom**

trentosur

Soluciones móviles para retail y trade marketing

Nos enfocamos en lo que importa del negocio sin perder de vista los detalles de su implementación.

Primer móvil Plataforma Android En la nube



Yesplan is veelzijdige software voor het efficiënt plannen van evenementen.

Yesplan is uiterst gebruiksvriendelijk, flexibel en makkelijk te koppelen met andere software.

Yesplan software interface showing a calendar and event planning tools.

airflowing

Organize your creative work

Sales, tasks and finances: your team and all that's essential in one place

Plans and Pricing

Manage your simple way

PharoCloud

Pharo platform as a Service: put your Smalltalk web-application online at Pharocloud in just 3 clicks

Try it for FREE

Romax Technology

Wind Energy

Get in touch

Related links

WEBDRUCK.CH

Web-To-Print Solution

- Design and create individual printed matter
- eShop with credit card payment
- High quality PDF output with Printing Process integration
- Thousands of orders for seven Swiss printing companies

WEBDRUCK.CH

Web-To-Print Solution

WEBDRUCK.CH website showing various printing services and a user interface.

Dedicated and cost-effective tools for software evolution

Software evolution tool interface.

Dedicated Analyses

Dedicated tools

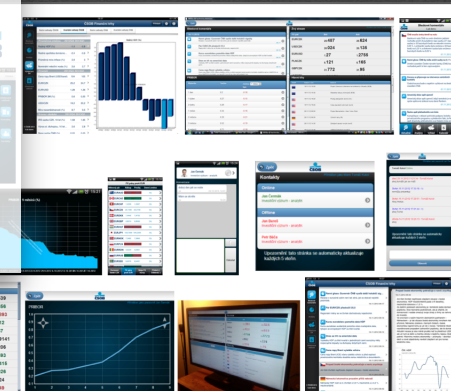
Decision making

NORRIZIK.COM

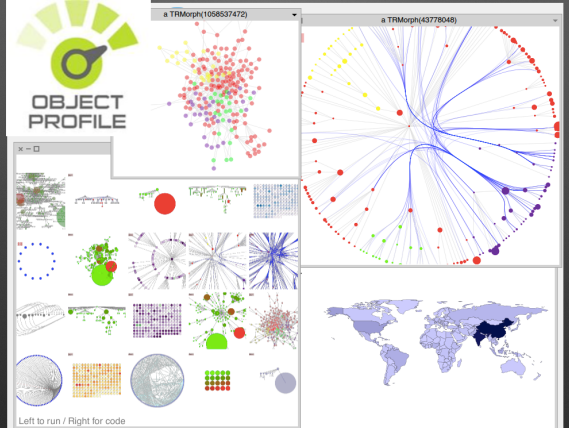
The world's first online platform fully supporting risk-based test management.

BETTER FASTER CHEAPER

COMPONENTS



Some Success Stories @ pharo.org/success



cmsbox

Das Content Management mit System

100% Inline-Editor

Drag & Drop

Copy / Paste

2denker

Continuous API Testing

keep your services under control 24/7

t3

t3 software interface showing a map and various data points.

Elegant!

- Full syntax on a postcard
- Simple and powerful objet model



method name parameter
exampleWithNumber: x

pragma
<syntaxOn: #postcard>
comment
"A "complete" Pharo syntax"

local variable
| y |

boolean literals
true & false

nil literal
not & (nil isNil)

block
ifFalse: [self perform: #add: with: x].

assignment
y := thisContext stack size + super size.

instance variable
byteArray := #[2 2r100 8r20 16rFF].

array generated at runtime
literal array
{ -42 . #(\$a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }

local block variable
do: [:each |

symbols
character
string
floating point
scaled decimal
| var |

block parameter
global variable
var := Transcript

keyword message
show: each class name;

keyword message
show: each printString].

keyword message
^ x < y

return instruction

other method definition examples:
unary
+ binaryMessageArgument
keyword: arg
keyword: arg1 withTwo: arg2

<https://www.pharo.org>

PLACE
STAMP
HERE

* H Ö R N U M

A Pure World of Objects

Only

objects + messages +

closures

mouse, booleans, arrays, numbers, strings, windows, scrollbars, canvas, files, trees, compilers, sound, url, socket, fonts, text, collections, stack, shortcut, streams, ...

A Fully Uniform Model

- Dynamically typed
- **Everything** is an object instance of a class
- All methods are public **virtual**
- All attributes are **protected**
- **Single** inheritance with **traits**

Less is more!

No constructors, no static methods, no operators

No type declaration, no primitive types,

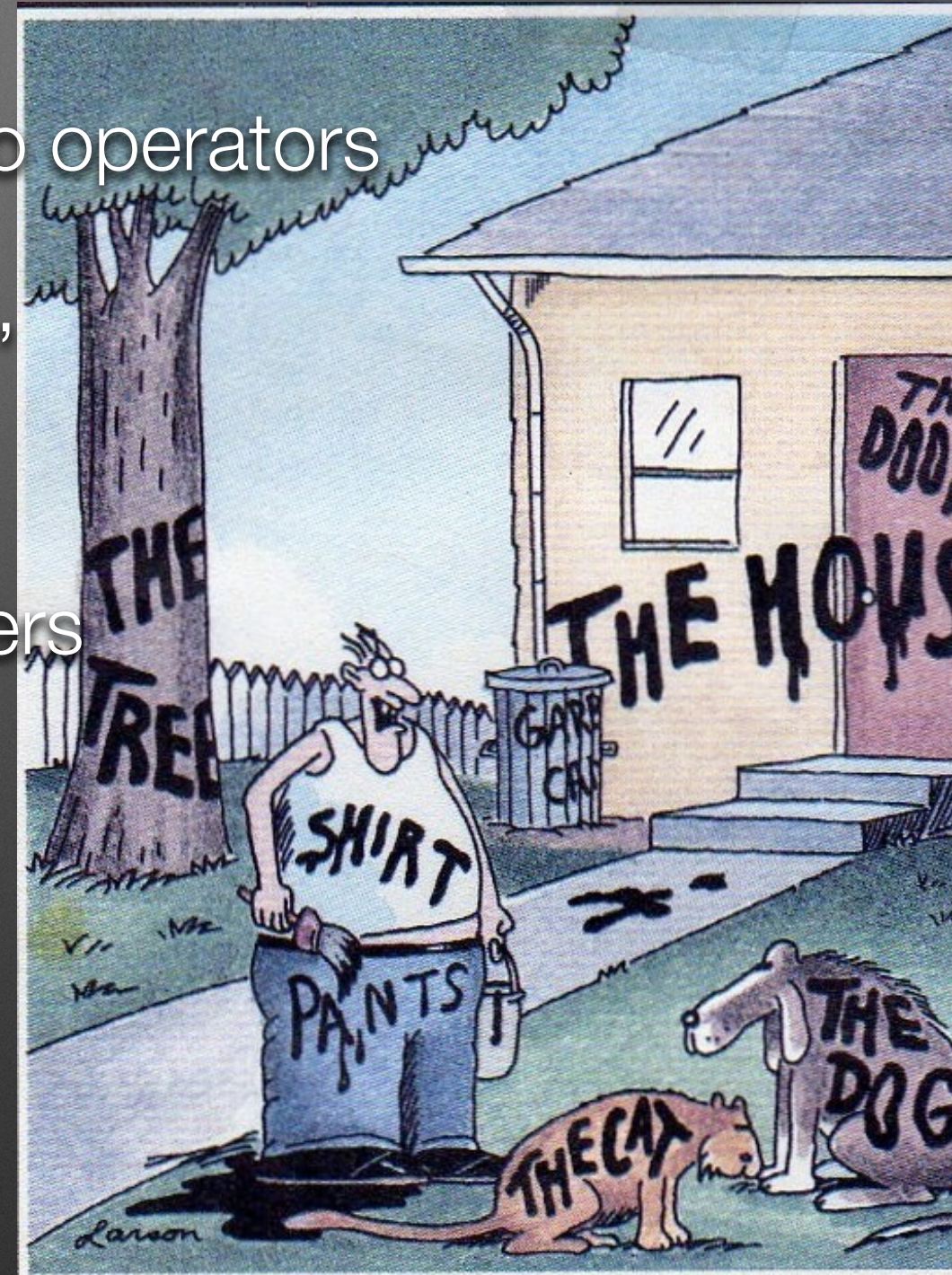
No interfaces, no need for factory

No packages/private/protected modifiers

No parametrized types

No boxing/unboxing

Still powerful



"Now! ... That should clear up a few things around here!"

Pharo is highly immersive

A large, deep blue aquarium tank is the central focus, filled with various marine life. A massive whale shark with a dark, spotted pattern swims horizontally across the middle of the frame. Above it, a large, spotted manta ray glides. The water is teeming with hundreds of smaller, silvery fish. In the foreground, the dark silhouettes of several people are visible, looking into the tank from behind a railing. The overall scene is one of a vast, immersive underwater world.

Immersing...

Pharo is not a blackbox

Everything is **fully inspectable** and
reflective



**You are immersed and
interacting with objects**

Workspace

```
| elements lay |
```

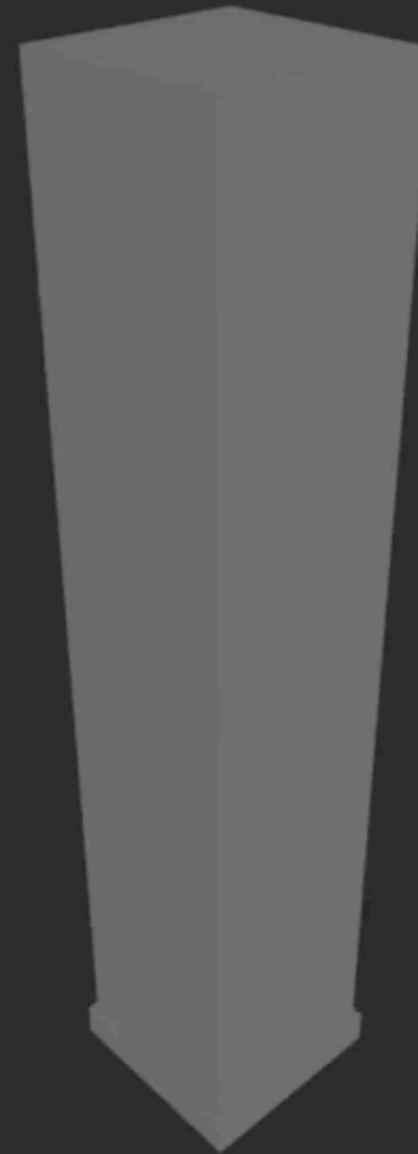
```
elements := (1 to: 5) collect: [ :ob |  
    (R3CubeShape new) elementOn: ob ].
```

I

```
lay := R3WallLayout new.  
lay on: elements.
```

```
UberPresenter present: elements
```

Uber Presenter



**We can do the same with
web app, sockets, networks,
sensors, living programming....**



Hackers
scripting live
the coffee
machine



Selected Infrastructure

Fully Written in Itself



Selected features

- First class instance variables (daemons, relationships...)
- Fast resumable exceptions
- Runtime classes and objects migration
- Customizable compiler
- Serializable and shareable execution stack
- Optional system virtualization
- Fully bootstrapped kernel(s) (down to 200kb)

Advanced reflective layer

- Versatile AST annotations and transformations @ runtime
- Full stack reification (continuations, exceptions...)
- Instance enumeration
- Causally connected “Software as Objects”
- Atomic bulk object swapping
- ... more but no time for that

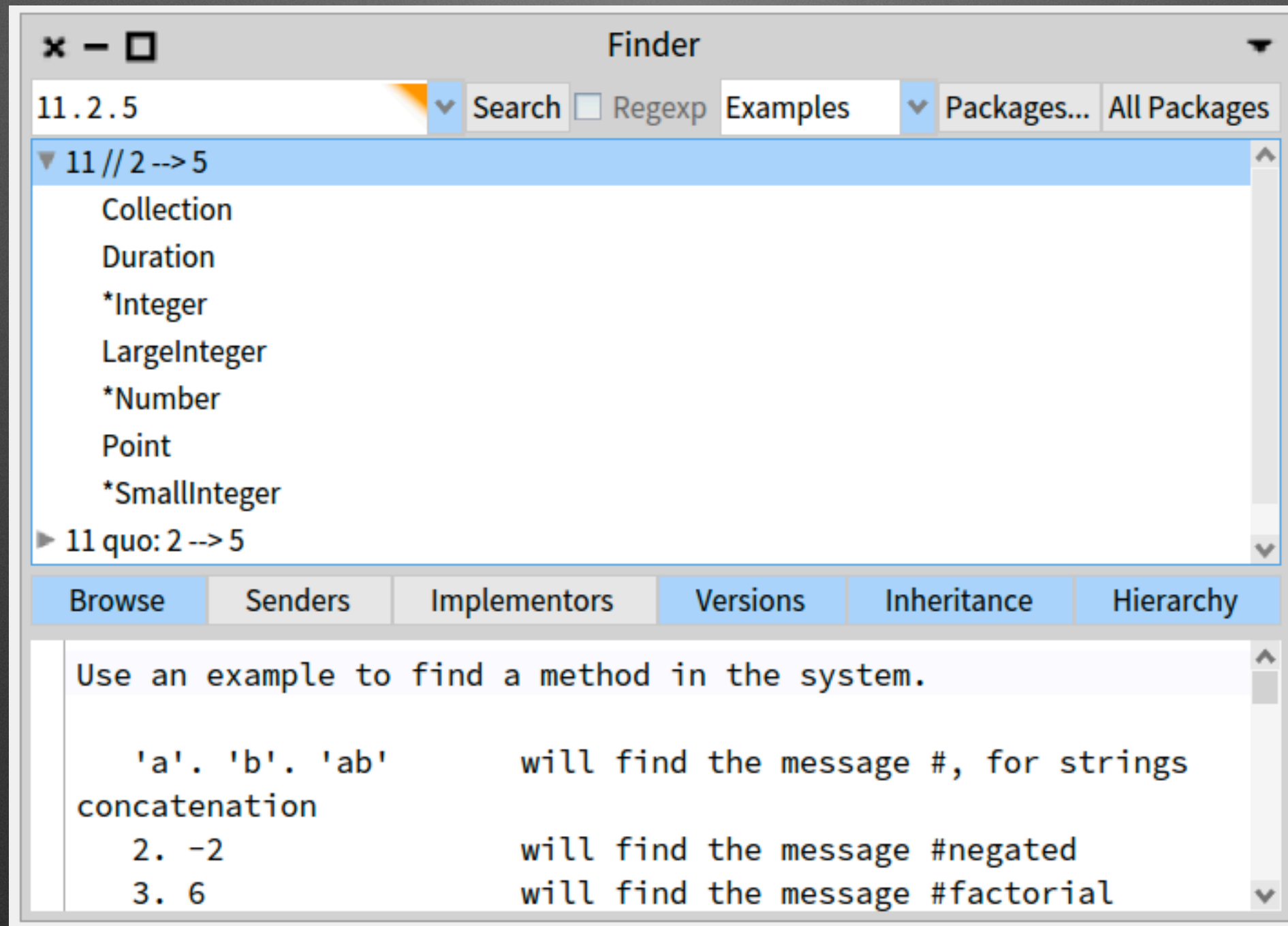


Tools are our friends

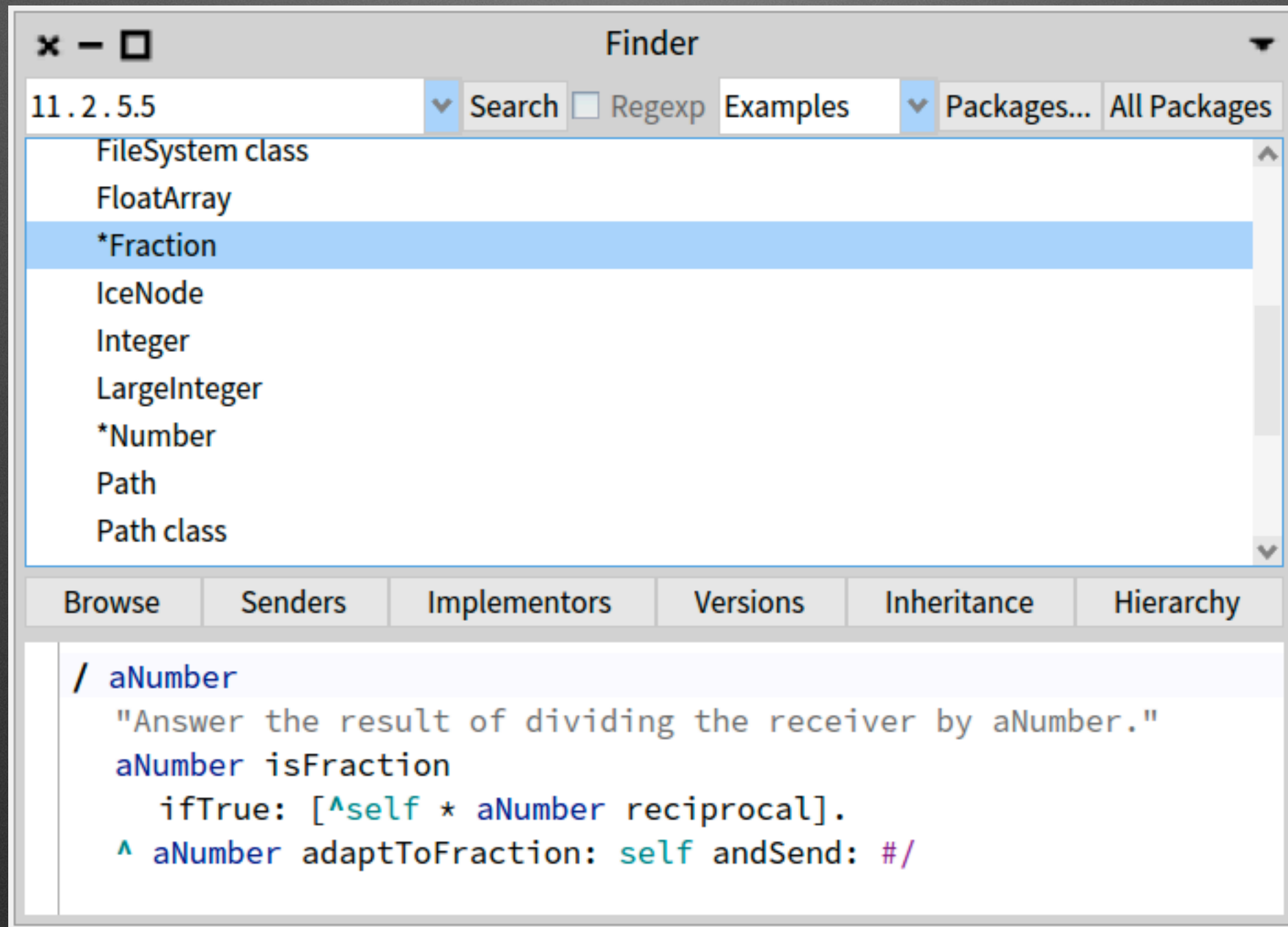
How to find information?

- Libraries are large
- You know what you want
- You do not know how to express it

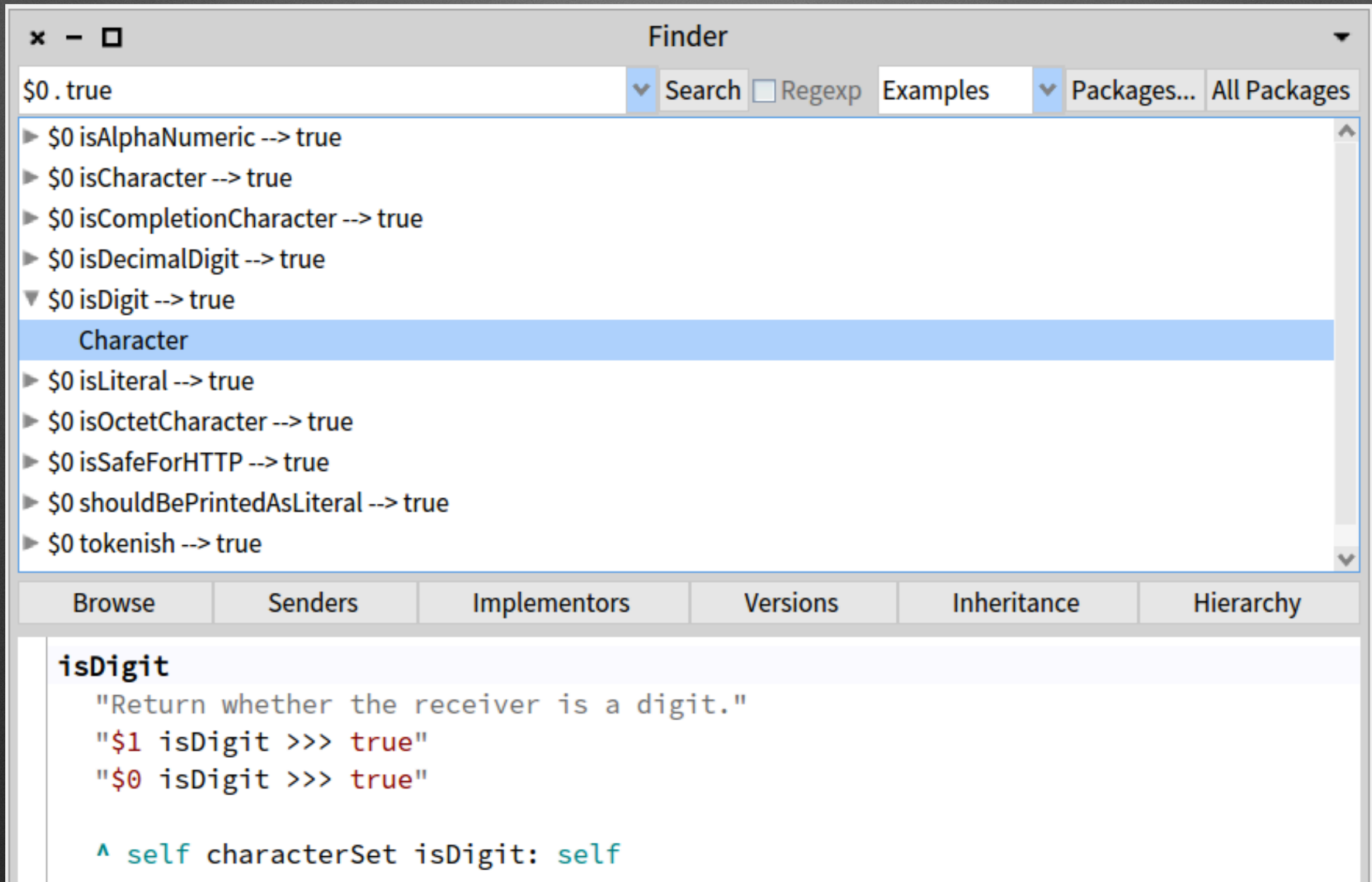
11 ??? 2 should give 5



11 ??? 2 should give 5.5



What are the messages send to \$0 that return true



The screenshot shows the Ruby Finder application window. The search bar at the top contains the text "\$0.isAlphaNumeric --> true". Below the search bar, a list of search results is displayed. The first result is "\$0.isAlphaNumeric --> true", followed by "\$0.isCharacter --> true", "\$0.isCompletionCharacter --> true", "\$0.isDecimalDigit --> true", and "\$0.isDigit --> true". The "\$0.isDigit --> true" result is expanded, showing a list of messages that return true for the isDigit method. These messages are: isLiteral, isOctetCharacter, isSafeForHTTP, shouldBePrintedAsLiteral, and tokenish. The "Character" class is highlighted in blue. At the bottom of the window, there is a tabbed interface with tabs for "Browse", "Senders", "Implementors", "Versions", "Inheritance", and "Hierarchy". The "Implementors" tab is selected, showing the implementation of the isDigit method in the Character class.

Finder

\$0.isAlphaNumeric --> true Search ☐ Regexp Examples Packages... All Packages

- ▶ \$0.isAlphaNumeric --> true
- ▶ \$0.isCharacter --> true
- ▶ \$0.isCompletionCharacter --> true
- ▶ \$0.isDecimalDigit --> true
- ▼ \$0.isDigit --> true
 - Character
 - ▶ \$0.isLiteral --> true
 - ▶ \$0.isOctetCharacter --> true
 - ▶ \$0.isSafeForHTTP --> true
 - ▶ \$0.shouldBePrintedAsLiteral --> true
 - ▶ \$0.tokenish --> true

Browse Senders Implementors Versions Inheritance Hierarchy

isDigit

```
"Return whether the receiver is a digit."  
"$1 isDigit >>> true"  
"$0 isDigit >>> true"  
  
^ self.characterSet isDigit: self
```


Tools

- Shape our mind...
- Pharo has moldable tools: **you CAN adapt them to you and your process** and not the inverse
- Build fast **your own** tools

Pharo has **amazing**
moldable tools

**Customizable object
interaction/presentations**

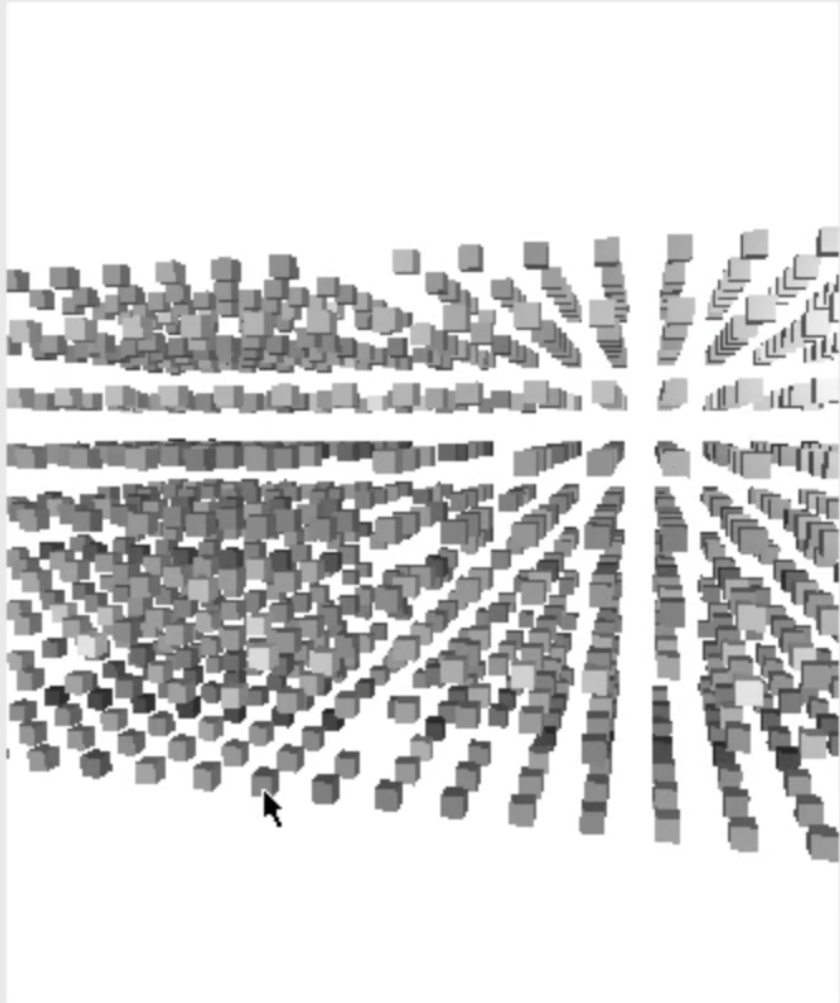
Inspecting live a 3D object

Playground

```
data := TestData new data.  
  
cube := MatrixCube new initWithData: data.  
cube view
```

a RWView

Raw Live Meta



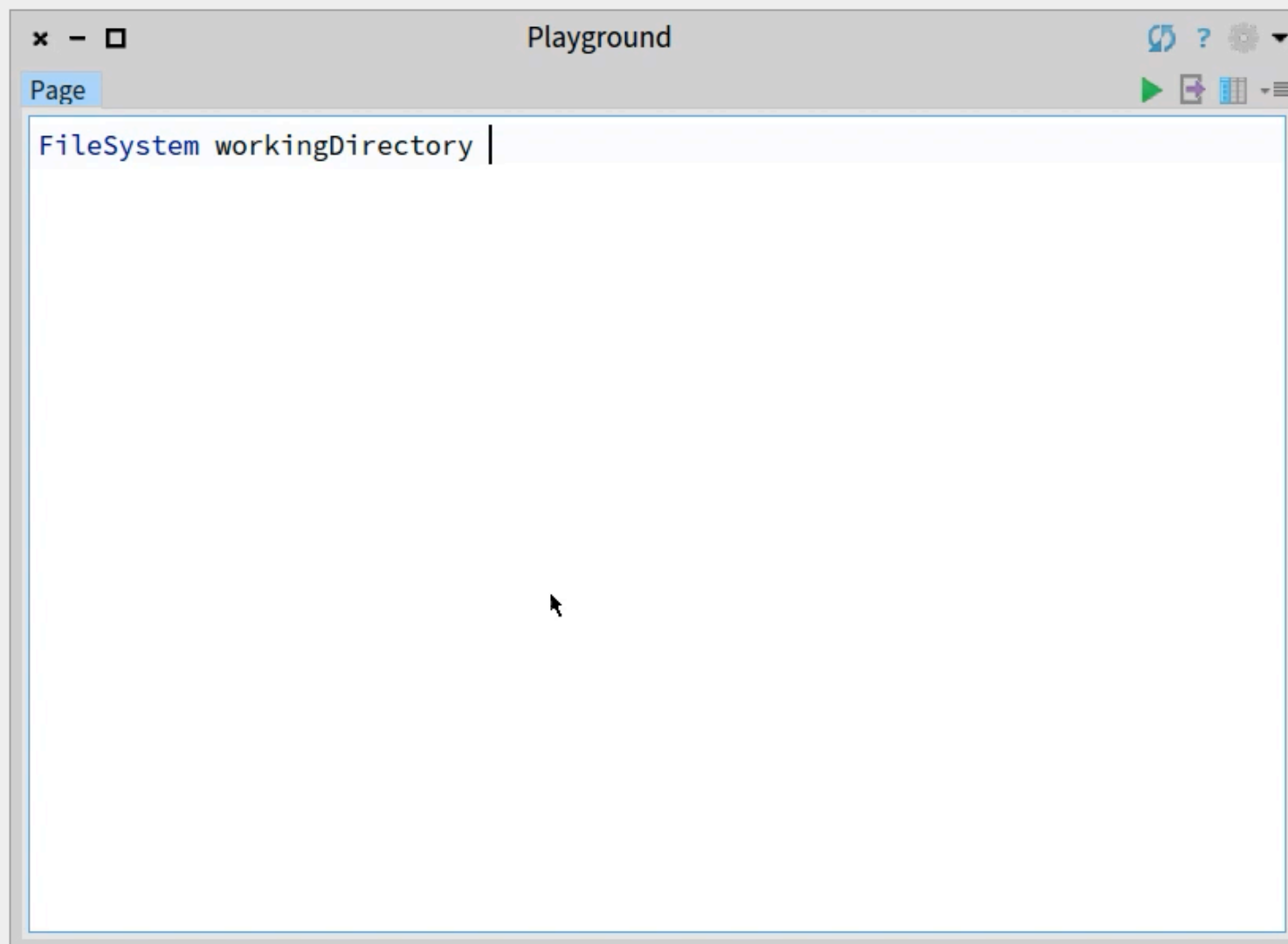
an Array [4 items] ('2000-01-01' 'Malawi' 'France' ...)

Items Raw Meta

Index	Item
1	'2000-01-01'
2	'Malawi'
3	'France'
4	'400.0'

enter search query (example: "each = 5")

The views of a file reference



It is cool but it is not magic
You can define your own

Implementing a pane!

The screenshot shows the GT-Inspector IDE interface. The title bar reads "AbstractFileReference>>gtInspectorPngIn:". The interface is divided into several panes:

- Left Pane:** A tree view showing the package structure. The "Public" package is selected, and its "Extensions" are listed, including "FileSystem-Disk", "FileSystem-Memory", "FileSystem-Path", "FileSystem-Tests-Attributes", "FileSystem-Tests-Core", "FileSystem-Tests-Disk", "FileSystem-Tests-Memory", "FileSystem-Zip", "Files", "Files-Prompt", and "Files-Tests".
- Middle Pane:** A list of classes and methods. The "AbstractFileReference" class is selected, and its methods are listed: "FileLocator", "FileReference", "FileSystem", "FileSystemDirectoryEntry", "DiskDirectoryEntry", "DiskSymlinkDirectoryEntry", "MemoryDirectoryEntry", and "FileSystemPermission".
- Right Pane:** A list of methods. The "gtInspectorPngIn:" method is selected, and its implementation is shown in the bottom pane.
- Bottom Pane:** The implementation of the "gtInspectorPngIn:" method, which is a composite method. The code is as follows:

```
gtInspectorPngIn: composite
  <gtInspectorPresentationOrder: 0>
  composite morph
    title: 'Picture';
    display: [ self binaryReadStreamDo: [ :stream | PNGReadWriter formFromStream: stream ] ];
    when: [ self isFile and:
      [ self mimeTypes notNil and:
        [ self mimeTypes first matches: ZnMimeType imagePng ] ] ]
```

The bottom status bar shows "8/8 [9]" and "GT-InspectorExtensions-Core" with a checked "extension" checkbox and an "F" button.

Files are
boring...
What about
inside
the
system?



A class is an object we can inspect!

The screenshot shows the 'Playground' application window. On the left, a text area contains the word 'Point'. On the right, a panel titled 'a Point class (Point)' displays a table of the object's attributes and their values.

Variable	Value
self	Point
superclass	Object
methodDict	a MethodDictionary [103 items] (size 103)
format	65538
layout	a FixedLayout
organization	a ClassOrganization
subclasses	nil
name	#Point
classPool	a Dictionary [0 items] /

Below the table, the string '"Point"' and the variable 'self' are shown in a code-like font.

“A class has a method dictionary”
they said... let us verify

Playground

a Point class (Point)

Raw Defir... Meth... All R... All Ref Com... Inst\... Insta... Meta

Variable	Value
self	Point
superclass	Object
methodDict	a MethodDictionary [103 items] (size 103)
format	65538
layout	a FixedLayout
organization	a ClassOrganization
subclasses	nil
name	#Point
classPool	a Dictionary [0 items] (size 0)

"Point"

self

a MethodDictionary [103 items] (size 103)

Items Keys Raw Meta

Key	Value
#reflectedAbout:	Point>>#reflectedAbout:
#rotateBy:centerAt:	Point>>#rotateBy:centerAt:
#adaptToNumber:andSend:	Point>>#adaptToNumber:andSend:
#squaredDistanceTo:	Point>>#squaredDistanceTo:
#adaptToCollection:andSend:	Point>>#adaptToCollection:andSend:
#theta	Point>>#theta
#transposed	Point>>#transposed
#-	Point>>#-
#fourDirections	Point>>#fourDirections
#crossProduct:	Point>>#crossProduct:
#scaleFrom:to:	Point>>#scaleFrom:to:
#veryDeepCopyWith:	Point>>#veryDeepCopyWith:

Dissecting one method object

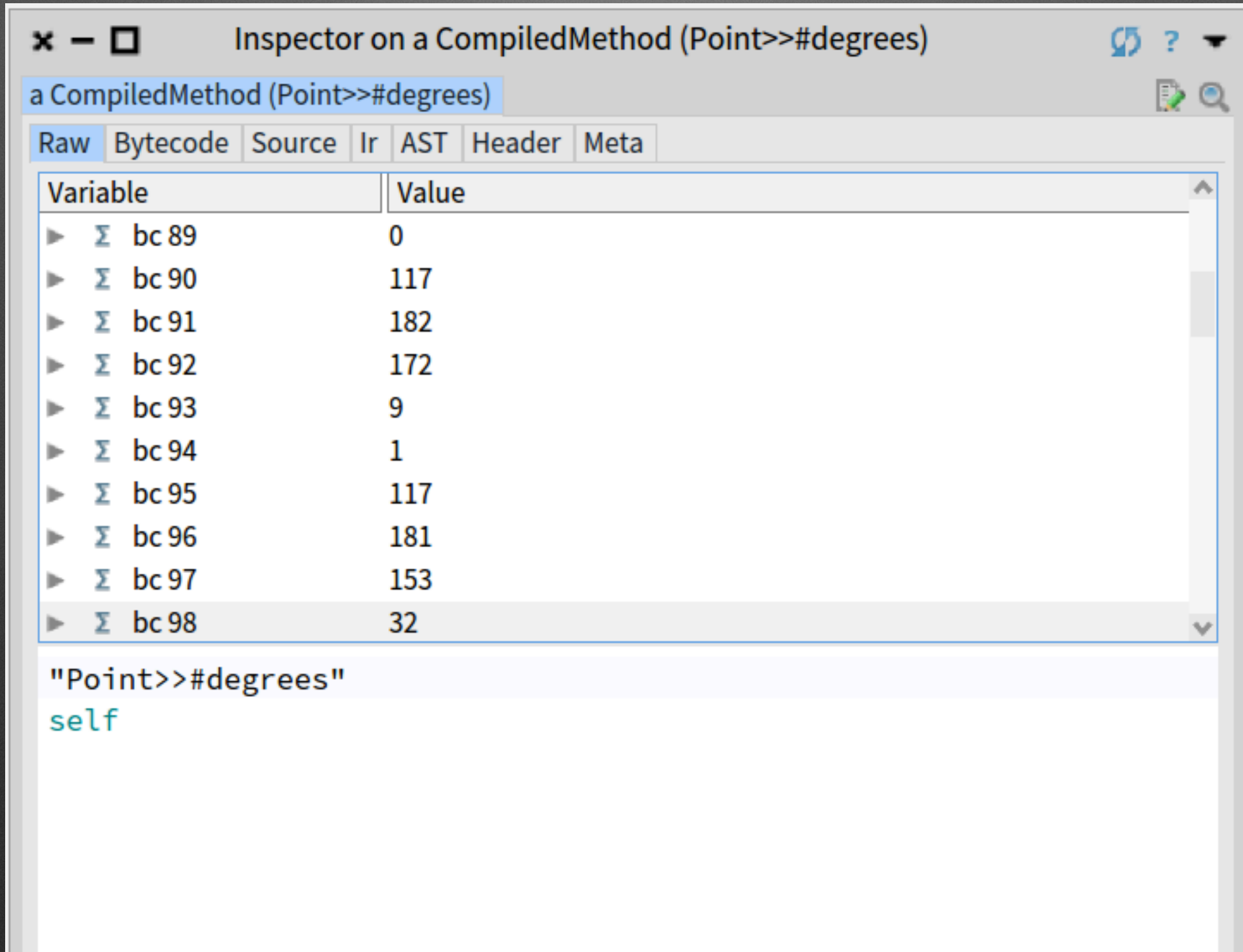
The screenshot shows the Ruby Inspector window titled "Inspector on a CompiledMethod (Point>>#degrees)". The selected object is "a CompiledMethod (Point>>#degrees)". The "Raw" tab is active, displaying a table of variables and their values.

Variable	Value
{ } self	Point>>#degrees
▶ Σ literal1	90.0
▶ Σ literal2	270.0
▶ ¶ literal3	#asFloat
▶ ¶ literal4	#arcTan
▶ ¶ literal5	#radiansToDegrees
▶ Σ literal6	360.0
▶ Σ literal7	180.0
▶ ¶ literal8	#ifTrue:ifFalse:
▶ Σ bc 89	0

Below the table, the source code for the method is displayed:

```
"Point>>#degrees"  
self
```


I do not want to be a compiler!

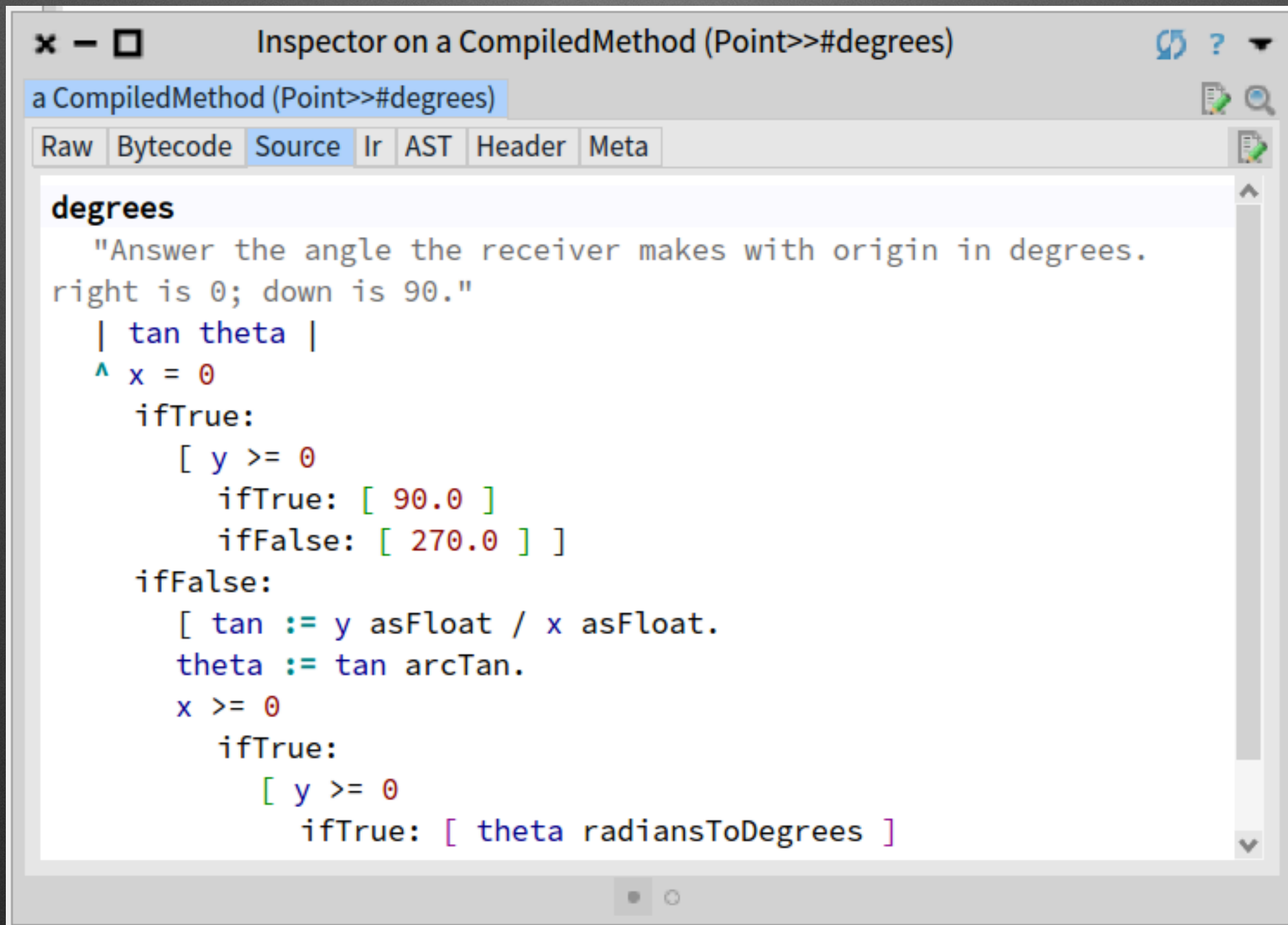


The screenshot shows a window titled "Inspector on a CompiledMethod (Point>>#degrees)". The window has a tab bar with "Raw", "Bytecode", "Source", "Ir", "AST", "Header", and "Meta". The "Raw" tab is selected. Below the tab bar is a table with two columns: "Variable" and "Value". The table contains 10 rows of data, each with a small icon, a variable name, and a value. The last row is highlighted. Below the table, the text "Point>>#degrees" is displayed, followed by "self" in a different color.

Variable	Value
bc 89	0
bc 90	117
bc 91	182
bc 92	172
bc 93	9
bc 94	1
bc 95	117
bc 96	181
bc 97	153
bc 98	32

"Point>>#degrees"
self

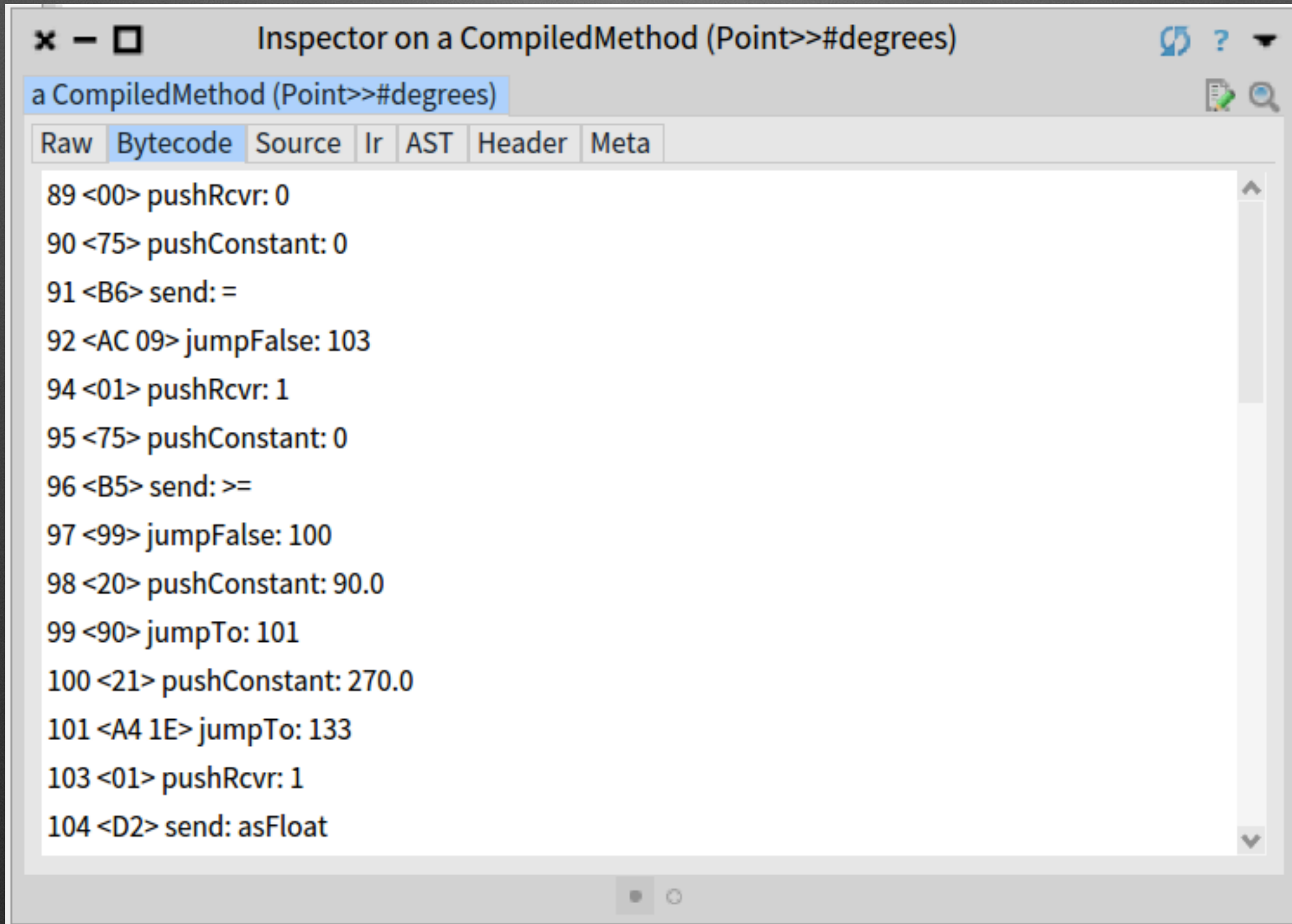
It looks like a method



The screenshot shows a Ruby Inspector window titled "Inspector on a CompiledMethod (Point>>#degrees)". The window displays the source code of the `#degrees` method. The code is as follows:

```
degrees
  "Answer the angle the receiver makes with origin in degrees.
  right is 0; down is 90."
  | tan theta |
  ^ x = 0
  ifTrue:
    [ y >= 0
      ifTrue: [ 90.0 ]
      ifFalse: [ 270.0 ] ]
  ifFalse:
    [ tan := y asFloat / x asFloat.
      theta := tan arcTan.
      x >= 0
        ifTrue:
          [ y >= 0
            ifTrue: [ theta radiansToDegrees ]
```


Numbers are not that obscure



The screenshot shows a window titled "Inspector on a CompiledMethod (Point>>#degrees)". The window has a tab bar with "Raw", "Bytecode", "Source", "Ir", "AST", "Header", and "Meta". The "Bytecode" tab is selected. The main area displays a list of bytecode instructions with their line numbers and offsets:

```
89 <00> pushRcvr: 0
90 <75> pushConstant: 0
91 <B6> send: =
92 <AC 09> jumpFalse: 103
94 <01> pushRcvr: 1
95 <75> pushConstant: 0
96 <B5> send: >=
97 <99> jumpFalse: 100
98 <20> pushConstant: 90.0
99 <90> jumpTo: 101
100 <21> pushConstant: 270.0
101 <A4 1E> jumpTo: 133
103 <01> pushRcvr: 1
104 <D2> send: asFloat
```


And mapping them to the good abstraction helps

The image shows a screenshot of the Ruby Inspector application, which is used for inspecting the Abstract Syntax Tree (AST) of Ruby code. The window is titled "Inspector on a CompiledMethod (Point>>#degrees)".

The left pane shows the AST structure for a compiled method. The root node is `RBMethodNode(degrees "Answer the angle the receiver makes with)`. It contains a sequence of nodes: `RBSequenceNode(| tan theta | ^ x = 0 ifTrue: [y >= 0 ifTrue:)`, `RBReturnNode(^ x = 0 ifTrue: [y >= 0 ifTrue: [90.0])`, and `RBMessageNode(x = 0 ifTrue: [y >= 0 ifTrue: [90.0])`. The `RBMessageNode(x = 0 ifTrue: [y >= 0 ifTrue: [90.0])` node is expanded, showing its children: `RBMessageNode(x = 0)`, `RBlockNode([y >= 0 ifTrue: [90.0] ifFalse: [270.0]])`, and `RBSequenceNode(y >= 0 ifTrue: [90.0] ifFalse: [270.0])`. The `RBSequenceNode(y >= 0 ifTrue: [90.0] ifFalse: [270.0])` node is further expanded, showing `RBMessageNode(y >= 0 ifTrue: [90.0] ifFalse: [270.0])`, `RBlockNode([90.0])`, and `RBlockNode([270.0])`.

The right pane shows the source code for the `degrees` method. The code is as follows:

```
degrees
  "Answer the angle the receiver makes with
  origin in degrees. right is 0; down is 90."
  | tan theta |
  ^ x = 0
  ifTrue:
    [ y >= 0
      ifTrue: [ 90.0 ]
      ifFalse: [ 270.0 ] ]
  ifFalse:
    [ tan := y asFloat / x asFloat.
      theta := tan arcTan.
      x >= 0
        ifTrue:
          [ y >= 0
            ifTrue: [ theta radiansToDegrees ]
            ifFalse: [ 360.0 + theta
              radiansToDegrees ] ]
```


Yes pushRcvr: 1 means the second field!

Inspector on a CompiledMethod (Point>>#degrees)

a CompiledMethod (Point>>#degrees)

Raw	Bytecode	Source	Ir	AST	Header	Meta
89	<00>	pushRcvr: 0				
90	<75>	pushConstant: 0				
91	<B6>	send: =				
92	<AC 09>	jumpFalse: 103				
94	<01>	pushRcvr: 1				
95	<75>	pushConstant: 0				
96	<B5>	send: >=				
97	<99>	jumpFalse: 100				
98	<20>	pushConstant: 90.0				
99	<90>	jumpTo: 101				
100	<21>	pushConstant: 270.0				
101	<A4 1E>	jumpTo: 133				
103	<01>	pushRcvr: 1				
104	<D2>	send: asFloat				
105	<00>	pushRcvr: 0				
106	<D2>	send: asFloat				

a SymbolicBytecode (94 <01> pushRcvr: 1)

Raw	Source	SourceNode	Meta
	origin in degrees. right is 0; down is 90."		
	tan theta		
	^ x = 0		
	ifTrue:		
	[y >= 0		
	ifTrue: [90.0]		
	ifFalse: [270.0]]		
	ifFalse:		
	[tan := y asFloat / x asFloat.		
	theta := tan arcTan.		
	x >= 0		
	ifTrue:		
	[y >= 0		
	ifTrue: [theta radiansToDegrees]		
	ifFalse: [360.0 + theta		
	radiansToDegrees]]		
	ifFalse: [180.0 + theta		
	radiansToDegrees]]		

Pharo Pro devs do XtremeTDD

- Get **productivity boost**
- Xtreme TDD
 - write test, test fails and
 - **code in debugger**

Counter

Commander2

Commander2-Deprecation

Commander2-Tests

Commander2-UI

Commander2-UI-Tests

Compression

Compression-Tests

ConfigurationCommandLi

ConfigurationCommandLi

Counter

Debugger-Actions

Filter...

All Packages

Scoped View

Inst. side

Class side

? Comment

+ New class

Object subclass: #NameOfSubclass

instanceVariableNames: ''

classVariableNames: ''

package: 'Counter'

Slots

Hot update on the fly
customizable debugger

Halt

Bytecode

Stack

[▶ Proceed](#) [↺ Restart](#) [↵ Into](#) [↗ Over](#) [↘ Through](#)

PDFCellElement	getSubElementsWith:styleSheet:
PDFCellElement(PDFComposite)	generateCodeSegmentsCollectionWi
PDFCellElement(PDFComposite)	generateCodeSegmentWith:styleShe
PDFDataTableElement(PDFComposite)	generateCodeSegmentsCollectionWi [:aSubElement aSubElement generateCodeSe
Array(SequenceableCollection)	collect:

Source

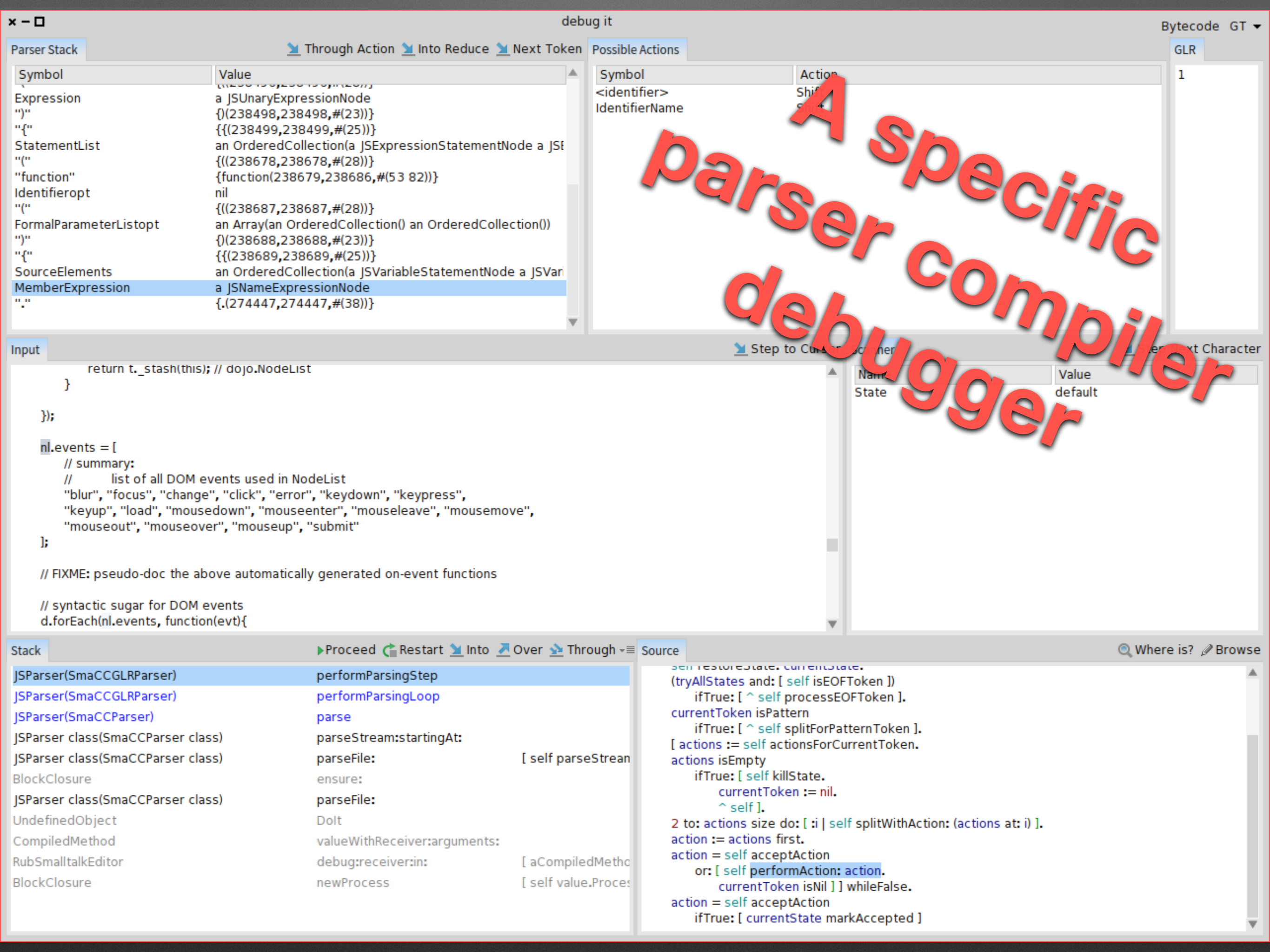
[Where is?](#) [Browse](#)

```
generateCodeSegmentsCollectionWith: aPDFGenerator styleSheet: compositeStyleSheet format: aFormat
^ (self getSubElementsWith: aPDFGenerator styleSheet: compositeStyleSheet)
  collect: [ :aSubElement |
    aSubElement
      generateCodeSegmentWith: aPDFGenerator
      styleSheet: (aSubElement buildCompositeStyleSheetFrom: compositeStyleSheet)
      format: aFormat ]
```

Variables

Type	Variable	Value
implicit	self	a PDFCellElement
parameter	aFormat	a PDFA4Format
parameter	aPDFGenerator	a PDFGenerator
parameter	compositeStyleSheet	a StyleSheet

dimension: 80 mm @ 20 mm;



A specific
parser compiler
debugger

Halt in OCDBox>>name:

Class	Method	Context
OCDBox	name:	UndefinedObject>>Dolt
UndefinedObject	Dolt	CompiledMethod>>valueWithReceiver
CompiledMethod	valueWithReceiver:arguments:	[aCompiledMethod
SmalltalkEditor	debug:receiver:in:	[self value.

Bytecode Breakpoints Sindarin

Type	Target	Method
<input type="checkbox"/> Breakpoint	self	Reflecting Examples>>examples
<input checked="" type="checkbox"/> Breakpoint	self	OCDBox>>initialize
<input checked="" type="checkbox"/> Breakpoint	self	OCDBox>>name:
<input checked="" type="checkbox"/> Breakpoint	self	OCDBox>>name:
<input checked="" type="checkbox"/> Halt	OCDBox	OCDBox>>name:
<input type="checkbox"/> Breakpoint	OCDBox	OCDBox>>name:
<input type="checkbox"/> Breakpoint	self	OCDBox>>removeElement:
<input type="checkbox"/> Breakpoint	self	OrderedCollection>>removeElement
<input type="checkbox"/> Halt	StHaltCacheTest	StHaltCacheTest>>testInitial
<input checked="" type="checkbox"/> Halt	StHaltCacheTest	StHaltCacheTest>>testInitial

```
1 name: anObject
2 self halt.
3 name := anObject
```

specific view

Receiver in: a StDebuggerContext (OCDBox>>name:)

Variable	Value
[arg] anObject	'i'
self	an OCDBox
{ } elements	an OrderedCollection [0 items] ()
name	"
{ } Temps	a Dictionary [1 item] (#anObject->'i')
anObject	'i'
stackTop	an OCDBox

an OCDBox

Raw	Breakpoints	Meta
Type	Target	Method
<input checked="" type="checkbox"/> Halt	OCDBox	OCDBox>>name:
1		
1	"an OCDBox"	
2	self	

Bytecode Breakpoints Sindarin

```
33 <4C> self
34 <80> send: halt
35 <D8> pop
36 <40> pushTemp: 0
37 <C9> popIntoRcvr: 1
38 <58> returnSelf
```

Variable	Value
stackTop	an OCDBox
0 [anObject]	'i'
rcvr: 0 [elements]	an OrderedCollection [0 items] ()
rcvr: 1 [name]	"

object centric +
adv scripting (AST + call stack)

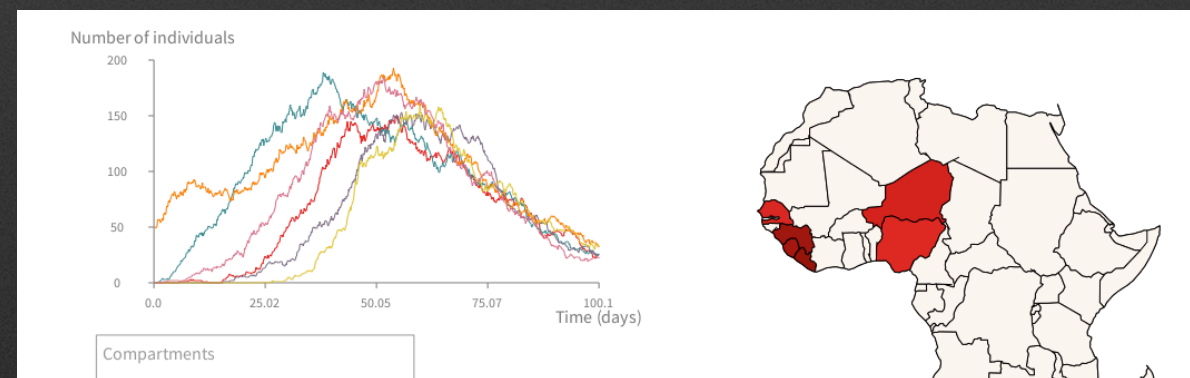
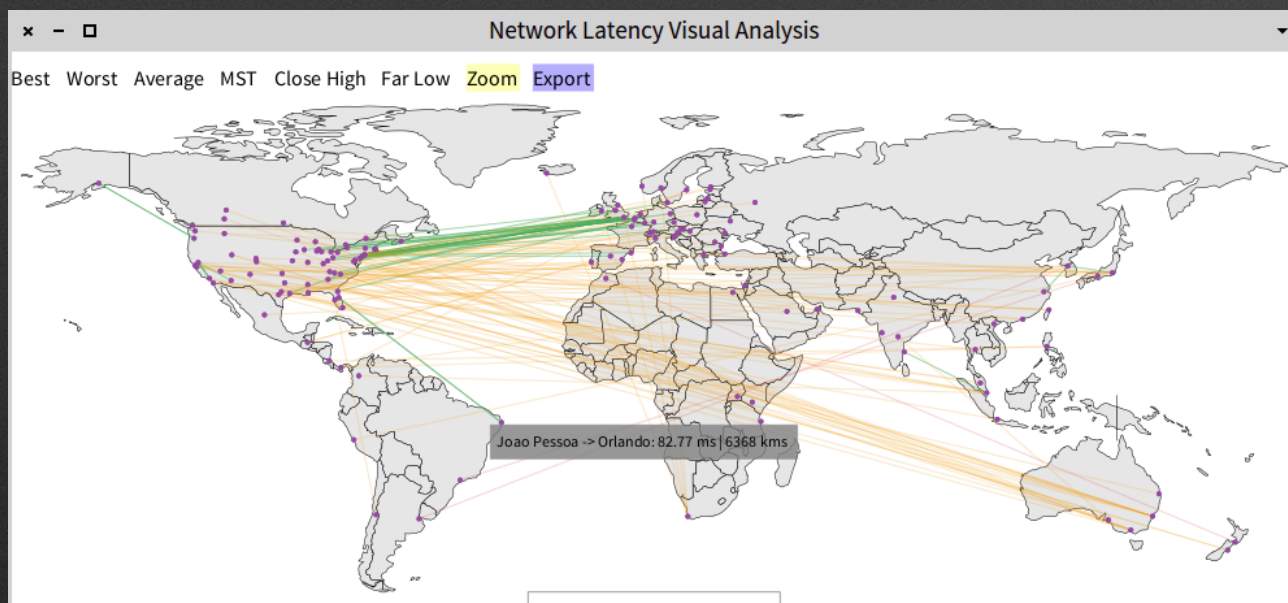
Live visualisation scripting

- The next level
- Roassal 3.0 by Prof. A. Bergel/Object Profile University of Chile at Santiago
- Simply gorgeous
- Check <http://agilevisualization.com>

Includes a DSL for Scripting visualisations

```
b := RTMondrian new.  
  b shape rectangle  
    withBorder;  
  width: [ :cls | cls numberOfVariables * 5 ];  
  height: [ :cls | cls numberOfMethods ].
```

```
b nodes: Collection withAllSubclasses.  
b edges connectToAll: [ :cls | cls subclasses ].  
b layout tree.  
b normalizer  
  normalizeColorAsGray: [ :cls |  
cls numberOfLinesOfCode ].  
b
```



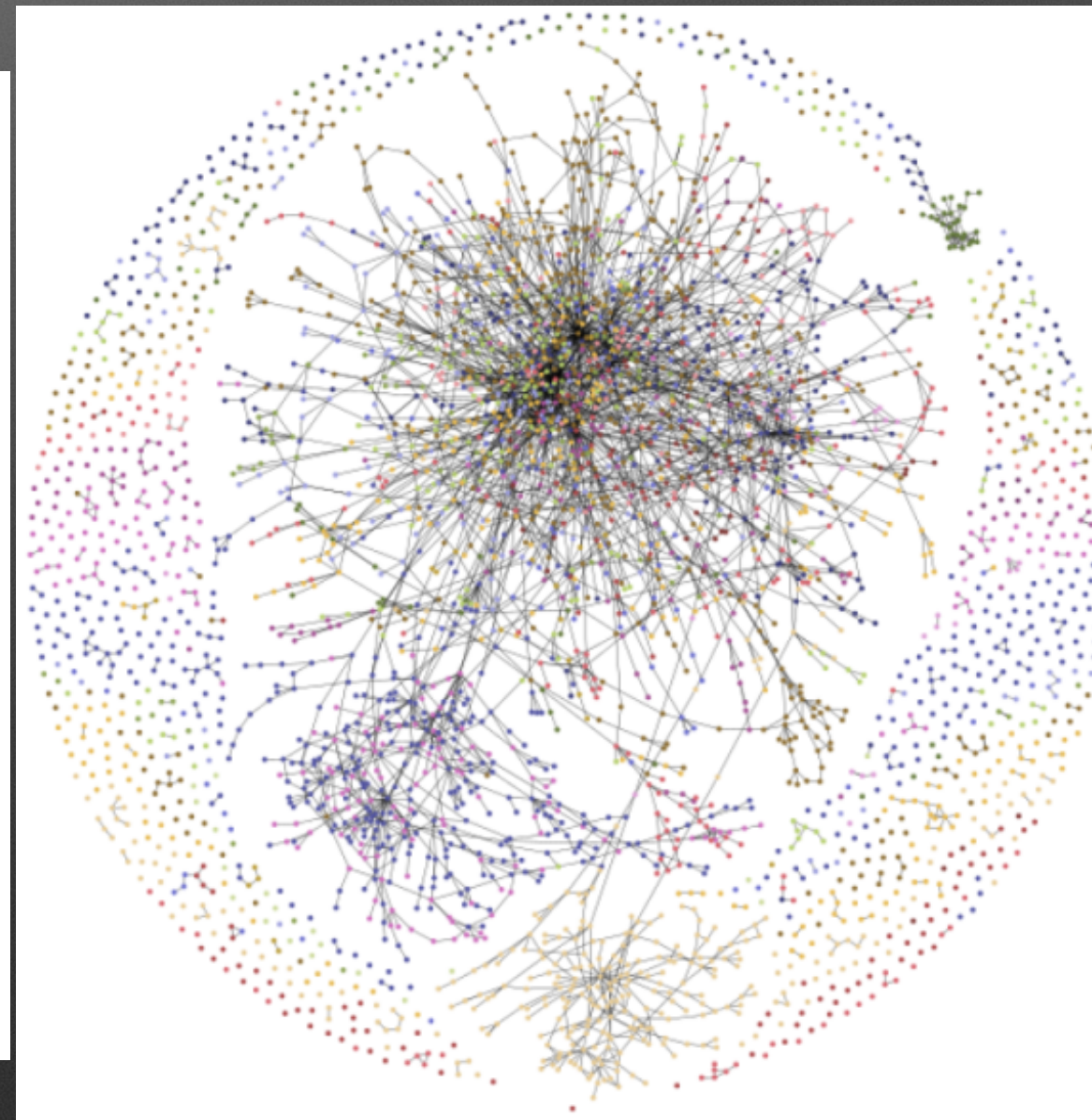
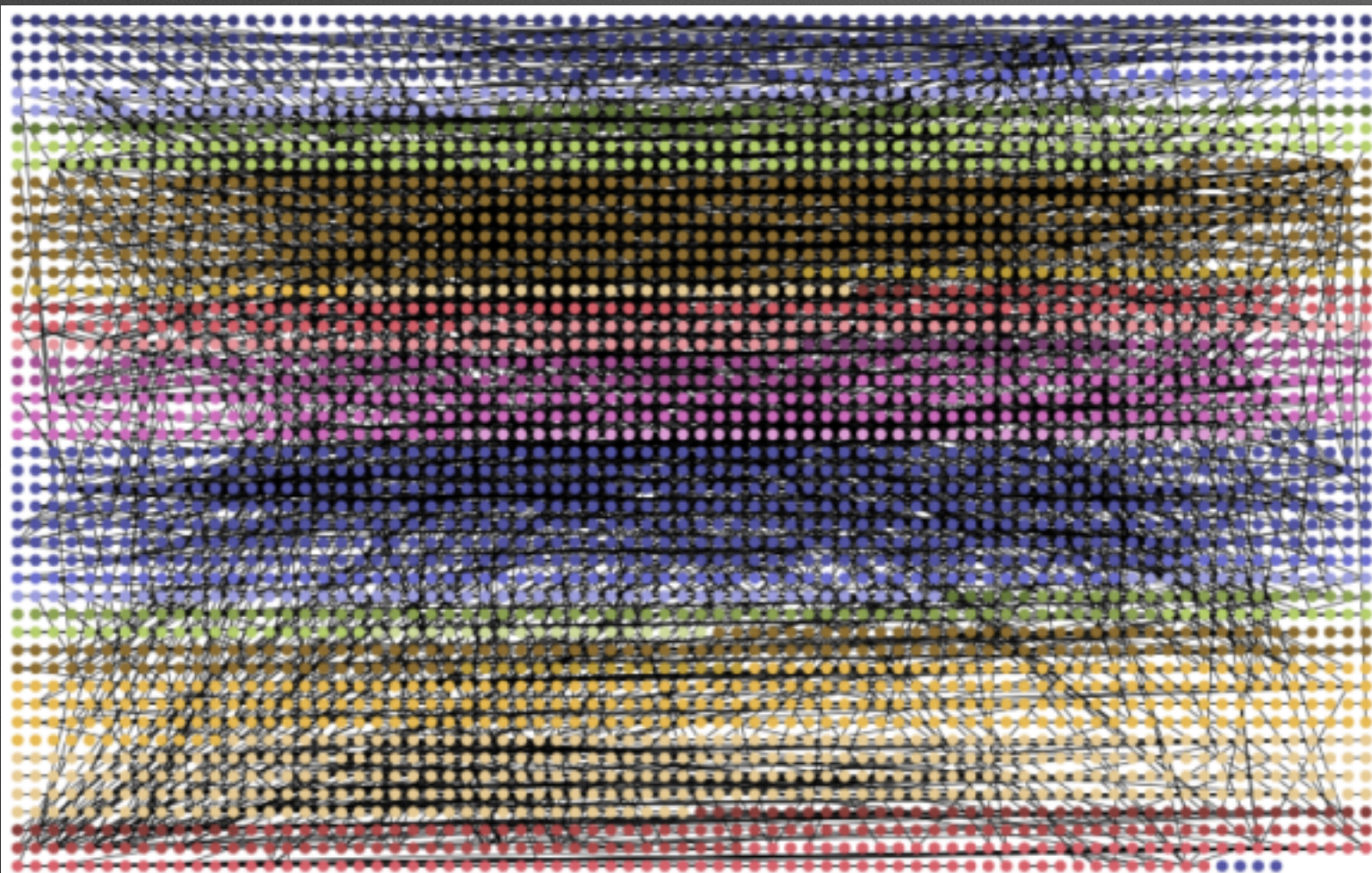
One hour about Basic

COPYRIGHT 1975 BY BILL GATES AND PAUL ALLEN

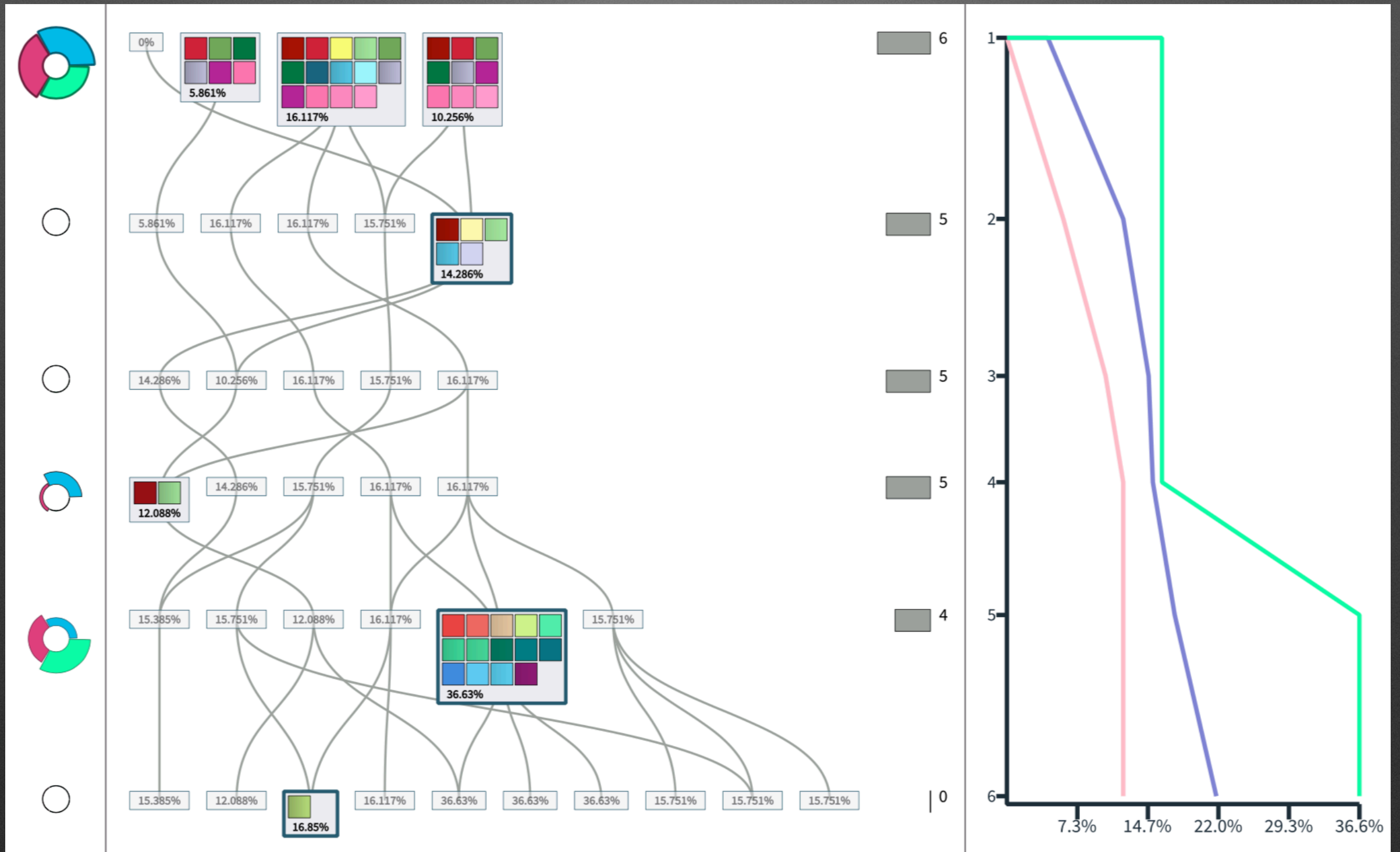
ORIGINALLY WRITTEN ON THE PDP-10 FROM
FEBRUARY 9 TO APRIL 9 1975

BILL GATES WROTE A LOT OF STUFF.
PAUL ALLEN WROTE A LOT OF OTHER STUFF AND FAST CODE.
MONTE DAVIDOFF WROTE THE MATH PACKAGE (F4I.MAC).

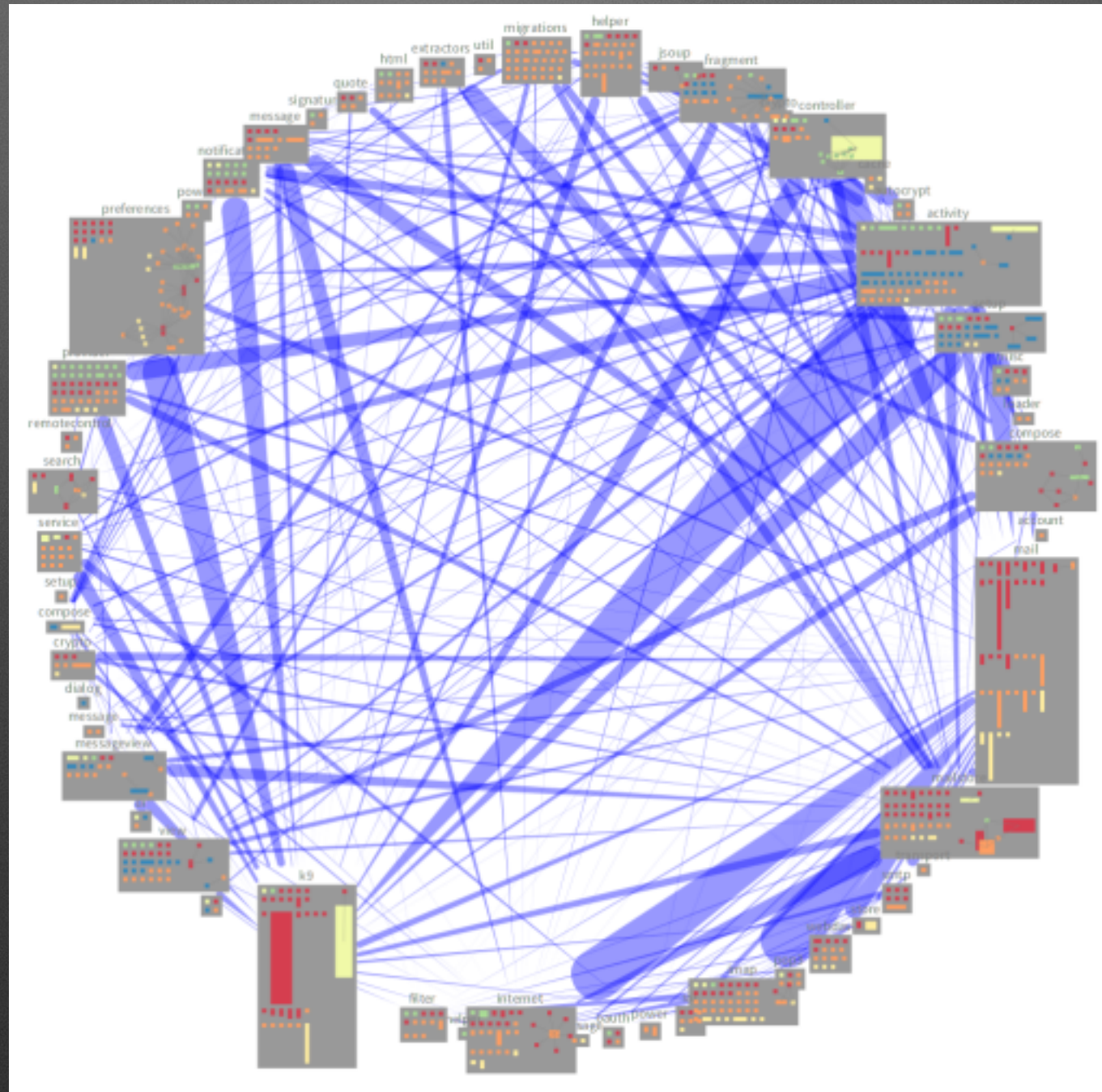
<https://pharowebly.wordpress.com/2020/05/24/roassal-1-hour-xp-assembly-code-of-gwbasic/>



Execution of IA generating tests



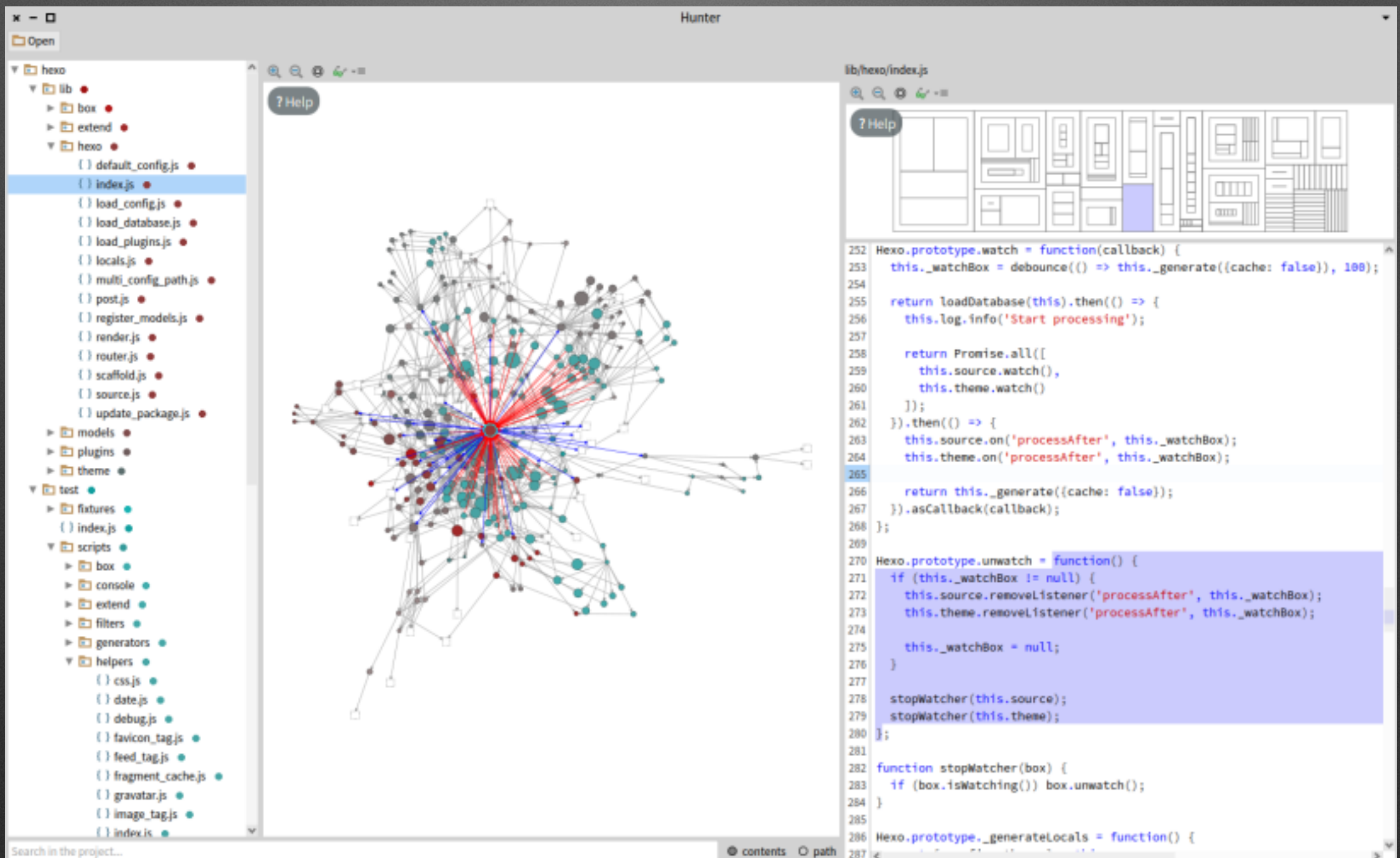
Analysis Android application



**Often developers
write their own tools**

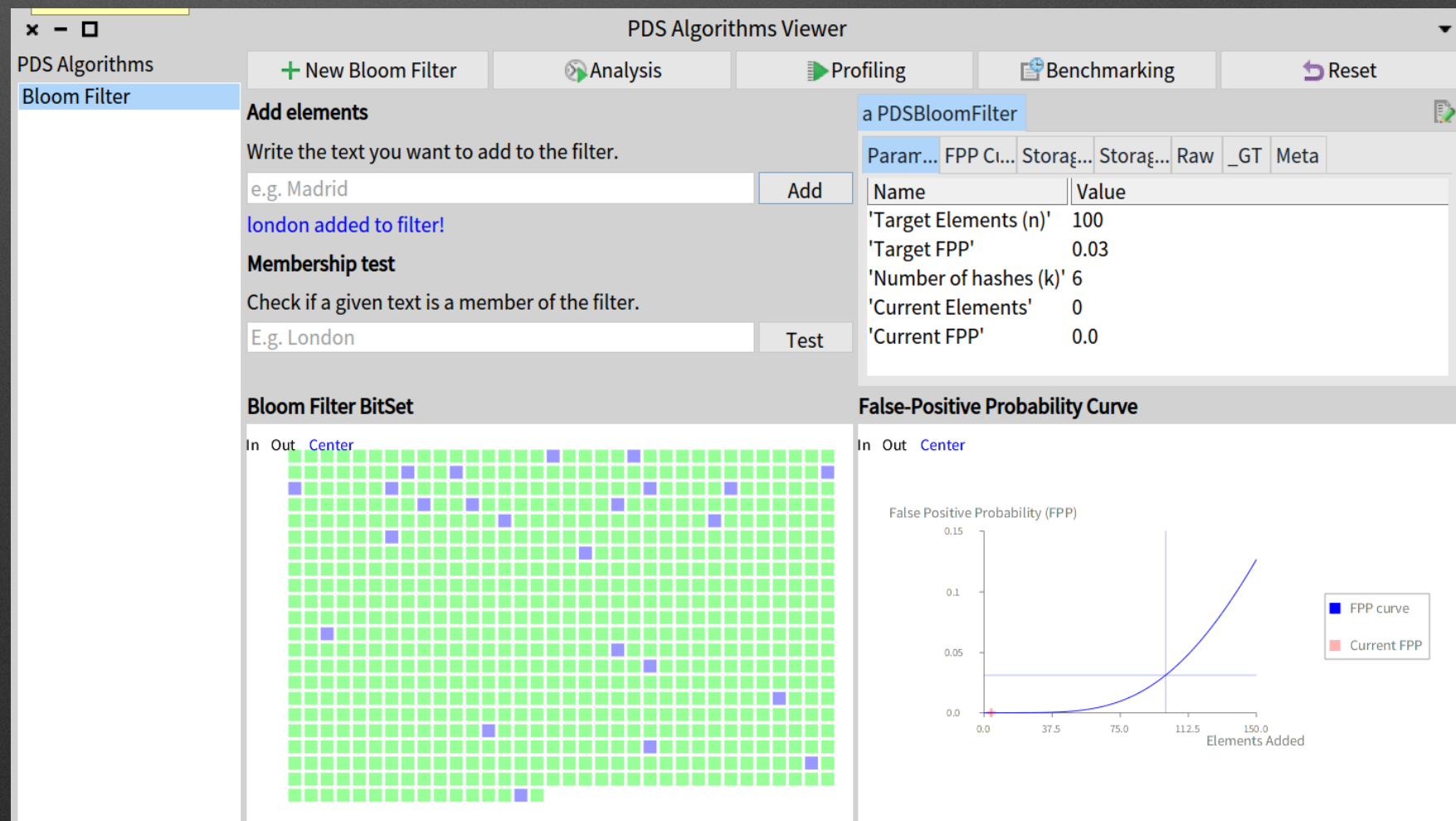
Building your own tool

- Example Javascript analysis



Probabilistic Data Structure

- <https://github.com/osoco/PharoPDS>
- Defined new data structure
- And the **analysis tools**



HTTP traffic analysis

- <http://youtu.be/rIBbeMdFCys>

The screenshot displays the Pharo IDE interface with several windows open for HTTP traffic analysis.

Pharo Image Window: Shows the Pharo logo and a list of packages in the Monticello Browser:

- ConfigurationOfHP35
- HP35-Calculator
- HP35-Seaside-Calculator

Monticello Browser Window: Shows a list of packages from a repository:

- HP35-Calculator-SvenVanCaekenberghe.17.mcz
- HP35-Calculator-SvenVanCaekenberghe.16.mcz
- HP35-Calculator-SvenVanCaekenberghe.15.mcz
- HP35-Calculator-SvenVanCaekenberghe.14.mcz
- HP35-Calculator-SvenVanCaekenberghe.13.mcz
- HP35-Calculator-SvenVanCaekenberghe.12.mcz
- HP35-Calculator-SvenVanCaekenberghe.11.mcz
- HP35-Calculator-SvenVanCaekenberghe.10.mcz
- HP35-Calculator-SvenVanCaekenberghe.9.mcz
- HP35-Calculator-SvenVanCaekenberghe.8.mcz
- HP35-Calculator-SvenVanCaekenberghe.7.mcz

Playground Window: Shows a list of announcements and a detailed view of a specific event.

Announcements Table:

Time	Announcement
2014-10-15T15:27:46.89322+0	2014-10-15 15:27:46 030 Connection Closed 128.93.162.53:80
2014-10-15T15:27:46.882202+	2014-10-15 15:27:46 029 GET /mc/SvenVanCaekenberghe/HP35/main/
2014-10-15T15:27:46.873225+	2014-10-15 15:27:46 028 Response Read a ZnResponse(200 OK text/
2014-10-15T15:27:46.764834+	2014-10-15 15:27:46 027 Request Written a ZnRequest(GET /mc/Sven
2014-10-15T15:27:46.749283+	2014-10-15 15:27:46 026 Connection Established smalltalkhub.com:80

Event Details: a ZnResponseReadEvent (2014-10-15 15:27:46 028 Response Read a ZnResponse(200 OK text/plain 1197B) 0ms)

Variable	Value
self	2014-10-15 15:27:46 028 Response Read a ZnResponse(200 OK text/plain 1197B) 0ms
clientid	nil
duration	0
id	28
response	a ZnResponse(200 OK text/plain 1197B)
timestamp	2014-10-15T15:27:46.873225+02:00

enter search query

Stepping ARM 64 bits asm

Untitled window

16r400	mov x0, x29	#[224 3 29 170]	lr	'16r0'	16r151E40	16r0
16r404	ret	#[192 3 95 214]	pc	'16r0'	16r151E48	16r0
16r408	mov x0, x28	#[224 3 28 170]	sp	'16r0'	16r151E50	16r0
16r40C	ret	#[192 3 95 214]	fp	'16r151E50'	16r151E58	16r0
16r410	str x10, [x28, #-8]!	#[138 143 31 248]	x0	'16r0'	16r151E60	16r0
16r414	ldr x10, #36	#[42 1 0 88]	x1	'16r0'	16r151E68	16r0
16r418	ldr x12, #40	#[76 1 0 88]	x2	classRegister '16r0'	16r151E70	16r151E50
16r41C	str x29, [x12]	#[157 1 0 249]	x3	'16r0'	16r151E78	16r151E40
16r420	mov x1, x28	#[225 3 28 170]	x4	'16r0'	16r151E80	16r0
16r424	adds x1, x1, #8	#[33 32 0 177]	x5	receiverRegister '16r0'	16r151E88	16r0
16r428	ldr x12, #32	#[12 1 0 88]			16r151E90	16r0
16r42C	str x1, [x12]	#[129 1 0 249]			16r151E98	16r0
16r430	ldr x10, [x28], #8	#[138 135 64 248]			16r151EA0	16r0
16r434	ret	#[192 3 95 214]			16r151EA8	16r0

Step

Disassemble at PC

Moments of grace...

**I want my halt to only
stop when called from
THAT test called testMe!**

mycode

self haltlf: #testMe

...

Use stack reification

Walk it

Halt if needed

(in 5 lines)

```
haltIf: aSelector  
| cntxt |  
cntxt := thisContext.  
[ cntxt isNil ] whileFalse: [  
    cntxt selector = aSelector  
    ifTrue: [ self halt ].  
    cntxt := cntxt sender ]
```


**Dynamically rewriting
deprecated calls @
runtime**

How to support migration to new versions

We deprecate API

How to help our users to migrate?

Rewriting deprecation

crLog: aString

self

deprecated: 'Please use trace* methods instead.'

transformWith:

'@receiver crLog: `@statements1'

-> '@receiver crTrace: `@statements1'.

self crTrace: aString

**Run your tests.
Your code and your
tests use the new
API!**

At notification time:

Walk the stack

Get caller AST

If should be rewritten

**Rewrite it and proceed
execution**

**Pharo is
research friendly**

International Research Groups

Lafhis (AR)

SCG (CH)

CAR (FR)

RMOD (FR)

Ummisco (IRD)

Reveal (CH)

Lysic (FR)

ENSTA-Bretagne (FR)

CEA-List (FR)

Ryerson (CAN)

OC (FR)

CCMI-FIT (CZ)

ASERG (BR)

Pleiad (CL)

Macau (UNO)

Cirad (FR)

USTH (Vietnam)

Soft-Qual (Serbia)

Uni. Quilmes (AR)

ENIT (FR)

Cochabamba (Bo)

Maroua (CAM)

ETS (CAN)

**We are ready
to help you
validate
your ideas**

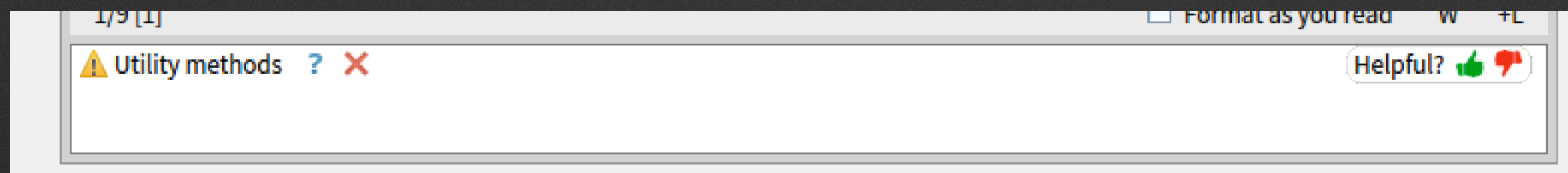
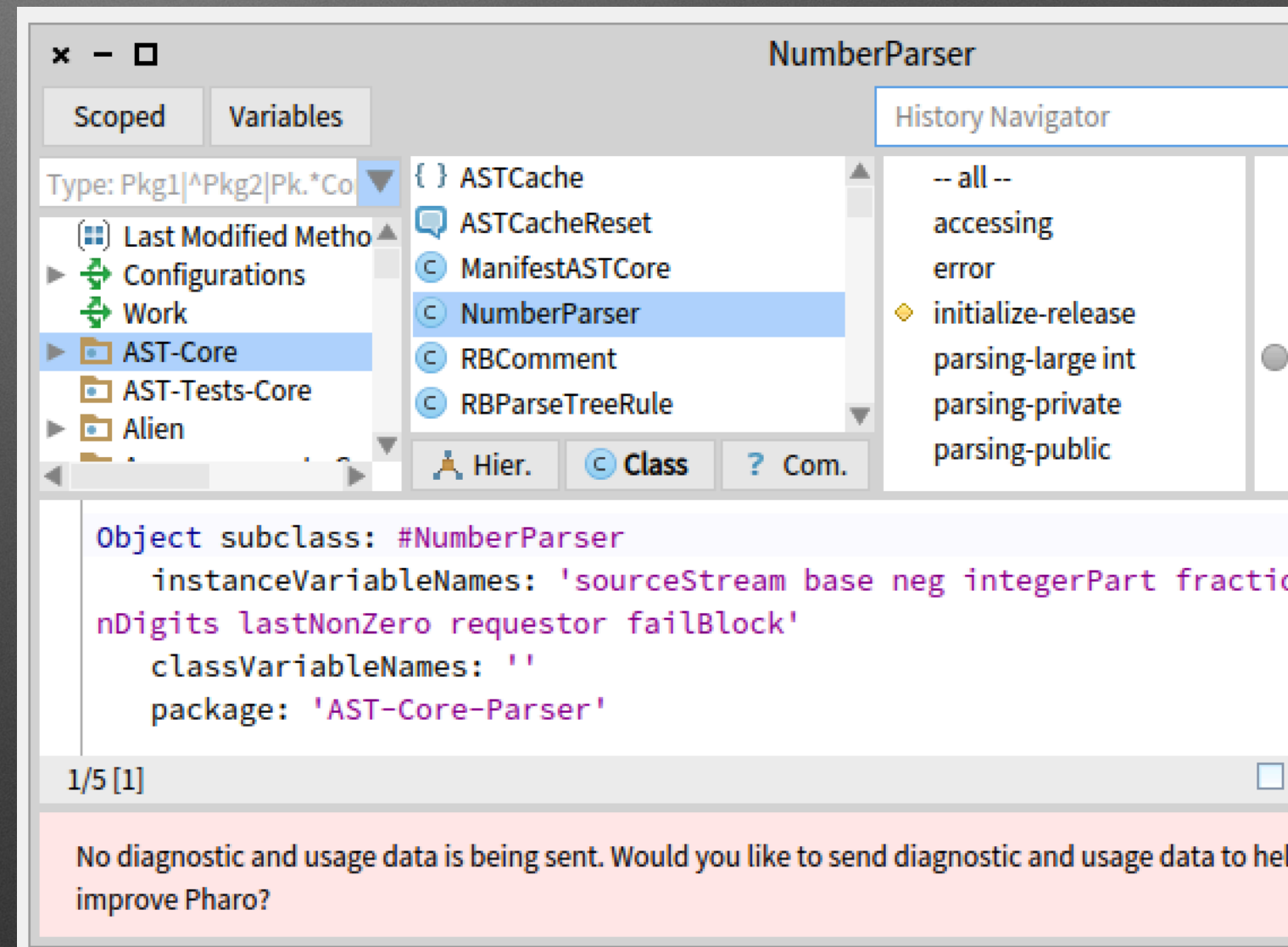
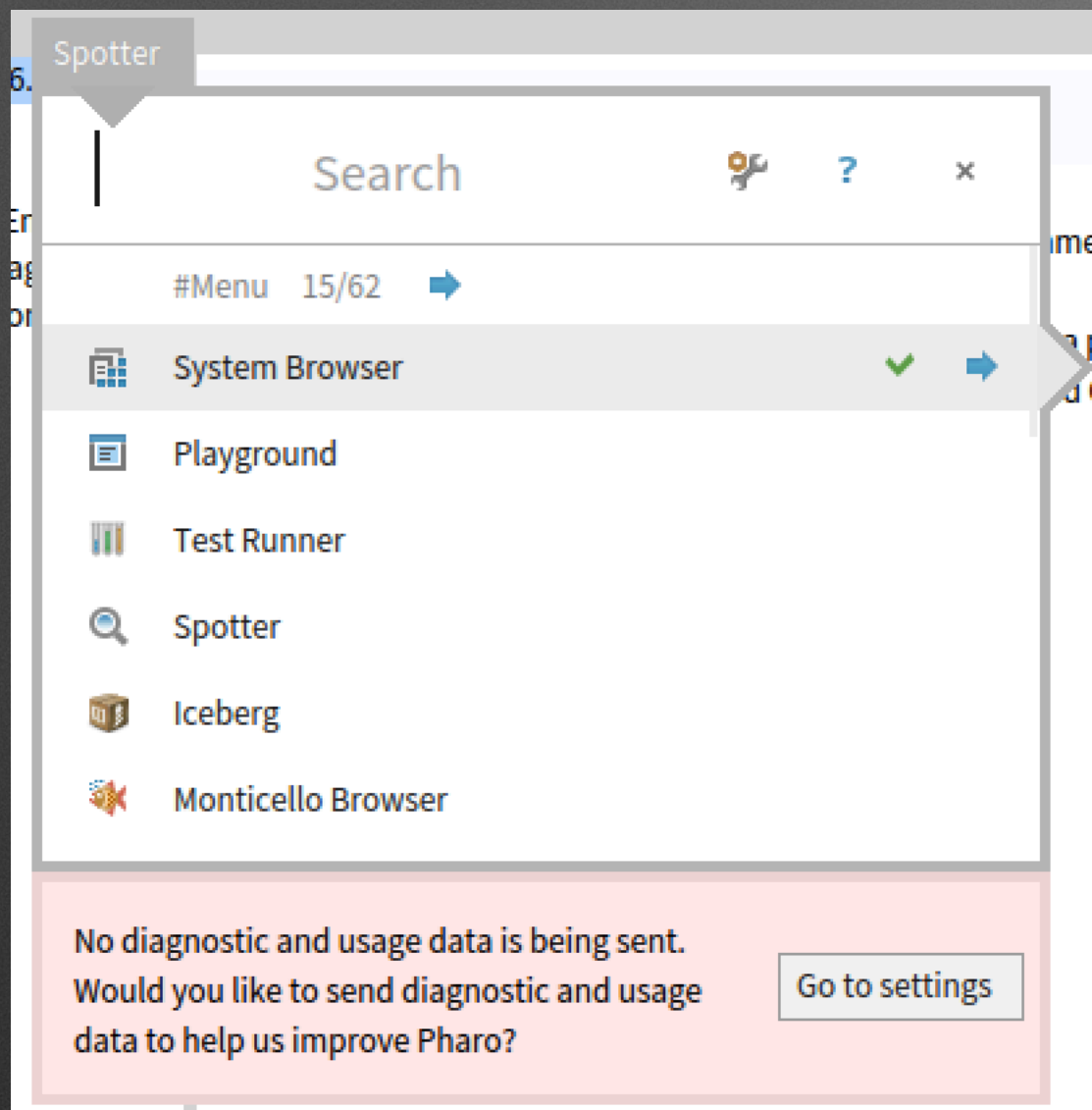
We are interested in

Tools, tests, refactorings, program transformation, visualisation, merge, code review, debugging, test selection, migration, navigation, IDE, code browsing, DSL, recommender, profiler, specific datastructure, type inference, code optimizers, ...

**We can be
your guinea
pigs,...**

(well kind of.... we have real users)

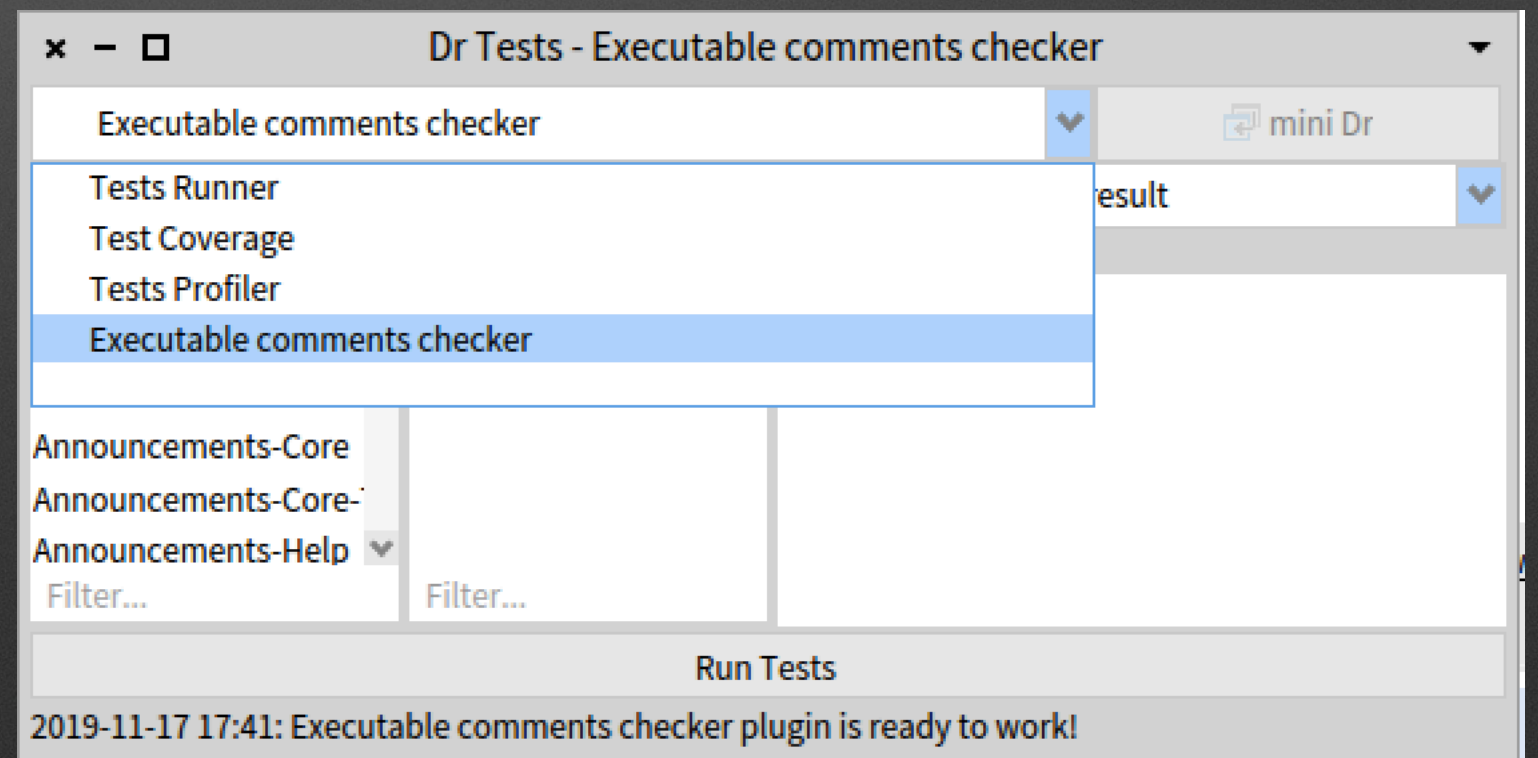
Actively supporting research eg. SCG from Uni. Berne



Best Paper Award @ ICPC'19

**Kubelka, Bergel, Robbes,
“Live Programming and Software Evolution: Questions
during a Programming Change Task”**

DrTests: a plugin-based architecture to plug test analyses



**We validated our 27000
tests for Rotten Green
Tests (ICSE'19)**

Test Amplification

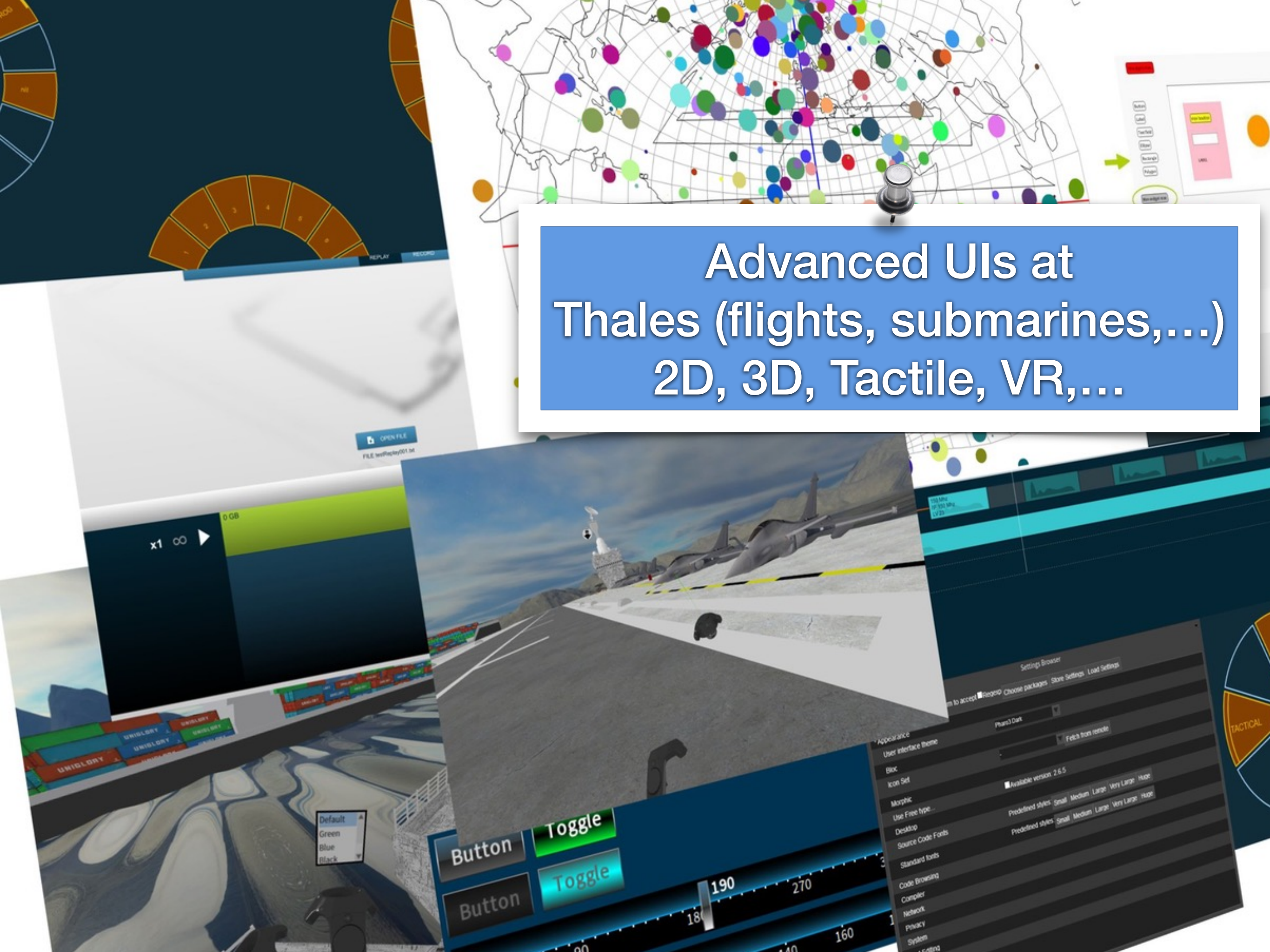
**by S. Demeyer, H. Rocha, M. Abdi of Antwerp
Universiteit**

Map reduce debugging

by M. Marra and Prof. E. Gonzales Boix from Vrije
Universiteit Brussel

Code Review

by A. Bachelli, A. Bergel / ObjectProfile



Advanced UIs at Thales (flights, submarines,...) 2D, 3D, Tactile, VR,...

Empowering is the right word

The immersive programming experience

Pharo is a pure object-oriented programming language *and* a powerful environment, focused on simplicity and immediate feedback (think IDE and OS rolled into one).

- Pharo is an energizing and creative environment
- Moldable tools are powerful
- Tried to share my feeling
- But “The idea of experience does not replace experience.” Alain

Discover

Learn more about Pharo's key
features and elegant design

Download

Download latest version (8.0)!
Read more about [here](#)

Learn

Access the Pharo Mooc!
3000 people registered and follow the
Pharo Mooc. You can find it [here](#).

A scenic view of a lighthouse on a cliff overlooking the ocean. The lighthouse is white with a black top, situated on a green, rocky cliff. The ocean is blue with white waves crashing against the shore. The sky is clear and blue.

Fun, simple

Pure & elegant

Productive

Empowering

Addictive

Full access