

## Editors

John Pugh and Paul White  
Carleton University & The Object People

## SIGS Publications Advisory Board

Tom Atwood, Object Design  
François Bancelhon, O<sub>2</sub> Technologies  
Grady Booch, Rational  
George Bosworth, Digitalk  
Jesse Michael Chonoles, ACC of Martin Marietta  
Adele Goldberg, ParcPlace Systems  
Tom Love  
Bertrand Meyer, ISE  
Melir Page-Jones, Wayland Systems  
Cliff Reeves, IBM  
Bjarne Stroustrup, AT&T Bell Labs  
Dave Thomas, Object Technology International

## THE SMALLTALK REPORT Editorial Board

Jim Anderson, Digitalk  
Adele Goldberg, ParcPlace Systems  
Reed Phillips, Knowledge Systems Corp.  
Mike Taylor, Digitalk  
Dave Thomas, Object Technology International

## Columnists

Kent Beck, First Class Software  
Juanita Ewing, Digitalk  
Greg Hendley, Knowledge Systems Corp.  
Tim Howard, RothWell International  
Ed Klimas, Linea Engineering Inc.  
Alan Knight, The Object People  
William Kohl, RothWell International  
Mark Lorenz, Hatteras Software, Inc.  
Eric Smith, Knowledge Systems Corp.  
Rebecca Wirfs-Brock, Digitalk

## SIGS PUBLICATIONS GROUP, INC.

Richard P. Friedman, Founder & Group Publisher

## Editorial/Production

Kristina Joukhadar, Editorial Director  
Elisa Varian, Production Manager  
Brian Sieber, Art Director  
Seth J. Bookey, Production Editor  
Margaret Conti, Advertising Production Coordinator  
Dan Olawski, Editorial Production Assistant

## Circulation

Bruce Shriver, Jr., Circulation Director  
John R. Wengler, Circulation Manager  
Kim Maureen Penney, Circulation Analyst

## Advertising/Marketing

Gary Purdie, Advertising Manager, East Coast/Canada/Europe  
Jeff Smith, Advertising Manager, Central U.S.  
Michael W. Pack, Advertising Representative  
212.242.7447 (v), 212.242.7574 (f)  
Kristine Vikanins, Advertising & Exhibit Sales  
212.242.7447 (v), 212.242.7574 (f)  
Sales Representative: Diane Fuller & Associates, West Coast  
408.255.2991 (v), 408.255.2992 (f)  
Sarah Hamilton, Manager of Promotions and Research  
Caren Palmer, Senior Graphic Designer

## Administration

Margherita R. Monck, General Manager  
David Chatterpaul, Senior Accounting Manager  
James Amenuvor, Business Manager  
Michele Watkins, Assistant to the Publisher

**SIGS**  
PUBLICATIONS

Publishers of JOURNAL OF OBJECT-ORIENTED PROGRAMMING, OBJECT MAGAZINE, C++ REPORT, SMALLTALK REPORT, THE X JOURNAL, REPORT ON OBJECT ANALYSIS & DESIGN, OBJECTS IN EUROPE, DIRECTORY OF OBJECT TECHNOLOGY, and OBJECT SPEKTRUM (GERMANY)

## Features

### ENVY software baselining process 4

*Barry Oglesby*

Baselines act to control versions of software released to external entities and to synchronize Classes, Applications, and Configurations between developers so that the most up-to-date versions of these are readily available to all.

### A technical overview of VisualWorks 2.0 9

*Jim Haungs*

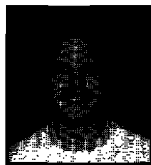
VisualWorks continues to improve as a GUI-building tool, and most of the serious drawbacks have been addressed.

### Making MVC code more reusable 15

*Bobby Woolf*

Two of the most popular advantages of Smalltalk are the ability to implement code that is highly reusable and the ability to easily produce GUIs. Ironically, Smalltalk code for producing GUIs has been very difficult to reuse.

## Columns



### Smalltalk Idioms Demand loading for VisualWorks 19

*Kent Beck*

A complimentary approach to reducing memory footprint is described—demand loading. Rather than discarding objects that may not be used, demand loading waits until a value is wanted, then loads it from disk.



### The best of comp.lang.smalltalk Safety and inheritance 24

*Alan Knight*

In theory computer systems can be reliable, but all software has bugs. In pursuit of the unattainable goal of perfect reliability, we too often fail to consider the consequences when systems do fail.



### Getting Real Multi-user Smalltalk 27

*Jay Almarode*

A new model of application development is emerging where the domain of objects is persistent and accessible to multiple users.

## Departments

### Editors' Corner 2

### Product Announcements 31

### Smalltalk Solutions Special Conference Preview Section 33

### Recruitment 45

The Smalltalk Report (ISSN# 1058-7978) is published 9 times a year, monthly except in Mar–Apr, July–Aug, and Nov–Dec. Published by SIGS Publications Inc., 71 West 23rd St., 3rd Floor, New York, NY 10010. © Copyright 1995 by SIGS Publications. All rights reserved. Reproduction of this material by electronic transmission, Xerox or any other method will be treated as a willful violation of the US Copyright Law and is flatly prohibited. Material may be reproduced with express permission from the publisher. Mailed First Class. Canada Post International Publications Mail Product Sales Agreement No. 290386.

Individual Subscription rates 1 year (8 issues): domestic \$79; Mexico and Canada \$104, Foreign \$119; Institutional/Library rates: domestic \$206, Canada & Mexico \$250, Foreign \$286. To submit articles, please send electronic files on disk to the Editors at 885 Meadowlands Drive #509, Ottawa, Ontario K2C 3N2, Canada, or via Internet to streport@objectpeople.on.ca. Preferred formats for figures are Mac or DOS EPS, TIF, or GIF formats. Always send a paper copy of your manuscript, including camera-ready copies of your figures (laser output is fine).

POSTMASTER: Send address changes and subscription orders to: The Smalltalk Report, P.O. Box 2027, Langhorne, PA 19047. For service on current subscriptions call 215.785.5988, 215.785.8073 (fax), P00978@pallink.com (email). PRINTED IN THE UNITED STATES.

# Editors' Corner

**B**y the time you read this issue, we hope you will be making your travel plans to attend Smalltalk Solutions '95 from February 21-24, 1994, in New York City. Over the past few months, we have been working hard to put together a program befitting the inaugural major vendor-independent Smalltalk conference. In addition to keynote presentations from Dave Thomas, Tom Atwood, and Ray Wells, the technical tracks feature sessions presented by such recognized Smalltalk experts as Kent Beck, Rebecca Wirfs-Brock, Sam Adams, Wilf LaLonde, and Amarjeet Garewal. With topics such as design patterns, performance, metrics, metalevel programming, client-server and distributed systems, there will be plenty to interest even the most experienced of Smalltalk programmers. There is always much to be learned from the experiences of others and with this in mind, Smalltalk Solutions will feature corporate case studies from organizations representing a variety of application domains; e.g., Texas Instruments, Caterpillar, CIGNA, and the Canadian Imperial Bank of Commerce. A dedicated track for managers will address the issues of managing and delivering large-scale Smalltalk projects.

We hope you to see you all in New York.

A common thread on the Smalltalk bulletin boards in recent months has been discussion over the changing focus of Smalltalk and its increased use by the MIS community. A refrain that has been repeated numerous times and which comes mainly from hard-core Smalltalkers (or so-called "greybeards") with many years of experience is that the Smalltalk vendors are positioning themselves as competitors to PowerBuilder and Visual Basic. Well, at least this makes a change from C++. Many of these Smalltalkers cut their Smalltalk teeth developing scientific and engineering applications in areas such as simulation, and increasingly feel deserted by the vendors and disturbed by the high salaries being offered to Smalltalk consultants by the corporate world.

We recently attended a briefing session held by one of the major vendors; the word Smalltalk was not mentioned until a full 20 minutes into the presentation. We have to admit this was a very strange experience.

But what does this all mean and to what degree, if at all, should we get worked up about it?

Much of the discussion has centered around the notion

that if all that is needed to develop applications is a good GUI builder and relational database interface then Smalltalk is a sledgehammer of a tool. This is true! Our own experience however is that while Smalltalk must compete at some level with the PowerBuilder and Visual Basics of this world, what ultimately distinguishes it from these tools is that Smalltalk scales and the other tools blatantly do not.

Smalltalk can and has been used to develop large-scale industrial strength applications. Smalltalk will score heavily when modeling objects from the business domain, where component reuse is expected, and where the scale of the project demands good (object-based) tools for configuration management and version control. Moreover, it is wrong to characterize the projects to which Smalltalk is being applied in the corporate world as "client-server systems with RDBMSs in the middle and Smalltalk GUI front-ends." We have become familiar with numerous Smalltalk projects in large banks, insurance companies, and telecommunications companies. The complexity of their applications matches anything we have seen in the scientific and engineering communities.

The vendors will continue to emphasize what they feel is important to increase their market share and penetrate new markets. And they are doing a pretty good job of it. Smalltalk is the fastest growing OOPL. The Smalltalk marketplace doubled in 1993 relative to 1992. We'll take bets it doubled again in 1994. So while the "greybeards" will continue to be nostalgic for the good old days of Smalltalk—and to be honest in many ways so will we—these really are exciting times for Smalltalk and Smalltalkers.

To end on a cautionary note... another fear that has been expressed is that in the rush to give the corporate world the components and features they crave for, some key problems closer to home may be too easily overlooked. We agree. The global namespace problem and the need for better browsers are two of our favorites. I am sure you have your own wish-list.

Enjoy this month's issue and have a happy and prosperous 1995.



JOHN PUGH



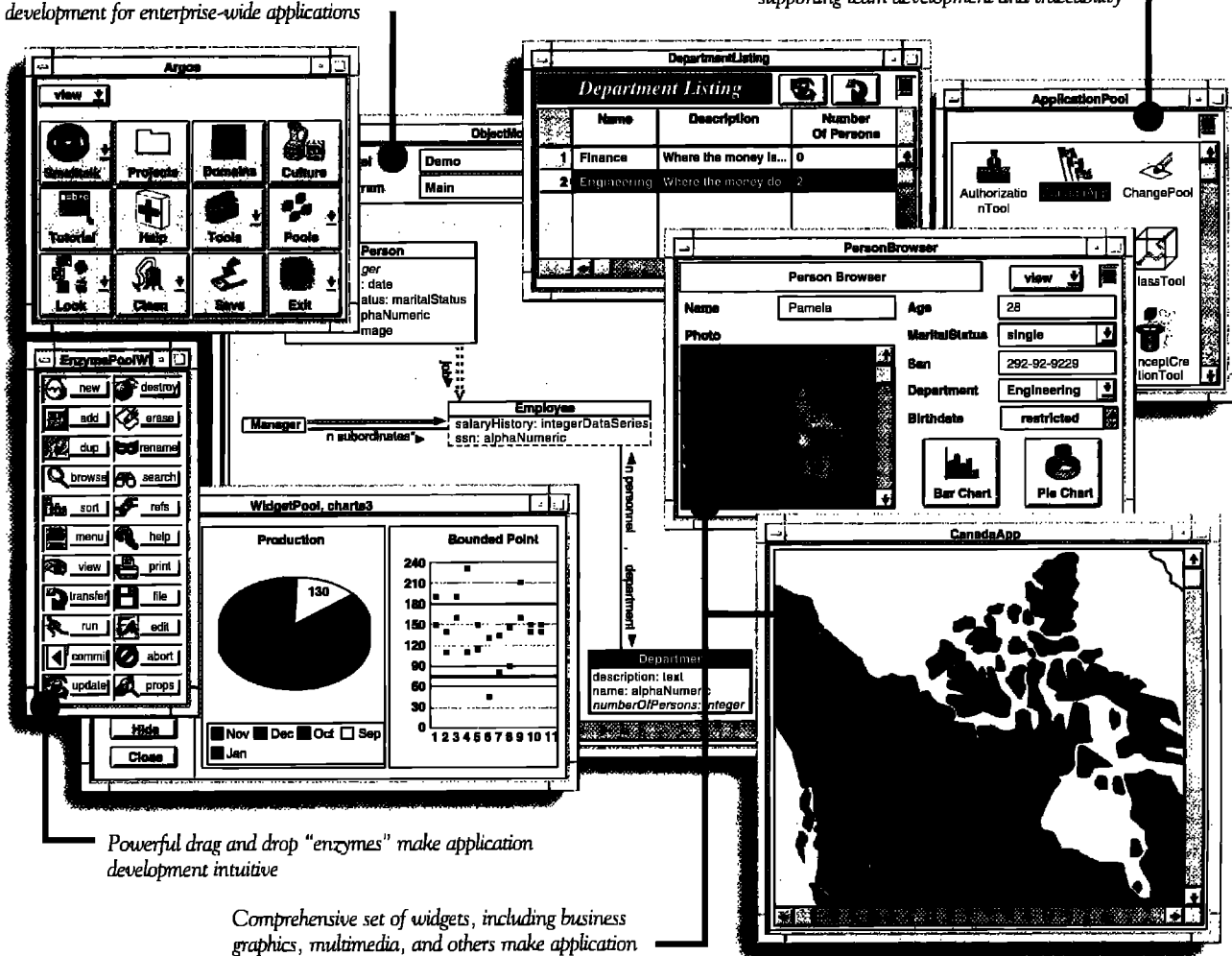
PAUL WHITE

# Introducing Argos

The only end-to-end object development and deployment solution

An integrated object modeling tool provides model-driven development for enterprise-wide applications

All object models are managed in a shared repository, supporting team development and traceability



Powerful drag and drop "enzymes" make application development intuitive

Comprehensive set of widgets, including business graphics, multimedia, and others make application development easy and powerful

VERSANT Argos™ is the only application development environment (ADE) that makes it easy to build and deploy powerful, enterprise-wide object applications. Easy because Argos features an embedded modeling tool and Smalltalk code generation that ensure synchronization between your models and applications. Powerful because Argos supports full traceability and workgroup development through a shared repository.

Argos automatically generates multi-user database applications that run on the industry-leading VERSANT ODBMS. Argos deals with critical issues such as locking and concurrency

control transparently. And only Argos is packaged as a completely visual ADE built on ParcPlace VisualWorks®.

Leading organizations — in industries from telecommunications to finance — are using Argos to deliver business-critical applications. Find out how Argos can help you deliver your critical applications in weeks, instead of years.

Contact us today at  
1-800-VERSANT, ext. 415  
or via e-mail at  
info@versant.com

**VERSANT**  
The Database For Objects™

1380 Willow Road • Menlo Park, CA 94025 • (415) 329-7500

---

# *ENVY software baselining process*

Barry Oglesby

**E**NVY is a Configuration Management tool built specifically for Smalltalk code. Among other functions, it serves to simplify the code baselining process. Building baselines in ENVY is an important part of the software life cycle. ENVY baselines act to control versions of software released to external entities and to synchronize Classes, Applications, and Configurations between developers so that the most up-to-date versions of these are readily available to the entire development team.

Baselines help to ensure that the system being developed works as an integrated whole, not just a series of unrelated pieces. They also help developers as they are developing their piece of the system to see the latest state of the other pieces of the system. In addition, they allow previous versions of the system to be easily retrieved. Two types of ENVY baselines exist, which I call partial and full baselining.

Outlined below are the general steps to be followed when performing both partial and full ENVY baselines. Readers should have a working knowledge of ENVY when reading this article.

## **PARTIAL BASELINING**

Partial baselining synchronizes the Applications or Classes of two developers to ensure that code that each has written is functioning properly. Partial baselining is not a formal process. It is an *ad hoc* practice performed by individual members of the development team between full baselines (see below). Often, developers in this situation will have been developing two parts of the same system behavior. Typically, one developer builds the system domain objects in one Application, while the other develops the user-interface objects in another. For the behavior to be fully tested, the developers must synchronize their Applications. Depending upon the behavior, this synchronization can happen as often as daily. It should happen at least weekly, but it should not be dependent upon the calendar. It should happen at logical points in the development phase, such as when behavior is fully defined.

## **PROCESS**

To accomplish this type of baselining, each developer versions his own Classes and releases them into the appropriate Application's current edition. The other developer can then access those new Class versions by loading (or reloading current) the Application edition.

## **TESTING**

At this point, testing between these developers can occur.

During this testing time, each developer may find errors in the other's code. Often, the developer who finds the error will also make the fix to the code, even if the developer is not the Class owner. This is perfectly permissible. This will involve creating a new edition of the other developer's Class, making the method fix, versioning the Class edition and, finally, notifying the other developer of the fix. The other developer may then incorporate that fix into his Class edition. Alternately, if an edition of the Class does not exist, the developer may release the version containing the fix into the Application edition. It is always the responsibility of Class owners to release versions of their Class into their appropriate Applications, even if they are not the version developers. Developers should never "change user" and release a version of a Class not owned by them without the knowledge and consent of the owner.

## **FULL BASELINING**

Full baselining synchronizes the Applications of the entire development team to ensure that the system as a whole is functioning properly. Full baselining is a formal process that is accomplished using ConfigurationMaps. ConfigurationMaps are named groupings of Applications. Usually, a ConfigurationMap contains all Applications necessary to create a fully functional system. Building full configurations is the responsibility of the Team Librarian. The Team Librarian usually is, but does not have to be, Library Supervisor.

The frequency of full baselines can be based upon either the calendar or system behavior. Calendar-based baselines should be performed, at most, every two weeks. System-behavior-based baselines should be performed on defined system iterations. Shorter, calendar-based baselines have the advantage of synchronizing the development team more often. The longer the interval between baselines, the more system behavior exists that must be synchronized. The partial baselining practice described above will reduce this problem. The disadvantage of calendar-based baselining is that ENVY history management should be determined by system behavior rather than the calendar, but the reality is that there are calendar-based schedules that must be met.

## **PROCESS**

Before the first time a full baseline is performed, the Team Librarian will have created the ConfigurationMap, thus becoming its manager. Once the ConfigurationMap is created, an edition is opened on it automatically. The Team Librarian will then add each of the appropriate Applications to the

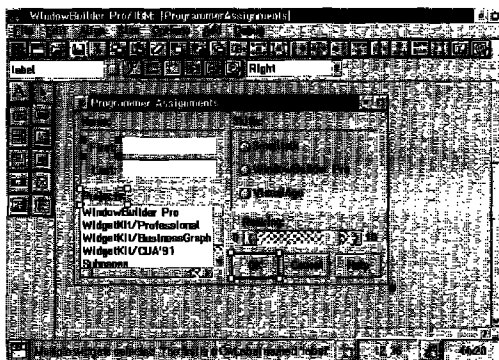
# “The Difference Between Success and Failure in IBM Smalltalk”



**WindowBuilder™ Pro** is an interactive tool that lets you build polished user interfaces fast in Smalltalk from Digitalk and IBM. WindowBuilder Pro (WBPro) saves you from the job of building UIs in code. It helps simplify maintenance and increase consistency.

## Like VB, with Real Objects

Select controls from a palette. Place and edit them interactively. Integrate the controls with your app easily. Build composites of controls to create your own reusable UI components. Place and edit them in WBPro just like the native controls. Get portability of your UIs across all the supported platforms of a Smalltalk family. Includes autosizing, automatic alignment, control of fonts, menus, colors, and more.



Building user interfaces is easy

## High-Level Controls for WBPro

When you use the high-level add-on controls like spreadsheets, business graphics, and others, your apps will be more powerful and polished. And you'll save even more time and effort. Inquire about specific offerings and platform availability.

*“For most Smalltalk/V programmers, WindowBuilder Pro/V is a survival tool—the difference between success and failure.”*

— Milan Sremac, President, Medical Software Systems

WINDOWBUILDER PRO/V (VER. 2)		
For Digitalk	<b>Windows</b> .....	<b>\$495</b>
Visual Smalltalk	<b>OS/2</b> .....	<b>\$495</b>
For Smalltalk/V Win16 (WBPro/V ver 1) .....		
		<b>\$295</b>

*“Unless you're totally comfortable with the Motif API, a tool like WindowBuilder Pro is the difference between success and failure in IBM Smalltalk.”*

— Gordon Sheppard, Senior Technologist, American Management Systems

WINDOWBUILDER PRO (FOR IBM SMALLTALK)			
OS/2 std. ....	<b>\$495</b>	Team .....	<b>\$695</b>
Windows std. ....	<b>\$495</b>	Team .....	<b>\$695</b>

No runtime fees are required for applications developed with WBPro. Free support for the first 90 days. All products include complete documentation. Support subscription available. WindowBuilder Pro/V is compatible with Team/V. Code generation in IBM Smalltalk is totally Motif compliant. © Objectshare Systems, Inc. 1994

## Visual Smalltalk

**WindowBuilder™ Pro/V** lets you build UIs interactively, save time, simplify maintenance. Version 2 is fully compatible with Visual Smalltalk and Visual Smalltalk Enterprise. Generate ViewManager subclasses, ApplicationCoordinator subclasses, or PARTS windows.

WINDOWBUILDER PRO/V			
Windows .....	<b>\$495</b>	OS/2 .....	<b>\$495</b>
Upgrade WindowBuilder Pro/V to version 2. We have special upgrade pricing to registered users. Please inquire.			

**Subpanes/V** provides columnar list box, hierarchial list box, table pane, bitmap pane, bitmap button, 3-D frames, and more. Requires WindowBuilder Pro/V ver 2 and Visual Smalltalk or Visual Smalltalk Enterprise.

SUBPANES/V			
Windows .....	<b>\$235</b>	OS/2 .....	<b>\$235</b>

## VisualAge

Spreadsheets, business graphics, and other high-level components are easy to add to your VisualAge™ based applications.

**WidgetKit™/Professional** has powerful spreadsheets and more. You get virtual spreadsheets, multi-column list boxes, table editor, graphic viewers for BMP, PCX, and GIF, input validation, file system widgets, and more.

WIDGETKIT/PROFESSIONAL			
Windows std. ....	<b>\$495</b>	Team .....	<b>\$795</b>

**WidgetKit/Business Graphics** has versatile graphs and charts. You get bar, pie, area, line, gantt, high-low-close, scatter, and more basic types. Options include 2-D and 3-D, fonts, colors, control of printing, and more.

WIDGETKIT/BUSINESS GRAPHICS			
OS/2 std. ....	<b>\$495</b>	Team .....	<b>\$795</b>
Windows std. ....	<b>\$495</b>	Team .....	<b>\$795</b>



Objectshare Systems, Inc.  
5 Town & Country Village  
Suite 735  
San Jose, CA 95128-2026  
Fax 408-970-7282  
CompuServe 76436,1063

## Call to order (408) 970-7280

Or call for free info. 9 AM to 5 PM PST, M-F. 30-day money-back guarantee

ConfigurationMap edition. Not only will this add Applications to the ConfigurationMap, but it will also release appropriate editions of these Applications. Now, a member of the development team need only load the edition of the ConfigurationMap to get the latest system configuration.

The process described below assumes an ENVY configuration using SubApplications. If no SubApplications are used, the steps involving them may be omitted.

#### Prebaseline Steps

Just before the full baseline, the development team will version their current Class editions and release them into their current SubApplication or Application editions. Then, the SubApplication managers will version their current SubApplication editions, immediately create new editions of their SubApplications and notify the Application manager. Frequently versioning and releasing Classes is a lightweight operation and a normal part of daily ENVY life. Frequently versioning Applications and SubApplications is not and should only be performed at the time of a full baseline configuration. Only Application and SubApplication managers may create editions of Applications and SubApplications. Developers should never “change user” and create an edition of an Application or SubApplication not managed by them without the knowledge and consent of the manager.

After being notified of the SubApplication versioning, Application managers must perform several tasks. First, they version their Application editions. Next, they create new editions of their Applications and release the new editions of the SubApplications into them. Finally, they notify the Team Librarian. Changing the state of an Application from edition to version will automatically be reflected in the ConfigurationMap edition, if the Application edition has been released into that ConfigurationMap edition.

While the baseline process is occurring, the development team can continue developing system behavior in the new Application and SubApplication editions. Creating or editing methods will cause new Class editions to be created. These Class editions can be versioned and released within the new Application and SubApplication editions without affecting the baseline.

#### Baseline Steps

To get a fully baselined image, the Team Librarian need only load the current ConfigurationMap edition into a virgin image. The full baseline image is now ready for testing. The Team Librarian should now execute a script or scenario to test the configuration. This scenario is usually developed over the life of the project and can range anywhere from a handwritten list of steps to be performed to a series of Smalltalk “doits” or test methods to be executed. Initially, the individual Application developers may execute their own example methods after a baseline. As the system becomes more robust, a richer scenario will be necessary.

Since the Team Librarian will be testing the new configuration in this image, no individual developer’s environment will affect the tests. Also, testing in an image containing only the Application

editions in the ConfigurationMap will help uncover any developer’s special environment dependencies, such as Global settings.

Assuming for the moment that the ConfigurationMap loaded without any problems, and the behavior of the system before and after the baseline is the same, the Team Librarian must now version the ConfigurationMap to complete the baseline.

#### Postbaseline Steps

Once the baseline is completed, the Team Librarian will create a new edition of the ConfigurationMap for the next baseline iteration and release the new Application editions into it.

Once this has been accomplished, the Team Librarian will reload the new ConfigurationMap edition into a virgin image. This image will now contain all the new editions of the



*Full baselining synchronizes an entire development team’s Applications to ensure the system as a whole functions properly...*

Applications and SubApplications released into the ConfigurationMap edition. The Team Librarian will then save a copy of this image.

Next, the Team Librarian will notify the development team that a new configuration is available. Two ways exist for a developer to obtain the new configuration. One is to load the new ConfigurationMap edition into his current image. Loading the new ConfigurationMap edition will cause the new Application and SubApplication editions to load. The developer can then continue development within the latest configuration. Another is to copy the image that the Team Librarian previously saved. Copying the Team Librarian’s image will assure a fresh configuration, but if individual developers have their own environments that they do not want to recreate, loading the new ConfigurationMap edition may be a better choice.

#### POTENTIAL PROBLEMS

This process will take a half day if no serious problems occur and longer if they do. Unfortunately, several problems can occur during a full baseline. These come in several categories, including problems encountered creating the baseline, critical system code problems that must be fixed in the current baseline, and noncritical system code problems that can be fixed in the next baseline. These categories of problems and how they should be fixed are described in greater detail below.

#### Baseline Problems

Baseline problems occur during the loading of the ConfigurationMap edition and cause the baseline to fail. Their causes include but are not limited to:

1. Application class>>loaded method is not defined correctly. A NotifierView will be displayed after loading the Application version. The fix is usually apparent after some investigation by the Application manager.

# Automatic Documentation - Easier Than Ever

With *Synopsis for Smalltalk/V Development Teams*

New!  
Release 2.0  
Builds Help Files

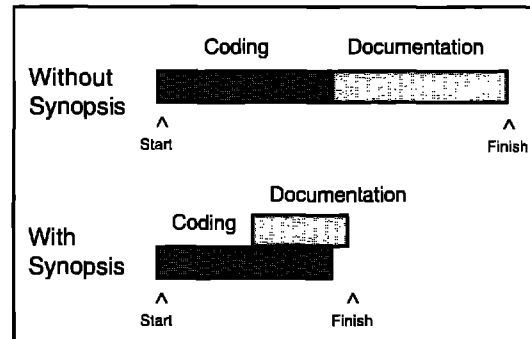
**Synopsis** produces high quality class documentation automatically. With the combination of Synopsis and Smalltalk/V, you *cut development time and eliminate the lag between the production of code and the availability of documentation.*

*Synopsis for Smalltalk/V*

- Documents Classes Automatically
- Provides Class Summaries and Source Code Listings
- Builds Class or Subsystem Encyclopedias
- Publishes Documentation on Word Processors
- Packages Documentation as Encyclopedia Files or as Help Files for Distribution
- Supports Personalized Documentation and Coding Conventions

Working with Synopsis is *easy*. Install Synopsis and see *immediate results* --- without changing a thing about the way you write Smalltalk code!

## Development Time Savings



Products: Synopsis for Smalltalk/V and Team/V  
Synopsis for ENVY/Developer  
Environments: Windows, Win32, OS/2  
Pricing: Smalltalk/V \$295, ENVY \$395  
Site licenses available.



**Synopsis Software**

8912 Oxbridge Court, Raleigh NC 27613  
Phone 919-847-2221 Fax 919-847-0650

2. Application class>>toBeLoadedCode variable is not defined correctly. A NotifierView will be displayed before loading the Application version. The fix is usually apparent after some investigation by the Application manager.
3. Identical methods in the same class warning will be written to the Transcript, and one Application version will be scratched. One of the Applications must delete the method.
4. Prerequisite Applications are not loaded. A Dialog will be displayed that asks whether to load the prerequisite Applications. Either load the prerequisite Application before loading the ConfigurationMap or include it in the ConfigurationMap to fix the problem.
5. Prerequisite Application versions are not ENVY-compatible with Application versions in the ConfigurationMap. An error will be written to the Transcript, and the load will not complete. One way to fix this is to set the ImageBuilder enforceCompatibility flag to "false" to turn off ENVY prerequisite checking. If this is the case, a warning instead of an error will be written to the Transcript, and the load will continue. Another way is to release the necessary prerequisite Application version into the Application edition in question using the ApplicationConfigBrowser.

### Critical System Problems

Critical system problems occur during the testing of the scenario and cause the system to behave unacceptably. Either behavior that the system was thought to exhibit before the baseline is different from the behavior it does exhibit after the baseline, or the system fails so that the sce-

nario cannot be completed properly. Causes include but are not limited to:

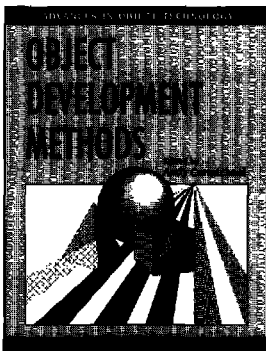
1. An incorrect Class version is released into its Application edition. The proper version of the Class must be released into the Application edition. The Class owner is responsible for ensuring that the released version of the Class is correct. Only the owner of a Class can release a version of that Class into the Application edition.
2. A developer has a special environment set up in his own image that does not get baselined. Usually, this environment consists of a Global or Class variable that either exists



*As with any process, custom-tailoring for each Smalltalk application may be necessary—it helps to ensure the success of any ENVY project*

in one image but not in the other or is set to one value in one image and to another value in the other. It may also consist of a change to a Method not in the developer's Application that is being relied upon but does not get into the baseline. In the Global or Class variable case, the environment may be easily recreated in the baseline image by

*Explore the  
Leading Methodologies...*  
**Object Development Methods**  
edited by Andy Carmichael



ISBN 0-9627477-9-3

*Object Development Methods* addresses how object technology can be applied to systems analysis and design. It includes a comprehensive survey and comparison of the leading methodologies including Booch, Texel and Rumbaugh.

The common concepts and underlying structure of each methodology is an important theme explored within this book. *Object Development Methods* proves an invaluable reference guide for those still exploring various methodologies and their benefits/drawbacks, and for those who have already made a methodology selection.

**Who Should Read This Book?**

Systems Analysts/Designers	IT Managers
Chief Scientists	Programmers
Project Managers	Software Engineers

347 pages  
Available at selected bookstores.  
Distributed by Prentice Hall.



Please rush me \_\_\_ copies of *Object Development Methods* at the low rate of \$39. ISBN 0-9627477-9-3

**METHOD OF PAYMENT**

Check Enclosed (payable to SIGS Books)  
 Charge My:  Visa  Mastercard  Amex

Card No. \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_

Zip \_\_\_\_\_ Country \_\_\_\_\_

*US orders add \$5 for shipping & handling; Canada add \$10, all others add \$15. New York State residents add applicable sales tax.*

**TO ORDER, MAIL THIS COUPON TO:**

SIGS Books, P.O. Box 99425, Collingswood, NJ 08108-9970  
FAX TO: 609.858.2007 OR PHONE: 212.242.7447

using the Application class>>loaded method. In the Method change case, the Class in question must be versioned and released into its containing Application.

3. System behavior spans multiple Applications managed by different developers. One developer is depending on a method from the other that is not built into the baseline. Either the method signature or the return value does not match what was expected. Often, one developer will make a change to his method without notifying the other developer before the baseline. Frequent partial baselining will largely prevent this from occurring.

**Noncritical System Problems**

Noncritical system problems also occur during the testing of the scenario. Their causes are identical to those of the critical system problems described previously, but their fixes can be deferred until the next development period. The difference is that these problems can be bypassed or are minor enough to allow the scenario to complete acceptably.

**Fixing Problems**

To fix most of the problems described above, an edition of the appropriate Application and/or SubApplication is necessary. The development edition created in the prebaseline steps should be used to incorporate the fix. Using the development edition of the Application and/or SubApplication may entail backing out any new development that has occurred.

First, any newly released Classes should be backed out by releasing the previous versions of the Classes in question. These newly released Class versions can easily be found by checking the differences between the previous version and the new edition of the SubApplication or Application. The fix can then be made and verified in the developers local environment.

While the fix is being made, the Team Librarian should release the new edition of the Application into the edition of the ConfigurationMap. Once the fix is made, the new edition of the ConfigurationMap can then be loaded to test it. Once the fix is verified by the Team Librarian, the prebaseline steps described previously should be followed for the Application and/or SubApplication to allow the baseline to continue.

When an error occurs during the build process, it is advisable, once the problem has been corrected, to reload the entire ConfigurationMap to ensure that the correction is proper. Also, a ConfigurationMap edition must be successfully loaded with no modifications of the contained Applications before it can be versioned.

**CONCLUSION**

The preceding partial and full ENVY Baseline steps provide an effective way to baseline Smalltalk code. As with any process, custom-tailoring for each Smalltalk application may be necessary to account for multiple ConfigurationMaps, external code incorporation, etc. Following processes such as these will help ensure the success of any ENVY project. ♀

**References**

1. ENVY/MANAGER USER MANUAL, Release 1.41, 1993, Object Technology International, Inc.



# A technical overview of VisualWorks 2.0

Jim Haungs

THE new release of ParcPlace VisualWorks version 2.0 contains many new features, improved database connectivity, new user interface design tools, and supports several new platforms. ParcPlace has done significant clean-up work on the base classes, but the development tools remain largely unchanged. This release demonstrates ParcPlace's ongoing commitment to binary portability across platforms.

## SYSTEM INFORMATION AND SET-UP

The first thing you'll notice about VisualWorks 2.0\* is the new Visual Launcher (see Fig. 1), which combines the system transcript, controls for the browsing and editing tools, and the functions from the old LauncherView. The menus control most of the tools and browsers; the tool bar provides immediate mouse access to the more common functions on the menu bar.

Off the File menu is the Settings notebook (see Fig. 2). The settings notebook replaces much of the manual text manipulation that used to be found in the Installation and System Workspaces. The configuration text and dialogs have been replaced with a single visually appealing and space-efficient paged notebook. The notebook control is patterned after IBM's CUA'91 Notebook Controls in OS/2. It has also been added as a widget to the VisualWorks toolbox, so it can be used in user applications.

The settings notebook contains pages for the locations of the source and changes files, the VisualWorks configuration settings, the UI look preferences, window replacement preferences, help file locations, font preferences, and the printer and time-zone settings. Pressing the Accept button on a settings page puts that setting into effect immediately.

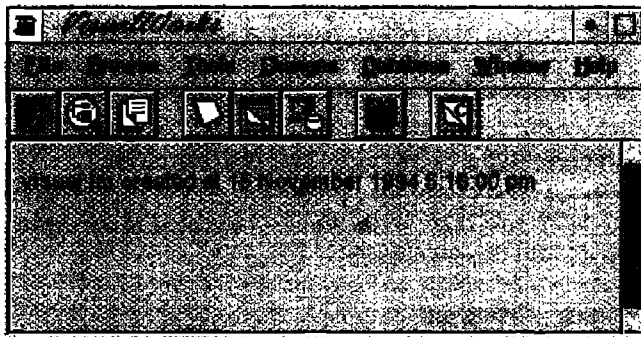


Figure 1. The VisualWorks Control Panel.

Note: Most of the information herein comes from the user manuals and from using VW 1.0 and 2.0. Additional information was obtained at the August 1994 ParcPlace Users' Conference, in particular, information regarding future directions for some of the product line.

One of the major improvements is the new on-line help browser (see Fig. 3). When you press the help button on the toolbar, the help browser presents a list of "books" that come with VisualWorks. Selecting a book gives you a list of chapters; selecting a chapter gives you to a list of topics; selecting a topic gives you a page with a "Strategy" section describing the topic, then a list of "Basic Steps" and common "Variants." The browser is complemented by a Search dialog that allows case-insen-

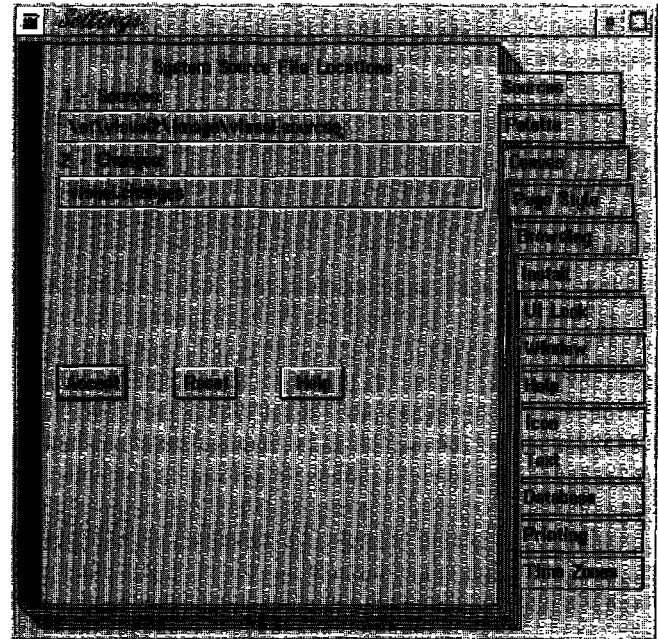


Figure 2. The VisualWorks Settings Notebook.

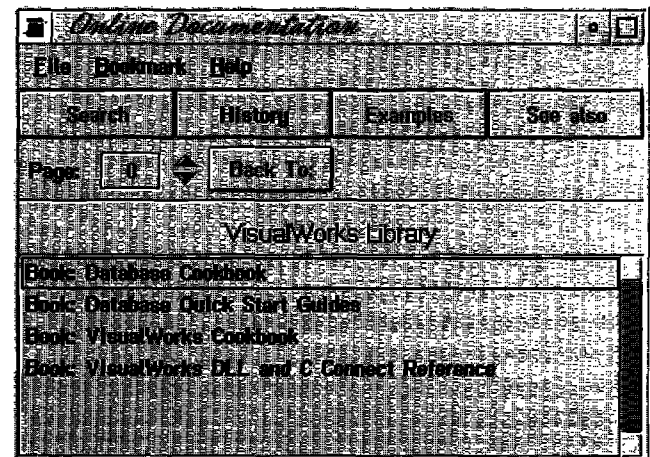


Figure 3. The Online Help Browser.

sitive and regular expression searching. The on-line information augments rather than duplicates the information in the printed manuals; it is more action-oriented than the printed materials; in particular, it has an example browser with a DoIt button.

In addition to the on-line help, VisualWorks 2.0 comes with more than 2000 pages of documentation. THE USER'S GUIDE (400 pages) covers the Smalltalk language and the base classes, the browsing and development tools, and the MVC paradigm. THE OBJECT REFERENCE (1300 pages) organizes the public classes in alphabetical order, describes their roles in the system, the application in which they are organized, their instance variables, and their public method interfaces. THE VISUALWORKS COOKBOOK (700 pages) covers the creation and manipulation of VisualWorks canvases (see Fig. 4). Additional manuals ship with various add-on tools: Database Tools, Advanced Programming Tools (formerly APOK), and the DLL and C Connect (formerly CPOK).

### THE VISUALWORKS CANVAS

The user interface design tools have been significantly improved in this release. In particular, all widget properties are accessible via notebook pages in the Property Tool. As each widget is selected, the Property Tool displays settings pages relevant to the class of the selected widget. The biggest improvement is when multiple widgets are selected: the settings pages reflect all properties common to the selected widgets. This means you can now set the font or color for all selected widgets on the canvas, e.g., all labels, in a single operation.

The Tool Palette supports several new and important widgets. I have already mentioned the Notebook widget. There is also a Combo Box widget that combines the capabilities of an entry field and a drop-down list; read-only Combo Boxes restrict the user to a prespecified list of choices; read-write Combo Boxes let the user enter a new value that is not found in the initial list. The Menu Button combines a read-only entry field and a drop-down or pop-up list. Dividers provide straight horizontal and vertical lines for separation of visual regions. Sliders permit a choice from a range of numbers; write-only sliders can be used to provide visual feedback on the progress of an operation.

Besides the Notebook widget, the biggest improvement to the widget set is the new table widgets. There are now several flavors of tables: the original TableInterface is still available, but the new DataSet widget is much better. In particular, DataSets support direct in-cell editing, and column widths that can be changed at runtime by either the program or by the user. DataSets can be completely specified via the Property Tool; once defined, the order of the columns can be changed simply by dragging the columns around. Certain columns can be designated as Fixed, i.e., they do not move when the table is scrolled. This can be used to place row and column titles in fixed positions and let the variable data scroll. Although the implementation is somewhat convoluted, you can also make formatted, multi-line column headings by designating a column head as type Image, and giving it an aspect method that returns a ComposedText. ComposedText labels can have their own alignment, fonts, and character attributes, such as bold

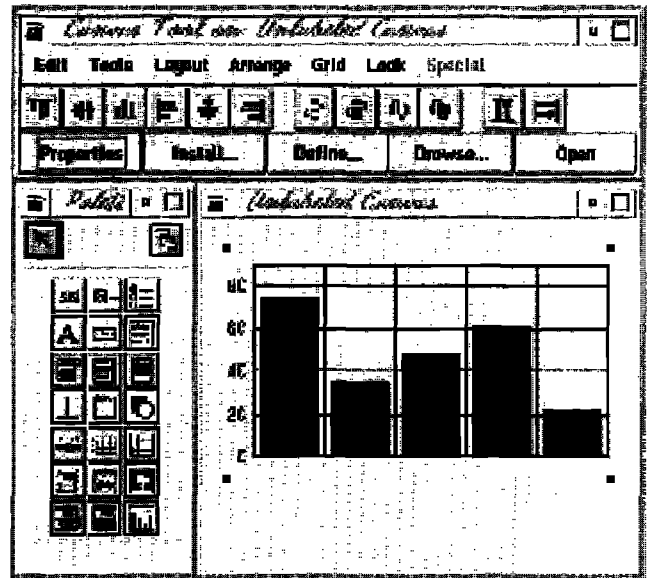


Figure 4. The VisualWorks Canvas.

and italic text.

VisualWorks 2.0 supports a number of new validation protocols for user data entry. The application model can be notified when the focus enters or leaves a field, and a validation method can be specified for each widget. The validation method can determine whether the focus should be allowed to leave the field or not. It can also change the color and other attributes of the field to bring validation errors to the user's attention. VisualWorks now provides built-in methods for validating common data values, such as Strings, Numbers, Dates, Times, Passwords, TimeStamps, and Booleans. In addition, a field can have a validation format, such as phone numbers or social security numbers. The validation language is a simple pattern match with wild cards; you can define your own patterns, or use one of several built-in patterns. Since VisualWorks is an extensible framework, any patterns you develop can easily be added to the Property Tool and reused.

### BUSINESS GRAPHICS

The Business Graphics package is also nicely integrated into the VisualWorks canvas. When you drag and drop a chart onto the canvas, a sample chart is shown with dummy data. The sample chart resizes and scales with the canvas. There are 11 different chart types to choose from: Bar, Pareto, Picture, StackedBar, Band, Layer, Line, Stacked Line, Step, XY, and Pie charts. Except for the Pie chart, any chart can be oriented either horizontally or vertically. You can also specify tick marks and legends and select their placement on the chart. Each chart type has myriad options; as you choose the options and press the Apply button, the sample chart on the canvas is updated to show you how the real chart will look.

### POSTSCRIPT PRINTING

VW 2.0 augments the existing graphics framework with a new PostScript printing framework, creating a consistent imaging model for both screen and printed graphics. Unfortunately, the framework only extends to PostScript

printing; it does not allow any platform-specific or printer-specific rendering, such as Windows GDI or PCL. Most of the framework seems quite robust and well thought out, and if you have a PostScript printer, is ideal for creating printed output that matches your graphical output. The fact that most operating systems do not support the querying of device capabilities, some rather unobvious things, such as the printer resolution, are inelegantly hard-coded into the framework.

### RELATIONAL DATABASE SUPPORT

VW 2.0 supports access to relational databases via a set of classes called the ObjectLens. This technology was initially implemented by a third party vendor, but with this release has been integrated into the base system. You need to buy a Database Connect kit for each brand of database you intend to access. Data can come from different sources simultaneously. The system currently supports Oracle and Sybase, with plans for DB2 and ODBC in the future.

Before I discuss the ObjectLens implementation, I must review some significant differences between the relational model and the object model. Although object databases are not yet as widespread as relational databases, the differences between the two models merits inspection. Nothing in the object model is impossible to translate to the relational model, but in several common cases, the performance is so atrocious that it is not worth it. There are several places where the relational model is weaker than the object model: when an instance variable is a collection, when a collection must be heterogeneous, when a model uses subclasses, inheritance, and polymorphism, or when the data must be encapsulated and manipulated only through its methods. The first is simply cumbersome; the second is difficult to achieve and still get good performance, and the third can only be handled with stored procedures, which is not true encapsulation.

The primary purpose of the ObjectLens is to transform relational data between the database representation and the object representation. However, there is a big difference between accessing your relational data as objects, and storing your object model persistently in an object database. While the Lens supports both reading and writing the data, it does not support the use of inheritance or polymorphism, and thus does not fully support the object model. Thus, if you are designing an object system with real subclasses, any use of inheritance, and therefore polymorphism, cannot be supported by the underlying database. For example, you could support both Employee and Manager tables in the database, but if you tried to model a Manager as a subclass of an Employee, the modeling tools do not support it. This is an understandable limitation, since the modeling of inheritance through joins results in unacceptably slow queries, and even slower updates. But the primary use of the ObjectLens must be viewed as accessing existing databases, typically legacy systems. (The ObjectLens Database Modeler is shown in Fig. 5)

However, once a system is designed in a relational model, necessarily without inheritance, encapsulation, or polymorphism, it is quite easy to map the tables onto Smalltalk objects. And there are certainly enough relational databases out there, and enough such data models, that the Lens technology is still

## Precise metrics for advanced OO development.



- Metrics collection facility for Smalltalk applications development
- Supports VisualWorks, Smalltalk/V for Windows, Win32s, Windows NT
- Complete graphical user interface
- Fully supports Envy (optional)

**objectSpace™**

SPECIALISTS IN OBJECT TECHNOLOGY

PRODUCTS • TRAINING • CONSULTING • MENTORING • AUDITING  
For more information call 1-800-OBJECT-1, Email: info@objectspace.com

Copyright ObjectSpace, Inc. ©1994. All names and trademarks are the property of their respective owners.

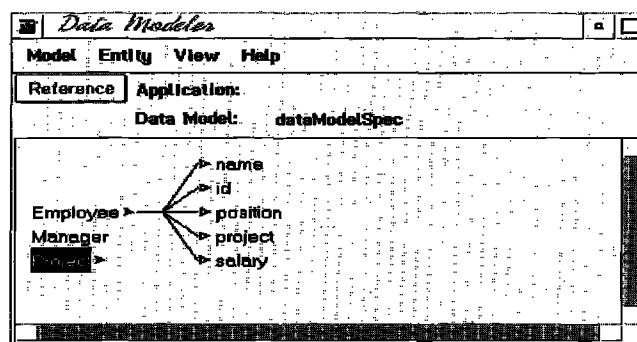


Figure 5. The ObjectLens Database Modeler.

an important breakthrough, primarily because of its tight integration into the base classes and into VisualWorks. The Lens Data Manager automatically transforms data between the database internal form and the Smalltalk instance representation. The data manager also generates DataSet widget specs from the table data models, making it simple to display and edit tabular data. There is even support for the non-SQL programmer in the form of the Query Assistant: a canvas of push-buttons that insert SQL syntax into the Query Editor window.

The Lens tools are the Ad Hoc Query tool, the Modeling tool, the Mapping tool, the Key Editor and the Query Editor. You use the Modeling tool to describe your tables as classes, and then use the Mapping tool to map the table columns to instance variables. If there are joins involved, the common keys are designated via the Key Editor. You can then store queries with the class, which are executed when the data is fetched.

There is considerable control and flexibility afforded via the use of sessions, answer streams and transactions. You can control the granularity of the queries as well as the quantity of the data returned with each fetch; you can also execute stored procedures in the database.

### **DLL AND C CONNECT**

The DLL and C Connect (DLLCC) product provides an interface to external libraries written in C. There are two modes of linking to C object code. Dynamic-link or shared libraries (DLLS) are linked into the process address space at run-time. Static linking is provided for those platforms that do not support DLLS; object code is statically linked into the Object Engine (the new name for the VM). Although this is structurally similar to creating user primitives, calling an entry point in a DLL is considerably faster than calling a primitive; in some future release, DLL access is likely to replace user primitives altogether.

To create an interface to a C API, you must first create an interface class. The External Interface Builder parses the #include header files of the API and generates methods for the API entry points. Because most calls to an API pass data of particular types to the entry points, DLLCC also defines the data interface, including #defines, typedefs, and structs. It is even possible to represent extern data variables.

The external interface can be built in either debug mode or optimized mode. Debug mode checks the types of all arguments to the external functions; optimized mode does not. Debug mode is used during development; the interfaces can be optimized once they have been debugged.

It is also possible to define callbacks from C to Smalltalk. When you want your C function to call a Smalltalk function, you create a callback object for the Smalltalk method and pass it to the C program. The callback object contains the address of a *thunk* (a chunk of machine code) that performs the transition between the C caller and the target Smalltalk code. There are some limitations on the usage and ordering of callbacks, but these are well-documented, and don't seem too onerous.

I have only one criticism of this implementation. It is far and away the best interface I've seen between Smalltalk and C, but it's incredibly difficult to debug on platforms that use segmented memory architectures, such as DOS. DLLCC is designed for flat 32-bit data and code addresses. If you attempt to use include files that do not explicitly include *\_far* designations on both the function *and* the parameters, the object engine calls the function incorrectly, and the image summarily crashes. The engines on certain platforms have an exception handler to trap faults during DLL calls and return them gracefully to the image. This support is not available on every platform, but because exceptions in an API call are almost always due to an incorrect interface definition, it is much better to fail the API call and produce a walk-back than to allow the image to crash.

### **THE ADVANCED PROGRAMMING TOOLS**

The Advanced Programming Tools (formerly APOK) contain a number of extraordinarily useful tools, some marginally useful ones, and some downright lame ones. Most of the tools really are for advanced developers: people whose livelihood depends on delivering good Smalltalk apps quickly. As a developer, I would

like to see the some of the good stuff bundled with the base system, but I can understand the appeal of the extra revenue stream.

The extremely useful tools include the Time and Space profiles; these let you find performance and memory hot spots in your code. No one should even consider deploying a Smalltalk application until they have run their applications under these tools. They will tell you things you probably couldn't have imagined, much less known. They are also among the easiest profilers I have used, and will pay for themselves many times over.

The other excellent tool is the Full Browser. I use this instead of the normal System Browser. It has an extra list pane that shows the inheritance hierarchy of the class you are browsing. Pressing the super/subs buttons includes the supers' protocols in the protocol pane, and the supers' methods in the method pane. The current class contents are bolded, and the supers' contents are normal text. This permits instant identification of inherited methods, and is of particular value in deep hierarchies like ValueModel or ComponentSpec, where the behavior is truly distributed. This is a tool that should be included in the base system, if only for its pedagogical value.

The Static Analysis tools are very good implementations of some not very interesting ideas. Their purpose is to detect inconsistencies in your image, such as messages sent but not implemented, methods that are never called, instance variables that are never referenced, etc. However, due to the dynamic nature of Smalltalk, you can never really tell that a method is never called, nor can you tell if a reference to a nonexistent method will ever get executed. Unreferenced variables waste space, but they won't cause your application to fail. The intent behind these tools is good, and they probably will help you find inconsistencies, but I have found them to be of limited utility in practice. The same goes for the class documentation tool. It generates a very pretty listing of the class comment, the superclasses, the instance variables, the method headers, and optionally, the method bodies. But given the dynamism of the typical Smalltalk image and the typical Smalltalk project, such a listing would be out of date before it hit the printer.

Some of the tools are pretty worthless; they probably found a niche somewhere, and so continue to be supported. Among these are the Project Browser, which lets you switch immediately among the projects in the current image. I don't know of anyone who actually uses projects, so being able to browse them is not very useful. The other tool of dubious value is the VT100 Terminal emulator. I can't imagine that someone would need such an emulator and couldn't find a shareware version with more features. This one lacks essential features like download protocols. The Window Browser is somewhat useful, but it could be much more so. It displays a list of the currently scheduled windows. When you select one, it expands to show successively lower levels of widgets and components under the window. The display consists of a simple textual hierarchy; such a graphical function would be much better served by a graphical display of the subcomponents.

The Parser Compiler lets you define the grammar for little languages and automatically generates the parser for them. It has found considerable use in some decidedly non-little languages. This tool was used to generate the C scanner and parser for the



# Object

E · X · P · O

THE NATIONAL CONFERENCE & EXPOSITION

## June 5-9, 1995

### NY Hilton New York City

## Setting New Strategies — Reaching New Goals

Object Expo returns to New York in 1995 bringing together the industry's most respected Object technology experts and leading companies. Whether you're just exploring the possibilities, implementing new Object-based ideas, or a seasoned professional, Object Expo has more to offer in 1995 than ever before.

### Object Expo '95 Offers an Extensive Technical Program

This year's program offers the strongest technical training available through real-life case studies presented by a prestigious faculty of OT pioneers and innovators. Tracks include:

- C++
- Fundamentals
- Business Strategies
- Databases
- Analysis and Design
- Smalltalk

*NEW* tracks focusing on:

- Client/Server Development
  - Project Management Strategies
  - User Experience
- Concentrate on the issues concerning you most!

### Special Educational Events Include:

- ▲ Keynote Speeches
- ▲ Walk-in Clinics with the Experts
- ▲ Product Education Sessions
- ▲ *NEW!* Product Briefings

### Added Attractions

Open to all—

- ▲ User Group Meetings
- ▲ Birds-of-a-Feather Sessions
- ▲ Panel Discussions

Exchange ideas and brainstorm with peers at these unique seminars fueled by group participation.

### Executive Briefing

*Back by popular demand!*  
An intense session focusing on the concerns of busy, upper-level executives. Register separately and learn first-hand how others have successfully implemented OT to solve business problems. *Seating is limited, register early!*

Don't miss the most high-powered gathering of OT professionals on the East Coast! Be a part of this week long event dedicated to setting new strategies with object technology that guide you to higher levels of software productivity.

Sponsored by:



Presented by:



Please send me more information on

### Object Expo

- Attending Technical Conference     Attending Executive Briefing     Exhibiting     Receiving Free Exhibits Pass

Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Day Phone \_\_\_\_\_

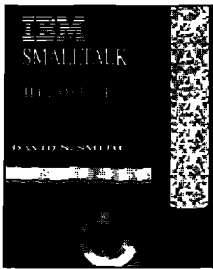
Fax \_\_\_\_\_

**Mail or Fax coupon to: Object Expo**

71 West 23rd Street, New York, NY 10010

**Fax: 212/242-7578**

## SMALLTALK WITH A BIG VOICE!



**IBM SMALLTALK:  
THE LANGUAGE**  
by David N. Smith, Sr.  
Programmer  
IBM T. J. Watson Research  
Center  
(30908-X)

Announcing the *first* detailed reference book covering the Smalltalk language found in IBM's **Smalltalk** and **Visual Age** products! Over 1400 examples guide you through the concepts of **IBM Smalltalk** in this practical tool that answers important questions about the language.

Engage in Smalltalk with the best—ask about **IBM Smalltalk: The Language** today!



THE BENJAMIN/CUMMINGS  
PUBLISHING COMPANY, INC.

*Available wherever fine technical books are sold.*

DLLCC toolkit, as well as the parser for the CORBA Interface Definition Language in HP's Distributed Smalltalk.

The advanced toolkit contains some interesting classes for numeric applications. The **FixedPoint** class permits calculations with business-type numbers: currency, rates, etc., which require fixed precision with minimal rounding errors. The **Complex** class represents numbers with real and imaginary parts. The **MetaNumbers** classes represent abstract numbers such as **Infinity**, **Infinesimal**, and **NotANumber**.

The last tool in the advanced toolkit is useful for measuring Smalltalk performance on various platforms. It provides a standard way to set up repeatable benchmarks. You can select which parts of the system you want to benchmark and how many iterations of the test you want to run. The output report lists the minimum, maximum and average timings on each of the tasks, and gives a cumulative result at the end. All else being equal, this test will give you some of the volumetrics and the performance requirements you will need to deploy a Smalltalk application. For example, if there is a significant speed difference between an 8 MB PC and a 16 MB PC, you can decide whether to spend the money on the extra memory.

### SUPPORTED PLATFORMS

This release of VisualWorks supports several additional platforms. The complete list includes: SunOS 4, Solaris 2, IBM RS/6000, Sequent, HP/UX, Silicon Graphics, MS Windows 3.1 and Windows NT, OS/2, and Macintosh. Images are binary compatible across all of these machines.

### NEW CLASSES

**TimeStamp** combines the **Date** and **Time** classes into a single entity. This is useful for the many cases where you needed a time and date combined, but had to use an **Array** instead. The **TimeStamp** class contains useful behavior such as **printOn:**, accessors for fetching date and time values, and methods for comparing and hashing time stamps.

There are now **Variable** versions of **CharacterAttributes** and **TextAttributes**. Fonts using these attributes will scale properly when moved among systems with differing resolutions and font engines.

The **Menu** classes have been completely rewritten for release 2.0. The new implementation has a very clean design and has simplified the creation of popup and cascaded menus, but introduces some incompatibilities with release 1.0 code. There is a file-in that restores some compatibility, but this is an area to be aware of when upgrading.

### SUMMARY

Release 2.0 is a long-awaited upgrade to the **ParcPlace** product. The goal of binary compatibility across platforms has been upheld even as additional platforms are added. **VisualWorks** continues to improve as a GUI-building tool, and most of the serious drawbacks have been addressed. The database connectivity is now excellent, allowing the rapid development of client/server applications. The **DLL** and **C** support are so outstanding, it is difficult to see where Smalltalk ends and the **C** support begins.

The biggest disappointment in this release is that the development tools are essentially unchanged. The class and protocol categories are still flat. They would be much more useful if they were hierarchical, and permitted sharing of subcomponents. There should be much more connectivity between some of the browsers; in particular, the source code should be modeled and implemented via dependencies, so when you change a method in the debugger, then return to a different browser on the same method, you see the updated source rather than clobber your new change with a stale copy.

For the future, **ParcPlace** has stated a desire to support native widgets in a portable fashion. Exactly how they will do this is an open question, but they express hope that it can be done. There is also talk of namespace support, which would eliminate clashes among source code from different vendors.

Should you buy it? If you currently have 1.0, or you are deploying applications on heterogeneous platforms, an unequivocal yes. If you are serious about Smalltalk, want the best and can afford it, yes. If you must have true native widgets, rather than simulated ones, or your application depends on features of specific operating systems or platforms, then no. ♀

**Jim Haungs** is the founder of **TeamTools, Inc.** He specializes in Smalltalk consulting, training, project management and software development. He has a **BSCS** from **RIT**, and an **MSE** degree from **Wang Institute**. He can be reached at [jhaungs@teamtools.com](mailto:jhaungs@teamtools.com).

# Making MVC code more reusable

Bobby Woolf

**T**WO of the most popular advantages that Smalltalk provides programmers are the ability to implement code that is highly reusable and the ability to easily produce graphical user interfaces (GUIs). Ironically, Smalltalk code for producing GUIs has been very difficult to reuse.

ParcPlace VisualWorks 1.0 introduces several new code frameworks that enhance Smalltalk's MVC (model-view-controller) architecture and make its code for producing GUIs far more reusable. Of these new frameworks, three are key:

- The *value subview* framework: Standard visual widgets that all follow a simple, generic accessing protocol.
- The *value model* framework: Model wrappers that can adapt any model to the generic accessing protocol.
- The *domain model and application model* framework: Separate objects that divide a model's domain state from the services it provides to its view.

These frameworks were added to support the UI Painter and UI Builder, VisualWorks tools that allow the user to easily craft GUI views with very little custom code. You can also employ these frameworks and the techniques they embody in your own code to make it better factored, more reusable, and easier to maintain. This article describes what these frameworks are, why they are necessary, and how they work.

## STANDARD MODEL-VIEW-CONTROLLER

In classic Blue Book Smalltalk (such as Objectworks 2.x), objects are typically complex and contain a lot of state information. Collaborating objects know how to access the various parts of each other's state and make use of them. This can reduce the encapsulation of the individual objects.

A well-known example is the Model-View-Controller architecture, where the model contains the state information and the view displays it to the user.<sup>†</sup> Each view has to interact intimately with its model, knowing exactly what messages to send to get and set each state item and what updates to listen for when the various state items change. Thus each kind of window requires its own view class, customized for its particular model class. For a view to display *n* items, the model has to contain those *n* items as state, and the view has to know exactly how to use each of them.

The major disadvantage of this approach is that all of this custom code is very difficult to generalize for code reuse. There needs to be a simpler way to tell the view that it is composed of

\*The "Blue Book" documents the original definition of Smalltalk.<sup>1</sup>

<sup>†</sup> For further information about the early days of MVC, see Krasner and Pope.<sup>2</sup>

subviews, where each subview displays a single part of the model's state.

## PLUGGABLE VIEWS

The first attempt at creating reusable subviews was the *pluggable view* framework. Pluggable views, which were actually used as subviews, were commonly used in Objectworks 2.5-4.1, and the framework is still available in VisualWorks 1.0 and 2.0 for backwards compatibility.<sup>‡</sup>

In this framework, each subview object contains state to store selectors.<sup>§</sup> Instance variables in the view contain selectors for messages that the view sends to its model. These selectors were used to interact with the model to manipulate that subview's part of the model's state and services. In this way, the subviews could be reused; however, each model class still needed a lot of custom code to lay out the subviews and plug in their selectors. In Objectworks 2.5, this custom code was stored in a separate subclass of *View*; for example, *Browser* had a corresponding *BrowserView* class. The redesigned visual hierarchy in Objectworks 4.0 eliminated the custom view class entirely by moving the custom code into the model.<sup>||</sup>

## An Example

As an example of how pluggable subviews work, consider a view (typically a full window) that displays three attributes for a person: name, address, and telephone number. The view's model would be *Person*, a subclass of *Model* that contains aspects for these three attributes. The view contains three pluggable subviews to display the three attributes. Each pluggable subview contains selectors for interacting with its aspect in the model. In this way, each subview sends messages directly to the model, which in turn is responsible for manipulating the requested aspect that it contains. Figure 1 shows the relationships between a *Person* model and its view.

## VALUE SUBVIEWS, VALUE MODELS, AND VISUAL SPECS

The second attempt at creating reusable subviews was the introduction of the value model framework and a library of

<sup>‡</sup> In fact, in VisualWorks 1.0 and 2.0, all of the Objectworks development tools such as the code browsers, the debugger, the change list, and the file browser are still implemented using pluggable views such as *ListView*, *TextView*, and *Button*.

<sup>§</sup> A selector is an instance of *Symbol* that represents the name of a method. This method name can be sent as a message to an object by performing the selector. To perform a selector on an object, send the object the message *perform...* (defined in *Object*) with the selector as the argument.

<sup>||</sup> For further information on the rearchitecture of the visual hierarchy in Objectworks 4.0, see Leibs and Rubin.<sup>3</sup>

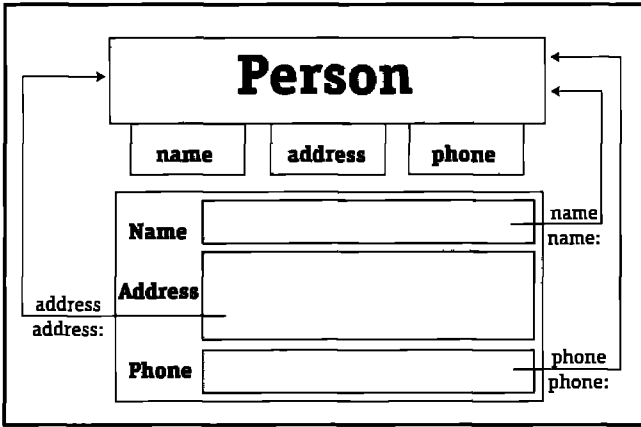


Figure 1. A Person model and view using pluggable subviews.

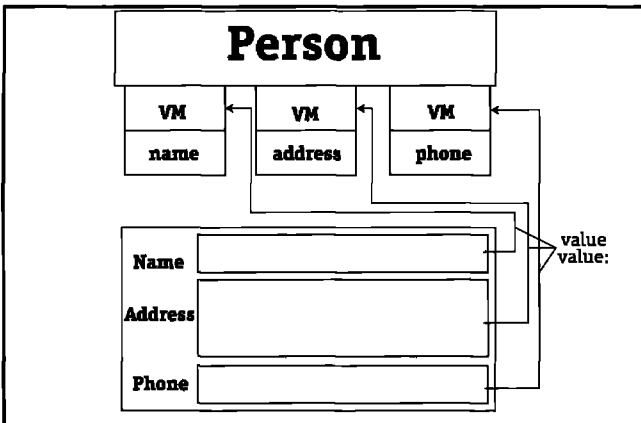


Figure 2. A Person model and view using value subviews and value models.

simplified value subviews. The value model framework was first introduced in Objectworks 4.1, but was underutilized until VisualWorks 1.0 introduced the UI Painter with its palette of standard visual widgets.

In VisualWorks, each *value subview* (analogous to the old pluggable views) displays a single value, which is contained in the view's model. All of the value subviews manipulate their values through the same standard interface protocol, treating the value as though its aspect is value. In this way, any subview can use any value with no customization necessary.

This simplified interface creates a problem, however. A typical model contains numerous state items, each of which is a separate value. The subviews want these values' aspects to be value, but a model cannot contain more than one state item with a particular aspect. A model has to give its state items domain aspects that have reasonable, descriptive names (such as name, address, and phone). So the descriptive aspect names must be translated into the generic value aspect that subviews use.

This translation from the model's descriptive aspect to the subview's generic aspect is performed by a value model. A *value model* links the value subview to the corresponding domain aspect of its view's model, presenting a uniform, generic interface on the widget side and employing whatever interface is necessary on the model side. The value model can also perform any translation necessary for converting the model's state object into a suitable subview value object and back again.

Thus the custom code is now concerned only with setting up the proper value models on top of the relevant model

## Bonus Pull-Out Section

# Object Buyer's Guide Winter '95 Issue

This concise eight-page removable insert is your handy source for shopping for the newest OT-related products and services.

For free, detailed vendor information—FAST—either contact the company directly or return the special **OBJECT BUYER'S GUIDE Reader Service Card**.

Stay current by discovering these useful new products and services.

The next **OBJECT BUYER'S GUIDE** will appear with the March/April Issue.

To Advertise in the  
Object Buyer's Guide,  
please contact

Michael Peck at  
212.242.7447

Next Issue:  
March/April 1995



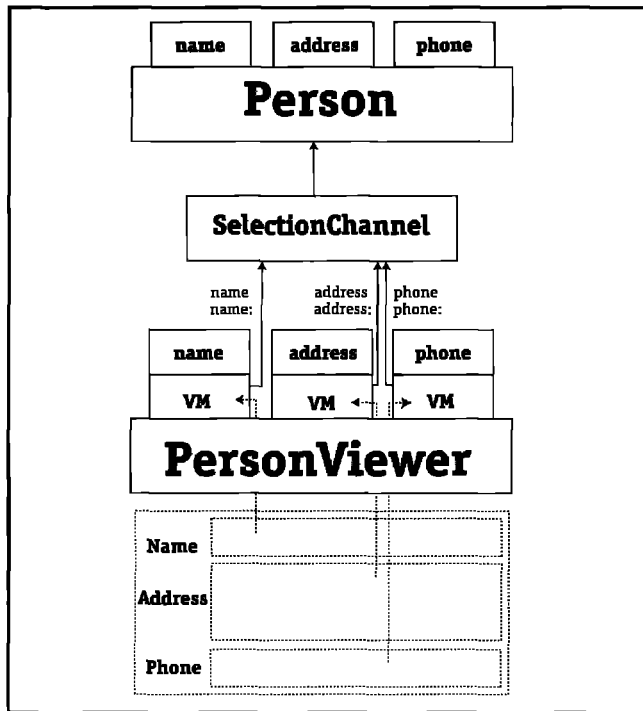


Figure 3. A PersonViewer application model and view interfacing with a Person domain object.

aspects; this moves the customization out of the view and into the model, encapsulating the customization completely within the model. The other custom code concerns the choice of visual widgets and their layout on the view; these particulars can be described in a *specification* that is stored in the model. Objectworks 4.0 was the first release to move the code for view specification into the model. VisualWorks 1.0 abstracted the specification further, storing it not as code but as an array of literals in a *visual spec* method, usually called *windowSpec*. Rather than each model containing code for interpreting the spec, all such code is stored in one model superclass, *ApplicationModel*. Thus no custom view classes are necessary; the models contain all of the custom code and their views are constructed from general-purpose, highly reusable visual widgets.

#### An example

Figure 2 shows the same Person model described earlier, but this time implemented with ValueModels to hold each aspect. In the view, the subviews look the same, but they are implemented using the new value subview framework rather than the old pluggable view framework. Thus each subview always uses the getter and setter messages *value* and *value:*. These messages are no longer sent to the model; instead, each visual widget sends its messages directly to the ValueModel that holds the widget's value.

In this way, the behavior needed to extract a value from the model and the behavior needed to display the value visually are factored into two separate objects. These two objects come from the value model and value subview frameworks respectively. Since any object from one framework will work with any from the other, how the value is retrieved and how it is displayed can be mixed and matched as necessary to produce the desired behavior.

# StaticSQL

StaticSQL is a powerful tool for building high-performance Smalltalk applications with data access to IBM DB2 databases

Combines flexibility and ease of development of dynamic SQL with enhanced performance and security features of static SQL

- Smalltalk/V and IBM Smalltalk
- OS/2       Source Code Included
- Tutorial       No Royalties

## Neva Object Technology, Inc.

1409 Stratford Street, Suite E, Brea, California, 92621, USA  
 Phone: (714) 671-4107 Fax: (714) 256-1916  
 Email: 74077.2656@compuserve.com

Copyright Neva Object Technology, Inc. 1994. All names and trademarks are the property of their respective owners.

#### DOMAIN MODELS AND APPLICATION MODELS

Although the value model translation layer leads to greater visual reuse, there is still a limitation in the reuse of model behavior. Often a particular model can be displayed in several different kinds of views; for example, one view might display the model as a table showing how totals were derived, whereas another might show the same model as totals graphed with annotations. Each view displays only certain parts of the model, allowing the user to picture the model in different ways. But requiring the model to contain code describing every possible view can become burdensome and limit reuse.

Furthermore, although some views display the model's state directly, others require that conversion and translation be performed on the state to generate the display data. For example, the model might not store the totals; instead, it would calculate them for the view. In fact, the sources of the data might be factored into multiple models. Maintenance is complicated when one must try to determine what conversion code is used for which view.

To address these problems, VisualWorks 1.0 divides the model into two pieces: the domain model and the application model.<sup>4</sup> The structure of a *domain model* fits the structure and responsibilities of the real world object it represents. The *application model's* structure fits the structure of the data its view will display.<sup>5</sup> Thus domain models do not have views; instead, the view is attached to an application model that in turn is attached

<sup>4</sup> The motivation for separating the domain model and the application model is discussed briefly in VisualWorks User's Guide.<sup>4</sup>

<sup>5</sup> James Rumbaugh discusses how to derive domain models and application models from use cases.<sup>5</sup>

## Database Solution for Smalltalk/V

A class library for ODBC Database Access



- ODBC 2.0 support
- Automatic class generation
- Native data type support
- online help, source included, no runtime fees

Available for Win16, Win32s, Win-NT, OS/2 and PARTS

"... simple but elegant ..." - Australian Gilt Securities

## Client Server Solution for Smalltalk/V

A class library for Windows Sockets Development



- UDP and TCP Sockets
- Synchronous and asynchronous support
- sample code for remote disk browser app
- online help, source included, no runtime fees

Available for Win16, Win32s, Win-NT



Consulting Services  
Tools for the Smalltalk developer

Tel: 416-787-5290  
Fax: 416-797-9214  
CompuServe: 73055,123  
Internet: lucc@tor.hookup.net

to the domain model. The application model is responsible for converting and translating the domain model's state as necessary to suit the view, and it contains the ValueModels that the view requires. Typically, if different styles of views can be opened on the same domain model, each view style is implemented in its own application model.

By separating the domain model from the application model, the domain model's behavior can be used by multiple application models. Like a client and a server, the application model can delegate many of its tasks to common services performed by the domain model. Often, much of the application model's state is virtual; when accessing messages are sent to the application model, it simply delegates some or all of these tasks back to the domain model. In this way, the view neither knows nor cares how much state is stored in the application model and how much is actually passed straight through from the domain model.

### An Example

To see how useful it is to separate the domain model from the application model, consider a system containing multiple Person objects. With the view built directly on the Person, displaying a different Person requires closing the first Person's view and opening a second identical view. This is inefficient in many ways! But if this behavior is divided into two objects, a Person and a PersonViewer, the code needed to support the view can go in one object and the state to be displayed can go

in another. With this architecture, to switch from one Person to another, the PersonViewer and its view simply switch from the first domain model to the second.

Figure 3 shows the two models in this example. The PersonViewer, an application model, contains the state needed by the view; it derives this state from a Person object, a domain model. Although this simple example shows Person only containing the three aspects that the viewer displays, the application model often displays only a subset of the domain object's state and often translates it heavily to display it. The domain probably consists of several people; the SelectionChannel allows the application model to easily switch from one Person to another.<sup>11</sup>

### CONCLUSION

New frameworks in VisualWorks 1.0 enable customized views to be created with a minimum of custom code. Views are composed of highly reusable value subviews. Each value subview interfaces directly with its aspect in the model via a generic protocol. All custom code concerning the view is stored in the model. Finally, the model's domain state and its view support have been separated into distinct objects so that multiple views can easily share a single domain object. This separation of responsibilities into distinct, collaborating objects leads to well-factored, well-encapsulated code that is not only highly reusable but also far easier to maintain.

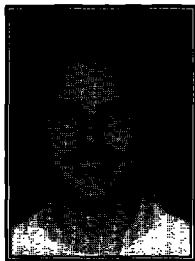
While VisualWorks enables nonprogrammers to develop sophisticated looking views, you can use your programming skills and the new frameworks in VisualWorks to produce equally sophisticated behavior in the model. In the process, you're developing code that is more reusable, better encapsulated, and simpler to maintain.

### References

1. Goldberg, A., and D. Robson, SMALLTALK-80: THE LANGUAGE AND ITS IMPLEMENTATION, Addison-Wesley, 1983.
2. Krasner, G. E., and S. T. Pope, A cookbook for using the Model-View-Controller user interface in Smalltalk-80, JOURNAL OF OBJECT-ORIENTED PROGRAMMING, 1(3):26-49.
3. Leibs, D. J., and K. S. Rubin, Reimplementing model-view-controller, THE SMALLTALK REPORT, Mar./Apr. 1992, 1(6):1-7.
4. VISUALWORKS USER'S GUIDE, Release 1.0, ParcPlace Systems, Inc., 1992, pp. 111-112, 121-122.
5. Rumbaugh, J., Getting started: Using use cases to capture requirements, JOURNAL OF OBJECT-ORIENTED PROGRAMMING, 7(5): 8-12, 23.
6. The discrete charm of the ValueModel, PARCNOTICES, Summer 1993, 4(2):1, 8-9.

**Bobby Woolf is a Member of Technical Staff at Knowledge Systems Corp., where he is developing techniques for using VisualWorks to develop highly reusable and easily maintainable software components. He has also been a Software Engineer at Ascent Logic Corp., where he developed and maintained reusable software components and views in Objectworks. Comments are welcome at woolf@acm.org.**

<sup>11</sup>This selection channel technique is described in ParcNotices.<sup>1</sup> It is employed by RolodexExample in VisualWorks 1.0.



KENT BECK

# Demand loading for VisualWorks

**B**efore I jump into this month's column, I'd like to tell you a little about what I learned at Digitalk's DevCon this year.

### RANT

One thing I learned is that Digitalk is finally coming around to the idea that third-party support is important to their success. The slow growth of the third-party parts market has hurt them, I think, and they want to fix that. Their Partners Program, the third party catalog to be shipped with their product, and their public and private statements at DevCon give me hope that they are coming around. Their past neglect was made painfully apparent by the number of repeat third party vendors showing in booths (three, I think).

The wackos are still around. Despite all of their best efforts to put Smalltalk in a blue suit with wingtip shoes, DevCon is still *the* place to meet interesting folks. I found more odd, interesting, intriguing, stimulating perspectives at DevCon than OOPSLA, the ParcPlace User's Conference (which was a great party for other reasons), and all the commercial object conferences combined. I don't know what it is about V, but it still attracts a fun crowd.

The last thing I learned is more disturbing. I talked to Greg and Bruno from Intellware in Toronto while I was there. You may recall that I mentioned how much they learned when I visited them a couple of years ago. Apparently some readers (not you, of course) read what I had written very differently than I had intended. Greg and Bruno are still taking grief from that one brief comment, as if I had said they were stupid or bad programmers.

Let me be very clear about this. I have a world of respect for them. They had the guts to start their own business on what was then a pretty risky technology, they had the smarts to make it work, shipping an impressive manufacturing application to the automotive industry, and I believe they have the technical talent to go a long way from there. Last but not least, they knew they needed help, so they asked for it and got it. They

Kent Beck has been discovering Smalltalk idioms for eight years at Tektronix, Apple Computer, and MasPar Computer. He is the founder of First Class Software, which develops and distributes reengineering products for Smalltalk. He can be reached at First Class Software, P.O. Box 226, Boulder Creek, CA 95006-0226, or at 408.338.4649 (phone), 408.338.3666 (fax), 707611216 (CompuServe).

really "got" patterns. I expect great things of them in the future.

The lesson for me is that I have to be very careful about what I write. People out there are taking what I write seriously, so I have to take what I write seriously, but without taking myself seriously (the dirty diapers help). The lesson for you is to read, yes, but make your own conclusions. I'm sitting here in my living room/office with a deadline, not coming off a mountain with stone tablets.

I feel better having gotten that off my chest.

### PROBLEM

No patterns this time. Instead, I'll show you some code I wrote for a client who was having performance problems. It is interesting for several reasons:

- It shows how you can use inheritance to separate the design from the implementation
- It shows an appropriate use of a dangerous low-level feature
- It shows how important it can be to understand your underlying Smalltalk implementation

I described this code to Jon Hylands at the Object People. He proceeded to show me an implementation of the same idea for V. Great minds run in the same gutters, eh Jon?

The client and I discovered that we were thrashing Windows virtual memory. The only solution was to reduce the memory requirements somehow. The Stripper begins to address the footprint problem by removing unused classes before deployment (in addition to removing certain classes to satisfy the standard runtime license agreement). Unfortunately, it is not always possible to statically determine which classes will be used at runtime. Aggressive stripping, removing classes that may be used but you think won't be used, further reduces footprint, but at the cost of correctness. You'll get a smaller image, but will it run?

This column describes a complimentary approach to reducing memory footprint- demand loading. Rather than discarding objects that may not be used, demand loading waits until a value is wanted, then loads it from disk. The implementation makes this process transparent to the runtime code. To invoke demand loading, you need only add a step to the standard Stripper that sets up the variables to be so loaded.

### OVERVIEW

When I have a VisualWorks problem that I can't solve, I talk it over with David Liebs. I generally only understand half of what he says, but the half I get is usually enough to get me unstuck. I had tried a couple of different approaches to demand loading without success, and there was a lot riding on an answer for me and the client, so I asked David. He immediately told me the trick that got me going (thanks, David!).

Global, Pool, and Class variables are stored as Associations, where the key is the name of the variable and the value is the value. If you have a method that uses the value of a global variable, the compiled version of that method refers to the same Association as the Dictionary that owns it. For example, if we compile:

```
Object>>foo
```

```
^Foo
```

where Foo is a global variable, we get Figure 1.

## Smalltalk Idioms

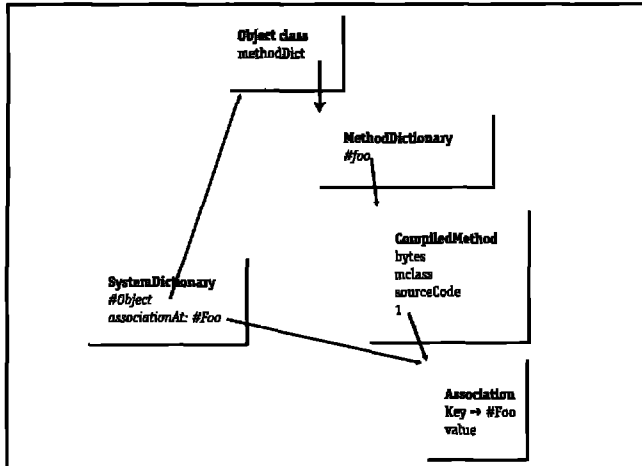


Figure 1. A compiled method referring to a global variable.

In VisualWorks, unbeknownst to you, when you access one of these variables, the dynamic translator sends "value" to the Association to get the value. This gives us the leverage we need to implement demand loading.

The strategy is to replace the standard Associations for demand loaded variables with a LoadingAssociation that overrides "value." If the value is already in the image, the LoadingAssociation just returns it. However, if the value has not yet been loaded, the LoadingAssociation uses a stored file name to retrieve the value before returning it.

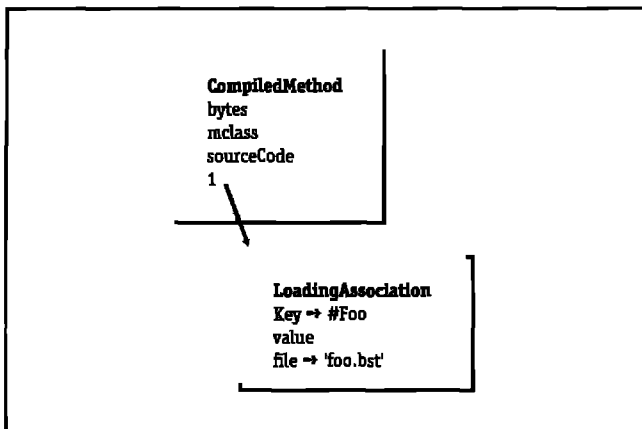


Figure 2. A demand loading global.

Actually implementing this requires a bit more detail, as classes are handled specially by the Binary Object Streaming Service (BOSS). LoadingAssociation provides the framework that ClassLoadingAssociation exploits to load classes.

### LOADINGASSOCIATION

LoadingAssociation is a subclass of Association. It will demand load arbitrary objects, so it could be used for lazily initializing large static data structures. For purposes of demand loading classes, however, it mostly acts as an abstract superclass. Since its implementation is simpler than ClassLoadingAssociation, it is still a good place to begin studying the implementation of demand loading.

```

Class: LoadingAssociation
superclass: Association
instance variables: file
  
```

The instance variable "file" holds a string, which is the name of the file from which the value will be loaded.

When a LoadingAssociation is asked for its value, it first checks to see whether it has a value. If not, it reads the value from the file.

```

LoadingAssociation>>value
value isNil ifTrue: [self readValue].
^super value
  
```

Reading a value requires that the LoadingAssociation open a BOSS reader on a ReadStream on the file, read the value, then make sure the file gets closed.

```

LoadingAssociation>>readValue
| stream |
stream := BinaryObjectStorage onOldNoScan:
self filename readStream.
[value := self readValueFrom: stream]
valueNowOrOnUnwindDo: [stream close]
  
```

Reading the value is placed in a separate method because reading classes will be different than reading arbitrary objects. The implementation here simply reads the next object from the BOSS stream:

```

LoadingAssociation>>readValueFrom: aStream
^aStream next
  
```

The LoadingAssociation comes up with a concrete file name by concatenating the file instance variable with a default directory. The implementation here uses the same directory in which the image file was found. You could use a class variable for the directory to gain more flexibility, or even add an instance variable to each LoadingAssociation so different objects could be found in different directories:

```

LoadingAssociation>>filename
^self directory construct: file
LoadingAssociation>>directory
^Filename defaultDirectory
  
```

Creating a LoadingAssociation is simple. The class message replace: anAssociation file: aString creates a LoadingAssociation, writes the value of anAssociation to disk and removes the reference to the value from the LoadingAssociation. If there are no other references to the object, it will be deallocated by the garbage collector:

```

LoadingAssociation class>>replace: anAssociation file: aString
^self new replace: anAssociation file: aString
  
```

There is also a convenience method to unload a global variable that takes a Symbol as a parameter instead of an Association:

```

LoadingAssociation class>>replaceGlobal: aSymbol file: aString
^self replace: (Smalltalk associationAt: aSymbol) file: aString
  
```

The implementation of replace:file: is a bit tricky.

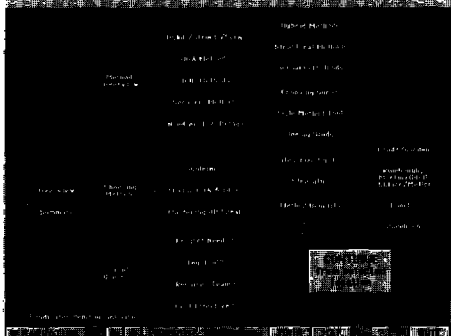
CompiledMethods contain references to the Associations that were found at compile time. Thus, if we want to demand load a global variable, it is not enough to simply remove the Association from the SystemDictionary and replace it with a LoadingAssociation. We have to change all references to the old Association into references to the LoadingAssociation that will replace it. Kids, don't try this at home!

```

LoadingAssociation>>replace: anAssociation file: aString
  
```

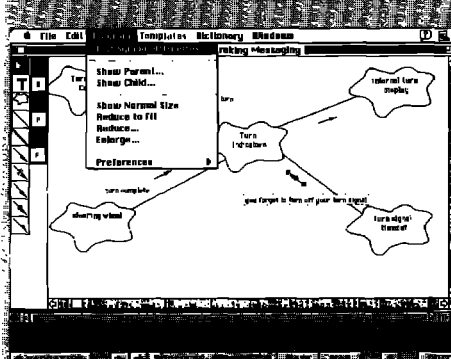
THE NEW WAY TO LEARN O-O METHODS FAST!

Interactive  
Methodology  
Training on a  
CD-ROM



Click on any of the presentations

Click on any of the presentations



Narrated software demonstrations show you how to implement popular O-O methodologies.

Written and narrated by Doug Rosenberg, President of ICONIX Software Engineering, Inc. in Santa Monica, CA.

Written and narrated by Doug Rosenberg, President of ICONIX Software Engineering, Inc. in Santa Monica, CA.

**HOW TO ORDER**

Send order form to:  
SIGS Books, Inc.  
P.O. Box 99425  
Collingswood, NJ 08108-9976

FOR FASTER SERVICE

Fax to 800-951-7097  
Phone 800-777-7777



# An Object Methodology Overview

SIGS Books has harnessed the latest in interactive multimedia and CD-ROM technology to bring you this desktop training seminar entitled **AN OBJECT METHODOLOGY OVERVIEW.**

## WHAT YOU'LL LEARN

- What are the different kinds of O-O methodologies?
- Which methods are strong for Analysis? Design?
- How can you mix O-O methods on a project?
- Is OOD really just "more OOA"?
- Which methods work best for client/server systems?
- What to consider when choosing an O-O CASE tool.

## ACTUAL ON-SCREEN DEMONSTRATIONS OF:

- Rumbaugh Method
- Jacobson Method
- Booch Method
- Coad/Yourdon Method
- How to Combine Different Methods

**15-Day  
Money Back  
Guarantee**

## BENEFITS OF CD-ROM TRAINING

- **Cost-Effective:** Augments or replaces on-site training seminars.
- **Time-Efficient, Convenient and Flexible:** Complete the seminar at your own pace and on your own schedule. You can save your place and resume the training seminar at any time.
- **Fully Interactive:** Simply drag a slider control to repeat or move to any part of a demo or narrated slide...Double-click the text of a bullet to repeat that bullet's narration...Click a button to display an overview of the presentation, from which you can instantly jump to any topic.

**Experience this accelerated training technique. Order today.**

SIGS BOOKS CD-ROM TRAINING ORDER FORM

**YES!** Please rush me **An Object Methodology Overview** for \$995.

I have 15 days to decide whether or not I want to keep the CD-ROM; if not, I'll return it for a full refund, no questions asked.

**METHOD OF PAYMENT**

Check Enclosed ( Payable to SIGS BOOKS )

Macintosh Version  PC Version\*

Charge My:  Visa  Mastercard  AMEX

CardNo \_\_\_\_\_ Exp \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Country/Postal Code \_\_\_\_\_

Phone \_\_\_\_\_

Fax \_\_\_\_\_

U.S. orders add \$5 for shipping/handling; Canada add \$10; Foreign add \$15; NY state residents add applicable sales tax. Allow 4-6 weeks for delivery. \*Available January 1995.

SBCDR

## Smalltalk Idioms

```
file := aString.  
key := anAssociation key.  
value := anAssociation value.  
self unloadValue.  
anAssociation become: self
```

Unloading the object just writes it to disk and removes the reference. It is in its own method because you may want to dynamically unload objects at some point in the future (this capability is not currently used). As currently written, `LoadingAssociation` is directed at reading, not writing, values. You could change this by adding a flag that was set when a `LoadingAssociation` was written to (by overriding `value`):

```
LoadingAssociation>>unloadValue  
self writeValue.  
value := nil
```

Writing a value is analogous to reading one. The inner unwind block is there in preparation for fail-soft value writing (although `unloadValue` still overwrites the `value` instance variable, duh):

```
LoadingAssociation>>writeValue  
| stream oldValue |  
stream := BinaryObjectStorage onNew: self filename writeStream.  
oldValue := value.  
[[self writeValueOn: stream]  
valueOnUnwindDo: [value := oldValue]]  
valueNowOrOnUnwindDo: [stream close]
```

Again, the details of writing the value on a stream are broken out to facilitate `ClassLoadingAssociation`:

```
LoadingAssociation>>writeValueOn: aStream  
aStream nextPut: value
```

**CLASSLOADINGASSOCIATION** `ClassLoadingAssociation` takes the framework provided by `LoadingAssociation` and adapts it to the details of reading and writing classes. All of the difficulties come because global variables are routinely accessed by `BOSS` and the `ClassBuilder`, which will be used to read and write the values of `ClassLoadingAssociation`. Thus, we have to worry about having a `ClassLoadingAssociation` invoked when it is already in the middle of being read or written. The first line of defense is adding an instance variable, `isLoading`, which will be true while the `ClassLoadingAssociation` is loading its class:

```
Class: ClassLoadingAssociation  
superclass: LoadingAssociation  
instance variables: isLoading  
ClassLoadingAssociation class>>new  
^super new initialize  
ClassLoadingAssociation>>initialize  
isLoading := false
```

Writing classes uses the `BOSS` protocol `nextClassesPut`: `aCollection`, which writes a `Collection` of classes. Classes are in a special format so that variable references can be correctly resolved when reading classes in and the classes can be installed in the `SystemDictionary` automatically. The current implementation writes a single class at a time. You could probably get performance improvements by reading clusters of related classes, but the external protocol for writing the classes, and the analy-

sis that would go into creating clusters, would make this solution more complex:

```
ClassLoadingAssociation>>writeValueOn: aStream  
aStream nextPutClasses: (Array with: value)
```

There are two pieces of system code that we have to worry about for re-entrancy. The first is the `ClassBuilder`. When it runs, it looks to find the current class to see if it should modify the existing class or create a new one from scratch. If there is no class (which is what we are simulating), the `ClassBuilder` does the right thing for us. But to convince the `ClassBuilder` that there is no class, the `ClassLoadingAssociation` has to remove itself entirely from the `SystemDictionary`. If it just returns nil (or some other bogus value), the `ClassBuilder` will complain that you are replacing a nonclass value with a class:

```
ClassLoadingAssociation>>readValueFrom: aStream  
isLoading := true.  
Smalltalk removeKey: key.  
^[aStream nextClasses first]  
valueNowOrOnUnwindDo:  
[isLoading := false.  
Smalltalk  
removeKey: key;  
add: self]
```

This method embodies yet more trickiness. Recall that all of the `CompiledMethods` in the system that refer to a variable have references to the (in this case `ClassLoading`) `Association`. After we have loaded the class, there is a new `Association` in the system referred to by the `SystemDictionary`. We have to remove the new `Association` entirely and put the `ClassLoadingAssociation` back in the `SystemDictionary` so all those references are consistent.

The second piece of system code that we have to worry about is `Dictionary`. In reading in a class and resolving its references, and in executing the "removeKey"s in `readValueFrom`: above, the `SystemDictionary` (a subclass of `Dictionary`) sends messages to its `Associations`. Thus, while a `ClassLoadingAssociation` is loading, it must avoid restarting the loading process:

```
value  
isLoading iffTrue: [^Object new].  
^super value
```

Second (and this is the trickiest part of this implementation), when `BOSS` is writing a class that accesses `Pool` dictionaries, it uses `keyAtValue`: to find the name of the `Dictionary`.

Unfortunately, `Dictionary>>keyAtValue`: sends `value` to each `Association`. If we have unloaded some `ClassLoadingAssociations` already, this causes them to be loaded again, negating the effects we are looking for. A little judicious slicing removes the need to send `value` in executing `keyAtValue`: First, we have `Associations` test whether their `value` is identical to a given `Object`:

```
Association>>valueIs: anObject  
^value == anObject
```

Second, we invoke this method instead of sending `value` in `keyAtValue`: Rather than replace the implementation in `Dictionary`, I decided to override the implementation in `SystemDictionary`, to reduce the possibility of any conflicts:

```
SystemDictionary>>keyAtValue: value ifAbsent: exceptionBlock
```

```
self associationsDo:
    [:each | (each values: value) ifTrue: [^each key]].
^exceptionBlock value
```

**USING DEMAND LOADING** All that remains is to create `ClassLoadingAssociations` as part of stripping. Stripper provides the method `postStrip`, which you can modify to unload classes you won't always need right away. Here is an example that unloads the business graphics classes:

```
Stripper>>postStrip
| classes |
classes := Smalltalk select:
    [:each |
        each isBehavior
            and: [(each name indexOfSubCollection: 'BG_' startingAt: 1)
                > 0]].
```

`ClassLoadingAssociations unloadClasses: classes` `UnloadClasses` writes to a contiguous set of file names. You should only invoke it once per strip to avoid generating the same file names more than once:

```
ClassLoadingAssociation class>>unloadClasses: aCollection
| classes |
classes := SystemUtils sortForLoading: aCollection.
classes
    with: (1 to: classes size)
    do:
        [:eachClass :eachIndex |
            self
                replaceGlobal: eachClass name
                file: eachIndex printString , 'c.bst']
```

You will have to experiment with this facility to find the right time/space tradeoff for your application.

## CONCLUSION

There are several interesting points to make about the code above:

You have to understand the virtual machine. As much as we'd like this not to be true, when you are writing system code (I'd certainly call `LoadingAssociation` system code, not application code), the more you know about the implementation the better.

The design is subtly platform specific. Jon Hyland's demand loader for V takes a very different approach to solving the same problem. Because `Digitalk`'s implementation of `become:` is so much slower than `ParcPlace`'s, he uses `doesNotUnderstand:` to forward all messages to the class.

You can use inheritance to separate design from implementation. In general, I am against using inheritance to express any important concept. In this case, though, it came in very handy. You can read the `LoadingAssociation` code and understand the design. All of the nasty details about how loading classes interferes with the system are hidden in the subclass. A similar use of inheritance is to create a simple, general superclass and an optimized subclass.

As always, I'm interested in your comments. My column on the testing framework actually garnered me half a dozen email messages. If you're using it for anything interesting, let me know. ♀

**SMALLTALK**  
**SOLUTIONS**  
**95**

Don't  
miss  
Smalltalk  
Solutions '95

## Special Conference Preview Section

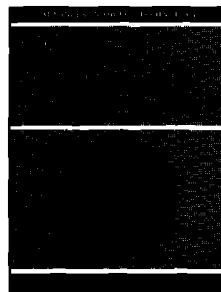
(See page 33)

February 21-24, 1995  
Omni Park Central  
New York, NY

AVAILABLE FROM SIGS BOOKS

## OBJECT LESSONS

by Tom Love



(ISBN: 0-9627477-3-4)

**\$29.00**

To order a copy of  
**Object Lessons**  
call (212) 242-7447  
Customer Service Dept.

Tom Love reveals the absolute do's and don'ts of adapting and managing object-oriented technology. Love's "trade secrets" demonstrate the how-to's of putting theory into practice. You will benefit from an insider's view of major companies' successes and failures in O-O projects. Save countless hours and dollars by learning from their costly mistakes.

*This book is written for those making decisions on design and management of large-scale commercial object-oriented software.*

**SIGS**  
**BOOKS**

Available at selected book stores. Distributed by Prentice Hall.



ALAN KNIGHT

## Safety and inheritance

I'M writing this column just after returning from OOP-SLA. This was a very thought-provoking experience, so rather than combing the net for postings of interest, I'm just going to talk about a couple of topics that sparked my imagination.

### SOFTWARE SAFETY

One of the most interesting talks at the conference had nothing to do with objects. It was Nancy Levenson's invited talk on "System Safety and Software Design." Like many in the Smalltalk community, my work has been mostly in research and business applications, so I hadn't had much exposure to the issues involved in safety-critical software systems. I found this a very frightening talk, particularly since many of the points were illustrated with anecdotes about things blowing up and people dying.

There were two main points. The first is that people have far too much faith in computer systems. More and more safety-critical systems are being turned over to computer control, and faith in the computer can lead to insufficient "traditional" safety precautions. The second point is a basic one in safety and reliability engineering, but new to me. It is the critical difference between safety and reliability.

A reliable system is one in which failures are rare. Measures like testing, code inspections, static type systems, formal methods, and assertions are all attempts to make our programs more reliable. Reliability is a good thing, but it shouldn't be confused with safety.

A safe system is one in which failures rarely cause bad things to happen. While it's true that a system that is 100% reliable is also safe, we all know that such systems don't exist. Although in theory a computer system can be completely reliable, in practice all software has bugs. In pursuit of the unattainable goal of perfect reliability, we too often fail to consider the consequences when our systems do fail.

### Safety and Smalltalk

This talk didn't even mention Smalltalk, but it certainly made me think about it. In particular, it made me think about an argument often employed by critics of dynamic typing. It's often said that dynamic typing is inherently unsafe and that

Alan Knight is a consultant with The Object People. He can be reached at 613.225.8812, or by email as knight@acm.org.

only by locking ourselves into the restrictive frameworks of current static typing systems can we make our systems safer. This often takes the form of cheap shots about "message not understood" errors in safety-critical systems.

For example, digging through my archives, I find that no less a luminary than Bertrand Meyer, inventor of the (statically typed) Eiffel language, wrote, in response to a comment about runtime checking:

Isn't run time perhaps a little late?

Men and women of the second battalion! Because of a small programming error, the Patriot missile meant for the other side is actually coming back and will blow up all of us in about 22 seconds. You will be glad to know, however, that thanks to the runtime type checking mechanism my screen just displayed MESSAGE NOT UNDERSTOOD. So there is nothing wrong with our software technology, and we can all die happy.

Long live our country! Long live dynamic type checking!

I've been known to respond to this kind of thing with my own cheap shots about locking yourself in a straightjacket to make it harder to shoot yourself in the foot. Nancy Levenson's talk, however, pointed out a much deeper fallacy in these arguments. They talk a lot about safety when what they really mean is reliability. This is easily seen from a couple more examples. Robert Martin (rmartin@rcmcon.com) wrote:

The issue is type safety. In C++, to get the undefined behavior mentioned above, you must explicitly break the type rules (downcast). In Smalltalk, there are no type rules to break. In both cases the result is a runtime fault. ... While I agree that Smalltalk behaves a little better than C++ during this runtime fault, I also assert that the difference doesn't amount much to the customer whose application just crashed.

And in a very concise statement of these ideas, Andrew Koenig (ark@research.att.com) wrote:

Message not found is Smalltalk's way of spelling "core dumped"

### Message not Understood = core dump

The critical assumption here is that there is no significant difference between a "message not understood" exception and the type of error that provokes a core dump in C or C++. This is simply not true. While both of these situations are equally serious in terms of reliability, the difference is in safety. In C or C++ we have few options for dealing with such an event and no guarantees that the machine and operating system state have not already been corrupted to the point where recovery is impossible. In Smalltalk, the system raises an exception as soon as the error occurs, and allows us to deal with that exception however we like. We can have some confidence in the system state, and we have a metaframework that allows us to examine and modify that state in more detail than is possible in most languages. If the situation is unrecoverable, we still have the ability to shut down in a disciplined way.

Does that mean I'd write a safety-critical system in Smalltalk? At this point I don't know. The main thing I'm sure of is that I don't know enough about writing safety-critical systems and that I'd need to learn a great deal more about the subject before I'd make any serious decisions. I'd be nervous writing a safety-critical system in Smalltalk, but I'd be nervous doing it in any language that I know of. I'm quite sure I wouldn't even attempt it in C or any of its derivatives.



# Announcing... a book whose time has come.

Part of the *Advances in Object Technology* Series  
from SIGS Books...

## Objectifying Real-Time Systems

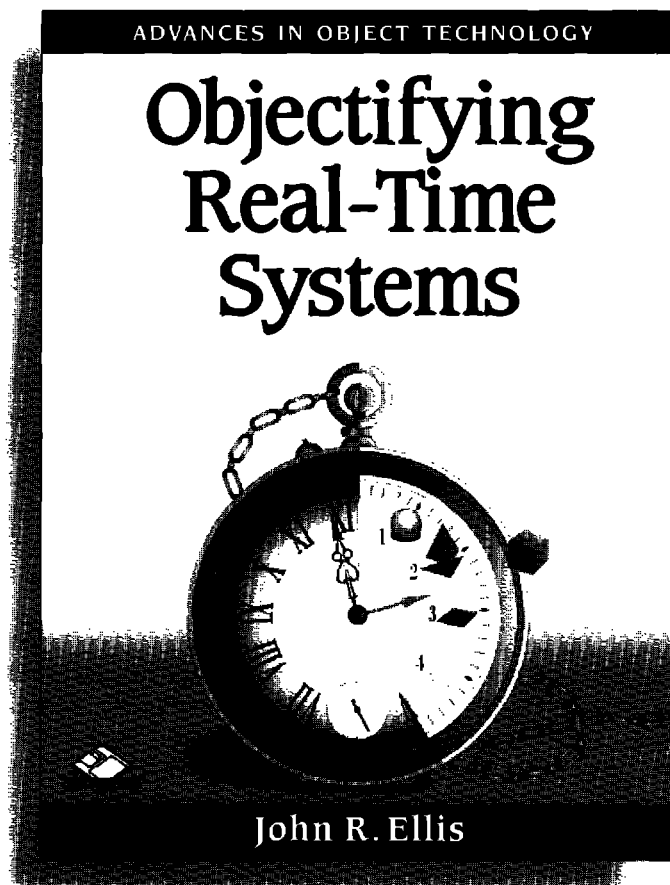
by John R. Ellis

**Objectifying Real-Time Systems** presents guidelines for creating a real-time information processing system Requirements Model. The methodology presented is an evolution of popular Real-Time Structured Analysis (RTSA) techniques into object-based Real-Time Object-Oriented Structured Analysis (RTOOSA). Although predominately concerned with identifying objects and their behaviors, this book adapts the notation of RTSA and capitalizes on its key products by using object technology.

**Objectifying Real-Time Systems** first introduces the reader to the basic concepts of O-O programming, establishing a common understanding of "objects". Then, the reader learns how to create each of six RTOOSA Requirements Model products and learns how these products interact to allow verification of a complete and consistent model. The author draws on twenty-seven years of development experience with real-time systems for examples.

### Who should read this book?

Anyone interested in developing object-based real-time systems. The reader does not need to possess a background in OT, but should have a solid grasp of RTSA.



ISBN: 0-9627477-8-5  
(525 pages with diskette)

### About the author...

John R. Ellis is a senior systems engineer at Harris Corporation's Government Aerospace Systems Division, where he has worked on a variety of commercial, military and NASA embedded real-time software systems for the last 15 years.

Available at selected bookstores.  
Distributed by Prentice Hall.



Please rush me \_\_\_\_\_ copies of  
**OBJECTIFYING REAL-TIME SYSTEMS**  
at the low rate of \$44 (including diskette)

### METHOD OF PAYMENT

- Check Enclosed (Payable to SIGS Books)  
 Charge My:  Visa  Mastercard  AmEx

Card # \_\_\_\_\_ Exp \_\_\_\_\_

Signature \_\_\_\_\_

U.S. orders add \$5 for shipping/handling; Canada add \$10; Foreign add \$15; NY State residents add applicable sales tax. Please allow 4-6 weeks for delivery.

ISBN: 0-9627477-8-5

### SHIP TO

Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Country/Postal Code \_\_\_\_\_

Phone \_\_\_\_\_ Fax \_\_\_\_\_

**TO ORDER- MAIL to:** SIGS Books, P.O. Box 99245, Collingswood, NJ 08108-9970  
**FAX to:** 609/858-2007 or **PHONE:** 212/242-7447

### INHERITANCE

The conference also got me thinking about the use of inheritance, one of the most oversold and misrepresented aspects of object-oriented programming. What does inheritance really represent? Is it:

- the “is-a” relation, which is rather vague but has something to do with mathematical subsets and “real-world” relationships
- subtyping or substitutability—being able to use a subclass anywhere that a superclass is expected
- re-use of superclass code and/or representation in subclasses
- all of the above

There seems to be a widespread impression that inheritance represents all of the above simultaneously. It's not true, and in fact these uses actually contradict one another. While “is-a” is commonly used for inheritance as a modeling concept, it simply doesn't map to implementations. A square most definitely “is-a” rectangle, but Square as a subclass of Rectangle won't work well for either subtyping or code sharing.

If your code isn't going to get very muddled, you really have to pick one of these uses as primary. In Smalltalk, it's code sharing. This has very significant consequences for the attitudes towards inheritance. While code sharing is nice, it's not essential, and it's not the foundation of our thought processes.

Alan Kay<sup>1</sup> wrote:

... since things can be done with a dynamic language that are difficult with a statically compiled one, I just decided to leave inheritance out as a feature in Smalltalk-72, knowing that we could simulate it back using Smalltalk's LISPlike flexibility.

And Kent Beck<sup>2</sup> wrote:

**Inheritance is the least important of the three facilities that make up objects. You can do valuable, interesting object-oriented programming without using inheritance at all.**

Inheritance is certainly very convenient, and I'd hate to have to do without it, but the exact form of the inheritance hierarchy isn't all that significant, and we expect to be able to change inheritance relations over time without causing major disruptions.

The general OO hype, however, puts a great deal of emphasis on inheritance. Partly in rejection of this hype, Smalltalkers often deny any additional significance to inheritance at all.

For example, Kent Beck also wrote:

**Inheritance is about code sharing, period.**

I've never been completely comfortable with this view either. While I feel sure that code inheritance is neither subtyping nor “is-a,” I do think there's something more to it than just an implementation convenience. When I look at an inheritance hierarchy that seems right, there's usually some sort of relationship beyond just re-using code. When I look at a hierarchy that intuitively seems wrong, I don't think it's just because the code sharing is suboptimal. I have a deeper feeling that those things don't belong together. I muddled along with this feeling for a while, sure that there was something more to inheritance, but not knowing what to call it until I came across a very interesting idea.

### Inheritance is not re-use

Richard Gabriel's JOOP (JOURNAL OF OBJECT-ORIENTED

PROGRAMMING) column is one of the most consistently interesting commentaries on OO issues. While I often disagree with his conclusions, his ideas are always provocative. In one discussion on the use of abstraction<sup>3</sup> I came across a particularly interesting idea about the nature of inheritance.

Inheritance is often described as being code reuse, which is sort of true, but not completely accurate. This column had a much better description of inheritance as code *compression*. Using inheritance allows us to remove redundancy, expressing the same idea in less code.

The critical point is that compression is not necessarily good. While it reduces the amount of code, it introduces coupling between superclasses and subclasses, making the code more brittle. If we make a change to a superclass and it should be propagated to the subclasses, then this coupling was good. If we don't want the change propagated then the coupling was bad.

This suggests a metric for inheritance. Two things should be related by inheritance when we expect them to change together. Why would we expect them to change together? Because there is some kind of relationship between them. This provides a justification for the idea that superclasses and subclasses should be related, even though all we're trying to do is share code. What it doesn't do is prescribe the nature of that relation. We might expect that they share some responsibilities, represent similar abstractions, or just that one is an incremental modification of the other. Ultimately, though, it's just a heuristic, and we should be prepared to abandon the inheritance relation if it's doing more harm than good.

This is also interesting for the perspective that it puts on multiple inheritance. In static languages where polymorphism depends on inheritance, multiple inheritance is an absolute necessity. Those who want inheritance to reflect “is-a” relations want multiple inheritance because “Spoon must clearly inherit from KitchenUtensil, MusicalInstrument, InventoryItem, and ElectricalConductor.”

If, on the other hand, the decision on how to organize our inheritance hierarchies is merely a trade-off between compression and coupling, then things look different. Use of multiple inheritance becomes an implementation decision. If it can significantly compress our code without making it too brittle, that's good. On the other hand, MI requires conflict resolution logic, which increases the complexity, and significant use of multiple inheritance can make the code more brittle. In cases where the amount inherited is small, this may actually cause more problems than simply copying a small amount of code. In any case, it is simply a decision with its own tradeoffs, not an essential part of all things object-oriented. This also reflects the common Smalltalk view that while MI might occasionally be convenient, overall it doesn't seem worth the trouble. ♀

### References

1. Kay, A. The early history of Smalltalk, HISTORY OF PROGRAMMING LANGUAGES (HOPL-II) CONFERENCE PROCEEDINGS, p. 69.
2. Beck, K. Inheritance: The rest of the story, THE SMALLTALK REPORT, 2(9):15.
3. Gabriel, R. Abstraction descant, part 2, JOURNAL OF OBJECT PROGRAMMING, 6(2):14.



JAY  
ALMARODE

# Multi- user Smalltalk

The original Smalltalk environments, and most Smalltalks today, are single-user systems. When developers or users start up Smalltalk, they create a single process that reads or writes object memory (the image) stored in a single file. To make objects persist over time, all of object memory is saved in one monolithic operation. With this architecture, the only way to implement client/server applications is to use some other means to share objects across multiple users in a heterogeneous network. In most cases, this has meant using Smalltalk as a front end to a relational database or other persistent data store, or to use remote object communication to send messages to objects in other Smalltalk images. In the first approach, the relational database provides persistent storage and sharing of object state, but not object behavior. In the second approach, object behavior is distributed and duplicated across multiple Smalltalk images.

Storing objects in a relational database has many drawbacks, as described in Loomis.<sup>1</sup> One of the main pitfalls of this approach is the impedance mismatch between the persistent data model and the object model provided by Smalltalk. This mismatch requires complex and hard to maintain mapping code between Smalltalk and the relational database. For example, to store objects in a relational database, objects must be decomposed, or "flattened," for storage into tables, and then reconstituted into objects when needed by the application. This two-way transformation impacts performance, and when the schema is modified, this mapping code must be updated as well.

There is another, more subtle, drawback to this architecture as well. In this approach, behavior only resides in the applications. This means that when an object's behavior is changed or extended, all applications must be updated. With this architecture, the behavior of objects is not encapsulated in a single location, but is distributed and duplicated throughout all applications. This seems exactly opposite of what object technology is supposed to provide! For example, imagine an application where financial consultants try out various investment strategies based upon

the impact of taxes. Tax laws in the United States are very dynamic, changing yearly and sometimes quarterly. When the behavior of calculating taxes for a particular type of investment changes, then every installation of the application must be updated with the changes. In this architecture, there is no central, shared repository of object behavior.

However, a new model of application development is emerging. In this model, the domain of objects is persistent and accessible by multiple users. Object memory is a global resource that is utilized by many users, and behavior is maintained and executed in the database, not just in applications. This model merges the best of languages, databases, and CASE tools to create a new way to build, deliver, and maintain applications. Smalltalk is uniquely positioned to exploit this new paradigm because of its total object-orientation, incremental compilation, interpreted execution, and automatic storage reclamation. The increasing acceptance of Smalltalk in financial, manufacturing, and engineering organizations illustrates that companies are starting to understand they need these features to solve their problems. The productivity of Smalltalk, coupled with a client/server architecture to access a repository of objects provides the means to easily build distributed applications, to deliver and install those applications in a timely fashion, and to maintain and extend those applications without disruption to the system.

Having a single object identity domain that is shared by multiple users makes implementing client/server applications quite different than implementations using relational databases (or other persistent stores) or remote object messaging. This is exemplified by the implementation of a shared set of employee objects using each technique. Using a relational database, a typical implementation might associate the set object with a single table in the database, which contains records for each employee. With this design, each unique set would need to know the name of its corresponding relational table. Another alternative is for a single relational table to contain records that map a unique set identifier to a unique employee identifier. This supports multiple sets where an employee can be contained in more than one set. In either design, an application developer must write mapping code that populates a set by making SQL calls to get back records, then constructing employee objects from the basic data types contained in the records. The implementer must make some provision for the situation where the employee object already exists in the image so that the set contains the unique employee object, rather than making a duplicate. Otherwise, multiple employee objects represent the same actual employee and the advantages of object identity are lost.

In addition, some provision must be made for updating the set when another user has added or removed an employee object (by either adding or removing a row in the table). This requires the application developer to devise some means to determine what has changed between the set of employees in the image and the table of

Jay Almarode can be reached at [almarode@slc.com](mailto:almarode@slc.com).

## Getting Real

records in the database. There are a number of ways this might be done, none of them particularly efficient in keeping the set of employee objects up to date (remember, each SQL call involves a round trip to the server). I'm certainly willing to listen to any solutions that readers might have to do this efficiently.

Finally, the Smalltalk programmer must write code so that when an employee is added or removed from the set, the appropriate SQL call is made to add or remove a record from the table. This probably means that the programmer creates a subclass of Set that overrides the add: and remove: methods (and all the appropriate variations) to generate the SQL statements. This is further complicated if the set can contain different kinds of objects. To implement a shared set using a relational database, the programmer cannot just reuse the existing Set class, but instead must do quite a bit of work.

Another possible way to implement a shared set in Smalltalk is to allow objects to communicate with one another remotely, i.e., objects in one Smalltalk image can send messages to objects in another Smalltalk image. This idea is not new; Bennett<sup>2</sup> describes a distributed Smalltalk, and Hewlett Packard currently sells a distrib-



*Reading and writing patterns gives you a way to communicate your design knowledge with others...*

uted Smalltalk that is compliant with CORBA.<sup>3</sup> With this architecture, there are two approaches to implementing a shared set of employees.

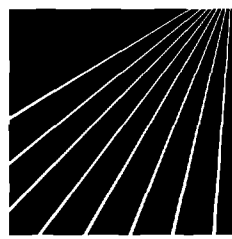
One approach is to designate one Smalltalk image to contain the set, and other images communicate with it to add, remove, or query the contents. Another approach is to duplicate the set in all images that need it, and write code that keeps them in all in synch. With the first approach, the set may contain both local and remote instances of employees. This could lead to performance problems if an application wants to iterate over the contents of the set to generate a list of addresses (an instance variable of an employee). In this case, iterating over all employees would cause remote messages to be sent to all Smalltalk images that have an employee in the set. This makes the time to iterate over the set unpredictable because it depends upon what is in the set and where it is located. And what if one of the Smalltalk images has died? This can be overcome if we require that the set can only contain local instances of employees. However, this

means that all messages to employees must be executed in the image that contains the set of employee objects. With a large number of users, this image quickly becomes the bottleneck. The latter design in which the shared set is duplicated in many Smalltalk images provides more balanced load sharing and fault tolerance. With this design, the programmer must make decisions about when and how to keep the shared sets in sync, i.e., so they all contain employee objects representing the same actual employees. This requires that the image know about any other Smalltalk image that contains the duplicated set and be able to accept new images when they duplicate the set.

So what does a multi-user Smalltalk look like? The key characteristic is that object memory is accessible by multiple users. Rather than starting up a single process that manages object memory by itself, users start a process that coordinates with a server to access object memory. Object memory is always on-line and available; users just establish a connection to it. A versatile client/server architecture for Smalltalk allows the image files (possibly more than one), the server, and multiple clients to be allocated anywhere in a heterogeneous network of workstations. Smalltalk's traditional platform independence is extended to include location independence as well. A client does not need to know where the image is located, it just needs to know how to connect to the server.

To implement a shared set of employees in a multi-user Smalltalk, all the programmer must do is to use the existing Set class. The set by its very nature is shared and is kept in sync between various users by the underlying transaction mechanism. The Smalltalk programmer is able to use the existing protocol for sets without modification. In addition, the behavior of the set is shared as well. For example, if the programmer builds a subclass of Set that performs additional constraint checking before elements are added, this behavior is shared by all users and does not have to be duplicated. The key difference between a multi-user Smalltalk and the architectures described earlier to build a client/server application is that a multi-user Smalltalk has a single-object address space; the other approaches have multiple-object address spaces that must be mapped into and kept in sync.

With multi-user Smalltalk, various responsibilities are distributed between the client, server, and possibly other processes that make up the entire Smalltalk system. The server acts as a global resource coordinator. It must synchronize critical activities to make sure object memory remains consistent. It must make sure that when multiple clients attempt to write their changes to object memory, it is done in such a way to ensure that changes do not conflict with one another. In some cases, this means denying a client the ability to write its changes to object memory. The server is also responsible for allocating key resources. For example, object identifiers (oops) must be unique across the entire domain of objects. A client cannot create oops since it cannot guarantee that other



# SIGS

CONFERENCES

## Success Depends on Knowledge

SIGS Conferences produces software educational events designed to increase your productivity and profitability. All classes are product neutral—you'll never receive a product pitch or marketing hype when you register for these events.

### Don't miss these opportunities to:

- ◆ Learn directly from recognized industry experts, language and methods originators, and well-known book authors.
- ◆ Advance your personal productivity.
- ◆ Demo new products and have your questions answered by leading vendors.
- ◆ Exchange ideas and new technologies with an international network of your peers.

**Jan 30–Feb 3, 1995**



Objekt-orientiertes Programmieren  
**OOP '95**  
MÜNCHEN

**Featuring C++ World**

The Sheraton Hotel  
Munich, Germany  
**Germany's Premier Event**

**February 21–24, 1995**



**SMALLTALK SOLUTIONS 95**

Omni Park Central  
New York, NY  
**Where All the Talk is Smalltalk**

**March 20–23, 1995**

**XWorld**  
featuring



**CROSS PLATFORM STRATEGIES**

NY Marriott Marquis, NY, NY  
**Software Portability Solutions**

**May 8–12, 1995**



**SOFTWARE DEVCON '95**  
*Neue Wege der Software Entwicklung*

Rhein-Main-Hallen  
Wiesbaden Germany  
**The German Development Conference and Exhibition**

**June 5–9, 1995**

**Object**  
E·X·P·O  
THE NATIONAL CONFERENCE & EXPOSITION

New York Hilton & Towers  
New York, NY  
**The Most Important Event Devoted to OT**

**September 25–29, 1995**

**Object**  
E·X·P·O  
*europa*

Queen Elizabeth II Conference Ctr.  
London, U.K.  
**Europe's Most Comprehensive OT Event**

**Oct. 30–Nov. 3, 1995**



**C++ WORLD**

The Users Conference and Exhibition  
Fairmont Hotel  
Chicago, IL  
**The Largest C++ Event in the World**

**Yes!**  
I'd like more information on...

- Exhibiting at the following events
- Attending the following events
- OOP '95 *Featuring C++ World*
- Smalltalk Solutions
- XWorld *featuring Cross Platform Strategies*
- Software DevCon
- Object Expo
- Object Expo Europe
- C++ World

Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

Phone \_\_\_\_\_

Fax \_\_\_\_\_

Please return to: SIGS Conferences  
71 West 23rd Street, NY, NY 10010  
Phone: (212) 242-7515 Fax: (212) 242-7578

## Getting Real

clients are not creating the same oops. The server must allocate oops for clients to be used for new object creation. Another example is locking an object. If a client desires to prevent other clients from reading or writing a particular object, it must obtain a lock through the server, who can then deny access to other clients.

The client process is the primary interface to the user application. It is responsible for providing a consistent view of objects to the application. It must coordinate access to object memory, including initiating disk activity and managing private and shared caches of objects. The client is where the execution engine, i.e., bytecode interpreter, executes Smalltalk methods. Using generational scavenging techniques, the client is able to perform some garbage collection during the course of application execution. However, due to the distributed nature of client/server Smalltalk, some garbage collection is probably best performed using mark-sweep algorithms by a dedicated client.

So what are the advantages of a client/server Smalltalk accessible by multiple users? Application developers no longer need to write complex mapping code to store the state of objects into some other external data store. Object memory itself is the persistent store. This offers the advantage that the behavior of an object is global and shared, as well as the state of an object. When changes are

made to an object's behavior, the change is manifested to all users of the object immediately. You do not have to file in new code that makes up your application; you just run your application and experience the new behavior that the application developer has made available.

In the investment application described earlier, when tax laws are changed, an application developer writes new methods that reflect the new laws. After private testing and validation, these methods are made public and become immediately available to end-users. Of course, sharing objects means that Smalltalkers must do things differently than they have in the past.

My next column will describe the issues that arise when constructing Smalltalk applications where multiple users modify the same shared objects and how these conflicts can be avoided. ♀

### References

1. Loomis, M. E. S. Hitting the relational wall, *JOURNAL OF OBJECT-ORIENTED PROGRAMMING*, Jan. 1994.
2. Bennett, J. The Design and implementation of distributed Smalltalk, *OOPSLA 1987*.
3. Eastman, J. The HP distributed Smalltalk IDL language binding, *THE SMALLTALK REPORT*, 3(3), Nov.-Dec. 1993.

# Call for Writers



*is seeking expert reports, tutorials, and technical papers. Articles should be instructive, product-neutral, and technical.*

Submit papers, discuss story ideas, or request  
Writers' Guidelines from:

John Pugh and Paul White, Editors

THE SMALLTALK REPORT

855 Meadowlands Dr. #509, Ottawa, ON K2C 3N2  
613.225.8812 (v), 613.225.5943 (f)  
streport@objectpeople.on.ca

## Editorial topics include:

### Applications

- Commercial, engineering & scientific applications
- Applications frameworks
- Project management
- Vertical (application) and horizontal (system) class libraries
- Portability issues
- Object library management

### Project management

- Rapid prototyping
- Version management
- Application management
- Team organization
- Organizing for reuse
- Introducing Smalltalk into an organization

### Tools

- User interface builders
- Object editors
- Application development tools
- Project management tools
- CASE tools

### Language issues

- Inheritance
- User interface paradigms
- Concurrency
- Persistent objects and databases
- Distributed Smalltalk issues
- Performance issues
- Typing
- Metalevel programming

*(Competitive stipend paid)*

## Product Announcements

Product Announcements are not reviews.

They are abstracted from press releases provided by vendors, and no endorsement is implied.

Vendors interested in being included in this feature should send press releases to

THE SMALLTALK REPORT,  
Product Announcements Dept., 885 Meadowlands Dr., #509  
Ottawa, ON K2C 3N2, Canada,  
613.225.8812 (v), 613.225.5943 (f).

### ObjectSpace, Inc. Announces New Object-Oriented Software Product Line that Supports Smalltalk

ObjectSpace, Inc., has announced a new product line that includes ObjectSockets and ObjectMetrics.

ObjectSockets is for Smalltalk developers using TCP/IP communications, and ObjectMetrics gathers object-oriented metrics for Smalltalk developers.

ObjectSockets is a class library with more than 40 classes representing all aspects of TCP/IP communications, including TCP sockets, UDP sockets, IP addresses, hosts, networks, protocols and services. It hides the details of complicated WinSock API calls, error detection, and cryptic flags behind a layer of object-oriented simplicity. This makes programs easier to build, understand, and maintain.

ObjectMetrics is one of the first products that provides a solution to obtaining valuable metrics that indicate problem areas such as overly complex code, unnecessary coupling between modules, and programmer productivity. Additionally, ObjectMetrics utilizes a clear, intuitive graphical user interface, making the selection of target classes, metrics, and output options easy.

ObjectSpace, Inc., provides consulting, training and products to assist in the adoption of object technology. The company's mission is to provide clients with the products and the foundation of knowledge and experience required to successfully implement object-oriented systems.

**ObjectSpace, 14881 Quorum Dr., Ste. 400, Dallas, TX v. 214.934.2496 (v), 214. 663.3959 (f).**

### HP Distributed Smalltalk 4.0 to Support Object Technology International's ENVY/Developer

Hewlett-Packard Company has announced that it plans to develop the extensions needed to allow HP Distributed Smalltalk 4.0 to support Object Technology International's ENVY/Developer team programming environment. The combination of ENVY/Developer with HP Distributed Smalltalk will give customers a solid engineering base for developing CORBA-standard distributed applications. The integration of ENVY/Developer's configuration management control with HP Distributed Smalltalk's industry-leading distributed development tools, provides such specific technical benefits such as versioning and configurations of Smalltalk code and IDL repository modules; and convenient implementation sharing between developers through ENVY/Developer's repository.

HP Distributed Smalltalk 4.0 software is a complete implementation of the Object Management Group (OMG) CORBA specification for object communication in a distributed object system. HP Distributed Smalltalk is built upon and extends ParcPlace System's VisualWorks 2.0 software to make a development environment for the creation and deployment of distributed-object client and server applications.

Distributed Smalltalk is available from HP for HP-UX, SunOS, Solaris and IBM AIX UNIX system-based platforms, as well as Windows 3.1, Windows NT and OS/2 platforms.

**Hewlett-Packard Company, 3404 E. Harmony Rd., Ft. Collins, CO 80525, 408.447.4722 (v).**

### UniSQL Announces Development of Interface to ParcPlace's VisualWorks

UniSQL, Inc., reached agreement today with ParcPlace Systems, Inc. to develop a seamless interface between ParcPlace's VisualWorks object-oriented application development environment and UniSQL's next-generation object-oriented database management systems. The new interface will improve connectivity between VisualWorks and the UniSQL/X Database Management System (DBMS) and the UniSQL/M Multidatabase System (MDBS), enhancing development of mission-critical applications with large-scale object-oriented and relational requirements.

Through this new interface, VisualWorks users will be able to more easily access the benefits of UniSQL's database management systems. The UniSQL/X DBMS and the UniSQL/M MDBS provide persistence to applications, allow users to manage multiple heterogeneous databases (including INGRES, ORACLE, and SYBASE databases), expose users to a full set of UniSQL's object-oriented ANSI SQL query capabilities, and add mission-critical database services. The combination of UniSQL and ParcPlace's development tools enable developers to increase productivity and save time building new applications. In addition, users are able to leverage the performances and reliability of UniSQL's databases for large scale applications.

For more information contact UniSQL, at 512.343.7297.  
**UniSQL, Inc., 8911 N. Capital of Texas Hwy., Ste. 2300, Austin, TX 78759, 512.343.7297 (v), 512.343.7383 (f).**

### HP Distributed Smalltalk 4.0 to Support ENVY/Developer

Hewlett-Packard has announced that it plans to develop the extensions needed to allow HP Distributed Smalltalk 4.0 to

## Product Announcements

support the Object Technology International's ENVY/Developer team programming environment.

The combination of ENVY/Developer with HP Distributed Smalltalk will give customers a solid engineering base for developing CORBA-standard distributed applications. The integration of ENVY/Developer's configuration management control with HP Distributed Smalltalk's industry-leading distributed development tools provides such specific technical benefits as follows: versioning and configurations of Smalltalk code and IDL repository modules; and convenient implementation sharing between developers through ENVY/Developer's repository.

ENVY/Developer is available from Object Technology International or one of its distributors. A single client or server license of ENVY/Developer is \$3,000. The product is available on the following platforms: HP-UX, SunOS, Solaris, IBM AIX, Sequent, Windows 3.1, Windows NT, OS/2, and Macintosh. **Hewlett-Packard Company, Direct Marketing Organization, P.O. Box 58059, MS511L-SJ, Santa Clara, CA 95051-8059.**

### HP Distributed Smalltalk Runs On Three PC, Four UNIX System-Based Platforms

Hewlett-Packard Company today announced Version 4.0 of its HP Distributed Smalltalk development environment. This version enables programmers to develop distributed computing applications that can run, without modification, on any combination of four UNIX system-based platforms and three PC platforms.

HP believes that the majority of its customers want to develop new client-server applications that use UNIX system-based servers and UNIX system-based or PC clients. Previously, HP Distributed Smalltalk supported the HP-UX, Solaris, SunOS and IBM AIX platforms; Version 4.0 now supports Windows 3.1, Windows NT, and OS/2 as well.

HP Distributed Smalltalk 4.0 software provides classes of objects that communicate over a network using an Object Request Broker, which is HP's implementation of the Object Management Group's (OMG) CORBA specification for object communications in a distributed object system. Distributed Smalltalk is built upon and extends ParcPlace System's VisualWorks 2.0 to create a development environment for peer-to-peer distributed-object applications.

Release 4.0 of Distributed Smalltalk includes two major enhancements: 1) It runs on these PC platforms: Windows 3.1, Windows NT 3.1, and OS/2 2.1. Users must supply a transport package for Windows 3.1 (Novell LAN Workplace for DOS 4.2 has been tested), and OS/2 (IBM TCP/IP Version 2.0 for OS/2); and 2) It supports ParcPlace VisualWorks 2.0.

Distributed Smalltalk is OMG CORBA-compliant, which ensures networked interoperability with other CORBA-compliant applications that use the same communications protocol (TCP/IP). Distributed Smalltalk supports the evolving Common Object Services (COS) standard and the proposed future specification of the Object Data Management Group.

HP Distributed Smalltalk Release 4.0 is expected to be available Dec. 1. A single-user license on PC platforms is \$2,995 and \$6,490 per license when bundled with ParcPlace's VisualWorks Smalltalk. A single-user license on UNIX system-

based platforms is \$4,995 and \$10,490 per license when bundled with ParcPlace's VisualWorks Smalltalk. Distributed Smalltalk is available from HP on the HP-UX, SunOS, Solaris and IBM AIX UNIX system-based platforms, and for Windows 3.1, Windows NT and OS/2 platforms.

Users purchasing a minimum of five Distributed Smalltalk licenses will receive a free copy of the VisualWorks Smalltalk development environment with each copy. This offer is in effect now and will be continued for a limited time.

**Hewlett-Packard Company, Direct Marketing Organization, P.O. Box 58059, MS511L-SJ, Santa Clara, CA 95051-8059.**

### ODBTalk for OS/2

LPC consulting services announced the availability of ODBTalk for OS/2. This product provides the Smalltalk/V developer with full 32-bit ODBC support using the OS/2 Q+E ODBC Driver Manager. The product, together with the appropriate ODBC database drivers, supports all ODBC conformance levels.

Currently databases supported by the Q+E Driver Manager include: Clipper, DB2 (MDI Gateway), DB2/2, DB2/6000, DBASE III and IV, Excel, Foxpro, FoxBASE, Informix 5, MS SQL Server, Oracle 6 and 7, SQL/400, SQL/DS, Sybase SQL Server 4, Teradata (through MDI Gateway), and text files.

The Watcom-SQL database is also supported by the ODBC API support which is included with watcom-SQL.

Smalltalk database access logic can be developed for all of the major Intel-based PC environments. ODBTalk for OS/2 is fully source-code compatible with our existing products, ODBTalk for Windows 16 bit, Win32s and Windows-NT.

Smalltalk developer has access to both low level classes (ODBC API level) and higher level classes (connection, statement, query, table...) providing various levels of encapsulations. This provides flexibility and allows the development of ODBC-compliant applications without having to learn the ODBC API itself.

ODBTalk for OS/2 requires OS/2 2.x. Smalltalk/V for OS/2, and the Q+E Driver Manager and the appropriate database drivers, or the Watcom-SQL database.

**LPC, 937 Briar Hill Ave., Toronto, ON Canada, M6B 1M1, Canada, 416.787.5290. (v)**

### ParcPlace Adds Team Programming Tools to Client and Server Product Line

ParcPlace Systems and Object Technology International (OTI) have signed a distribution agreement allowing ParcPlace to market, distribute, and support OTI's ENVY/Developer, the leading group development tool for managing software components in the Smalltalk environment. ENVY/Developer's team programming environment is fully compatible with VisualWorks, ParcPlace's powerful client and server tool for building portable applications using object-oriented technology.

ENVY/Developer provides an integrated team programming environment for VisualWorks developers, supporting the prototyping, development, release and deployment of software applications written in ParcPlace Smalltalk. ENVY/Developer is the only Smalltalk development tool currently available that scales across platforms, teams and project size.

**ParcPlace Systems, 999 E. Arques Ave Sunnyvale, CA 94086, 408.720.7514 (v), snichols@parcplace.com.**



**SMALLTALK  
SOLUTIONS**

*Where All the  
Talk is Smalltalk*

**Dear Smalltalk Colleague:**

As two industry professionals very much involved with the Smalltalk community, we are pleased to present **Smalltalk Solutions '95**, the first major vendor-independent Smalltalk conference.

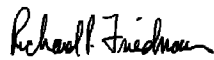
These are exciting times for Smalltalk enthusiasts. The Smalltalk marketplace doubled in 1993, and Smalltalk is currently the fastest growing object-oriented programming language. In our capacity as the President of SIGS Publications and the Co-Editor of **THE SMALLTALK REPORT**, we are proud to have been a part of the launch of **THE SMALLTALK REPORT** as a newsletter back in 1991. Thirty issues later, it has grown into a full-fledged and respected magazine, and is still the only independent publication serving the needs of the Smalltalk community.

We are similarly excited about the launch of **Smalltalk Solutions '95** in commemoration of Smalltalk's 25th anniversary. We anticipate that this event will become the annual gathering for Smalltalk professionals for many years to come.

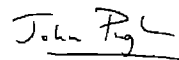
Over the past few months, we have been working hard to put together a technical program which will cater to the needs of all involved in Smalltalk development: programmers, consultants, project managers, educators. In addition to keynote presentations from Dave Thomas, Tom Atwood and Ray Wells, the technical tracks feature sessions presented by such recognized Smalltalk experts as Kent Beck, Rebecca Wirfs-Brock, Sam Adams, Wilf LaLonde, and Kenny Rubin. With topics such as design patterns, performance, metrics, meta-level programming, visual programming, and client/server and distributed systems, there is plenty to interest even the most experienced Smalltalk programmer. We also realize there is much to be learned from the experiences of others, and with this in mind, **Smalltalk Solutions** features corporate case studies from organizations such as Texas Instruments, Caterpillar, CIGNA, and the Canadian Imperial Bank of Commerce. We've also created a dedicated class track for managers to address the issues of managing and delivering large-scale Smalltalk projects.

We hope you will be able to join us at **Smalltalk Solutions** in New York. It will be a unique opportunity to learn from the experts and to share experiences with your peers in the Smalltalk community. If you'd like more detailed information, please contact our **Smalltalk Solutions** registrar by phone: (212)242-7515, fax: (212)242-7578, or email: [sigsconf@ix.netcom.com](mailto:sigsconf@ix.netcom.com), and we'll rush you our complete delegate brochure with full class descriptions. Or simply return the registration form found at the end of this Special Conference Preview Section.

See you in New York!  
Sincerely,



Richard P. Friedman  
President, SIGS Publications &  
SIGS Conferences



John Pugh  
Conference Technical Chair  
Co-Editor, The Smalltalk Report



Richard P. Friedman



John Pugh

**Celebrating  
25  
Years of  
Smalltalk**

Sponsored by

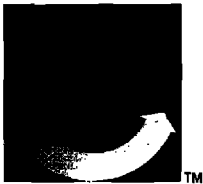
**THE  
Smalltalk  
REPORT**

Presented by

**SIGS  
CONFERENCES**

*Special Conference Preview Section*

# Oddly enough, the most productive application development tool for Windows isn't from Microsoft.®



**Introducing  
VisualAge™ for  
Windows™ and  
IBM Smalltalk.**

Now you have the power to develop industrial-strength client/server applications for the Windows environment in the blink of an eye.

VisualAge and IBM Smalltalk Version 2.0 extend IBM's powerful new vision of programming. They provide multiple platform support including Windows and OS/2,\* connectivity to business-critical data and

applications, scalability to create applications from the desktop to the enterprise, a fully object-oriented development environment, and a robust team version for greater programming productivity.

**Break the code barrier  
with VisualAge.**

With the simplicity of visual construction, you can create complex applications with amazing speed. What's more, you get the added flexibility of a completely integrated Smalltalk object-oriented base.

**The language of objects™**

IBM Smalltalk, included with VisualAge, is now available separately. Smalltalk programmers now have the standards-compliant, integrated development environment they need to easily develop fully portable applications.

To order or for more information, call 1 800 IBM-CALL, Dept. SA005. In Canada, call 1 800 565-SW4U, ext. 279, or contact your favorite reseller.

SOFTWARE FOR  
OBJECT-ORIENTED TECHNOLOGY

See us at Booth #101 at Smalltalk Solutions '95

Special Conference Preview Section



Outside North America, call: (Austria) 0222.21145.2500, (Belgium) 02.2253333, (Denmark) 80304545, (Finland) 90.459.4176, (France) 05.030303, (Germany) 0130.4567.111, (Italy) 1670.17001, (Netherlands) 030.384040, (Norway) 66.999300, (S. Africa) 27.11.2249.111, (Spain) 900.100400, (Sweden) 08.7934004, (Switzerland) 01.4366233, (UK) 081.5757700, or contact your local IBM office. IBM and OS/2 are registered trademarks and VisualAge and "The language of objects" are trademarks of International Business Machines Corporation. Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation. ©1994 IBM Corp.

# Smalltalk Solutions '95 — Conference At-A-Glance

TUESDAY — FEBRUARY 21

## PRE-CONFERENCE TUTORIALS

9:00-5:00	<b>T1</b> Discovering Smalltalk <i>John Pugh</i> THE OBJECT PEOPLE	<b>T2</b> Implementing Custom Graphical Views <i>Bobby Woolf &amp; Kyle Brown</i> KNOWLEDGE SYSTEMS CORP.	<b>T3</b> Mastering Responsibility-Driven Design <i>Rebecca Wirfs-Brock</i> DIGITALK	<b>T4</b> Object-Oriented Project Management Strategies <i>Kenny Rubin</i> PARCPLACE SYSTEMS
-----------	---	--	---	---

WEDNESDAY — FEBRUARY 22

## TECHNICAL      TECHNICAL      CORPORATE CASE STUDIES      MANAGEMENT

9:00-9:45	<b>KEYNOTE: Component-Based Software Construction: The Transition from Craft to Engineering</b> — <i>Dave Thomas, OTI</i>			
10:00-12:00	<b>W1</b> Writing High-Performance Smalltalk Programs <i>Kent Beck</i> FIRST CLASS SOFTWARE	<b>W2</b> Designing Responsible Objects Using Stereotypes <i>Rebecca Wirfs-Brock</i> DIGITALK	<b>W3</b> Organizational Impacts of Advanced Technology <i>Douglas Kittelsen</i> AMERICAN MANAGEMENT SYSTEMS INC.	<b>W4</b> What Really Happens When You Adopt Smalltalk <i>Ken Auer</i> KNOWLEDGE SYSTEMS CORP.
12:00-6:00	<b>Exhibit Hall Open</b>			
2:00-5:00	<b>W5</b> Smalltalk in the Large: Scaling Up Without Losing the Benefits <i>Jan Steinman &amp; Barbara Yates</i> BYTE SMITHS	<b>W6</b> Designing with Patterns <i>Kent Beck</i> FIRST CLASS SOFTWARE	<b>W7</b> I. Semiconductor Manufacturing Process Re-engineering Using Smalltalk <i>John McGehee</i> TEXAS INSTRUMENTS  II. Forging Steel Planning & Problem Resolution <i>Mike Baker</i> CATERPILLAR INC.  III. Smalltalk on the Internet <i>John Richards</i> IBM — TJ WATSON RESEARCH CENTER	<b>W8</b> Using Smalltalk and Teams: A Synergistic Solution <i>Jeff McKenna</i> THE MCKENNA CONSULTING GROUP
5:00-6:00	Welcome Reception			

THURSDAY — FEBRUARY 23

## TECHNICAL      TECHNICAL      CORPORATE CASE STUDIES      MANAGEMENT

9:00-9:45	<b>KEYNOTE: Smalltalk and ODBMS: The New Foundation for Client/Server Applications at Fortune 1000 Firms</b> — <i>Thomas Atwood, OBJECT DESIGN</i>			
10:00-12:00	<b>Th1</b> Writing Reusable Smalltalk Classes <i>Juanita Ewing</i> DIGITALK	<b>Th2</b> PANEL PRESENTATION How Does Smalltalk Really Scale? Moderator: <i>Ken Auer</i> KNOWLEDGE SYSTEMS CORP.	<b>Th3</b> Connecting Object Applications to Relational Databases <i>Timo Salo &amp; John Shelton</i> PARCPLACE SYSTEMS	<b>Th4</b> An Insiders' Guide to the Smalltalk Standards Initiative <i>Yen-Ping Shan &amp; Rick DeNatale</i> IBM
12:00-6:00	<b>Exhibit Hall Open</b>			
2:00-5:00	<b>Th5</b> Applications of Meta-Level Programming <i>Wilf LaLonde</i> THE OBJECT PEOPLE	<b>Th6</b> Designing and Building Distributed Solutions <i>Jeff Eastman</i> OBJECT ARCHITECTURE	<b>Th7</b> I. Modeling Business Objects for Medical Plans <i>Jim Dykas &amp; Alan Kirk</i> CIGNA  II. Creating an Auto Parts Lookup System with Smalltalk <i>Richard Goulet &amp; Mark Winter</i> CANADIAN TIRE  III. Building Banking Systems with Smalltalk <i>Al Woolfrey</i> CIBC	<b>Th8</b> Smalltalk Object Persistence in Heterogeneous Database Environments <i>Glenn Reid</i> QSYS SYSTEMS CONSULTANTS

FRIDAY — FEBRUARY 24

## TECHNICAL      TECHNICAL      CORPORATE CASE STUDIES      MANAGEMENT

9:00-9:45	<b>KEYNOTE: Smalltalk Objects: The Enabling Technology for New Business Process</b> — <i>Ray Wells, IBM CONSULTING GROUP</i>			
10:00-12:00	<b>F1</b> Metrics for Smalltalk <i>Mark Lorenz</i> HATTERAS SOFTWARE	<b>F2</b> Building a Cost Efficient Client/Server Architecture Using Smalltalk <i>Amarjeet Garewal</i> OBJECT EDGE	<b>F3</b> Practical Guidelines for Delivering Product Quality Smalltalk Applications <i>S. Sridhar &amp; Eric Clayberg</i> OBJECTSHARE SYSTEMS, INC.	<b>F4</b> PANEL PRESENTATION Setting Up a Smalltalk Shop Moderator: <i>Paul White</i> THE OBJECT PEOPLE
2:00-5:00	<b>F5</b> Visual Programming Techniques <i>Martin Nally</i> IBM	<b>F6</b> Techniques for Implementing Smalltalk in Client/Server Systems <i>Trevor Hopkins</i> UNIVERSITY OF MANCHESTER	<b>F7</b> Smalltalk Writ Large: Lessons Learned in Applying Smalltalk to Large-Scale Business Systems <i>Sam Adams</i> IBM	<b>F8</b> A Distributed Object Architecture for Corporate Information Systems <i>Jeff Sutherland</i> EASEL CORPORATION

**EXHIBIT HALL HOURS: WEDNESDAY, FEBRUARY 22, 12-6 PM; THURSDAY, FEBRUARY 23, 12-6 PM**

# STEP INTO THE FUTURE WITH THE COMPANY THAT DEFINED OBJECT TECHNOLOGY SERVICES

When object oriented programming was in its infancy, Knowledge Systems Corporation was already putting it to work in companies like yours. Today, we're positioned to take you into the future of object technology in ways that no other company can. With the most complete range of services in the industry, KSC can assure your successful object transition every step of the way. Classroom instruction, project-focused apprenticeships, and consulting are all part of our exclusive commitment to object technology services.

Once you've made the decision to move to object technology, you want to get the benefits as quickly as possible. KSC offers a complete curriculum of classroom education, at your site or in our corporate training facility. These courses help you establish a firm foundation in object technology concepts and Smalltalk programming.

To cut months off your transition time, we've developed an exclusive Smalltalk Apprentice Program (STAP). Already proven in companies

such as American Management Systems, GE Capital Corporation, IBM, Northern Telecom, The Prudential, Southern California Edison and Sprint, the STAP is a total immersion, project-focused program that compresses six to ten months of learning experience into four to six weeks.

KSC can also tackle your object technology projects head-on with the most experienced analysts, designers and programmers in the business. You can outsource the entire job, or use our consultants to lend expertise to your own development group.

In addition to our service offerings, KSC is a distributor of third party tools such as ENVY®/Developer, the premier Smalltalk team development environment.

If you're ready to step into the future of object technology, call the one company that will lead you

there—Knowledge Systems Corporation, 919-481-4000. Or email: [salesinfo@ksccary.com](mailto:salesinfo@ksccary.com). 4001 Weston Parkway, Cary, North Carolina 27513.



See us at Booth #201 at Smalltalk Solutions '95

Special Conference Preview Section



## KNOWLEDGE SYSTEMS CORPORATION

919-481-4000

# Product Education Sessions

**SMALLTALK  
SOLUTIONS**

**KNOWLEDGE  
SYSTEMS CORPORATION**

Wednesday, Feb. 22, 12:00 p.m.

## How to Transition to Smalltalk—The KSC Approach

*Presented by: Ronald Schultz – Director, Transition Solutions*

Corporation specializes in transitioning client projects, staff, processes, and software products to object technology. Since 1985, KSC has engineered and evolved a suite of offerings focused on expediting O-O adoption and Smalltalk. These offerings include Intro and Advanced Smalltalk (for IBM, Digitalk, and ParcPlace), O-O Analysis and Design, Project Kickoff Workshops, the Smalltalk Apprentice Program™, consulting, mentoring, and turnkey projects.

**IBM**

Wednesday, Feb. 22, 1:00 p.m.

## IBM's Visual Age Family (Including IBM Smalltalk)

*Presented by: Bill Allen – IBM Channel Marketing Manager*

Step up to the enhanced productivity of object-oriented programming with Visual Age and IBM Smalltalk, both from IBM. Develop "industrial-strength" client/server applications in record time. Both stand-alone and team versions are backed by IBM service and support. Visual Age success stories, potential, and demo are included in this presentation. See why PC Week called Visual Age "a developer's dream come true."

**MARK WINTER & ASSOCIATES**

Wednesday, Feb. 22, 2:00 p.m.

## Implementing Fuzzy Logic Algorithms in Smalltalk

*Presented by: Mircea Mihaescu – Senior Consultant*

Fuzzy Logic is a collection of powerful intelligent decision making algorithms used in various business

software environments. This lecture presents a short review of fuzzy set theory and object oriented programming concepts, followed by a detailed description of an object oriented framework written in Smalltalk that provides an extensible structure of classes for building business decision support systems. This approach is different from the few published fuzzy logic implementations by taking full advantage of the power of an object oriented language.

**EASEL CORPORATION**

Wednesday, Feb. 22, 3:00 p.m.

## Improving Smalltalk Developer Productivity

*Presented by: Jeff Sutherland – Vice President of Object Technology*

Smalltalk is renowned as the most productive software development environment. However, significant incremental productivity improvements are possible by integrating analysis/design tools and persistent object mapping tools with Smalltalk. Easel will present and demonstrate its implementation of such an environment through the Object Studio family of products.

**DIGITAL**

Wednesday, Feb. 22, 4:00 p.m.

## Building Applications from Components: From Myth to Reality

*Presented by: Thomas Murphy – Product Manager*

The software development world is clamoring for the religion of components. While components are not a panacea for all, they are an important part of building robust systems. For components to be effective you must have the ability to describe your components to others and find, understand, and easily extend and assemble the components available to you.

**THE OBJECT PEOPLE**

Thursday, Feb. 23, 12:00 p.m.

## The Object People Corporate Services

*Presented by: Paul White – Vice President & Ron Charron – Manager, Professional Development*

The Object People have been leaders in the field of Smalltalk training and development for over five years. Since their inception, they have focused exclusively on Smalltalk, and have a reputation as world leaders in the field of Smalltalk training.

Their training covers all major dialects of Smalltalk, including IBM Smalltalk and Visual Age, VisualWorks, and Smalltalk/V.

This presentation will outline our corporate training services and will discuss in detail the benefits for your organization.

**MARK WINTER & ASSOCIATES**

Thursday, Feb. 23, 1:00 p.m.

## Object Modeling

*Presented by: Mircea Mihaescu – Senior Consultant*

This seminar will summarize practical object oriented analysis and design techniques using ENFIN Smalltalk. The presentation will discuss different approaches to solve "business software" problems through mapping between the object model and a relational database. Details will be given regarding pessimistic concurrency implications in a persistent object layer implemented in Smalltalk.

**DIGITAL**

Thursday, Feb. 23, 2:00 p.m.

## Visual Smalltalk: O-O Components for Your Enterprise

*Presented by: Thomas Murphy – Project Manager*

The Visual Smalltalk product line is the leading Smalltalk solution and the only Smalltalk solution to pervasively make use of component technology. Come see how components and Smalltalk can solve big problems fast.



*Special Conference Preview Section*

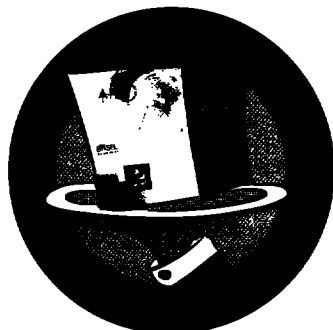
Presenting

# Object Studio

## SYNCHRONICITY ENFIN SMALLTALK TEAMBUILDER

*What makes Object Studio the most productive development environment?*

- BUILD HIGH-LEVEL BUSINESS OBJECTS BEFORE CODING BEGINS
- AUTOMATICALLY GENERATES SMALLTALK CODE TO IMPLEMENT BUSINESS OBJECTS
- BUSINESS MODEL & APPLICATION REMAIN SYNCHRONIZED AND EASILY MAINTAINED
- INSULATION FROM CLIENT/SERVER COMPLEXITIES
- POINT & CLICK INTEGRATION WITH RELATIONAL DATABASES
- UNMATCHED CONNECTIVITY



Visit booth 103 at Smalltalk Solutions  
to see Object Studio in action.

Object Studio is a complete object-oriented toolset for rapid development of large-scale production client/server applications on Windows, OS/2 & UNIX.

Corporate IS organizations can quickly achieve the productivity gains of object technology with Synchronicity.

Synchronicity's Business Object Modeling Tool allows the design of object models that mirror a business process, then automatically generates Smalltalk code.

Synchronicity's Persistent Object Mapping Tool quickly maps the models to relational databases.

ENFIN Smalltalk is a powerful visual development environment that supports the industry's widest range of connectivity options. TeamBuilder enables the development of ENFIN applications in a workgroup environment.

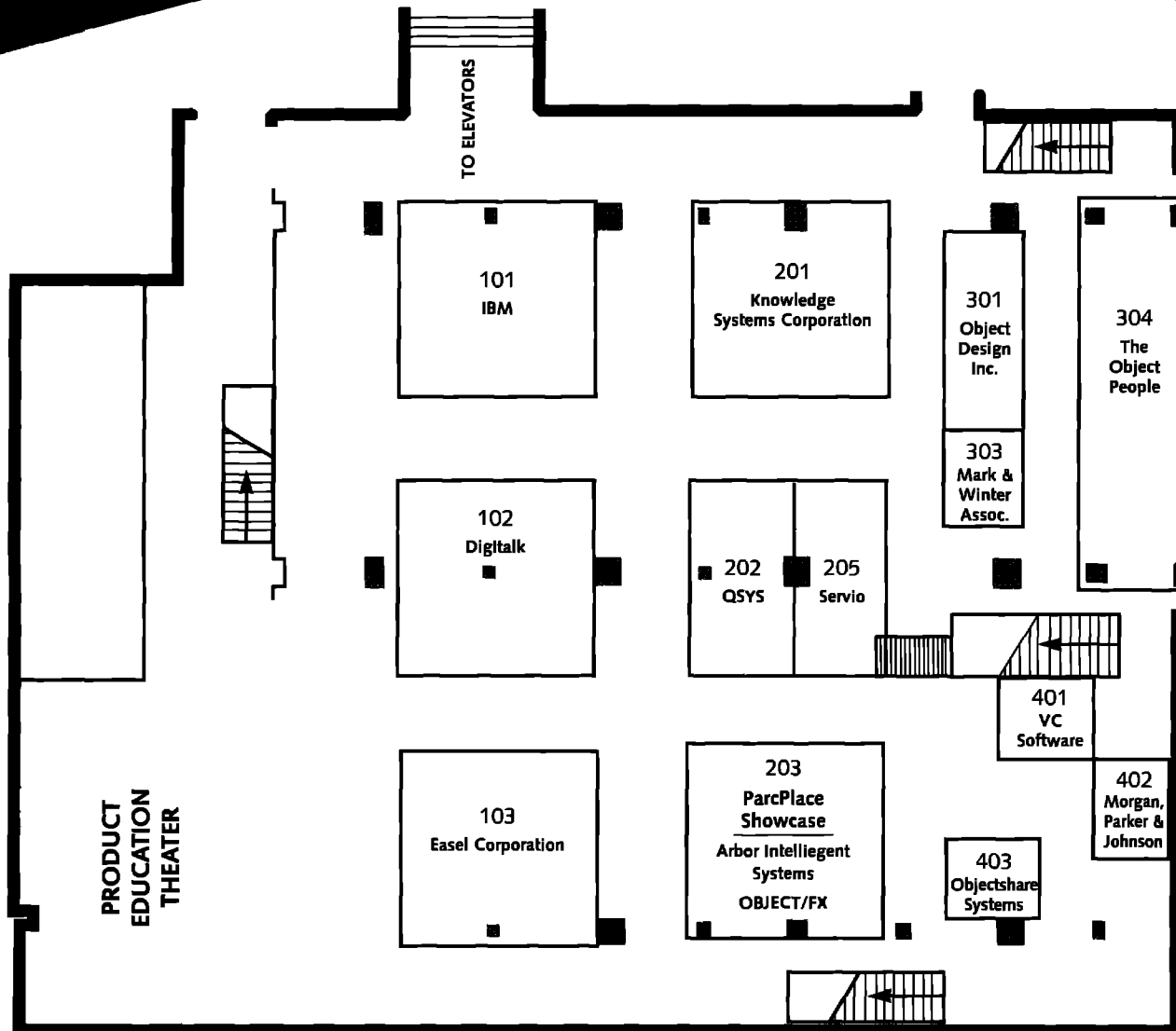
**EASEL**  
Corporation

25 Corporate Dr., Burlington, MA 01803 1-800- OBJECTS

Object Studio, ENFIN, Synchronicity, and TeamBuilder are trademarks of Easel Corporation. Other trademarks belong to their respective owners.

Special Conference Preview Section

# Floor Plan



## Smalltalk Solutions '95 Exhibitors Include:

- Arbor Intelligent Systems • Digitalt • Easel Corporation • IBM •
- JOURNAL OF OBJECT-ORIENTED PROGRAMMING • Knowledge Systems Corporation •
- Mark Winter & Associates • Morgan, Parker & Johnson •
- Object Design Inc. • OBJECT/FX • OBJECT MAGAZINE • The Object People
- Objectshare Systems, Inc. • ParcPlace • QSYS • ROAD • Servio •
- SIGS Books • SIGS Conferences • THE SMALLTALK REPORT •
- VC Software • THE X JOURNAL •



*Plus, 2nd Floor Mezzanine with more product displays!*

*Special Conference Preview Section*

TRANSITIONING TO SMALLTALK TECHNOLOGY?  
INTRODUCING SMALLTALK TO YOUR ORGANIZATION?  
*Travel with the team that knows the way...*

# THE OBJECT PEOPLE

*Your Smalltalk Experts*

**SMALLTALK**

## Education & Training

VisualAge  
IBM Smalltalk  
Smalltalk/V  
VisualWorks  
ENVY/Developer  
Analysis & Design  
Project Management  
In-House & Open Courses

## Project Related Services

Object Immersion Program  
Project Mentoring  
Custom Software Development  
Tools Construction

The Object People Inc. 509-885 Meadowlands Dr., Ottawa, Ontario, K2C 3N2  
Telephone: (613) 225-8812 FAX: (613) 225-5943

Smalltalk/V is a registered trademark of Digitalk, Inc.  
VisualWorks is a trademark of ParcPlace Systems Inc.

VisualAge and IBM Smalltalk are registered trademarks of IBM.  
ENVY is a registered trademark of OTI Inc.

*See us at Booth #304 at Smalltalk Solutions '95*

*Special Conference Preview Section*





## Digitalk

5 Hutton Centre Drive  
Santa Ana, CA 92707  
Tel: (714) 513-3000  
Fax: (714) 513-3100

### DIGITALTALK

Digitalk is the leading Smalltalk vendor. Since shipping the first commercial Smalltalk for the IBM PC in 1985, Digitalk has pushed Smalltalk forward with a complete set of development tools and services.

Visual Smalltalk Enterprise combines the fastest Smalltalk with visual component assembly, PVCS backed version control, and configuration management to build enterprise applications fast.

## Easel Corporation

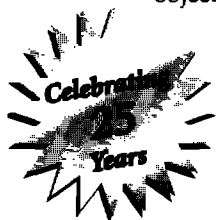


25 Corporate Drive  
Burlington, MA 01803  
Tel: (617) 221-2100  
Fax: (617) 221-2198

With 14 years of leadership in visual programming and object technology tools and services, Easel Corporation provides a true partnership solution for corporate IS developers seeking a rapid and smooth migration to today's client/server architectures. Easel Corporation products have delivered enhanced productivity gains while simplifying client/server computing in over 5,000 organizations worldwide.

Object Studio™ is a complete object-oriented toolset for rapid development of large-scale production client/server applications. Corporate IS organizations quickly achieve the productivity gains of object technology with Object Studio, which includes Synchronicity™ for business object modeling and persistent object mapping, ENFIN

Smalltalk™ for visual design and connectivity options, and TeamBuilder™ for group development.



## IBM

11000 Regency Parkway  
Cary, NC 27513  
Tel: (800) IBM-CARY  
Fax: (919) 469-7423



IBM provides world-class object-oriented tools for application developers. IBM Consulting and Professional Services has a wide array of world-wide education, training/mentoring, consulting, and services in support of Visual Age and IBM Smalltalk.

The Visual Age family of products will revolutionize how you create line-of-business applications. With the simplicity of visual construction, teamed with the powerful, object-oriented IBM Smalltalk environment, complex applications can be developed on Windows or OS/2 in a fraction of the time normally required.

## Knowledge Systems Corporation

4001 Weston Parkway  
Cary, NC 27513  
Tel: (919) 481-4000  
Fax: (919) 677-0074



### KNOWLEDGE SYSTEMS CORPORATION

Knowledge Systems Corporation (KSC) specializes in object technology. The company provides integrated services and tools for improving application time to market, quality and maintainability. All major Smalltalk dialects are supported on multiple platforms. Classroom instruction, facilitated analysis and design activities, project-focused apprenticeships and custom development services are available for making the transition to object technology.

## The Object People

885 Meadowlands Drive, Suite 509  
Ottawa, Ontario  
Canada K2C 3N2  
Tel: (613) 225-8812  
Fax: (613) 225-5943



### The Object People are

leaders in providing training, mentoring, and software development services in Smalltalk and object-oriented technology. Object People Staff have many years experience assisting clients utilize Smalltalk technology in a wide variety of application domains. Courses are offered in all major dialects of Smalltalk, VisualAge, Envy/Developer and O-O Analysis & Design. Courses are offered in-house or at the Object People professional development center in Ottawa.

The principals, John Pugh, Wilf LaLonde and Paul White, are well known as authors of four Smalltalk books, as Smalltalk columnists in the JOOP and as co-editors of THE SMALLTALK REPORT.

### Don't Miss the Educational Opportunities at Smalltalk Solutions '95:

A **TECHNICAL PROGRAM** developed and presented by leading Smalltalk experts... Thought-provoking **KEYNOTE PRESENTATIONS** from Dave Thomas, Tom Atwood, and Ray Wells... In-depth **PRODUCT EDUCATION SESSIONS** offering training on leading Smalltalk products from companies including Digitalk, Easel, IBM, Knowledge Systems, Mark Winter & Associates, and The Object People... Other **SPECIAL EVENTS** including Walk-In Clinics with the conference speakers and "Birds-of-a-Feather" roundtable sessions... And the opportunity to meet "outside the classroom" with other Smalltalk professionals in an informal setting... You'll come away with new contacts, fresh ideas, and practical tips and techniques that will prove invaluable to you in your job every day.



# Keynote Speakers

**SMALLTALK  
SOLUTIONS**

WEDNESDAY, FEB. 22

## Component-Based Software Construction: The Transition from Craft to Engineering

Dave Thomas  
Object Technology International

Object-oriented programming facilitates the development of libraries of pre-built components. This talk will describe experiences with component-based applications and product development. We will discuss the critical success factors for component-based development as well as various processes and component libraries and management tools used in a wide variety of engineering and IT projects.



THURSDAY, FEB. 23

## Smalltalk and Object DBMS: The New Foundation for Client/Server Applications at Fortune 1000 Firms

Thomas Atwood  
Object Design, Inc.

Smalltalk may become the COBOL of the '90s. As companies move from the mainframe to client/server computing, they are also beginning to do their new development in modern object programming environments. This presentation will discuss the ODBMS binding to Smalltalk-based, 4GL-like products, and will compare an ODBMS implementation with the Relational DBMS alternative. It will conclude with a discussion on insulating 04GL developers from Relational or IMS databases by using the ODBMS.



FRIDAY, FEB. 24

## Smalltalk Objects: The Enabling Technology for New Business Processes

Raymond B. Wells, PhD  
IBM Consulting Group

Today most large corporations are preparing for global competition, slashing costs and reengineering their business processes. Object technology is being used more and more as the enabling technology for information systems supporting the new business processes. Smalltalk is the development environment of choice. Dr. Wells will discuss the experiences of IBM's Object Technology Practice in working with some of North America's largest enterprises to solve their business problems with Smalltalk-based solutions.



## Your Free Pass to Exhibits and More!

To pre-register and receive your pass in the mail, complete this form and return on or before February 6, 1995. After February 6, please bring this completed form to the show. Please copy for each additional attendee.

Name \_\_\_\_\_  
Title \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_  
State \_\_\_\_\_ Zip \_\_\_\_\_  
Country \_\_\_\_\_ Postal Code \_\_\_\_\_  
Phone \_\_\_\_\_ Fax \_\_\_\_\_  
email \_\_\_\_\_

### Your pass entitles you to attend:

- Exhibit Hall
- Keynote Presentations
- Product Education Sessions
- "Birds-of-a-Feather" Sessions
- Walk-In Clinics

Photocopy  
and Mail or  
Fax Today!

Return to: Smalltalk Solutions, c/o RHS, 701 East Plano Parkway, Suite 300, Plano, TX 75074  
Fax: 212.242.7578 Phone: 212.242.7515

**SMALLTALK  
SOLUTIONS**  
95





**February 21-24, 1995**  
**Omni Park Central**  
**870 7th Avenue**  
**New York, NY**

**To Register:**

Complete the registration form to the right, and indicate the method of payment and the sessions you plan to attend. Make sure you choose only one session per time period.

**By Mail:**

71 West 23rd Street, 3rd Floor  
 New York, NY 10010

**By Phone: (212) 242-7515**

Use credit card/purchase order and call 9:00-5:00 EST, Monday to Friday

**By Fax: (212) 242-7578**

Send this form with credit card/purchase order

**Group Discounts**

Save by training your entire development team at once - discount of \$200 per person on the 3-day conference fee when a group of four or more register together from the same company. Early Bird discount does not apply to group rates.

**Hotel Accommodations**

Rooms at the Omni Park Central have been reserved for Smalltalk Solutions attendees at a special discount: Single or Double: \$99. Call the hotel at (212)247-8000 and mention Smalltalk Solutions.

**Airline Discounts**

American Airlines is offering Smalltalk Solutions attendees 5% off the lowest fare available, or 10% off discounted unrestricted coach fare. Rates are valid for travel from February 19 through 26. To take advantage of these special fares, have your credit card ready for payment, and call Dorothy Fradera at World Travel Specialists at 1-800-431-1112 between 8:30 a.m. and 6:00 p.m. EST. Mention that you will be attending Smalltalk Solutions.

**Cancellations**

Cancellations made by February 3, 1995 will be subject to a \$100 cancellation fee. Cancellations made after February 3, as well as "no-shows" are liable for the full registration fee. Substitutions may be made at any time. Cancellations must be made in writing and are only valid when you receive a cancellation number.

**REGISTRATION FORM**

(Photocopy this form for additional delegates)

**1 Please register me for Smalltalk Solutions '95**

Name (please print) \_\_\_\_\_

Signature \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ ZIP \_\_\_\_\_

Country \_\_\_\_\_ Postal Code \_\_\_\_\_

Day Phone \_\_\_\_\_ Fax \_\_\_\_\_

**2 Conference Package Options**

FILL IN AMOUNT

**3-DAY CONFERENCE (February 22, 23, 24)**

- Early Bird Rate (before January 20, 1995): **\$995\***
- Regular Rate: **\$1095\***
- Group Registration (group of 4 or more registering from the same company): **\$895 per person\***

**PRE-CONFERENCE TUTORIAL (February 21)**

- With 3-day conference: **\$325**
- Without 3-day conference: **\$450**

**ANY 1 DAY: \$450, ANY 2 DAYS: \$850**

**TOTAL \$** \_\_\_\_\_

**3 Method of Payment**

Payment due in full on or before the conference dates.

- Check enclosed (Payable to SIGS Conferences, in US dollars, drawn on a US bank)
- Purchase Order # \_\_\_\_\_

Invoice my company, Attn \_\_\_\_\_

- Charge my:  Visa  MasterCard  American Express

Card # \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

Cardholder name & address (if different from above)

**4 Tutorials & Lectures**

Please circle the sessions you plan to attend. Choose only (1) course per time period.

Tuesday	Full Day	T1	T2	T3	T4
Wednesday	Morning	W1	W2	W3	W4
	Afternoon	W5	W6	W7	W8
Thursday	Morning	TH1	TH2	TH3	TH4
	Afternoon	TH5	TH6	TH7	TH8
Friday	Morning	F1	F2	F3	F4
	Afternoon	F5	F6	F7	F8

Sponsored by:



\*\$20 of your registration fee applies towards your subscription to OBJECT MAGAZINE. If you are a current subscriber, your subscription will be extended one year (a \$39 value). If you do not wish to receive OBJECT MAGAZINE or are a subscriber and do not wish to renew, please deduct \$20 from this registration.

# Recruitment

## Join a Winning Team

### **Keane/Research Triangle Park, North Carolina A Great Place to Work – A Great Place to Live**

Keane, Inc. is a project oriented consulting firm that helps Fortune 500 companies by aligning their information systems with changing business objectives. Keane is the largest and fastest-growing software services company in its market segment, with more than 4,000 technical and business professionals and a network of 45 branch offices throughout North America.

If you're an experienced software professional with vision and solid technical skills, this is an opportunity to put your talents to work as a member of Keane's dynamic technical team.

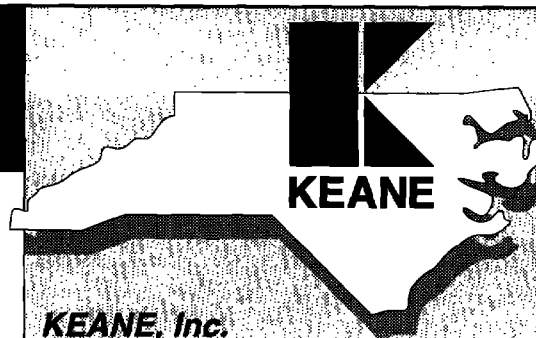
We have multiple needs for people with **SMALLTALK** and **OBJECT ORIENTED DESIGN/PROGRAMMING** skills for our offices in Raleigh/Durham, Charlotte and Greensboro/Winston-Salem.

### **The Advantage of Experience**

If you desire to be part of an industry-leading organization that recognizes excellence and rewards performance, and you wish to live in one of the best areas in the country, please send/fax your resume to:



Keane, Inc.  
2525 Meridian Parkway  
Suite 150  
Durham, NC 27713  
Fax: (919) 544-0895  
An Equal Opportunity Employer



### **KEANE, Inc.**

- 29 years of successful operation
- Recognized as one of the best companies in America, 5 consecutive years – *Forbes and Business Week*
- Will double 1993 revenues to approximately \$350 million
- One of the premier information systems consulting firms in North Carolina

### **Research Triangle Park, N.C.**

- Rated # 1 best place to live in 1994 – *Money Magazine*
- Rated # 1 best place in America to do business – *1993 Fortune*
- Rated # 6 quality place – *Places Rated Almanac 1993*
- Top 25 markets where homeowner will fare best – *US News & World Report 1993*

## Smalltalk Programmer/Analyst

The Capital Group, Inc. is one of the nation's most successful mutual fund and investment management organizations. We currently have an opening for an experienced Programmer/Analyst in our southern California office.

You will be a member of a team dedicated to application analysis, design and development for OS/2 Smalltalk V Client Server projects and systems using HP/UX-Sybase servers. The ideal candidate will possess 2-3 years experience in building and maintaining Smalltalk production client server applications as well as skills in object oriented analysis and design.

We offer a competitive salary as well as an excellent benefits package, including medical, dental and vision care coverage, educational reimbursement, health club subsidy and an outstanding retirement plan. To apply, please send your resume to: The Capital Group, Attn: Human Resources, SR/#342, P.O. Box 2215, Brea, CA 92622-2215. You may also apply by faxing your resume to (714)255-8504. EOE



THE CAPITAL GROUP, INC.

## Authors Wanted For Two Innovative Book Series

### **Managing Object Technology**

*edited by Charles F. Bowman*

For more information please contact:  
**Charles F. Bowman, Series Editor**  
(p) 914-357-6285, (f) 914-357-6524  
71700,3570@compuserve.com

and

### **Advances in Object Technology**

*edited by Dr. Richard S. Wiener*

For more information please contact:  
**Dr. Richard S. Wiener**  
135 Rugely Court  
Colorado Springs, CO 80906  
(phone & fax) 719-579-9616

**SIGS  
BOOKS**

# Recruitment

## ENGINEER THE FUTURE OF HEALTHCARE

### SOFTWARE ENGINEERS

HBO & Company (HBOC) is a leading international developer and provider of software solutions for hospitals and the healthcare enterprise. With over 2500 employees and 1994 revenues anticipated to exceed \$300 million, we are continuing 20 years of success and profitable growth. Join the leader and grow your career with us in our Atlanta, GA, Minneapolis, MN or Amherst, MA offices.

We seek talented individuals to design and develop our next generation of software products using the latest technologies. We currently have the following openings for Information Technology professionals.

#### Smalltalk

The ideal candidates will have experience with object-oriented analysis and design, PC software development, and Smalltalk programming.

#### Visual C++

Positions require 2+ years of development experience with Visual C++ in a Windows environment.

The professionals we seek must possess excellent communications skills and the ability to work in a team environment.

HBOC offers excellent benefits, competitive salaries and a team-oriented professional work environment where promotion from within is the norm. If you possess energy and vision and wish to join a company committed to excellence, please forward/fax your resume to: **HBO & Company, Corporate Recruiting, OOD1/95, 301 Perimeter Center North, Atlanta, GA 30346, fax to: (404) 393-6063.** No phone calls or agencies, please.

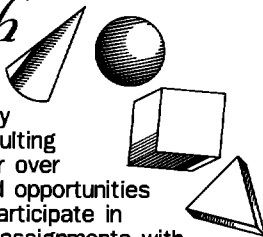


An Equal Opportunity Employer M/F/D/V

**HBO & Company**

## SUCCESS

With



### Smalltalk Developers

At QSYS we have successfully provided Object Oriented consulting services to our customers for over seven years. This has created opportunities for Smalltalk Specialists to participate in leading edge, mission critical assignments with our Fortune 1000 clients.

If you have demonstrated experience implementing large OO systems using IBM Smalltalk or Visual Age,™ ParcPlace VisualWorks,® Digitalk Smalltalk/V,® we would like to hear from you!

For further information, contact  
**Elspeth Koor at 1-800-999-9776.**

1 Yonge Street, Suite 1801, Toronto, Canada  
M5E 1W7 Fax: (416) 369-0515



90 Park Avenue, Suite 1600, New York, NY  
10016 Telephone: (212) 984-0715

Email: 72072.2575@compuserve.com

### HOW TO CONTACT SIGS PUBLICATIONS

#### To submit materials for publication

Article proposals, outlines, and manuscripts; industry news; press briefings; letters to the editor; and product announcements—send to  
John Pugh and Paul White, Editors

#### THE SMALLTALK REPORT,

Product Announcements Dept.,

885 Meadowlands Dr., #509

Ottawa, ON K2C 3N2, Canada,

613.225.8812 (v), 613.225.5943 (f).

#### Customer Service

In the US—

Phone: 215.785.5996 Fax: 215.785.6073

email: p00976@psilink.com

In Europe—

Theresa Procter

SIGS Conferences Ltd.

Phone: 44.0306.631.331

#### To order a subscription or change the name/address of an existing subscription

In the US—

OM, P.O. Box 2029

Langhorne, PA 19047

Phone: 215.785.5996 Fax: 215.785.6073

In Europe—

Theresa Procter

SIGS Conferences Ltd.

Phone: 44.0306.631.331

#### To order back issues

Back Issue Order Dept.

SIGS Publications

71 West 23rd Street, 3rd floor

New York, NY 10010

Phone: 212.242.7447 Fax: 212.242.7574

#### For information on list rentals, contact:

Rubin Response

Phone: 708.619.9800 Fax: 708.619.0149

#### For information on reprints of published material, contact:

Duane Dagen

Reprint Management Services

505 East Airport Road, Box 5363

Lancaster, PA 17601

Phone: 717.560.2001 Fax: 717.560.2063

#### To place an advertisement or request a media kit for any SIGS publications, contact:

Director of Sales

SIGS Publications

Phone: 212.242.7447 Fax: 212.242.7574

#### For information on SIGS Books and SIGS Conferences

Phone: 212.242.7447 Fax: 212.242.7574

# Don't Miss Germany's Most Attended Object Technology/C++ Conference!

## Now in its 4th Year!

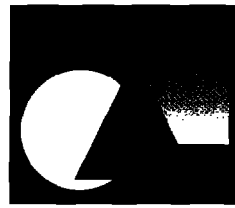
Object technology and C++ join forces at this annual technical affair dedicated to providing OOP professionals with the latest on Germany's fastest growing software technology and computer language in use today. Whether you are a seasoned professional or just exploring the possibilities, OOP '95 featuring C++ World is the perfect opportunity to gather with over 1,500 expected attendees and partake in technical courses, attend informative lectures and visit an impressive floor of exhibits.

No other conference in Germany provides such an educational range of classes, in-depth advice and hands-on demonstrations in 1995!

## New for 1995!

- ▼ An integrated and newly developed OT and C++ program including 6 topic tracks with 8 new classes, and 5 full-day post-conference tutorials.
- ▼ The exhibit floor spans one large level to readily showcase over 60 OT and C++ companies, products and services.
- ▼ The technical program has been devised to incorporate more internationally known German and English speakers. Classes are conducted in both German and English.
- ▼ An Executive Briefing. Register to attend this special event geared toward upper-management levels addressing the issues concerning them most. Learn through the experiences of others and how they implemented OT into their corporate worlds.

January 30 - February 3, 1995  
Munich Sheraton, Germany



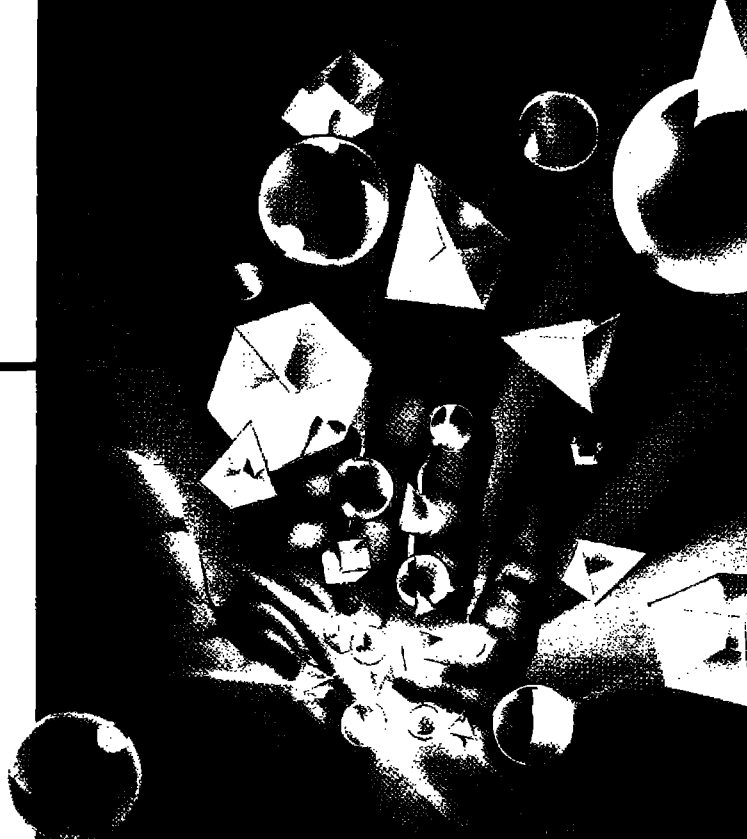
Objekt-orientiertes Programmieren

OOP '95

M Ü N C H E N

Featuring C++ World

## Moving Forward with Object Technology



A SIGS Conference

For information on exhibiting or attending OOP'95 featuring C++ World, please contact:

### Exhibit Sales

Nadine Tillack  
SIGS Conferences GmbH  
Birkenhoheweg 12, D-51465  
Bergisch-Citaback  
v: 022.029.36810  
f: 022.029.368122

### Attendee Information

Norbert Amthor  
SIGS Conferences GmbH  
Cosimastr. 306  
Munich, Germany 81952  
v: 089.957.7047  
f: 089.957.9125

## Attend Classes Presented by Industry Experts!

Don't miss the opportunity to learn from a select group of distinguished industry experts such as:

Tom Atwood, Ken Auer, Sandeepan Banerjee, Walter Bischofberger, Frank Buschmann, Matthias Bucker, Michael Cantone, Peter Coad, James Coplien, Peter Eichhorst, Don Firesmith, Bogdan Franczyk, Stuart Frost, Marc Gille, Marc Goldberg, Neil Goldstein, Mike Guttman, Georg Heeg, Peter Hruschka, Tom Jell, Gerti Kappel, Tim Korson, Marie Lenzi, Bertrand Meyer, Paul Micke, Annrai O'Toole, John Pugh, Dave Reed, Doug Rosenberg, Martin Röscher, Heinz Schneeweiss, Berndt Schmidt, Harry Sneed, Michael Stal, Uwe Steinmüller, John Vlissides, Raymund Vorwerk, Tony Wasserman, Uwe Werner and John Williams among others!

## Register for technical classes such as:

O-O Client Server Development, Patterns Panel, Writing Efficient Software Programs, Concurrent O-O Network Programming with C++, ODMG, Object Models: Strategies Patterns & Applications, Shlaer/Mellor vs. OMT, Tools and Frameworks, and Writing Efficient Smalltalk Programs.

FRANCES PAULISCH, TECHNICAL CHAIR

## Sponsored by:

JOURNAL OF  
OBJECT-ORIENTED  
programming

C++REPORT

OBJECT  
magazine

OBJEKTspektrum

## Presented by:

SIGS  
CONFERENCES

Buy 2 or more books and receive a FREE gift!



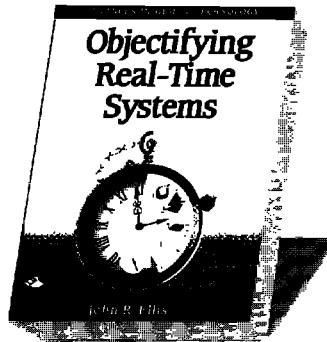
## Object Development Methods

edited by Andy Carmichael

A must-read for becoming familiar with and choosing from the leading methodologies. Provides valuable insight into the object-oriented methods of Shlaer/Mellor, Jacobson, Rumbaugh, Booch, Texel and Coad/Yourdon among others.

For systems analysts/designers, programmers, project managers, software engineers, IT managers or chief scientists. (347 pages.)

Available at selected bookstores. Distributed by Prentice Hall.

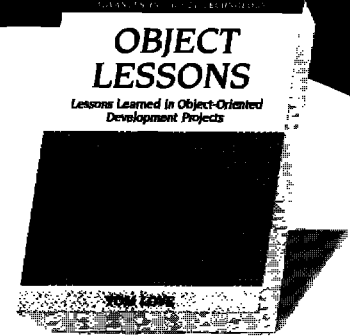


## Objectifying Real-Time Systems

by John R. Ellis

Engineers developing real systems can now apply object-oriented techniques to their daily projects. Ellis' methodology represents the evolution of RTSA into RTOOSA, and the accompanying diskette allows readers to experiment on their own without having to key in code.

For anyone interested in developing object-based real-time systems. The reader should have a solid grasp of RTSA. (525 pages with diskette.)



## Object Lessons

by Tom Love

An indispensable reference guide for all professionals in the field of object-oriented software engineering. Love encourages technical leaders and managers to avoid disaster by learning from the mistakes of others rather than repeating them.

For software managers, systems analysts/designers, implementors, applications programmers, project leaders, and technical managers. (276 pages.)

FREE GIFT!



Buy 2 or more books and receive a FREE copy of FOCUS ON ODBMS, a 140-page reference guide that provides invaluable advice and insight needed to manage, build, and implement object databases.

## SIGS BOOKS ORDER FORM

### YES! Please send me the following book(s).

If I buy two or more I will receive a FREE copy of Focus on ODBMS. If I am not totally satisfied, I may return the book(s) within 14 days and receive a complete refund.

- Object Lessons, by Tom Love (ISBN: 0-9627477-3-4).....\$29.00
- Objectifying Real-Time Systems (diskette included) by John R. Ellis (ISBN: 0-9627477-8-5).....\$44.00
- Object Development Methods edited by Andy Carmichael (ISBN:0-9627477-9-3).....\$39.00
- I am buying 2 or more books. Please send me Focus on ODBMS absolutely FREE!

<input type="checkbox"/> Check payable to SIGS Books
<input type="checkbox"/> Visa <input type="checkbox"/> American Express <input type="checkbox"/> MasterCard
Card# _____ Exp. _____
Signature _____

Name \_\_\_\_\_

Title \_\_\_\_\_

Address \_\_\_\_\_

Company \_\_\_\_\_

City/State/Zip \_\_\_\_\_

Country/Postal Code \_\_\_\_\_

Phone \_\_\_\_\_ Fax \_\_\_\_\_

195TA

SEND TO:  
SIGS Books, P.O. Box 99425  
Collingswood, NJ 08108-9970  
Phone: 212.242.7447 Fax: 609.858.2007

**SIGS BOOKS**

SHIPPING AND HANDLING: For US orders, please add \$5 for shipping and handling; Canada add \$10; Foreign add \$15. IMPORTANT: NY State residents add applicable sales tax.

COMPLETE MONEY-BACK GUARANTEE