UNIVERSITÄT BERN

Dynamic Object-Oriented Programming with Smalltalk

1. Introduction

Prof. O. Nierstrasz

Autumn Semester 2009

What is surprising about Smalltalk

- Everything is an object
- Everything happens by sending messages
- All the source code is there all the time
- You can't lose code
- You can change everything
- You can change things without restarting the system
- The Debugger is your Friend

Why Smalltalk?

- *Pure* object-oriented language and environment
  - "Everything is an object"
- Origin of *many innovations* in OO development
  - RDD, IDE, MVC, XUnit …
- Improves on many of its successors

  - Fully interactive and dynamic

What is Smalltalk?

- ***Pure OO language***
  - Single inheritance
  - Dynamically typed
- ***Language and environment***
  - Guiding principle: *"Everything is an Object"*
  - Class browser, debugger, inspector, …
  - Mature class library and tools
- ***Virtual machine***
  - Objects exist in a persistent *image* [+ *changes*]
  - Incremental compilation

Smalltalk vs. C++ vs. Java

	Smalltalk	***C++***	***Java***
Object model	Pure	Hybrid	Hybrid
Garbage collection	Automatic	Manual	Automatic
Inheritance	Single	Multiple	Single
Types	Dynamic	Static	Static
Reflection	Fully reflective	Introspection	Introspection
Concurrency	Semaphores, Monitors	Some libraries	Monitors
Modules	Categories, namespaces	Namespaces	Packages

Smalltalk: a State of Mind

- ***Small and uniform language***
  - Syntax fits on one sheet of paper
- ***Large library of reusable classes***
  - Basic Data Structures, GUI classes, Database Access, Internet, Graphics
- ***Advanced development tools***
  - Browsers, GUI Builders, Inspectors, Change Management Tools, Crash Recovery Tools, Project Management Tools
- ***Interactive virtual machine technology***
  - Truly platform-independent
- ***Team Working Environment***
  - Releasing, versioning, deploying

Origins of Smalltalk

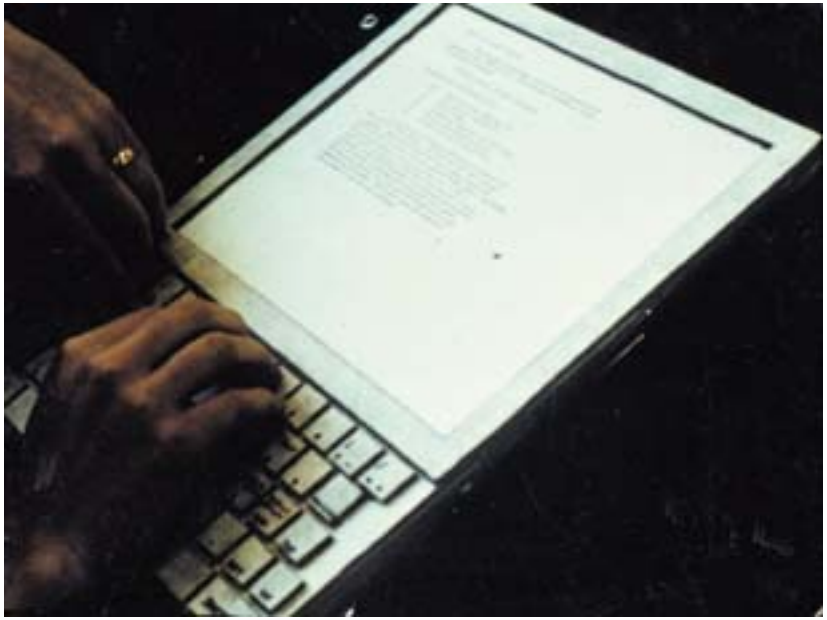
- **Project at Xerox PARC in 1970s**
  - Language and environment for new generation of graphical workstations (target: “Dynabook”)
- **In Smalltalk-72, every object was an independent entity**
  - Language was designed for children (!)
  - Evolved towards a meta-reflective architecture
- **Smalltalk-80 is the standard**

Smalltalk — The Inspiration

- **Flex** (Alan Kay, 1969)
- **Lisp** (Interpreter, Blocks, Garbage Collection)
- Turtle graphics (The **Logo** Project, Programming for Children)
- Direct Manipulation Interfaces (**Sketchpad**, Alan Sutherland, 1960)
- **NLS**, (Doug Engelbart, 1968), “the augmentation of human intellect”
- **Simula** (Classes and Message Sending)
- Xerox PARC (Palo Alto Research Center)
- **DynaBook**: a Laptop Computer for Children

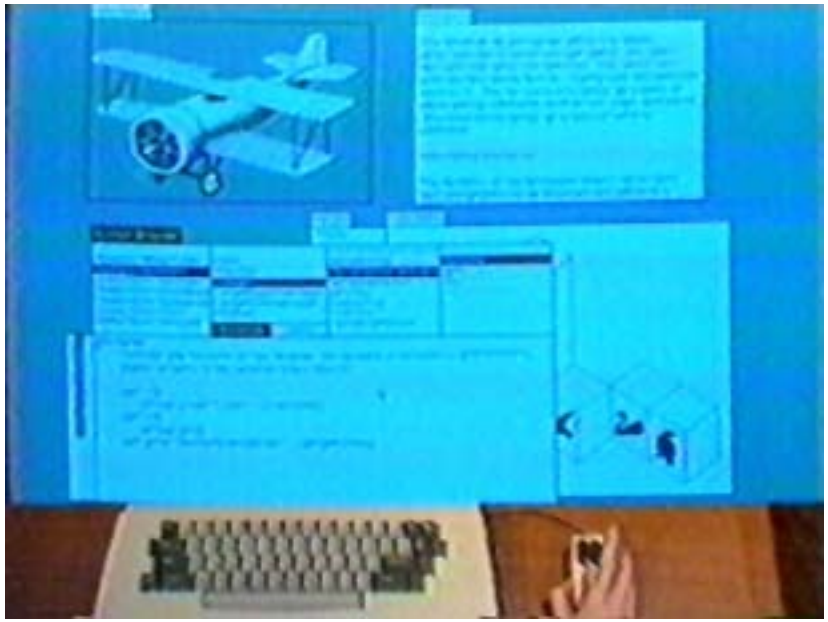
— www.smalltalk.org/smalltalk/TheEarlyHistoryOfSmalltalk_Abstract.html

Dynabook Mockup



www.artmuseum.net/w2vr/archives/Kay/01_Dynabook.html

Alto: a Machine to Run Smalltalk

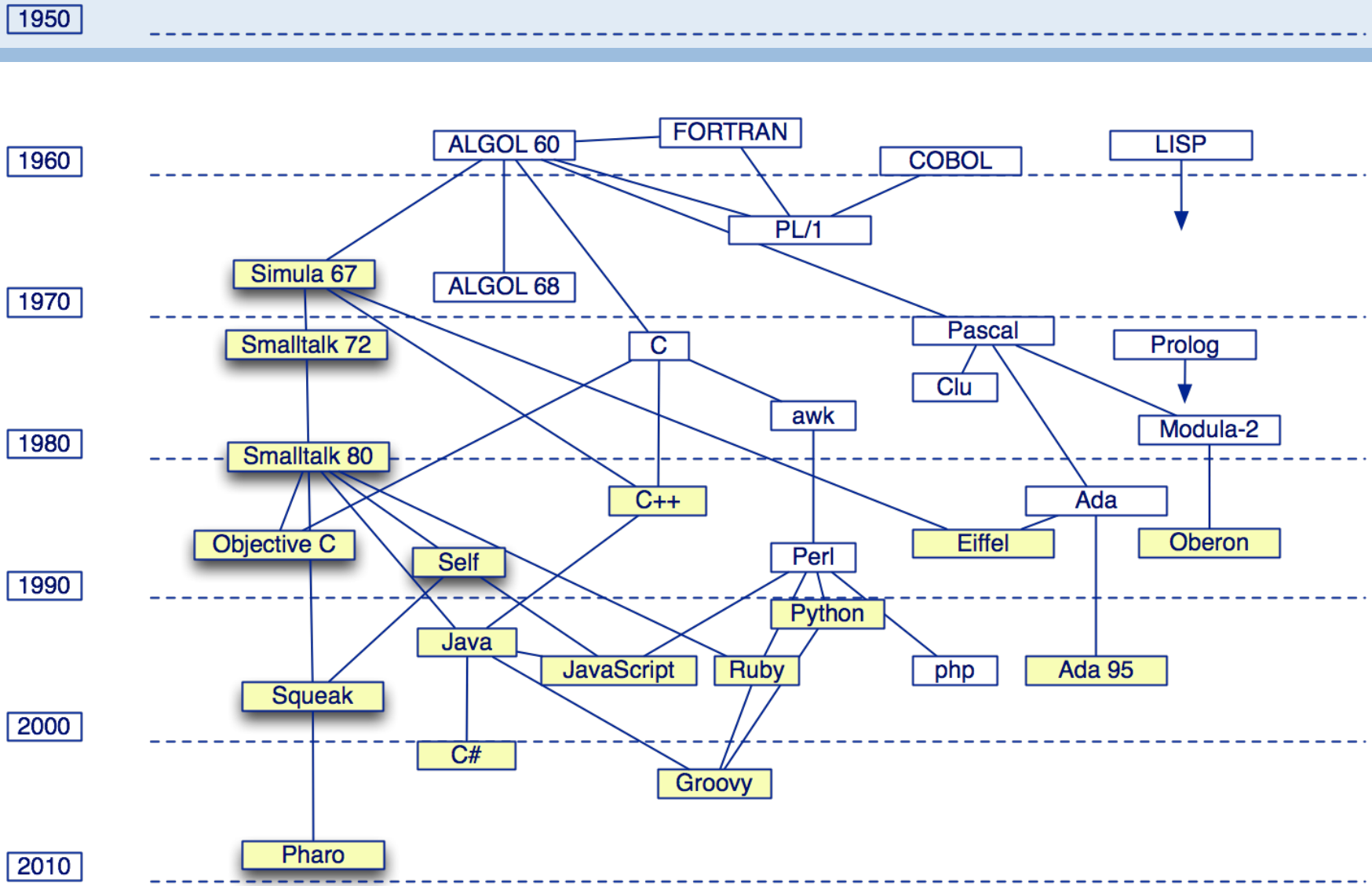


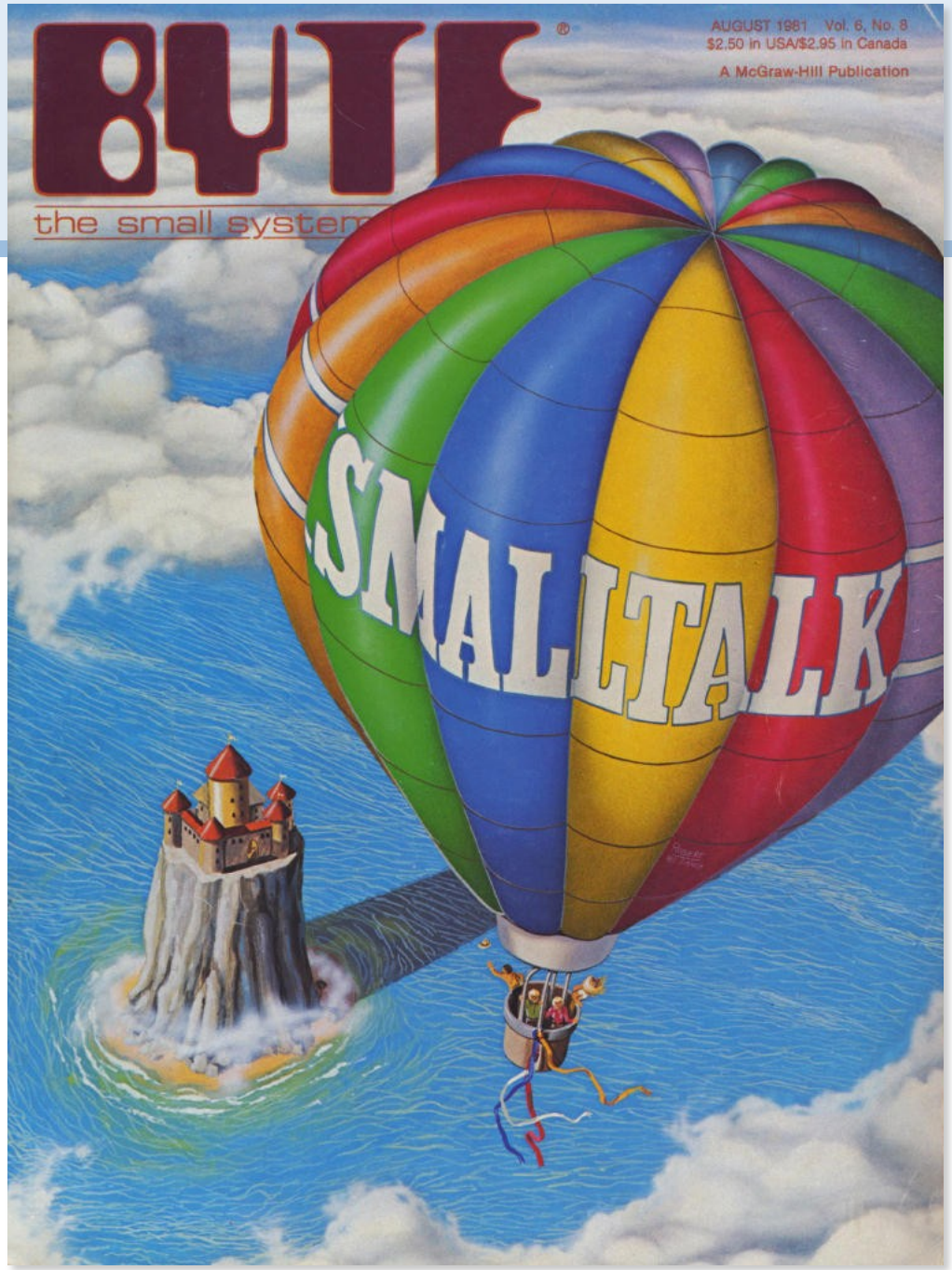
Smalltalk on Alto III

Precursor, Innovator & Visionary

- First to be based on Graphics
  - Multi-Windowing Environment (Overlapping Windows)
  - Integrated Development Environment: Debugger, Compiler, Text Editor, Browser
- With a pointing device  *yes, a Mouse*
- Ideas were taken over
  - Apple Lisa, Mac
  - Microsoft Windows 1.0
- Platform-independent Virtual Machine
- Garbage Collector
- Just-in-time Compilation
- Everything was there, the complete Source Code

History





The History (External)

- **1980 — Smalltalk-80**
  - ASCII, cleaning primitives for portability, metaclasses, blocks as first-class objects, MVC.
  - Projects: Gallery Editor (mixing text, painting and animations) + Alternate Reality Kit (physics simulation)
- **1981 — Books + 4 external virtual machines**
  - Dec, Apple, HP and Tektronix
  - GC by generation scavenging
- **1988 — Creation of Parc Place Systems**
- **1992 — ANSI Draft**
- **1995 — New Smalltalk implementations**
  - MT, Dolphin, **Squeak**, Smalltalk/X, GNU Smalltalk
- **2000 — Fscript, GNU Smalltalk, SmallScript**
- **2002 — Smalltalk as OS: 128k ram**

What are Squeak and Pharo?

- Squeak is a modern, open-source, highly portable, fast, full-featured Smalltalk implementation
  - Based on original Smalltalk-80 code

- Pharo is a lean and clean fork of Squeak
  - www.pharo-project.org

Smalltalk — Key Concepts

- *Everything is an object*
  - numbers, files, editors, compilers, points, tools, booleans …
- Everything happens by *sending messages*
- Every object is an instance of one class
  - which is also an object
  - A class defines the structure and the behavior of its instances.
- Objects have private (protected) state
  - Encapsulation boundary is the object
- Dynamic binding
  - Variables are dynamically typed and bound

Objects and Classes

- *Every object is an instance of a class*
  - A class specifies the structure and the behaviour of all its instances
  - Instances of a class share the same behavior and have a specific state
  - *Classes are objects* that create other instances
  - Metaclasses are classes that create classes as instances
  - Metaclasses describe class behaviour and state (subclasses, method dictionary, instance variables...)

Messages and Methods

- Message — which action to perform

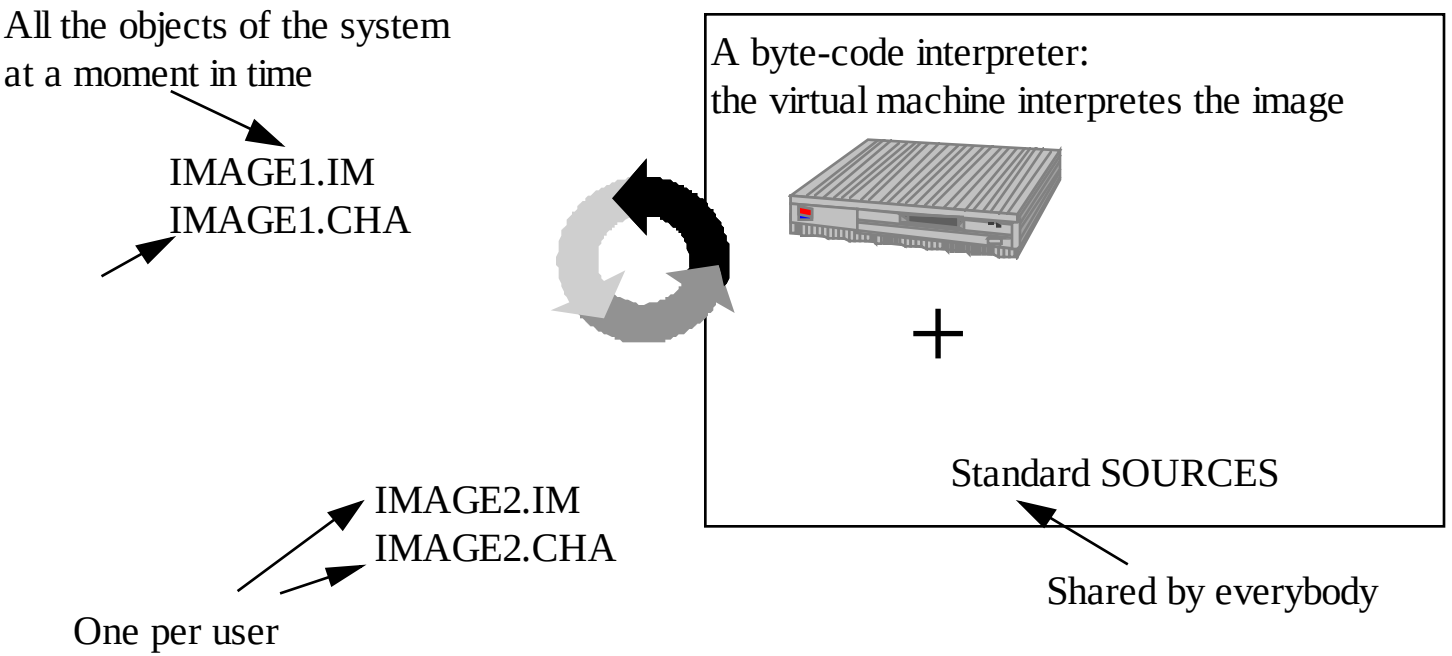
```
aWorkstation accept: aPacket
aMonster eat: aCookie
```

- Method — how to carry out the action

```
accept: aPacket
    (aPacket isAddressedTo: self)
        ifTrue:[
            Transcript show:
                'A packet is accepted by the Workstation ',
            self name asString ]
        ifFalse: [super accept: aPacket]
```

Smalltalk Run-Time Architecture

- Virtual Machine + Image + Changes and Sources

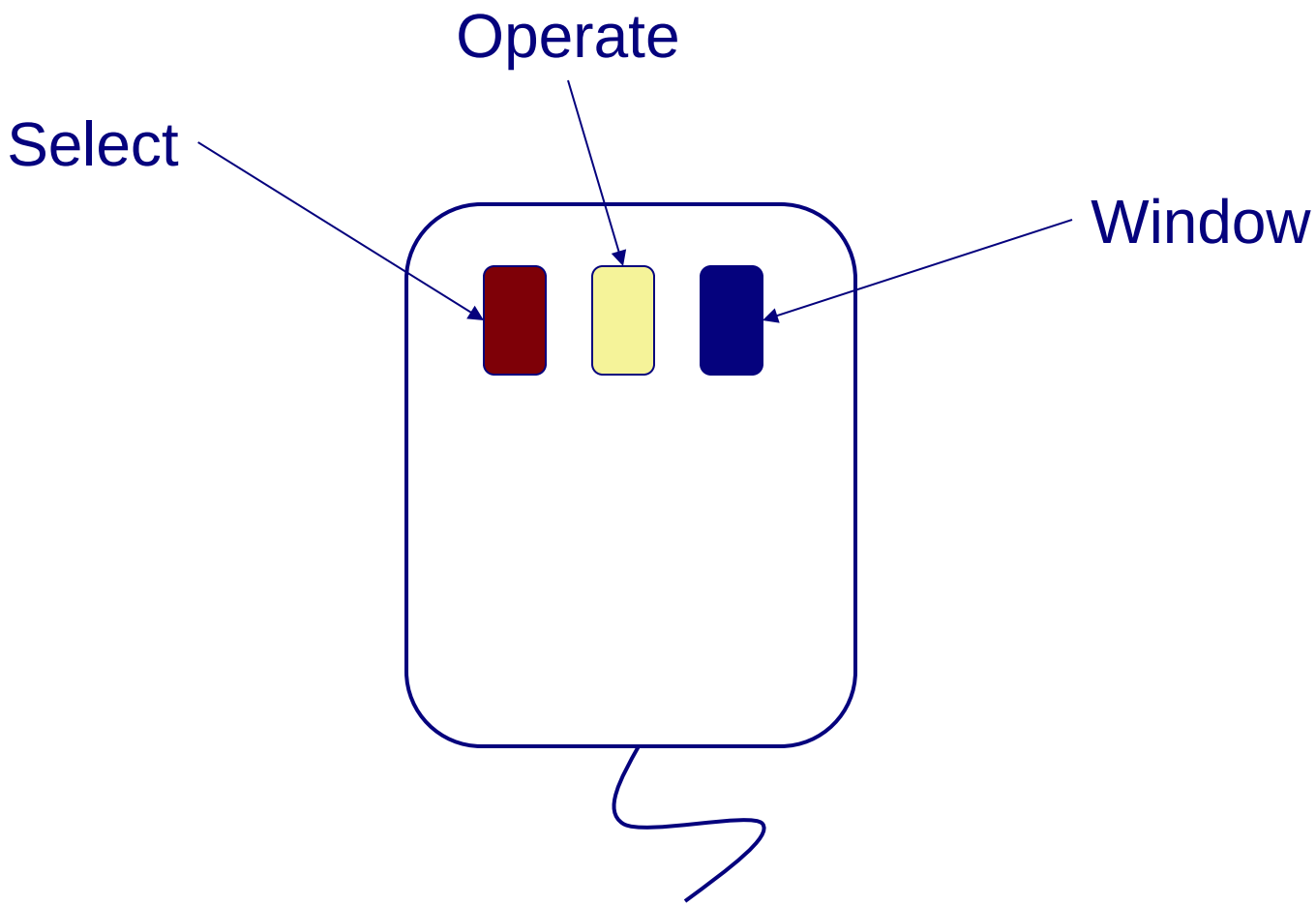


- Image = bytecodes
- Sources and changes = code (text)

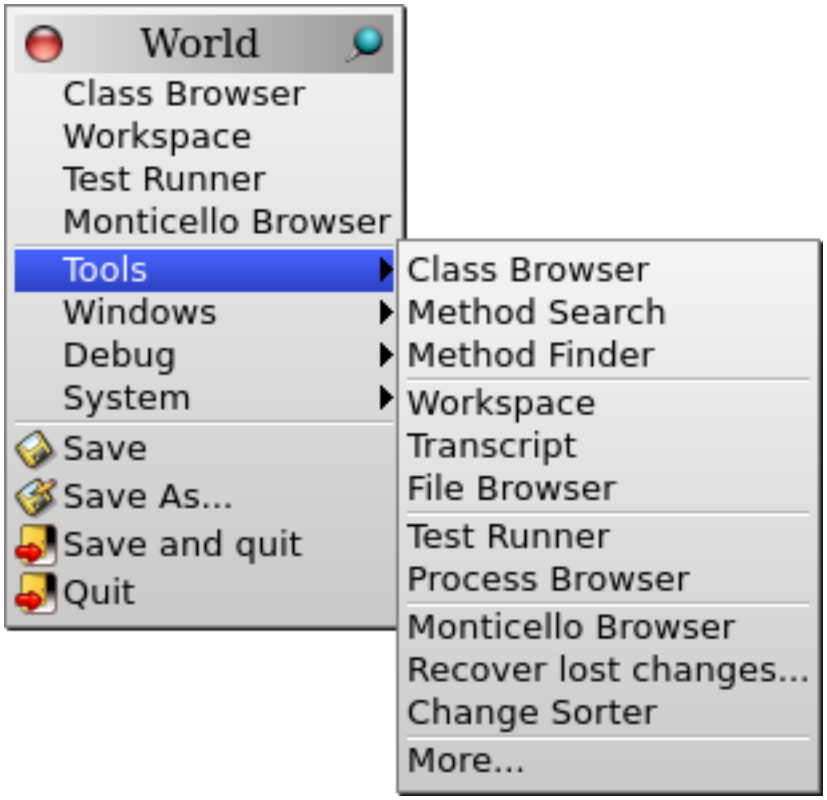
Smalltalk Run-Time Architecture

- Byte-code is translated to native code by a just-in-time compiler
  - Some Smalltalks, but not Pharo
- Source and changes are not needed to interpret the byte-code.
  - Just needed for development
  - Normally removed for deployment
- An application can be delivered as byte-code files that will be executed with a VM.
  - The development image is stripped to remove the unnecessary development components.

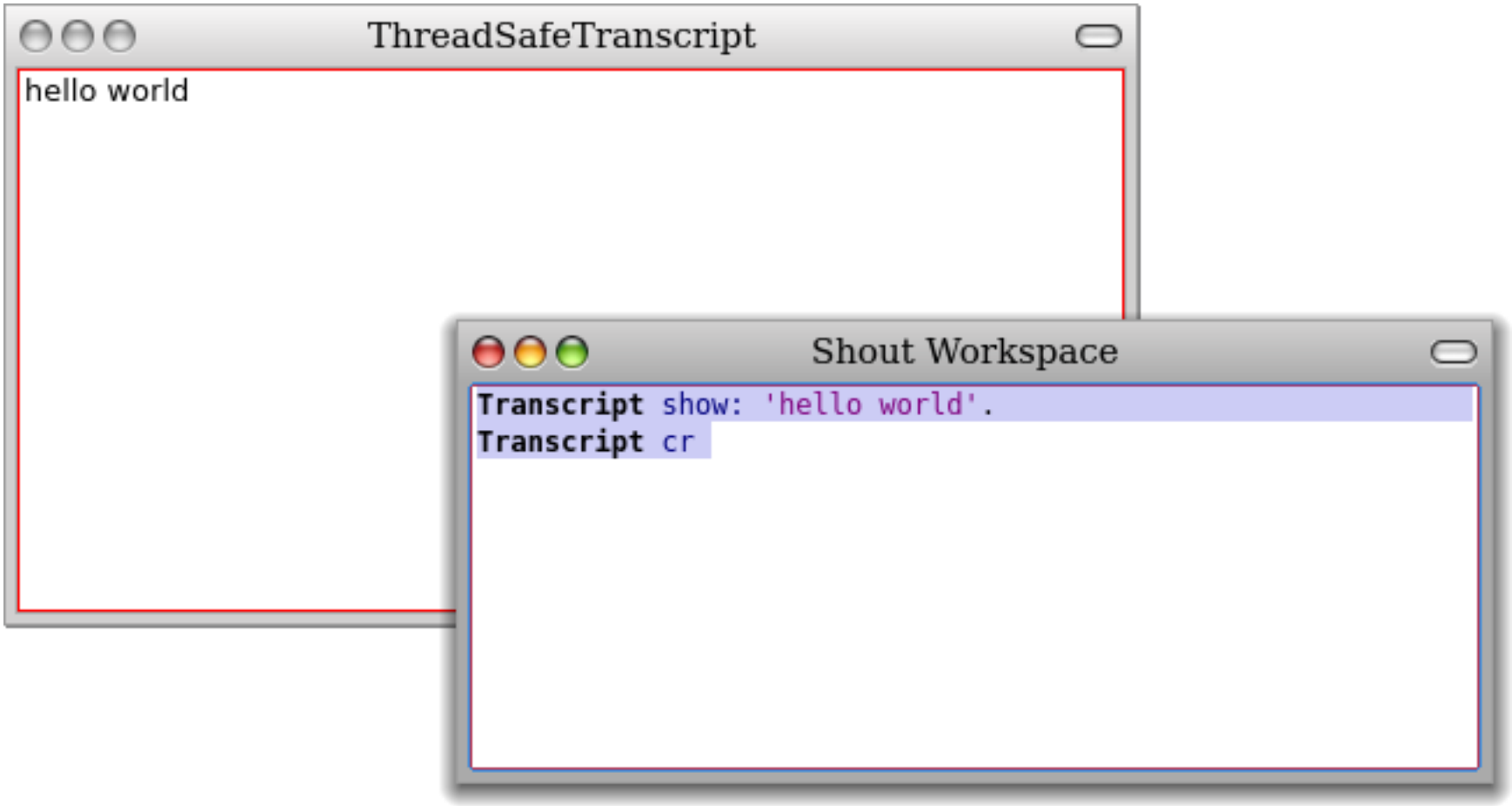
Mouse Semantics



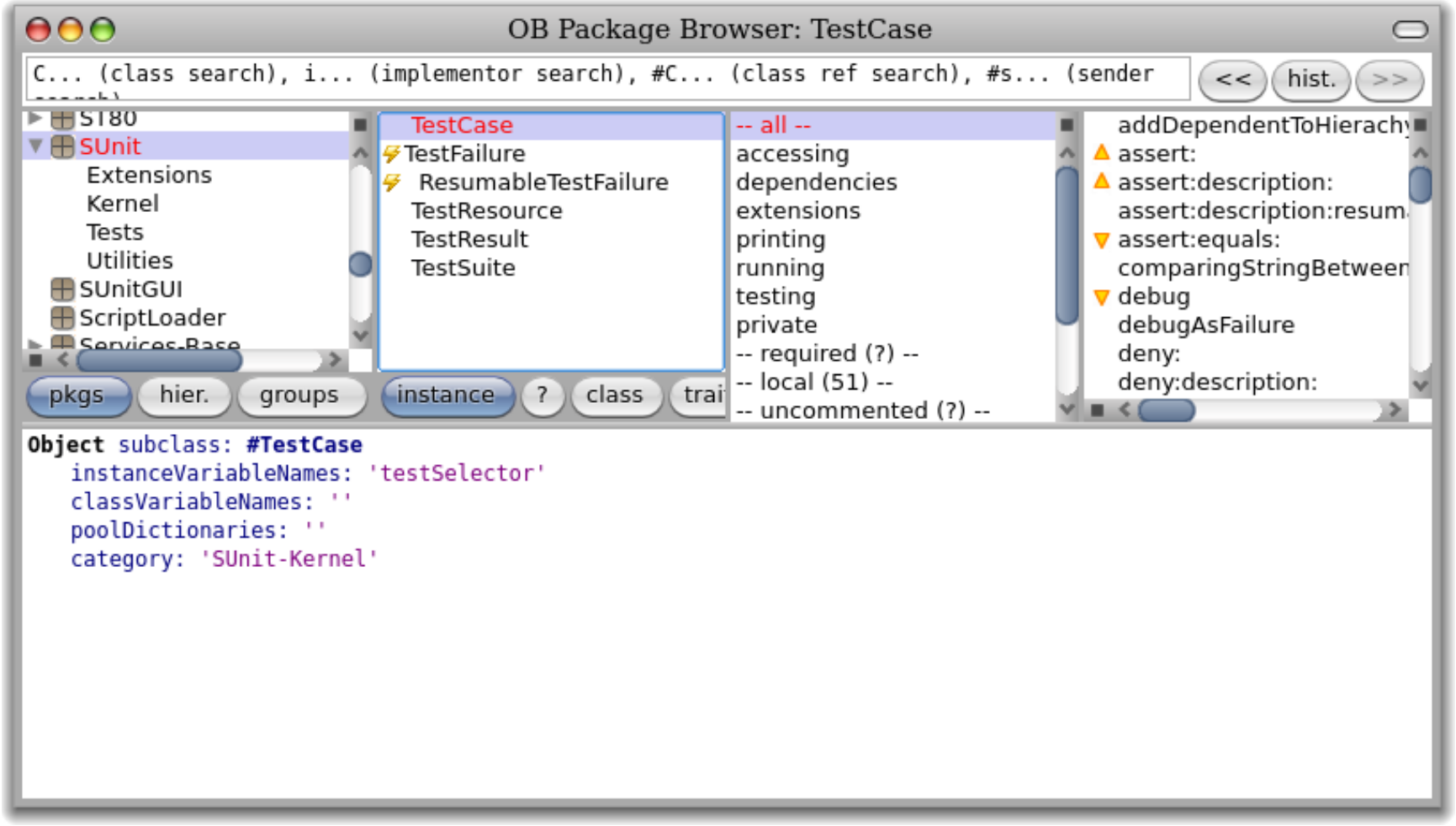
World Menu



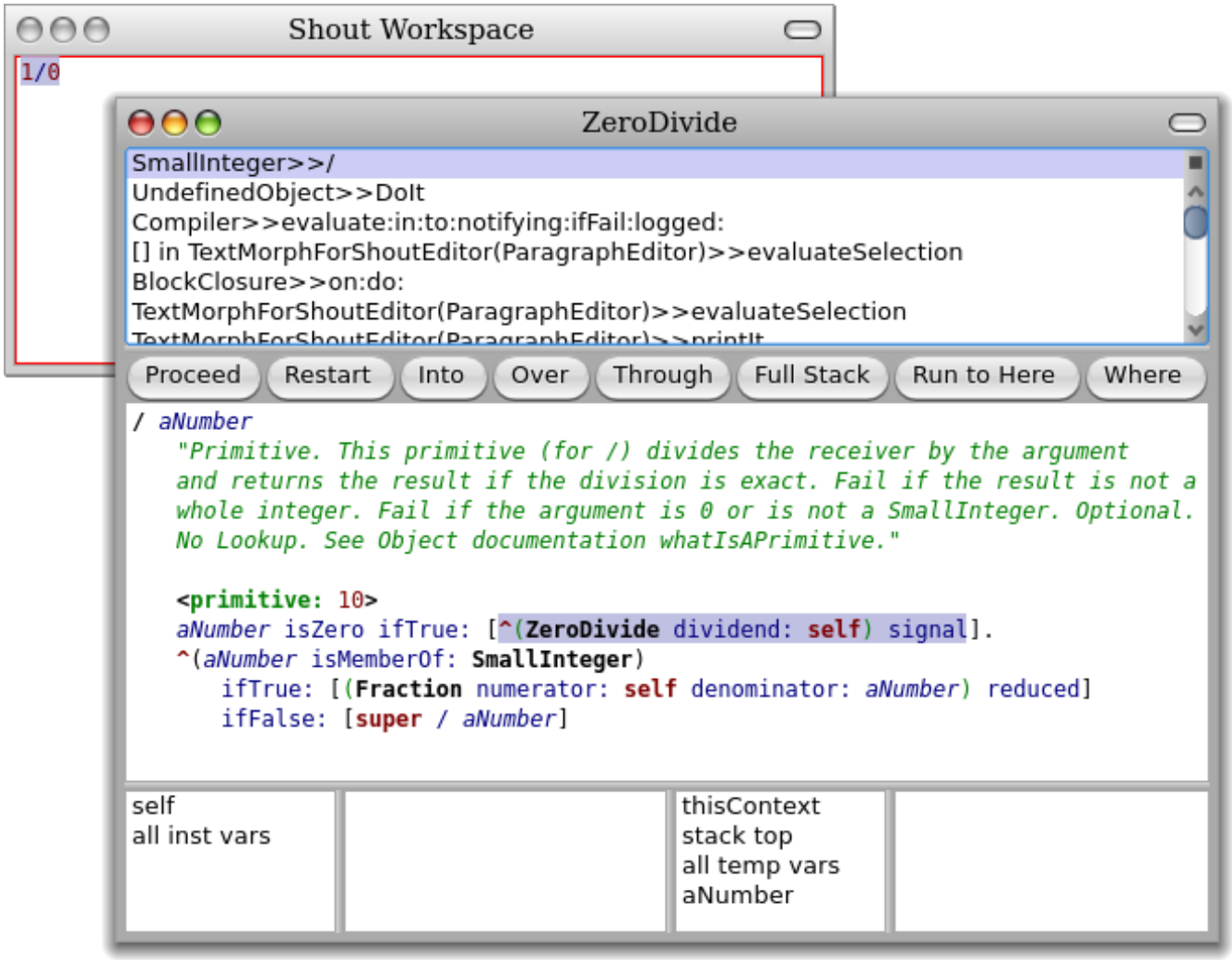
“Hello World”



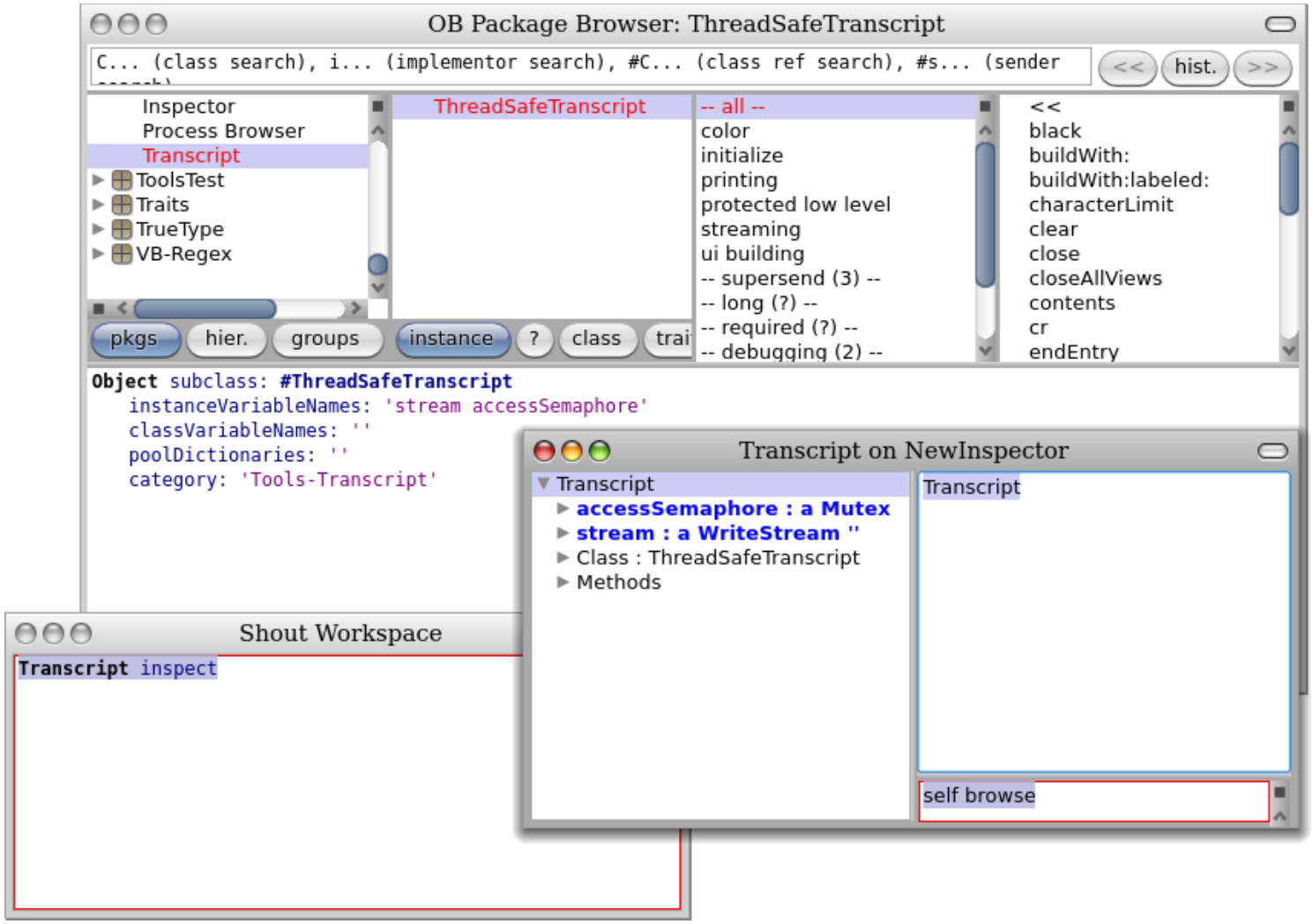
The Smalltalk Browser



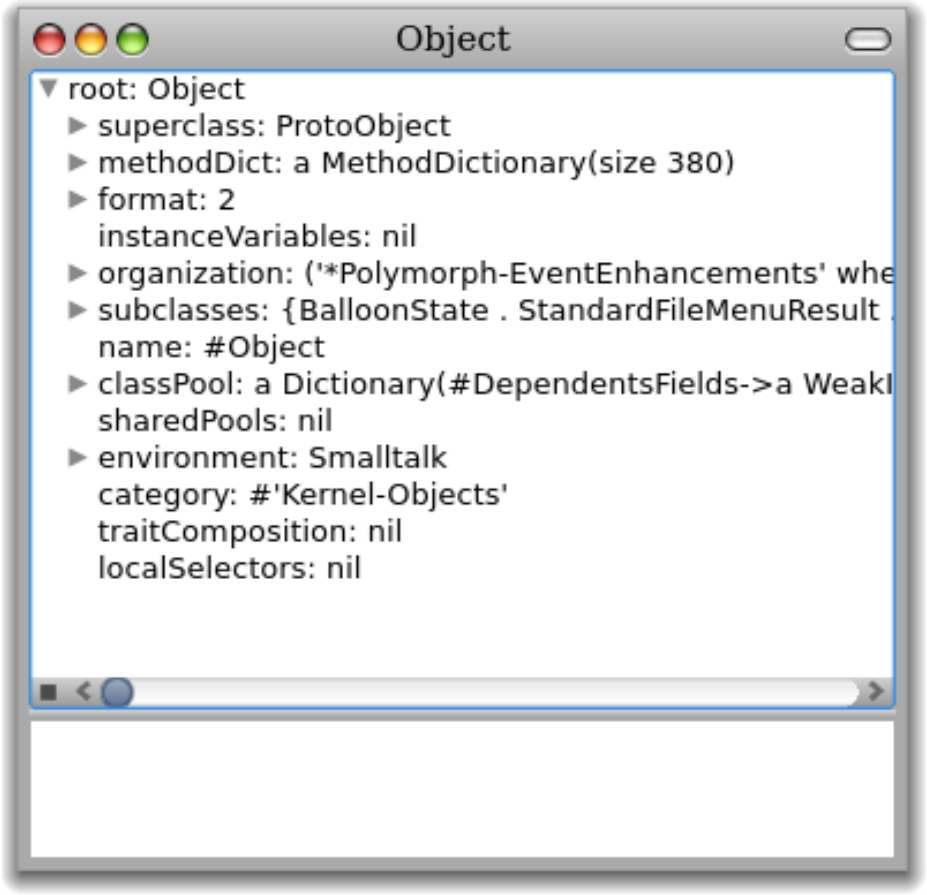
The Debugger



The Inspector



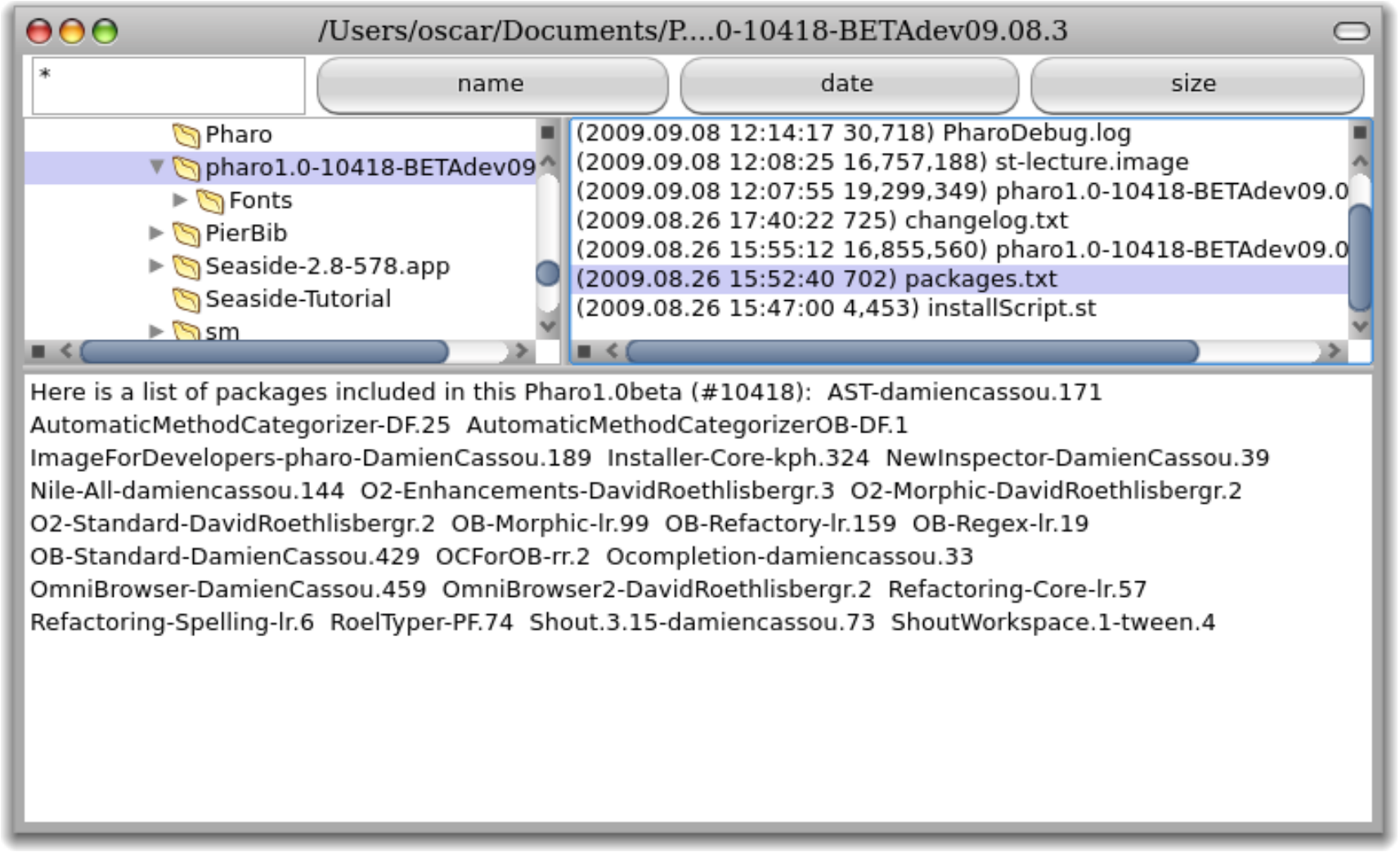
The Explorer



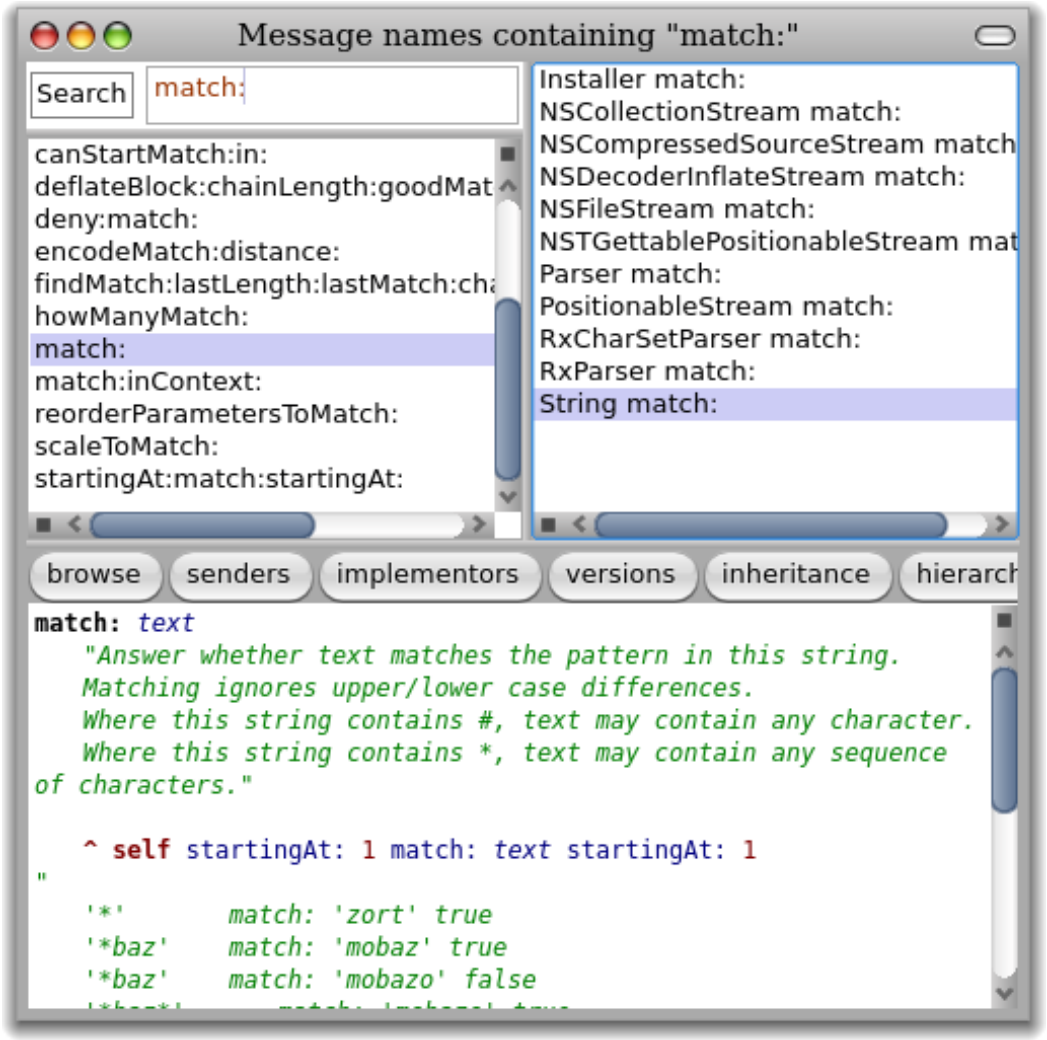
Other Tools

- File Browser
  - *Browse, import, open files*
- Method Finder, Message Name tool
  - *Find methods by name, behaviour*
- Change Sorter
  - *Name, organize all source code changes*
- SUnit Test Runner
  - *Manage & run unit tests*

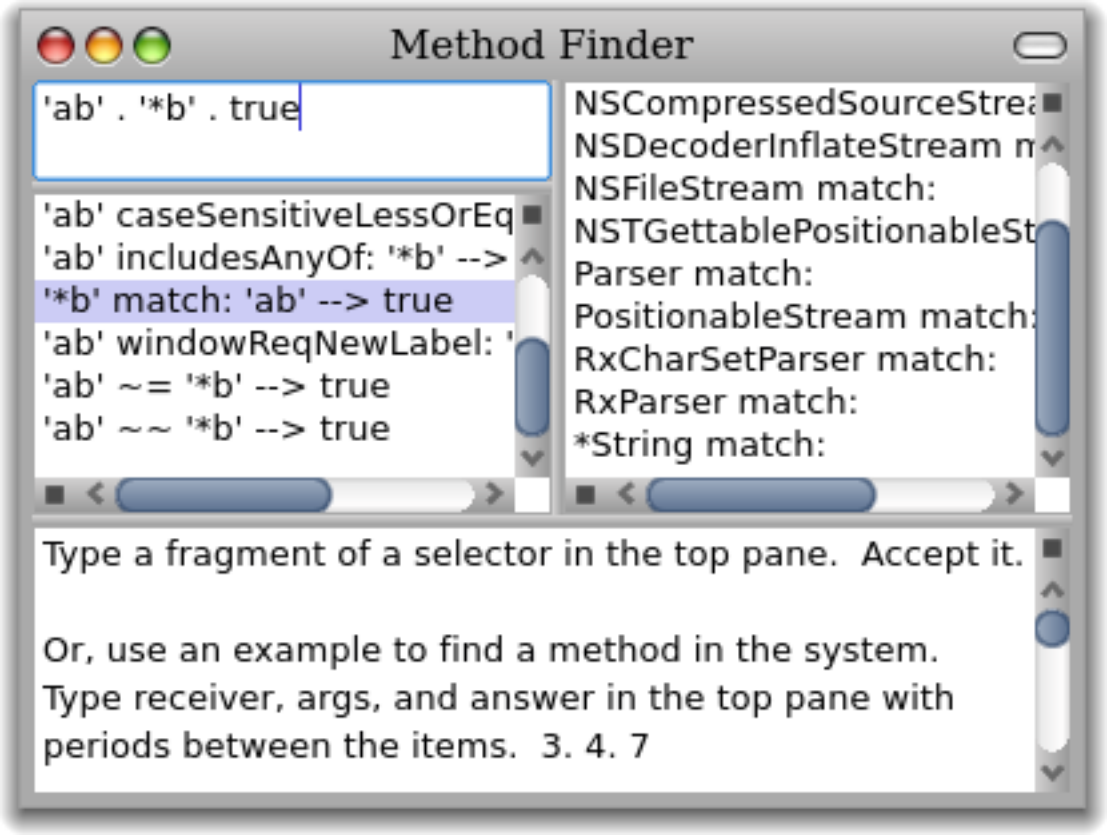
File Browser



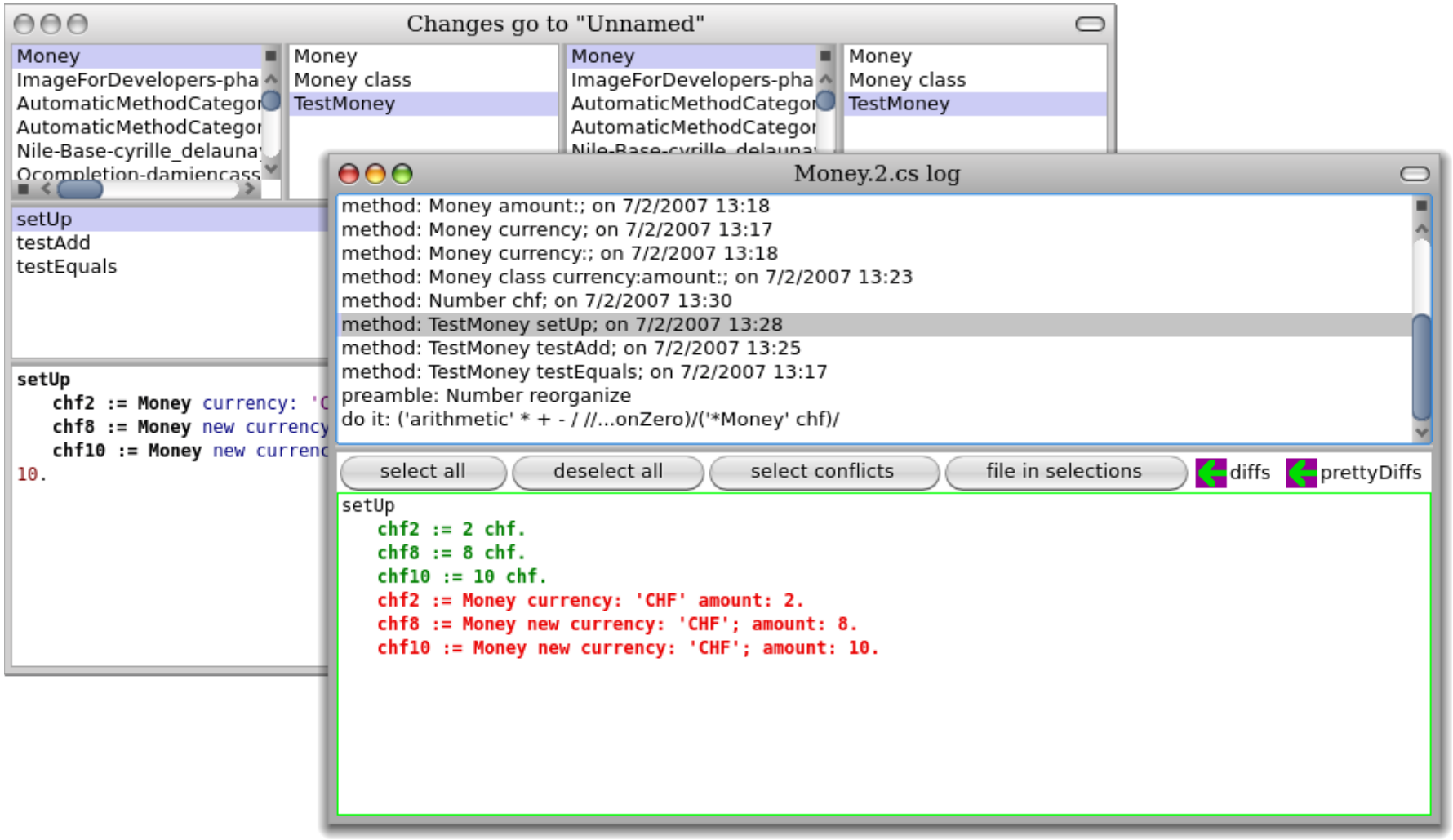
Message Name Finder



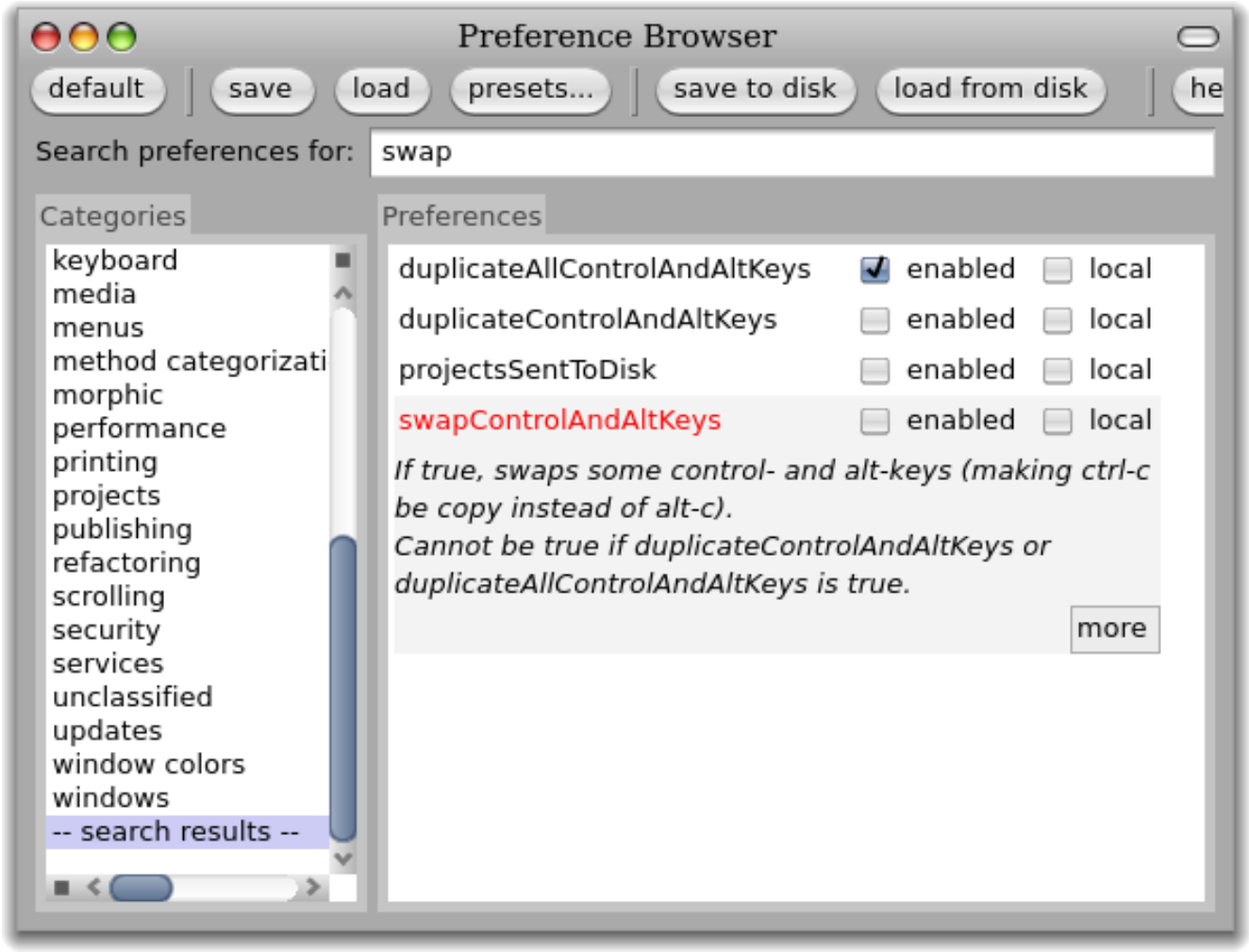
Method Finder



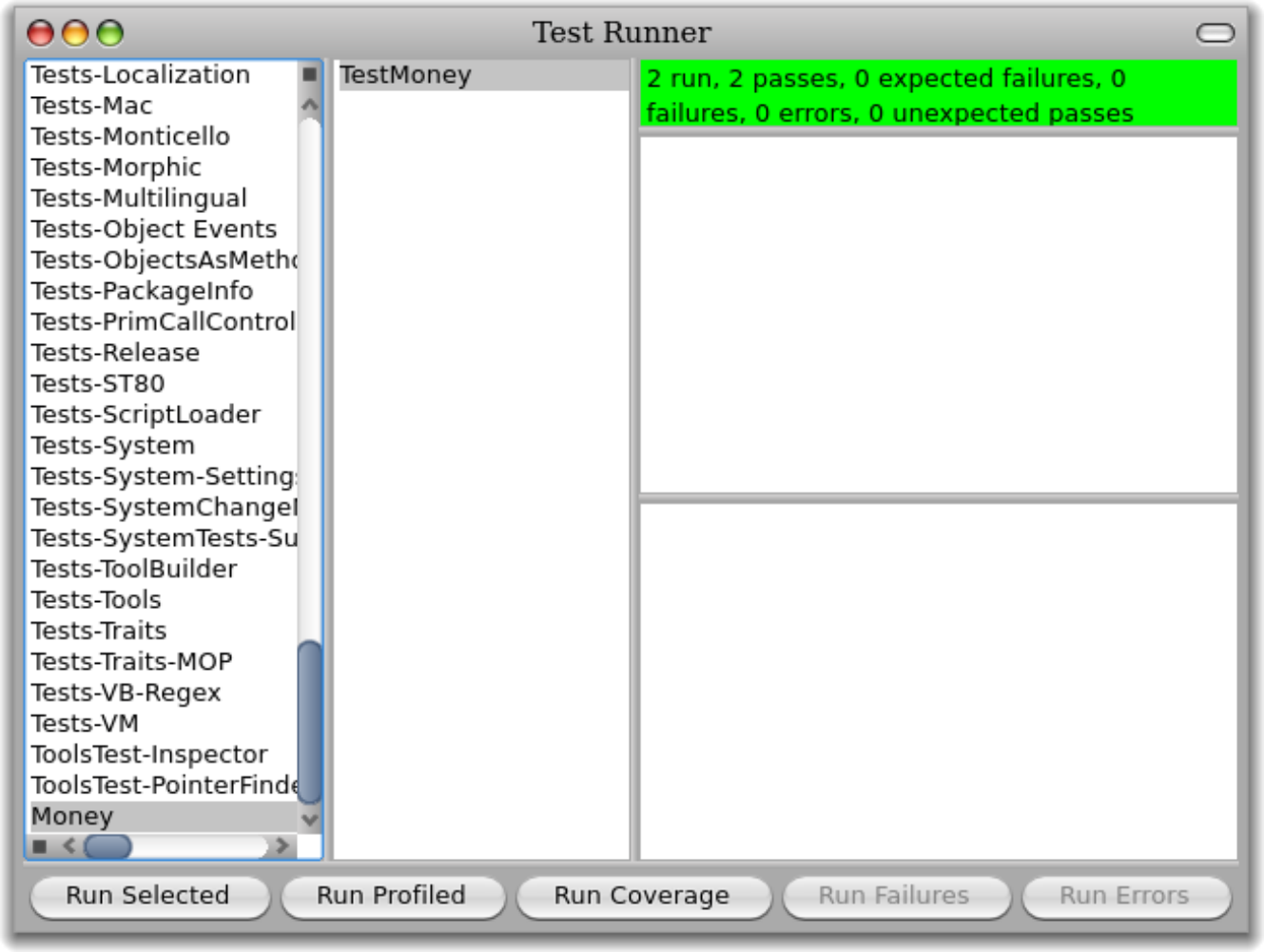
Methods in ChangeSets & Versions



Preferences



SUnit



What you should know!

- *How does Smalltalk differ from Java or C++?*
- *Where are Smalltalk programs stored?*
- *Where are objects stored?*
- *What was the Dynabook?*
- *Is a class an object?*
- *What is dynamic binding?*
- *What is the difference between a message and a method?*

License

http://creativecommons.org/licenses/by-sa/3.0/