A Modular Journey to a World Spanning Virtual Image

Allen Wirfs-Brock
Research Fellow
Mozilla Corp
@awbjs www.wirfs-brock.com/allen

Smalltalks 2014
November 6, 2014
Córdoba, Argentina

Tektronix Invented the Precision Oscilloscope

Tektronix Type 511 Oscilloscope

Versatility...Plus

The Tektronix Type 511 is a portable wide band oscilloscope providing facilities formerly available only in very expensive, cumbersome instruments.

SWEEP CHARACTERISTICS

Continuously variable .1 second to 1 microsecond (10 cm. deflection).
Direct reading sweep speed dial.
Choice of triggered, recurrent or single sweeps at all speeds.
Triggers on sine waves to 10 mc. or pulses over .05 microsecond.
Any 20% of sweep may be expanded 5 times.
DC coupled PP amplifier for external sweep input.

VERTICAL DEFLECTION SYSTEM

Amplifier Bandwidth 10 mc., 1 stage; 8 mc., 2 stages.
Rise Time .04 microsec., 1 stage; .05 microsec., 2 stages.
Maximum Sensitivity .27 V/cm. (Peak to Peak).
Input Impedance Direct 1 meg., 40 mmf.; Probe 10 meg., 11 mmf.

MISCELLANEOUS

Calibrating voltage 0-1, 0-10, 0-100 volts, 60 cycles.
CRT 5CP1A, 5CP7A or 5CP11A operating at 3 kv.
Direct connection to all plates from side panel.
Total weight 65 pounds, self contained.

Price $795.00 f.o.b. Portland

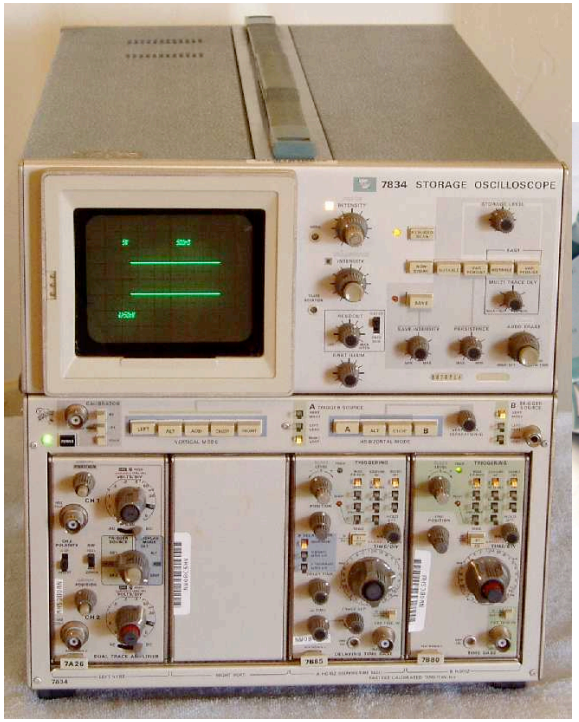
Your inquiry will bring more detailed information and name of the nearest Field Engineering Representative.

TEKTRONIX INC.

Phone, EAst 4885
Cables, TEKTRONIX

712 S. E. Hawthorne Blvd.
Portland 14, Oregon

ELECTRONICS — September, 1948



And by 1970 was one of the worlds largest and most important electronics companies.

1980

- Tek's Graphic Computer System (GCS) was the business unit responsible for desktop graphic computers.
- I was working for GCS building a systems programming oriented Pascal compiler for the Motorola 68000.

GCS Pascal was used as systems programming language for these products.

Tek 4041 BASIC Language 68000-based Computer Disguised as Electronics Instrument

Tek 4909 "networked" File Server

Rebecca Wirfs-Brock was lead SW Engineer

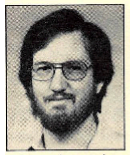
http://www.wirfs-brock.com/allen/files/tek/gcsPascal.pdf

GCS Pascal Modularity

- Interface Units and Implementation Units
- Imports and Exports
- Why Modules?
  - Multiple people developing one program
  - Unmanageable as a single source file

A PASCAL COMPILER FOR MOTOROLA 68000 FIRMWARE DEVELOPMENT

Allen Wirfs-Brock is a software design engineer in the Systems Engineering and Technology Group, part of the Design Automation Division (DAD). In the early 70s, Allen was a programmer in Data Processing. He has since worked on assemblers for the 8002, investigated interactive program environments in Tek Labs, was the project leader for GCS Pascal – he wrote the code generator, and now, in DAD, is investigating object-oriented programming languages. Allen was awarded a BS in computer science at the University of Oregon in 1977, culminating a two-year departure for that purpose.



Paul L. McCullough is a software design engineer in the Systems Engineering and Technology Group, part of the the Design Automation Division. Paul has been implementor or co-implementor of three Pascal compilers. He joined Tektronix four and a half years ago. Previously, Paul was involved in the design and implementation of operating systems and data base management systems at Burroughs Corporation. Paul is currently exploring software engineering environments; in particular, object-oriented programming systems.

Background

Pascal is a computer programming language known for its unique combination of simplicity, power, portability, and rigor. For several years, interest in using Pascal as a microprocessor systems implementation language has existed within Tektronix. However, the lack of high-quality Pascal compilers for commonly used microprocessors has limited the use of Pascal for the development of product firmware. Pascal interpreters (such as UCSD Pascal) have been available, but their performance has not been adequate for most firmware applications.

In early 1980, GCS engineering was about to start several firmware-intensive product development efforts employing the Motorola 68000 microprocessor. Because of the size and complexity of the projects, a high-level language was considered to be an essential implementation tool. Unfortunately the only high-level language available for the 68000 at that time (a Pascal compiler developed outside of Tektronix) was neither powerful nor reliable enough for Tek product firmware. For these reasons, GCS engineering chose to develop its own 68000 Pascal compiler.

Specification of the Language

Pascal was originally designed to be an instructional language for mainframe computers. For this reason, Pascal lacks several features that are generally considered essential for a microprocessor system implementation language. Such features include:

- separate compilation of Pascal procedures,
- the ability to call assembly language routines from Pascal,
- the ability to write interrupt service routines in Pascal,
- the association of Pascal variables with absolute memory locations (primarily to support memory-mapped input and output), and
- efficient manipulation of bit fields.

Many implementations of Pascal have attempted to correct these deficiencies via numerous extensions to the language (for example, some compilers extend Pascal variable declarations to include an absolute address specification). Such extensions often result in a plethora of special cases which destroy the elegant consistency of Pascal.

The design of GCS 68000 Pascal attempts to avoid such special cases. It implements the language as defined in the proposed international standard for Pascal. Standard Pascal is sufficently flexible so that a well designed implementation may support several systems-programming features without modifying the language definition. For example, Pascal subrange types may be used to declare unsigned, or "short" integer variables, and Pascal sets may be used to manipulate bits. GCS Pascal recognizes such special usages and attempts to generate optimal code for them.

The standard language was augmented with a small, consistent set of extensions to support systems programming. Minor extensions, which have been widely accepted by Pascal users, include non-decimal numeric constants and a default case statement alternative. The only major extension supports modular programming.

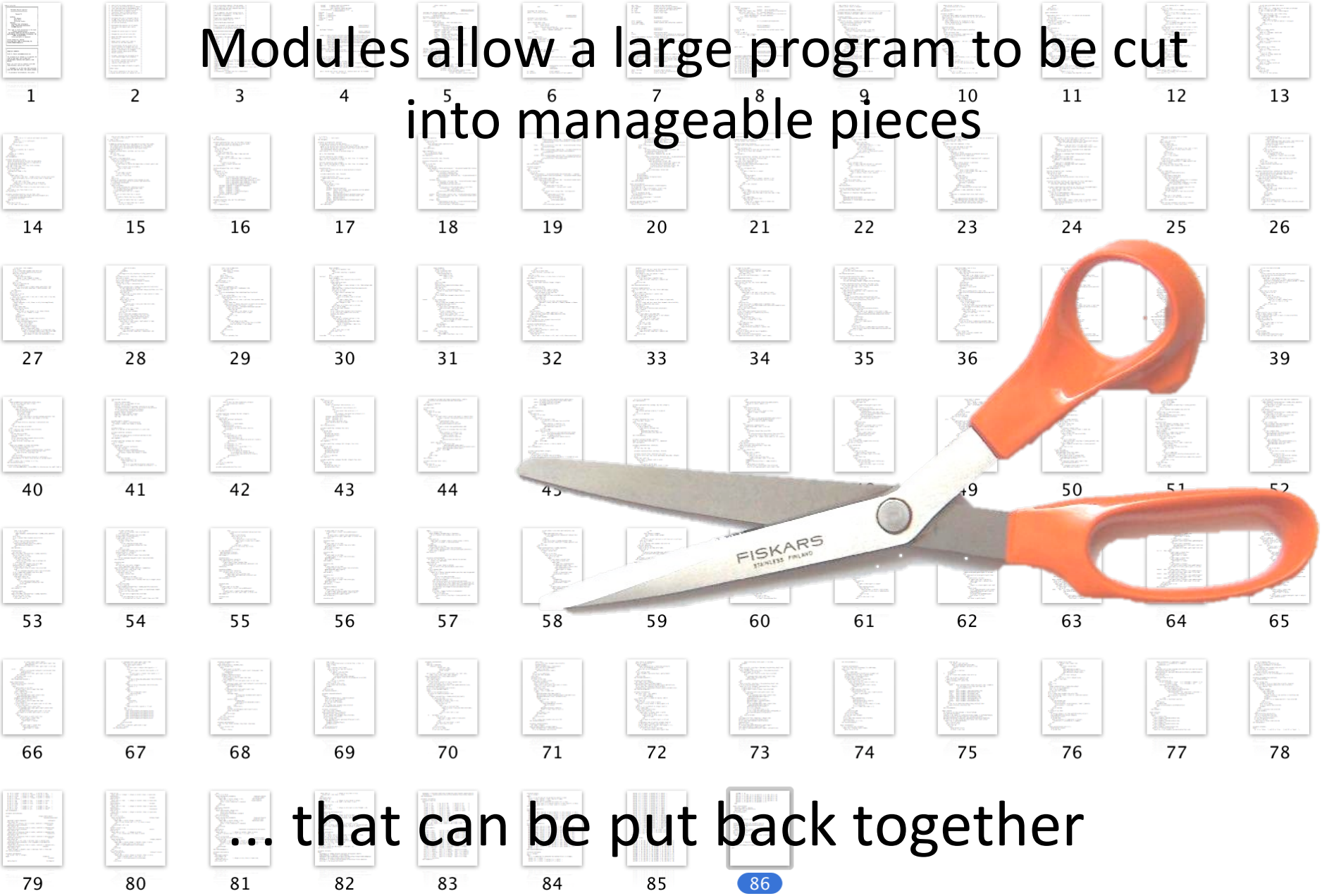
Modularity Features

GCS Pascal supports three forms of separately compilable modules, referred to as units. A *program unit* is a Pascal main program. An *interface unit* provides definitions of objects (constants, types, variables, procedures, and functions) that may be used within other units. An *implementation unit* implements the objects that are defined in an interface unit. An implementation unit can be written in either assembly language or Pascal.

The modularity features of GCS Pascal are quite powerful. The runtime routines that handle text input/output (that is, reading integers or characters, or the writing of integers, strings, Booleans, and characters) for GCS Pascal are entirely written in GCS Pascal. As another example, several of the

http://www.wirfs-brock.com/allen/files/tek/gcsPascal.pdf

Modules allow a large program to be cut into manageable pieces



... that can be put back together

Phase 1 – Review the Book

Tektronix
COMMITTED TO EXCELLENCE

INTER-OFFICE COMMUNICATION

TO: List

DATE: 10-14-80

FROM: Paul L. McCullough

SUBJECT: Smalltalk Review

The first meeting of the Smalltalk review group will be Thursday, October 16, in building 63, conference room W-52, from 9:30 to 11:00.

The agenda is:

* Overall impressions concerning the portion of the book that we have received.
* Specific comments on the prologue, each chapter, and appendix.
* Summation of comments and impressions.
* Discussion of whether to return these chapters or retain them a while longer.

PM:jkb

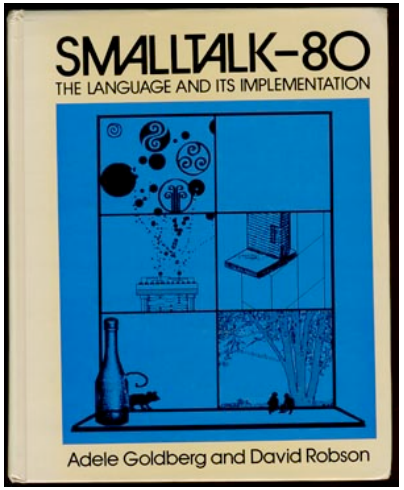
List: Jack Grimes
Dave Heinen
Larry Katz
Bob Reed
Rick Samco
Don Williams
Allen Wirfs-Brock

000-8691-00

SMALLTALK–80 PROJECT: PHASE I

* BOOK REVIEW FOR XEROX
* 12 CHAPTERS:
  + 10 RECEIVED
  + 9 REVIEWED
* 5 PEOPLE:
  + L. KATZ
  + P. McCULLOUGH GCS/DAD
  + A. WIRFS–BROCK
  R. SAMCO – TEK LABS
  R. REED – T&D
* BEGAN SEPT. 1980

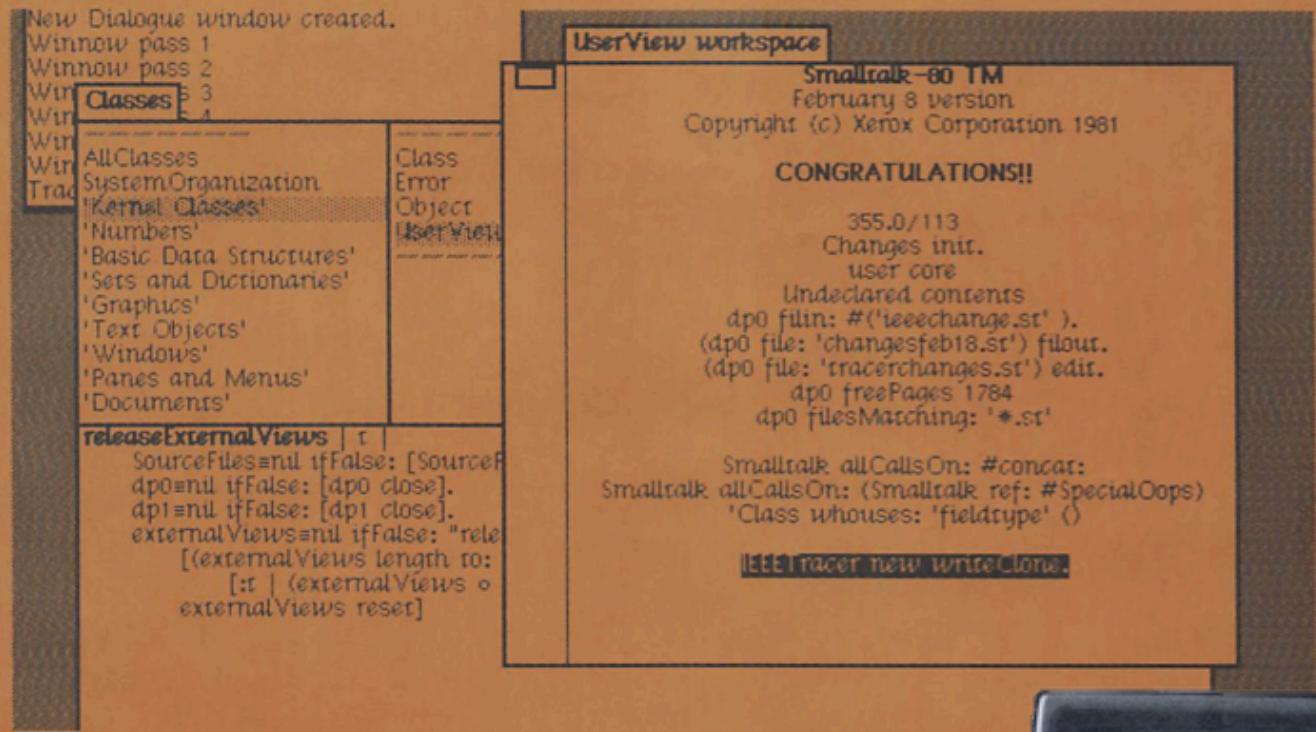
February 1981 Status



http://www.wirfs-brock.com/allen/files/tek/1980-10-14-1st%20book-review-meeting.pdf
http://www.wirfs-brock.com/allen/files/tek/1981-2-Larry-Katz-intro-and-status.pdf

July 1981

First display output of a Tektronix Smalltalk implementation.

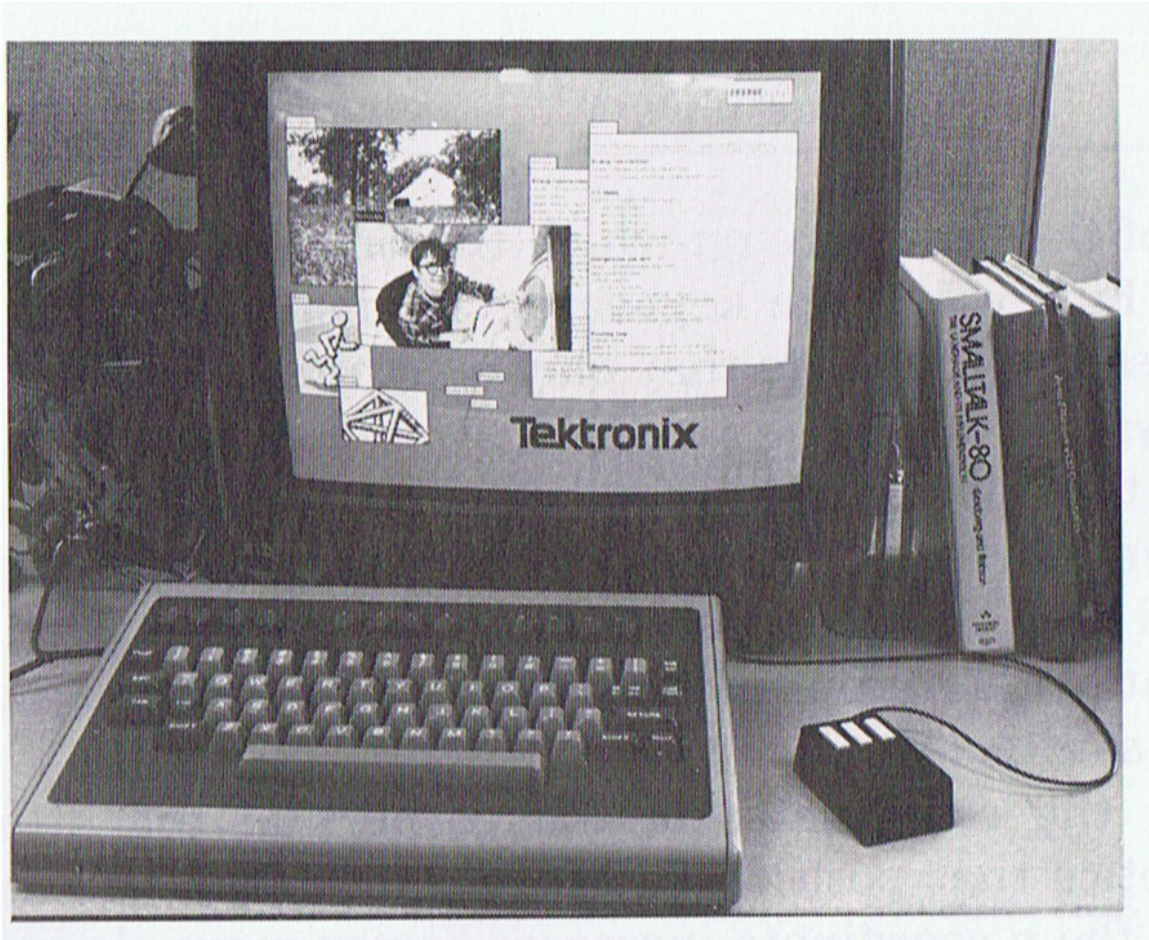


68000 computer, virtual machine coded in GCS Pascal. RS-232 interface to a Tek 4025 raster graphics terminal.

Rendering this image took over an hour. 4025 display memory was exhausted before the complete screen could be rendered.

http://www.wirfs-brock.com/allen/files/tek/1981-7-first-welcome-screen.pdf

Magnolia Smalltalk rapidly became the primary language for CS researchers within Tek Labs.



In late 1982 and again in 1983 Magnolias running Smalltalk where the hit of the Tek Labs "science fair" where lab projects were show cased to the entire Tek engineering community.

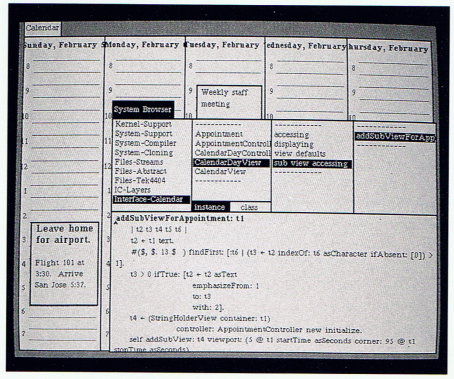
Several key senior executives said: "We really should do something with this..."

http://www.wirfs-brock.com/allen/things/smalltalk-things/tektronix-smalltalk-document-archive

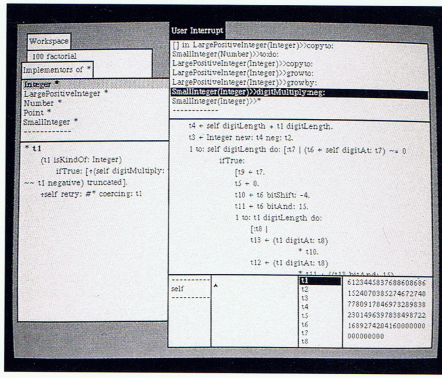
1984

THE TEK 4404: THE FIRST PERSONAL AI DEVELOPMENT SYSTEM.

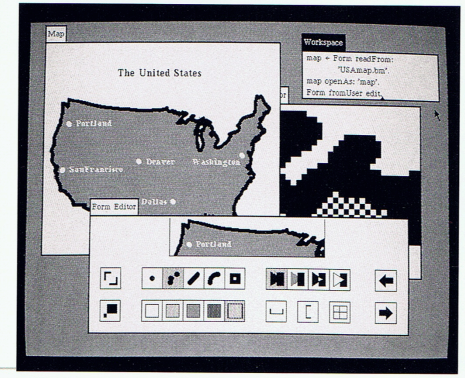
AI DEVELOPMENT AT THE SPEED OF THOUGHT. AT THE PRICE OF A PERSONAL SYSTEM.



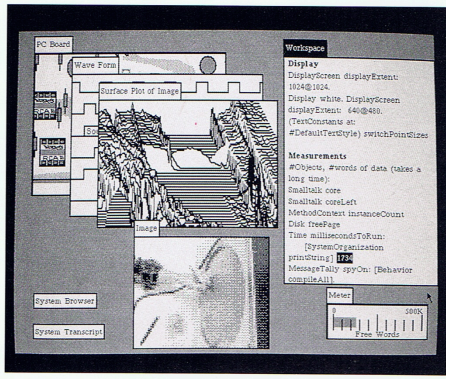
The 4404 facilitates quick, efficient prototyping. Page-on-demand memory management provides a large, 8-Mbyte virtual memory address space that permits development of complex programs without segmentation or overlays.



The bit-mapped display facilitates advanced user information concepts such as overlapping windows, "pop-up" menus, and mouse input.



The 4404's 640x480 display functions as a window into a 1024x1024 bit-mapped memory, with smooth panning whenever the cursor reaches a physical display edge. Users also have the ability to point with the mouse and integral joydisk.



With its proprietary Smalltalk-80 implementation, graphics performance on the 4404 makes on-screen animation possible. The Smalltalk-80 package offers advantages of a highly integrated programming environment and object-oriented language with an excellent user interface.

Tektronix®
COMMITTED TO EXCELLENCE

PRICED UNDER $15,000;* THE 4404 IS AN UNMATCHED VALUE FOR AI ENVIRONMENTS.

Announced: August 6, 1984,
AAAI Conference, Austin Texas

http://www.wirfs-brock.com/allen/files/tek/4404-Flyer.pdf

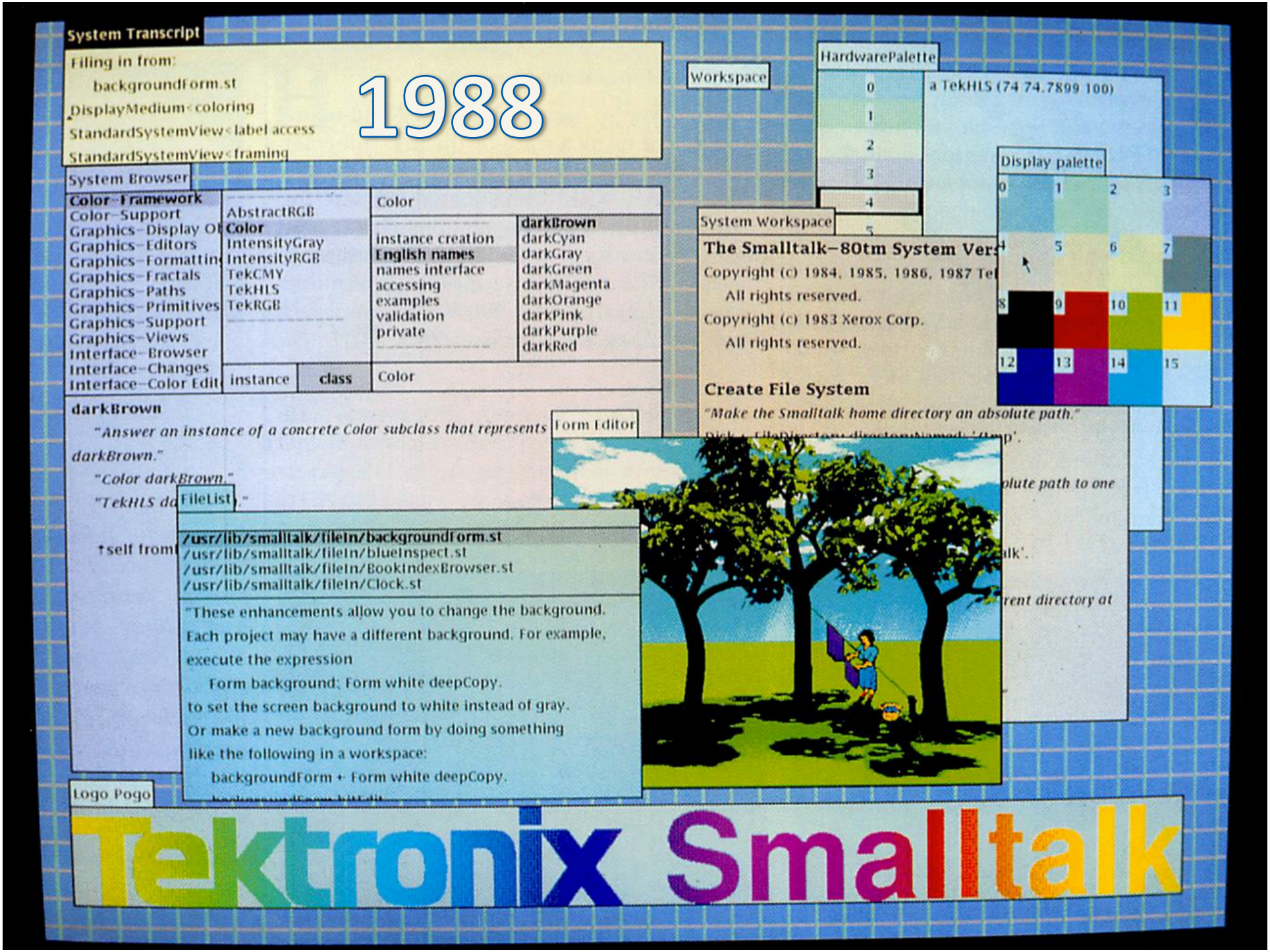
Tek LOS (Large Object Space) Smalltalk

1985-1986

Designed for 68020-based 4405 and 4406
Near Dorado Performance, 19” 1280x1024 display

- 32-bit object pointers
- No object table
- 31-bit small integers
- Multi-generation GC
- Large (>64KB) objects
- Large object GC regions
- Overlapping, stack allocated contexts
- Optimized for 68020 instructions set

We think that the Tek LOS Smalltalk may have been the first shipping commercial product, running on an off-the-self processor, to use a generational GC.

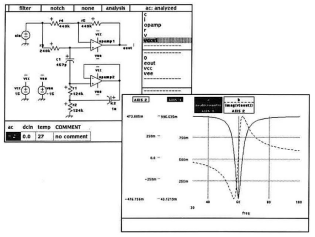

System Transcript
Filing in from:
backgroundForm.st
DisplayMedium<coloring
StandardSystemView<label access
StandardSystemView<framing
1988
Workspace
HardwarePalette
a TekHLS (74 74.7899 100)
Display palette
System Browser
Color-Framework
Color-Support
Graphics-Display Of
Graphics-Editors
Graphics-Formattin
Graphics-Fractals
Graphics-Paths
Graphics-Primitives
Graphics-Support
Graphics-Views
Interface-Browser
Interface-Changes
Interface-Color Edit
AbstractRGB
Color
IntensityGray
IntensityRGB
TekCMY
TekHLS
TekRGB
instance creation
English names
names interface
accessing
examples
validation
private
darkBrown
darkCyan
darkGray
darkGreen
darkMagenta
darkOrange
darkPink
darkPurple
darkRed
instance
class
System Workspace
The Smalltalk-80tm System Vers
Copyright (c) 1984, 1985, 1986, 1987 Tel
All rights reserved.
Copyright (c) 1983 Xerox Corp.
All rights reserved.
Create File System
"Make the Smalltalk home directory an absolute path."
darkBrown
"Answer an instance of a concrete Color subclass that represents darkBrown."
"Color darkBrown."
Form Editor
FileList
/usr/lib/smalltalk/fileIn/backgroundForm.st
/usr/lib/smalltalk/fileIn/blueInspect.st
/usr/lib/smalltalk/fileIn/BookIndexBrowser.st
/usr/lib/smalltalk/fileIn/Clock.st
"These enhancements allow you to change the background.
Each project may have a different background. For example,
execute the expression
Form background; Form white deepCopy.
to set the screen background to white instead of gray.
Or make a new background form by doing something
like the following in a workspace:
backgroundForm ← Form white deepCopy.
Logo Pogo
Tektronix Smalltalk

Analog Design System

USERS GUIDE

Analog Design System

Version 2C



CAX Center
EE Simulation Group

COMPANY CONFIDENTIAL

October 2, 1989

Section 2. ADS BASICS

ADS is interactive beyond what you may have experienced with other simulation systems. The design processes supported in ADS are tightly coupled with tasks such as schematic entry, plotting, and model development supported in windows by menus specific to the context. These windows, menus, scroll bars, and mouse clicks speed and simplify your interaction with the system. This figure shows an example of an ADS display.

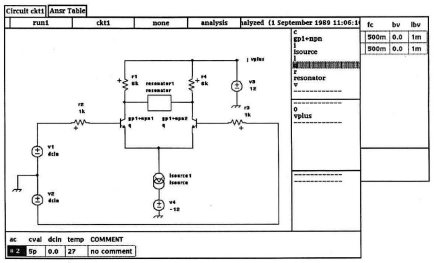


Figure 2-1. A typical ADS screen display

The display shows two windows and an icon representing a collapsed window. Windows are identified by tabs in the upper-left corners, most easily managed if you overlap them with the tabs showing as you would file folders.

ADS WALK THROUGH *ADS USERS GUIDE*

Using a Subcircuit

Return to the **MAIN CIRCUIT EDITOR** (in edit mode). **select** the current source and the ground symbol by dragging the cursor diagonally through a box that covers them while the left-button is depressed.

1. **cut** the selected components.
2. Click on (highlight) *isource* in the model list and **paste** the symbol in place of the current source.
3. Add a -12 volt source below the current source model. **paste** *v* from the model list and use the **ELEMENT BROWSER** to set its *dc* parameter.
4. Connect *isource* and the new voltage source into the circuit, grounding the bottom of the voltage source. (If the subcircuit and source are not connected with a wire between, the middle-button menu for a node is not available.)
5. Select the global node *vplus* and **paste** it onto the node connected to element v3.

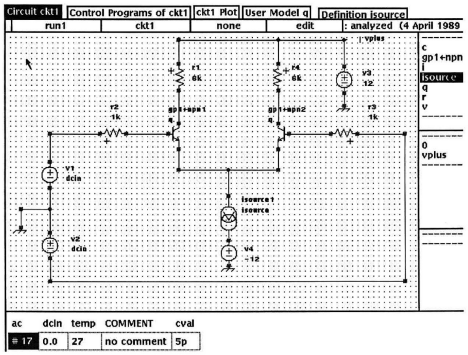
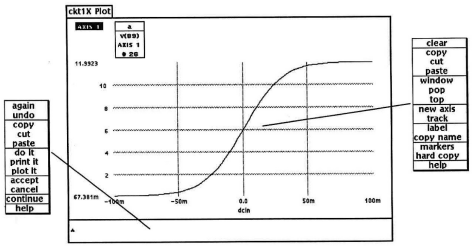


Figure 30. Revised circuit with current source



PLOT Window Map

ADS stands for Analog Integrated Circuit Design System (ADS). Written in ParcPlace Smalltalk-80 (2100 classes, 33.8K methods, 12Mb source file as of 10-9-92), ADS provides a fully integrated system for drawing schematics, viewing the results of simulations, and producing design documentation. ADS has been in production use within Tektronix since June 1988. There were over 125 ADS users in 1992. The ADS program is still in production at Tektronix (as of March 2001).

Dale Henrichs started the project using the Tek 4404 in January of 1985. When Tektronix got out of the Smalltalk business he ported ADS to ObjectWorks.

ADS WALK THROUGH *ADS USERS GUIDE*

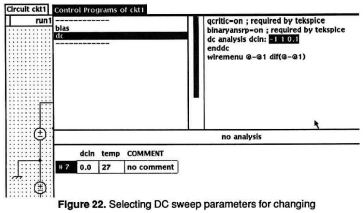
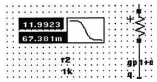


Figure 22. Selecting DC sweep parameters for changing

Click on **accept** in the middle-button menu to record the changes.

What is varied during the analysis is the variable **dcIn**. While this variable is given a default value of zero in the SIMULATION CONTROL PANE below, during a dc analysis the variable is swept over the range in the increments that you specify in the dc analysis command.

3. Move the cursor to the SIMULATION CONTROL PANE (bottom) and select **execute** from the middle-button menu.
4. When the analysis finishes, return to the **SCHEMATIC EDITOR**. If in edit mode, switch to analysis mode by clicking the left-button on **edit** in the EDIT/ANALYSIS PANE, (you toggle between edit and analysis in this pane). This pane is the second one from the right at the top of the window. Point at a wire connected to the collector of one of the transistors and press the middle-button. Release the button on **v( )** and move the mouse to place the "postage stamp" plot near the collector. Click the left-button to fix the plot on the schematic.

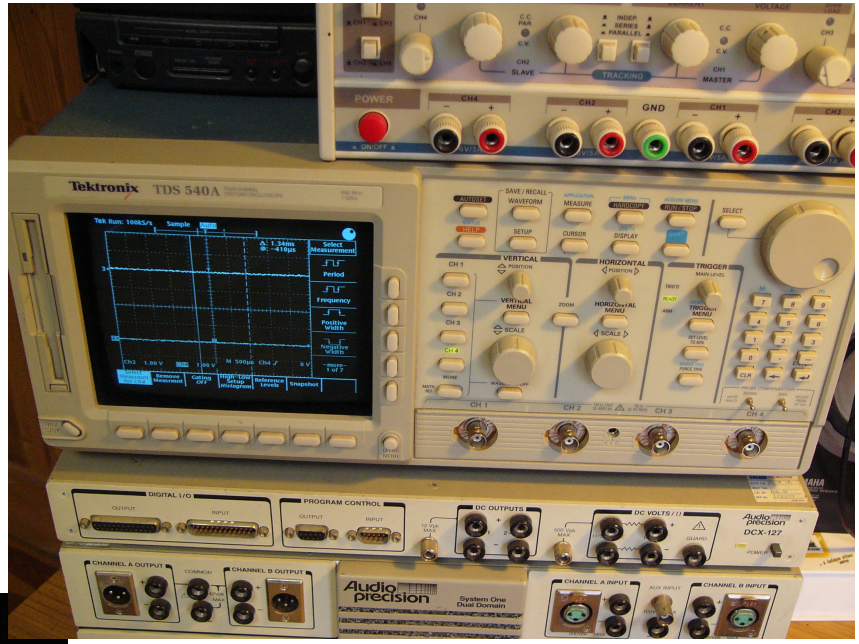


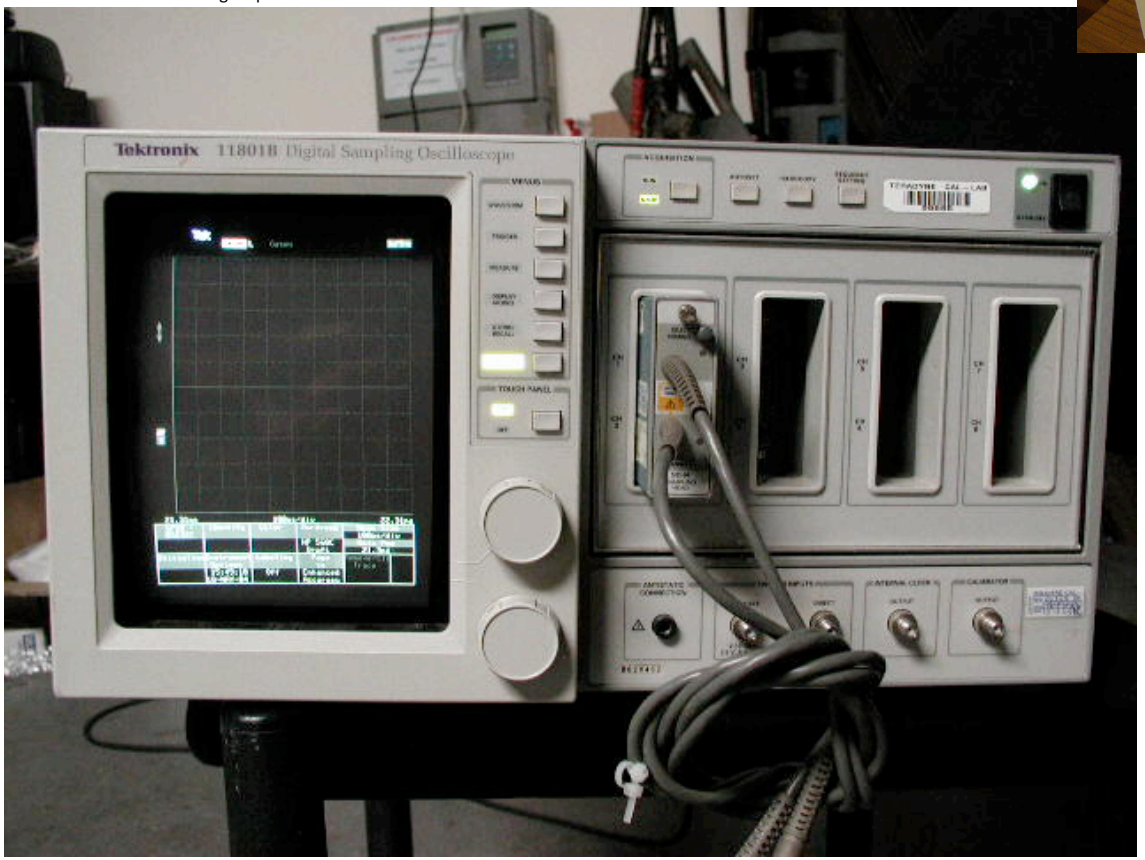
Smalltalk-Based Oscilloscopes

From: http://www1.tek.com/forum/viewtopic.php?f=5&t=5526#p10552
Re: Console port for TDS5/7xxD oscilloscopes

Postby sschnelle on Mon Feb 11, 2013 9:45 am
Example console log i captured from my TDS794D (you can also enter commands on the console, see the 'i' command at the end):

No PCMCIA option board detected.
FLOPPY: Detected
Adding 7131 symbols for standalone.
CPU: 68EC040. Processor #0.
Memory Size: 0x1000000. BSP version 1.0.
Executing Smalltalk
-> Executing Diagnostics from Menu
Start Power-On Diag Sequence





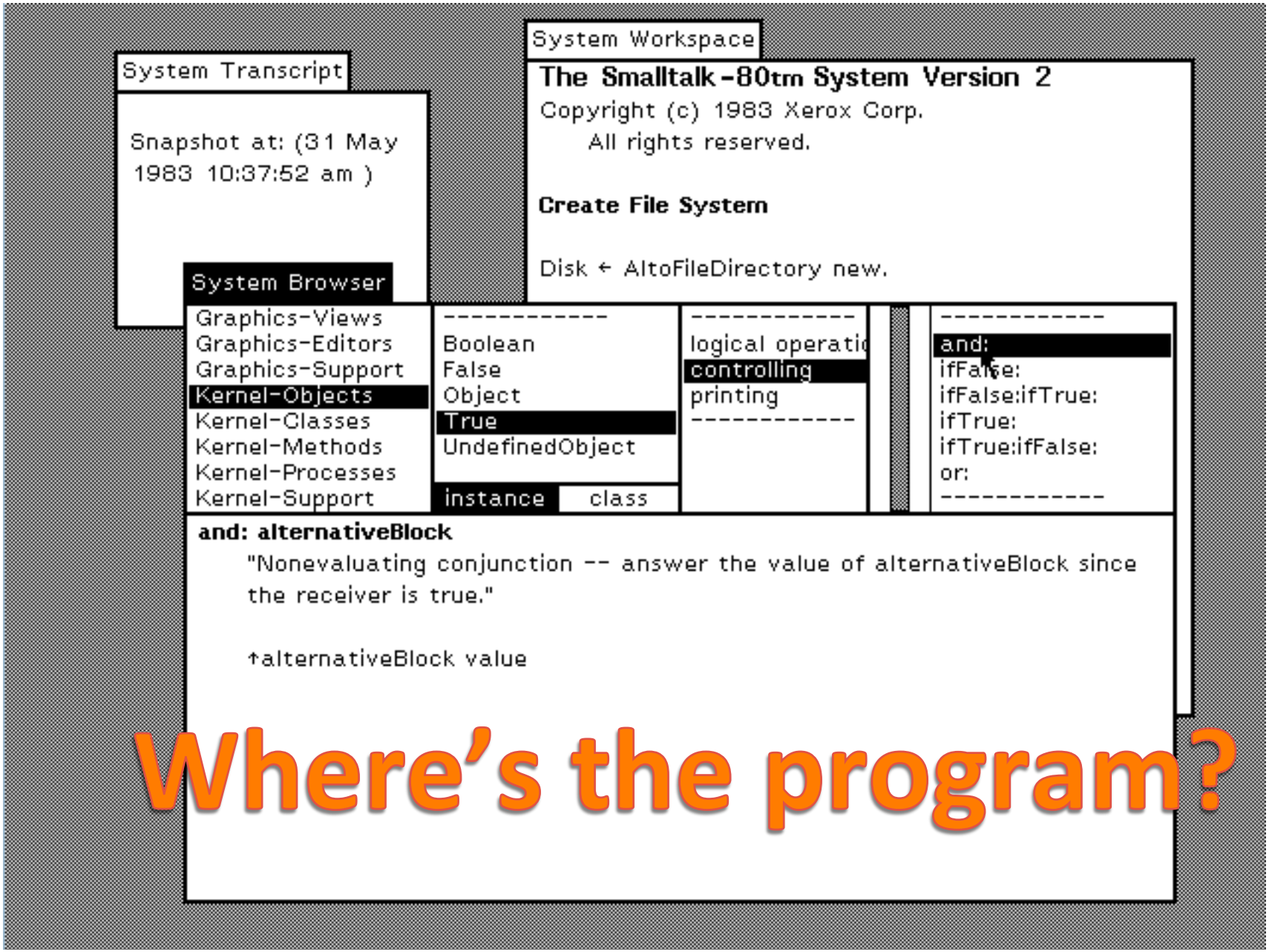
From: http://c2.com/cgi/wiki?TektronixElevenKayScope

The Tektronix Inc 11xxx series of sampling oscilloscopes is an example of a successful embedded deployment of Smalltalk Language. The 11k, as it is commonly known, was a staple of the Tek product line for quite a few years.

The 11k featured a 68000 processor, ample memory, and a little light on the main circuit board that turned on whenever the Garbage Collector ran. It used an embedded Smalltalk environment from OTI, and contrary to many doubters, performance was not an issue. (Unfortunately, the product abandoned the traditional UI model that oscilloscopes have, and got a reputation for being difficult to use.)

The Smalltalk environment was used in several other Tek scopes as well, the 11k was the one which survived the longest.

The Live Programming Experience



Where's the program?

Issues With “File-in”

- Dependencies upon current image state
  - File-in order
  - Conflicts extending system classes
  - No fixed semantics

Saving a “program “as a file-in didn’t guarantee that it could loaded in the future or in somebody else’s image.

Issues With a Virtual Image

- How did it get to its current state?
- Is it reproducible?
- Rolling-back mistakes?
- What if it gets irreparably corrupted?
- How can multiple people work on a project?

Repeatable Deployment?

- How can you reliably, repeatedly, reproducibly deliver a Smalltalk based application
- Managing differing configurations?
- Removing the development tools.
- Maintaining old releases

Organic versus Mechanical

http://2.bp.blogspot.com/-XyStYXcP674/Ul1wUhs5fQI/AAAAAAAAQI0/pd359h_gRNc/s1600/tropical+rainforest.jpg

Modular Smalltalk

An Overview of Modular Smalltalk

Allen Wirfs-Brock
(503) 242-0725

Instantiations, Inc.
1020 SW Taylor St., Suite 200
Portland, OR 97205

Brian Wilkerson
brianw%spt.tek.com@relay.cs.net
(503) 627-3294

P.O. Box 500, Mail Sta. 50-470
Tektronix, Inc.
Beaverton, OR 97077

Abstract

This paper introduces the programming language Modular Smalltalk, a descendant of the Smalltalk-80 programming language. Modular Smalltalk was designed to support teams of software engineers developing production application programs that can run independently of the environment in which they are developed. We first discuss our motivation for designing Modular Smalltalk. Specifically, we examine the properties of Smalltalk-80 that make it inappropriate for our purposes. We then present an overview of the design of Modular Smalltalk, with an emphasis on how it overcomes these weaknesses.

Introduction

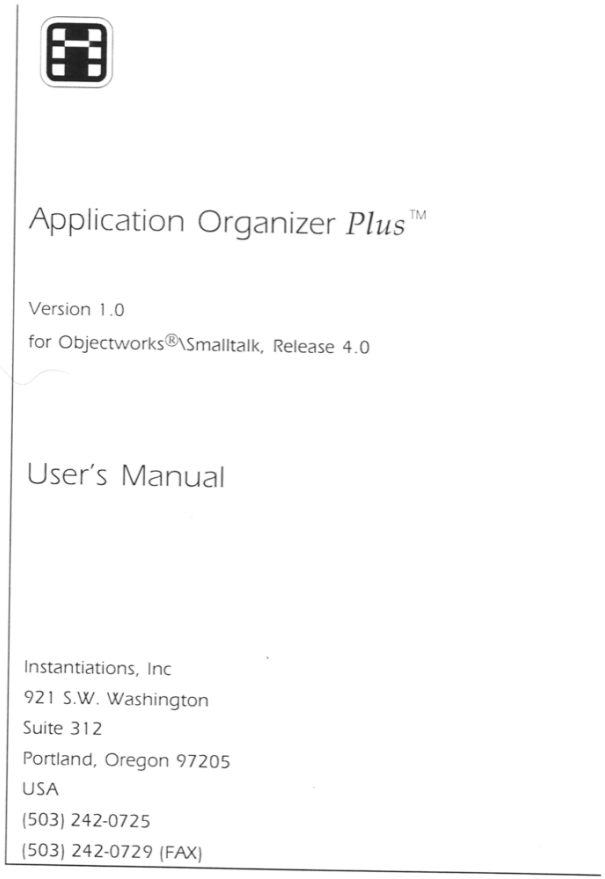
Modular Smalltalk is an evolution of the Smalltalk programming language and system designed to support teams of software engineers developing production application programs that can operate under the control of standard operating systems and display environments.

The Smalltalk programming language and system was originally intended to be the software component of the Dynabook, a portable personal information management tool [Kay77a, Kay77b]. As described by one of its developers, its purpose was "to support children of all ages in the world of information" [Inga78]. Smalltalk is a uniformly object-oriented system which integrates a programming language and its implementation, development tools for the language, a window-oriented user interface manager, and other system software services. The development of Smalltalk was an evolutionary process which took place over an extended period [Inga83]. Its developers typically built a version of the system, experimented with it, and finally used what they learned to build the next version upon the base of the current version. The final result of this process was the Smalltalk-80™ system [Gold83, Gold84].

As Smalltalk became available to a broader group of users, it first found acceptance as a rapid prototyping system. The fact that Smalltalk proved to be an excellent prototyping tool should not be surprising, as Smalltalk's developers had themselves used the system in this manner. However, outside of research laboratories, prototypes are not viewed as ends unto themselves but rather steps on the path towards the development of a final product or solution. The success of prototype applications developed using Smalltalk has led many Smalltalk programmers to look for ways to develop and deliver the final production versions of applications using Smalltalk. These attempts have so far had only limited success.


```
Module 'PlayingCards'
    "This module defines four named objects –
    CardSuits, CardRanks, Card and CardDeck
    – that are used to implement the functionality
    of a deck of playing cards.  Only the class
    CardDeck is exported."

  imports Object from 'Kernel'
  imports List from 'Collections'
  imports UniformDistribution from
  'ProbabilityDistributions'

  CardSuits -> #('heart' 'club' 'diamond' 'spade')
    "The symbol '->' means 'is defined as'."

  CardRanks -> 1 to: 13

  Card -> Class
    refines Object

  instance behavior

    accessing

      variable suit suit: (private)
            "Answer and set the suit of the
      receiver.  The suit should be an element of
      <CardSuits>."

      variable rank rank: (private)
            "Answer and set the rank of the
      receiver.  The rank of jacks, queens and
      kings is 11, 12 and 13, respectively."

      value
        "Answer the face value of the receiver."

        ↑self rank min: 10

    testing

      = aCard
        "Answer <true> if the receiver represents
      the same card as <aCard>."

        ↑self suit = aCard suit
          and: [self rank = aCard rank]
```

From Tektronix to Instantiations



Application Organizer *Plus*™

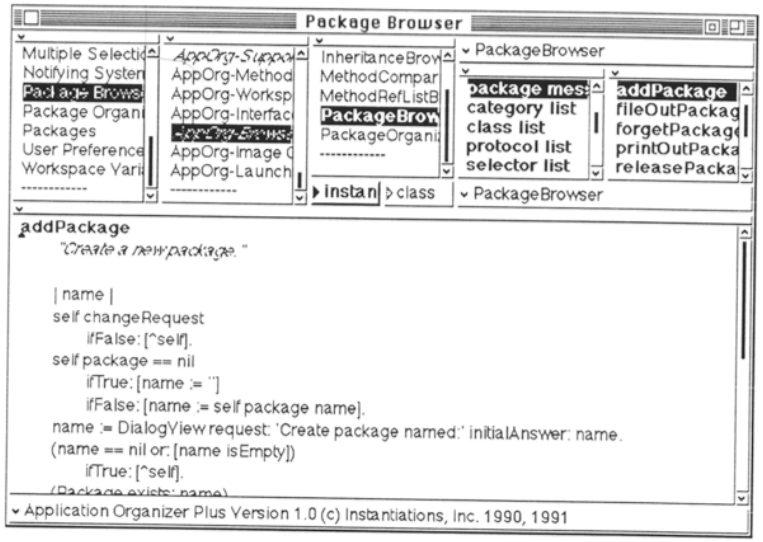
Version 1.0

for Objectworks®\Smalltalk, Release 4.0

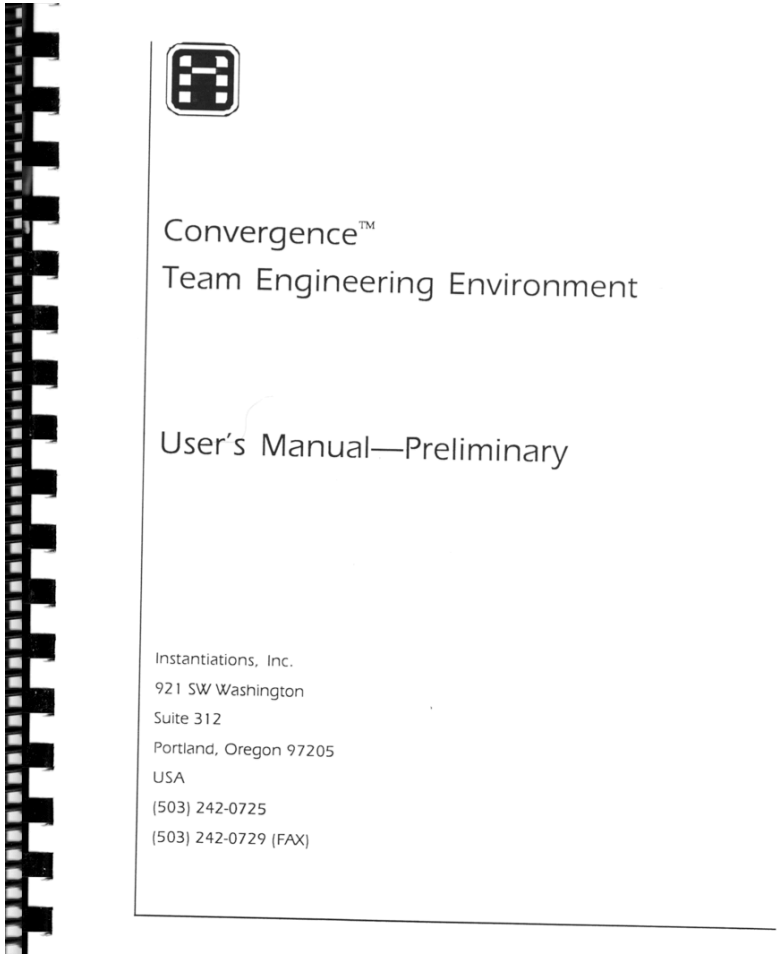
User's Manual

Instantiations, Inc
921 S.W. Washington
Suite 312
Portland, Oregon 97205
USA
(503) 242-0725
(503) 242-0729 (FAX)

- Application Organizer:
  - Single User Tool
  - "file-ins" formalized as "packages"
  - Package Browser supporting on the fly organization of image changes as packages



And then Convergence


Convergence™
Team Engineering Environment

User's Manual—Preliminary

Instantiations, Inc.
921 SW Washington
Suite 312
Portland, Oregon 97205
USA
(503) 242-0725
(503) 242-0729 (FAX)

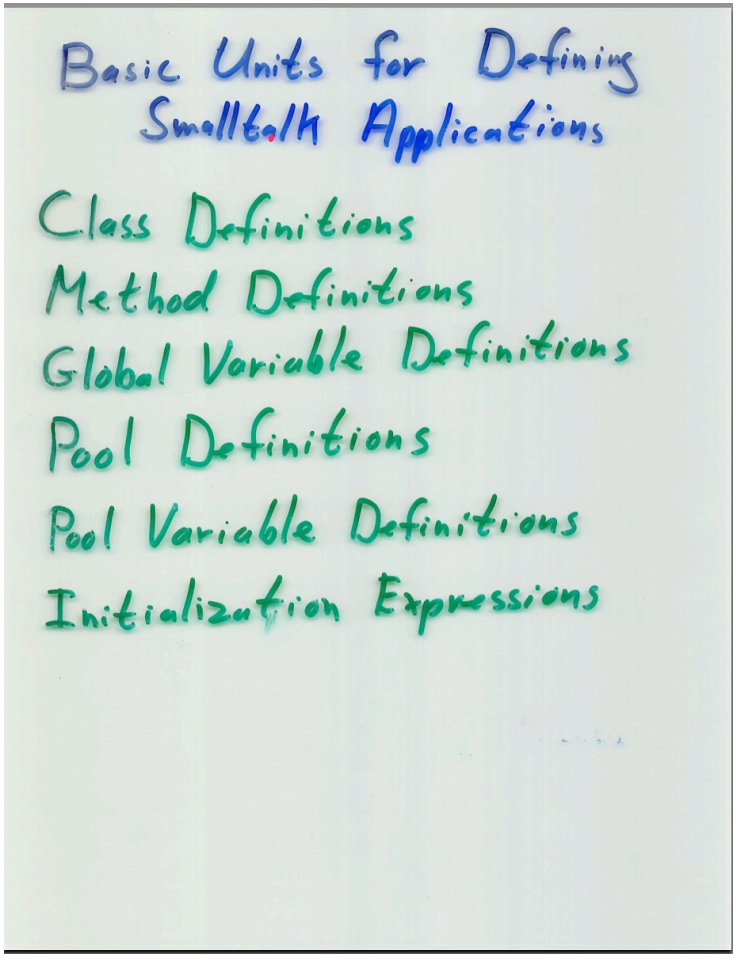
- Packages versioned and stored in multiuser repositories.
- Atomic loading/ unloading of packages
  - With conflict detection
- Tooling support for view and modify unloaded packages

1991, Supporting ParcPlace Objectworks Smalltalk

Declarative Program Structure

Instead of:

```
Point subclass: #Point
        instanceVariableNames: 'x y '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'Graphics-Geometry'!
Point comment:
'Class Point represents an x-y pair of numbers
usually designating a location on the screen." !

Point class
        instanceVariableNames: ''!

!Point methodsFor: 'accessing'!

x
        "Answer the x coordinate."

        ^x!

x: xInteger
        "Set the x coordinate."

        x := xInteger!
```



From Instantiations To Digitalk

Goal

Create an environment that allows organizations to create, deliver, and maintain complex Smalltalk applications over extended periods of time.

Requirements

Support structuring of applications

Help teams coordinate their work

Allow reconstruction from source code

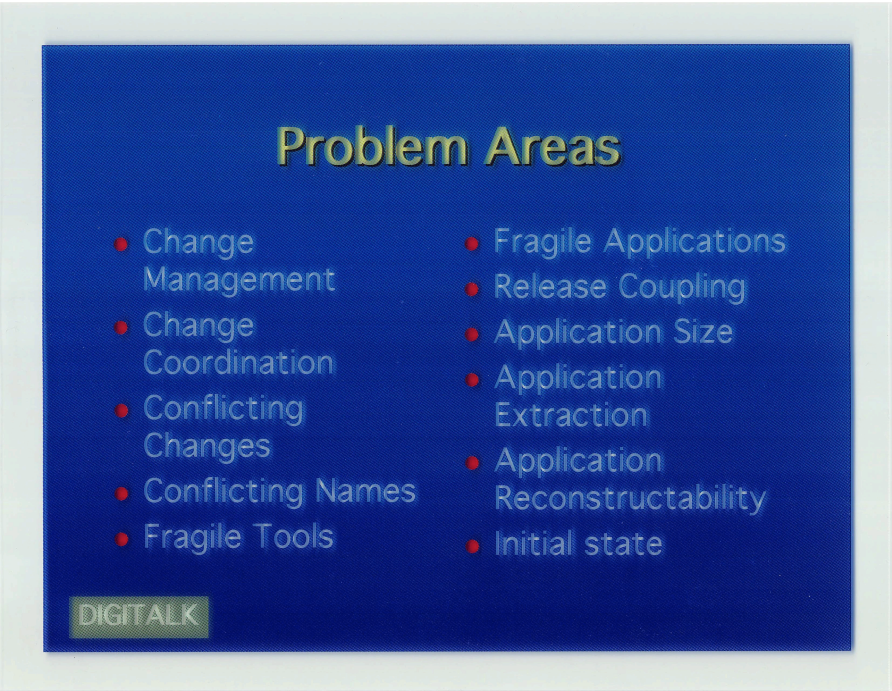
Minimize application dependencies upon base image changes

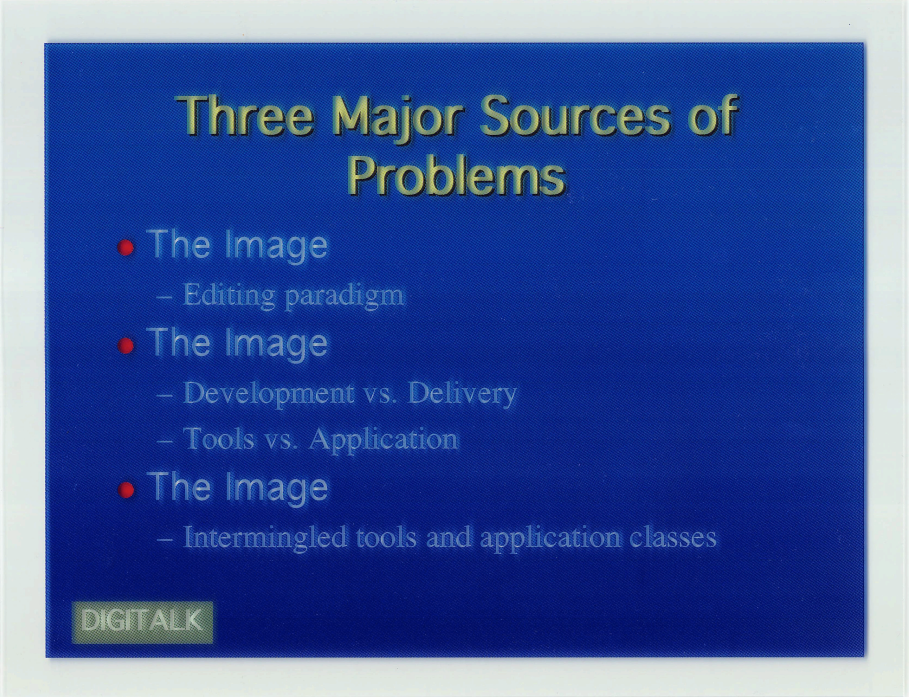
While...

Maintaining a productive, interactive, incremental, "Smalltalk-style" development environment

Change Smalltalk development process from "editing" an image to "defining" an application

Blame it on the Image





Solutions

- Change the conceptual basis of Smalltalk programming from editing an image to defining a program
  - A Smalltalk program should be a formal description of a computational process, not the last image "snapshot" made by a programmer

DIGITALK

The Implementation

- Finish defining the language
  - Change from imperative to declarative program description
  - Formalize all language elements
  - Introduce modularity constructs
  - Define initialization and execution order

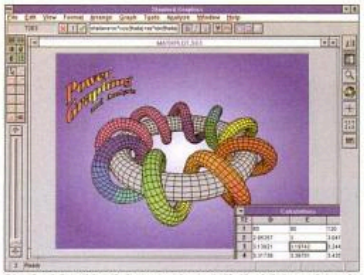
DIGITALK

The Implementation

- Separate execution meta-structures from tool meta-structures
- Define abstraction model (and API) of a Smalltalk program structure
- Eliminate unnecessary reflection

DIGITALK

Digitalk Team/V – Visual Smalltalk



Stanford Graphics 2.1 can analyze statistical data and graphically represent it.

Beta users like speedier Stanford Graphics update

By Kelley Damore

Beta testers lauded the speed improvements in the latest release of 3D Visions' PC graphing package.

"The product has a lot more speed and more user configurable details for auto-processing graphing," said C. Bret Jessee, a manager at Bausch and Lomb, in Rochester, N.Y.

Stanford Graphics 2.1 analyzes statistical data and graphically represents it, unlike programs such as DeltaGraph that can only graph it, Jessee said.

The update also offers rotatable TrueType axis titles, customized graph defaults, and Object Linking and Embedding client and server support.

For Tim Horning, an independent consultant in Omaha, Neb., the context-sensitive menus were very helpful.

"In the previous version, when you selected a frame, you had to go to the top of the menu bar to see your options," Horning said. "With the new version, if you click on the frame, it gives you a floating menu in the middle of the screen that tells you the options. It is instant and quick."

Other features include intelligent redraw, a feature that allows users to make annotations without having to wait for the full screen to redraw.

It also allows any object or graph to be filled with clip art.

Stanford Graphics 2.1 will ship in March for $495. Upgrades will cost $79.95. Users who buy Version 2.0 after January will receive a free upgrade.

3D Visions Corp. is in Torrance, Calif., at (310) 325-1339.

DocuComp integrates with Word

By Steve Polilli

Advanced Software Inc. recently added a feature to its document comparison utility that seamlessly integrates the software tool into a Word for Windows pull-down menu.

The update of DocuComp II, priced at $199.95, includes a special install routine for users of Microsoft Corp.'s Word for Windows. Once the comparison utility is installed it is listed as a Word menu function.

"Users don't need to exit Word, run DocuComp and bring Word back up," said Larry Lightman, president of Advanced Software.

Intended for those who work with words, DocuComp II compares an earlier version of a document with a subsequent version and creates a third composite version with changes noted in three ways.

First, the composite document, marked with line numbers, indicates deletions, insertions, replacements, and moves. A comparison summary report lists the two documents' sizes, dates, lengths, and number of each type of change. A revision list shows each change by page and line number.

DocuComp is a valuable documentation tool for pharmaceutical manufacturers, said Will Andrews, senior technical writer at Abbott Laboratories, in Mountain View, Calif.

"DocuComp has strong reporting capabilities," Andrews said. "In our industry it gets pretty tricky as far as contents of our manuals. There are stringent FDA requirements, and a [documentation] mistake could cost someone their life."

Advanced Software is in Sunnyvale, Calif., at (408) 733-0745.

Development tool introductions heat up ObjectWorld in Boston

By Kelley Damore

BOSTON — Despite the subzero weather, the show floor here at ObjectWorld was ablaze last week with new products.

Companies such as Digitalk Inc., Inference Software Corp., Hewlett-Packard Co., Digital Equipment Corp., and Pure Software Inc. demonstrated object-oriented development tools for the workstation and PC marketplace.

Digitalk Inc. demonstrated a relational database interface for its Parts Assembly and Reuse Tool Set (PARTS) Workbench product. This interface allows users to create graphical front-end applications for a variety of databases including OS/2 Database Manager, DB/2, SQL/DS, and SQL/400.

The company also plans to support other relational databases, including Sybase and Oracle. The product is available now for $995 per server.

Digitalk also showed a PARTS Cobol Wrapper for its PARTS Workbench. Programmers can wrap new or existing Cobol code into a part that can be reused in the Workbench product.

Users can also create graphical links into a Cobol program without having to write any code, Digitalk officials said. The PARTS Cobol Wrapper is available now for $1,995.

The company also demonstrated its programming environment that allows a team of Smalltalk/V users to work together.

Called Team/V for Smalltalk/V for OS/2, the program organizes code into modular units called packages that can be shared among developers. The program includes a package browser that lets a user create and organize definitions, view package structure, and browse classes.

The program also features a definition organizer that lets a user view and reorganize definitions within a package. The program is currently in beta testing and will ship this quarter for $1,495.

Digitalk's Team/V for Smalltalk/V organizes code into units that can be shared among developers.

Inference Corp., maker of expert systems, has entered the client/server market with a development tool called Art-Enterprise. The product includes GUI class libraries, object-oriented programming, and data modeling capabilities. It will be available on Windows, Macintosh, OS/2, Unix, and MVS.

ArtEnterprise is in beta on the Windows platform and will ship in September, company officials said. Other platforms will be available by year end. The product is priced at $6,995 for the Windows, OS/2, and Macintosh platforms and $9,995 for the Unix platform.

Hewlett-Packard Co. announced HP Distributed Smalltalk, an implementation of the Object Management Group's COBRA specification.

The program is based on the Smalltalk programming language and lets users simultaneously develop object-oriented applications. These objects can link to information stored anywhere in the enterprise using HP's OpenODB. Pricing and ship date have not been set.

From Pure Software Inc. comes a run-time detection tool for C and C++ Unix developers that eliminates run-time errors, memory access errors, and memory leaks. The product is priced at $4,000 per floating network license, according to the company.

Digital Equipment Corp. announced that it will offer its Common Object Request Broker Software on IBM's AIX, HP's HP-UX, Apple's System 7, and its own OpenVMS.

Easel offers client/server Workbench

By Ed Scannell

To ride the client/server wave, Easel Corp. is offering users a version of its Workbench tools that lets them build client/server applications to access corporatewide data.

Version 2.0 of Easel Workbench features an integrated set of object-based tools that enable developers to build a more capable set of corporatewide solutions.

The new program supports several client/server architectures, such as Windows, OS/2, and DOS-based systems.

"We think the product is unique in that it supports a range of client/server architectures from database server, transaction processing, and peer-to-peer projects to distributed presentation applications, including the [PC] renovation of mainframe applications," said Doug Kahn, Easel president and CEO.

Workbench 2.0's integrated set of tools includes a layout editor, an attribute editor, text and vector-drawing editors, a source-level debugger, and an incremental compiler.

The program's improved interface makes it easier to access program components and development tools, a representative said.

The source-level debugger lets developers visually monitor and control the execution of applications within the development environment. Developers can now set break points, step through their code, and examine and change variable values.

Version 2.0 lets developers select tools, objects, and source-code modules from the program's object-based Parts Catalog. This allows them to "snap" together components of a client/server application more quickly.

The program's WYSIWYG menu editor makes it possible for developers to construct menu and action bars visually.

Another benefit of the new version is that all compiling can now be done in the background, letting developers engage in another task during large compiles or recompiles.

Available now, Easel Workbench 2.0 comes in two versions: a SQL Edition for creating advanced SQL access applications and the Corporate Edition, which includes the SQL Edition plus other client/server options including peer-to-peer communications.

The Easel Workbench SQL Editions for Windows and OS/2 are priced at $3,995 and $5,995, respectively.

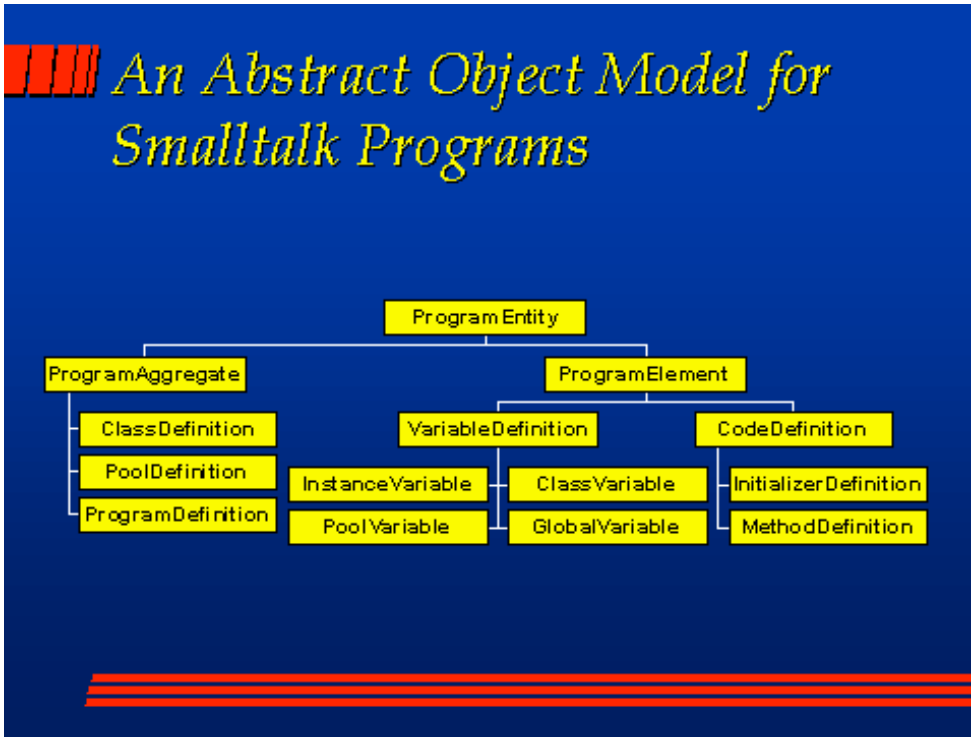
Easel Workbench Corporate Editions for DOS, Windows, and OS/2 are $7,900, $9,900, and $10,900, respectively.

Easel of Burlington, Mass., can be reached at (617) 221-2100.

Team/V Advances

- Clusters – Configuration Management
  - Hierarchical grouping of packages
  - Purely structural or Strictly Versioned Configurations
- Package Migrator
  - Atomic load/unload packages/clusters
  - Migrate forward/backwards between package/cluster versions in a running image
- Complete "program model" and tool API.
  - Object model of full declarative structure of packages
  - Tool API completely independent of reflection API
  - Browsing/editing of "unloaded" packages

Declarative Program Model;



In other words: A full AST

Evolved into the ANSI Smalltalk Declarative Definition of the Smalltalk language.

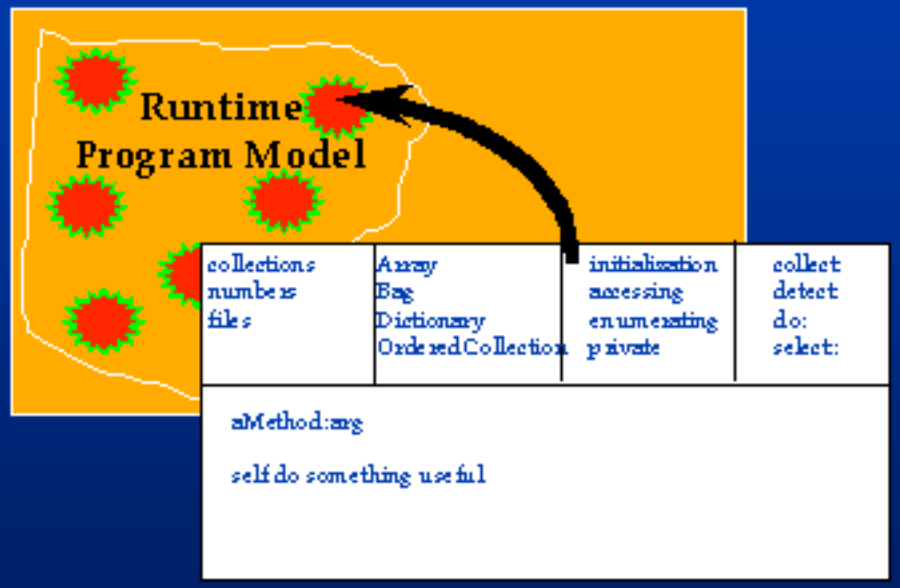
Reflection versus the Declarative Model

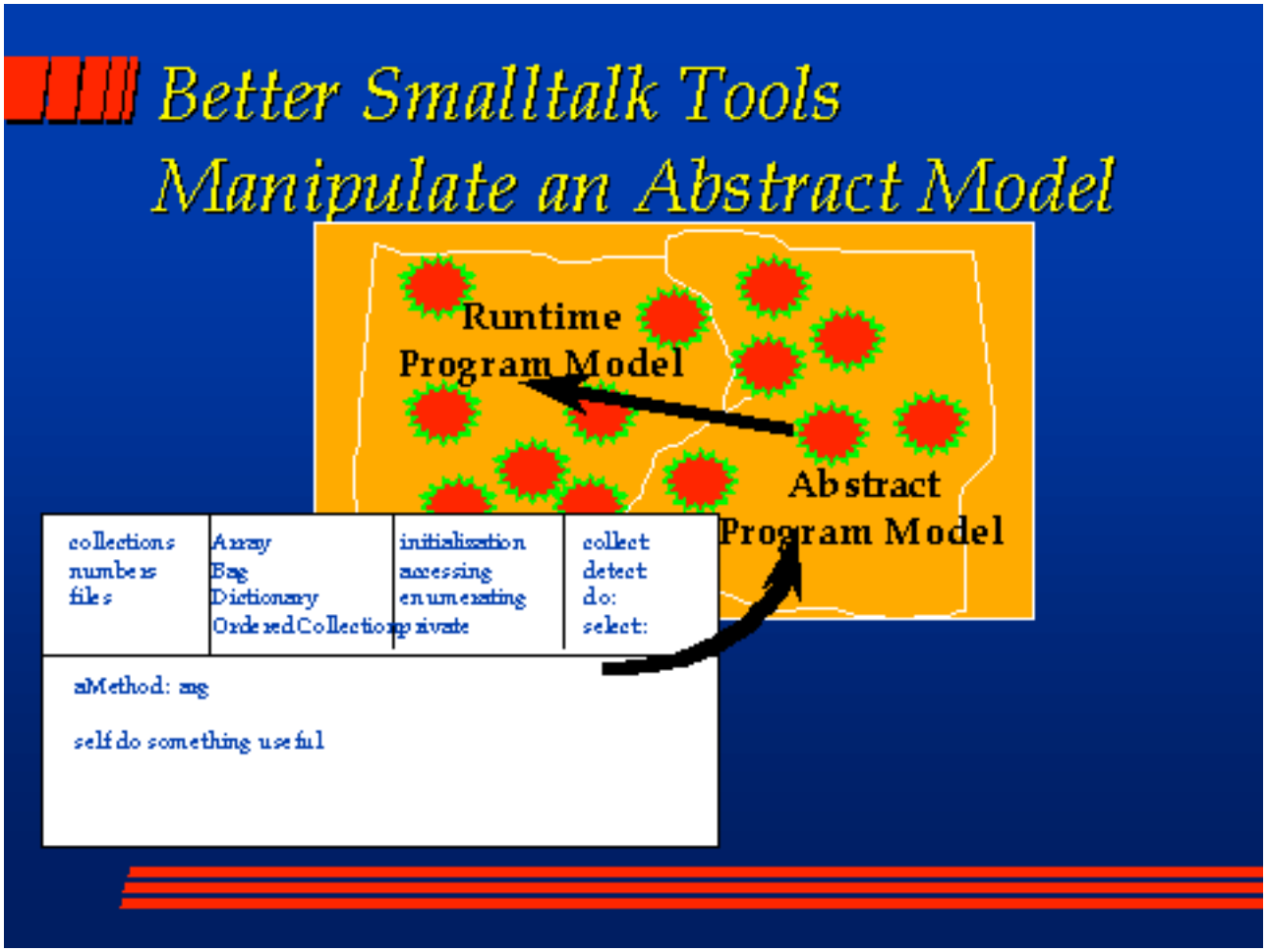
- The declarative model statically describes a program prior to execution
- Reflection occurs dynamically during program execution
- The declarative model neither requires nor precludes reflection

Reflection: Doing Better

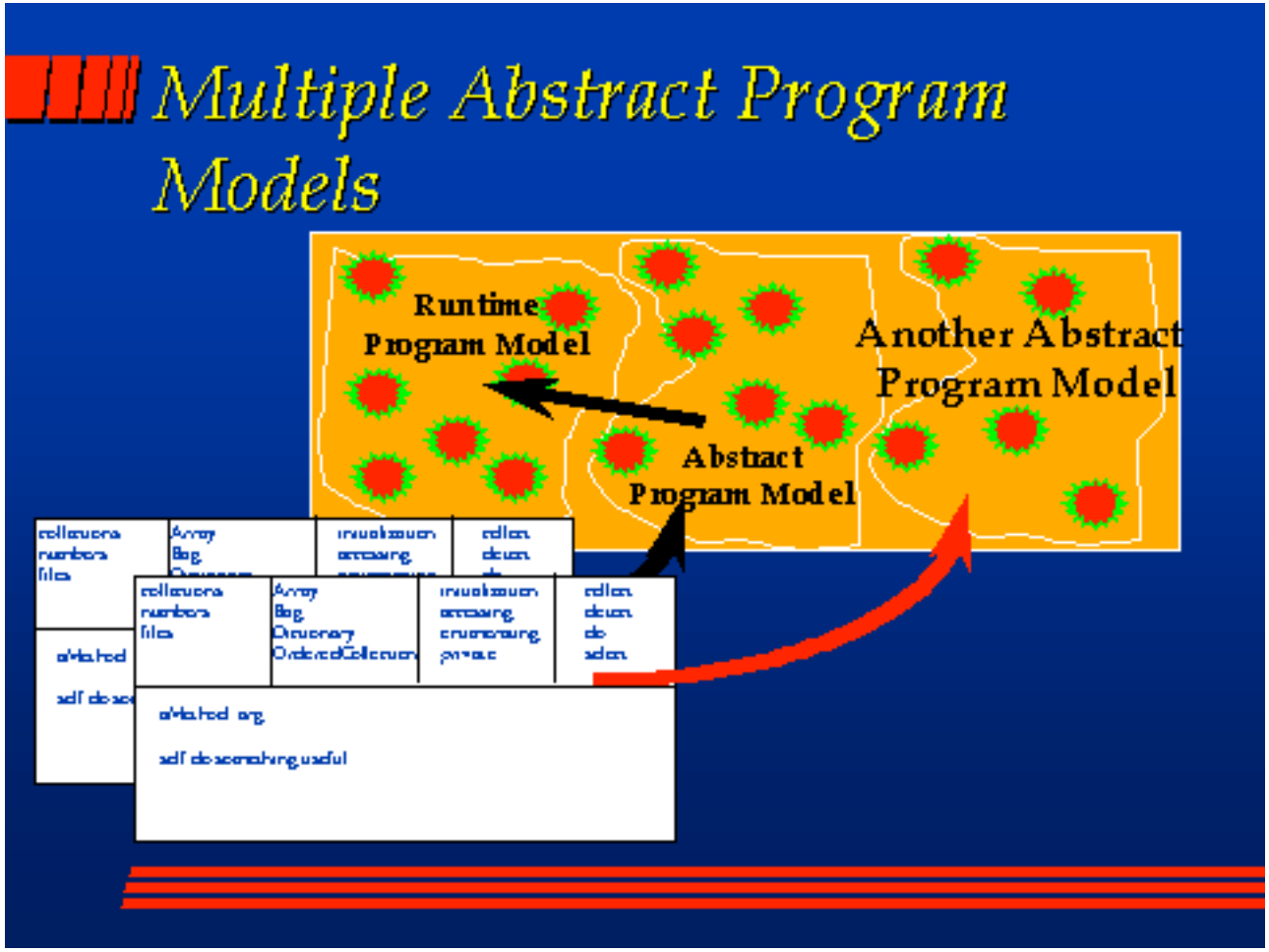
- Traditional Smalltalk reflection is inherently implementation dependent
- Why not objectify the abstract declarative description of a Smalltalk program?

Traditional Smalltalk Tools Manipulate the Runtime Model

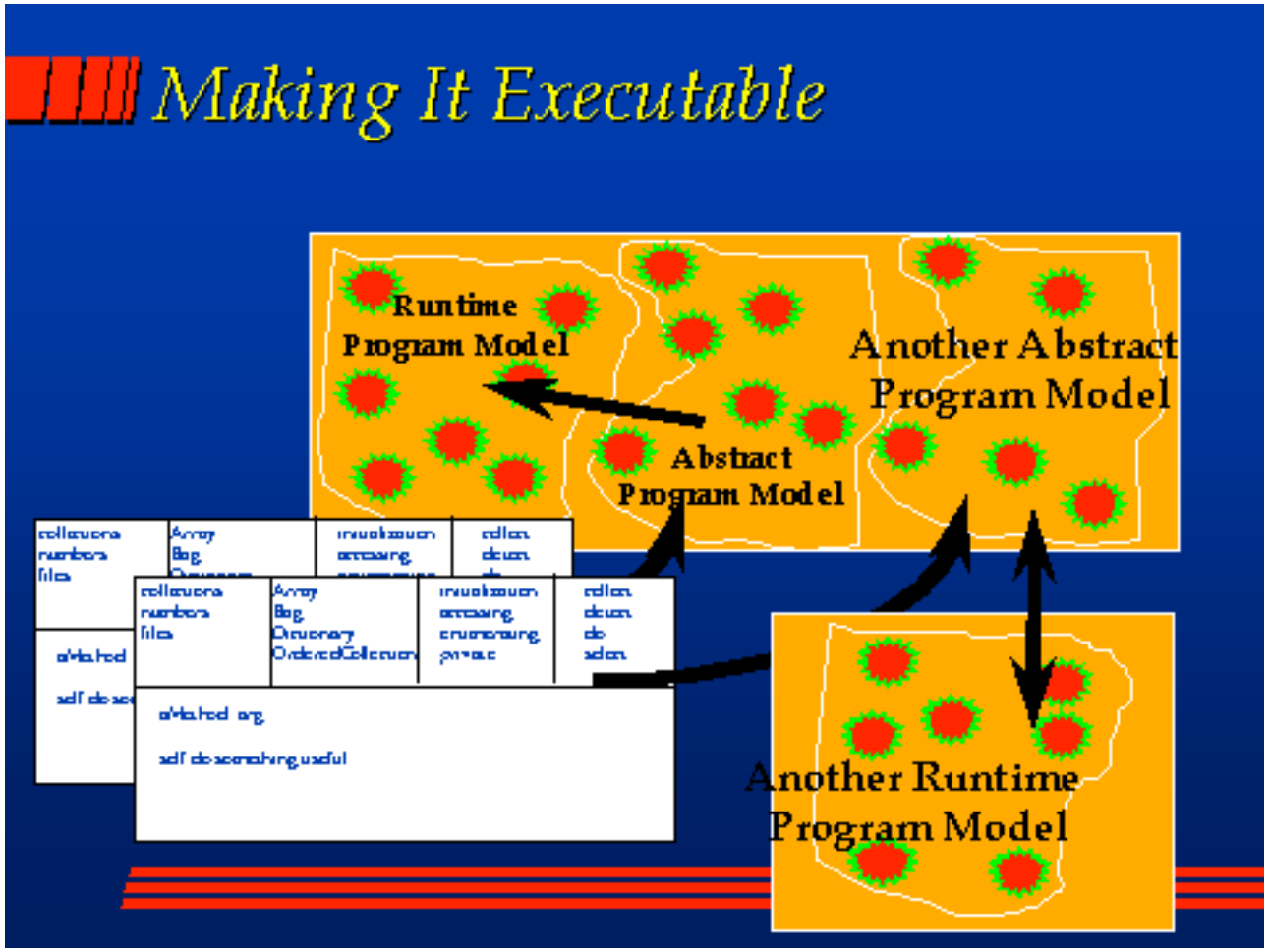




Today we'd say:
The objects of the program model act as mirrors on the runtime objects.

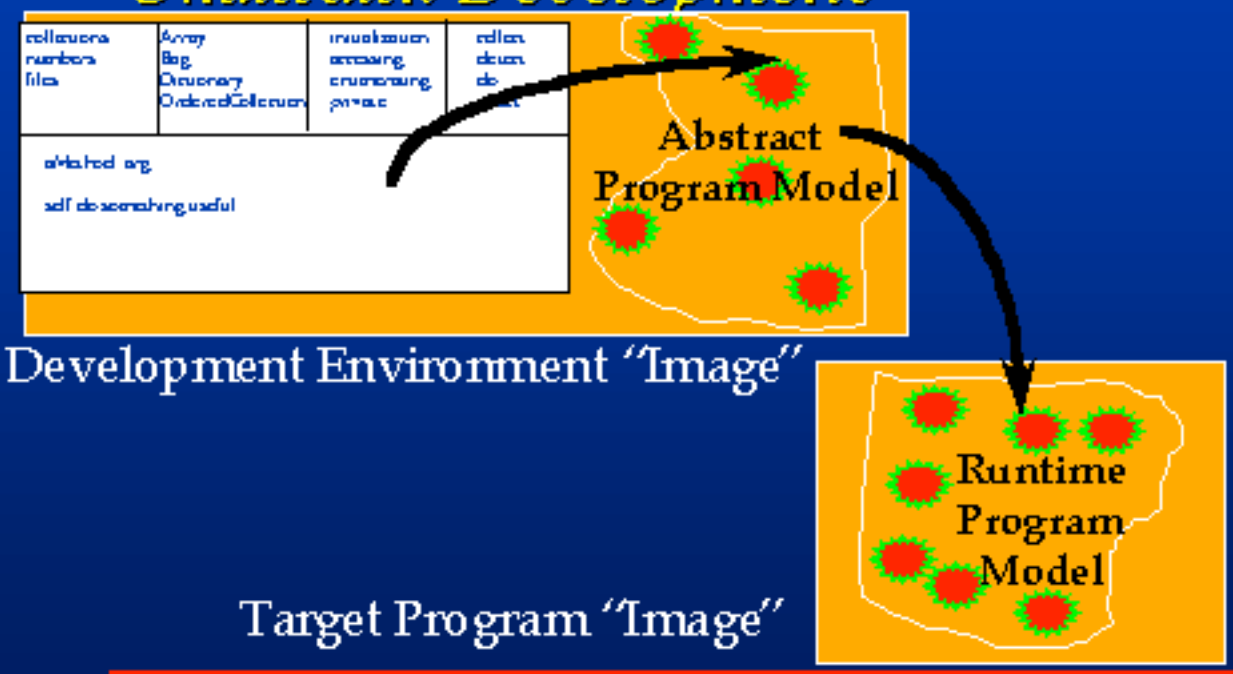


Models of multiple programs could coexist in a single image.



The mirrored runtime objects didn’t have to be in the same image.

A New Architecture for Smalltalk Development



Development Environment "Image"

Target Program "Image"

New Architecture Characteristics

- Users construct a declarative definition of a Smalltalk program instead of editing an image.
  - Programs are completely specified
  - Reproducible from source code
  - Invalid programs are editable
  - No "stripping" required for delivery

New Architecture Characteristics

- Target program class library is separate and distinct from the implementation of the development environment
  - Target program changes do not impact development tools.
  - Development tool changes do not impact target program
  - Release decoupling

New Architecture Characteristics

- Fully incremental, interactive program creation, testing, and debugging
- Target program failure will not crash development environment
- Simplified Class library

Firewall

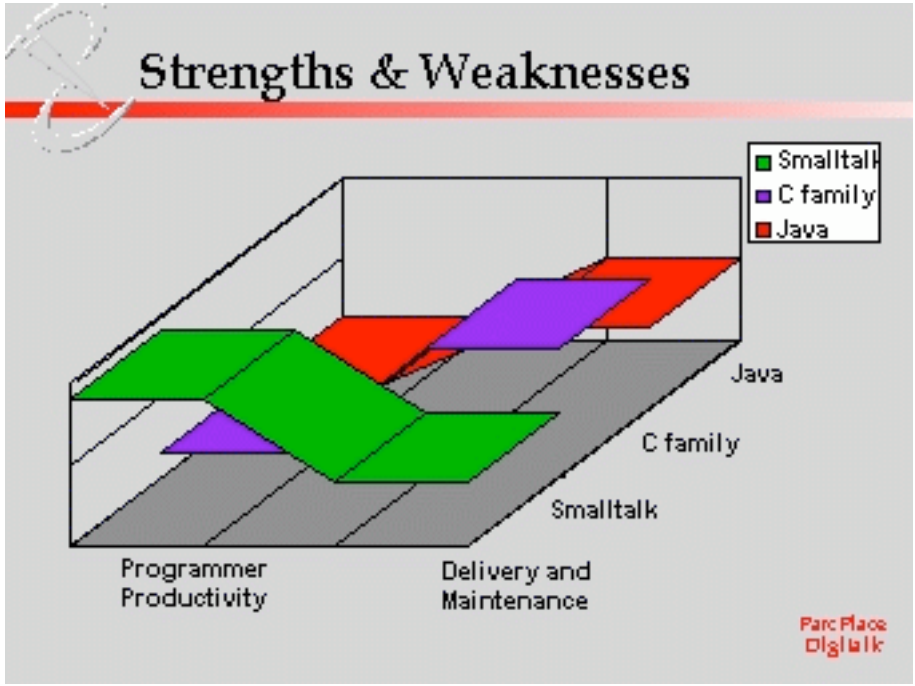
Is the Architecture Feasible?

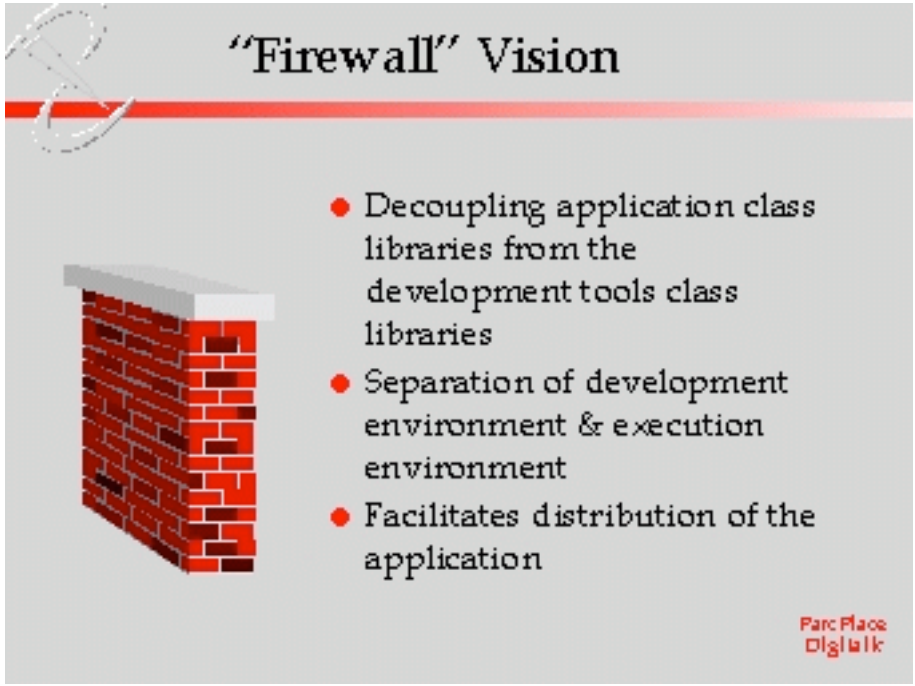
- "Firewall" prototype operational
  - Target program fully decoupled from development tools
  - Target program executes in separate process
  - Full incremental programming and debugging

"Firewall" Accomplishments

- Very small application program images
  - "3+4" image < 10K
  - Utilities & applets 30K - 200k
  - Full GUI Applications 500k-2m
- First complete regeneration of a "Xerox Smalltalk" system from source code since 1976

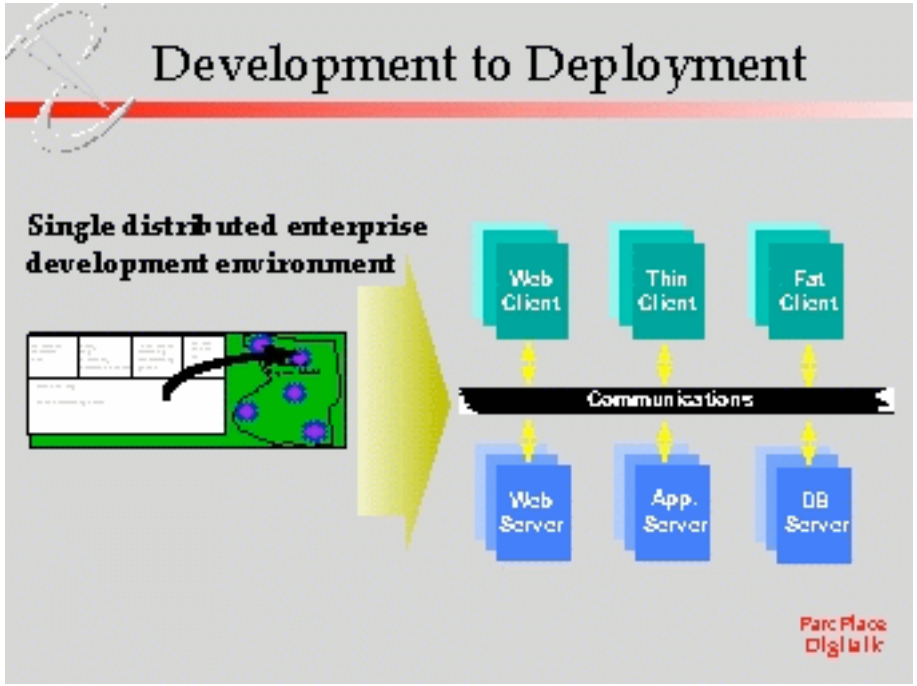
From Digitalk to ParcPlace-Digitalk

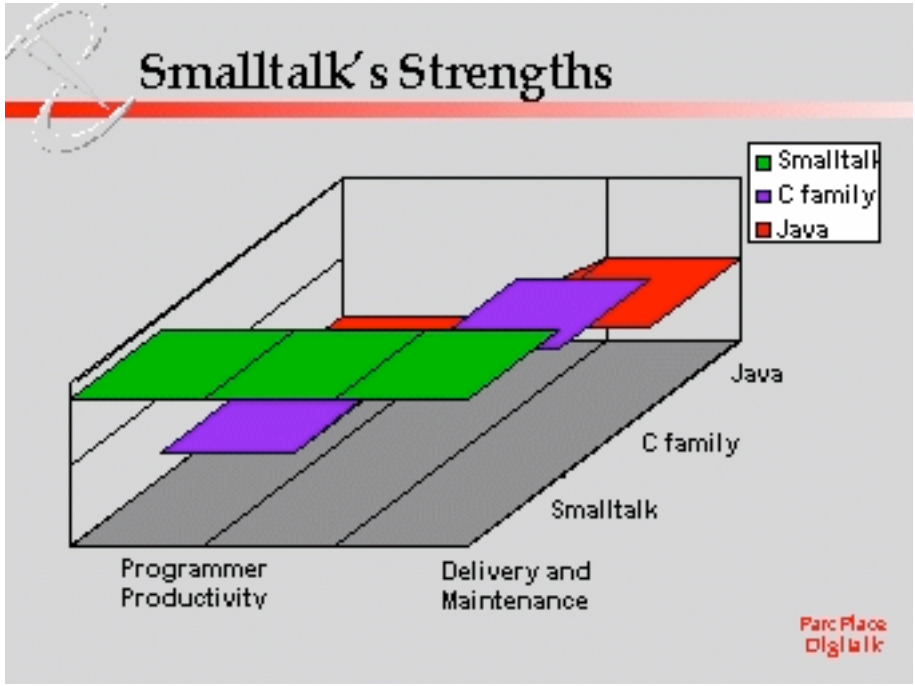


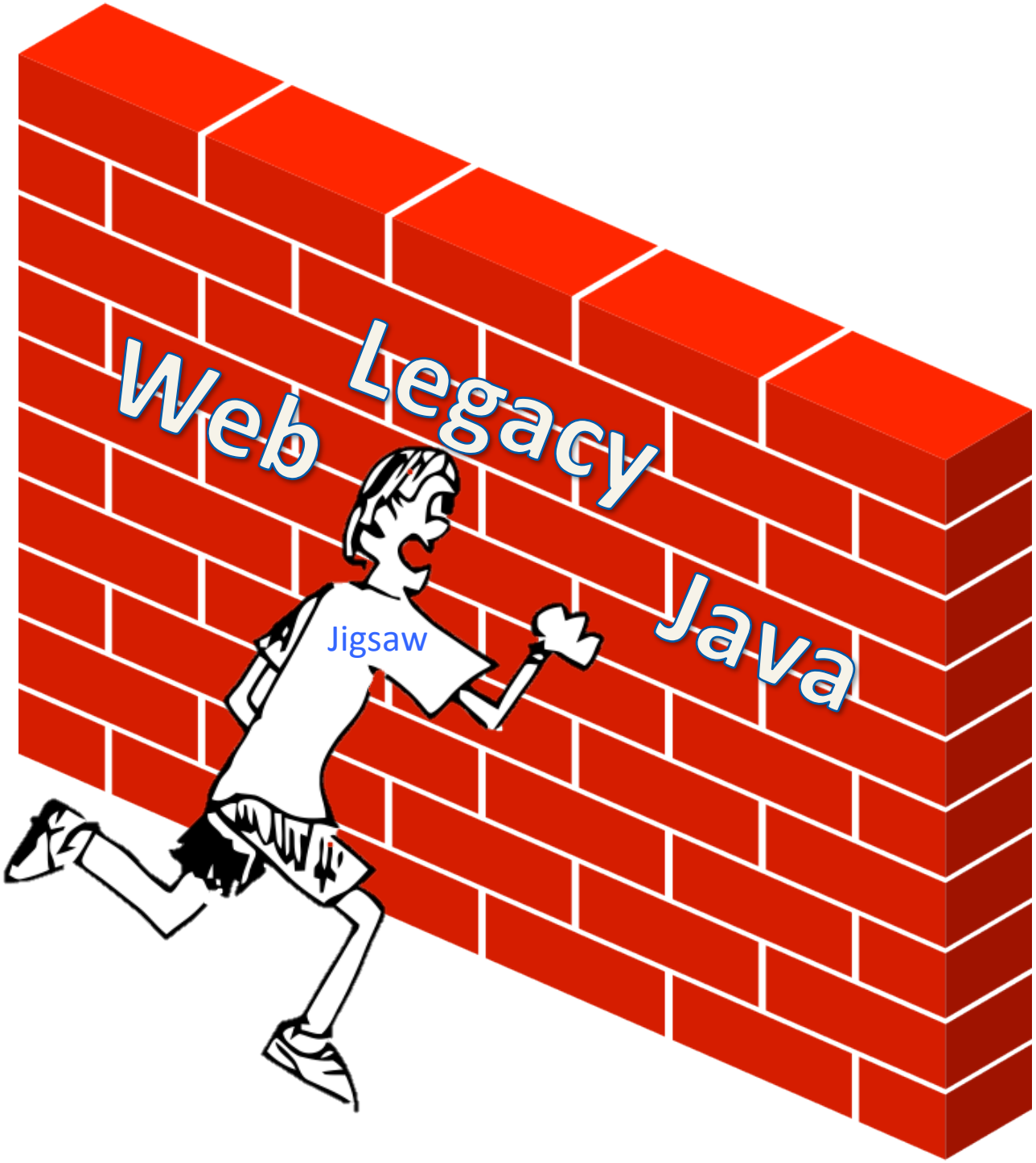


Jigsaw

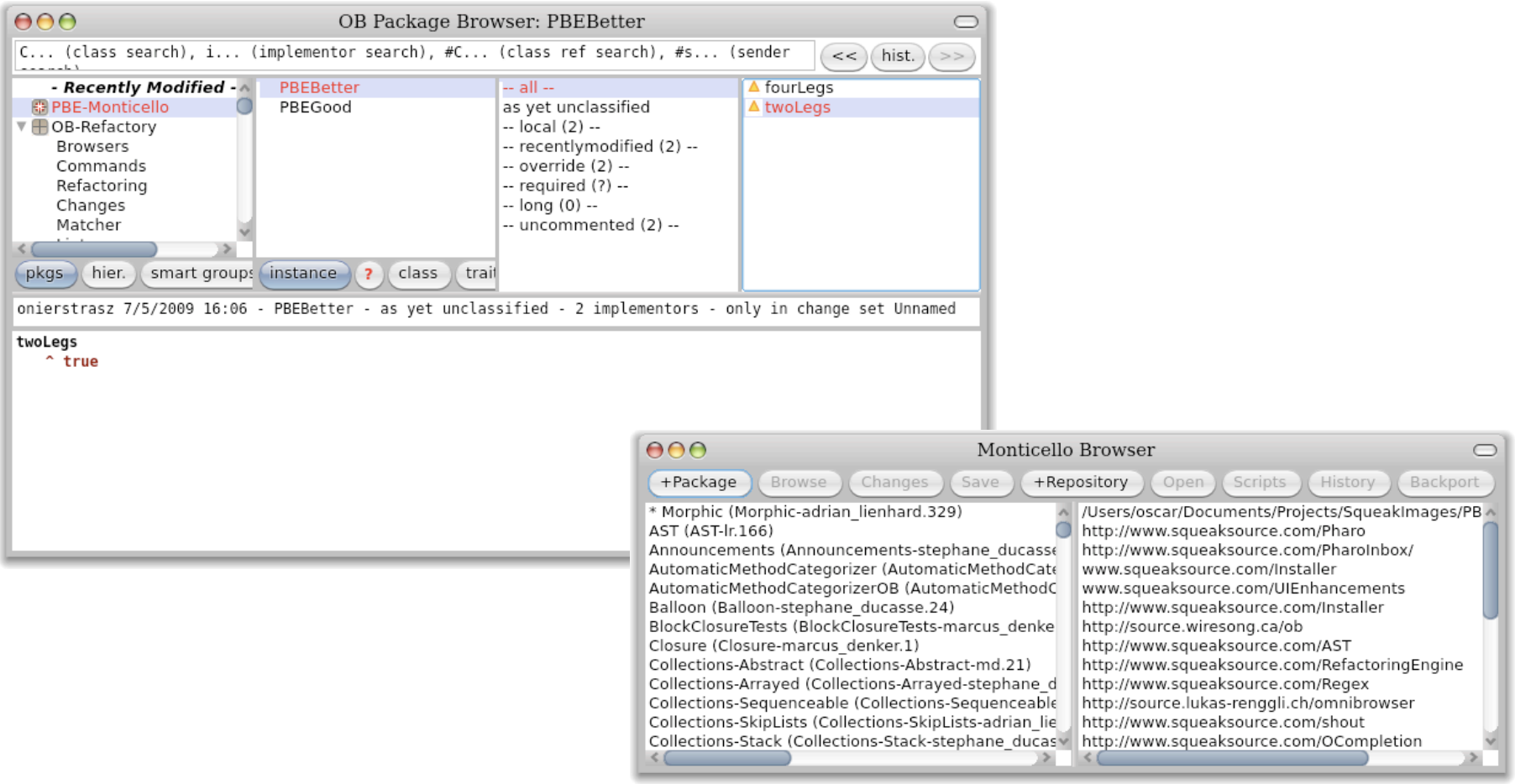
Jigsaw Development Vision





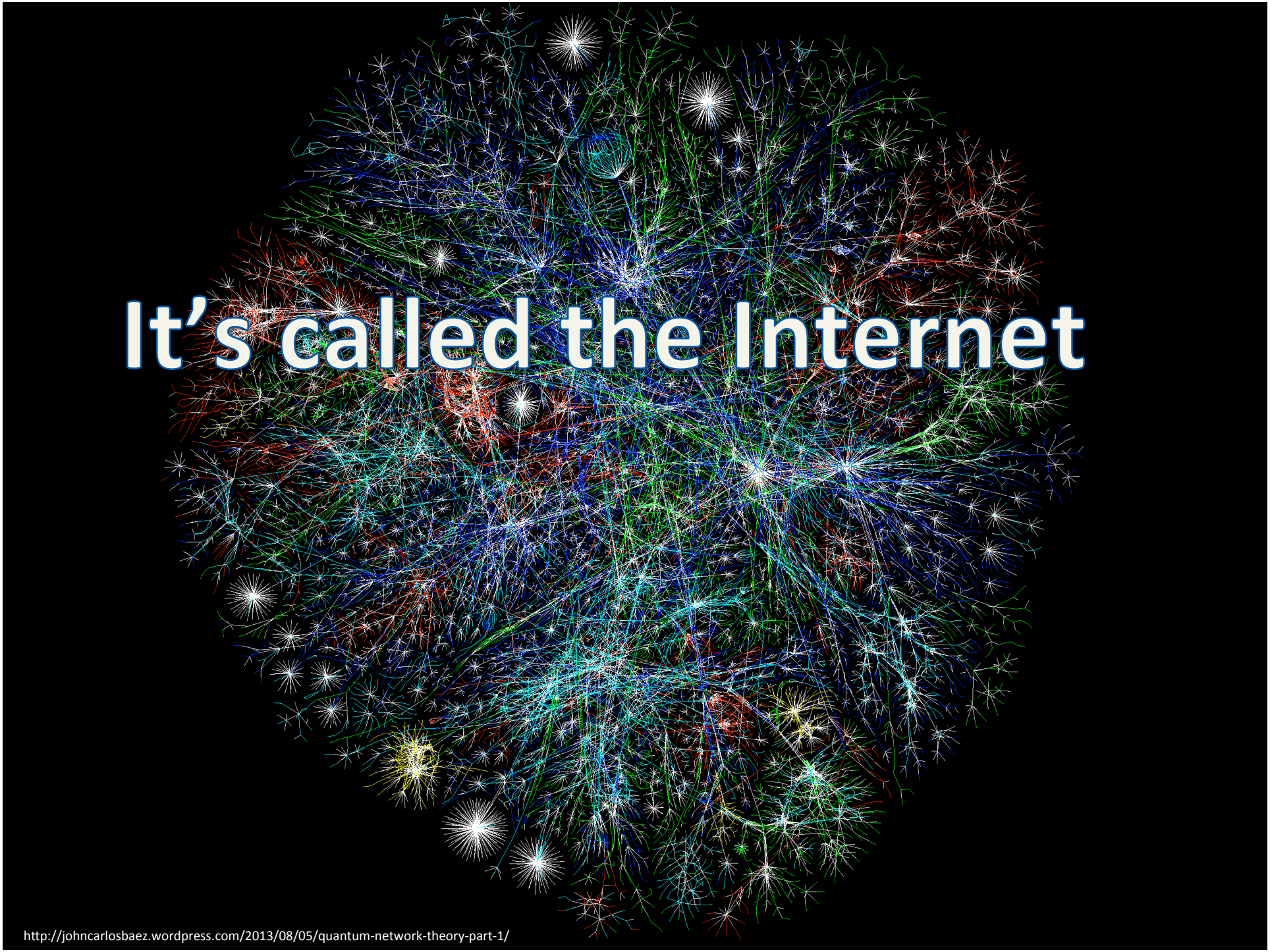

Web
Legacy
Java
Jigsaw

But ideas live on



Every day millions of programmer sit down in from of their computers, and start writing code within a dynamic, persistent, stateful, non-restartable, object-based computing environment.

It’s not a Smalltalk virtual image, but it’s very similar to a massively scaled up one.



http://johncarlosbaez.wordpress.com/2013/08/05/quantum-network-theory-part-1/

Think of the Internet as a global spanning virtual image

- It consists of transient and persistent objects exchanging messages
- There is no syntax for describing's its macro structures
- Object-level behaviors a defined using textual programming
- It’s live programmed.
- It can’t be restarted
- It can’t be reconstructed from source
- Buggy programs leave broken objects lying around

Smalltalkers understand virtual images

Think bigger –use your Smalltalk experience to create great programming experiences for the ambient web