

Lessons in Software Evolution Learned by Listening to Smalltalk

Oscar Nierstrasz

scg.unibe.ch

Research Team



Adrian Lienhard



Adrian Kuhn



Fabrizio Perin



Lukas Renggli



Jorge Ressia

David Roethlisberger



Niko Schwarz



Toon Verwaest



Erwann Wernli



Roadmap



The trouble with change

Smalltalk in a Nutshell

Taming Software Change

Epilogue — Bringing Models Closer to Code

Roadmap



The trouble with change

Smalltalk in a Nutshell

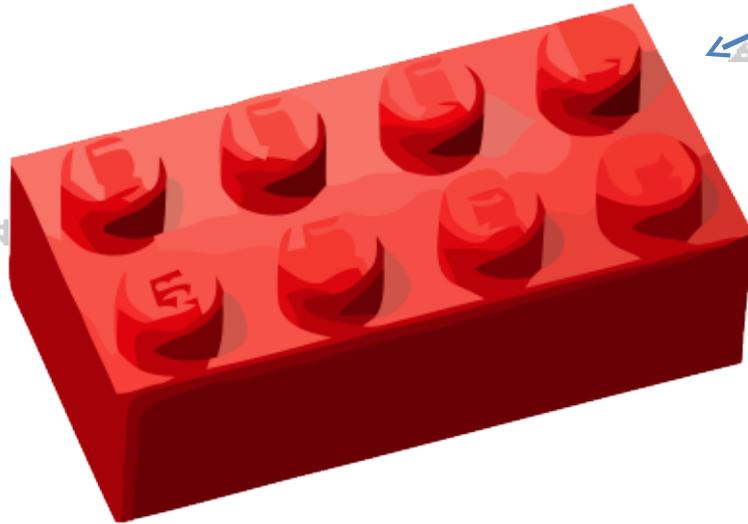
Taming Software Change

Epilogue — Bringing Models Closer to Code

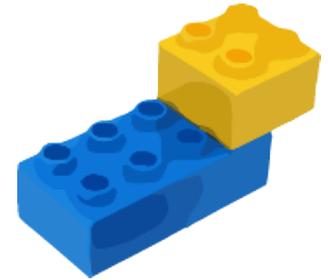
How we enable change

Package what
is stable

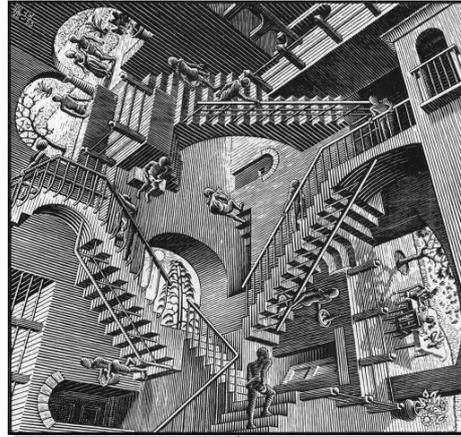
Define an *interface*
for configuration



A brief history of software packaging



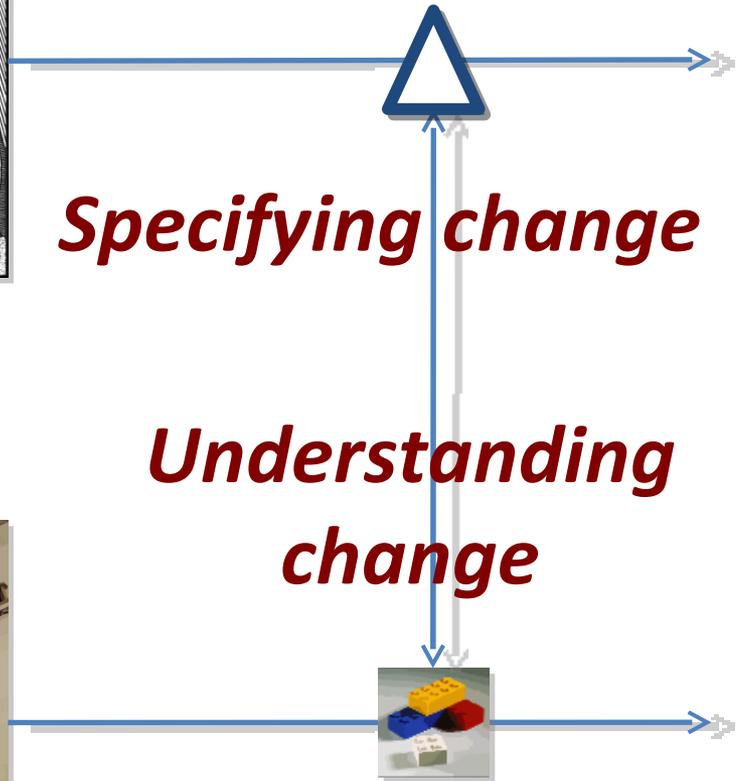
The Trouble with Software Change



Bridging the gap



Understanding the system

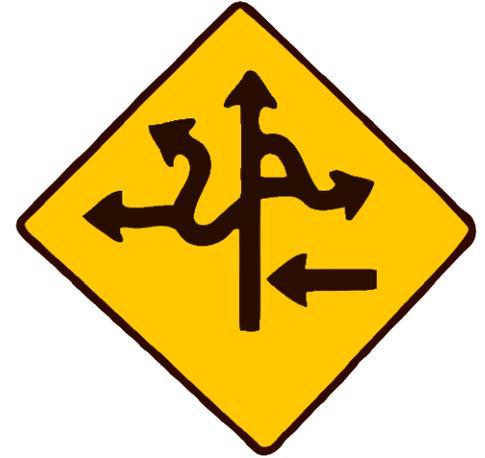


Specifying change

Understanding change

Managing change

Roadmap



The trouble with change

Smalltalk in a Nutshell

Taming Software Change

Epilogue — Bringing Models Closer to Code

To thine own self be true

Less is More

The object model “is as simple as possible, but no simpler”

Reify Everything

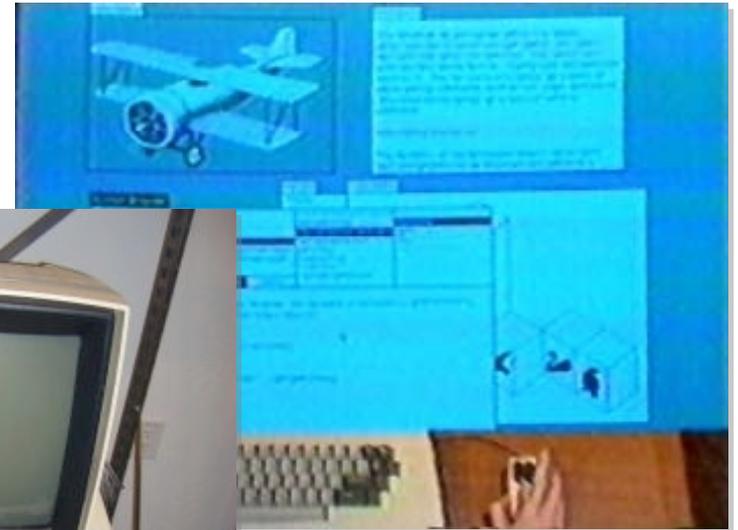
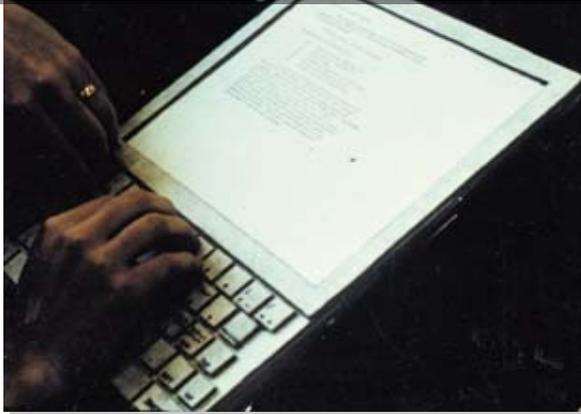
*“Everything is an object”
is applied consistently*

You can change a Running System

*Smalltalk models itself, so changing
the model changes the system*

The origins of Smalltalk

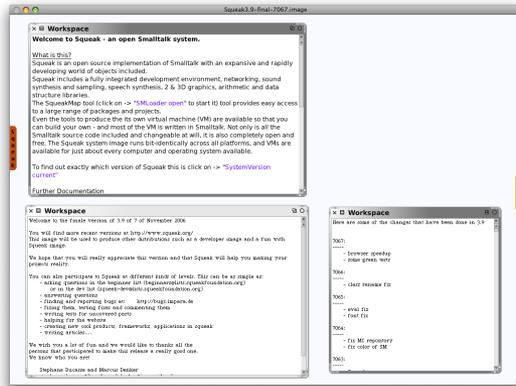
Alan Kay's Dynabook project
(1968)



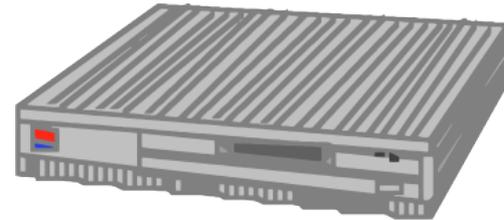
Alto — Xerox
PARC (1973)

Smalltalk architecture

Image



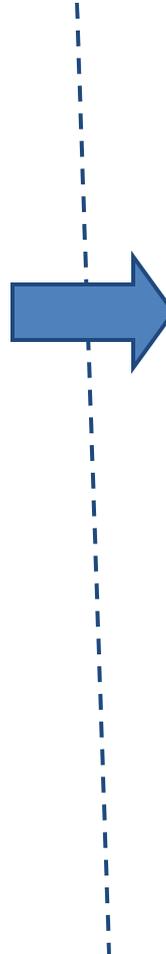
Virtual machine



Changes



Sources



Everything happens by sending messages

First unary, then binary, then keyword:

2 raisedTo: 1 + 3 factorial

128

A typical method

Method name

Argument

Comment

Return

```
<= aPoint  
  "Answer whether the receiver is neither  
  below nor to the right of aPoint."
```

```
^ x <= aPoint x and: [y <= aPoint y]
```

Block

Binary message

Keyword message

Instance variable

Unary message

```
(2@3) <= (5@6)
```

```
true
```

Everything is an object

The screenshot displays the Pharo IDE interface. At the top, the title bar reads "pharo1.0-10505-rc1dev10.01.1.image". The main workspace, titled "Shout Workspace", contains the code `BouncingAtomsMorph new openInWorld`. Below this, a green rectangular area represents the world, populated with several small blue squares and one red square. A red dot is also visible to the right of the green area. An inspector window, titled "an AtomMorph(715) on Inspector", is open over the red square in the world. It shows the following object structure:

- an AtomMorph(715)
- borderColor : Color
- borderWidth : 0
- bounds : 359@418 co
- color : Color red
- extension : a MorphE
- fullBounds : 359@418
- owner : a Bouncing
- submorphs : an Array
- velocity : 0@0

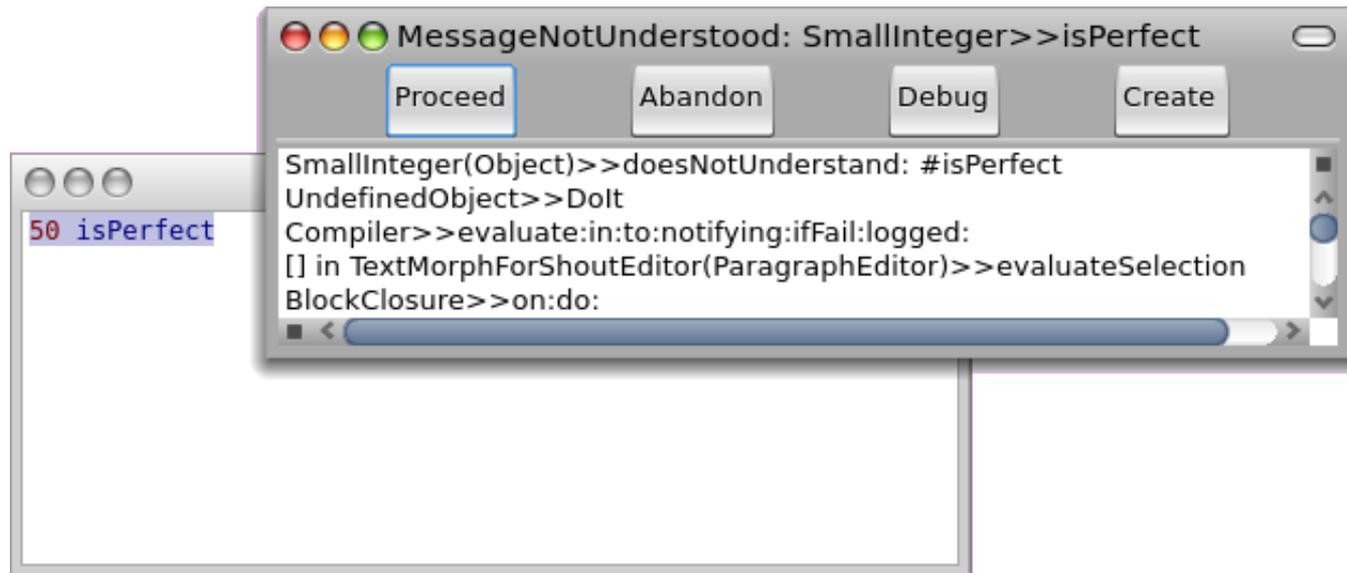
The inspector also shows "self velocity: 2@3". To the right, another inspector window, titled "a BouncingAtomsMorph", shows the root object's structure:

- root: a BouncingAtomsMorph(2742)
- bounds: 79@275 corner: 479@525
- owner: a PasteUpMorph(1622) [wo
- submorphs: an Array(an AtomMorph(7
- 1: an AtomMorph(78)
- 2: an AtomMorph(3007)
- 3: an AtomMorph(943)
- 4: an AtomMorph(780)
- 5: an AtomMorph(2213)
- 6: an AtomMorph(2506)
- 7: an AtomMorph(2443)
- 8: an AtomMorph(3195)
- 9: an AtomMorph(2923)
- 10: an AtomMorph(536)
- 11: an AtomMorph(2632)
- 12: an AtomMorph(305)
- 13: an AtomMorph(358)
- 14: an AtomMorph(2807)
- 15: an AtomMorph(1204)
- 16: an AtomMorph(3044)
- 17: an AtomMorph(1101)

The bottom of the IDE shows a tab bar with three tabs: "Shout Workspace", "a BouncingAtomsMorph(2742)", and "an AtomMorph(715) on Inspector".



You *can* change a running system



AKA: "The Debugger is your Friend"



Pharo — an multi-platform, open-source Smalltalk

The screenshot shows a web browser window with the address bar containing 'http://www.pharo-project.org/'. The page features a large 'Pharo' logo with a lighthouse icon inside the letter 'o'. The main heading reads 'Clean, innovative, open-source Smalltalk environment'. The page is organized into several sections: 'Home' with links to News, About, Success stories, Screenshots, and Board; 'Download' with a link to Media files; 'Community' with a link to Issue Tracking; 'License'; 'Documentation' with links to FAQ, Getting started, and Tutorials/Books; and 'Pharo by Example'. A 'Mission' section describes Pharo's goal as a clean, innovative, free open-source Smalltalk environment. A 'Quick links' section includes 'Getting started' and 'Report a bug'. A 'New Pharo book available' section mentions 'Pharo by Example' by Andrew Black et al. A search bar is located at the bottom left.

Pharo Open Source Smalltalk - Home

http://www.pharo-project.org/

Google

Pharo

Clean, innovative, open-source Smalltalk environment

Home

[News](#)
[About](#)
[Success stories](#)
[Screenshots](#)
[Board](#)

Download

[Media files](#)

Community

[Issue Tracking](#)

License

Documentation

[FAQ](#)
[Getting started](#)
[Tutorials/Books](#)

Pharo by Example

Search

Mission

Pharo's goal is to deliver a clean, innovative, free open-source Smalltalk environment. By providing a stable and small core system, excellent developer tools, and maintained releases, Pharo is an attractive platform to build and deploy mission critical Smalltalk applications. Pharo is MIT licensed and is steered by a board of benevolent dictators. The board makes final decisions if no consensus can be reached within the community. Pharo fosters a healthy ecosystem of both private and commercial contributors who advance and maintain the core system and its external packages.

Pharo provides...

- a pure object-oriented language
- a stable core system with unit tests
- a clean look and feel
- full block closures
- low memory footprint

Quick links

[Getting started](#)
[Report a bug](#)

New Pharo book available

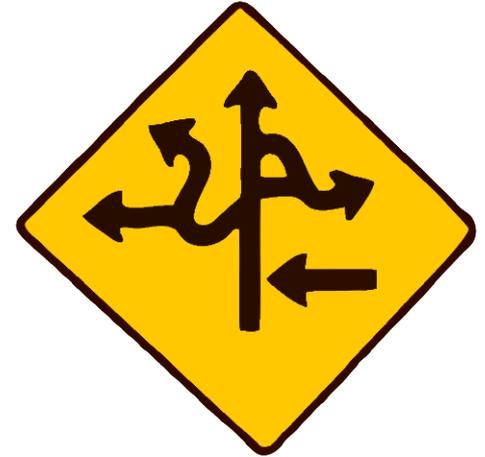
[Pharo by Example](#) has been released and can be downloaded for free.



and developers, will guide you gently through the

Pharo by Example. Andrew Black, et al.
Square Bracket Associates, 2009.

Roadmap



The trouble with change

Smalltalk in a Nutshell

Taming Software Change

Epilogue — Bringing Models Closer to Code

Moose: a platform for reverse engineering

The screenshot displays the Moose Finder application window, titled "Moose Finder - igeEnt86-2009-05-25 (MooseModel)". The interface is divided into two main panes. The left pane, titled "igeEnt86-2009-05-25 - MooseModel", contains a list of model classes and their counts. The right pane, titled "ClassGroup - 1814 items", displays a class hierarchy diagram with tabs for "Properties", "Complexity", and "Evaluator".

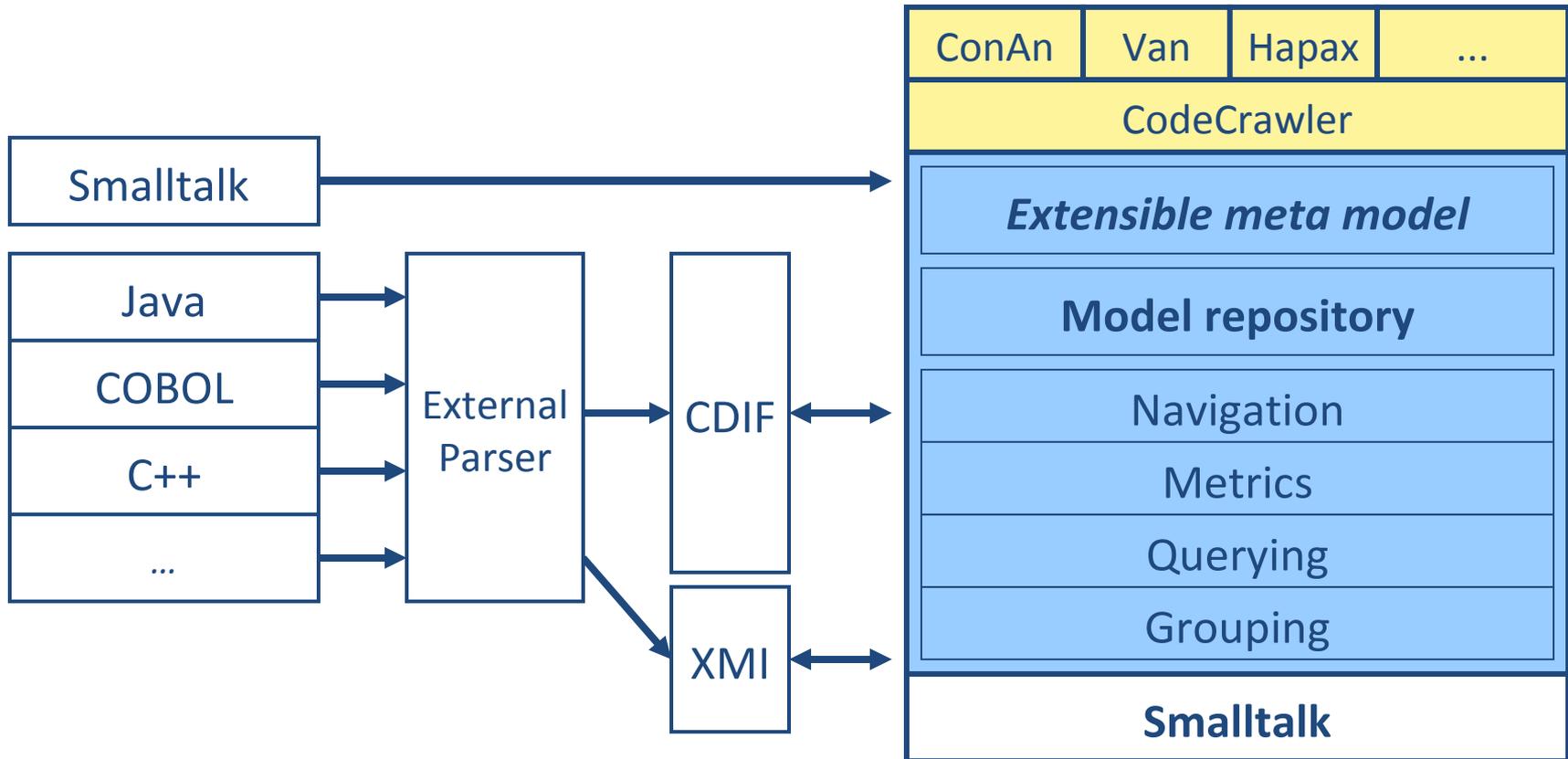
Left Pane: Model Classes

- All famixaccess (32789 FAMIXAccesses)
- All famixannotationinstance (3351 FAMIXAnnotationInstances)
- All famixannotationtype (11 FAMIXAnnotationTypes)
- All famixattribute (7036 FAMIXAttributes)
- All famixcaughtexception (2279 FAMIXCaughtExceptions)
- All famixclass (2447 FAMIXClasses)
- All famixdeclaredexception (5209 FAMIXDeclaredExceptions)
- All famixfunction (2 FAMIXFunctions)
- All famixinheritance (3338 FAMIXInheritances)
- All famixinvocation (35864 FAMIXInvocations)
- All famixlocalvariable (14303 FAMIXLocalVariables)
- All famixmethod (13827 FAMIXMethods)
- All famixnamespace (307 FAMIXNamespaces)
- All famixparameter (11958 FAMIXParameters)
- All famixprimitivetype (9 FAMIXPrimitiveTypes)
- All famixsessionbean (39 FAMIXSessionBeans)
- All famixthrownexception (869 FAMIXThrownExceptions)
- All model classes (1814 FAMIXClasses)**
- All model namespaces (238 FAMIXNamespaces)
- Group (515 FAMIXMethods)

Right Pane: ClassGroup - 1814 items

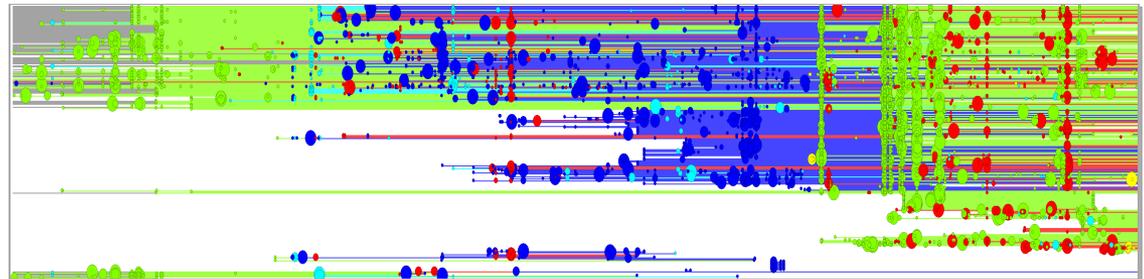
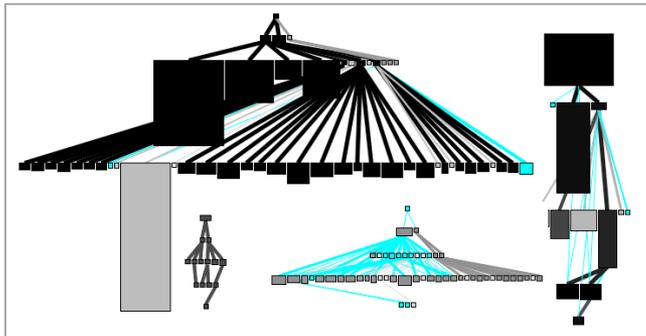
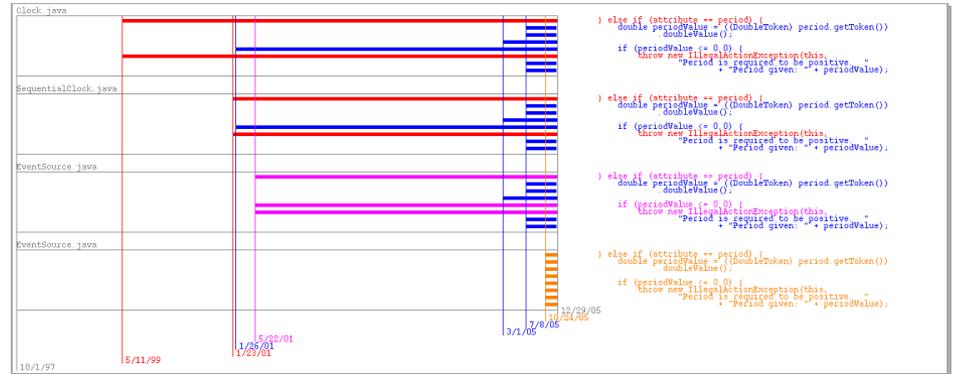
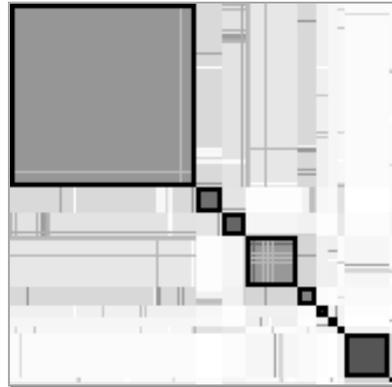
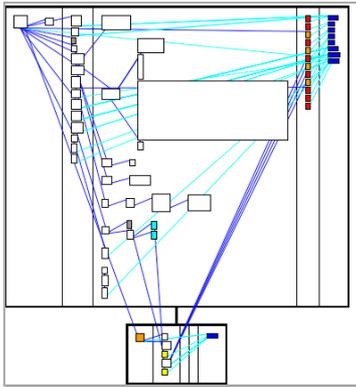
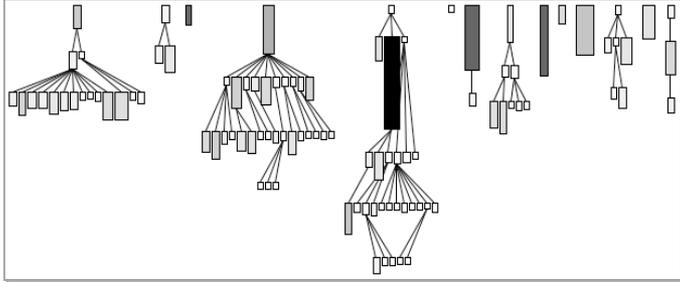
The right pane shows a class hierarchy diagram. The diagram consists of a root node at the top, which branches into several child nodes. These child nodes further branch into more nodes, creating a complex tree structure. The nodes are represented by small squares, and the connections between them are lines. The diagram is currently displayed in the "Complexity" tab.

Explicit metamodels enable change

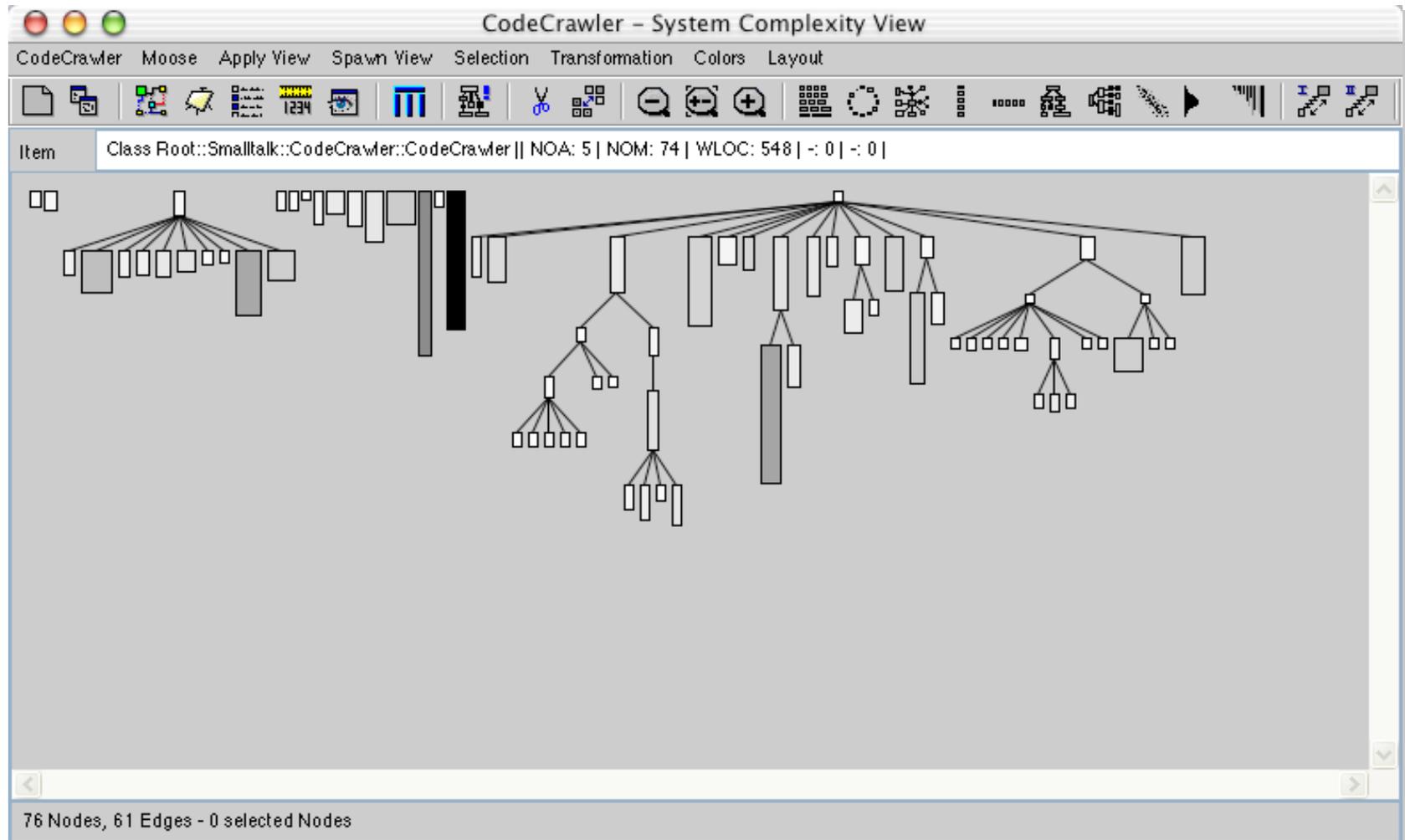


The Story of Moose: an Agile Reengineering Environment. Oscar Nierstrasz, Stéphane Ducasse, and Tudor Gîrba. ESEC/FSE 2005

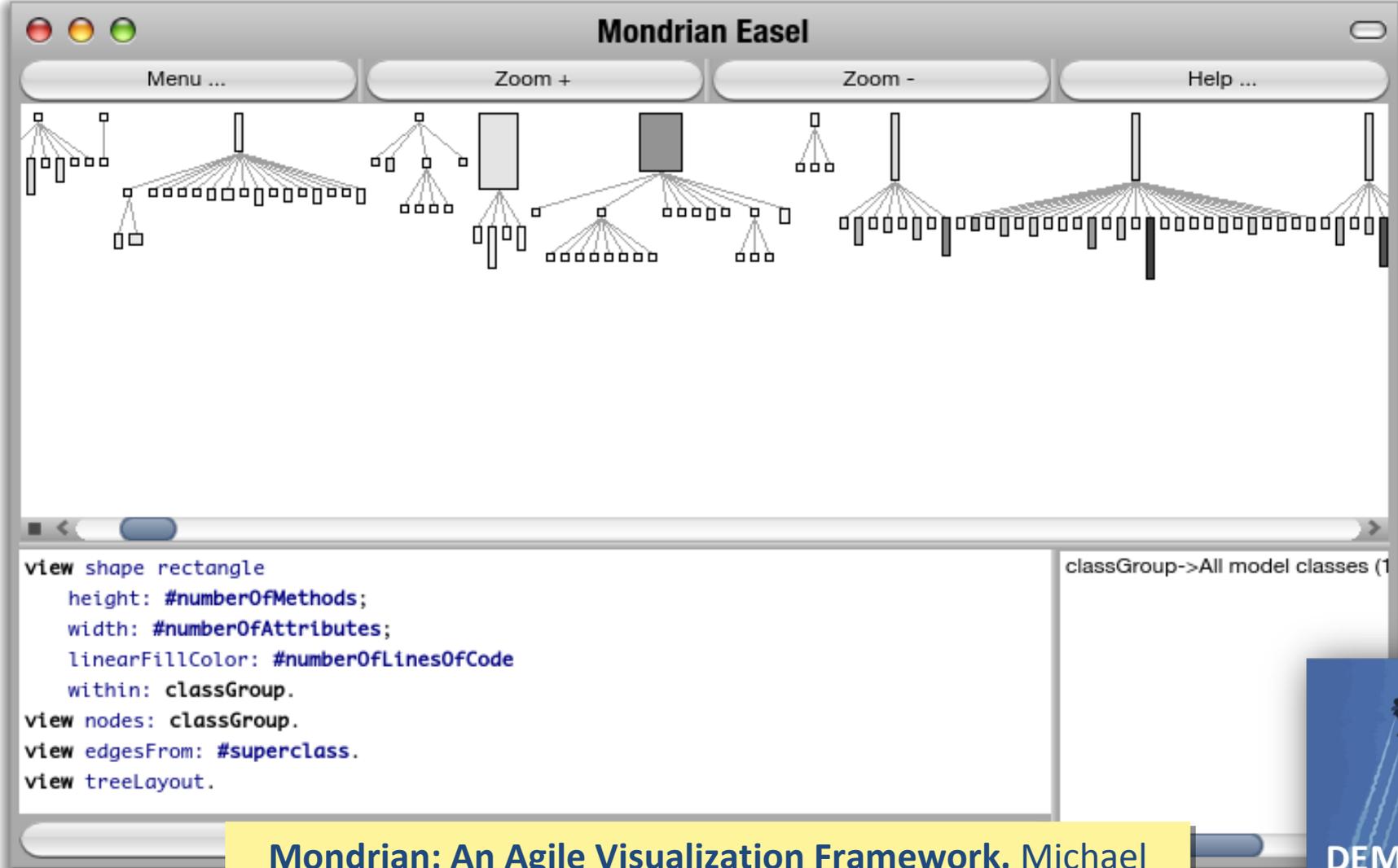
Moose visualizations



Programming visualizations with CodeCrawler



Scripting visualizations with Mondrian



The screenshot shows the Mondrian Easel application window. The title bar reads "Mondrian Easel". Below the title bar are four buttons: "Menu ...", "Zoom +", "Zoom -", and "Help ...". The main area displays a class hierarchy visualization with nodes and edges. The nodes are represented by rectangles of varying sizes and colors (white, light gray, dark gray). The edges are represented by lines connecting the nodes. The visualization is a tree structure with a root node at the top left and several child nodes branching out. The nodes are arranged in a way that shows the relationships between classes, including inheritance and associations. The script editor at the bottom left contains the following code:

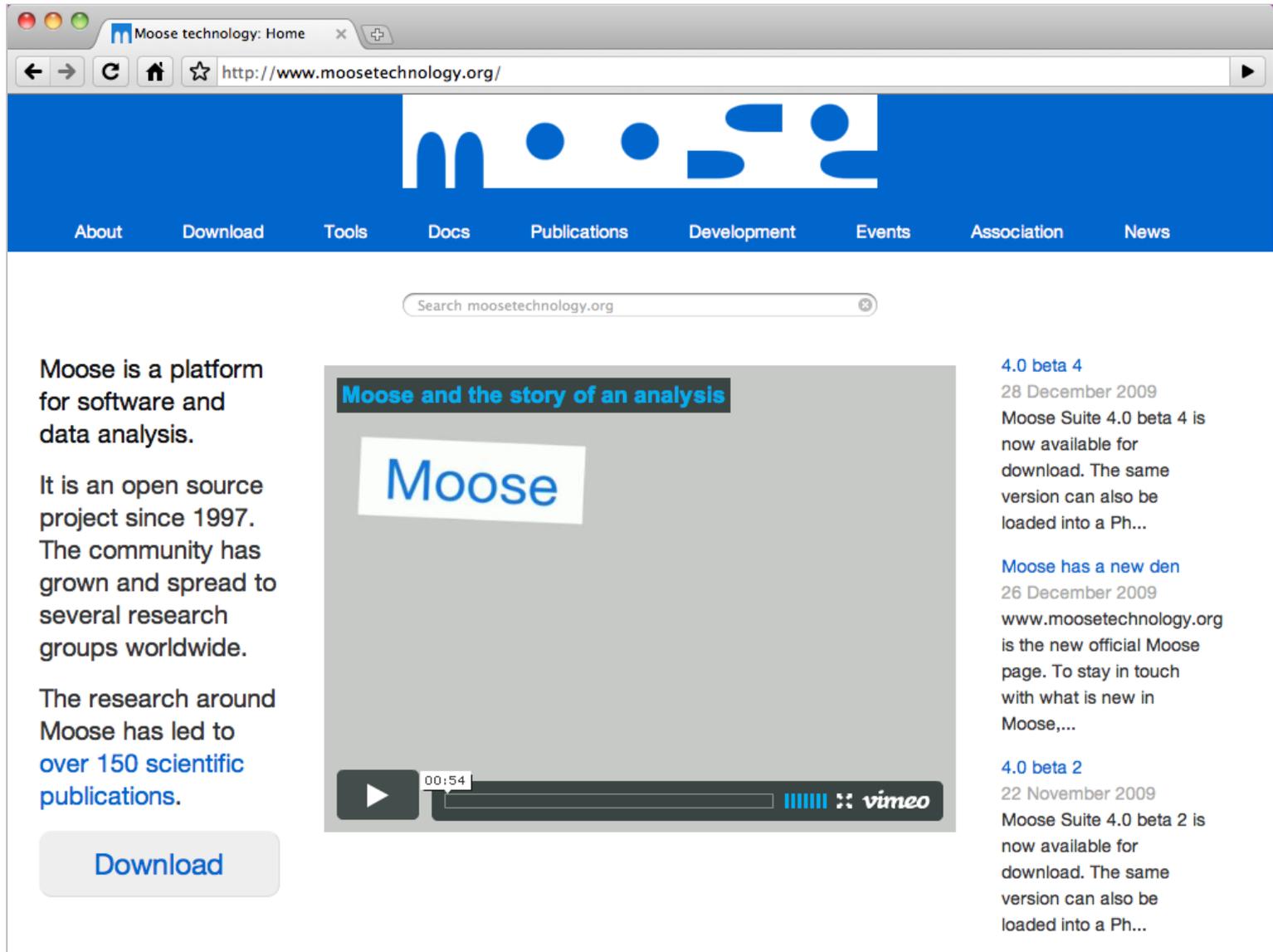
```
view shape rectangle
  height: #numberOfMethods;
  width: #numberOfAttributes;
  linearFillColor: #numberOfLinesOfCode
  within: classGroup.
view nodes: classGroup.
view edgesFrom: #superclass.
view treeLayout.
```

On the right side of the script editor, there is a small preview window showing a class group: "classGroup->All model classes (1".

Mondrian: An Agile Visualization Framework. Michael Meyer, Tudor Gîrba, and Mircea Lungu. SoftVis 2006



Moose — a platform for collaborative research



The screenshot shows a web browser window with the URL <http://www.moosetechnology.org/>. The page features a blue header with the Moose logo and a navigation menu with links for About, Download, Tools, Docs, Publications, Development, Events, Association, and News. Below the header is a search bar and a main content area. On the left, there is a text block describing Moose as a platform for software and data analysis, an open source project since 1997, and a platform that has led to over 150 scientific publications. A 'Download' button is located below this text. In the center, there is a video player with the title 'Moose and the story of an analysis' and a play button. On the right, there are two news items: '4.0 beta 4' dated 28 December 2009 and 'Moose has a new den' dated 26 December 2009. The video player shows a play button, a progress bar at 00:54, and the Vimeo logo.

Moose technology: Home

http://www.moosetechnology.org/

About Download Tools Docs Publications Development Events Association News

Search moosetechnology.org

Moose is a platform for software and data analysis.

It is an open source project since 1997. The community has grown and spread to several research groups worldwide.

The research around Moose has led to [over 150 scientific publications](#).

[Download](#)

Moose and the story of an analysis

Moose

00:54

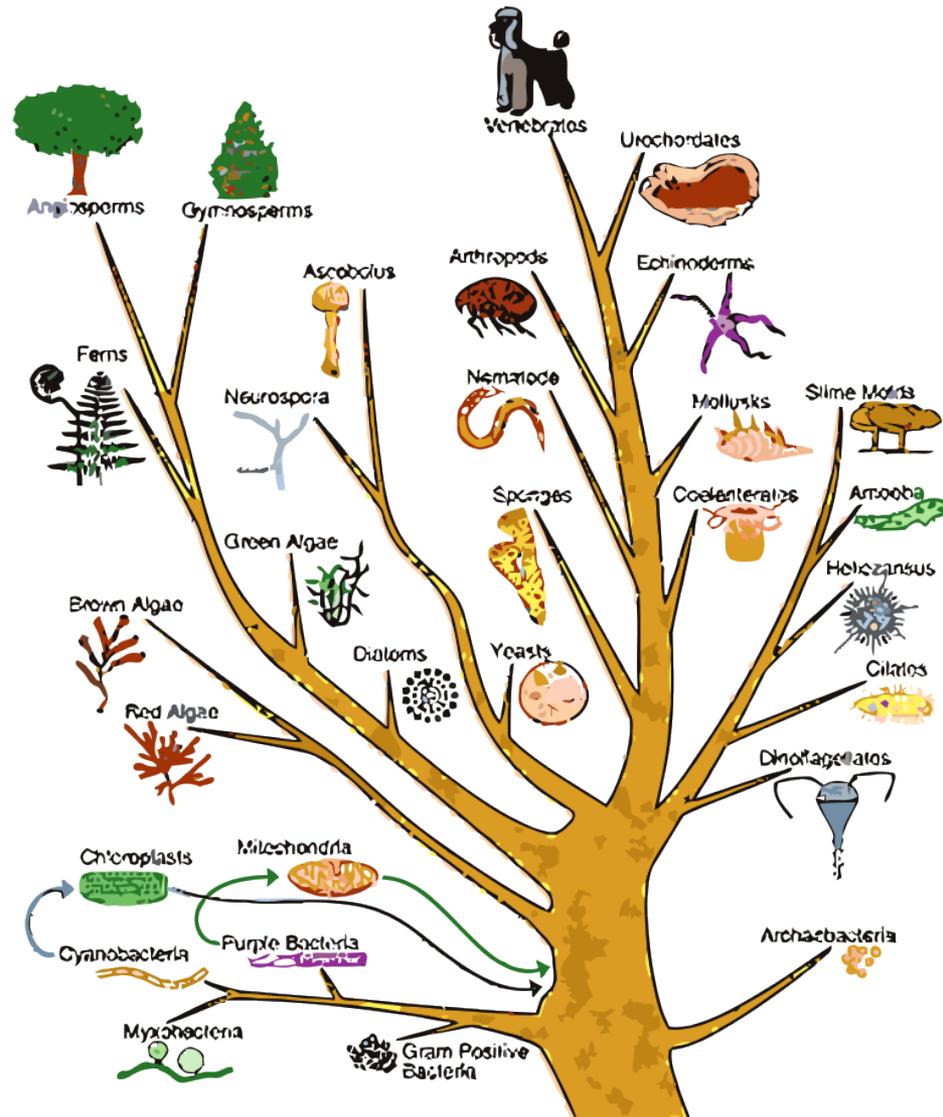
vimeo

4.0 beta 4
28 December 2009
Moose Suite 4.0 beta 4 is now available for download. The same version can also be loaded into a Ph...

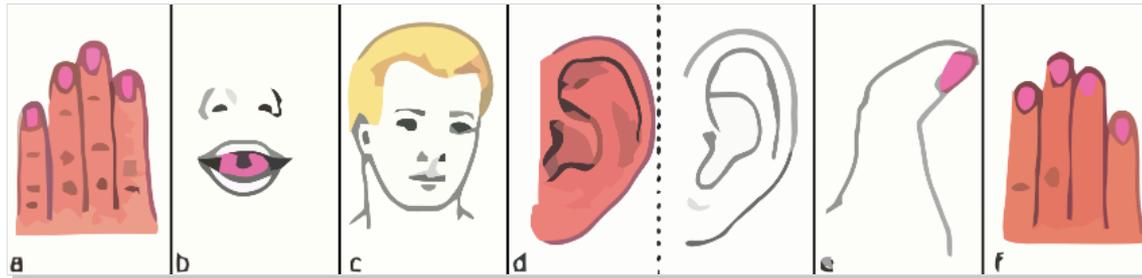
Moose has a new den
26 December 2009
www.moosetechnology.org is the new official Moose page. To stay in touch with what is new in Moose,...

4.0 beta 2
22 November 2009
Moose Suite 4.0 beta 2 is now available for download. The same version can also be loaded into a Ph...

Feature inheritance ≠ specialization

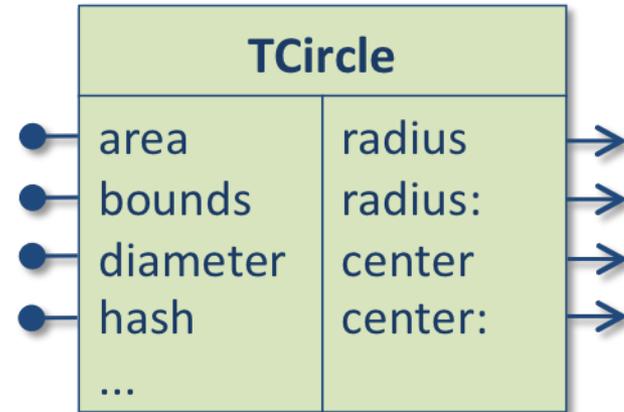
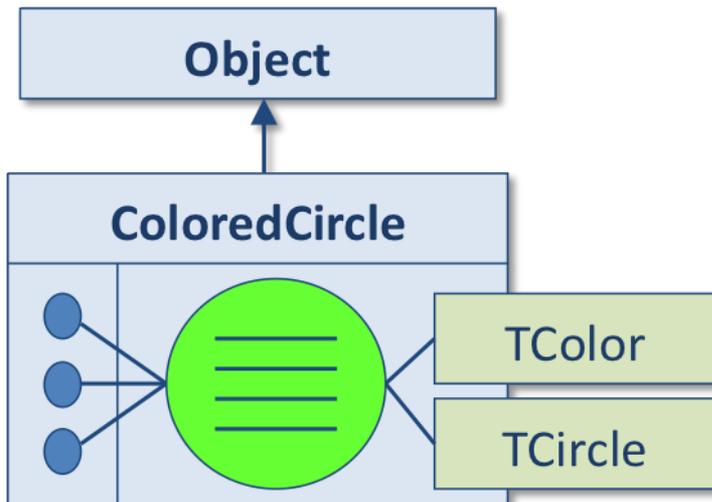


First-class traits enable fine-grained reuse



Class = superclass + state + traits + glue

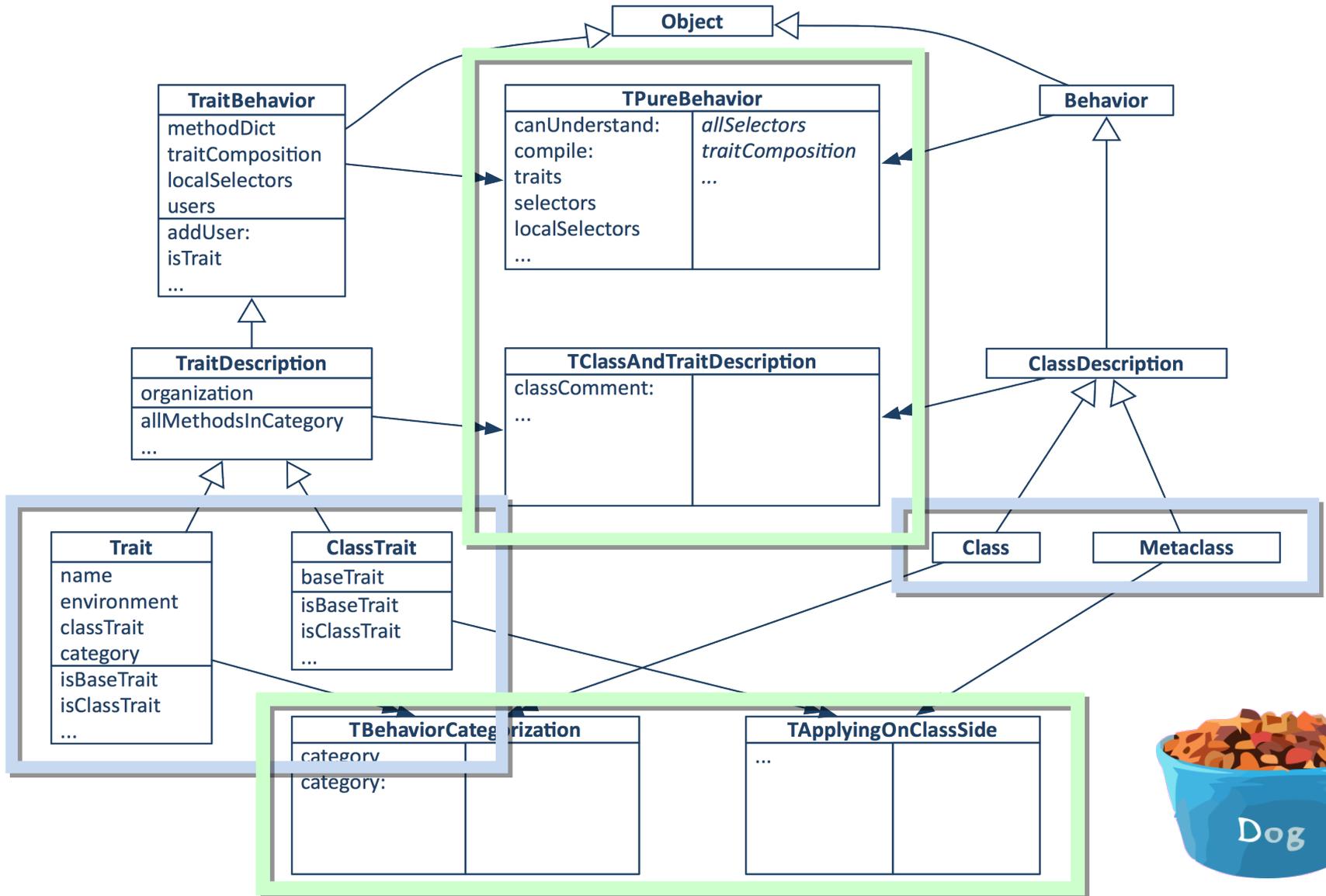
Traits provide and require methods



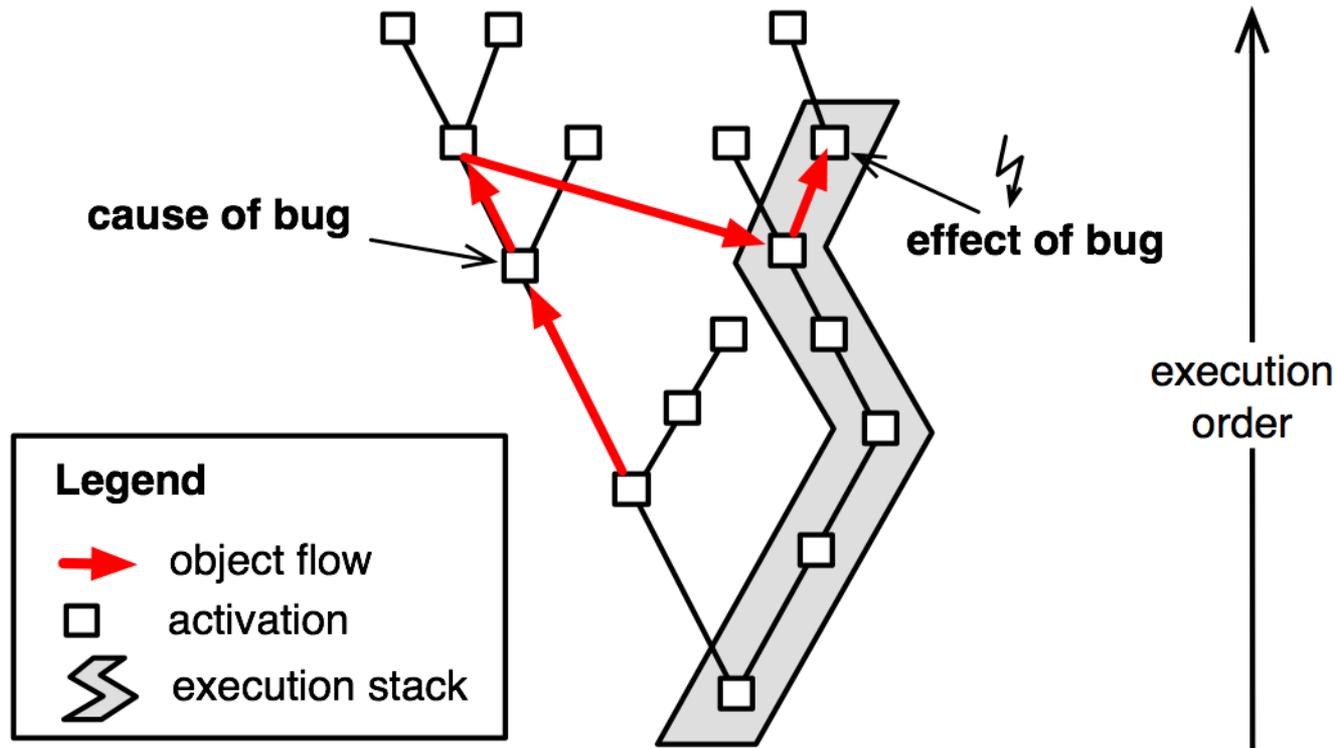
The composing class retains control

Traits: A Mechanism for fine-grained Reuse. Stéphane Ducasse, Oscar Nierstrasz, Nathanael Schärli, Roel Wuyts and Andrew Black. ACM TOPLAS, March 2006

Traits and Classes share common behaviour

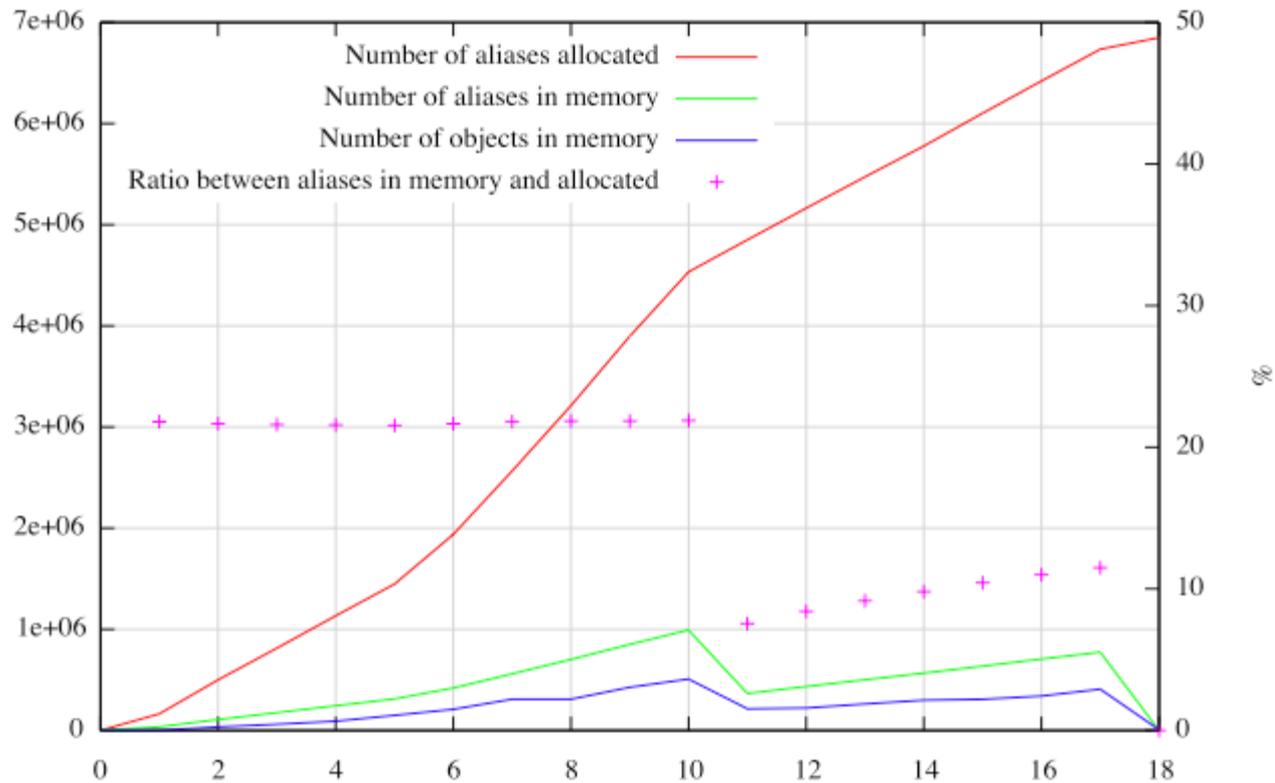


First-class aliases enable object flow analysis

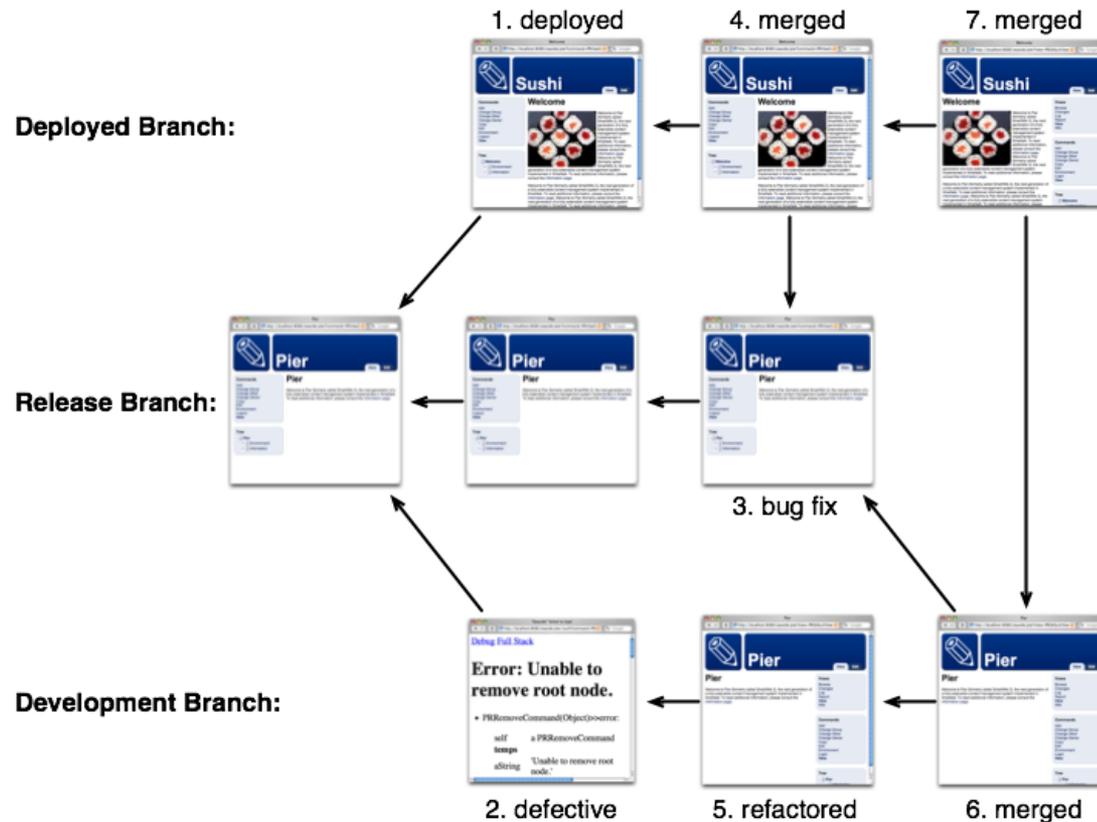


Practical Object-Oriented Back-in-Time Debugging. Adrian Lienhard, Tudor Gîrba, and Oscar Nierstrasz. ECOOP 2008.

GC remembers only what is needed

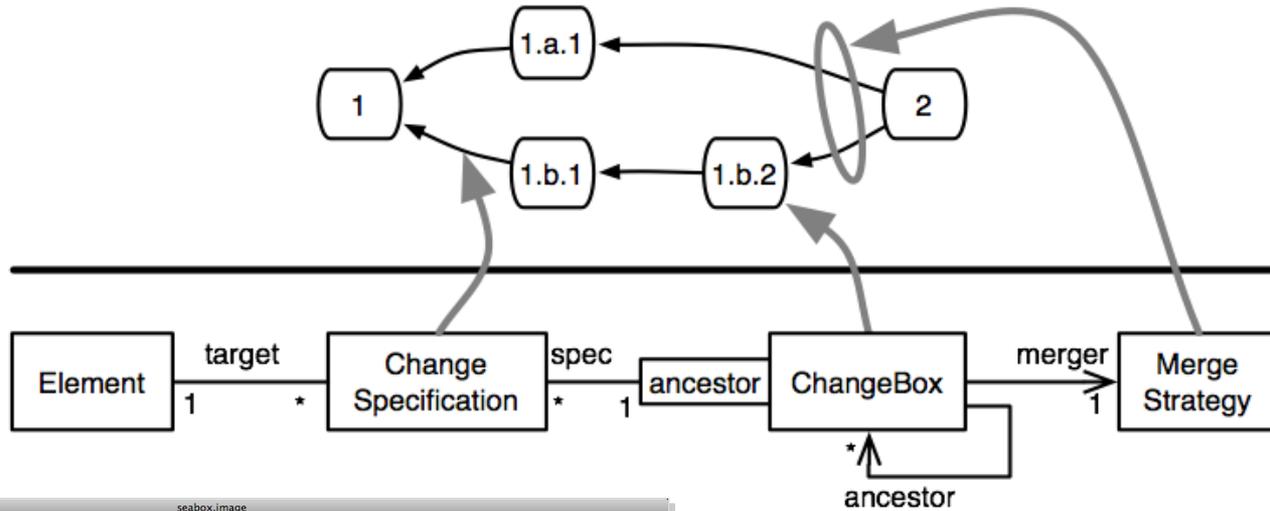


How to manage change in a running system?



Encapsulating and Exploiting Change with
Changeboxes. Marcus Denker, et al. ICDL 2007.

First-class changes enable scoped evolution



Workspace

```
WAKom stop; startOn: 8080.
WARRegistry clearAllHandlers.
http://localhost:8080/seaside/sushi-
http://localhost:8080/seaside/sushi-
```

Work Session Browser

```
sushi-devel CBX9933048(WASStoreItem)>>
sushi-prod CBX929486507(WASStoreItem)>
           CBX706896473(WASStoreItem)>
```

System Browser: WASStoreItem (in st)

```
Seaside-Examples-Store-M
Seaside-Tests-Previous
Seaside-Tests-Canvas
Seaside-HTTP
Seaside-Libraries
Seaside-Platform
Seaside-Rendering
Seaside-RenderLoop
Seaside-RequestHandler
Seaside-Session
WASStoreAddr
WASStoreCart
WASStoreCreditCard
WASStoreMasterCard
WASStoreVisaCard
WASStoreInventory
WASStoreItem
```

Seaside Sushi Store: Fill your cart

Search:

Browse

- Aji Horse Mackerel
- Akagai Ark Shell Clam
- Akami Maguro Red Tuna
- Anago Conger Eel
- Awabi Abalone
- California Roll
- Chakinzushi Omelet-wrapped Sushi
- Chutoro Maguro Fatty Tuna

Search:

- Aji Horse Mackerel Price: \$2.50
- Akagai Ark Shell Clam Price: \$3.00
- Akami Maguro Red Tuna Price: \$1.50
- Anago Conger Eel Price: \$3.00
- Awabi Abalone Price: \$3.00
- California Roll Price: \$2.50
- Chakinzushi Omelet-wrapped Sushi Price: \$3.00
- Chutoro Maguro Fatty Tuna Price: \$2.75

DEMO

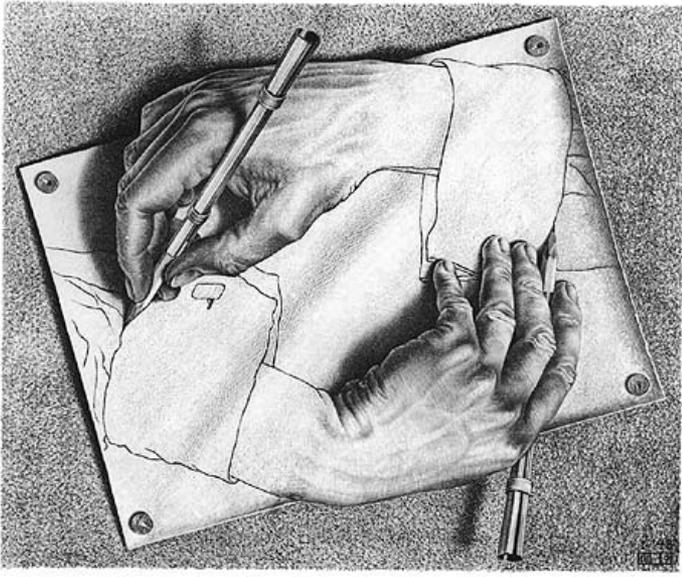
Roadmap



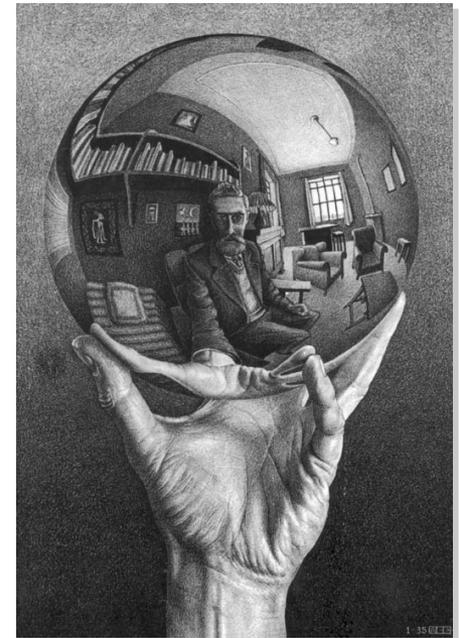
The trouble with change
Smalltalk in a Nutshell
Taming Software Change

Epilogue — Bringing Models Closer to Code

Systems that support change need to be *model-centric* and *context-aware*



Model-centric systems are self-describing and can be adapted dynamically



Context-aware systems control the scope of change to static or dynamic contexts