

Object-Centric Reflection

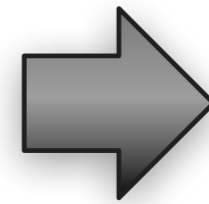
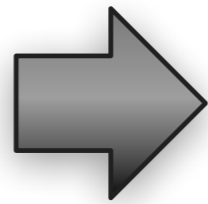
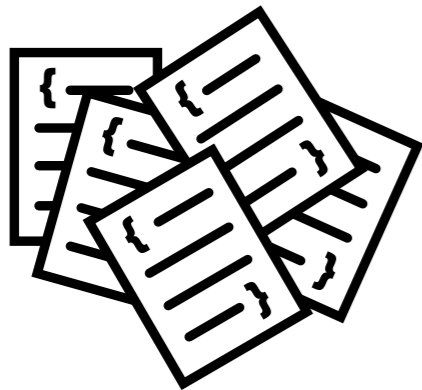
Jorge Ressia
Software Composition Group

Profiling

Profiling:

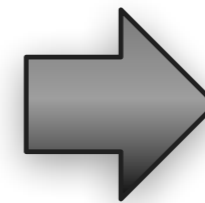
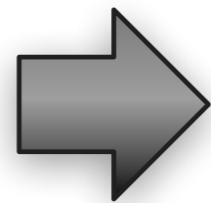
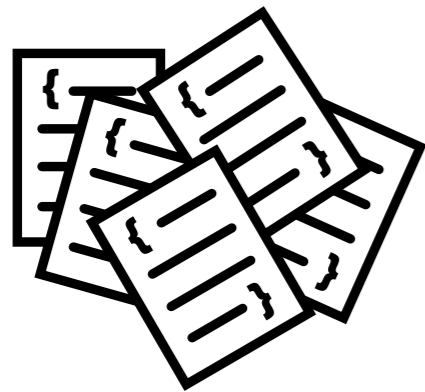
Is the activity of analyzing a program execution.

Profile

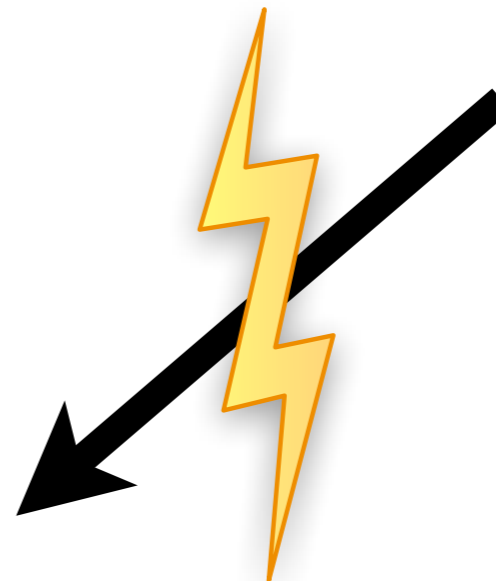


```
54.8% {11501ms} MOCanvas>>drawOn:  
54.8% {11501ms} MORoot(MONode)>>displayOn:  
30.9% {6485ms} MONode>>displayOn:  
  | 18.1% {3799ms} MOEdge>>displayOn:  
  ...  
  | 8.4% {1763ms} MOEdge>>displayOn:  
  | | 8.0% {1679ms} MOStraightLineShape>>display:on:  
  | | 2.6% {546ms} FormCanvas>>line:to:width:color: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

Profile

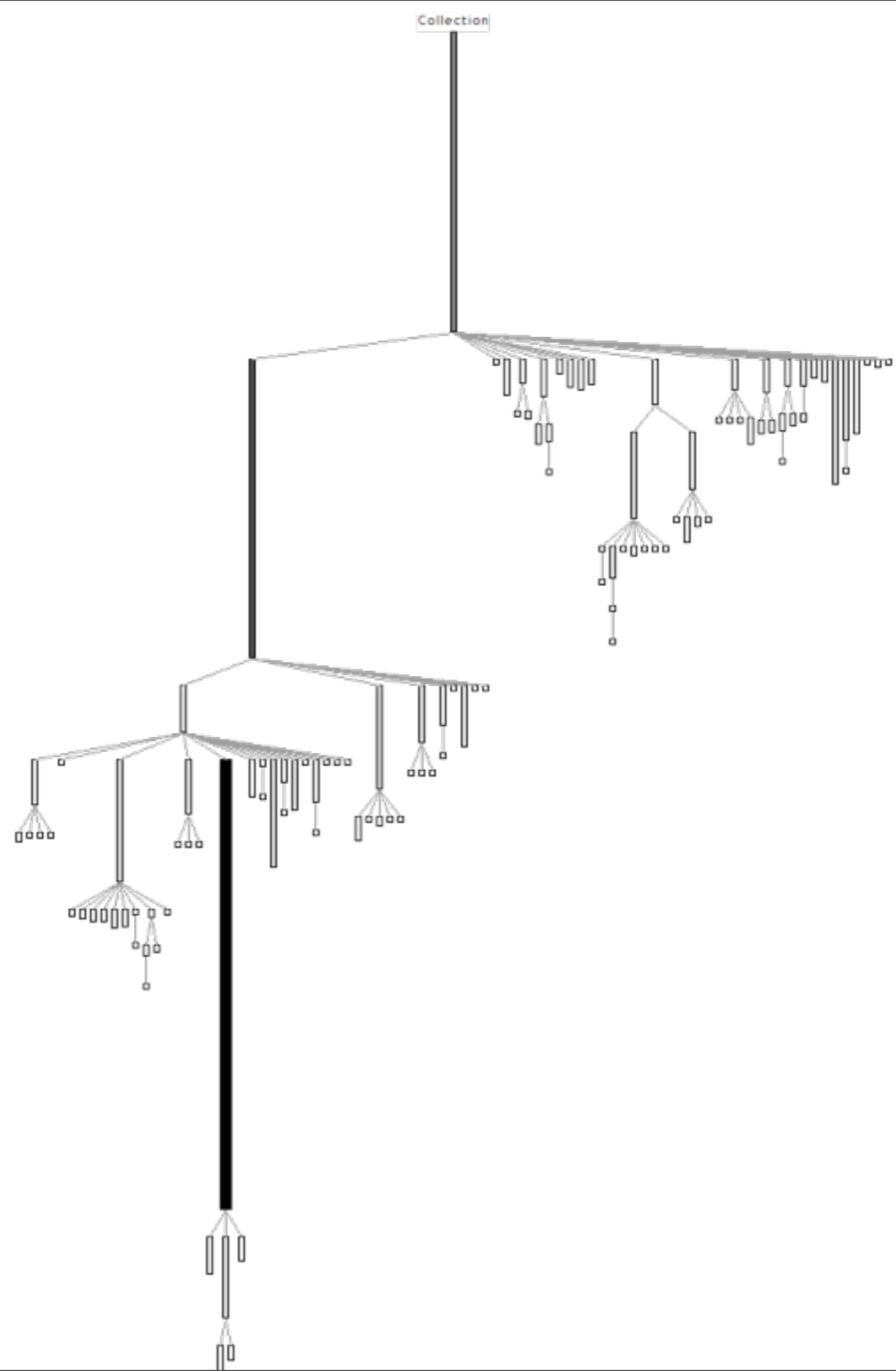


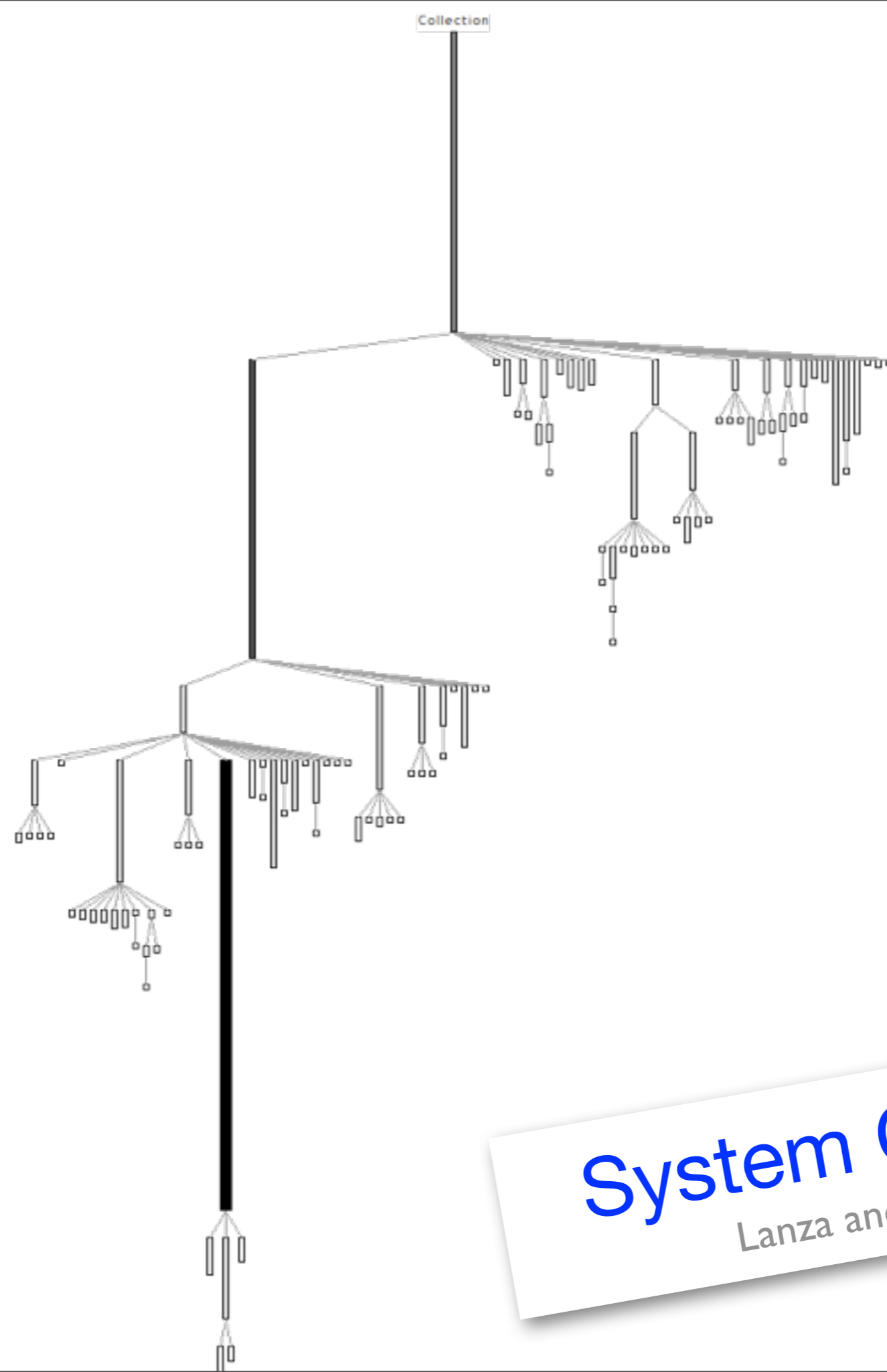
```
54.8% {11501ms} MOCanvas>>drawOn:  
54.8% {11501ms} MORoot(MONode)>>displayOn:  
30.9% {6485ms} MONode>>displayOn:  
  | 18.1% {3799ms} MOEdge>>displayOn:  
  ...  
  | 8.4% {1763ms} MOEdge>>displayOn:  
  | 8.0% {1679ms} MOStraightLineShape>>display:on:  
  | 2.6% {546ms} FormCanvas>>line:to:width:color: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



Domain

Mondrian





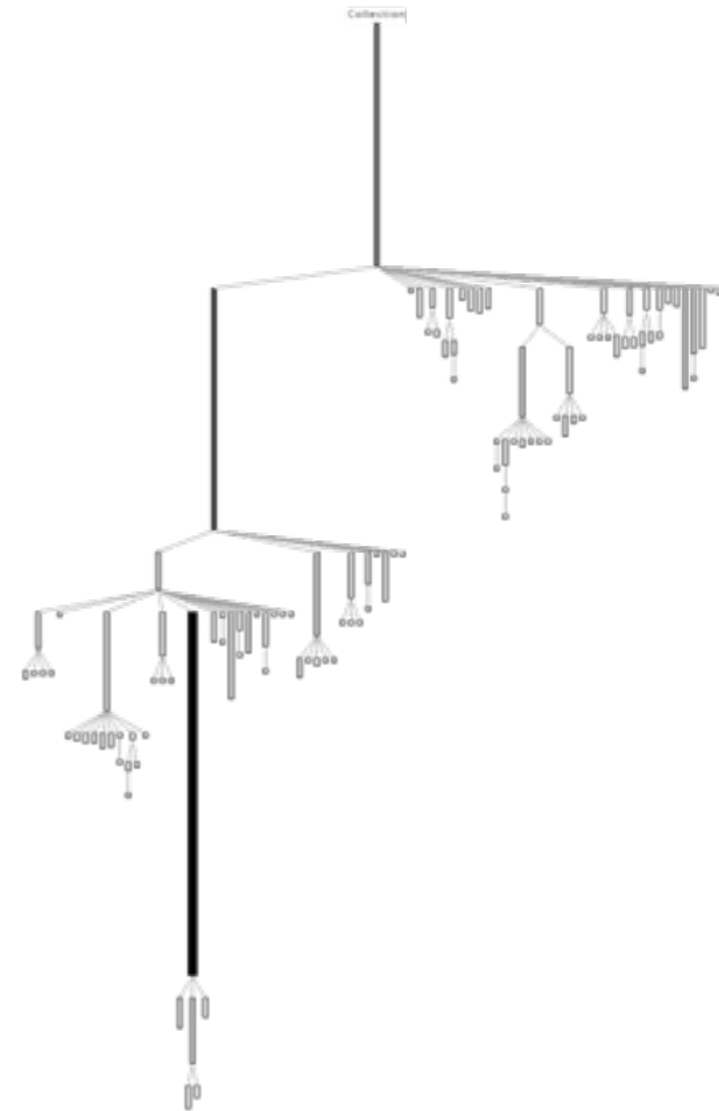
System Complexity

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:  
| ...
```

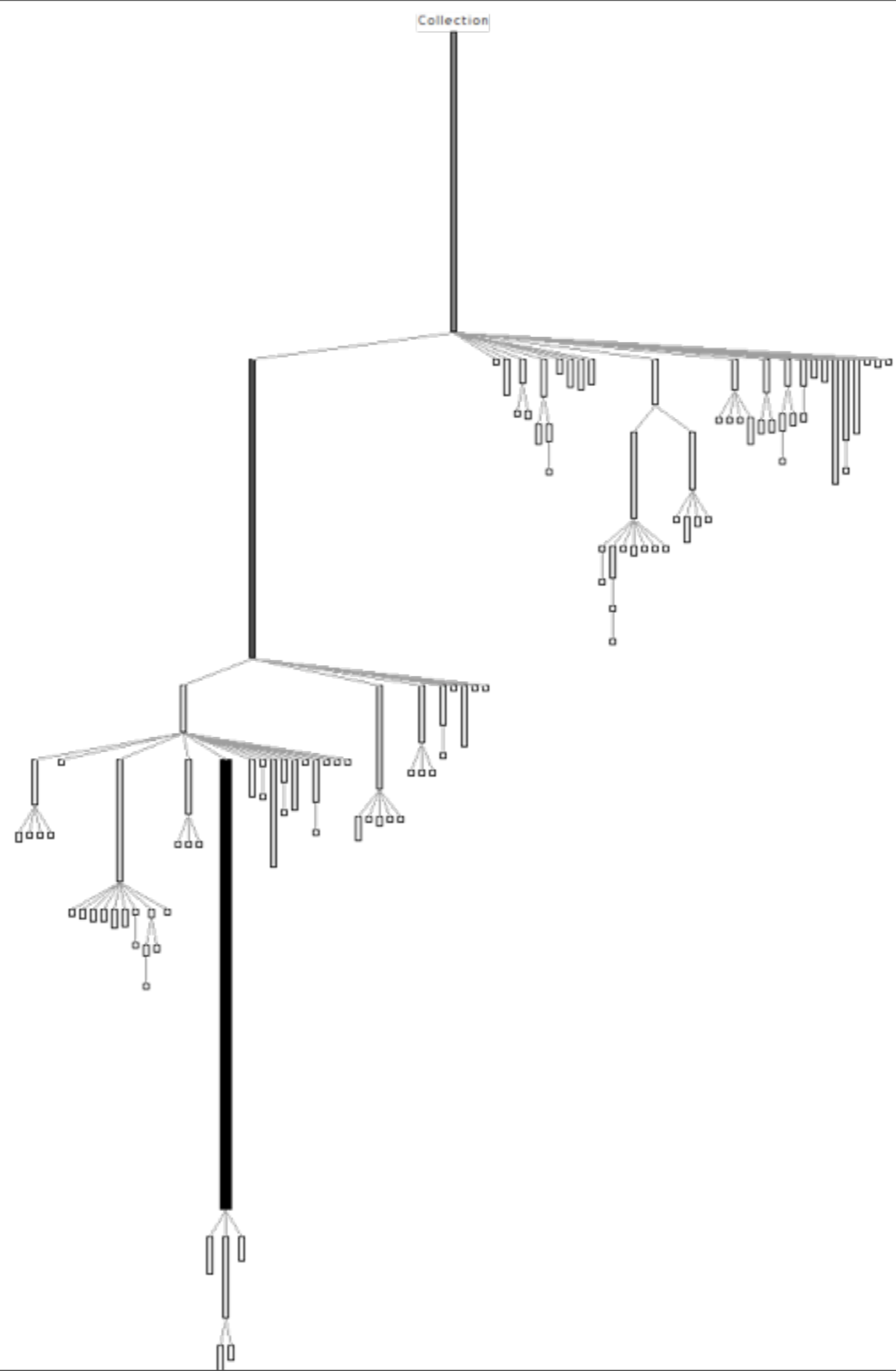


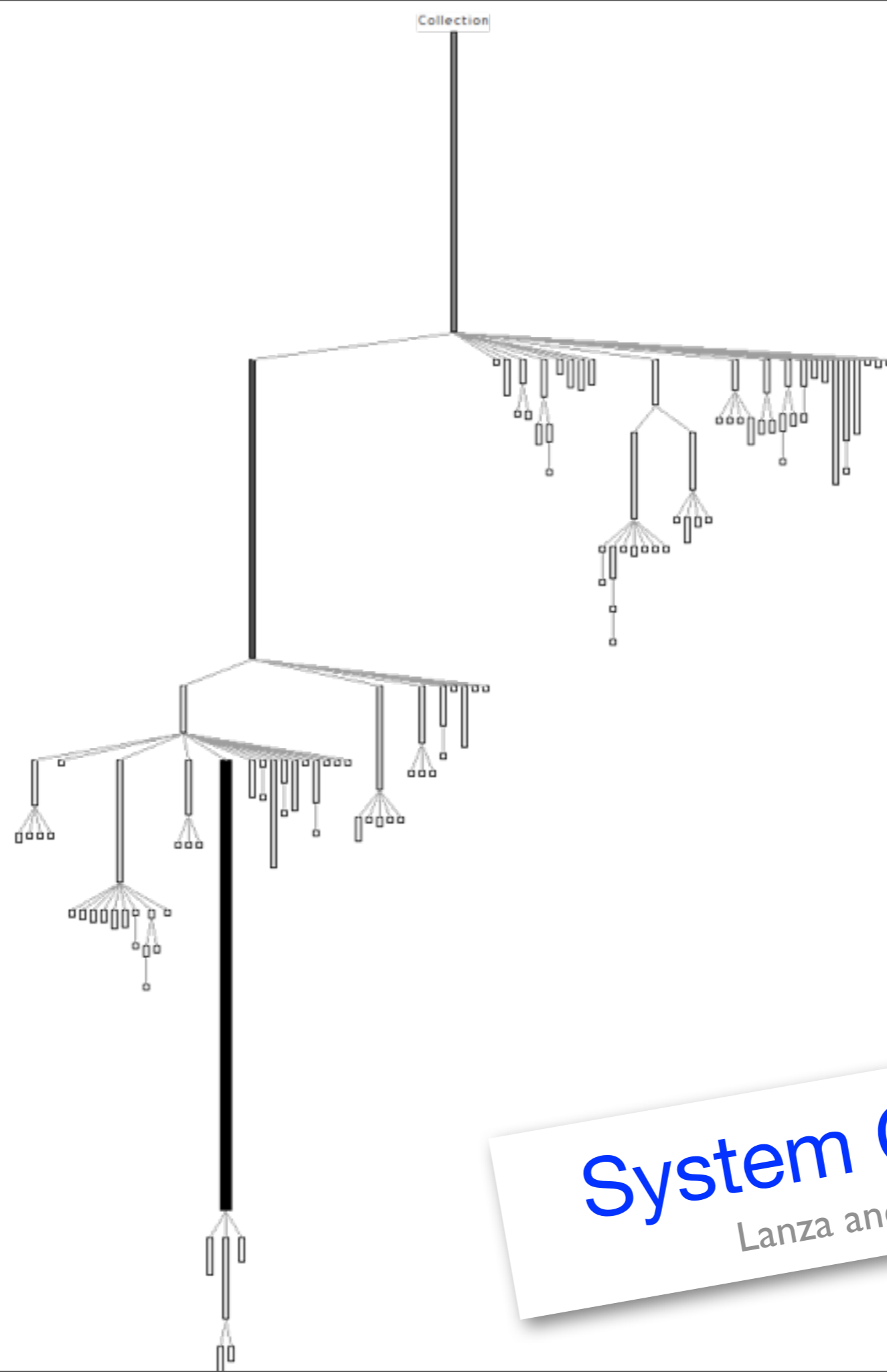
Debugging

Debugging:

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

Mondrian



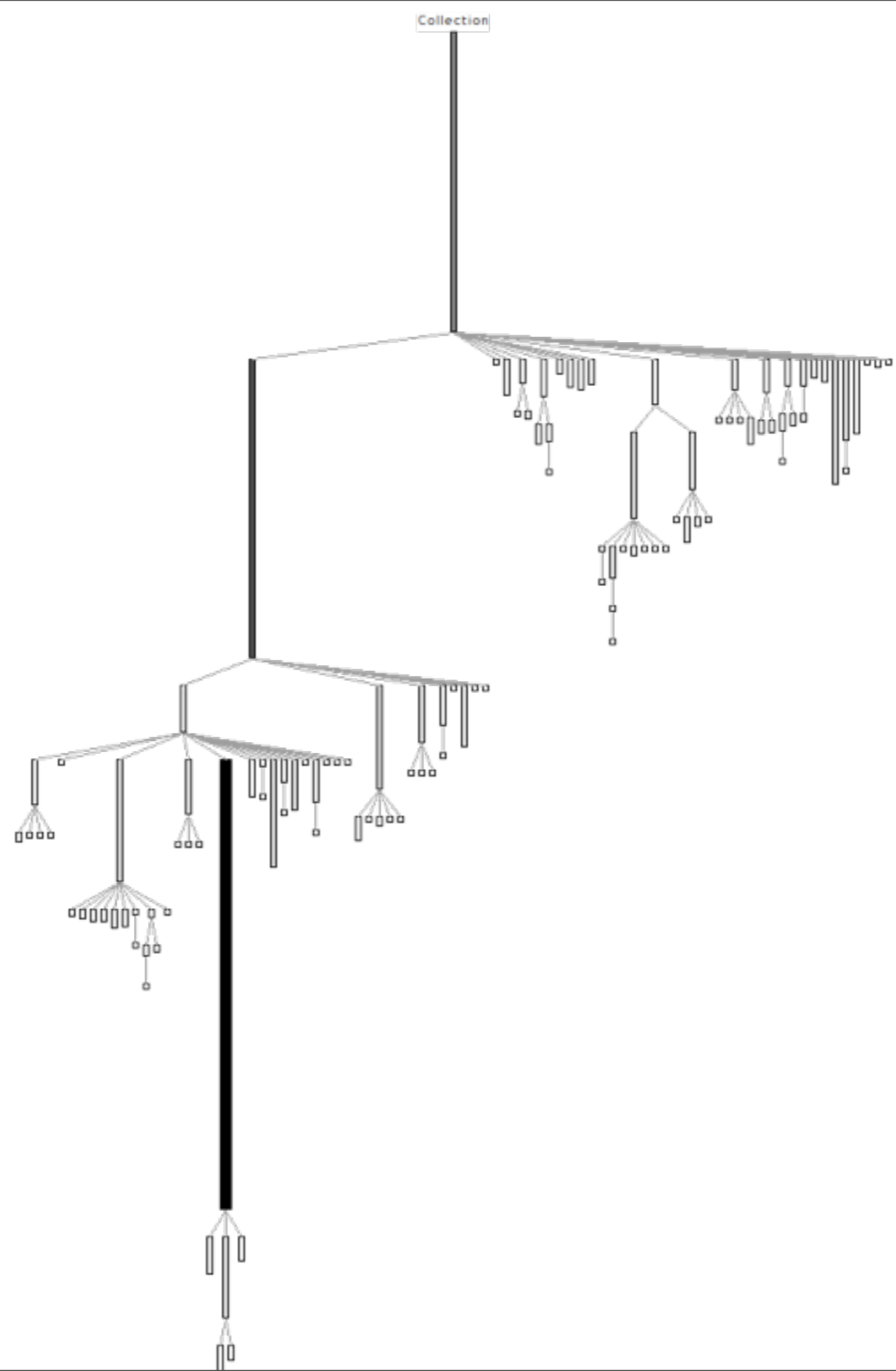


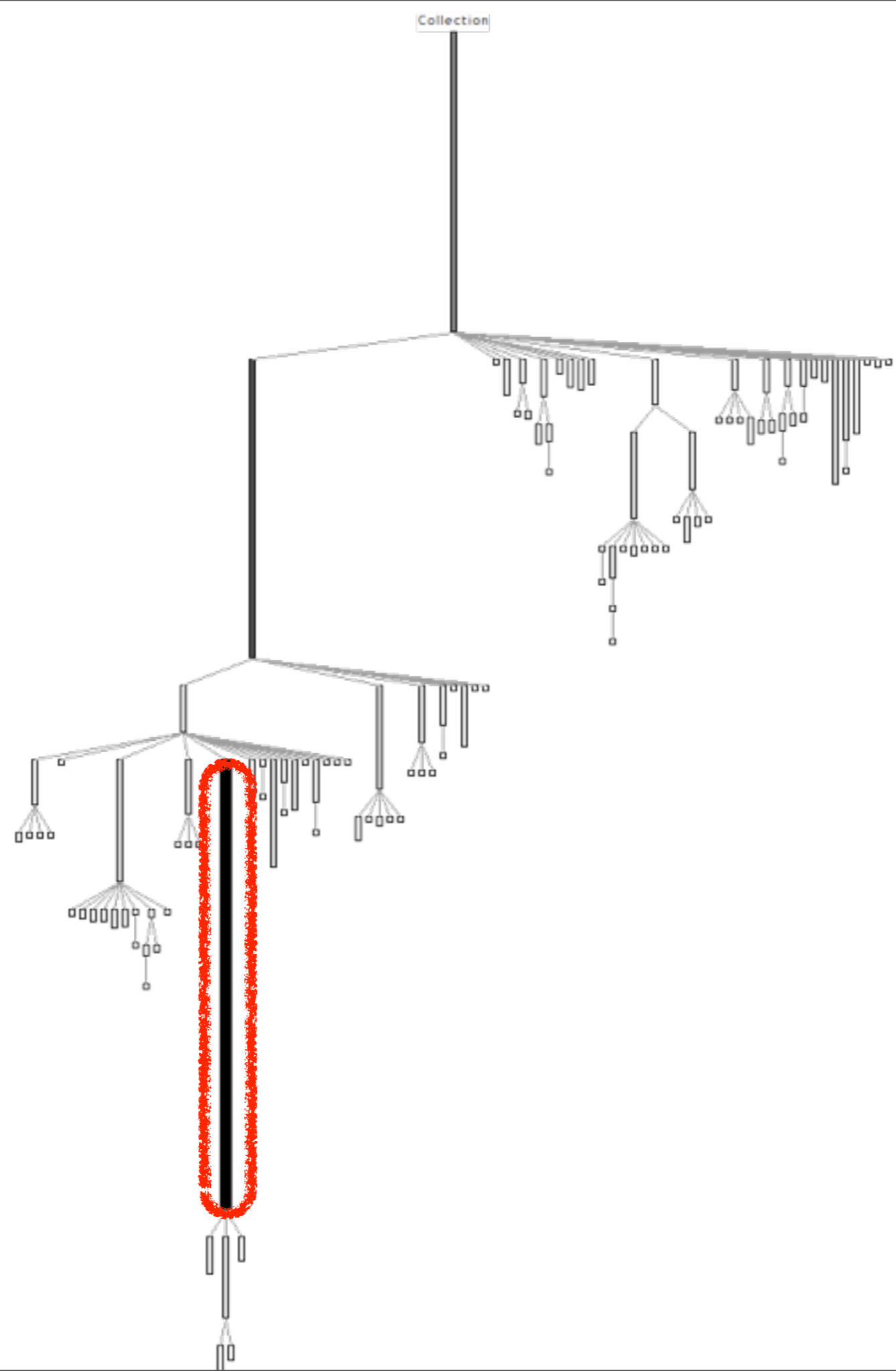
System Complexity

Lanza and Ducasse 2003

Rendering

Shape and Nodes

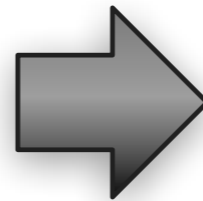
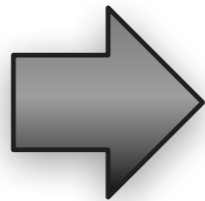
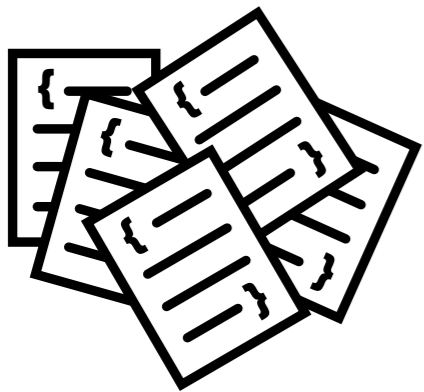




**How do we debug
this?**

Breakpoints

Conditional Breakpoints



```
TransparentBreakpoint
[receiver selector arguments] self metaClass | unlessFrom self TransparentBreakpoint signal | in [eachSelector | afterObjectTaken eachLine
MORootMORode] >> write@MetaCacheResursively
MORootObject >> write@MORode executeMethod
@MORode @MORode @MORode @MORode @MORode
@MORode @MORode @MORode @MORode @MORode
MORoot >> applyLayout
UndefinedObject >> nil
[anObject Do] in SmalltalkStart >> debugReceiverin
self valueProcessor terminateActive in BlockClosure >> nearProcess

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

name: selector with: arguments in: receiver
[aCompiledMethod]
aCompiledMethod = [: methodDict
                    at: selector
                    ifAbsent: [ * (receiver class methodDict) at: selector
                                runOriginalBehaviorFor: selector
                                with: arguments
                                in: receiver ] generate

methodDict
at: selector
put: aCompiledMethod

receiver
withArgs: arguments
executeMethod: aCompiledMethod

self
all mol vars
boundObjects
methodDict
methodDict:mal
commands
currentContext

thisContext
stack top
all temp vars
selector
arguments
receiver
aCompiledMethod
```


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 et al.

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)



```
TalentTestObject>>halt
TalentTest>>testDefineMethodComposition
TalentTest(TTestCase)>>performTest
[self setUp:self performTest] in TalentTest(TTestCase)>>runCase
BlockClosure>>ensure
TalentTest(TTestCase)>>runCase
[aTalent announce: TestCaseStarted withResult: self;aTalent runCase;aTalent announce: TestCaseEnded withResult: self;self addPa
BlockClosure>>onDo:
TestResult>>runCase
TalentTest(TTestCase)>>run
[each | self changed: each each run: aResult] in TestSuite>>run
ObjectCollection>>do

Proceed Restart Info Over Through Full Stack Run to Here Where Create

testDefineMethodComposition
[aTalent, anObject, anotherTalent, compoundTalent]
anObject = BFDualCounter.new;
aTalent = Talent.new;
anotherTalent = Talent.new;
self.halt;
aTalent.defineMethodNamed :#counter3 performing :counter3 counter2 = counter2 + 1;
anotherTalent.defineMethodNamed :#counter4 performing :counter4 counter2 = counter2 + 1;

compoundTalent = aTalent, anotherTalent;
anObject.acquires compoundTalent;

*anObject acquires: aTalent;
anObject acquires: anotherTalent;

anObject.instVarNamed :#counter1 put: 0;
anObject.instVarNamed :#counter2 put: 0;
anObject.counter3;
self.assert anObject.counter1 == 0;
self.assert anObject.counter2 == 1;
anObject.counter4;
self.assert anObject.counter1 == 0.

self
all inst vars
testSelector

thisContext
stack top
all temp vars
aTalent
anObject
anotherTalent
compoundTalent
```

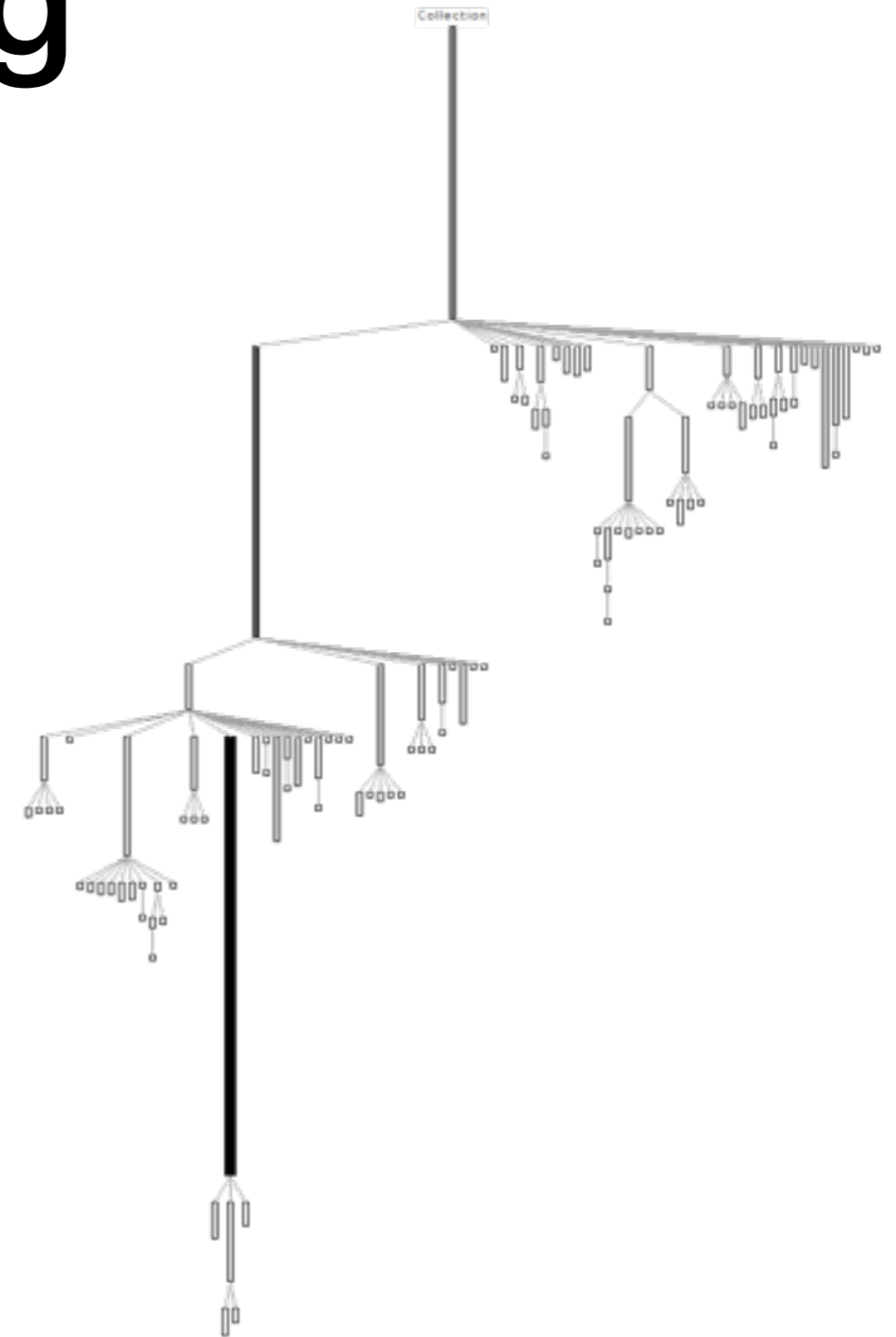
**What is the
problem?**

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)

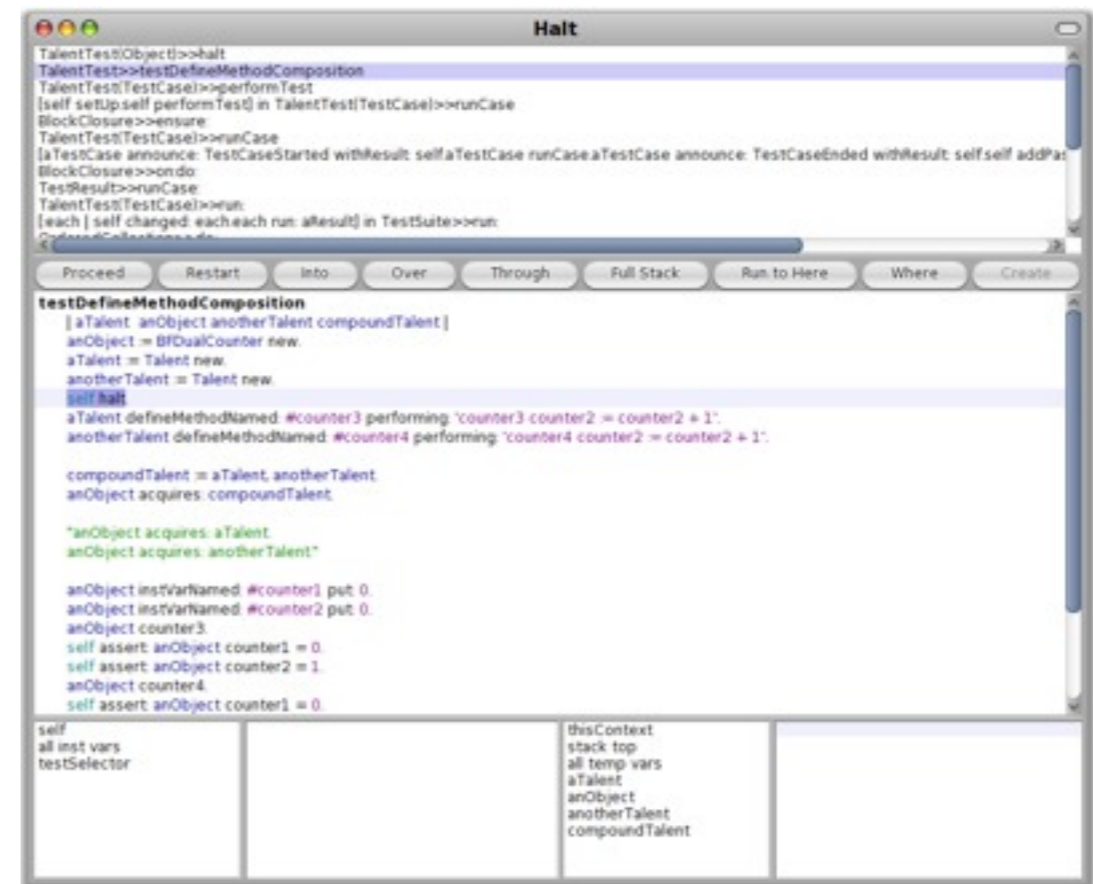
What are the values of the argument at runtime? (Q.19)

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



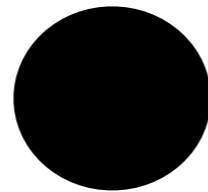
Organize the Meta-level

Explicit Meta-objects

Class



Meta-object



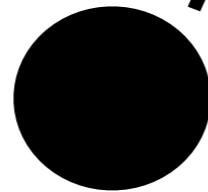
Object



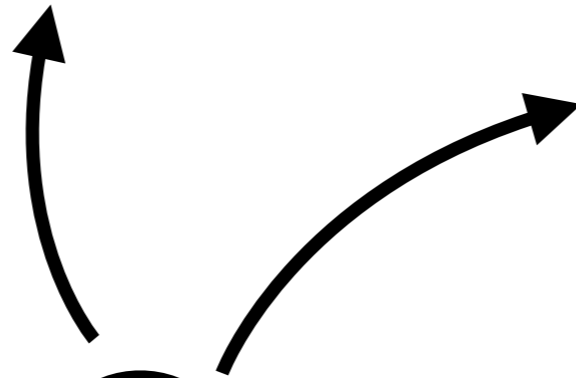
Class



Meta-object



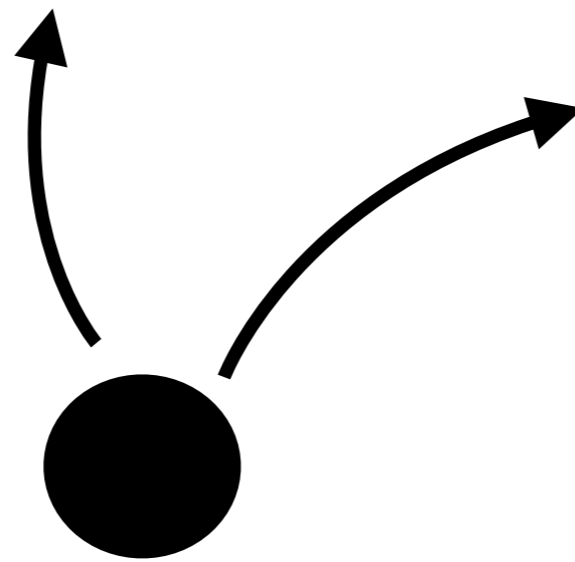
Object



Class



Meta-object



Evolved Object

Debugging

Profiling

**Execution
Reification**

**Structure
Evolution**

Debugging

Profiling

**Execution
Reification**

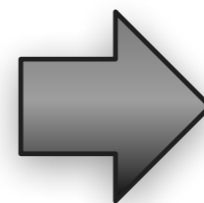
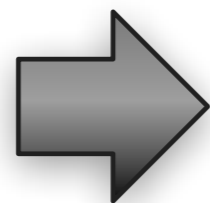
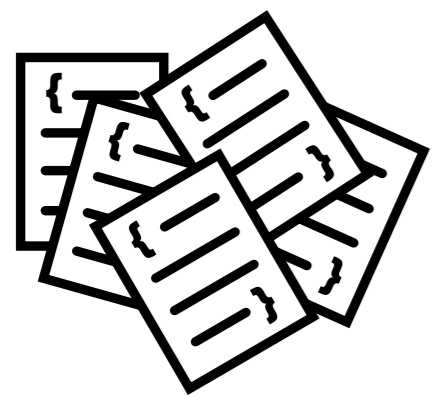
**Structure
Evolution**

Object-Centric Debugging

Object-Centric Debugging

ICSE 2012

J. Ressa, A. Bergel and O. Nierstrasz



```
TransparentBreakpoint
[receiver selector arguments] self metaClass | unloadFrom: self TransparentBreakpoint signal: in: [eachSelector | afterObjectTaken: eachSel
MCRoomMCRModel>>withMetaCacheResursively
MCRoomMCRModel>>withArgs: executeMethod
@MCRoomMCRModelObject>>withMetaCache
@MCRoomMCRModelObject>>withMetaCache
MCRoom>>applyLayout
UndefinedObject>>DoIt
[anObject DoIt] in: SmalltalkEditor>>debugReceiver:
self valueProcessor terminateActiveIn: BlockClosure>>nearProcess

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

name: selector with: arguments in: afterEver
[aCompletedMethod]
 aCompletedMethod = [: methodDict
  at selector
  ifAbsent | " [after receiver class methodDict] at selector
    runOriginalBehaviorFor selector
    with arguments
    in: afterEver | generate

methodDict
  at selector
  put: aCompletedMethod

~ [receiver
withArgs arguments
executeMethod: aCompletedMethod

self
all mol vars
boundObjects
methodDict
methodDict:mal
commands
currentContext

thisContext
stack top
all temp vars
selector
arguments
afterEver
aCompletedMethod
```





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:

```

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	#testGetSource	thisContext	nil
all inst vars		stack top	
testSelector		all temp vars	
		object	
		originalCode	
		newSource	
		t	

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 run to here
object (t) adaptationInsensitiveSource.
object (t) adaptationInsensitiveSource.
newSource (DictAt: #increment) generate adaptationInsensitiveSource.
self as
object

```

- 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 (l)
- 🛑 Debug it (D)
- 🛑 Profile it
- 🔍 Find...(f)
- 🔍 Find again (g)
- 🔍 Extended search...
- 🔄 Do again (j)
- ↶ Undo (z)
- 📄 Copy (c)

	thisContext	nil
	stack top	
	all temp vars	
	object	
	originalCode	
	newSource	
	t	

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

```

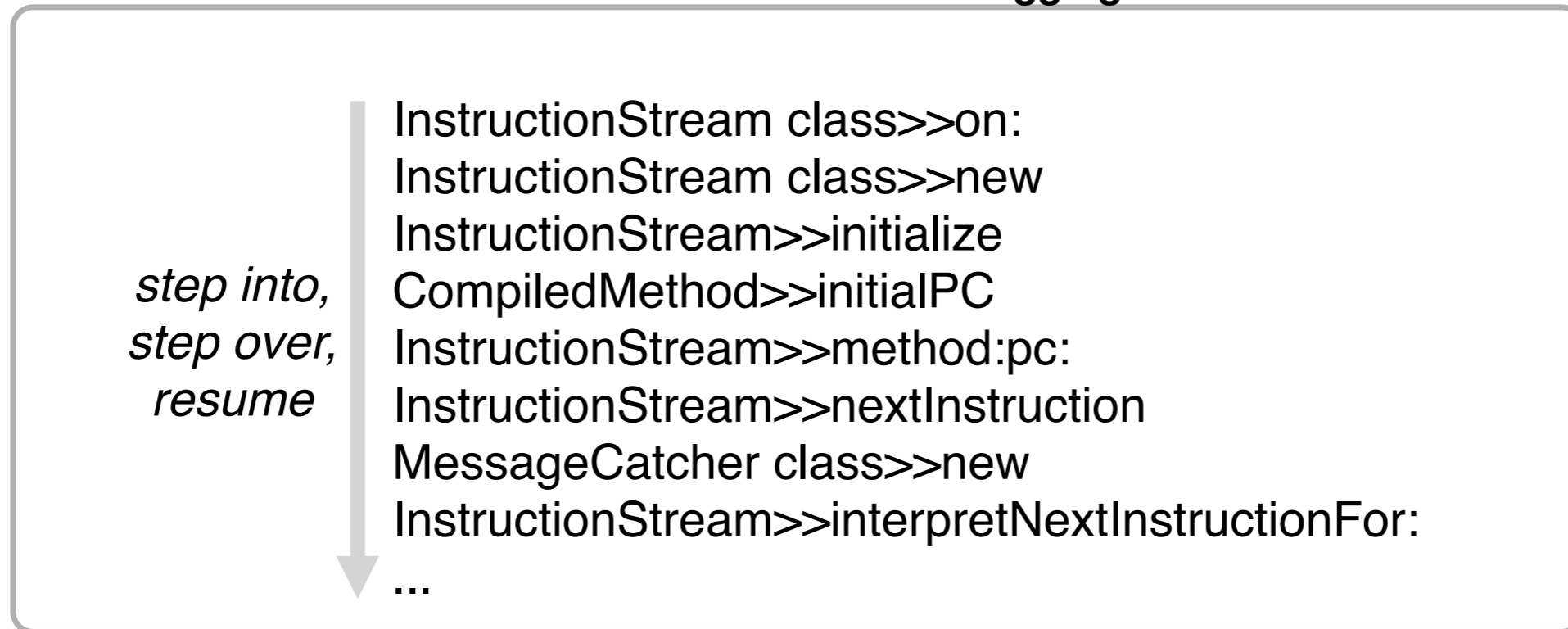
| object originalCode newSource t |
self halt.
originalCode := (MockObject >> #increment) adaptationInsensitiveSource.
object := MockObject new.
object haltAtStateChange.
newSource := (object metaObject methodDictAt: #increment) generate adaptationInsensitiveSource.
self assert: originalCode = newSource.
object metaObject unbindFrom: object.

```

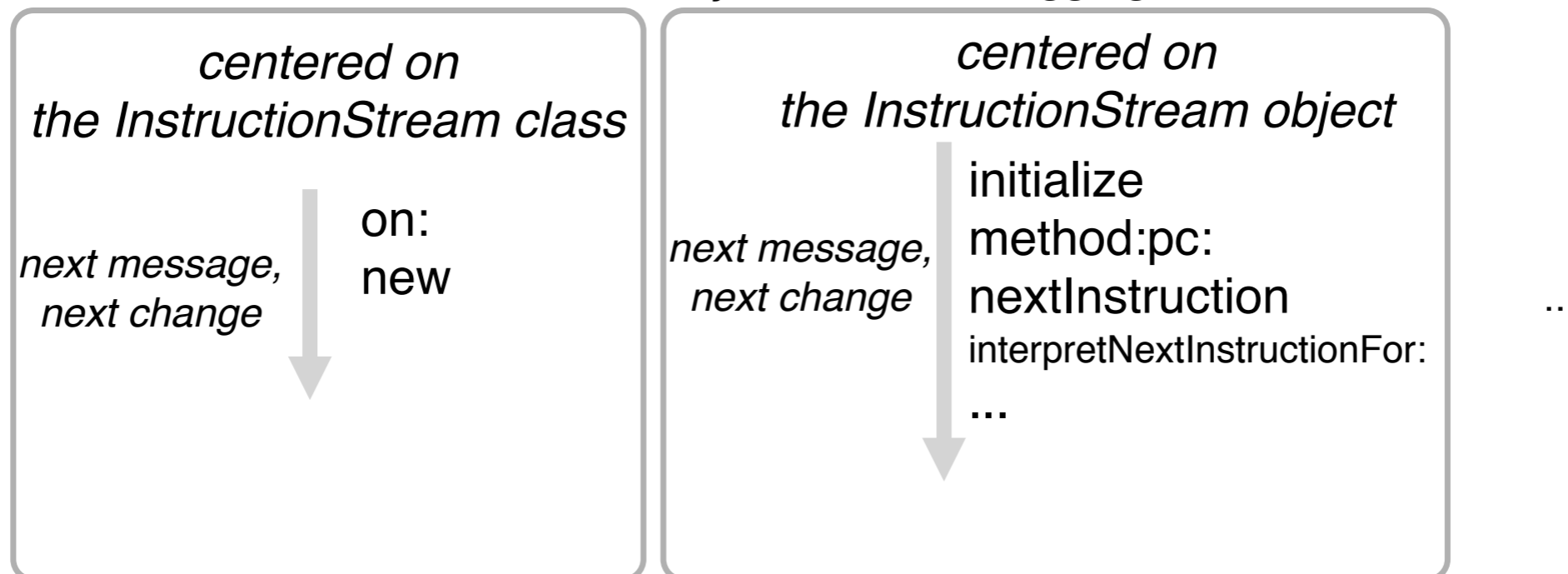
self	#testGetSource	thisContext	nil
all inst vars		stack top	
testSelect		all temp vars	
		object	
		originalCode	
		newSource	
		t	

- Inspect (i)
- Explore (l)
- 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

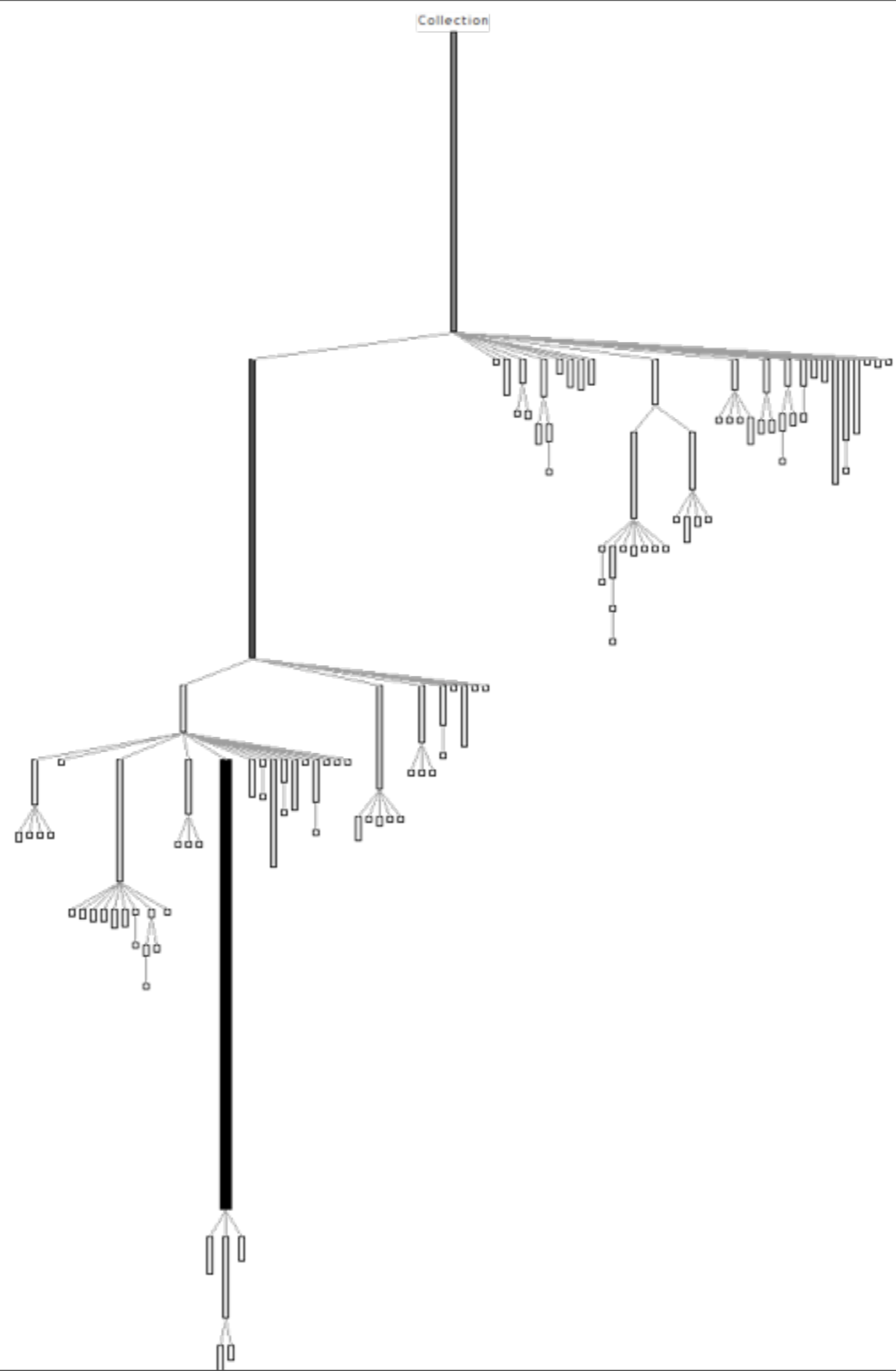


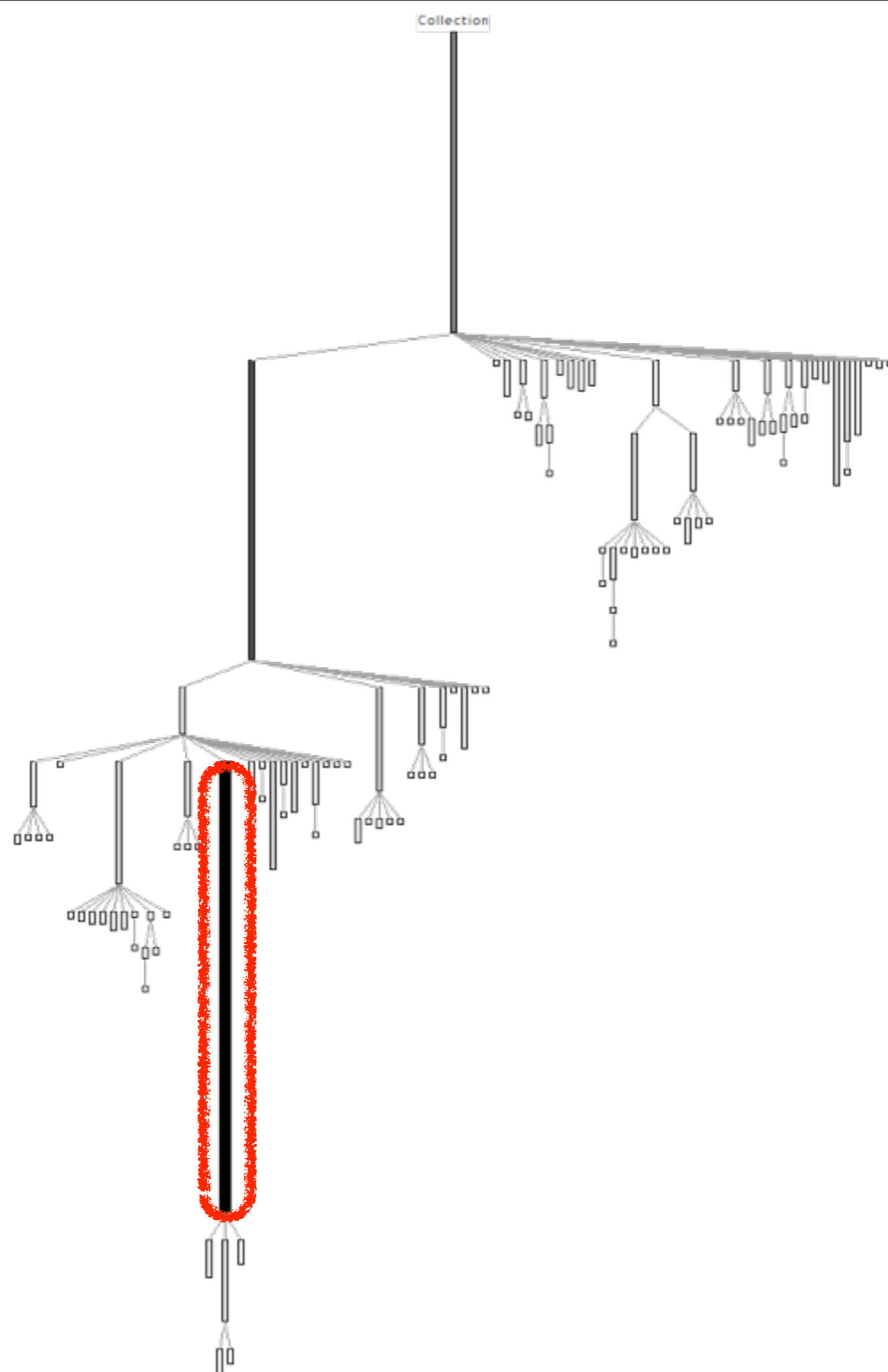
object-centric debugging



Mondrian

Shape and Nodes





**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

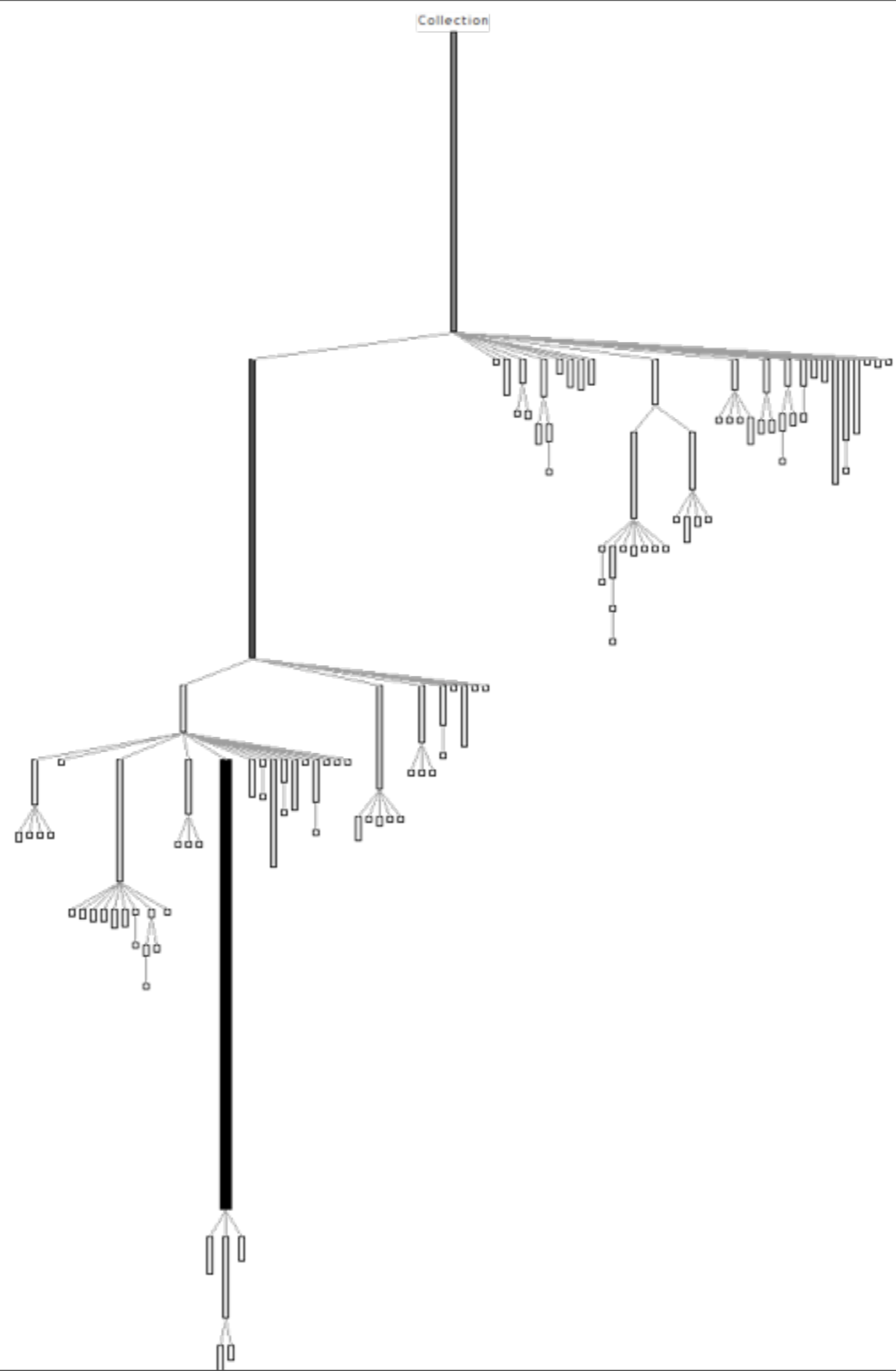
Structure
Evolution

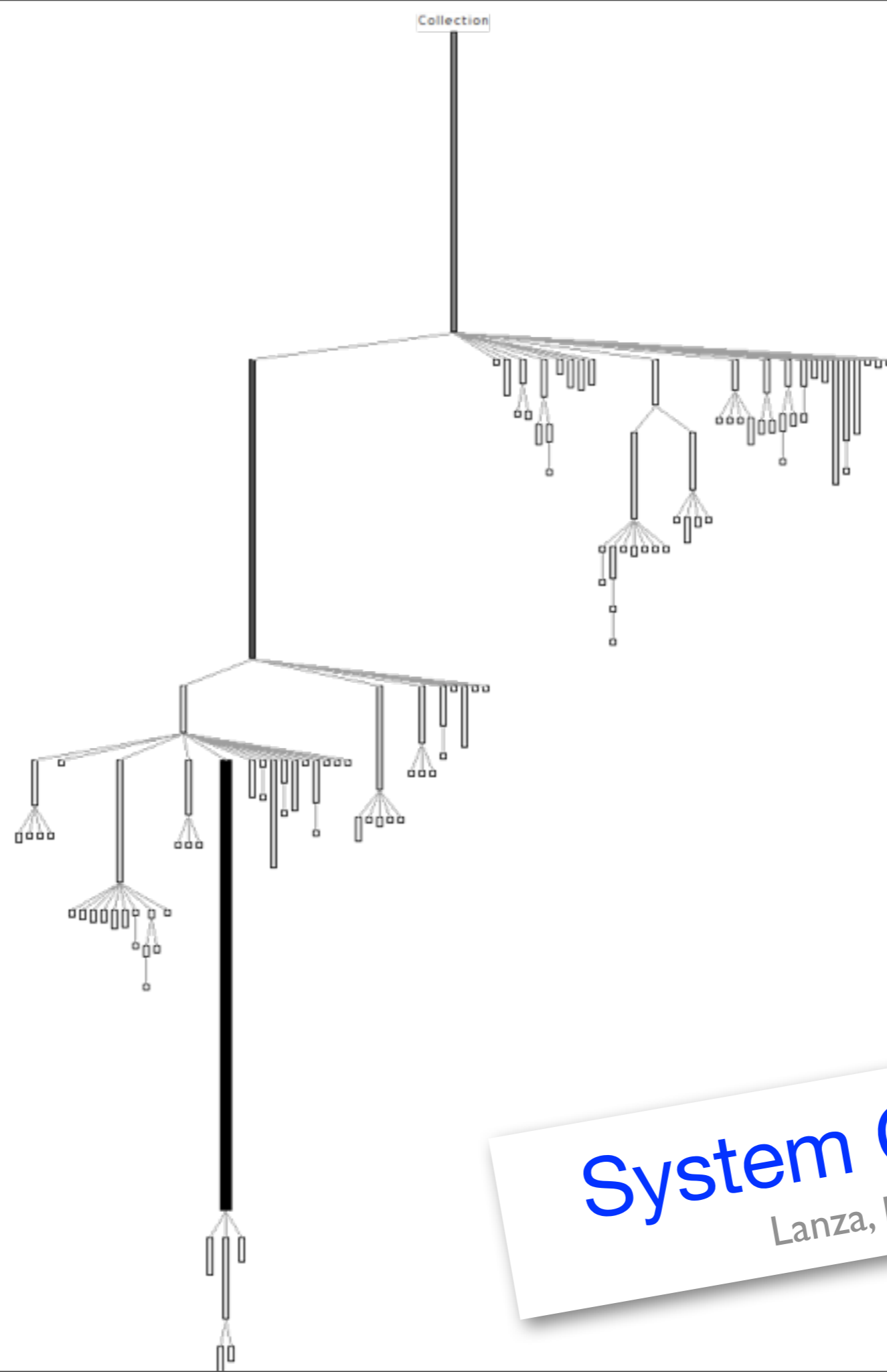
MetaSpy

MetaSpy

TOOLS 2011
Bergel et al.

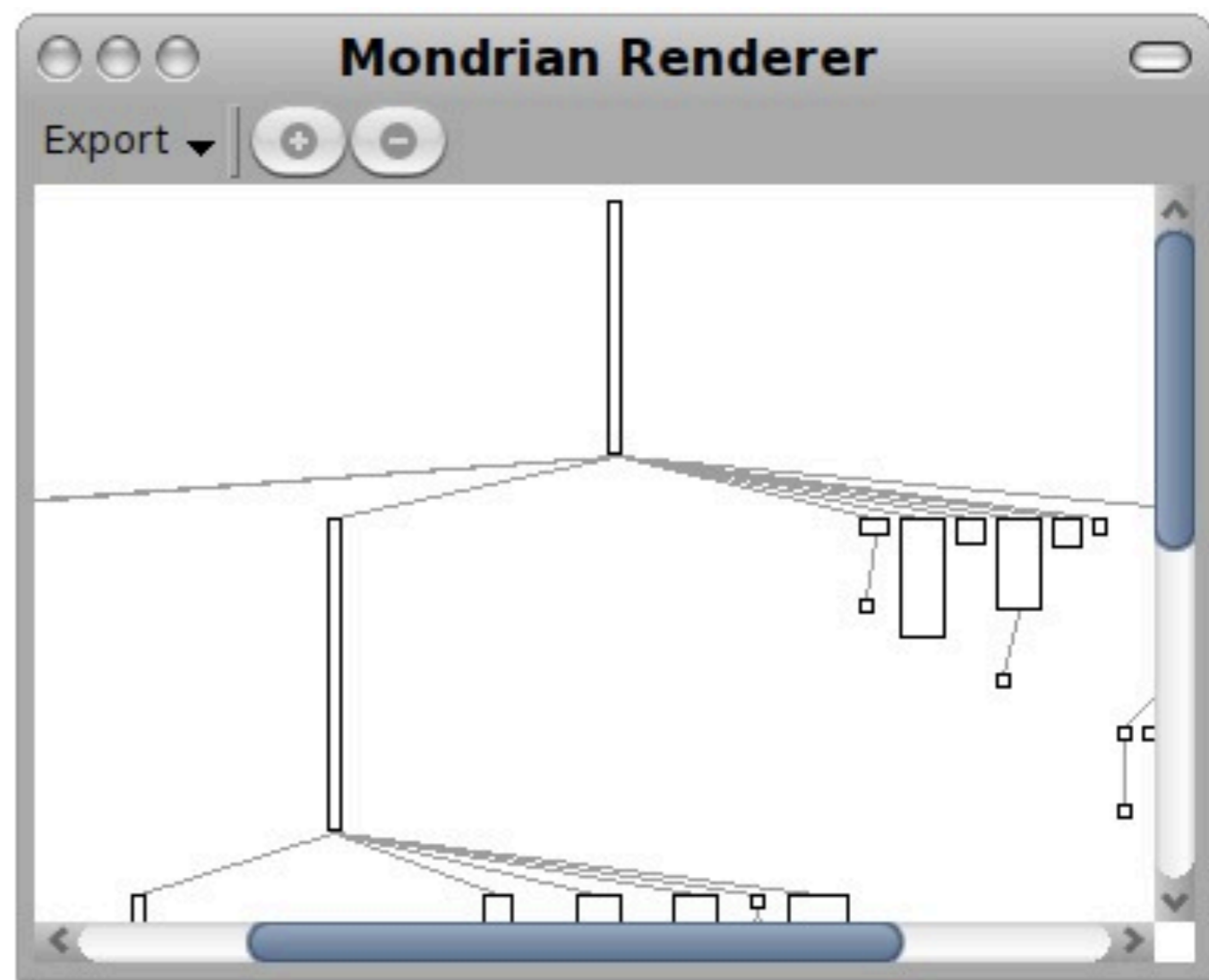
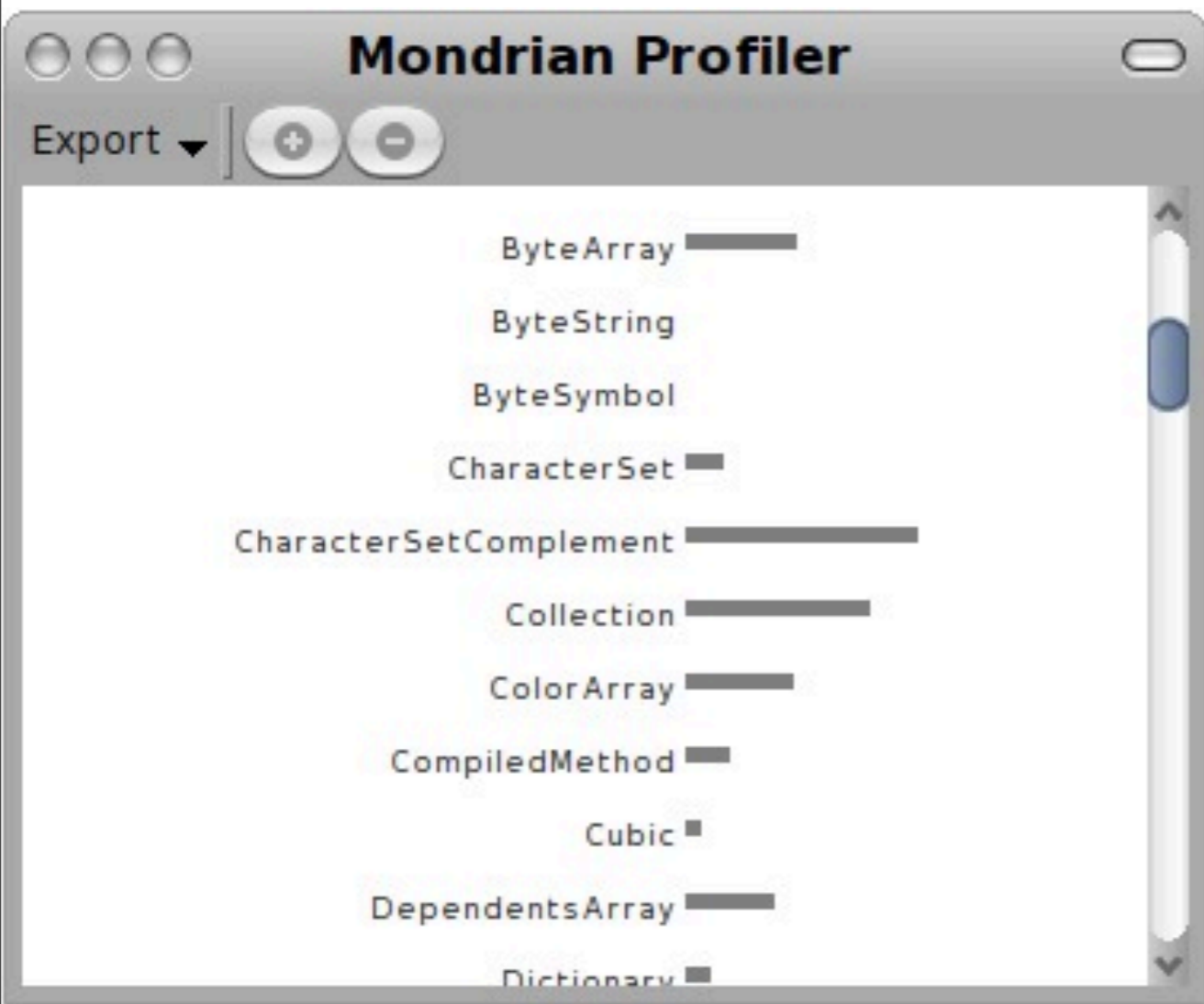
Mondrian Profiler





System Complexity

Lanza, Ducasse 2003



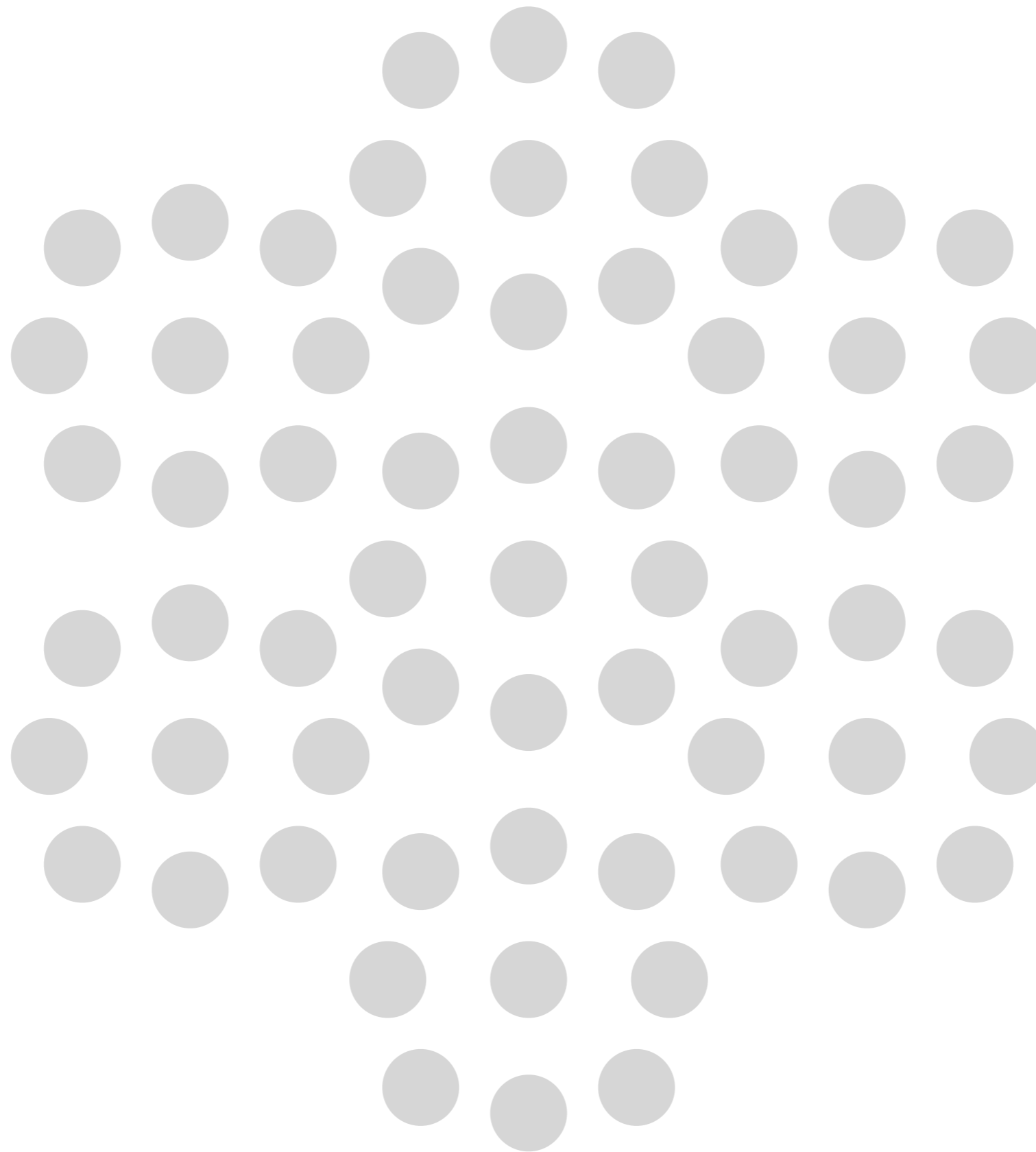
Debugging

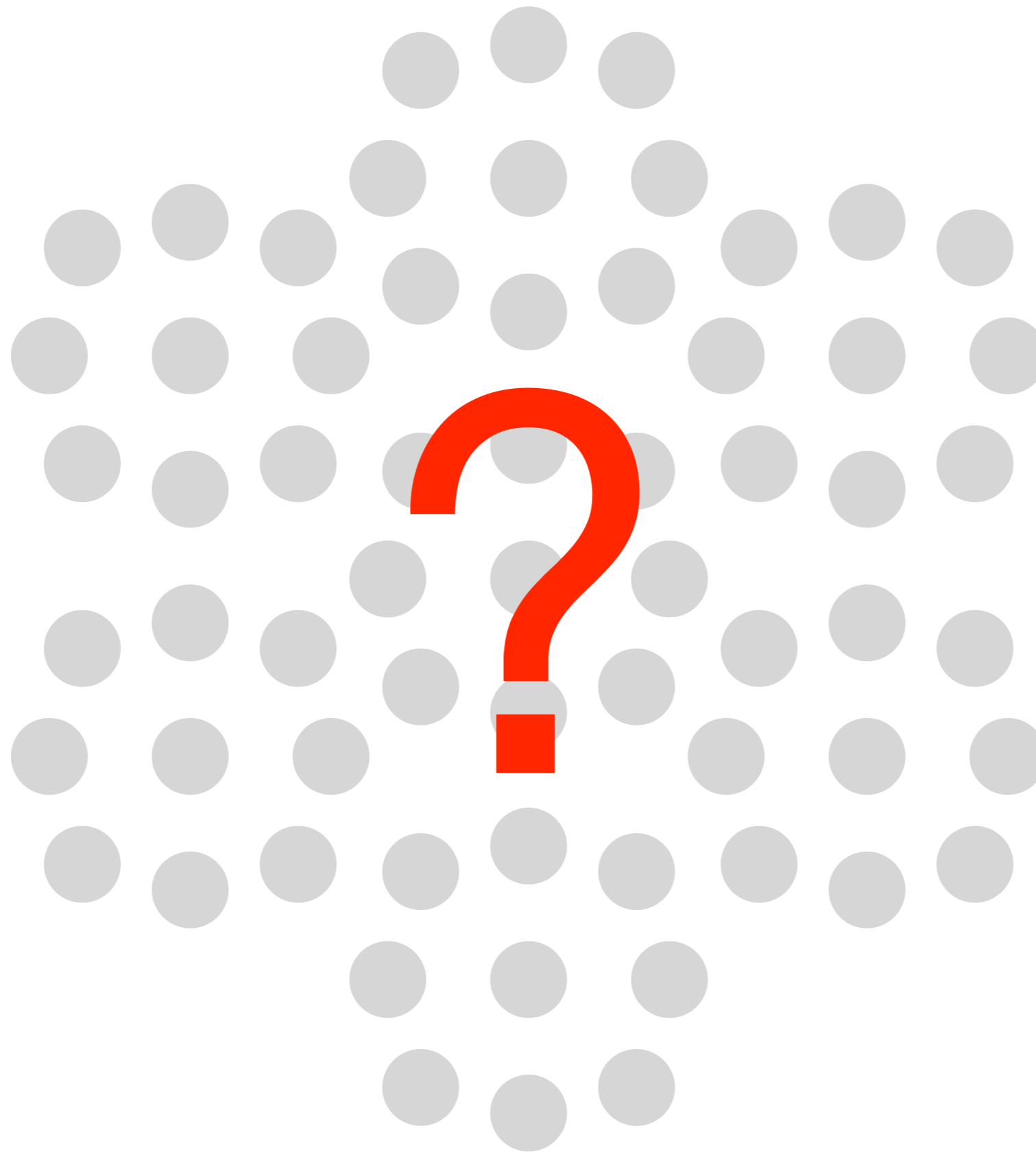
Profiling

**Execution
Reification**

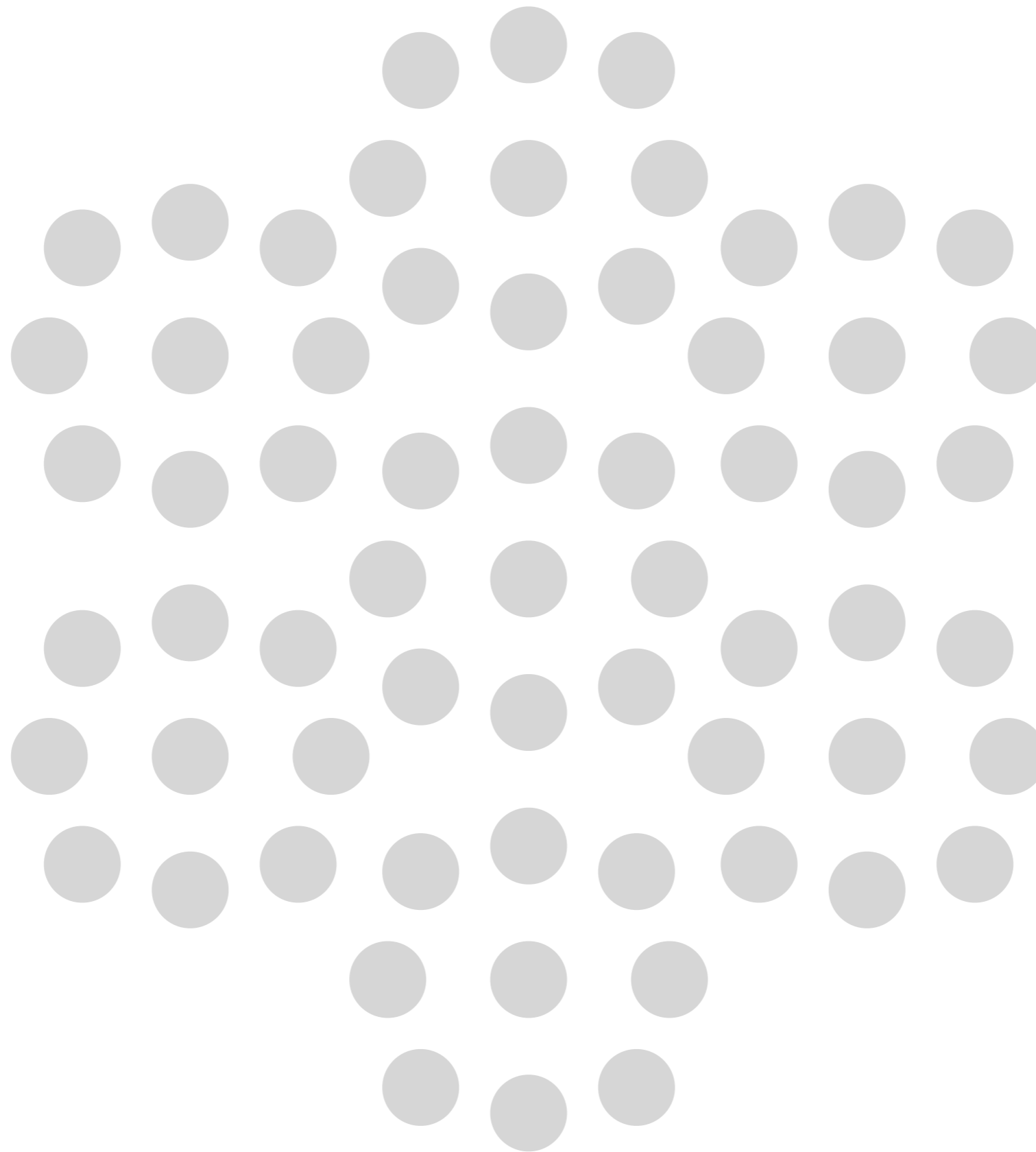
**Structure
Evolution**

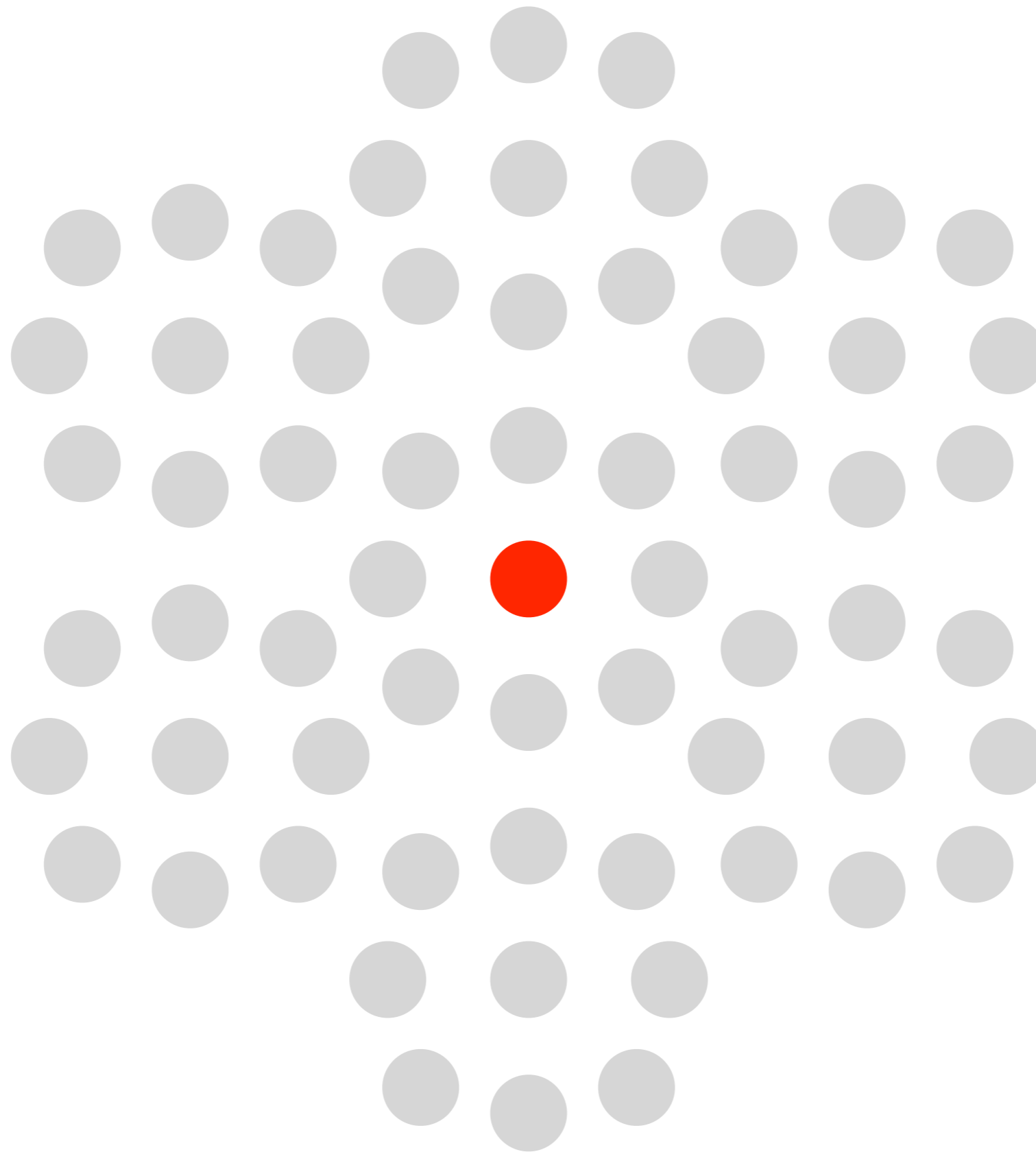
**What if we do not
know what to evolve?**

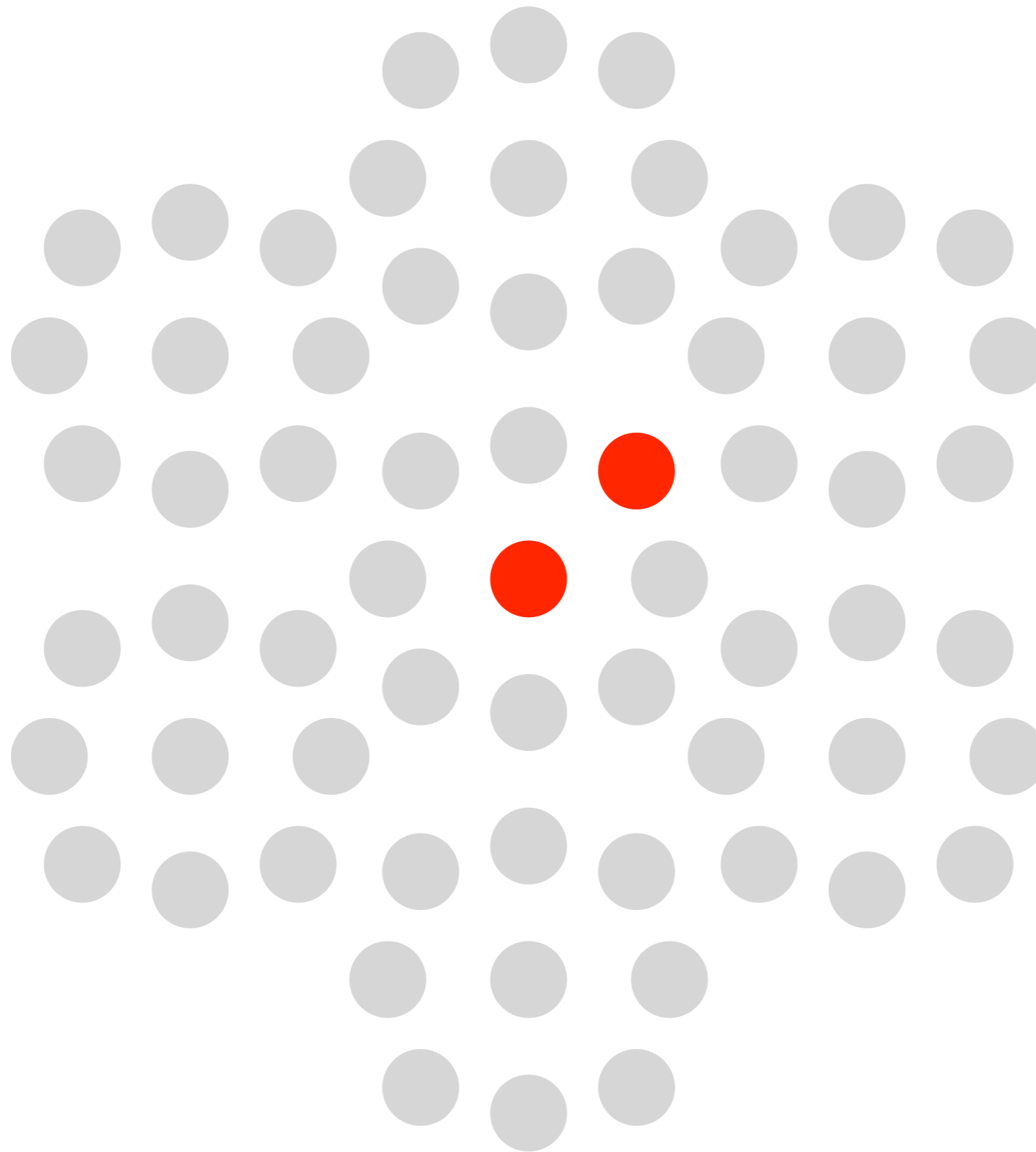


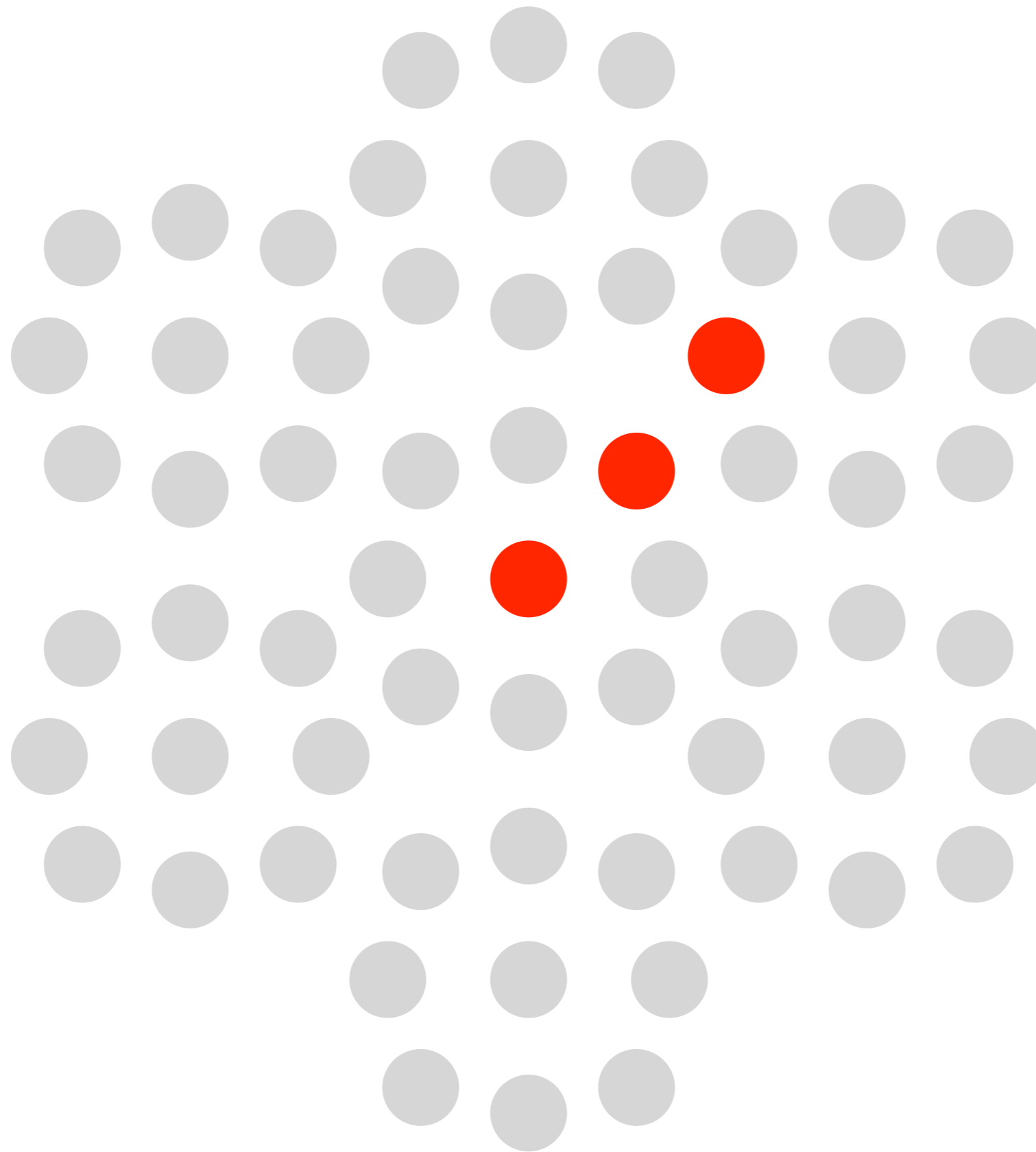


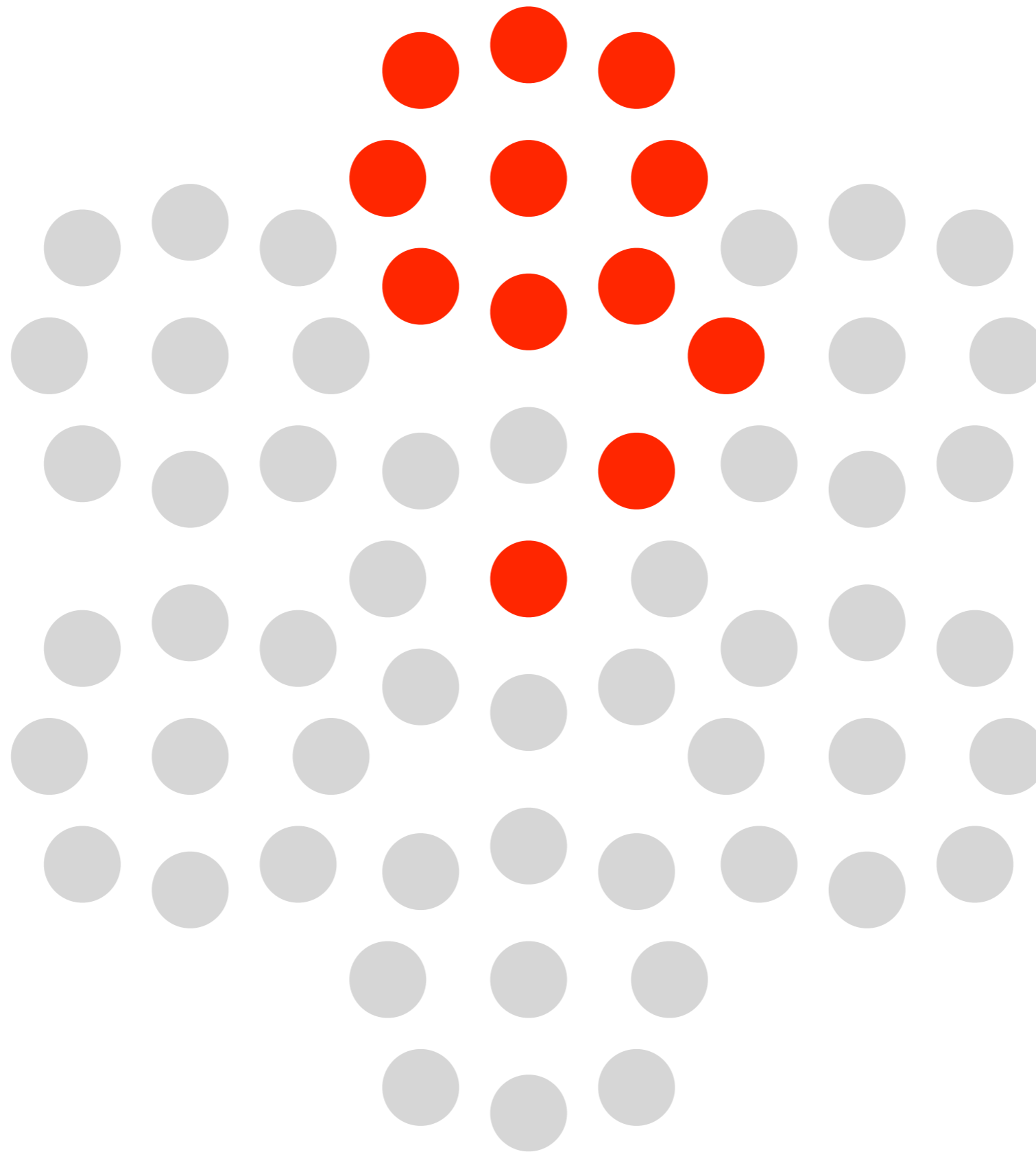
Prisma







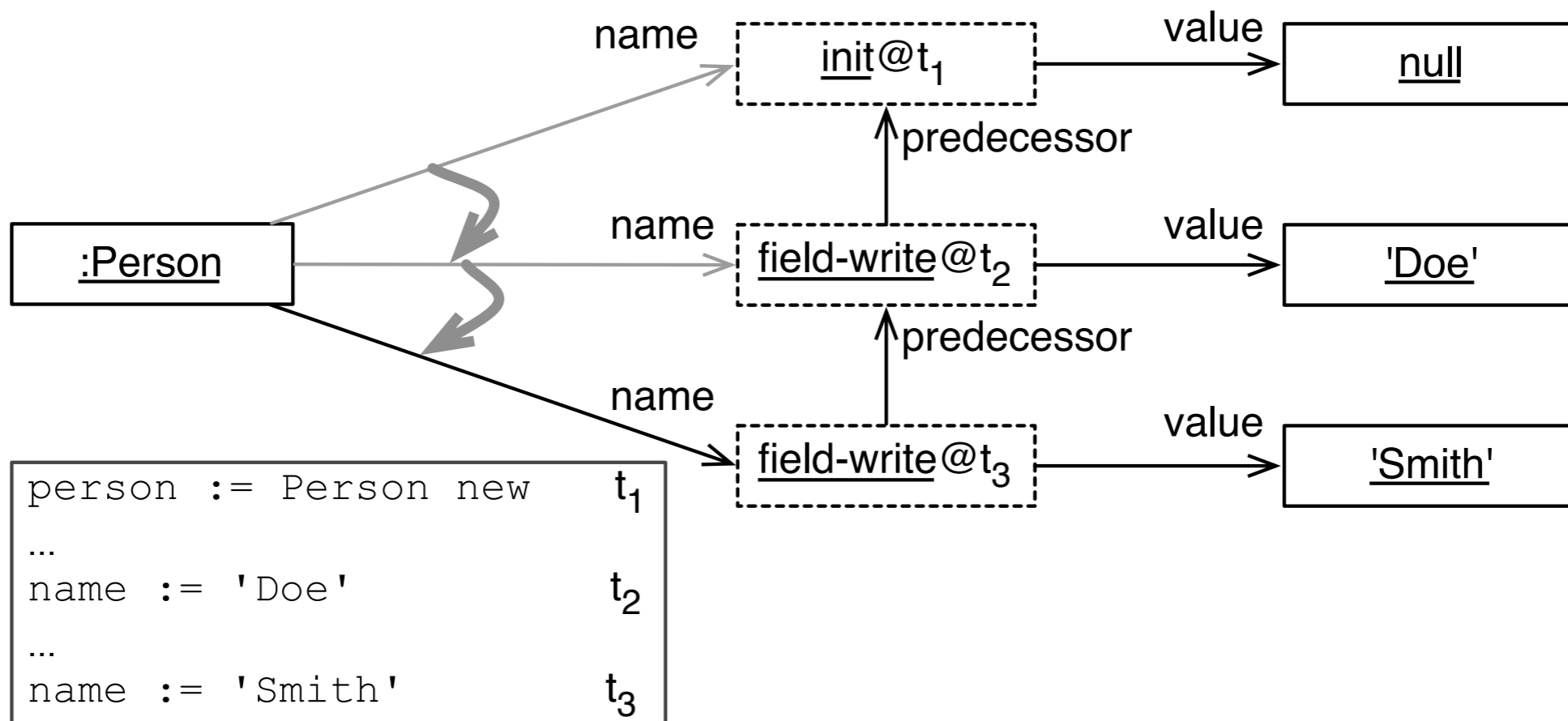




Back in time Debugger

Back in time Debugger

Object Flow Debugger
Lienhard et al. ECOOP 2008



Debugging

Profiling

**Execution
Reification**

**Structure
Evolution**

Talents

scg.unibe.ch/research/talents

Talents

IWST 2011

J. Ressia, T. Gîrba, O. Nierstrasz, F. Perin and
L. Renggli

scg.unibe.ch/research/talents

Dynamically composable units of reuse

Streams

Bifrost

scg.unibe.ch/research/bifrost

Biffrüst

Object-
Centric
Debugger

Prisma

Subjectopia

MetaSpy

Talents

Chameleon

Jenkins

[People](#)

[Build History](#)

Build Queue

No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle



Albedo		All	Bifrost	Stable	Unstable				
S	W	Job ↓	Last Success	Last Failure	Last Duration				
		Bifrost	17 hr (#209)	N/A	1 min 21 sec				
		Bifrost-Unstable	17 hr (#45)	N/A	1 min 5 sec				
		Chameleon	17 hr (#195)	N/A	21 sec				
		Development	1 day 6 hr (#34)	1 day 6 hr (#33)	2 min 53 sec				
		Development-Unstable	1 day 6 hr (#26)	1 day 6 hr (#25)	2 min 41 sec				
		ObjectDebugger	17 hr (#35)	N/A	5.8 sec				
		Pharo	1 day 22 hr (#38)	1 mo 1 day (#33)	37 sec				
		Pharo-Unstable	1 day 6 hr (#14)	23 days (#9)	6.6 sec				
		Pharogenesis	4 mo 29 days (#7)	N/A	2 min 24 sec				
		Prisma	1 day 23 hr (#51)	N/A	4.6 sec				
		Seaside3	1 day 6 hr (#5)	N/A	3 min 8 sec				
		Talents	17 hr (#25)	N/A	18 sec				
		Talents-UI	17 hr (#17)	19 hr (#13)	3 min 4 sec				
		TextLint	1 day 6 hr (#10)	23 days (#2)	3 min 50 sec				

Icon: [S](#) [M](#) [L](#)

[Legend](#)
[RSS for all](#)
[RSS for failures](#)
[RSS for just latest builds](#)

scg.unibe.ch/jenkins/



Alexandre
Bergel



Marcus
Denker



Stéphane
Ducasse



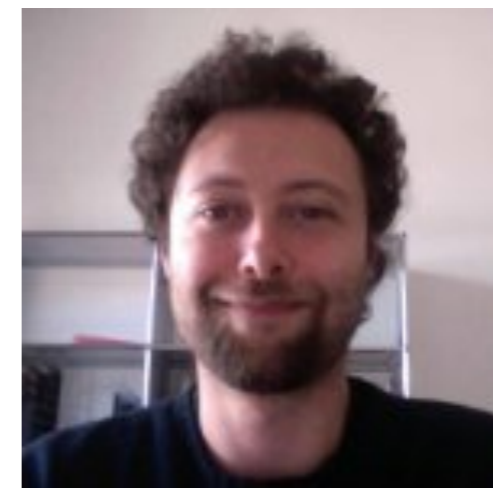
Oscar
Nierstrasz



Lukas
Renggli



Tudor
Gîrba



Fabrizio
Perin

Bifrost

scg.unibe.ch/research/bifrost