UNIVERSITÄT
BERN

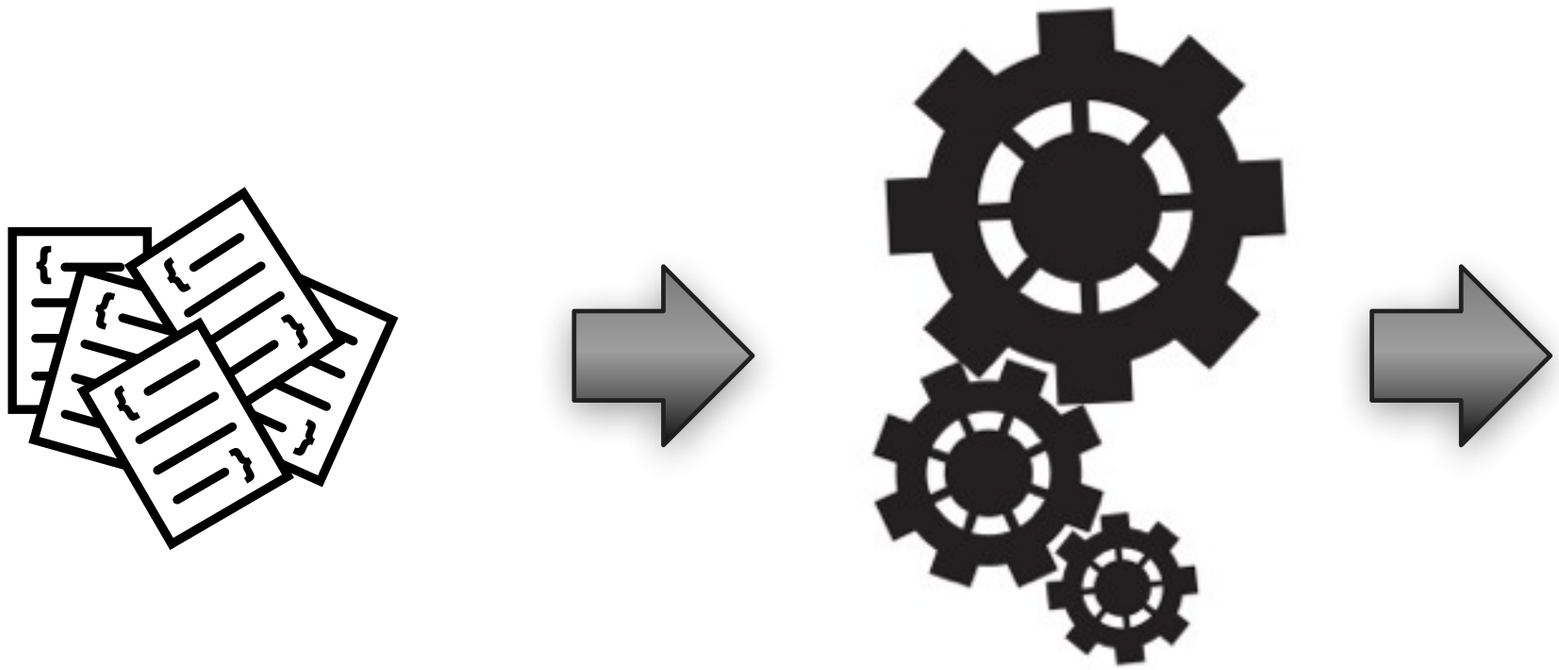
Object-Centric Reflection

Jorge Ressia
Software Composition Group

Profiling

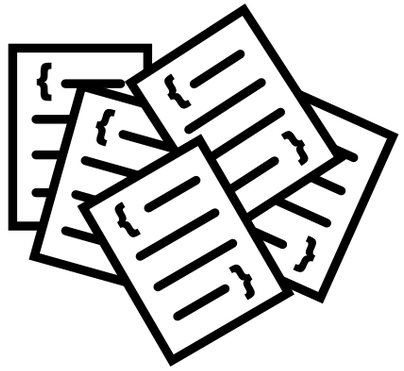
Profiling:
Is the activity of analyzing a program execution.

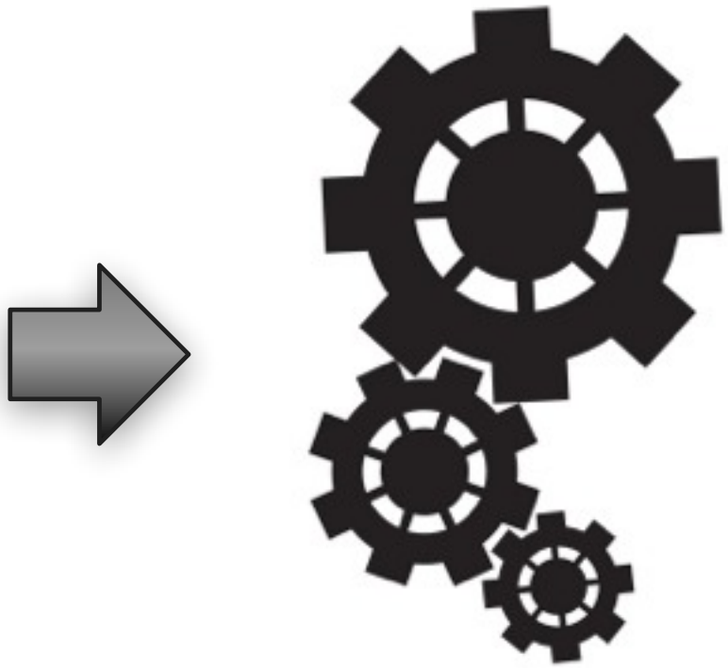
Profile



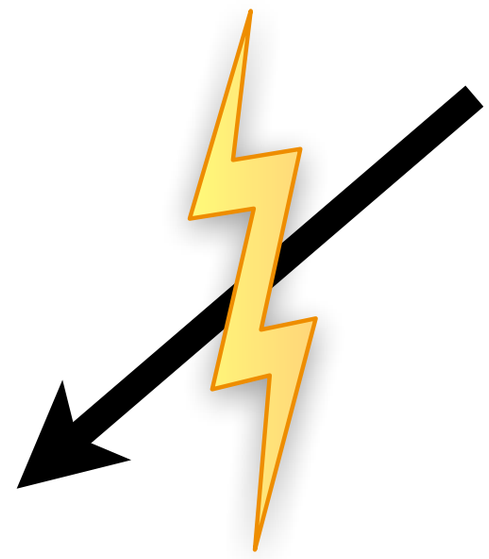
```
54.8% {11501ms} MOCanvas>>drawOn:
  54.8% {11501ms} MORoot(MONode)>>displayOn:
   30.9% {6485ms} MONode>>displayOn:
     | 18.1% {3799ms} MOEdge>>displayOn:
         ...
     |  8.4% {1763ms} MOEdge>>displayOn:
     |     | 8.0% {1679ms} MOStraightLineShape>>display:on:
     |     |  2.6% {546ms} FormCanvas>>line:to:width:color:
         ...
   23.4% {4911ms} MOEdge>>displayOn:
         ...
```

Profile



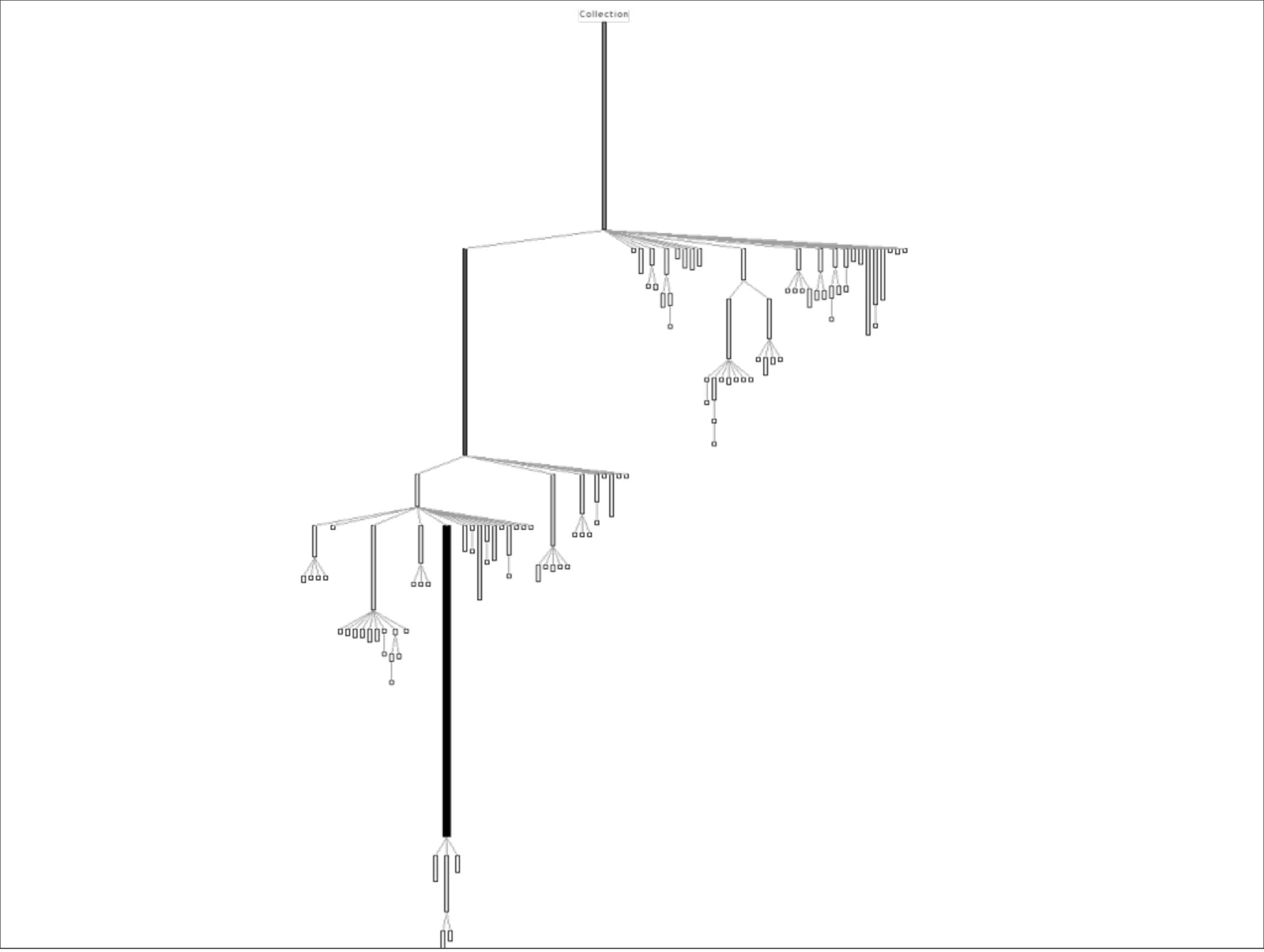


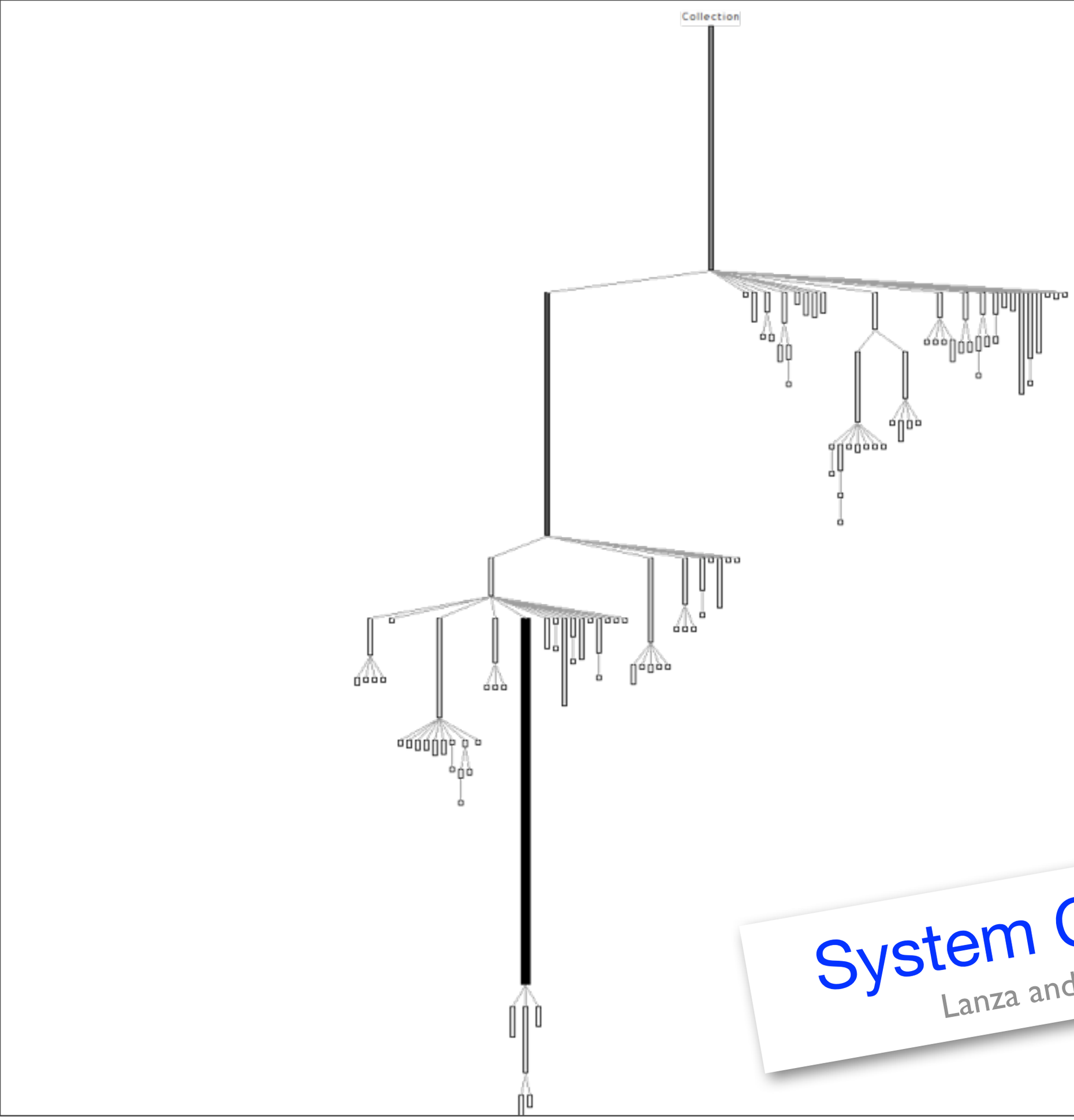
```
54.8% {11501ms} MOCanvas>>drawOn:
  54.8% {11501ms} MORoot(MONode)>>displayOn:
   30.9% {6485ms} MONode>>displayOn:
     | 18.1% {3799ms} MOEdge>>displayOn:
         ...
     |  8.4% {1763ms} MOEdge>>displayOn:
     |     | 8.0% {1679ms} MOStraightLineShape>>display:on:
     |     |  2.6% {546ms} FormCanvas>>line:to:width:color:
         ...
   23.4% {4911ms} MOEdge>>displayOn:
         ...
```



Domain

Mondrian


Collection



System Complexity

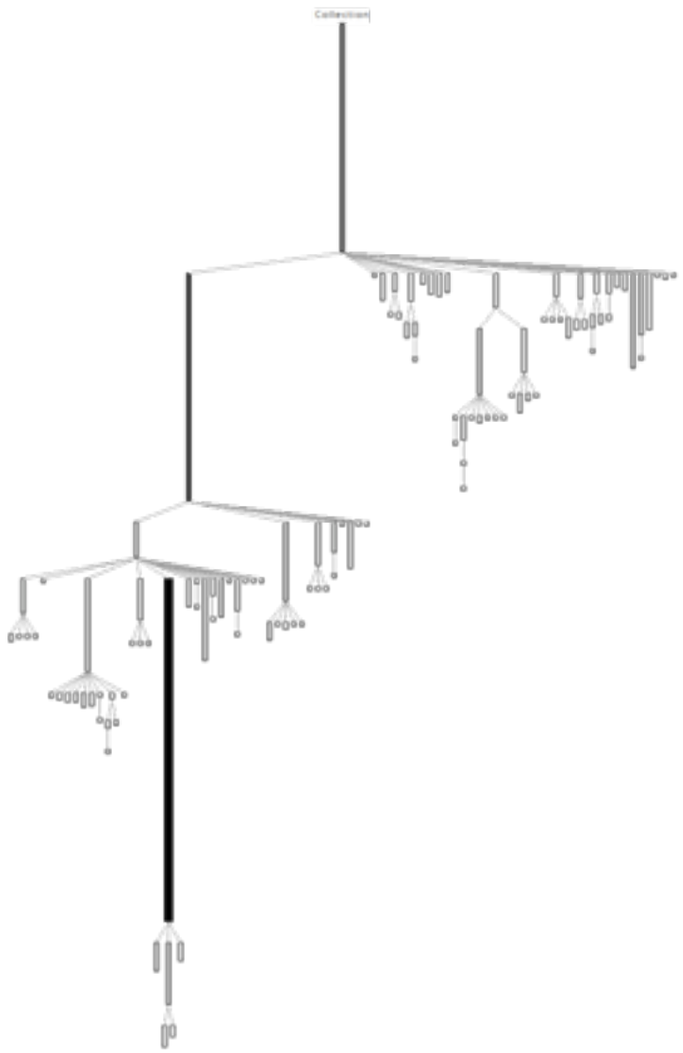
Lanza and Ducasse 2003

```
54.8% {11501ms} MOCanvas>>drawOn:
  54.8% {11501ms} MORoot(MONode)>>displayOn:
   30.9% {6485ms} MONode>>displayOn:
      | 18.1% {3799ms} MOEdge>>displayOn:
          ...
      |  8.4% {1763ms} MOEdge>>displayOn:
      |    | 8.0% {1679ms} MOStraightLineShape>>display:on:
      |    |  2.6% {546ms} FormCanvas>>line:to:width:color:
        ...
  23.4% {4911ms} MOEdge>>displayOn:
        ...
```

Which is the relationship?

```
54.8% {11501ms} MOCanvas>>drawOn:
  54.8% {11501ms} MORoot(MONode)>>displayOn:
   30.9% {6485ms} MONode>>displayOn:
      | 18.1% {3799ms} MOEdge>>displayOn:
        ...
      |  8.4% {1763ms} MOEdge>>displayOn:
      |     | 8.0% {1679ms} MOStraightLineShape>>display:on:
      |     |  2.6% {546ms} FormCanvas>>line:to:width:color:
        ...
  23.4% {4911ms} MOEdge>>displayOn:
        ...
```

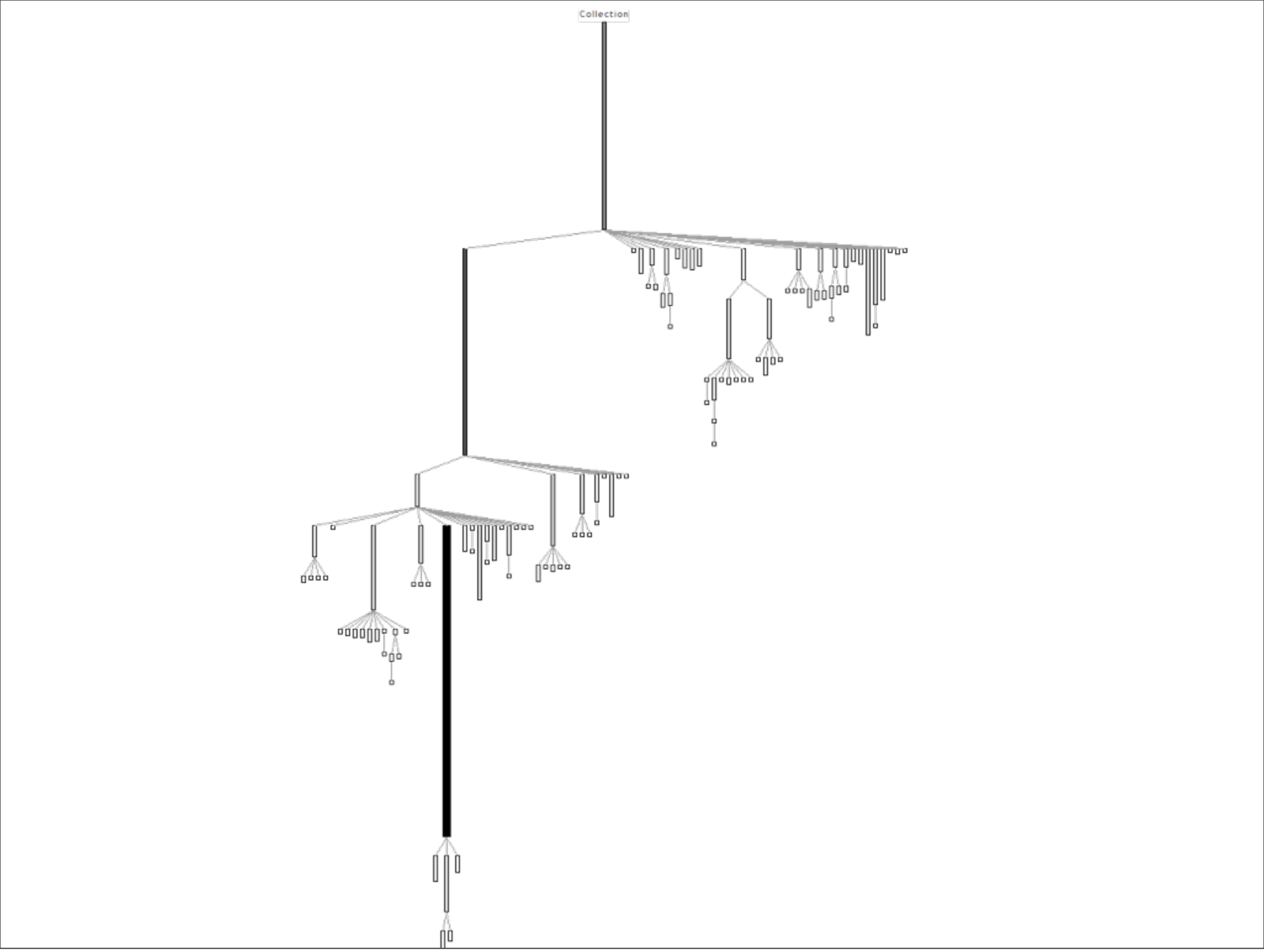
?

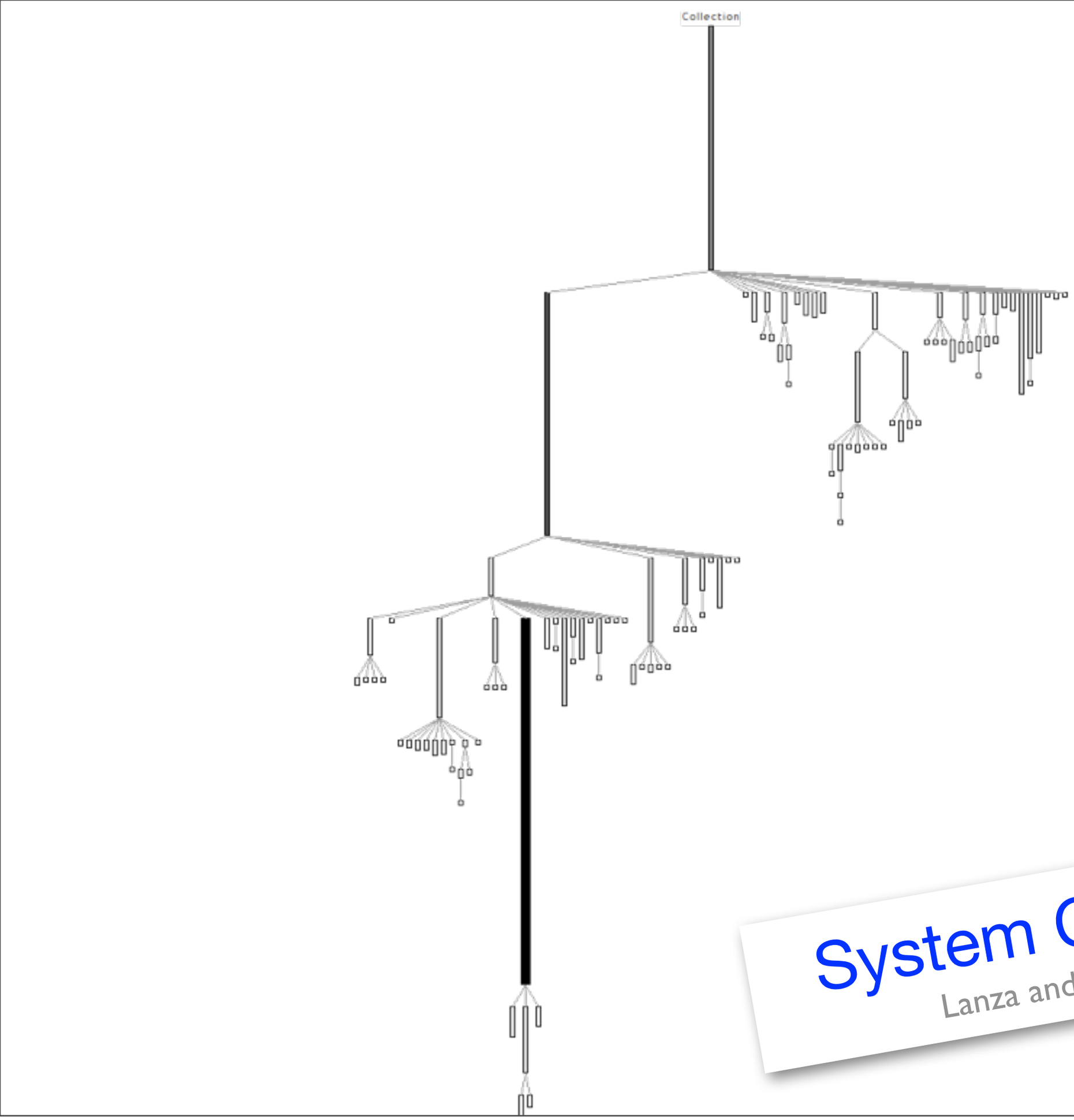


Debugging

Debugging:
Is the process of interacting with a running software system to test and understand its current behavior.

Mondrian


Collection

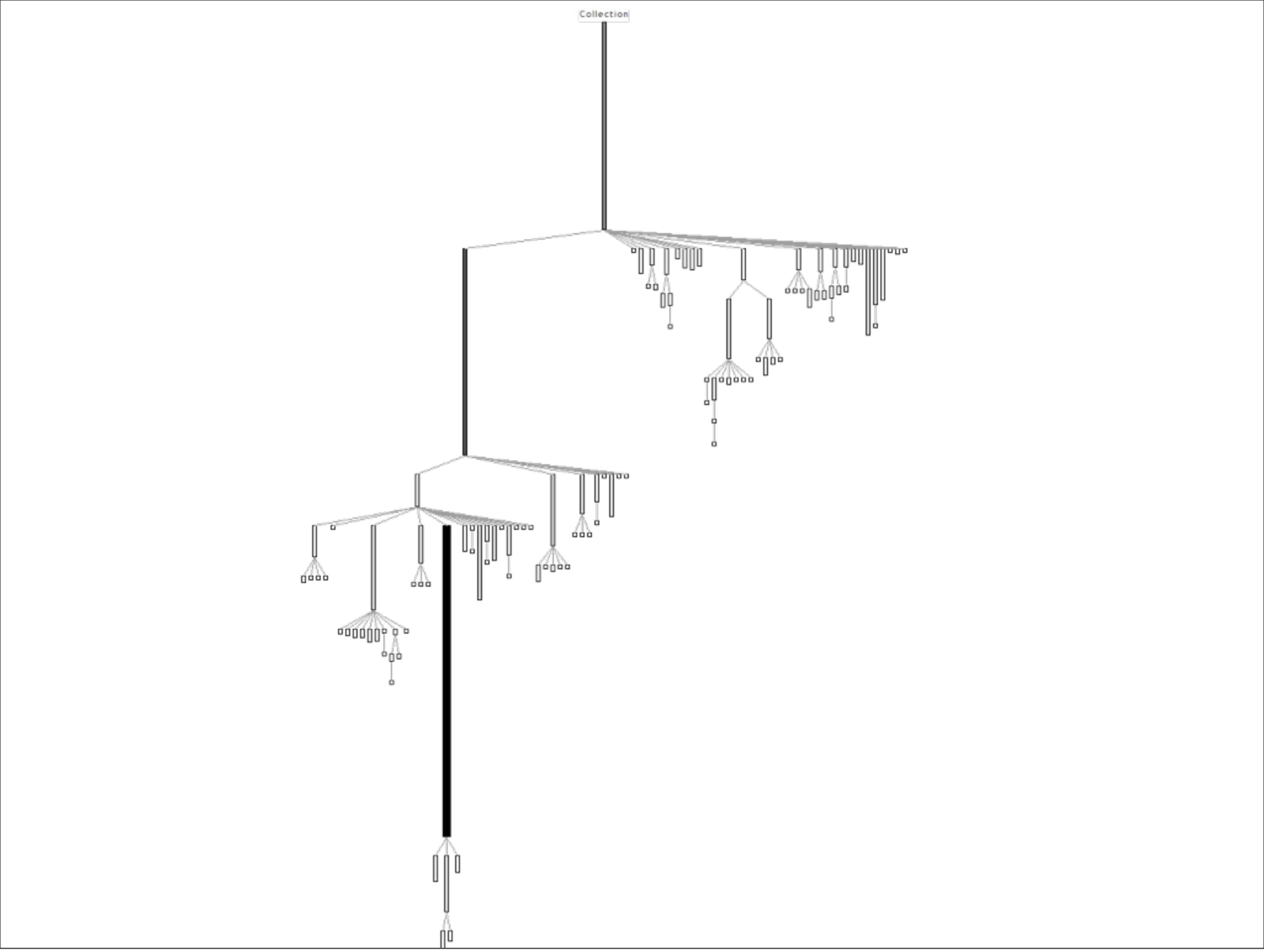


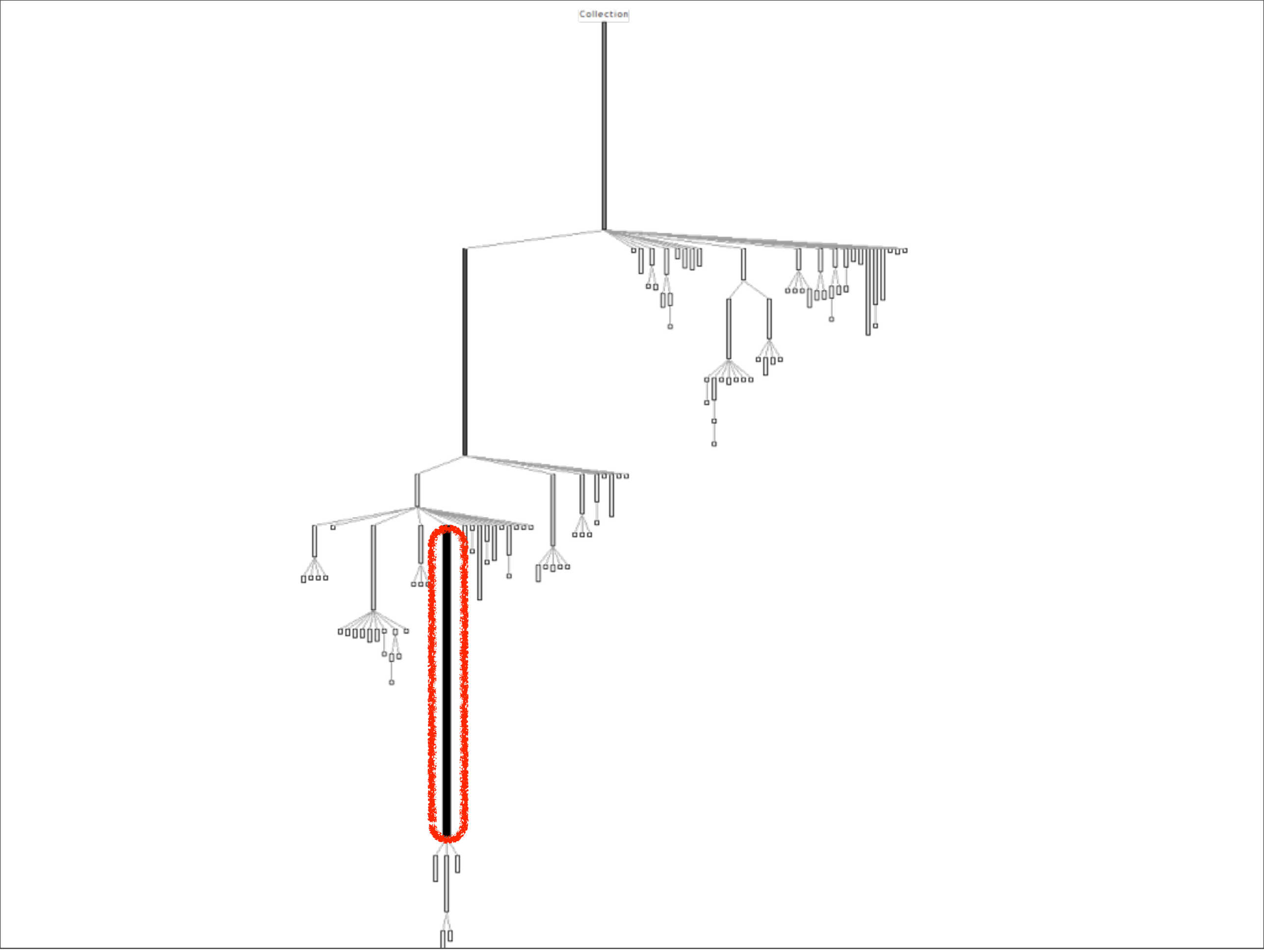
System Complexity

Lanza and Ducasse 2003

Rendering

Shape and Nodes


Collection

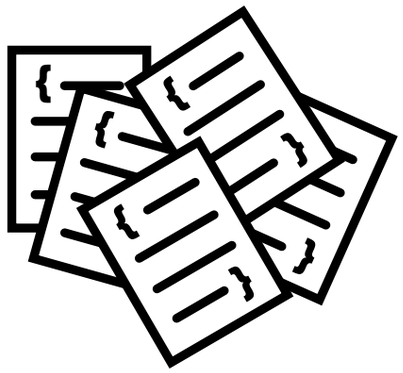

Collection

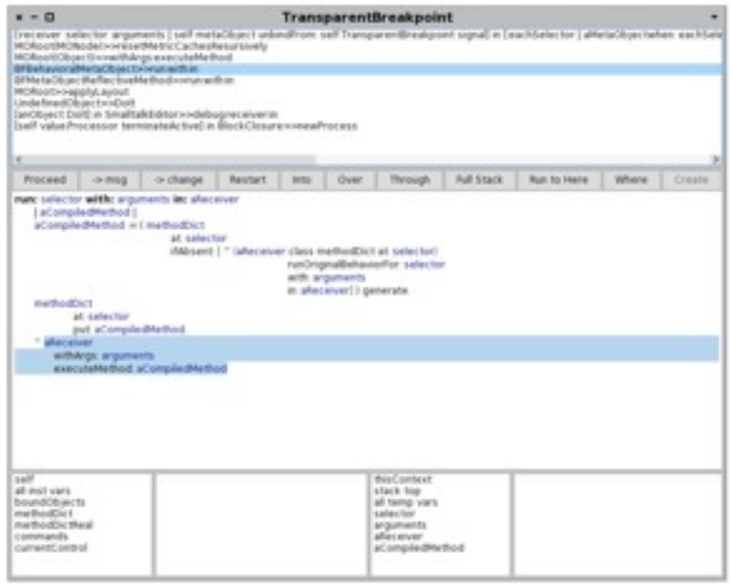
How do we debug this?

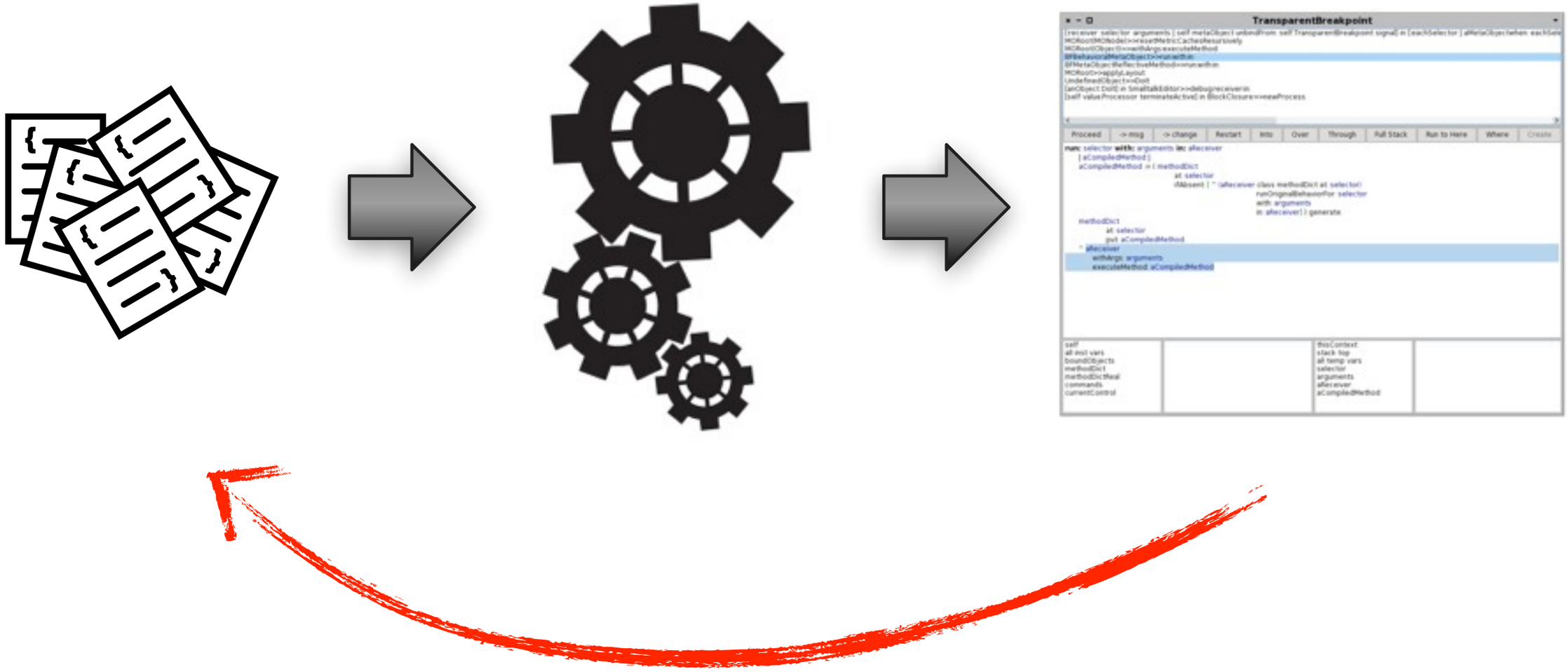
Breakpoints

Conditional

Breakpoints




TransparentBreakpoint
Proceed
-> msg
-> change
Restart
Into
Over
Through
Full Stack
Run to Here
Where
Create
run: selector with: arguments in: aReceiver
| aCompiledMethod |
aCompiledMethod := (methodDict
at: selector
ifAbsent: [" (aReceiver class methodDict at: selector)
runOriginalBehaviorFor: selector
with: arguments
in: aReceiver]) generate.
methodDict
at: selector
put: aCompiledMethod.
" aReceiver
withArgs: arguments
executeMethod: aCompiledMethod
self
all inst vars
boundObjects
methodDict
methodDictReal
commands
currentControl
thisContext
stack top
all temp vars
selector
arguments
aReceiver
aCompiledMethod


TransparentBreakpoint
Proceed
-> msg
-> change
Restart
Into
Over
Through
Full Stack
Run to Here
Where
Create

Developer Questions

When during the execution is this method called? (Q.13)

Where are instances of this class created? (Q.14)

Where is this variable or data structure being accessed? (Q.15)

What are the values of the argument at runtime? (Q.19)

What data is being modified in this code? (Q.20)

How are these types or objects related? (Q.22)

How can data be passed to (or accessed at) this point in the code? (Q.28)

What parts of this data structure are accessed in this code? (Q.33)

When during the execution is this method called? (Q.13)

Where are instances of this class created? (Q.14)

Where is this variable or data structure being accessed? (Q.15)

What are the values of the argument at runtime? (Q.19)

What data is being modified in this code? (Q.20)

How are these types or objects related? (Q.22)

How can data be passed to (or accessed at) this point in the code? (Q.28)

What parts of this data structure are accessed in this code? (Q.33)

Sillito etal.

Questions programmers ask during software evolution tasks. 2008

Which is the relationship?

When during the execution is this method called? (Q.13)

Where are instances of this class created? (Q.14)

Where is this variable or data structure being accessed? (Q.15)

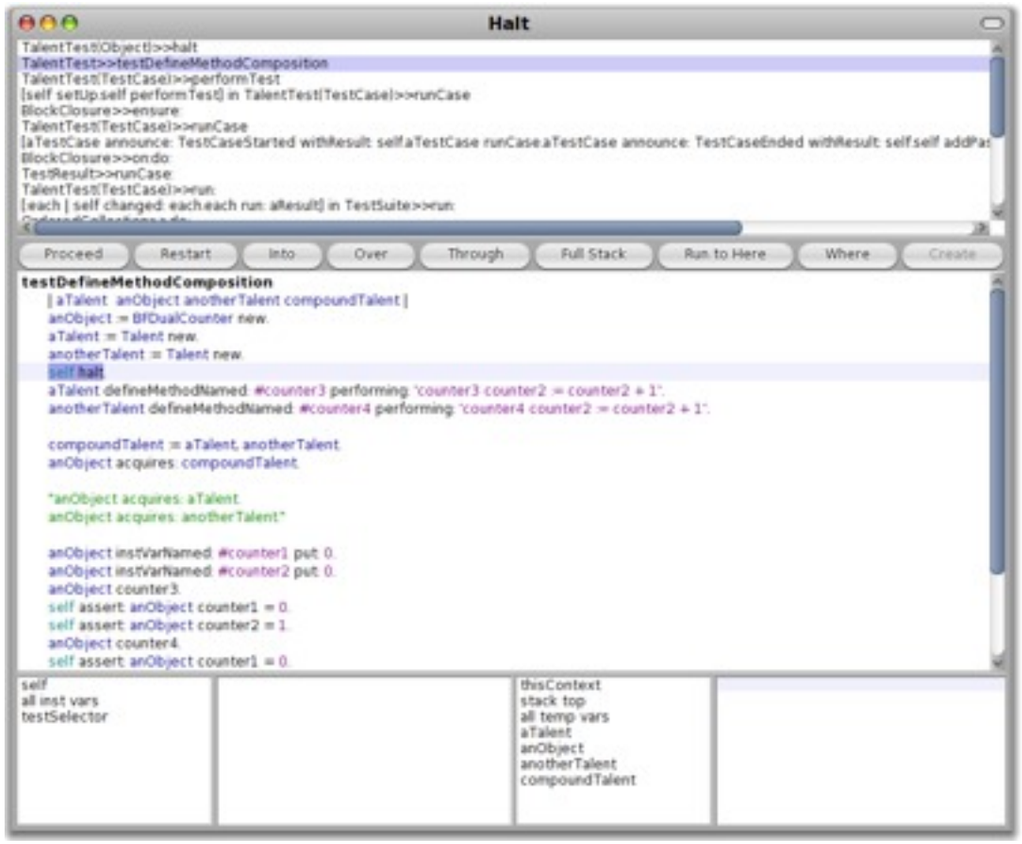
What are the values of the argument at runtime? (Q.19)

What data is being modified in this code? (Q.20)

How are these types or objects related? (Q.22)

How can data be passed to (or accessed at) this point in the code? (Q.28)

What parts of this data structure are accessed in this code? (Q.33)



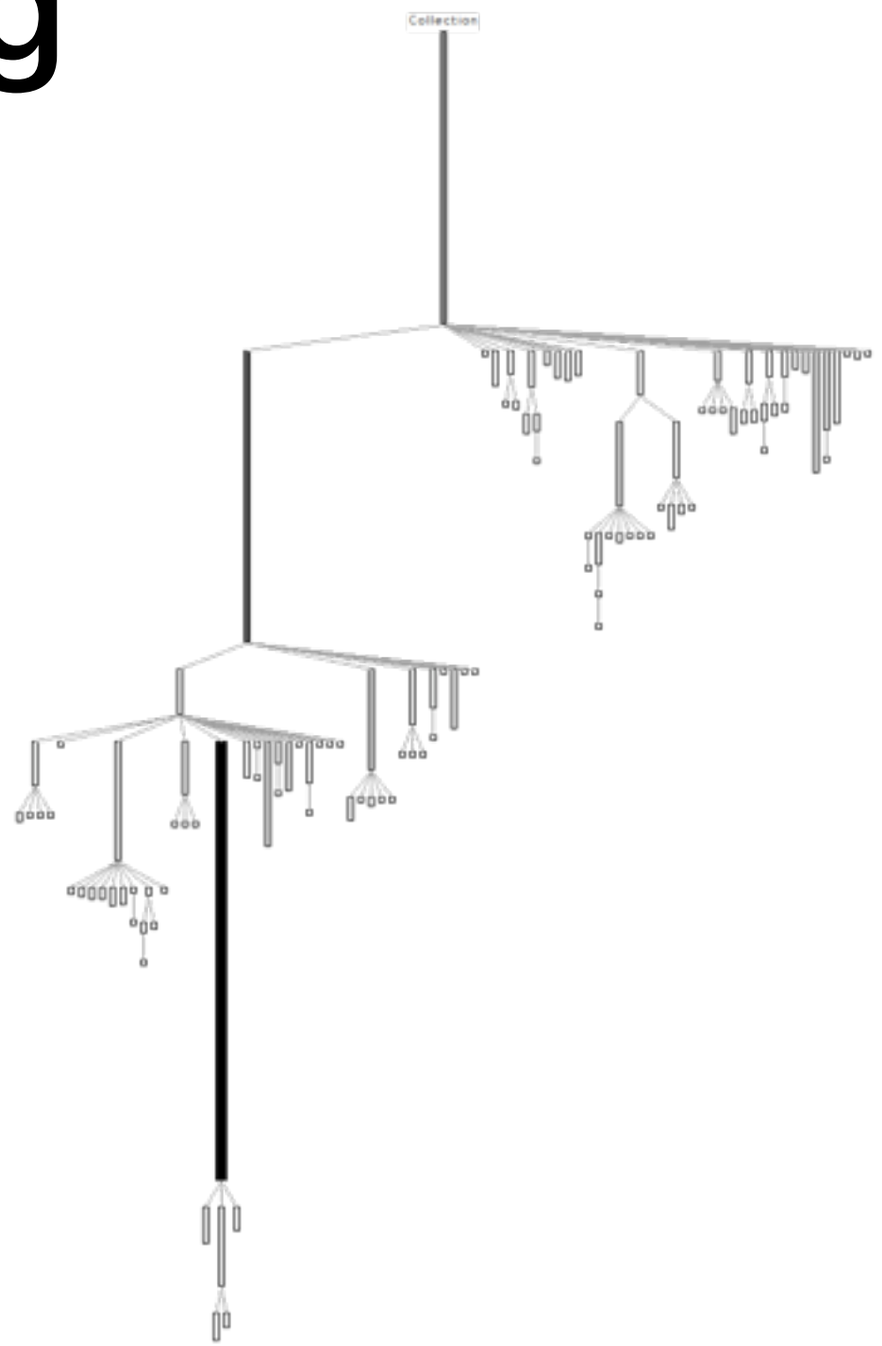
What is the problem?

Traditional Reflection

Profiling

```
54.8% {11501ms} MOCanvas>>drawOn:
  54.8% {11501ms} MORoot(MONode)>>displayOn:
   30.9% {6485ms} MONode>>displayOn:
      | 18.1% {3799ms} MOEdge>>displayOn:
          ...
      |  8.4% {1763ms} MOEdge>>displayOn:
      |     | 8.0% {1679ms} MOStraightLineShape>>display:on:
      |     |  2.6% {546ms} FormCanvas>>line:to:width:color:
          ...
   23.4% {4911ms} MOEdge>>displayOn:
          ...
```

?



Debugging

When during the execution is this method called? (Q.13)

Where are instances of this class created? (Q.14)

Where is this variable or data structure being accessed? (Q.15)

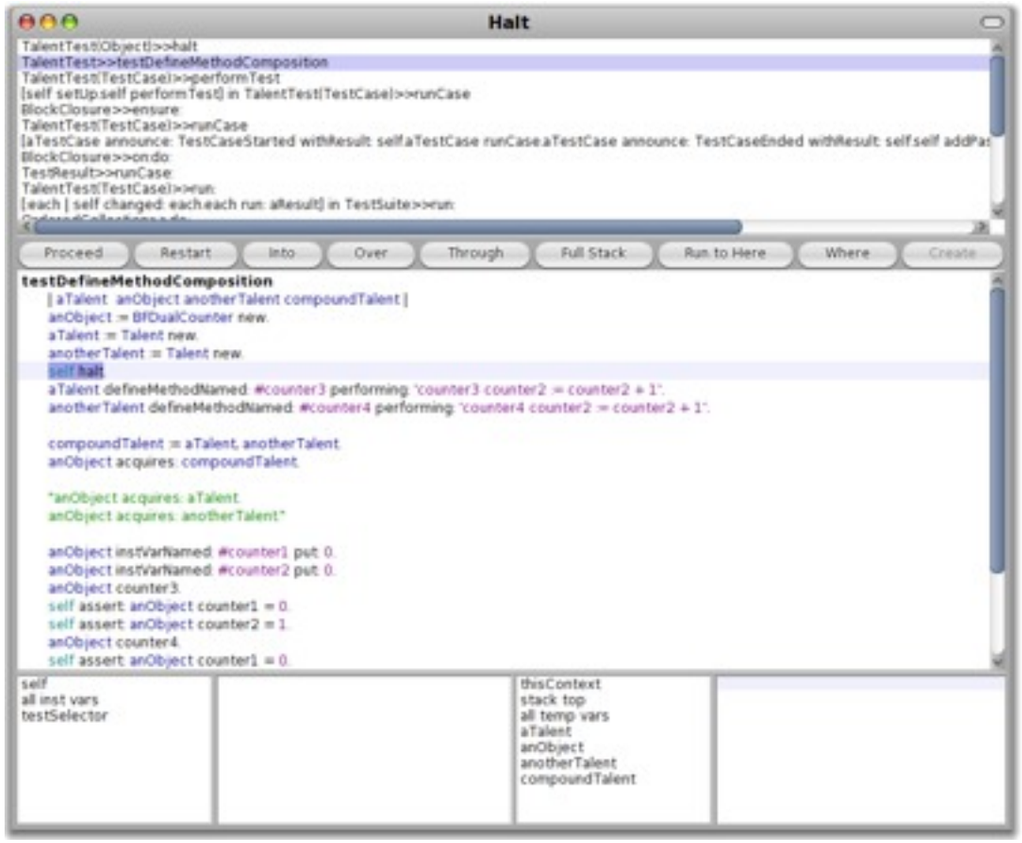
What are the values of the argument at runtime? (Q.19)

What data is being modified in this code? (Q.20)

How are these types or objects related? (Q.22)

How can data be passed to (or accessed at) this point in the code? (Q.28)

What parts of this data structure are accessed in this code? (Q.33)



Object Paradox

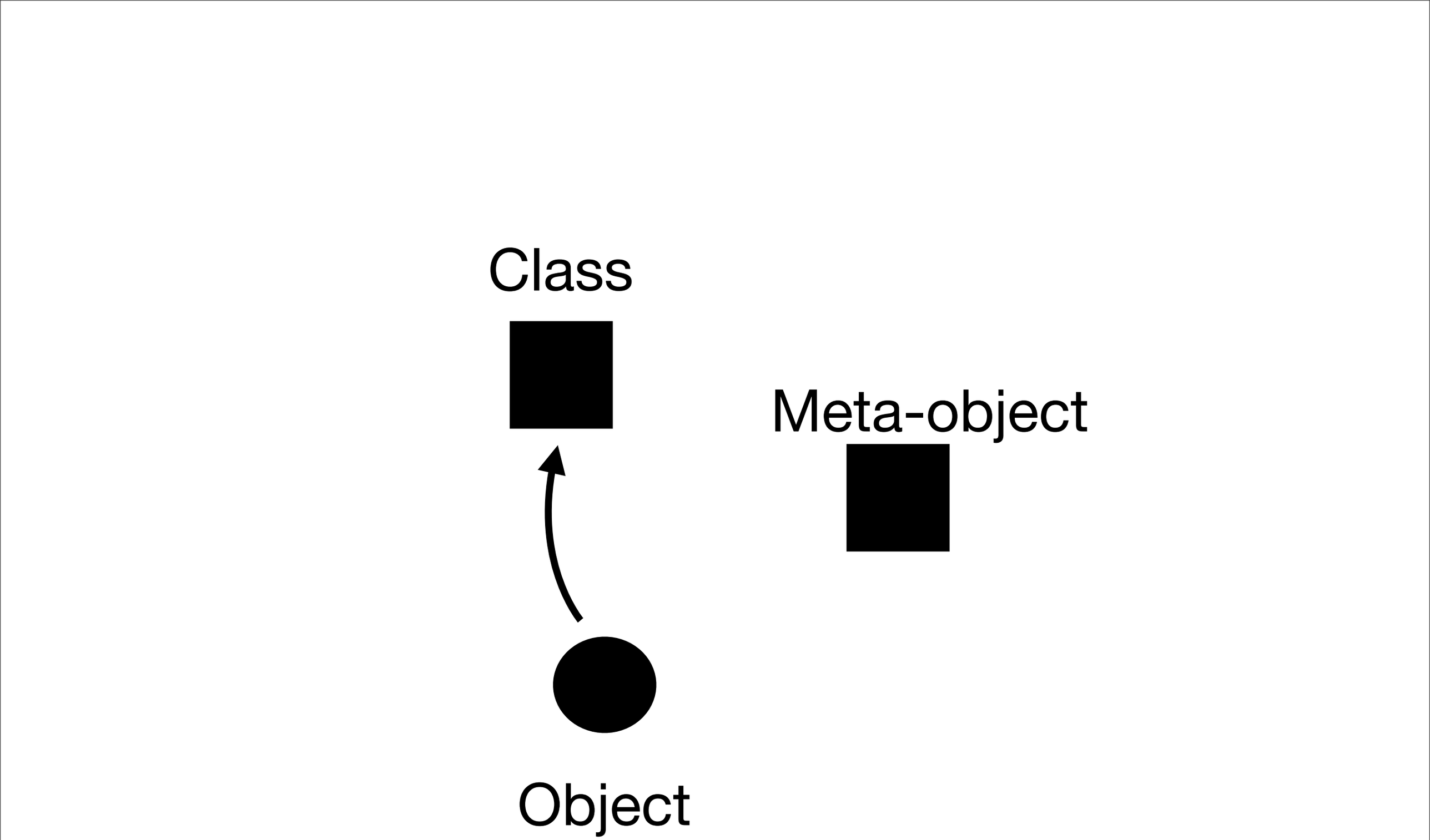
Object-Centric Reflection

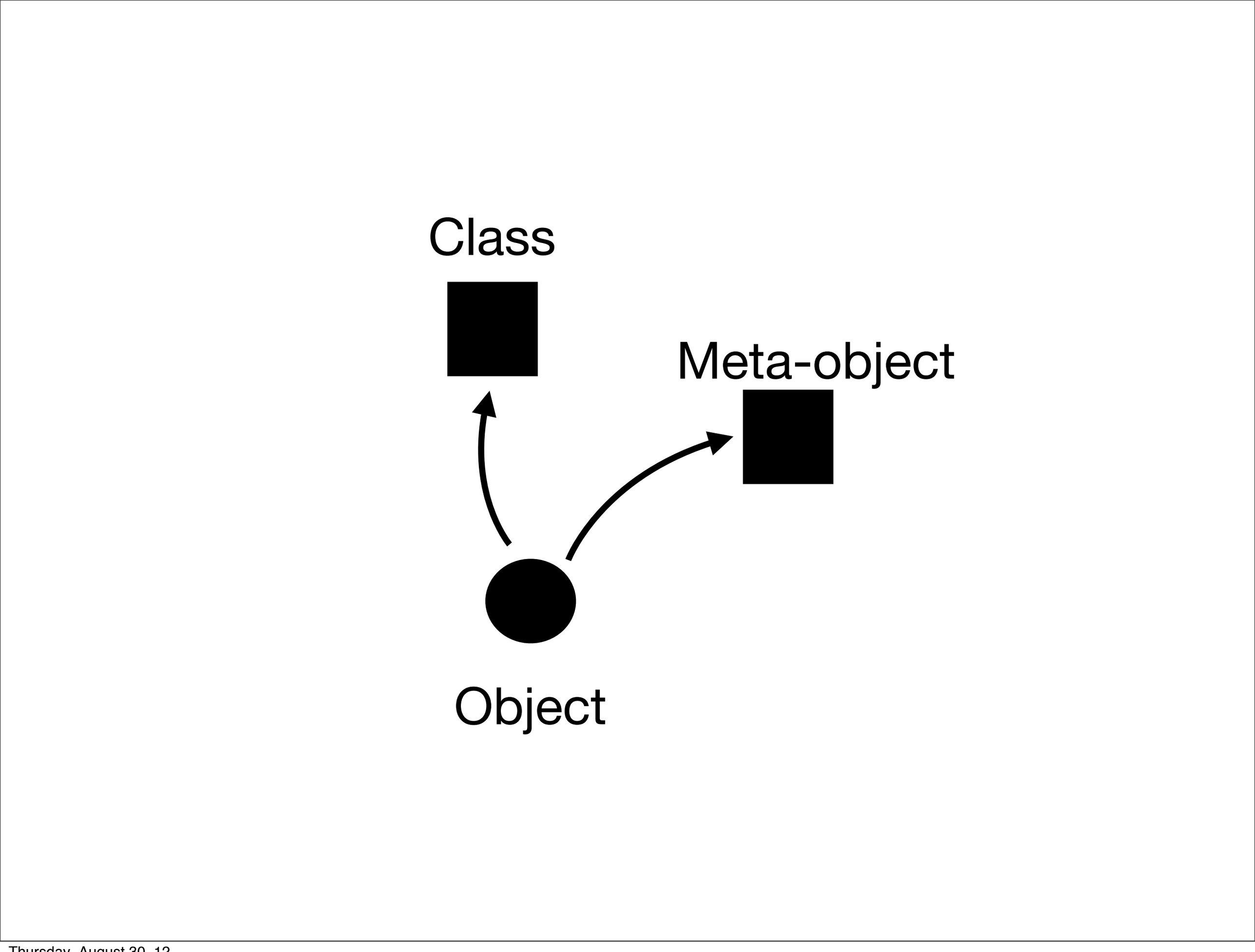

Bifröst

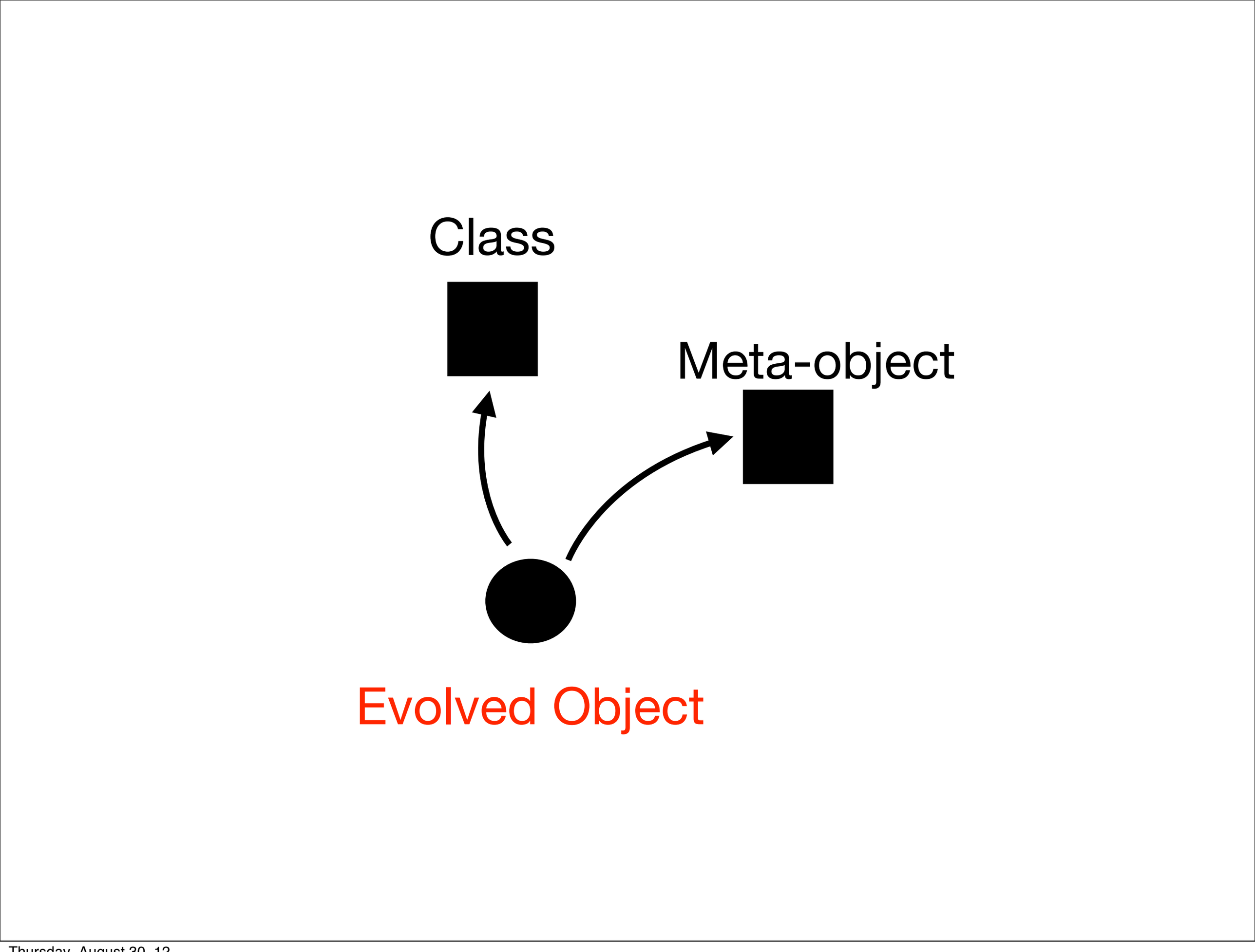
Organize the Meta-level

Explicit

Meta-objects


Class
Meta-object
Object


Class
Meta-object
Object


Class
Meta-object
Evolved Object

Debugging

Profiling

Execution
Reification

Structure
Evolution

Debugging

Profiling

Execution
Reification

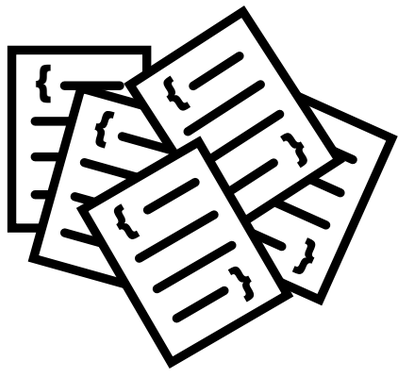
Structure
Evolution

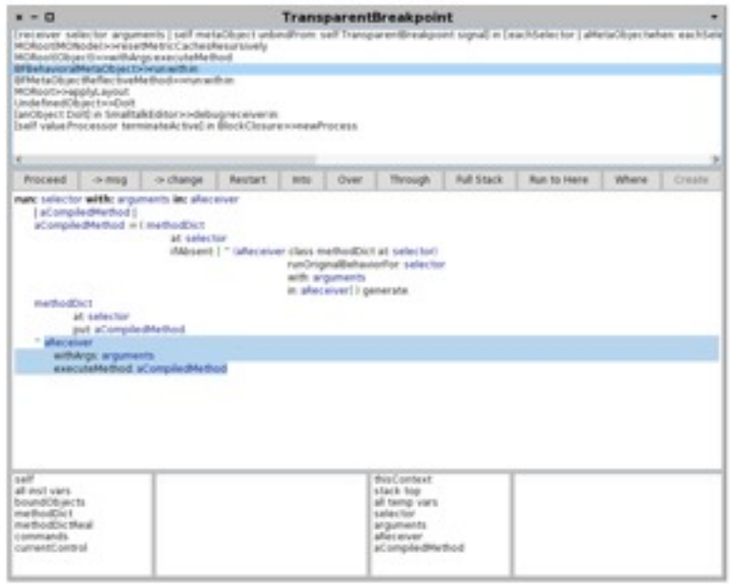
Object-Centric Debugging

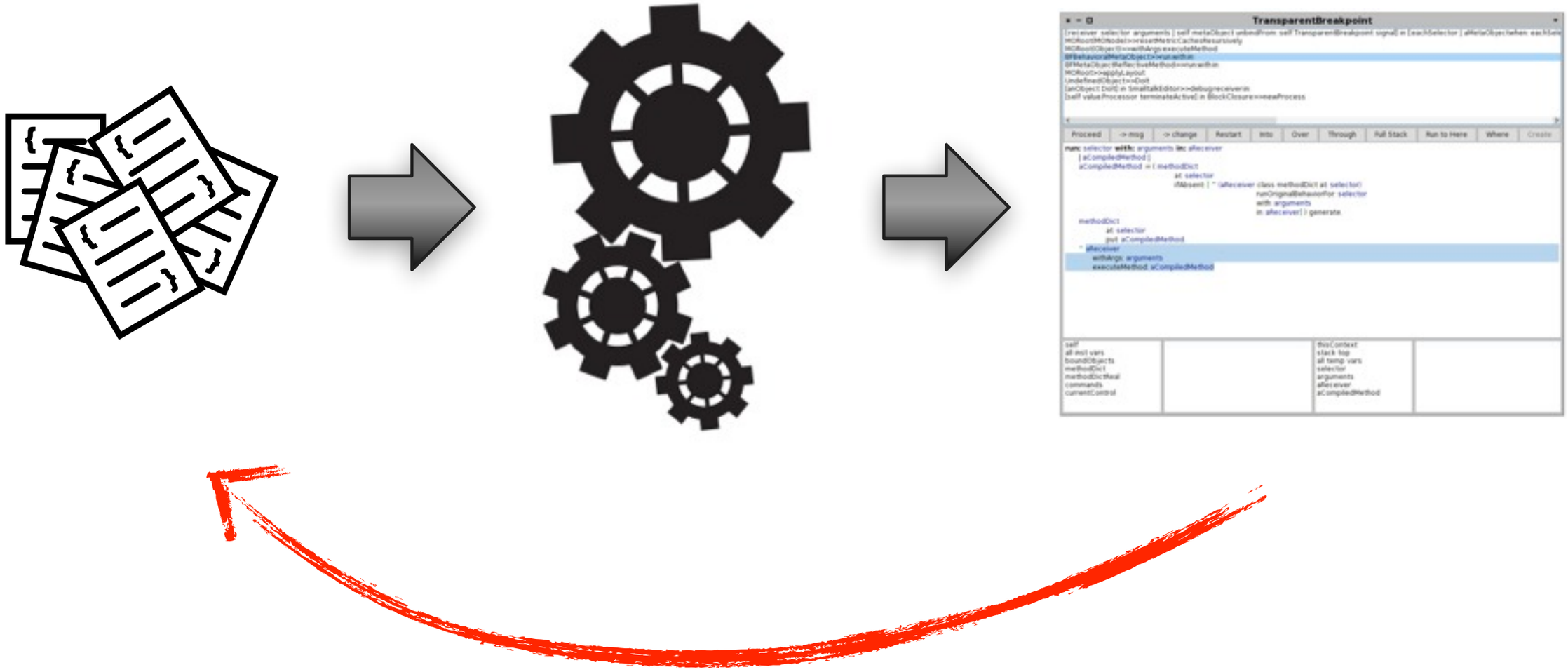
Object-Centric Debugging

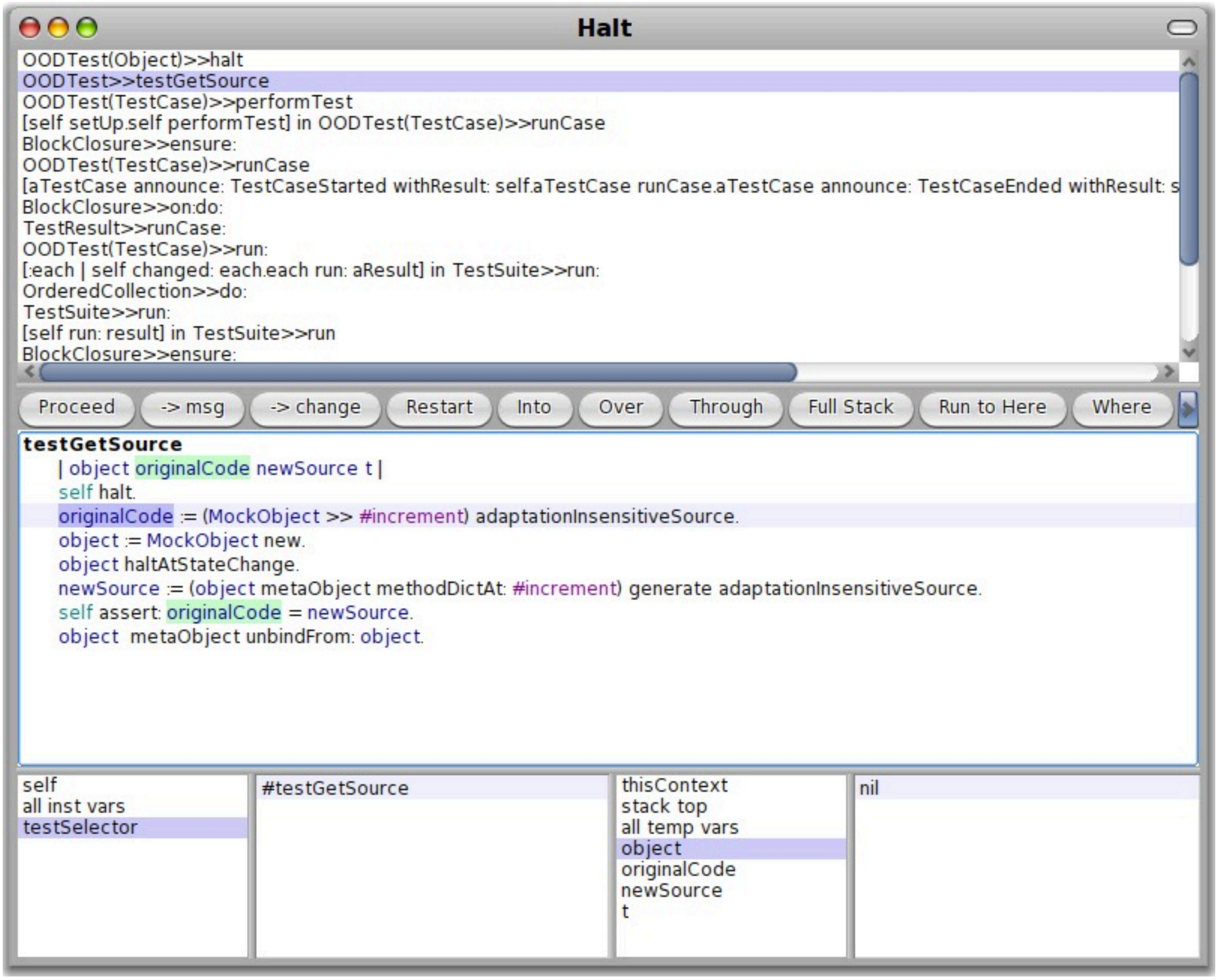
ICSE 2012

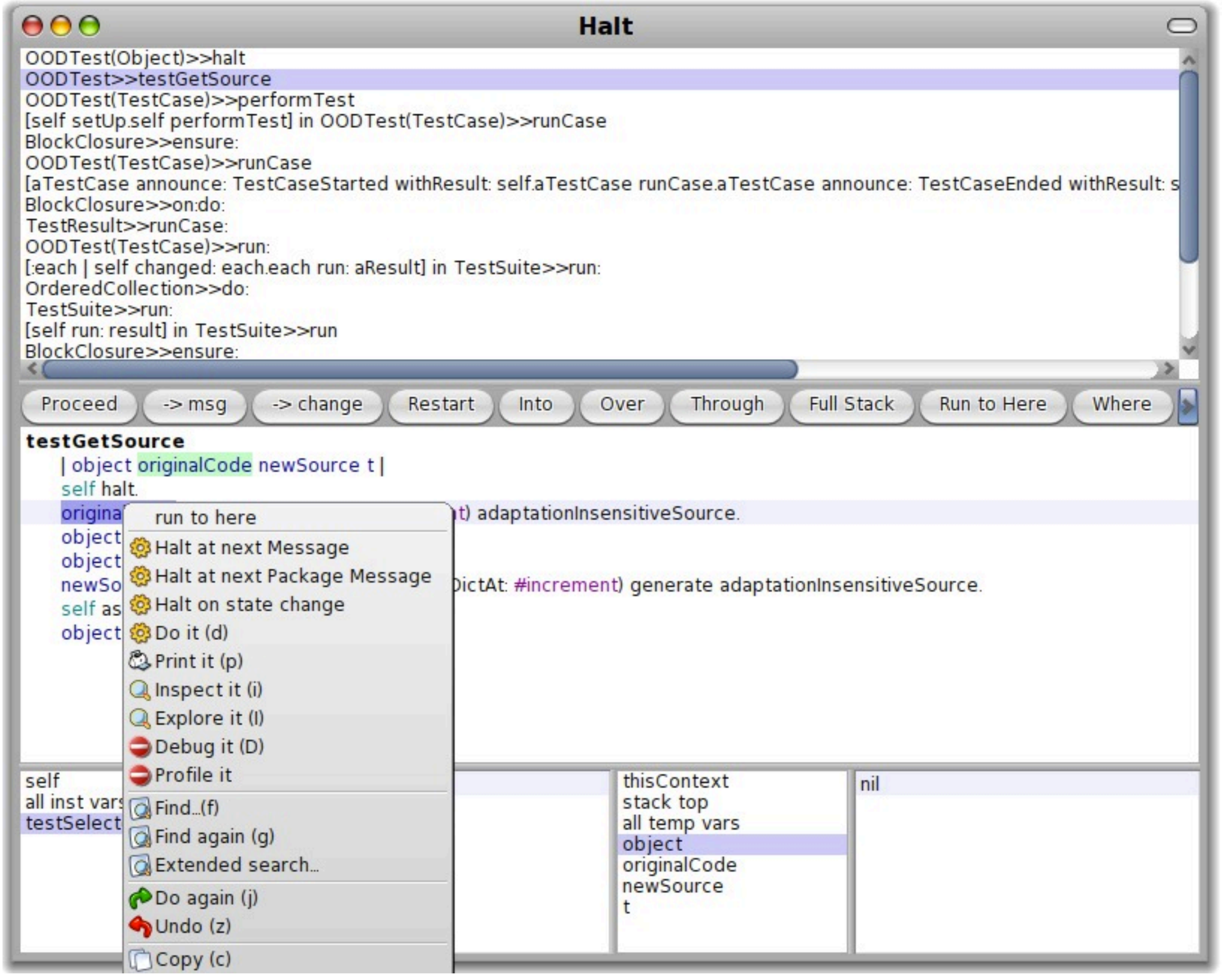
J. Ressia, A, Bergel and O. Nierstrasz

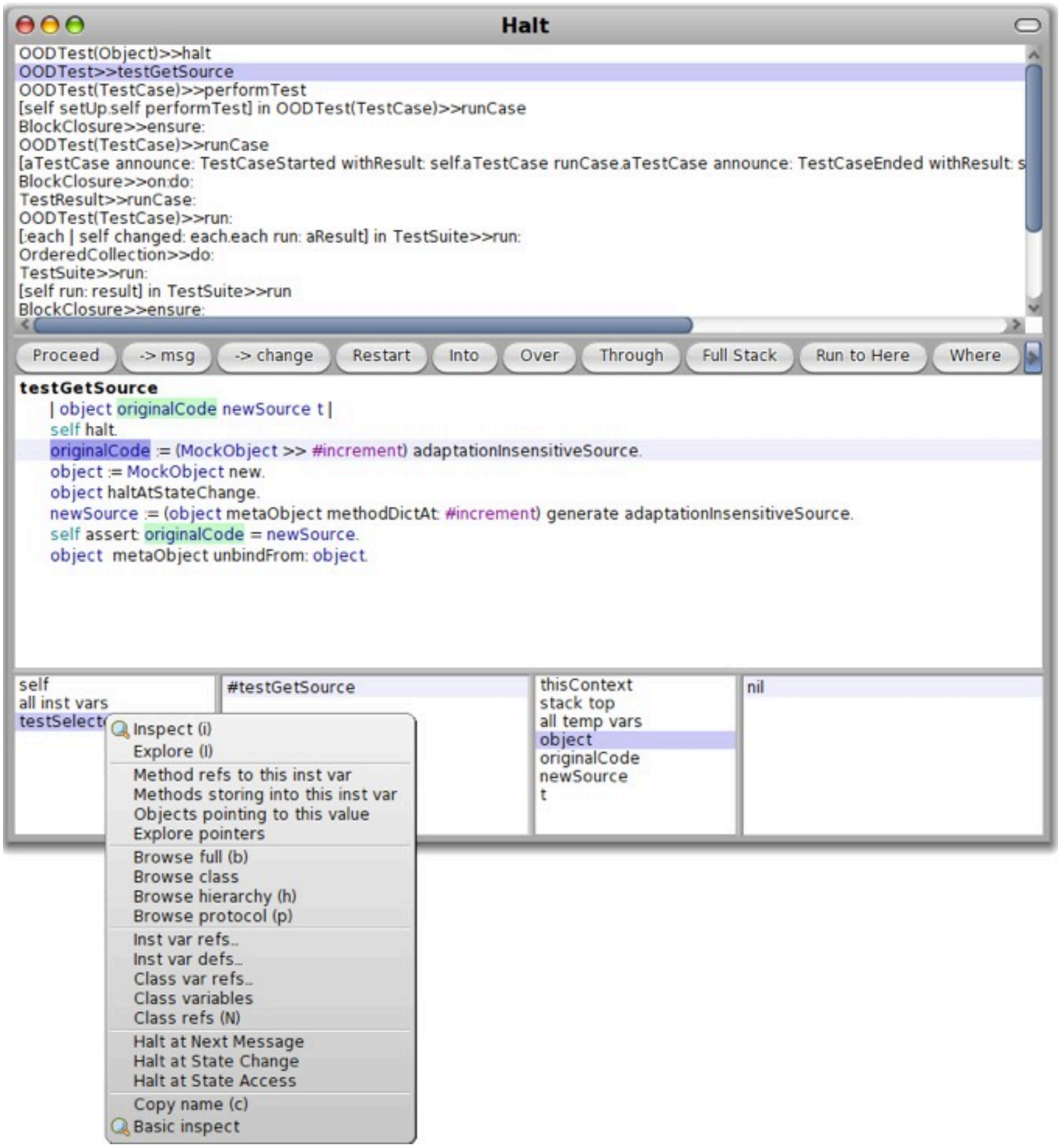



TransparentBreakpoint
Proceed
-> msg
-> change
Restart
Into
Over
Through
Full Stack
Run to Here
Where
Create
run: selector with: arguments in: aReceiver
| aCompiledMethod |
aCompiledMethod := (methodDict
at: selector
ifAbsent: [" (aReceiver class methodDict at: selector)
runOriginalBehaviorFor: selector
with: arguments
in: aReceiver]) generate.
methodDict
at: selector
put: aCompiledMethod.
" aReceiver
withArgs: arguments
executeMethod: aCompiledMethod
self
all inst vars
boundObjects
methodDict
methodDictReal
commands
currentControl
thisContext
stack top
all temp vars
selector
arguments
aReceiver
aCompiledMethod


TransparentBreakpoint
Proceed
-> msg
-> change
Restart
Into
Over
Through
Full Stack
Run to Here
Where
Create


Halt
OODTest(Object)>>halt
OODTest>>testGetSource
OODTest(TestCase)>>performTest
[self setUp.self performTest] in OODTest(TestCase)>>runCase
BlockClosure>>ensure:
OODTest(TestCase)>>runCase
[aTestCase announce: TestCaseStarted withResult: self.aTestCase runCase.aTestCase announce: TestCaseEnded withResult: s
BlockClosure>>on:do:
TestResult>>runCase:
OODTest(TestCase)>>run:
[:each | self changed: each.each run: aResult] in TestSuite>>run:
OrderedCollection>>do:
TestSuite>>run:
[self run: result] in TestSuite>>run
BlockClosure>>ensure:
Proceed
-> msg
-> change
Restart
Into
Over
Through
Full Stack
Run to Here
Where
testGetSource
| object originalCode newSource t |
self halt.
originalCode := (MockObject >> #increment) adaptationInsensitiveSource.
object := MockObject new.
object haltAtStateChange.
newSource := (object metaObject methodDictAt: #increment) generate adaptationInsensitiveSource.
self assert: originalCode = newSource.
object metaObject unbindFrom: object.
self
all inst vars
testSelector
#testGetSource
thisContext
stack top
all temp vars
object
originalCode
newSource
t
nil


Halt
OODTest(Object)>>halt
OODTest>>testGetSource
OODTest(TestCase)>>performTest
[self setUp.self performTest] in OODTest(TestCase)>>runCase
BlockClosure>>ensure:
OODTest(TestCase)>>runCase
[aTestCase announce: TestCaseStarted withResult: self.aTestCase runCase.aTestCase announce: TestCaseEnded withResult: s
BlockClosure>>on:do:
TestResult>>runCase:
OODTest(TestCase)>>run:
[:each | self changed: each.each run: aResult] in TestSuite>>run:
OrderedCollection>>do:
TestSuite>>run:
[self run: result] in TestSuite>>run
BlockClosure>>ensure:
Proceed
-> msg
-> change
Restart
Into
Over
Through
Full Stack
Run to Here
Where
testGetSource
| object originalCode newSource t |
self halt.
t) adaptationInsensitiveSource.
DictAt: #increment) generate adaptationInsensitiveSource.
run to here
Halt at next Message
Halt at next Package Message
Halt on state change
Do it (d)
Print it (p)
Inspect it (i)
Explore it (I)
Debug it (D)
Profile it
Find...(f)
Find again (g)
Extended search...
Do again (j)
Undo (z)
Copy (c)
self
all inst vars
testSelect
thisContext
stack top
all temp vars
object
originalCode
newSource
t
nil


Halt
OODTest(Object)>>halt
OODTest>>testGetSource
OODTest(TestCase)>>performTest
[self setUp.self performTest] in OODTest(TestCase)>>runCase
BlockClosure>>ensure:
OODTest(TestCase)>>runCase
[aTestCase announce: TestCaseStarted withResult: self.aTestCase runCase.aTestCase announce: TestCaseEnded withResult: s
BlockClosure>>on:do:
TestResult>>runCase:
OODTest(TestCase)>>run:
[:each | self changed: each.each run: aResult] in TestSuite>>run:
OrderedCollection>>do:
TestSuite>>run:
[self run: result] in TestSuite>>run
BlockClosure>>ensure:
Proceed
-> msg
-> change
Restart
Into
Over
Through
Full Stack
Run to Here
Where
testGetSource
| object originalCode newSource t |
self halt.
originalCode := (MockObject >> #increment) adaptationInsensitiveSource.
object := MockObject new.
object haltAtStateChange.
newSource := (object metaObject methodDictAt: #increment) generate adaptationInsensitiveSource.
self assert: originalCode = newSource.
object metaObject unbindFrom: object.
self
all inst vars
#testGetSource
thisContext
stack top
all temp vars
object
originalCode
newSource
t
nil
Inspect (i)
Explore (I)
Method refs to this inst var
Methods storing into this inst var
Objects pointing to this value
Explore pointers
Browse full (b)
Browse class
Browse hierarchy (h)
Browse protocol (p)
Inst var refs...
Inst var defs...
Class var refs...
Class variables
Class refs (N)
Halt at Next Message
Halt at State Change
Halt at State Access
Copy name (c)
Basic inspect

stack-centric debugging

step into, step over, resume

InstructionStream class>>on:
InstructionStream class>>new
InstructionStream>>initialize
CompiledMethod>>initialPC
InstructionStream>>method:pc:
InstructionStream>>nextInstruction
MessageCatcher class>>new
InstructionStream>>interpretNextInstructionFor:
...

object-centric debugging

centered on the InstructionStream class

next message, next change

on:
new

centered on the InstructionStream object

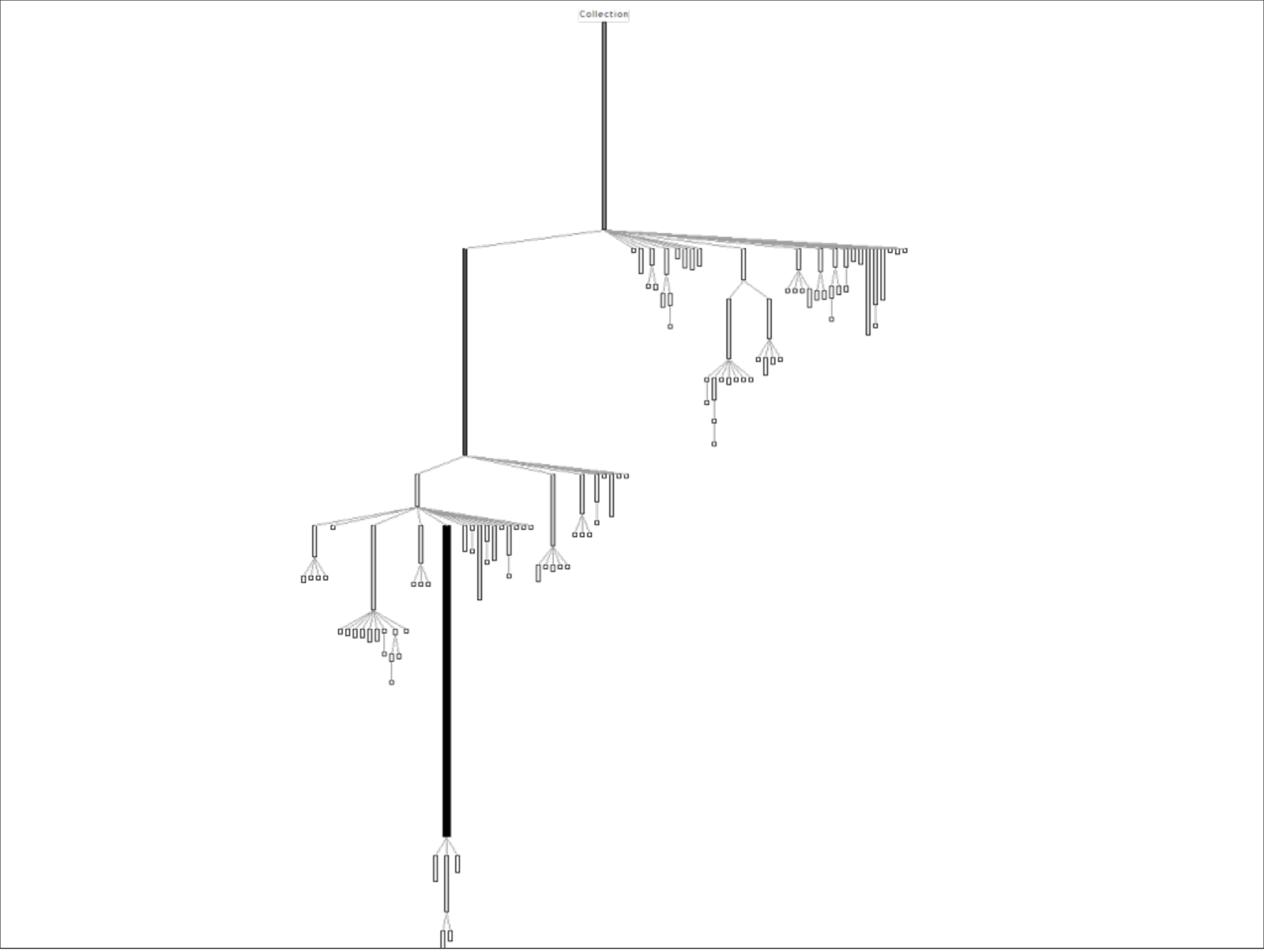
next message, next change

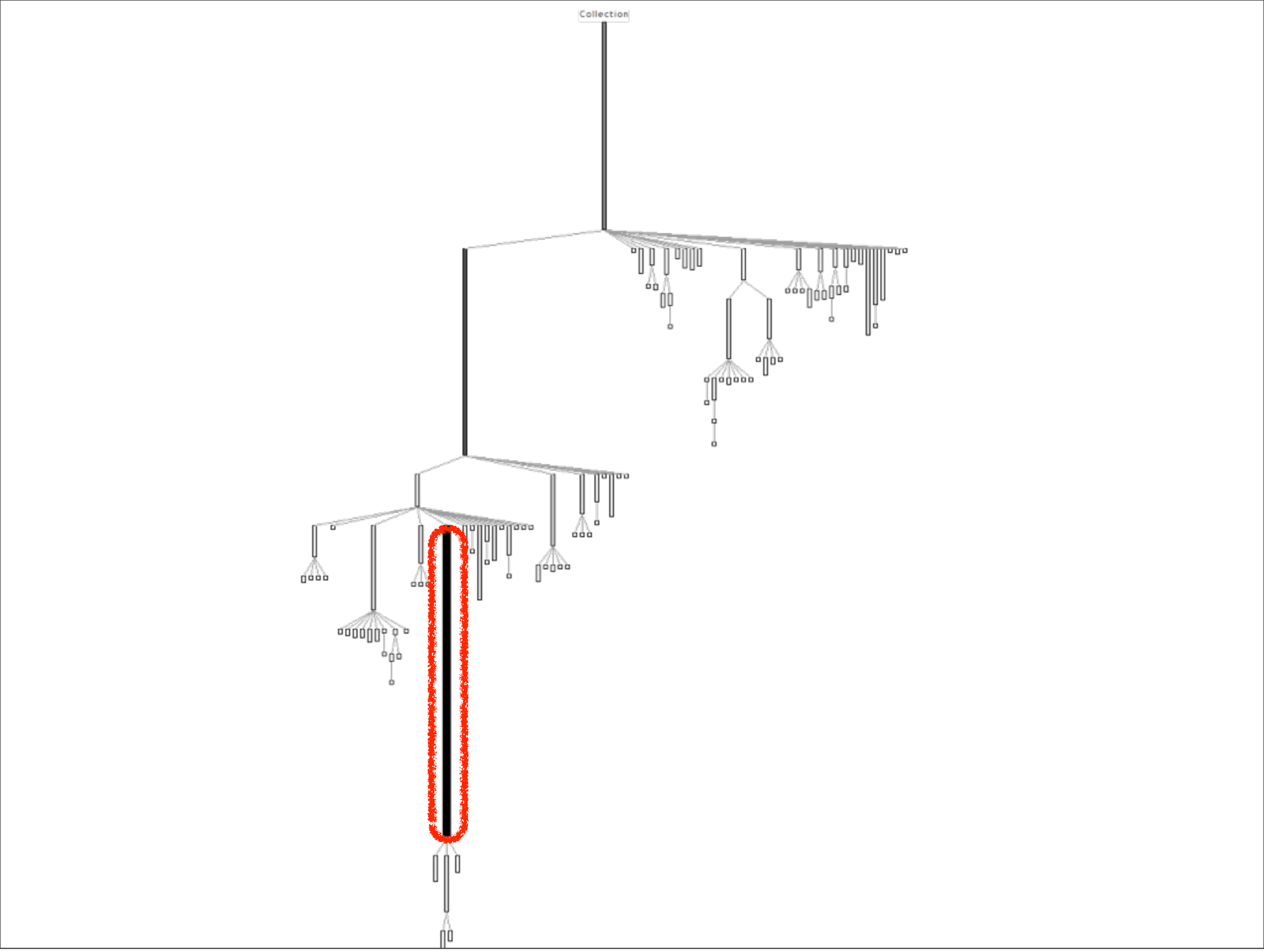
initialize
method:pc:
nextInstruction
interpretNextInstructionFor:
...

...

Mondrian

Shape and Nodes


Collection


Collection

halt on object in call

Halt on next message

Halt on next message/s named

Halt on state change

Halt on state change named

Halt on next inherited message

Halt on next overloaded message

Halt on object/s in call

Halt on next message from package

Debugging

Profiling

Execution
Reification

Structure
Evolution

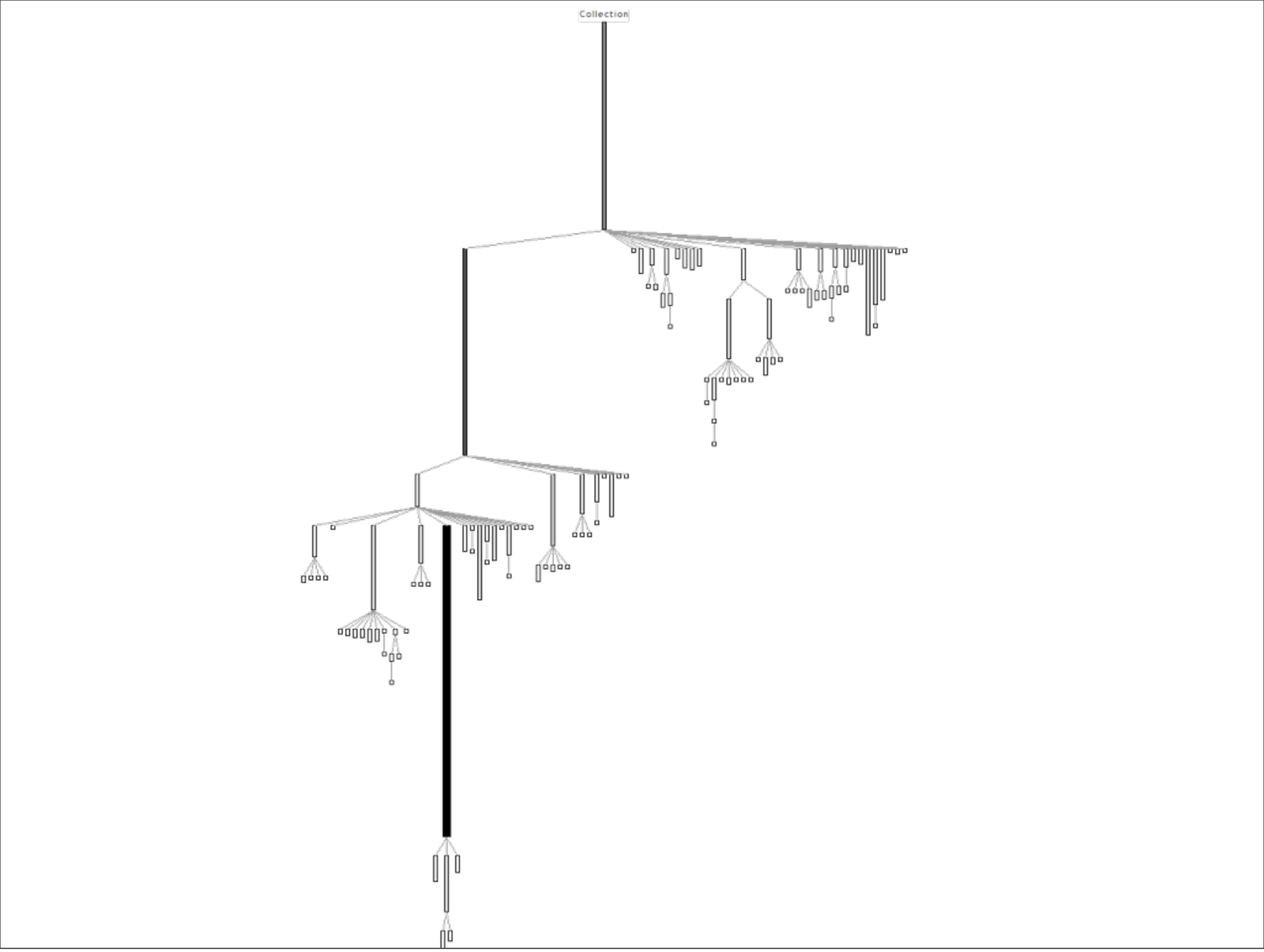
MetaSpy

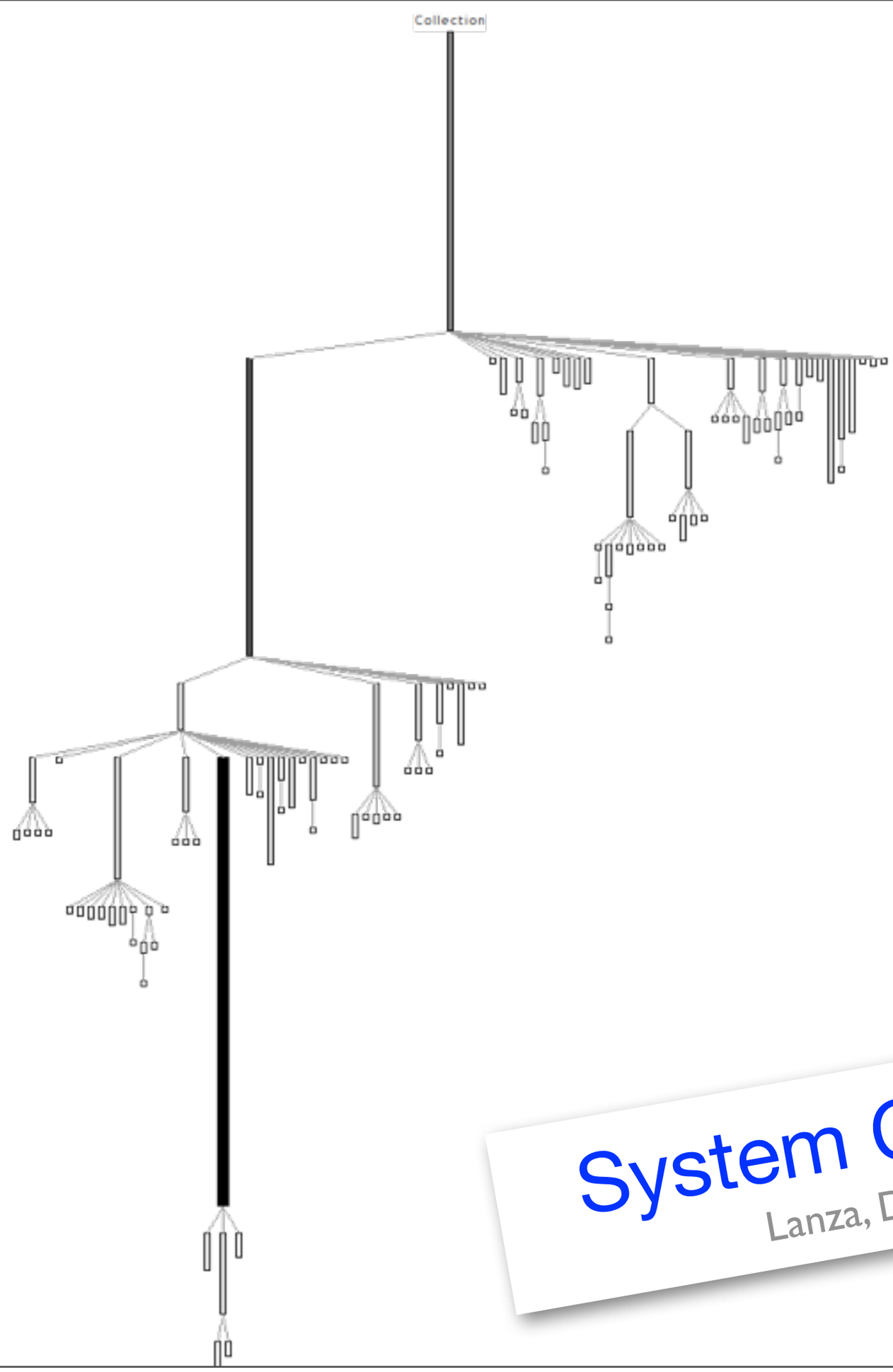
MetaSpy

TOOLS 2011

Bergel etal.

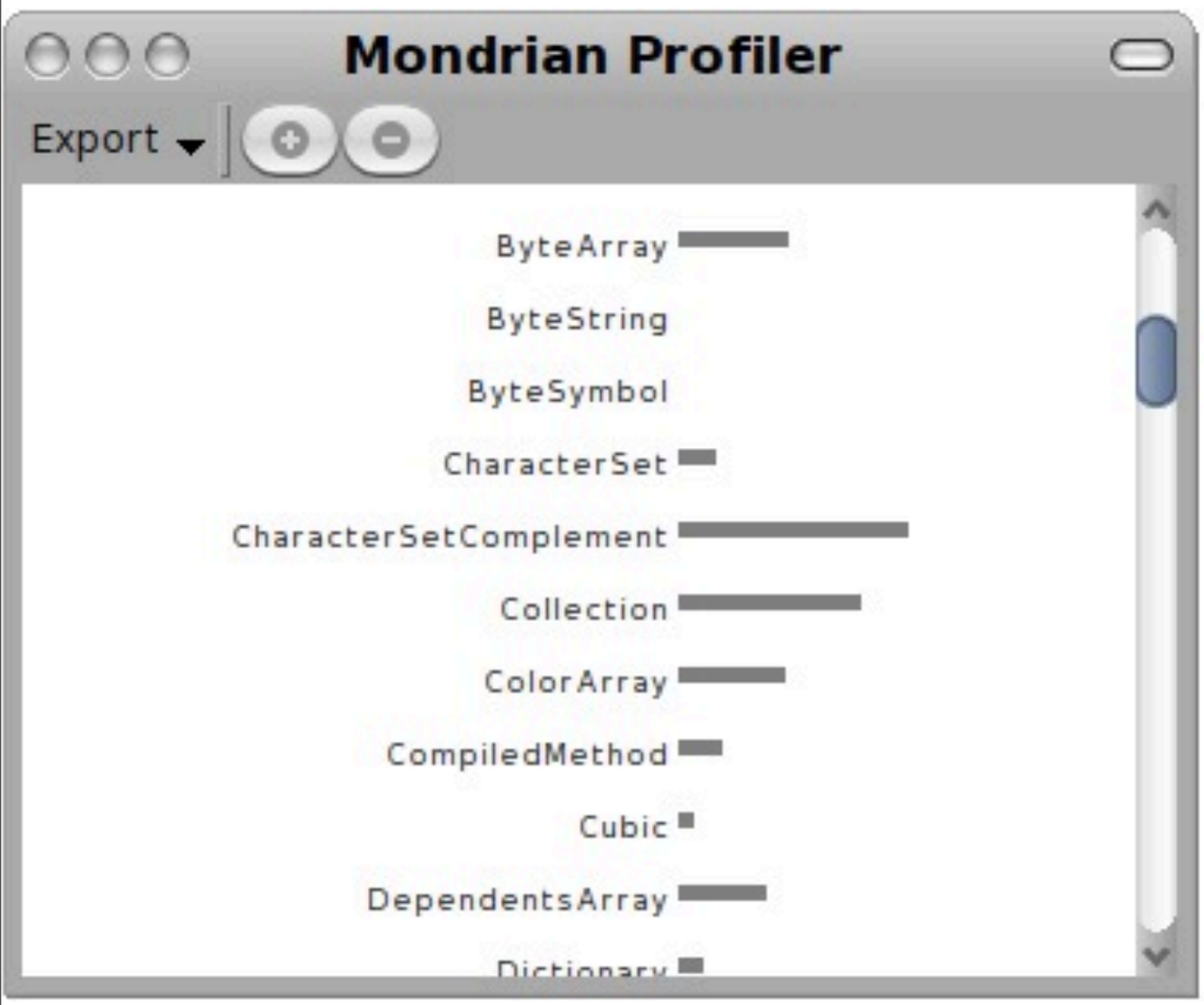
Mondrian Profiler

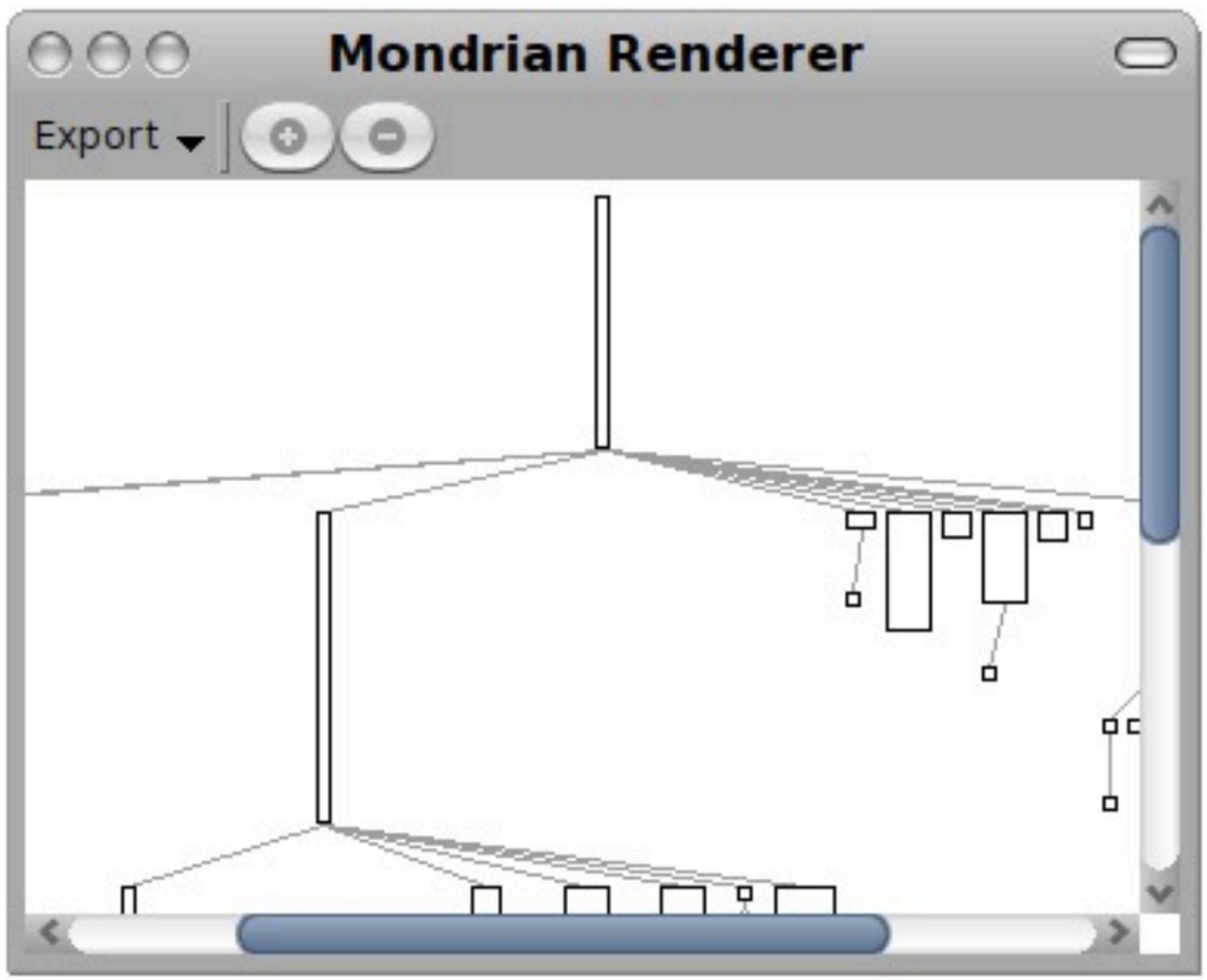

Collection



System Complexity

Lanza, Ducasse 2003


Mondrian Profiler
Export
ByteArray
ByteString
ByteSymbol
CharacterSet
CharacterSetComplement
Collection
ColorArray
CompiledMethod
Cubic
DependentsArray
Dictionary


Mondrian Renderer
Export

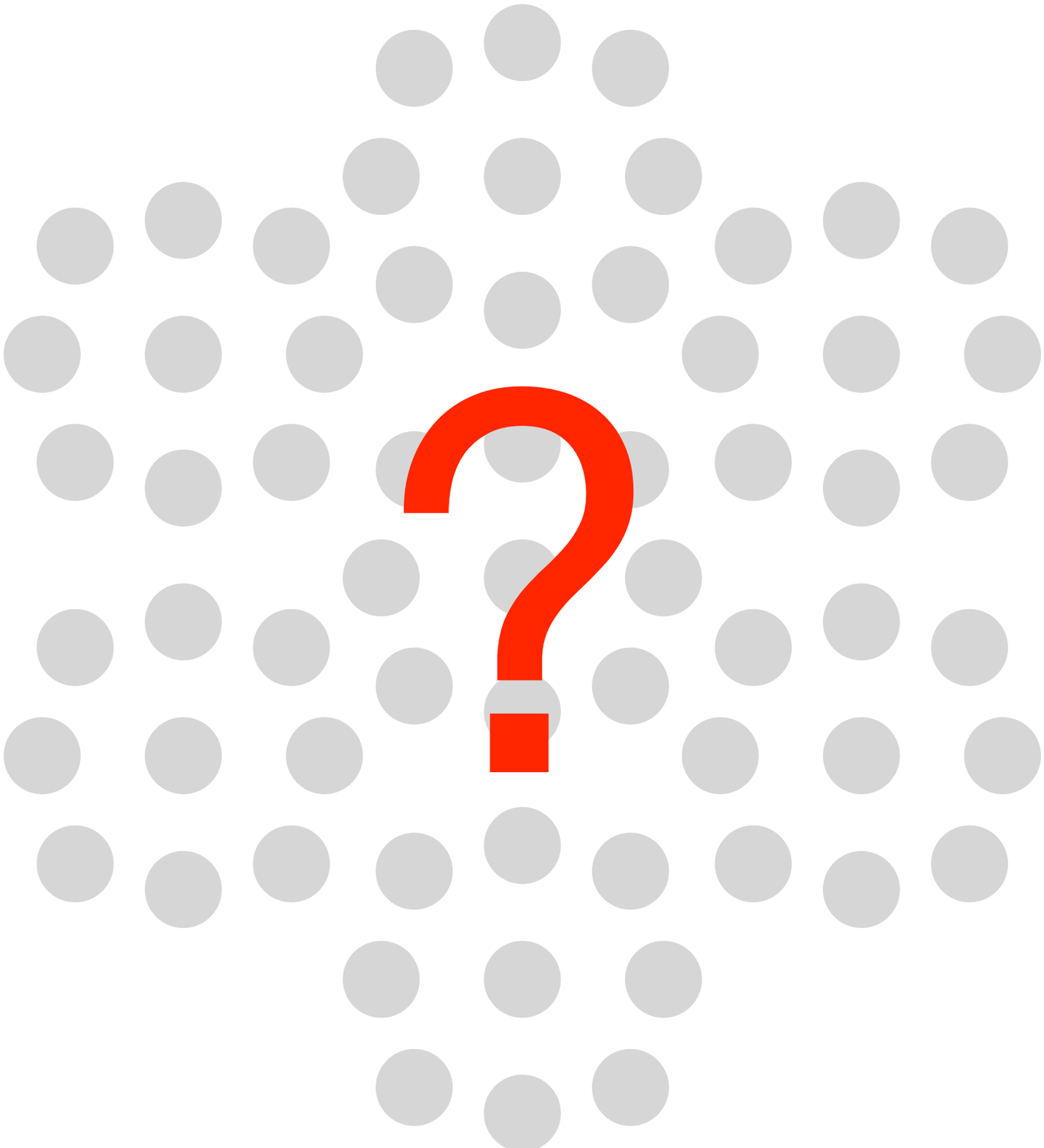
Debugging

Profiling

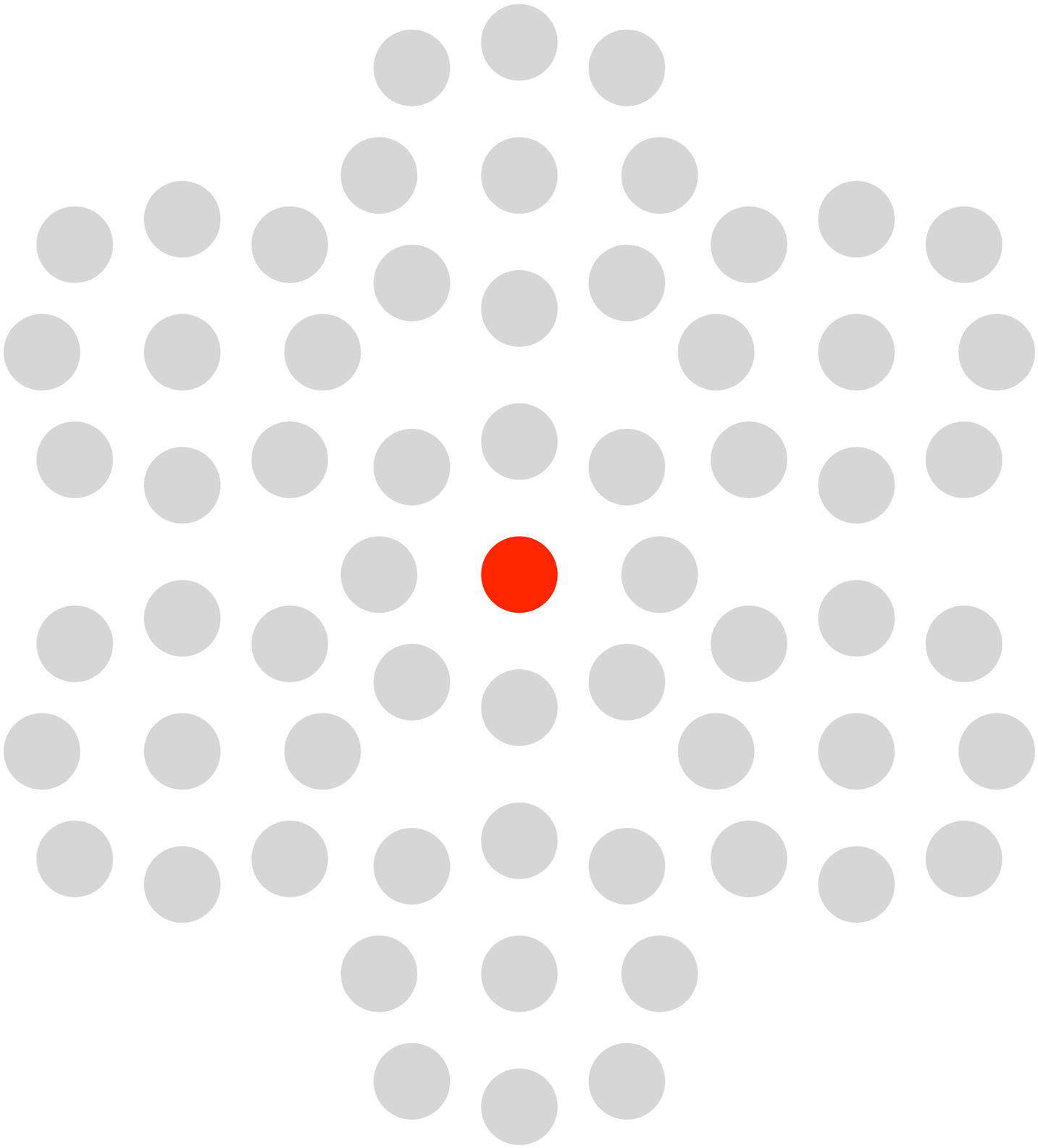
Execution
Reification

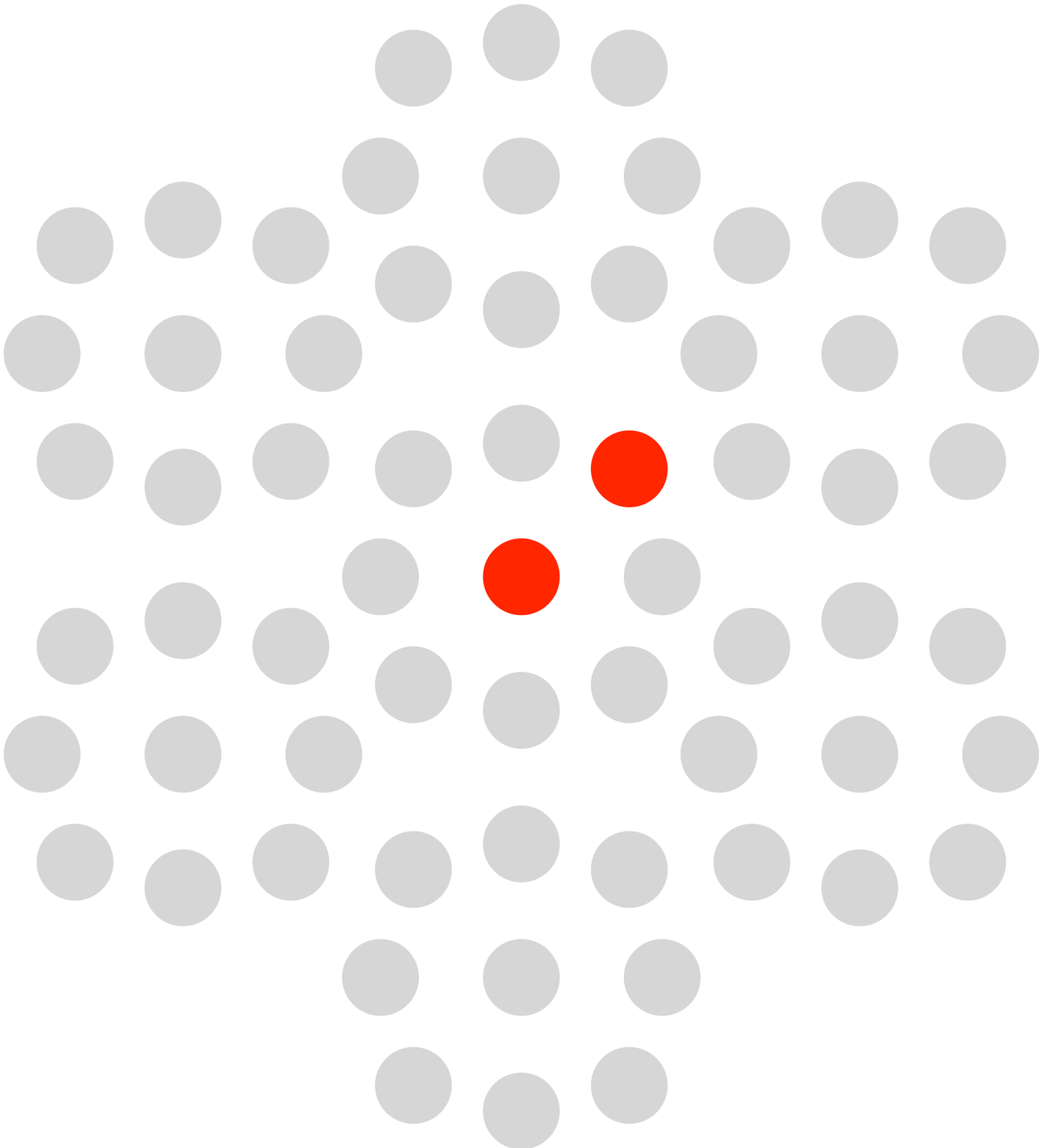
Structure
Evolution

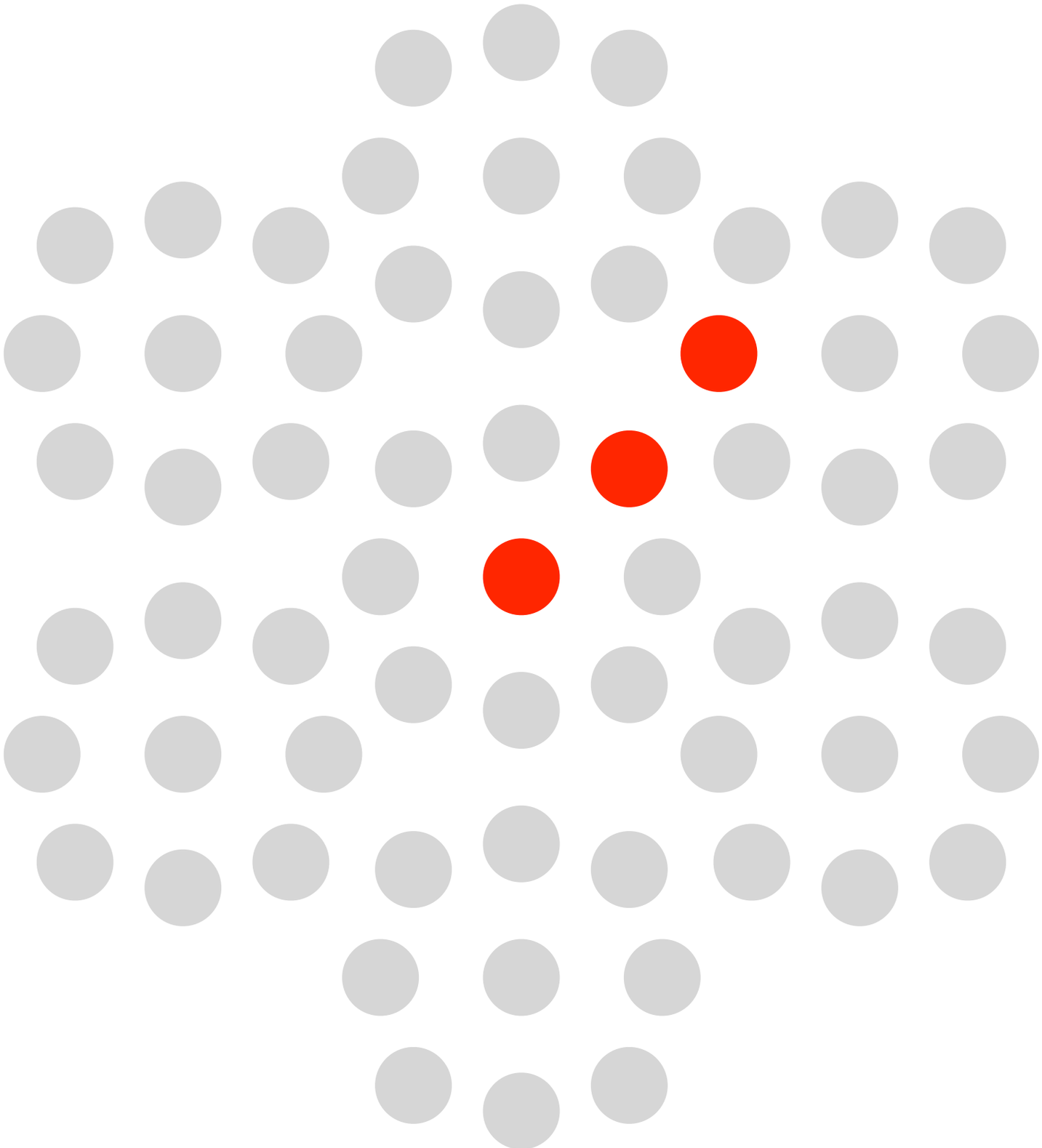
What if we do not know what to evolve?

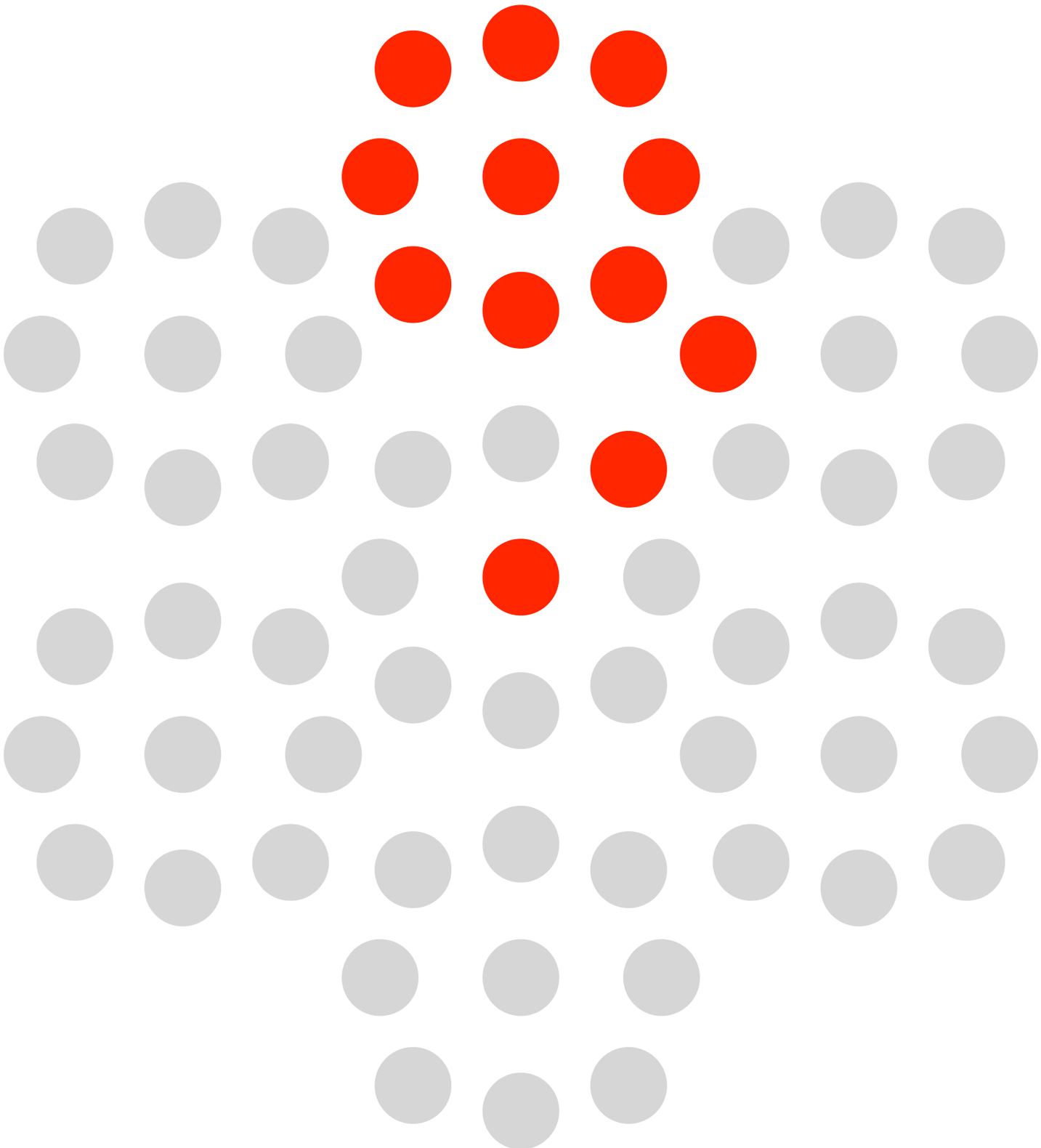


Prisma









Back in time

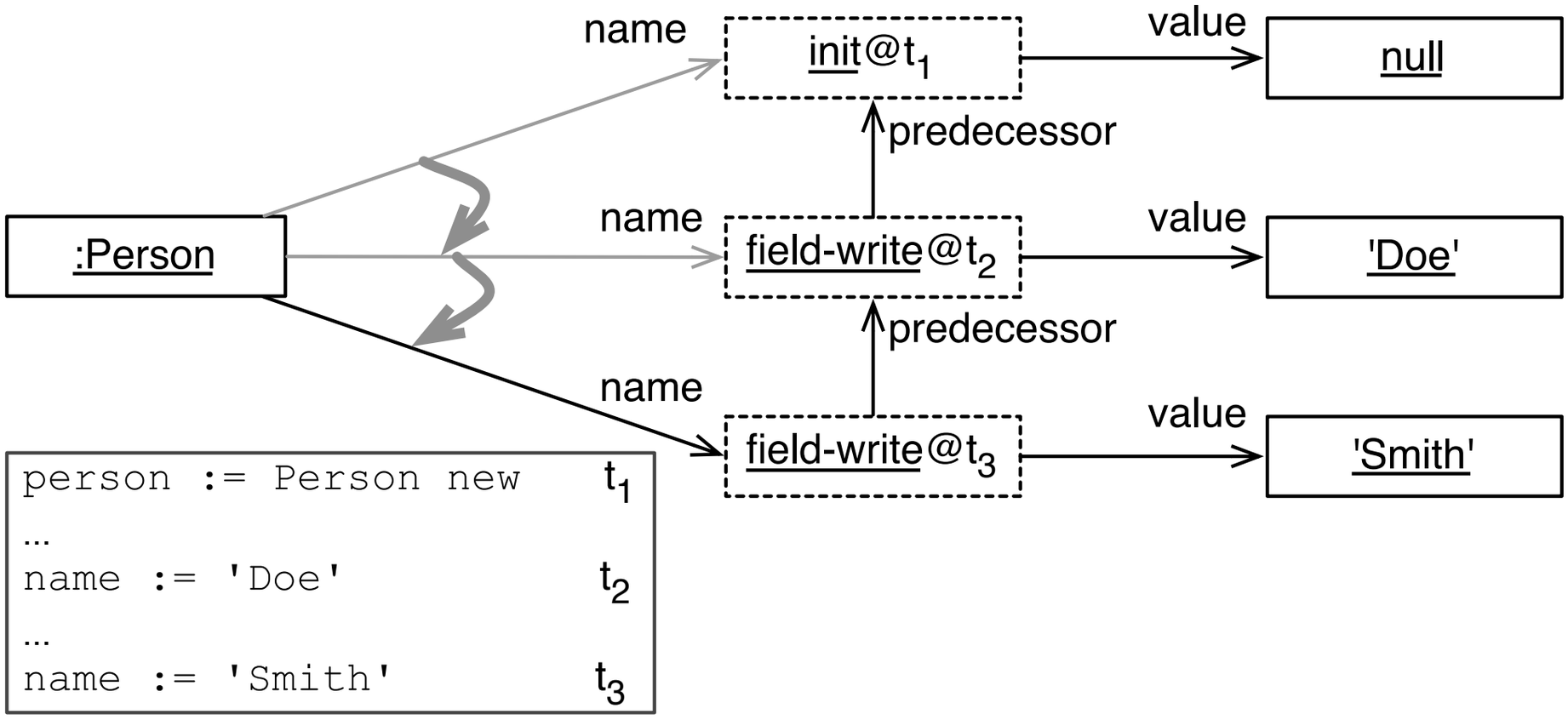
Debugger

Back in time

Debugger

Object Flow Debugger

Lienhard etal. ECOOP 2008


name
init@t1
value
null
predecessor
:Person
name
field-write@t2
value
'Doe'
predecessor
name
field-write@t3
value
'Smith'
person := Person new t1
...
name := 'Doe' t2
...
name := 'Smith' t3

Debugging

Profiling

Execution
Reification

Structure
Evolution

Talents

scg.unibe.ch/research/talents

Talents

scg.unibe.ch/research/talents

Dynamically composable units of reuse

Streams

Bifröst

scg.unibe.ch/research/bifrost

Object-Centric Debugger

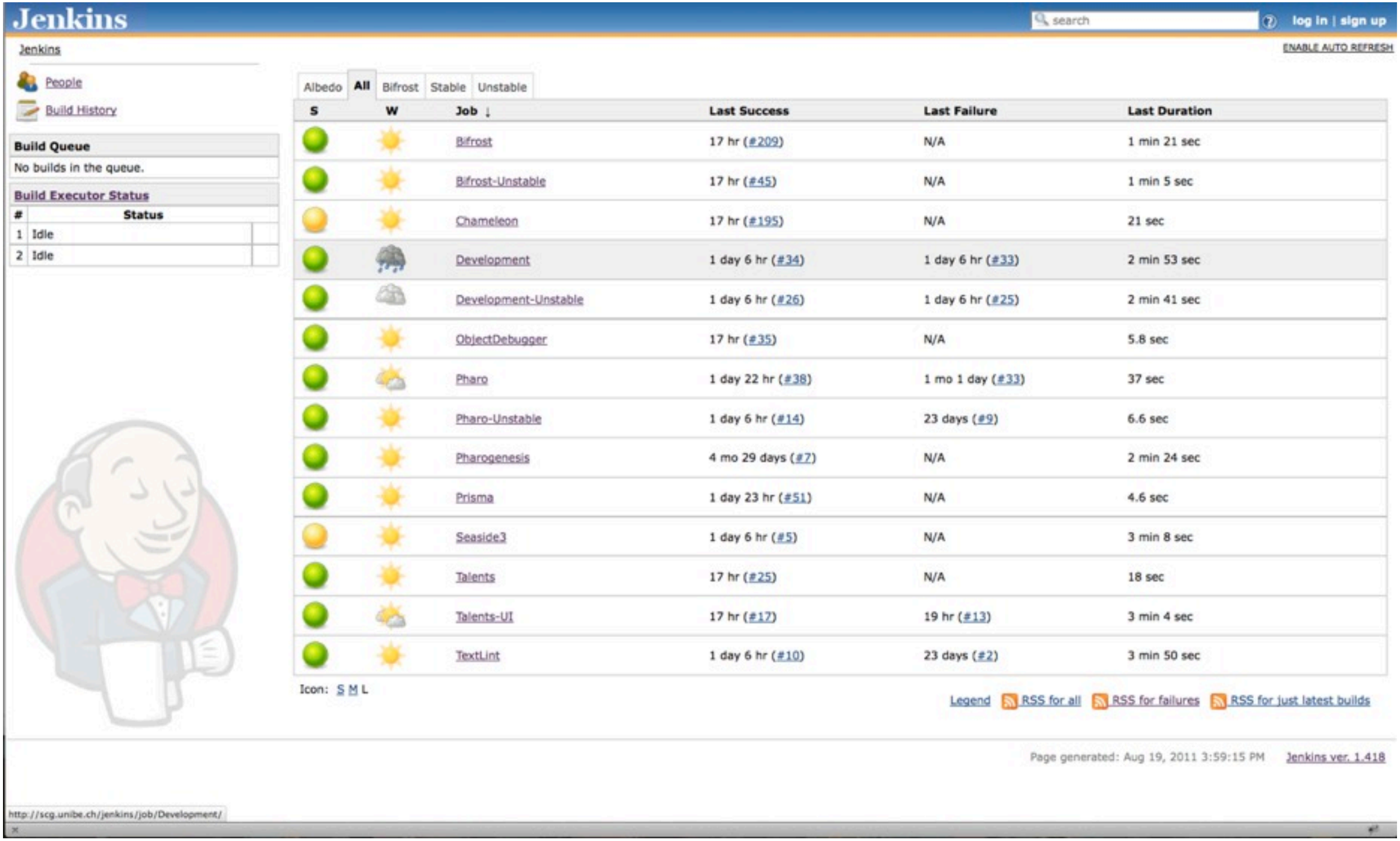
Prisma

Subjectopia

MetaSpy

Talents

Chameleon



scg.unibe.ch/jenkins/

Alexandre Bergel

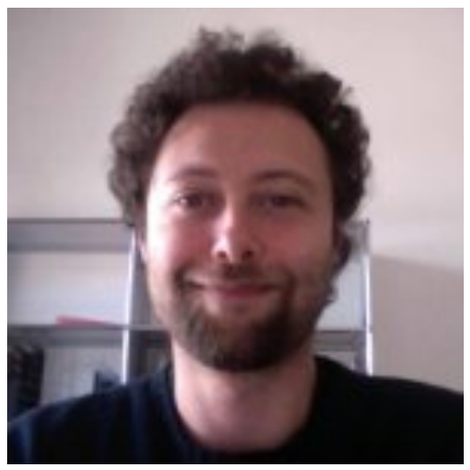
Marcus Denker

Stéphane Ducasse

Oscar Nierstrasz

Lukas Renggli

Tudor Gîrba



Fabrizio Perin

Bifröst

scg.unibe.ch/research/bifrost