Proposals for the Reborn Pharo Developer

Simon Denier, Damien Pollet, Stéphane Ducasse

My name is Simon Denier

and I have nothing to show

What do I have?

Some backstory

I’m a newcomer to Smalltalk

less than a year

Did my time in Java

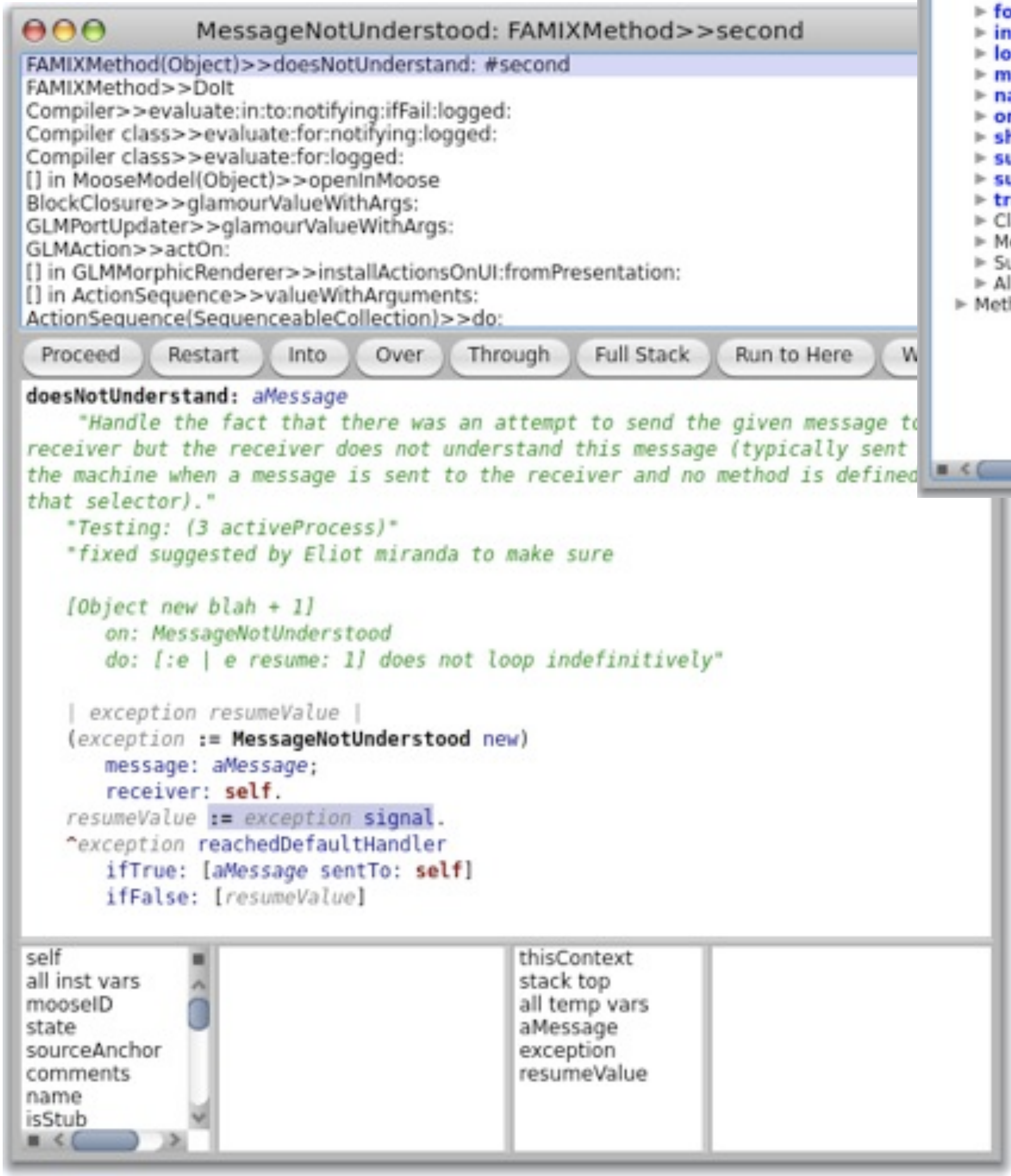
5 years with eclipse...

So it’s quite a shock

And sometimes I wonder

“what the f...?”

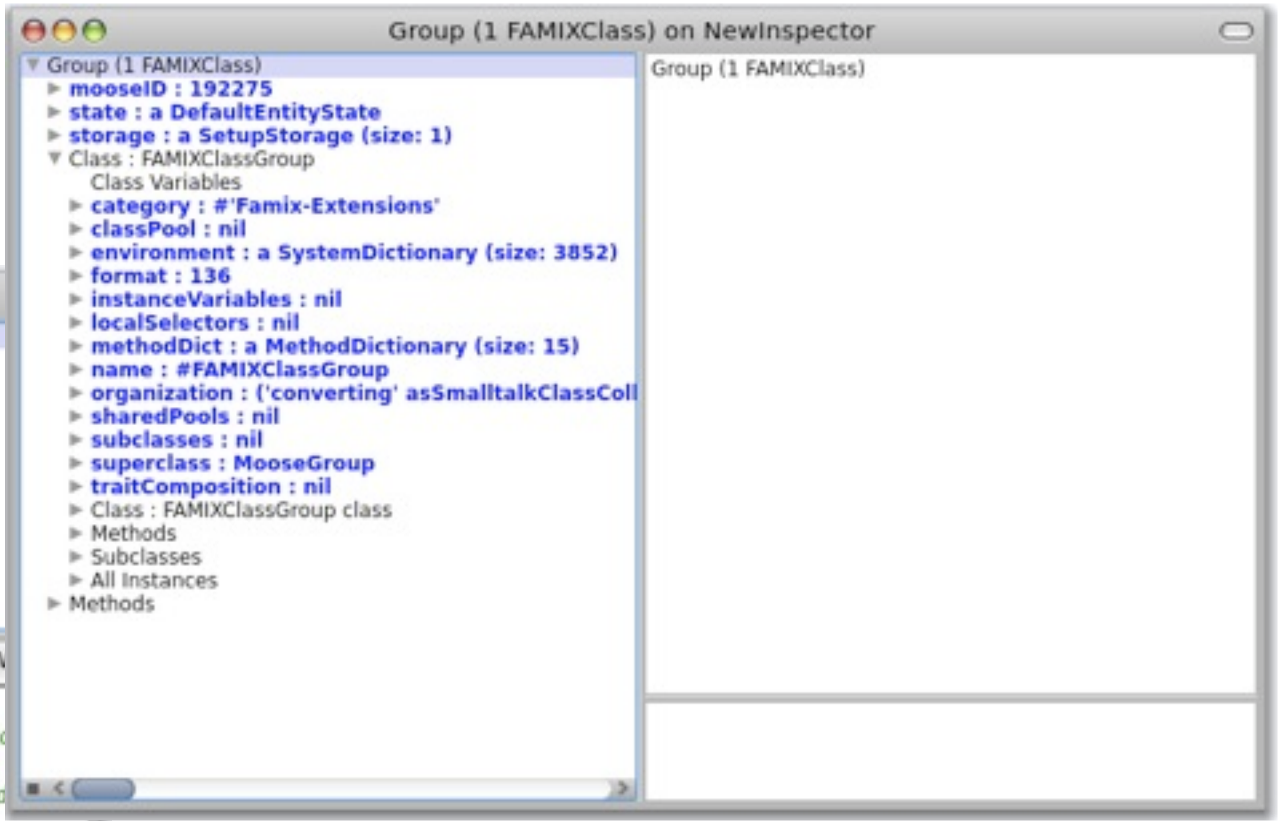
Good stuff



```
FAMIXMethod(Object)>>doesNotUnderstand: #second
FAMIXMethod>>DoIt
Compiler>>evaluate:in:to:notifying:ifFail:logged:
Compiler class>>evaluate:for:notifying:logged:
Compiler class>>evaluate:for:logged:
[] in MooseModel(Object)>>openInMoose
BlockClosure>>glamourValueWithArgs:
GLMPortUpdater>>glamourValueWithArgs:
GLMAction>>actOn:
[] in GLMMorphicRenderer>>installActionsOnUI:fromPresentation:
[] in ActionSequence>>valueWithArguments:
ActionSequence(SequenceableCollection)>>do:
```

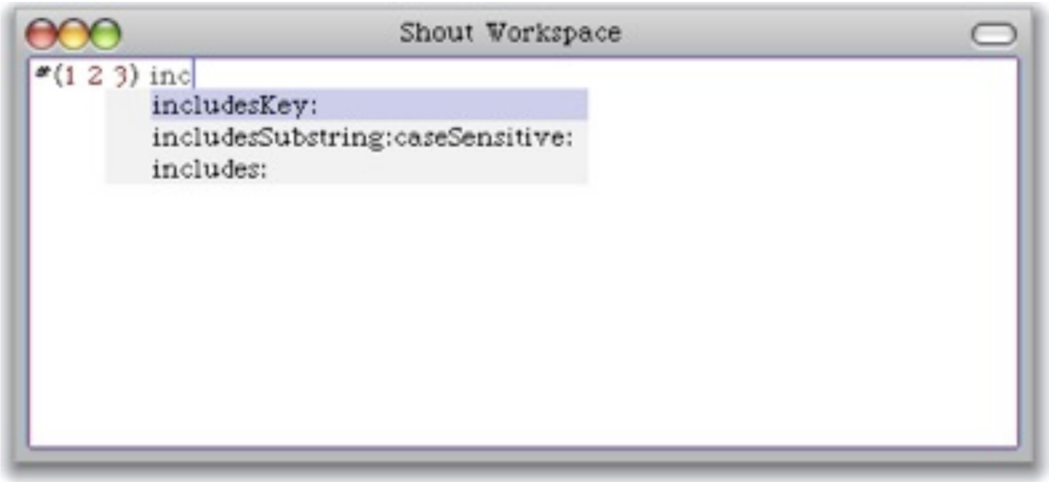
```
doesNotUnderstand: aMessage
	"Handle the fact that there was an attempt to send the given message to
receiver but the receiver does not understand this message (typically sent
the machine when a message is sent to the receiver and no method is defined
that selector)."
	"Testing: (3 activeProcess)"
	"fixed suggested by Eliot miranda to make sure

	[Object new blah + 1]
		on: MessageNotUnderstood
		do: [:e | e resume: 1] does not loop indefinitively"

	| exception resumeValue |
	(exception := MessageNotUnderstood new)
		message: aMessage;
		receiver: self.
	resumeValue := exception signal.
	^exception reachedDefaultHandler
		ifTrue: [aMessage sentTo: self]
		ifFalse: [resumeValue]
```

```
self            thisContext
all inst vars   stack top
mooseID         all temp vars
state           aMessage
sourceAnchor    exception
comments        resumeValue
name
isStub
```



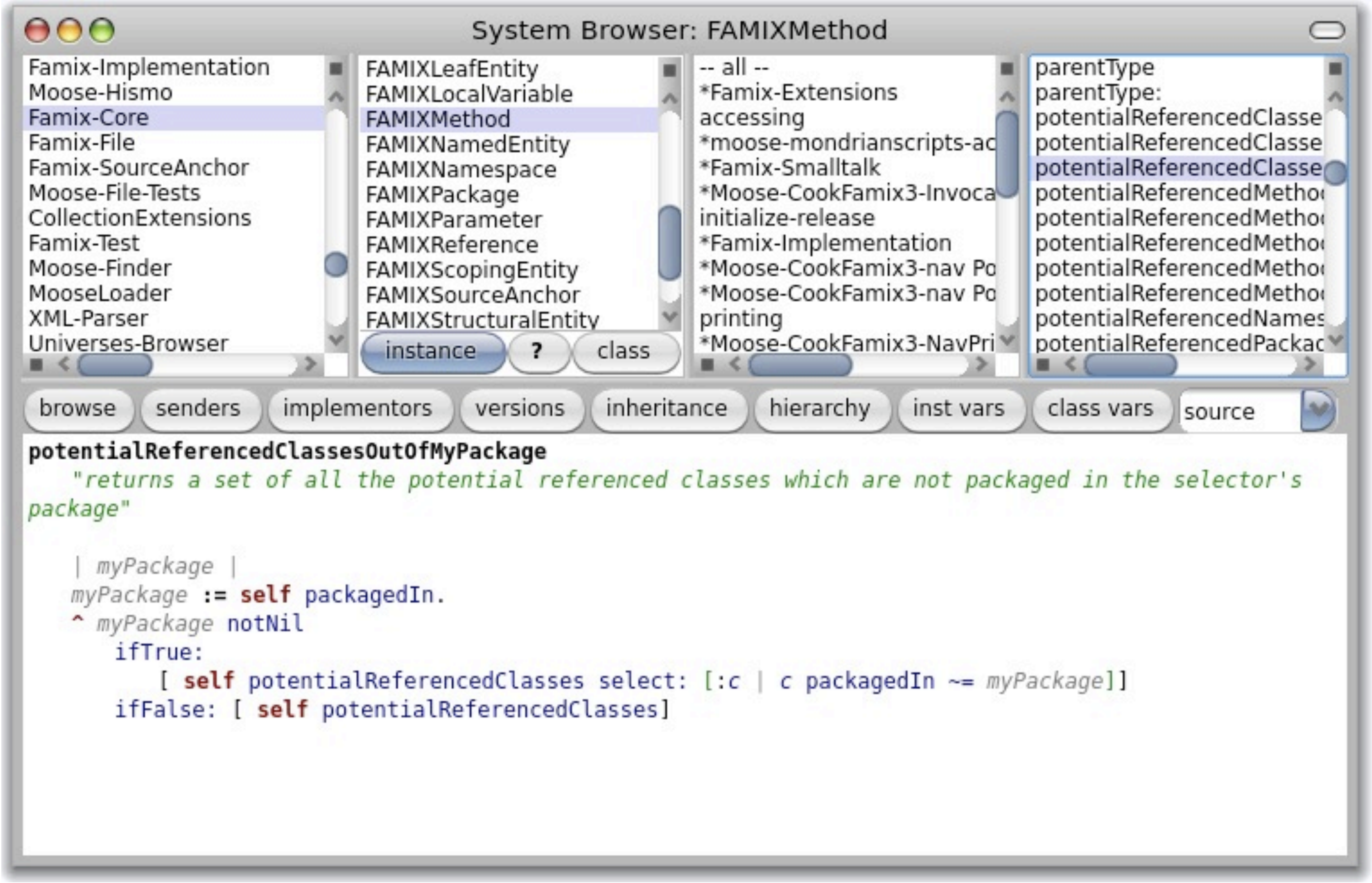
```
▼ Group (1 FAMIXClass)
  ▶ mooseID : 192275
  ▶ state : a DefaultEntityState
  ▶ storage : a SetupStorage (size: 1)
  ▼ Class : FAMIXClassGroup
      Class Variables
    ▶ category : #'Famix-Extensions'
    ▶ classPool : nil
    ▶ environment : a SystemDictionary (size: 3852)
    ▶ format : 136
    ▶ instanceVariables : nil
    ▶ localSelectors : nil
    ▶ methodDict : a MethodDictionary (size: 15)
    ▶ name : #FAMIXClassGroup
    ▶ organization : ('converting' asSmalltalkClassColl
    ▶ sharedPools : nil
    ▶ subclasses : nil
    ▶ superclass : MooseGroup
    ▶ traitComposition : nil
    ▶ Class : FAMIXClassGroup class
    ▶ Methods
    ▶ Subclasses
    ▶ All Instances
  ▶ Methods
```

Group (1 FAMIXClass)



```
#(1 2 3) inc
      includesKey:
      includesSubstring:caseSensitive:
      includes:
```

Not so good stuff



```
potentialReferencedClassesOutOfMyPackage
    "returns a set of all the potential referenced classes which are not packaged in the selector's
package"

    | myPackage |
    myPackage := self packagedIn.
    ^ myPackage notNil
        ifTrue:
            [ self potentialReferencedClasses select: [:c | c packagedIn ~= myPackage]]
        ifFalse: [ self potentialReferencedClasses]
```

So I have some ideas

and I want yours

Focus

When I code

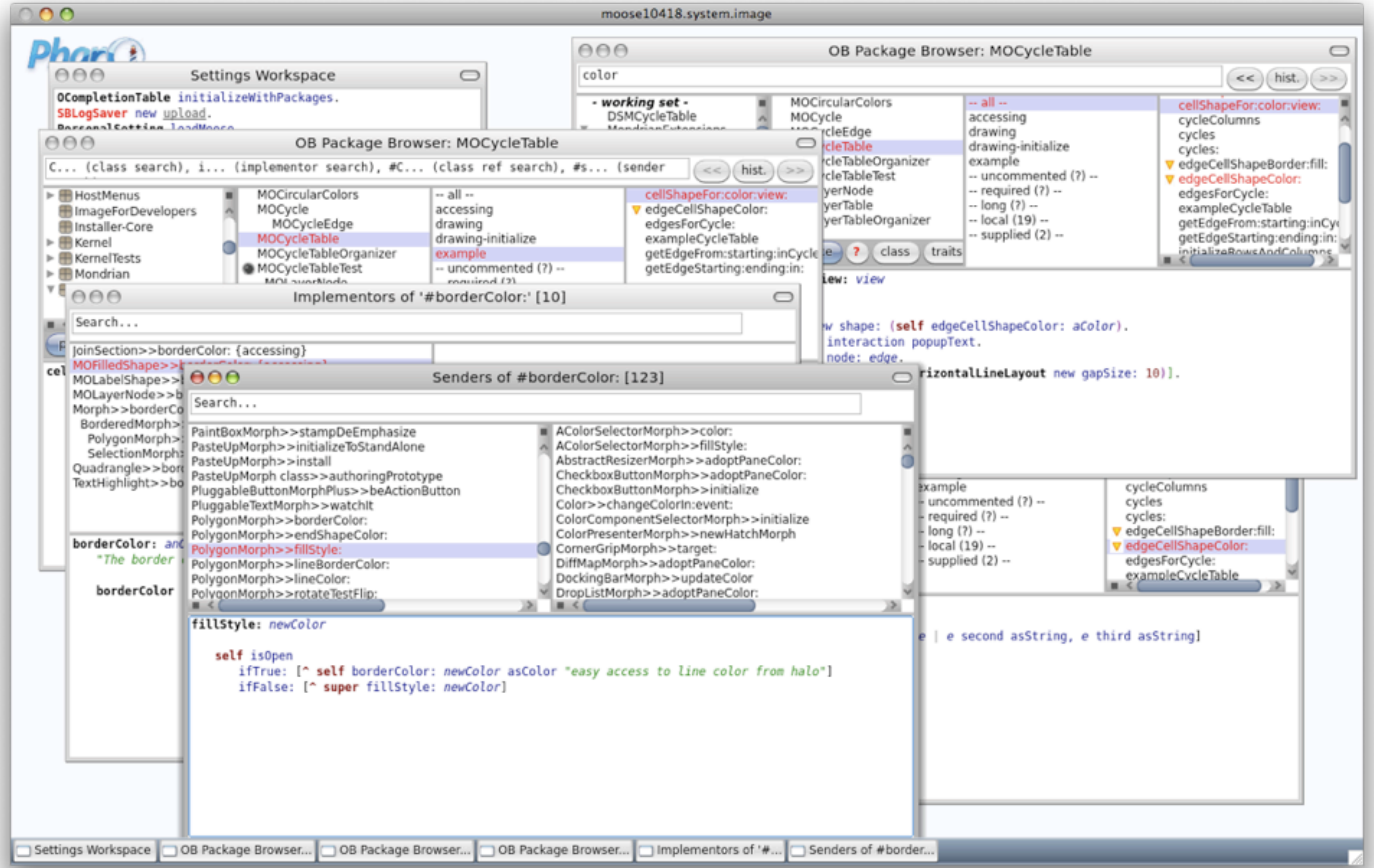
I do one thing at a time

Coding is task-oriented

I may browse the system

But I always come back to a few classes

Unfortunately



moose10418.system.image

Settings Workspace

```
OCompletionTable initializeWithPackages.
SBLogSaver new upload.
PersonalSetting loadMoose
```

OB Package Browser: MOCycleTable

color

<< hist. >>

- working set -	MOCircularColors	-- all --	cellShapeFor:color:view:
DSMCycleTable	MOCycle	accessing	cycleColumns
MondrianExtensions	cleEdge	drawing	cycles
	cleTable	drawing-initialize	cycles:
	cleTableOrganizer	example	edgeCellShapeBorder:fill:
	cleTableTest	-- uncommented (?) --	edgeCellShapeColor:
	yerNode	-- required (?) --	edgesForCycle:
	yerTable	-- long (?) --	exampleCycleTable
	yerTableOrganizer	-- local (19) --	getEdgeFrom:starting:inCy
		-- supplied (2) --	getEdgeStarting:ending:in:
			initializeRowsAndColumns

? class traits

```
iew: view

w shape: (self edgeCellShapeColor: aColor).
 interaction popupText.
 node: edge.
rizontalLineLayout new gapSize: 10)].
```

OB Package Browser: MOCycleTable

C... (class search), i... (implementor search), #C... (class ref search), #s... (sender

<< hist. >>

HostMenus	MOCircularColors	-- all --	cellShapeFor:color:view:
ImageForDevelopers	MOCycle	accessing	edgeCellShapeColor:
Installer-Core	MOCycleEdge	drawing	edgesForCycle:
Kernel	MOCycleTable	drawing-initialize	exampleCycleTable
KernelTests	MOCycleTableOrganizer	example	getEdgeFrom:starting:inCycle
Mondrian	MOCycleTableTest	-- uncommented (?) --	getEdgeStarting:ending:in:
	MOLayerNode	-- required (?) --	

Implementors of '#borderColor:' [10]

Search...

JoinSection>>borderColor: {accessing}
MOFilledShape>>borderColor: {accessing}
MOLabelShape>>b
MOLayerNode>>b
Morph>>borderCo
BorderedMorph>
PolygonMorph>
SelectionMorph
Quadrangle>>bor
TextHighlight>>bo

```
borderColor: an
   "The border

   borderColor
```

Senders of #borderColor: [123]

Search...

PaintBoxMorph>>stampDeEmphasize	AColorSelectorMorph>>color:
PasteUpMorph>>initializeToStandAlone	AColorSelectorMorph>>fillStyle:
PasteUpMorph>>install	AbstractResizerMorph>>adoptPaneColor:
PasteUpMorph class>>authoringPrototype	CheckboxButtonMorph>>adoptPaneColor:
PluggableButtonMorphPlus>>beActionButton	CheckboxButtonMorph>>initialize
PluggableTextMorph>>watchIt	Color>>changeColorIn:event:
PolygonMorph>>borderColor:	ColorComponentSelectorMorph>>initialize
PolygonMorph>>endShapeColor:	ColorPresenterMorph>>newHatchMorph
PolygonMorph>>fillStyle:	CornerGripMorph>>target:
PolygonMorph>>lineBorderColor:	DiffMapMorph>>adoptPaneColor:
PolygonMorph>>lineColor:	DockingBarMorph>>updateColor
PolygonMorph>>rotateTestFlip:	DropListMorph>>adoptPaneColor:

```
fillStyle: newColor

   self isOpen
      ifTrue: [^ self borderColor: newColor asColor "easy access to line color from halo"]
      ifFalse: [^ super fillStyle: newColor]
```

xample	cycleColumns
-- uncommented (?) --	cycles
-- required (?) --	cycles:
-- long (?) --	edgeCellShapeBorder:fill:
-- local (19) --	edgeCellShapeColor:
-- supplied (2) --	edgesForCycle:
	exampleCycleTable

```
e | e second asString, e third asString]
```

Settings Workspace | OB Package Browser... | OB Package Browser... | OB Package Browser... | Implementors of '#... | Senders of #border...

Working set

Set of interesting items

Selected items

Unsaved items

History items

Tools built around the working set

Working set
=
the new workhorse

Ubiquity

Select `text` then...

do it, print it,
inspect it, debug it...

browse class, senders, implementors

It’s a marvellous thing

Unfortunately

Not every interaction is so seamless

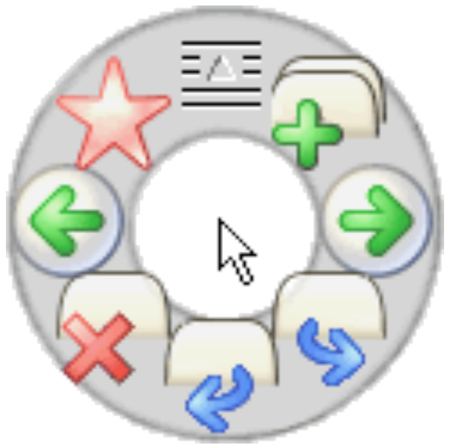
Menus are
less accessible
than toolbars

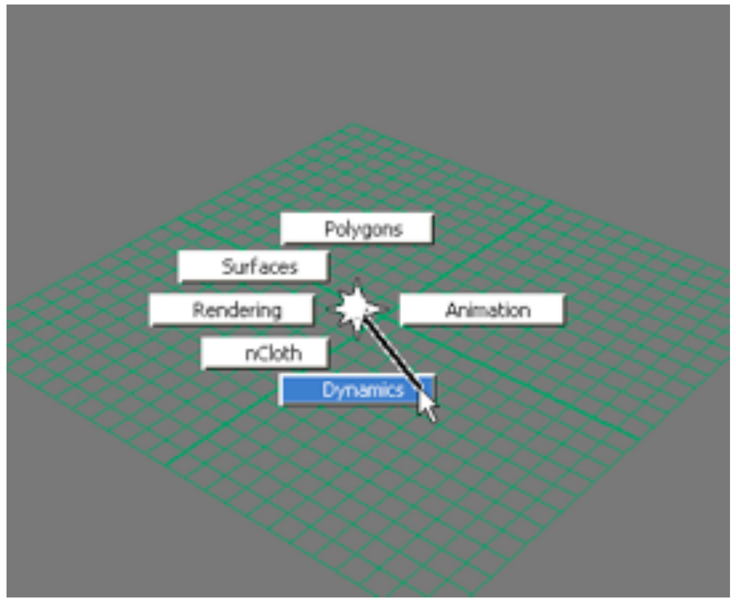
Menus are cumbersome for most-used actions

Menus become cluttered with many items

What is better?

Pie menus





Good for most used items

Good with mouse gestures

Hyperlink

Semi-modal hyperlinks in text pane

control+click browse definition/implementors

control+alt+click browse references/senders

Ubiquity means seamless interaction everywhere

Navigation

So ubiquity is cool for browsing code

but what about focus?

Did you try browsing senders of #=?

Do you want implementors of #new in package?

We need focus for navigation and search

Look for
senders of #=
in this class

Look for implementors of #new in package

Look for class definition of String in the system

Look for methods of Collection in its hierarchy

Do you see the pattern?

Look for *aspect*
of *target*
in *scope*

Look for *senders*
of #=
in *this class*

Look for *implementors*
of *#new*
in *package*

Look for *class definition* of *String* in *the system*

Look for *methods*
of *Collection*
in *its hierarchy*

(all?) search can be expressed in this wannabe API

Remember

focus+ubiquity+navigation

=

new Pharo experience!