

AT LAST! TOOLS FOR BUILDING RELIABLE ORB APPS

OBJECT

magazine

IMPROVING SOFTWARE QUALITY THROUGH OBJECT TECHNOLOGY

MODELING YOUR BUSINESS

A SIGS Publication
4(7), Nov-Dec 1994
\$4.50 US/\$5 Canada

- Which elements should be objectified?
- The best approach
- Shlaer/Mellor in action

ARE YOUR CLASS HIERARCHIES
TOO DEEP? See p. 35

HOW TO SELL
OT WITHIN

HARNESSING
ENCAPSULATION

DELIVERING ON EXPECTATIONS
WITH SMALLTALK

What you need to know
OLE/COM vs. CORBA

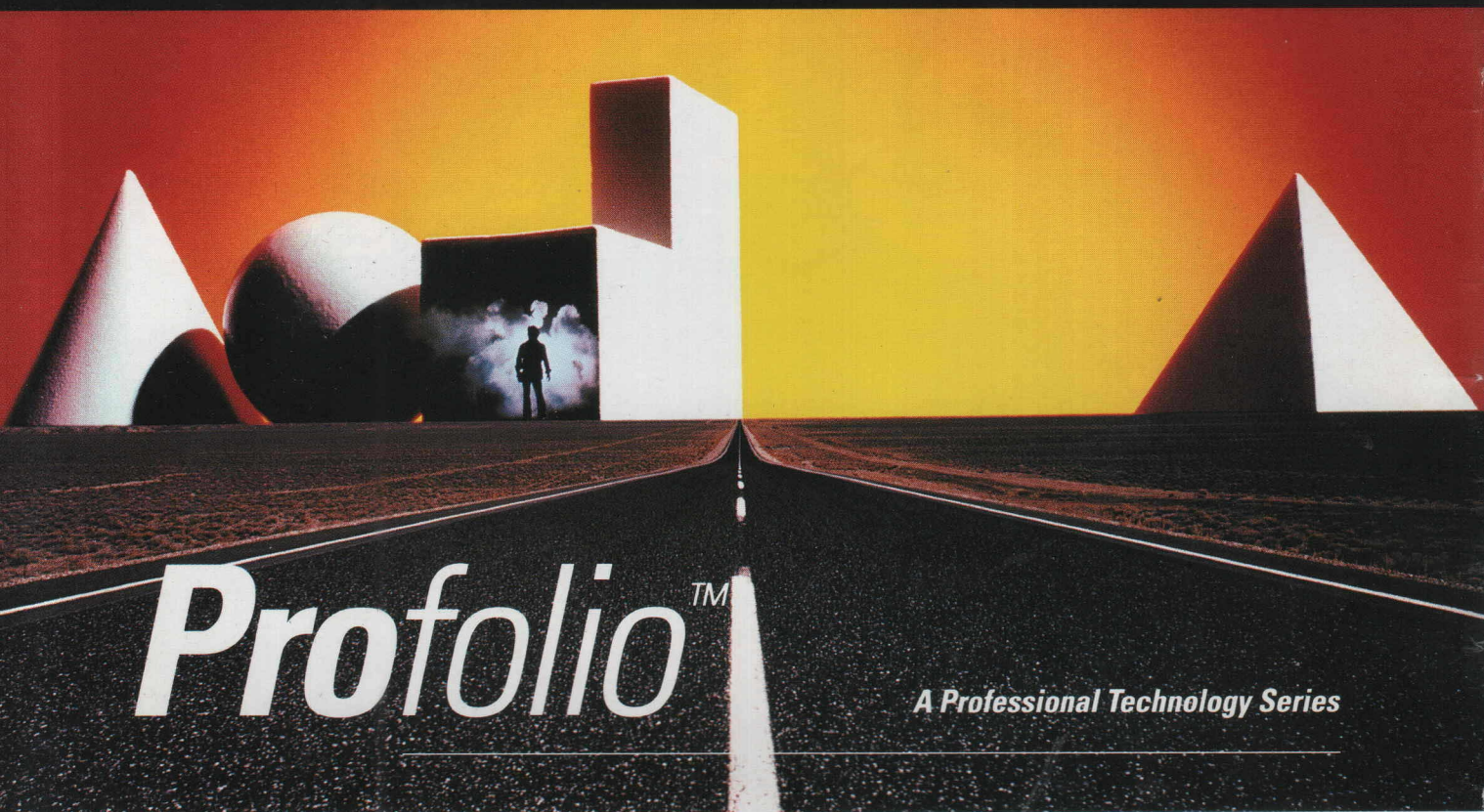
IBM'S OBSESSION
WITH OBJECTS

Revealed
The 7 snafus that
fail projects
& how to
avoid them



78779

Drive object-oriented technology into your organization . . . *Fast!*



Profolio™

A Professional Technology Series

Martin Marietta's Profolio™ couples *Advanced Concepts Center*™ training and *Interactive Development Environments* software to provide a complete and cost-effective way for your business to assimilate object-oriented technologies.

You get everything you need – in one package – to train and equip up to 15 of your IS professionals so they can quickly and confidently shift your development into high gear. Packages can be tailored to your individual needs.

To accelerate your transition, Profolio features a five-day object-oriented analysis and design class, led by Frederick Eddy, Senior GE Researcher and co-author of the best-selling text *Object-Oriented Modeling and Design*.

Included is Software through Pictures® for the Object Modeling Technique (StP/OMT). This application development tool automates OMT and guides your developers in the use of object technology.

A typical package includes:

- **Object-Oriented Analysis and Design** (5 days with Frederick Eddy)
- **Object-Oriented Analysis with StP/OMT** (5-day course)
- **Object-Oriented Design** (3-day course)
- **C++ Programming** (4-day course)
- **StP/OMT** (Software)

MARTIN MARIETTA
Advanced Concepts Center

640 Freedom Business Center
King of Prussia, Pennsylvania 19406
Phone +1 610-992-6200, Fax +1 610-992-6299

IDE **INTERACTIVE
DEVELOPMENT
ENVIRONMENTS**
Software through Pictures®

Register now for the future.
1-800-438-7246

™Advanced Concepts Center is a trademark of the Martin Marietta Corporation. ™Profolio is a trademark of the Martin Marietta Corporation.
®Software through Pictures and the IDE logo are registered trademarks of Interactive Development Environments, Inc.

Circle 1 on Reader Service Card

So many
industry
leaders
agree
on one
object
database,



Scott McNeely
Chairman and CEO
Sun Microsystems

Ray Noorda
President and CEO
Novell, Inc.

Steve Mills
General Manager
IBM Software Solutions

isn't it
worth a
phone call?
1-800-962-9620

It sure is. Because when you call Object Design®, you'll get the inside story on the object database from a company Ray Noorda called "the driving force behind the future of ODBMS," Scott McNeely described as "the foundation for a new generation of enterprise-wide applications" and Steve Mills declared is "the leader in object database technology."

It's called ObjectStore™, and discovering what these leaders already know is easy. Just call for your free Object Database Evaluation Kit, featuring an Evaluation Guide, user stories and our video.

With the Kit, you'll learn what to look for in an object database. You'll see the importance of interactive performance, scalability and persistence independent of type. And you'll discover how our innovative virtual memory mapping architecture delivers those benefits, making ObjectStore the world's leading object database.

It's all in the Kit. Along with details on exclusive features and user stories. So if you're ready to choose an object database, get your Evaluation Kit now. Call **1-800-962-9620** and see what everyone is talking – and agreeing – about.



OBJECT DESIGN

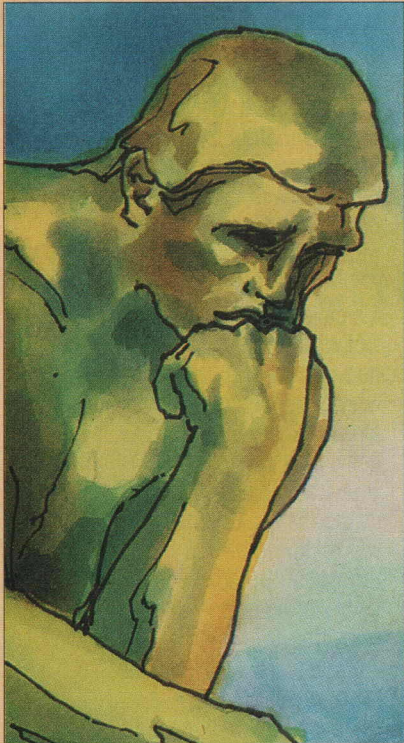
Leadership by Design®

Object Design, Inc., Corporate Headquarters, Burlington, MA USA

Wiesbaden, Germany 49-611-97719-0 Swindon, UK 44-793-486111 Sydney, Australia 61-2-212-2766

Tokyo, Japan 81-3-3263-9870 Boulogne, France 33-1-46-99-13-00

© 1994 Object Design, Inc. Object Design and Leadership by Design are registered trademarks and ObjectStore is a trademark of Object Design, Inc.



28



38



54

OBJECT

magazine

A SIGS Publication

ANALYSIS & DESIGN

28 Modelling rules in object-oriented analysis

Sanjiv Gossain

Increasingly, organisations are using object modelling techniques to model their businesses. Adequately capturing the business rules that reflect the policies of your organisation and place constraints upon the way your business functions is essential to this process.

35 Is a deep class hierarchy considered harmful?

Barbara Lin Wang & Jin Wang

A major advantage of object technology is its lower software maintenance cost and easier extension via component reuse. However, a class hierarchy that is too deep can discourage reuse on this level, depriving the organization of the expected cost benefits.

DISTRIBUTED OBJECT COMPUTING

38 Choosing between OLE/COM and CORBA

Thomas J Mowbray

The choice between OLE/COM and CORBA for corporate developers and system integrators requires a rational decision-making technique, including independent investigation of the facts behind vendor claims, and the distributed computing needs of your organization.

CASE STUDY

48 Developing a medical diagnostic system with OOA/RD

Jim Ladd

To realize the full benefits of objects, the object paradigm must extend beyond the programming phase of software development.

PROJECT MANAGEMENT

54 Managing customer expectations in Smalltalk development

Douglas Camp

The object-oriented development process is markedly different than the procedural one, due to its iterative nature and rapid prototyping. If project managers don't actively address customer expectations, delivered systems may be deemed failures simply due to unrealistic expectations of functionality, delivery time, or costs.

Table of Contents

4(7) November–December 1994

ABSTRACTIONS

60 About the paradigm

Viewed and used as a "frame of reference," a way of "seeing," a method of "understanding," a type of "map," the object-oriented paradigm provides a different yet powerful means with which to examine and analyze the business environment.

Paul Schwork

64 Lessons from the experts

Should you care about going object-oriented? **Yes**—the experts say the benefits are there, as long as your approach is careful and committed.

Richard W Koontz

COLUMNS

12 Object Insider: The future of object technology

James Cannavino

16 Methodology: An integrated approach to modeling

Adrian Bowles

20 Objects in Business: The challenge of change

Ted Shelton

24 ORB: Building reliable ORB applications

Mark Roy & Alan Ewald

68 Migration Strategies: You was my object...but I done you wrong

Ian Graham

71 Standards: Defining interfaces

Richard Soley

74 Education & Training: Making the most of object technology conferences

Susan Lilly

DEPARTMENTS

4 From the Editor: Is it mission critical or just a fancy query?

Marie A Lenzi

6 Letters

8 Executive Brief

Michael Kei Stewart

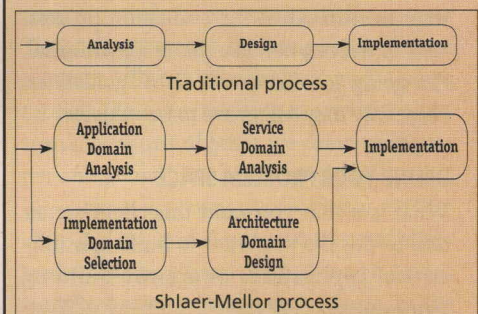
72 Ad Index

76 Product Track

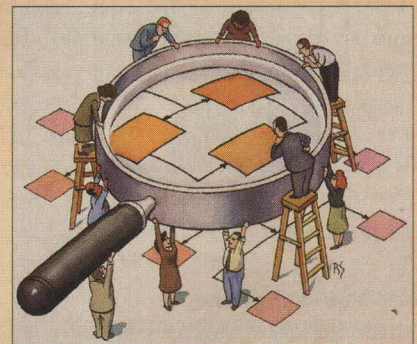
Jane M Grau



Cover Illustration by Lorraine Silvestri/SIS



52



60



Object Insider
James Cannavino



Is it mission critical or just a fancy query?

I HEARD A STATISTIC recently that quoted 250 client/server products on the market today. This is a staggering number. First of all, 250 of almost anything is a big number. 250 entries in a software product class has to be viewed with a questioning and critical eye. 250 is entirely too many products for anyone to analyze and evaluate. There have to be ways to trim down the evaluation process. And of course, because this is my editorial, I'm going to offer you a set of guidelines. Hopefully they will prove to be valuable.

DEFINE YOUR PROBLEM SPACE

What is it that you want this client/server tool to do? Do you want to build mission-critical applications or are you building fancy queries (Fig 1)?

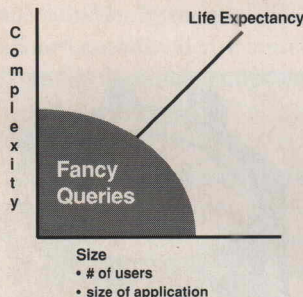


Figure 1

Mission-critical systems

Mission-critical systems are typically large, complex, and have a fairly long life expectancy. For the mission criticals you want a mechanism that allows for understanding and management of the business model independently of the presentation options (GUI?), and independent of the storage mech-

anisms (DBMS, VSAM?)

(Fig 2). The business model typically remains pretty much intact for extensive periods of time, but when things do change you must be certain beyond any question that they change consistently and throughout the system. This demands that logic be maintained in a single centralized location—the business model—and then disbursed via the independent presentation layer. In addition, the presentation and the storage is subject to change regularly and this is becoming more of an issue as organizations continue to move toward interoperability and heterogeneous environments. It's not that you want MS Windows or Motif, it's that you want MS Windows, *and* Motif *and* OS/2 PM. It's not that you have to access DB/2 or spin tapes, it's that you have to access DB/2, VSAM, Sybase, *and* tapes.

Fancy queries

For the fancy queries, where there is low complexity, small scale, and you expect a fairly short life for the code, a solution with-

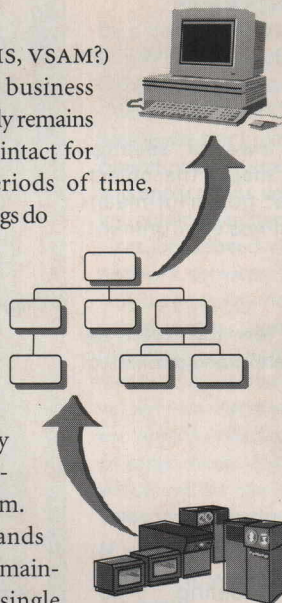


Figure 2

EDITOR—MARIE A LENZI

SIGS ADVISORY BOARD

Thomas Atwood, *Object Design*
François Bancelhon, *O2 Technology*
Grady Booch, *Rational*
George Bosworth, *Digital*
Adele Goldberg, *ParcPlace Systems*
R Douglas Kahn, *Easel Corp.*
Thomas Love
Bertrand Meyer, *ISE*
Meilir Page-Jones, *Wayland Systems*
Bjarne Stroustrup, *AT&T Bell Labs*
Dave Thomas, *Object Technology International*

EDITORIAL BOARD

Kenneth Auer, *Knowledge Systems Corp*
Avron Barr, *Aldo Ventures Inc.*
Larry L Constantine, *Consultant*
Steve Cook, *Object Designers, Ltd*
Paul Gross, *Borland International*
Brian Henderson-Sellers, *University of Technology, Sydney*
Jon Hopkins, *Palladio Software*
Mary ES Loomis, *Hewlett-Packard Labs*
Rob Murray, *AT&T Bell Labs*
Andy Novobilski, *NovoTech*
Cliff Reeves, *IBM*
Kenneth S Rubin, *ParcPlace Systems*
Jacob Stein, *Sybase*

COLUMNISTS

Douglas Barry, *Databases*, dougbarry@aol.com
Grady Booch, *Practical Objects*, egb@igor.rational.com
Adrian Bowles, *Methodology*, 71155,3646@compuserve.com
Tony Durham, *European Correspondent*, tdurham@cix.compulink.co.uk
Alan Ewald, *ORB*, ane@netlinks
Adele Goldberg, *Wishful Thinking*, adele@parcplace.com
Ian Graham, *Migration Strategies*, ian_graham%ccmail@gb.swissbank.com
Susan Lilly, *Education & Training*, slilly@vnet.ibm.com
John Palmer, *Antihype*, 75300,742@compuserve.com
Daniel W Rasmus, *AI and Objects*, danielr9@aol.com
Mark Roy, *ORB*, mpr@netlinks
Ted Shelton, *Objects in Business*, ems@its.com
Jim Stikeleather, *Client/Server*, stike@delphi.com
David Taylor, *Easing into Objects*, dtaylor@engines.com

CONTRIBUTING EDITORS

Jane M Grau
Michael Kei Stewart

SIGS PUBLICATIONS, INC

Richard P Friedman, *Founder & Group Publisher*

EDITORIAL/PRODUCTION

Kristina Joukhadar, *Managing Editor*
Andrea Cammarata, *Art Director*
Elizabeth A Upp, *Production Editor*
Margaret Conti, *Advertising Production Coordinator*
Tanya Trowell, *Editorial Production Assistant*

CIRCULATION

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

ADVERTISING/MARKETING

Shirley Sax, *Director of Sales*
Gary Portie, *Advertising Manager, Eastern US, Canada, and Europe*
Kristine Viksnins, *Account Executive & Exhibit Sales*
Michael Peck, *Advertising Sales Assistant*
Diane Fuller, *West Coast*
Diane Fuller & Associates
408.255.2991, f: 408.255.2992
Sarah Hamilton, *Director of Promotions and Research*
Karen Polner, *Promotions Graphic Artist*

ADMINISTRATION

Margherita R Monck, *General Manager*
David Chatterpaul, *Accounting Manager*
James Amenuvor, *Bookkeeper*
Michele Watkins, *Assistant to the Publisher*
Shannon Smith, *Administrative Assistant*



Publishers of OBJECT MAGAZINE, JOURNAL OF OBJECT-ORIENTED PROGRAMMING, THE X JOURNAL, C++ REPORT, REPORT ON OBJECT ANALYSIS & DESIGN, THE SMALLTALK REPORT, and OBJEKT SPEKTRUM (Germany)

OBJECT MAGAZINE (ISSN 1055-3614) is published nine times per year, monthly except for Mar/Apr, Jul/Aug and Nov/Dec. Published by SIGS Publications, Inc. 71 W. 23rd St., New York, NY 10010. ©Copyright 1994 by SIGS Publications, Inc. All rights reserved. Second Class postage paid at New York, NY and at additional entries. Statements of opinion and fact are made on the responsibility of the authors alone and do not imply an opinion on the part of SIGS PUBLICATIONS INC. or the editorial staff. All trademarks are the property of their respective owners.

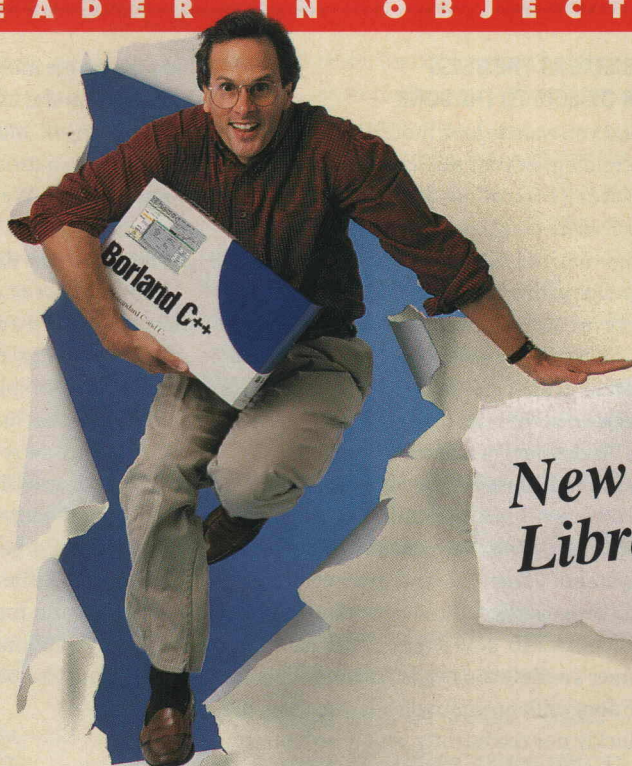
Send submissions to: Editor, OBJECT MAGAZINE, 236 West 26th Street, Suite 75W, New York, NY 10001, phone: 212.229.0320, fax: 212.807.9111, email: 70754.1111@compuserve.com

Subscription rates—one year (nine issues) domestic \$39; foreign and Canada \$74.

Postmaster: Send address changes and subscription orders to: Object Magazine, P.O. Box 2029, Langhorne, PA 19047. For service of existing orders: phone: 215.785.5996, fax: 215.785.6073, p00976@pslink.com (e-mail).

Printed in the USA. Canada Post International Publications Mail Product Sales Agreement No. 290386.

THE LEADER IN OBJECT COMPUTING



*New! ObjectWindows
Library for OS/2*

Get Borland C++ today, *and* receive OS/2 2.1 *free!*

There'll never be a better time to try OS/2 development

Borland® C++ is the world's most popular 32-bit compiler. OS/2® is the world's fastest-growing 32-bit operating system. Together in one box, they're the world's best deal.

“The easiest OS/2 environment I've worked in.”

Brett Glass, PC Techniques

32-bit power and Borland C++ flexibility

Borland C++ 2.0 for OS/2 gives you the most flexible Integrated Development Environment (IDE) going. Its advanced visual tool

set, powerful integrated editor, and fully integrated GUI debugger help speed you through even the most complex development project. Together with OS/2 version 2.1 for Windows, you have almost unlimited 32-bit power, and the best tools to develop it.

New ObjectWindows Library for OS/2

Borland's popular ObjectWindows® Library (OWL) is now available for OS/2 Presentation Manager. OS/2 developers will love this fully object-oriented application framework and Windows developers will be able to move their OWL programs to OS/2 quickly and easily.

Limited-time offer

But you'll have to act fast. So see your local Borland languages dealer.

Borland C++ 2.0 for OS/2 with OS/2 2.1. There'll never be a better time to try OS/2 development.

Call now!

1-800-336-6464, ext. 9569

to get everything you need to get started in OS/2 development!

Borland
The Upsizing Company

out the business model layer is adequate (Figure 3). Recognize however, that any business logic you devise is typically stuffed into the presentation code. This is fine if there are not a lot of screens. What happens here is that business logic is duplicated, and that means, of course, a multiplicity of maintenance headaches: First in finding all the places where the logic has to change, next making all the changes and making them consistently, then testing all the changes, and finally in being sure that, in fact, you have found and made all of the changes . . . consistently and correctly. For small-scale environments this is doable. When systems start to grow beyond the magical number 7+2 (anything) the risk increases.

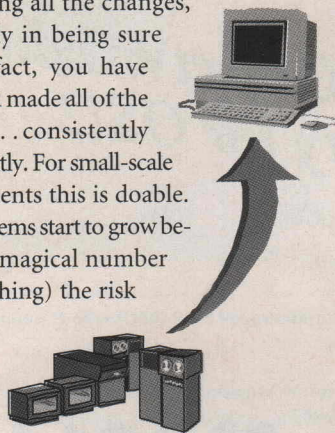


Figure 3

OBJECTS AT THE GLASS OR OBJECTS TO THE BONE

Next you want to look more deeply inside the technology of your implementation options. In today's age of not just client/server, but even more complex distributed client/server you want and need the ability to cleanly and intelligently build and distribute software components among clients and servers. Objects at the glass alone do absolutely nothing to support us in this area; in fact, objects at the glass alone do nothing for the developer . . . period.

As developers delivering mission-critical solutions to users/customers in a timely manner, we want real objects in the code—well defined, discrete components that can be understood, enhanced, and extended cleanly and then distributed and redistributed across processors to best support the user/customer's and system needs. Of course this can be done with procedural code, however, not quickly nor confidently, and it isn't pretty.

For the fancy queries, full object orienta-

tion makes life much easier for the developers as well as the users/customers, and even partial, sort-of, and pseudo-object orientation provides a stronger solution; however, it is not necessary. We have been living with Focus for a long time. It's pretty quick, not very pretty, but it gets the job done . . . for now.

But the cost of software is growing and the demand for complete and correct responsiveness from the users/customers is expanding. The simple solutions may work for the moment, but just for the moment. Don't expect to get away with these stopgaps for much longer, even for the fancy queries. While satisfying the short term user/customer demands we must be planning a longer-term strategy. Otherwise we will continue to grind away more and more money building and rebuilding solutions to a single old problem. That sounds real boring to me.

Pharid Lezi

Letters

GUI DESIGN PHILOSOPHY

Walrath and Hayden's article "The philosophy of designing a GUI" (OBJECT MAGAZINE 4[4]) provided very useful guidance and insight into this all-important aspect of automating the world. But isn't there something else we can do?

It is very good to make the screen "a virtual metaphor for the real world," but we designers owe more to our users. It is interesting the authors chose the calculator as the sample for this metaphor. The real world first had abaci and the old-fashioned, hand-crank adding machines statisticians used to love. Fortunately, the inventors of the calculator improved on these designs with technology advancements. We need to continue in this vein to make the world a little friendlier.

The screen as a metaphor for the real world is very challenging. We must determine not only what the real world looks like, but also what the intended using populace perceives it to be like. Let's take into account all five senses. Ask our users (and ourselves) not only what the object looks like, but what it sounds like, feels like, smells like, tastes like. Only then can we understand what it behaves like.

The authors discuss resizing text through pull-down menus. Why not let the user resize text by clicking and dragging? Seems faster and friendlier than bringing up a menu bar, scrolling through it, and picking a font size only to have it be wrong.

Concerning "acting for the user," the authors avoid discussing consequence determination. A great thing about machines is that, if properly designed, they protect us from our own humanism.

(I can't shove my car into reverse while going forward at 65 mph.) Conversely, ill-thought out design lets me do some incredibly stupid, but very human things, like erase the wrong disk. I am reminded of a cartoon in which the boss is leaning over the assistant's shoulder screaming "Oh no! You did it exactly the way I asked you to!!!"

The real world can be scary and unfriendly. Good GUI shows me that. Really good GUI prevents it.

STEVEN R TURNER
SPRINGFIELD, VA 22151

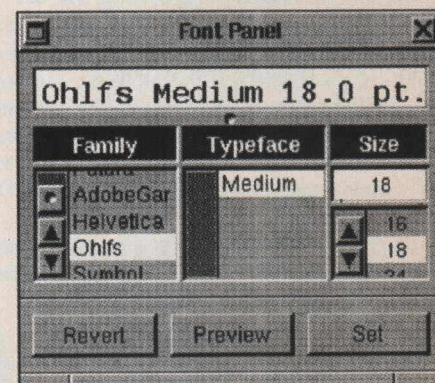
Author's Response: Mr. Turner makes some excellent points about GUI design. Any designer would be wise to take them into consideration.

As for resizing text through direct manipulation—this is an interesting idea that may be taken advantage of often enough to be worth implementing. Almost anytime the user clicks on text, the purpose is to position a cursor. The typical user often wants to select text and perform editing operations on it, but rarely wants to resize the text. And when a user wants to resize the text, they usually also want to choose a different font for the text at the same time.

Given this, most GUIs provide mouse functionality that lets the user easily manipulate the contents of the text. Other less common operations are performed through menus.

Yet Mr. Turner does make a very valid point even here—the user should not have to go through an arduous process to give a command (such as

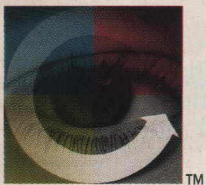
changing font size) only to get an undesirable result. To avoid this situation, NEXTSTEP always tries to provide feedback on what the result of a serious operation will be before the user decides to make the change.



In this case, NEXTSTEP provides a font panel. The user can call up a font panel to select the desired font family, typeface, and size, and most importantly, can see what their choice looks like before committing to it. Sample text in the font panel changes to show the appearance of the selected font and size. The font panel combines common operations and helps the user keep from making a mistake—hopefully an example of really good GUI.

RONALD HAYDEN
MANAGER OF DEVELOPER PUBLICATIONS,
NEXT

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



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.

EXECUTIVE BRIEF

Compiled by Michael Kei Stewart

Microsoft, Candle propose CORBA/OLE linking spec...Ending months of speculation, Microsoft Corp has finally moved to ease the fears of developers regarding incompatibility between OLE and the Object Management Group's Common Object Request Broker Architecture (CORBA). At a meeting of the OMG in Dublin, Microsoft announced support for a specification created jointly with chief developer Candle Corp that will provide interoperability between CORBA and the Common Object Model (COM) on which OLE is based, via a translation process that takes advantage of an "object adapter" in the second release of CORBA, CORBA 2. The proposal also has the backing of Digital Equipment Corp, which also helped develop COM. It is now up to the OMG to weigh this proposal against any others it may receive. The group expects to arrive at a final specification in the next six to ten months. Analysts agree that Microsoft's participation in an interoperability standard is important to its success, and that Microsoft has also realized the importance of interoperability to OLE's success.

Mainframes are back via objects, says IBM...IBM is heavily promoting the use of object technology as the method of choice for revitalizing both the role of mainframes and the mountains of legacy code that run on them. It is announcing a class library set called the Application Class Library Services, and is expected to release commercial, MVS-based versions of its C++, Smalltalk, object-oriented COBOL, and SOMobjects, now in beta, by late next year. IBM's vision of mainframes as ideal servers for distributed data and applications, supported by object technology, is partially supported by a survey of CIOs conducted by Business Science International. The study reports high continued usage of mainframes in service organizations, and relatively low active replacement of mainframes with smaller systems. It agrees that mainframes are increasingly viewed by some as potentially effective in a client/server environment, but also cautions that some of the high mainframe interest stems from short-term budgetary or strategic restraint, preventing change. Other analysts also question IBM's ability to overcome mainframer resistance to object technology and to compete in the object arena with companies like Microsoft.

EDS picks Smalltalk for client/server development...EDS and ParcPlace Systems Inc announced that they have agreed to a master contract governing EDS' use of ParcPlace's Smalltalk-based VisualWorks client/server application development tool. EDS will use VisualWorks to develop and maintain applications for its customers in fields including the communications, financial, government, and manufacturing industries.

Oberon spins off UNIX SynchroWorks company...Oberon, which recently decided to discontinue development of the UNIX version of its CORBA-compliant SynchroWorks in favor of a Windows version, has formed a new venture, Distributed Object Technologies, to develop and sell the UNIX flavor of the visual application development environment. The new company, headed by former Oberon sales and marketing Vice-President Winfried Burke, also seeks VAR contracts with selected CORBA-compliant ORB vendors, with the intent of bundling SynchroWorks and the ORB to provide visual assembly of ORB components into applications. The company, which is actively seeking investors, will also investigate other distributed object computing tools for possible inclusion in its product line; provide distributed object computing training and management and development consulting services; and work with application vendors to decompose their applications into ORB object components for sale through a component catalog.

Microsoft, mainframer forging links...Mainframe software vendor SAP America Inc, maker of the R/3 integrated client/server application suite, announced support for Microsoft's OLE 2.0, and says the two companies will pursue joint marketing efforts to promote the Windows/NT version of R/3 released this year. Analysts say the OLE support will help R/3 work more easily with desktop applications; the relationship may also lend credibility to each company on its partner's home turf, giving Microsoft an easier entry to the corporate MIS world, and making SAP seem more approachable to desktop users.

Novell trims staff, drops Foundation...Smarting from disappointing fiscal third-quarter results, the effect of low sales of WordPerfect and Quattro Pro and the unexpectedly high costs of consolidating WordPerfect Corp, Novell Inc is cutting about 1,750 employees, and is reportedly phasing out support for its AppWare Foundation cross-platform development tool, which was largely funded by WordPerfect sales. The software, which allows developers to create a single set of code to run on Windows, OS/2, System 7, or UNIX, is included in the first release of Visual AppBuilder, but the company apparently does not plan to support it for more than a year or two, and will instead focus on development of the visual application creation portions of its AppWare program. Some analysts and users question whether Novell can be competitive with yet another GUI builder in an already crowded market.

Your most complete source on using, implementing and managing object technology...

OBJECT magazine

SAVE UP TO 21%



Enter my subscription at the term below. This is a risk-free offer. If I'm not completely satisfied I may cancel and get a prompt refund for the unused portion.

Personal Rate

- ☐ 1 Year (9 issues)
☐ \$39
☐ \$74 Outside U.S. (air service included)
- ☐ 2 Year (18 issues)
☐ \$62 **SAVE 21%**
☐ \$132 Outside U.S. (air service included)

Corporate/Library Rate

- ☐ 1 Year (9 issues)
☐ \$89
☐ \$124 Outside U.S. (air service included)
- ☐ 2 Year (18 issues)
☐ \$162 **SAVE \$16!**
☐ \$232 Outside U.S. (air service included)

Payment

- ☐ Check Enclosed ☐ Bill me ☐ Charge my: ☐ Visa ☐ MasterCard ☐ AmEx

Card # _____ Exp. Date _____
 Signature _____
 NAME _____
 TITLE _____
 COMPANY _____
 ADDRESS _____
 CITY _____ STATE _____ ZIP _____ COUNTRY _____
 PHONE _____ FAX _____

IMPORTANT: Make checks payable to Object Magazine. Orders for subscriptions outside the U.S. must be prepaid. Checks must be in U.S. dollars, drawn on a U.S. bank. Discounts based on 2 year rates before added shipping costs. Basic annual rate is \$39; corporate/library rate \$89.

RETURN TO: Object Magazine, PO Box 2029, Langhorne, PA 19047-9029 or FAX to: 215/785-6073

Please answer the following questions to help our editors better serve your needs.

1. My primary function is:

- ☐ 1 Corporate Management
☐ 2 Systems Integration
☐ 3 Systems Architect, Business/Syst. Analyst
☐ 4 Technical Mgr., Group/Project Mgr.
☐ 5 Programming, Hardware/Software Devel.
☐ 6 Engineering, Scientific, R&D, Education
☐ 7 Consulting
☐ 8 Other _____

2. My company's primary business is (check one):

- ☐ A Manufacturing (computer or data systems hardware & peripherals)
☐ B Manufacturing (other than computer)
☐ C Systems Integration, VAR, Computer Service Bureau
☐ D Software Development
☐ E Financial, Banking, Insurance, Real Estate
☐ F Retail, Wholesale, Distribution
☐ G Utilities, Communication Services, Transp.
☐ H Government and/or Military
☐ I Health Services/Institutions
☐ J Education, Consulting
☐ K Agriculture, Mining, Construction
☐ L R&D, Testing, Evaluation Labs
☐ M Other _____

3. The language I am most interested in is:

- ☐ 1 Eiffel ☐ 5 C++
☐ 2 Smalltalk ☐ 6 CLOS
☐ 3 Objective C ☐ 7 Proprietary
☐ 4 Object Pascal
☐ 8 Other _____

4. The type of platform I work on is:

- ☐ A Mainframe
☐ B Mini
☐ C PC
☐ D Workstation
☐ E Macintosh
☐ F Other _____

Signature _____ Date _____

OBJECT magazine

SAVE UP TO 21%



Enter my subscription at the term below. This is a risk-free offer. If I'm not completely satisfied I may cancel and get a prompt refund for the unused portion.

Personal Rate

- ☐ 1 Year (9 issues)
☐ \$39
☐ \$74 Outside U.S. (air service included)
- ☐ 2 Year (18 issues)
☐ \$62 **SAVE 21%**
☐ \$132 Outside U.S. (air service included)

Corporate/Library Rate

- ☐ 1 Year (9 issues)
☐ \$89
☐ \$124 Outside U.S. (air service included)
- ☐ 2 Year (18 issues)
☐ \$162 **SAVE \$16!**
☐ \$232 Outside U.S. (air service included)

Payment

- ☐ Check Enclosed ☐ Bill me ☐ Charge my: ☐ Visa ☐ MasterCard ☐ AmEx

Card # _____ Exp. Date _____
 Signature _____
 NAME _____
 TITLE _____
 COMPANY _____
 ADDRESS _____
 CITY _____ STATE _____ ZIP _____ COUNTRY _____
 PHONE _____ FAX _____

IMPORTANT: Make checks payable to Object Magazine. Orders for subscriptions outside the U.S. must be prepaid. Checks must be in U.S. dollars, drawn on a U.S. bank. Discounts based on 2 year rates before added shipping costs. Basic annual rate is \$39; corporate/library rate \$89.

RETURN TO: Object Magazine, PO Box 2029, Langhorne, PA 19047-9029 or FAX to: 215/785-6073

Please answer the following questions to help our editors better serve your needs.

1. My primary function is:

- ☐ 1 Corporate Management
☐ 2 Systems Integration
☐ 3 Systems Architect, Business/Syst. Analyst
☐ 4 Technical Mgr., Group/Project Mgr.
☐ 5 Programming, Hardware/Software Devel.
☐ 6 Engineering, Scientific, R&D, Education
☐ 7 Consulting
☐ 8 Other _____

2. My company's primary business is (check one):

- ☐ A Manufacturing (computer or data systems hardware & peripherals)
☐ B Manufacturing (other than computer)
☐ C Systems Integration, VAR, Computer Service Bureau
☐ D Software Development
☐ E Financial, Banking, Insurance, Real Estate
☐ F Retail, Wholesale, Distribution
☐ G Utilities, Communication Services, Transp.
☐ H Government and/or Military
☐ I Health Services/Institutions
☐ J Education, Consulting
☐ K Agriculture, Mining, Construction
☐ L R&D, Testing, Evaluation Labs
☐ M Other _____

3. The language I am most interested in is:

- ☐ 1 Eiffel ☐ 5 C++
☐ 2 Smalltalk ☐ 6 CLOS
☐ 3 Objective C ☐ 7 Proprietary
☐ 4 Object Pascal
☐ 8 Other _____

4. The type of platform I work on is:

- ☐ A Mainframe
☐ B Mini
☐ C PC
☐ D Workstation
☐ E Macintosh
☐ F Other _____

Signature _____ Date _____

Lock in your savings...

Take out a 2-year subscription to **OBJECT MAGAZINE** today!

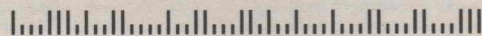


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 2034 LANGHORNE, PA

POSTAGE WILL BE PAID BY ADDRESSEE

OBJECT MAGAZINE
PO BOX 2034
LANGHORNE, PA 19047-9711

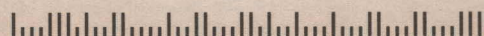


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 2034 LANGHORNE, PA

POSTAGE WILL BE PAID BY ADDRESSEE

OBJECT MAGAZINE
PO BOX 2034
LANGHORNE, PA 19047-9711



Don't even think about using a Relational Database System !



Use the POET Object Database for C++

C++ and class libraries have made the GUI development much easier. After all, it is only logical that more and more developers think in objects. But object orientation shouldn't end at the user interface programming level.

The Problem: Without POET, a C++ programmer must use flat files or a RDBMS to store objects. He has to write code to overcome the mismatch between the application and the database model. This leads to design restrictions, performance penalties and more code to write and maintain.

The Solution: POET operates at the object level, it speeds up the development process and provides greater performance. Furthermore, the developer can simply take the object-oriented application design in C++ and map it 1 to 1 into the database. Without any compromise at all!

Full featured database:

POET provides complete support for Encapsulation, Inheritance, Polymorphism and Containers as well as queries, object locking, and transaction handling.

True cross platform support: Complete interoperability makes the development of network enabled applications a breeze. POET supports Windows 3.1, Windows NT, Win32s, Windows for Workgroups, Novell Netware (NLM), SUN, AIX, HP-UX, SGI, SCO, Macintosh, Power Macintosh, OS/2 and NeXTStep.

Comparing Object Oriented and Relational Design Methodologies

Feature	POET ODBMS	Relational RDBMS
Storing Objects	As Objects	Break Objects into Tables
Database Model	User Application Model	Separate Database Model Required
C++ Integration	Total	Poor
Database Operations	At Object Level	Must Write Code
Productivity	Increased	Reduced
Complex Object Performance	Excellent	Poor

Call 1-800-950-8845



POET
Software

Circle 6 on Reader Service Card

POET Software Corporation, 4633 Old Ironsides Dr., Suite 110, Santa Clara, CA 95054, Tel.: (408) 970-4640 Fax: (408) 970-4630
United Kingdom (Silicon River) +44 81-316-7777, Germany (POET Software) +49 40-60-99-00, Australia (Microway) +61-3-580-1333, France (LCI) +33-1-34-65-7777
Netherlands (Protools Software) +31 20 645 5023, Sweden (Duke Systems) +46 8 703 2781

IDC projects continued rapid growth in ODBMS market... According to a new study from market research firm International Data Corporation and consulting firm Barry Associates, demand for object database management systems grew 110% in 1993, and is expected to continue growing rapidly over the next five years. From a 1993 level of about \$76 million, the ODBMS market is expected to reach \$1.5 billion by 1998. The report provides a discussion of ODBMS technology, trends, and selection criteria, and describes 12 vendors and products, including Illustra, Objectivity, ONTOS, Object Design (the 1993 market share leader, according to the report), Poet, and Versant, in detail.

Lotus joins CIL but keeps object options open... Component Integration Laboratories Inc, the industry consortium promoting adoption of its OpenDoc compound document architecture, announced that Lotus Development Corp has joined the group as a full member. Lotus has already shipped products that support Bento, the portable object storage format that is part of the OpenDoc architecture along with SOM and other technologies. Lotus itself helped create Bento in collaboration with Apple Computer Inc and other organizations. John Landry, Chief Technology Officer of Lotus, said his company is "seriously evaluating" SOM, and will evaluate other parts of OpenDoc as they develop. Membership in CIL gives Lotus access to changes in important object technologies such as Bento, but the company will also continue to use OLE technology, according to General Manager of Product Line Architecture Alex Morrow. He said the company is also looking at technology developed by other companies, such as Taligent and NeXT, for use in its products. No dates were announced for release of products containing CIL technology.

OpenDoc SDK lacks full network object sharing... Delays in porting some SOM runtime libraries to System 7 mean that the first SDK release of the OpenDoc compound document architecture will not completely support the sharing of objects over networks. Releases from WordPerfect and Apple Computer Inc will probably not include full Enterprise Distributed SOM (DSOM) support. The libraries, known as Workgroup modules, allow object sharing across networks using TCP/IP, IPX/SPX, and NetBIOS protocols. Their lack in the first release apparently means that OpenDoc applications on OS/2 and Windows will be able to share objects with each other, but not with Macintosh applications. Wendy Tajima, OpenDoc Product Line Manager for Apple Computer, says her company will complete the port, but did not give a release date.

NeXT releases OpenStep spec... NeXT Computer Inc announced that the final version of the OpenStep operating-system-independent object standard specification is available via anonymous ftp from NeXT's archive server, ftp.next.com. NeXT, Sun Microsystems Inc, Hewlett-Packard Company, and Digital Equipment Corporation have all committed to shipping OpenStep-compliant products in 1995, using NeXT-created implementations of the specification. NeXT is also said to be considering a port to Windows 95, the next version of Windows.

Next has also introduced an on-site support program, Object Experts, to help developers creating NEXTSTEP applications, and said that Martin Marietta Information Systems & Technologies and Tensor Information Systems have both joined NeXT's Object Channel program and will use NEXTSTEP and OpenStep in developing applications for customers.

Microsoft, Symantec plan C++ migration kit... Microsoft Corp and Symantec Corp are reportedly planning a toolkit that will help developers port their 16-bit C code to 32-bit C++ Microsoft Foundation Class (MFC) code. The kit should make it easier for developers to reuse 16-bit code in developing applications for Microsoft's targets of choice, Windows NT and Windows 95 (Windows 4.0). Symantec, MetaWare Inc, Watcom International Corp, Bristol Technology Inc, and MainSoft Corp are said to be licensing MFC for their development products. Meanwhile, Microsoft is continuing its cross-platform Visual C++ push, with versions for Macintosh and Windows NT running on MIPS systems at or close to completion, and with a version for DEC's alpha chip reportedly planned.

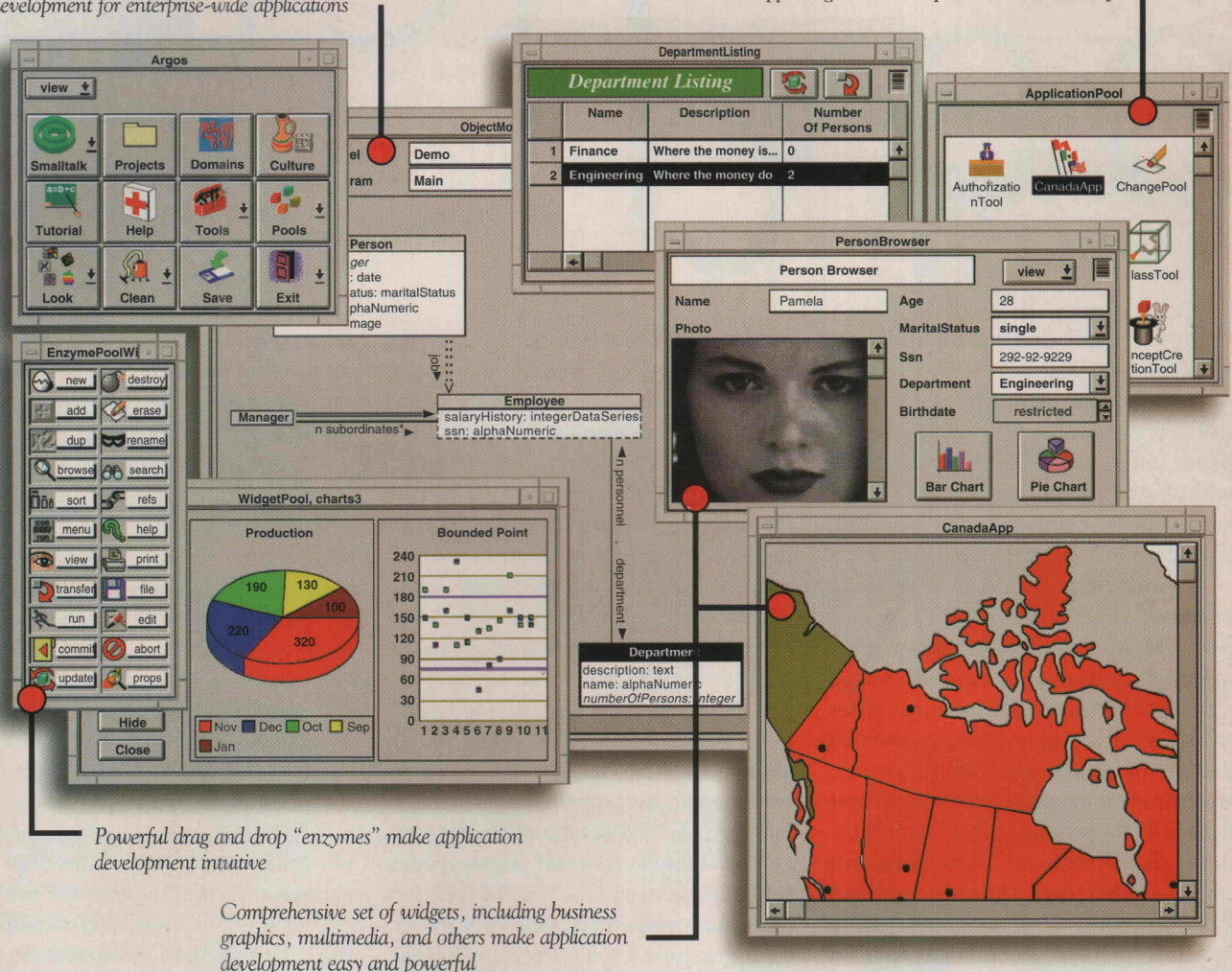
New Visual Basic to ease OLE app creation... The next version of Visual Basic, Version 4, now undergoing beta testing, reportedly features an "object-oriented rewrite" of its language engine and easier integration with third-party tools. Most significantly, however, it will also include support for the creation of OLE automation server objects such as Line of Business Objects, which developers must now use Visual C++ to accomplish. OLE automation support means programmers can more easily create software components, objects that perform specific functions and can also be accessed and manipulated by other applications. Easier OLE object creation could give OLE 2.0 a huge boost. Microsoft is also reportedly considering making Visual Basic the scripting language for Cairo, the object-oriented operating system, but will not make a final decision for some time. Microsoft also has not announced a release date for the new version of Visual Basic.

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

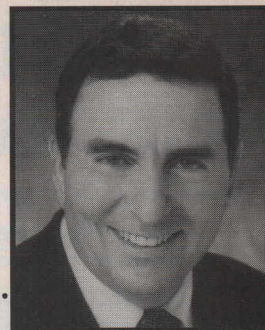
control transparently. And only Argos is packaged as a completely visual ADE built on ParcPlace VisualWorks®.

Find out why 3 of every 4 customers who've seen Argos purchase it to deliver their business-critical applications in weeks rather than years. If your organization needs objects today, let us prove it to you with Argos.

VERSANT
The Database For Objects™

**Contact us today at
1-800-VERSANT, ext. 416
or via e-mail at
info@versant.com**

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



James A. Cannavino

The future of object technology

THE FIERCE GLOBAL competition of the 1990s has made today's business environment a battleground for many companies. Demand for more service, better value, and new products, coupled with shorter development times and decreased profit margins, have strained the capabilities of many corporate departments.

All businesses must deal with the accelerated pace of change as part of their standard operating procedure. Success in the marketplace depends on the ability to capitalize on fleeting market opportunities, which, in turn, requires the fast, accurate delivery of information. This puts an extraordinary burden on information technology (IT) organizations—they must understand and respond to user needs with unprecedented speed. Object technology is a key to meeting these changing needs.

Historically, IT organizations have not had complete access to the mainstream of day-to-day business decision-making. It was felt that programmers should be segregated from users so that they could produce code without distraction. Users were kept ignorant of systems issues, which, it was felt, they were unable to understand anyway. This gap between providers and consumers gave rise to support groups that served as mediators between system requirements and business needs. The result was a convoluted path of communication, a lack of synergy, and a lot of wasted energy.

Today's business requirements, marked by an ever-growing need for responsiveness, flexibility, and decision support across plat-

forms, locations, and companies, will not be met effectively with many of the traditional programs and environments.

Closed systems are giving way to open environments. Distributed, client/server, network-centric environments are the path to the future, but must also support existing environments so that current investments in hardware, software, applications, and data can be protected.

The huge, feature-laden applications that have become omnipresent are starting to give way to smaller, more nimble, component-based applications. Responsiveness and the ability to adapt to individual needs is becoming more significant than the everything-into-the-pot school of software development.

The ability to pick and choose specific functions as needed will make for more dexterous computing and is another necessary element in the process of reengineering tasks, systems, and entire businesses. Object technology is helping to make this transition from application-oriented to task-centered computing. The industry is looking for a value continuum—a common infrastructure and set of tools that links the elements of the process and leverages the efforts of every individual and group. Object technology facilitates the creation of this value continuum, ensuring each link in the chain leverages previous results and adds unique value. It allows a logical progression of development, support, and improvement.

In object-oriented systems, the goal is the development of robust and flexible solutions. Newer operating systems allow developers to take advantage of new hardware platforms while at the same time protect-

ing their existing investments. Existing systems are not necessarily replaced wholesale, but often merely augmented by the gradual introduction of objects.

These fundamental changes are being brought about by the adoption of object technology and will occur at the broadest level. In the future, the monolithic systems supplied by single vendors will give way to systems developed as standards, by strategic alliances, and through partnerships and collaborations. Multiple cooperative vendors will be the norm. Most importantly, object technology allows customers to meet their vertical business needs as well as cross-industry requirements.

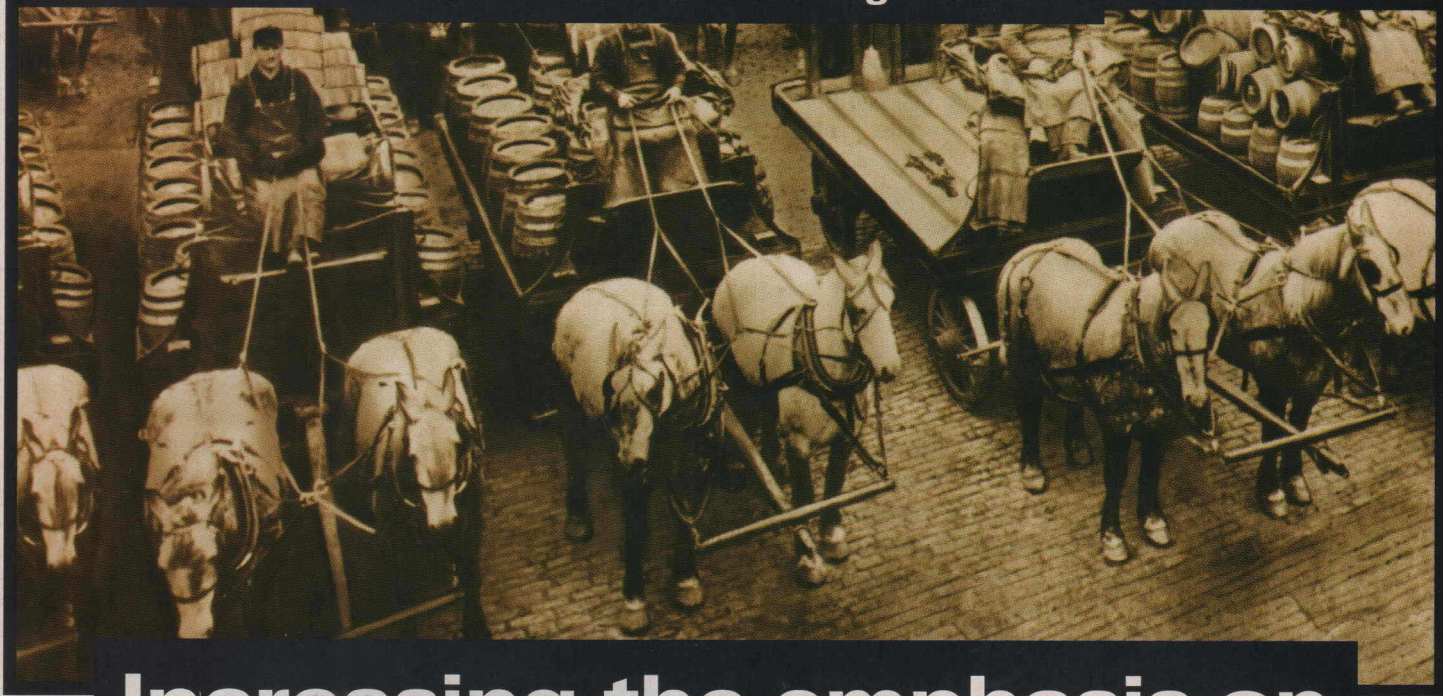
New object-oriented environments will allow users and programmers to think about functional requirements instead of worrying about hardware limitations. The rapid change that has become the status quo of today's business environment is easily handled by the value continuum. And the improved communication that this fosters makes successful business transition more tangible.

IBM has responded to these new business challenges with a four-part technical strategy providing an infusion of object technology into every segment of the value continuum:

- Distribution services technology makes it possible to share objects across platforms, operating systems, and programming environments.
- Component technology supports the building and integration of the reusable application building blocks that address the inevitable industry trend toward multi-

James A. Cannavino is Senior Vice President of Strategy and Development at IBM Corporation, Armonk, NY.

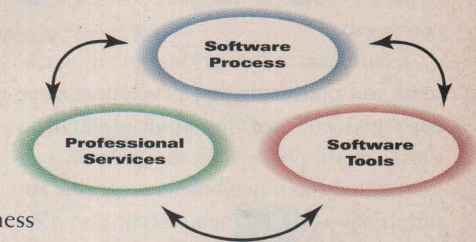
The rationale for choosing Rational.



Increasing the emphasis on reducing time to market.

In the headlong rush to develop new technologies, some companies would like you to believe that innovation is a goal unto itself.

At Rational, we take a different view. And that is, technology is simply the means to a profitable end. If information is what drives business in the 90s, the first step toward devising a winning software strategy lies in linking information technology goals with overall business goals. In converting emerging technologies from attractive novelties into competitive weapons with which you can gain a measurable business advantage.



The Rational Approach



Going one step further, the faster you can create application software the better your chances of success. Of course, time to market is not the only issue. The new business environment also demands a new level of return on investment, so application reuse, cost and schedule predictability, as well as increased software quality are also key factors.

It can add up to a rather daunting problem, to say the least. We call our solution The Rational Approach. And it's working today for more than 10,000 people in every major industry in the world. Give us a call at 1-800-767-3237 and we'll show you how Rational can help move your business forward – at a full gallop.

RATIONAL
SOFTWARE CORPORATION

Rational Software Corporation, 2800 San Tomas Expressway, Santa Clara, CA 95051-0951 (408) 496-3600. Canada 613-599-8581; France +33-1-30-12-09-50; Germany +49-89-797-021; U.K. +44-273-624814; Sweden +46-8-761-0600; Australia +61-3-521-0507; Taiwan +886-2-720-1938. Representatives: Italy +39-2-264-0107; Spain +34-1-579-21-13; Israel +972-3-575-55-50; Japan +81-03-3779-2541

©1994 by Rational Software Corporation. The words "Rational" and "The Rational Approach" as well as Rational's products are trademarks of Rational Software Corporation. References to other companies and their products use trademarks owned by the respective companies and are for reference purposes only.

Circle 8 on Reader Service Card

vendor, multiplatform, task-centered environments and compound documents.

- Enablers and tools (compilers, browsers, libraries, and the like) empower users to fully utilize object technology while maximizing existing tools and skills.
- End-user environment products reshape the user experience by creating a knowledge space where computers serve as a catalyst toward increased productivity, by leveraging learning and capitalizing on a far-flung base of information and experience.

Our approach to distribution services is the IBM System Object Model (SOM). SOM is a software standard developed to assure the portability of objects across platforms and development languages. It is an object packaging technology that is language neutral, platform independent, and already has the support of many third-party vendors.

Key enterprise-wide challenges such as diversity, multiple standards, and constant change are met by SOM. Today, SOM is the only software mechanism available that links disparate elements and bridges implementation gaps.

In addition to the various operating systems and platforms that developers must cope with, end users face multiple applications—word processors, spreadsheet programs, and their own business-specific applications—that are inconsistent, complex, and don't work well together. One of the greatest benefits of component-based software will be the ability to generate and integrate dynamic, customized, people-oriented applications solutions.

OpenDoc from CI Lab, the non-profit corporation formed by industry leaders including Apple, IBM, Novell, Taligent, WordPerfect, and Xerox, is a standard component software architecture for creating reusable software parts that can be customized and assembled by end users into applications and documents. OpenDoc parts coexist and can be integrated as compound documents—files that contain elements from multiple live applications. Even disparate types like text, formulas, graphics and multimedia can be seamlessly combined in a compound document running in a heterogeneous environment.

The OpenDoc metaphor of parts represents a significant change in the way software is developed and user environments are created. It encourages business solutions across competitive platforms and networks rather than promoting a single standard. It allows for the combination of input from disparate sources rather than fostering the bigger-is-better attitude toward application development that predominates today. It also facilitates an orderly evolution of existing applications and data into open environments.

The capabilities of OpenDoc are not to be confused with Object Linking and Embedding (OLE), Microsoft's standard for

**The convergence of a
number of new products
based on object technology
is creating the energized
end-user environment
of the future, which builds
upon advances in hardware
and software.**

data sharing. OLE is a Windows-only solution that works only at the data level to integrate Windows applications. OpenDoc is a superset of OLE and will interoperate with OLE applications, and OLE parts will work in OpenDoc. OpenDoc's level of interoperability is an integral part of the operating systems that support it. It is facilitated by OpenDoc's use of SOM as its distribution mechanism.

The benefits of component parts and compound documents can be extended beyond the creation and use of commercial desktop software to include enterprise-wide systems. The creation of reusable elements utilizes the learning and experience of individuals, as well as the long-term efforts of organizations. Extending reusability to the component level encourages collaboration and streamlines the entire business.

Within the enablers and tools context, IBM has devoted considerable energy to ob-

ject development in the past several years including considerable support in its new versions of C++ and Smalltalk.

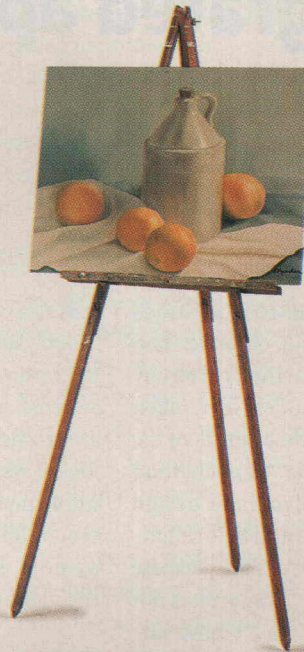
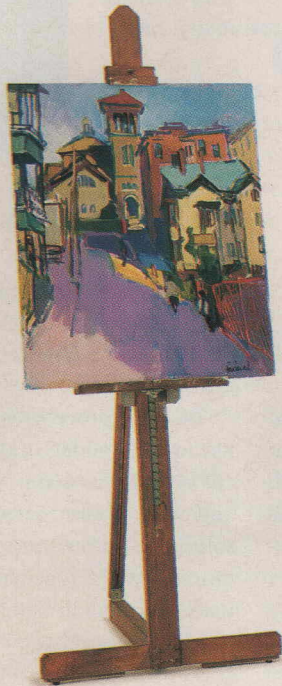
The ability to save objects in collections as libraries or frameworks for future development is one of the key advantages of object technology. However, building a class library can be a difficult and time-consuming task. To meet this need for reusable code, libraries, and frameworks, many vendors including IBM supply collections of general-purpose objects such as graphical user interfaces, database access, communications, and task management. There are also more vertically oriented libraries known as application classes (ACLS) that address the needs of a specific industry or system function.

In contrast to class libraries, frameworks optimize the relationships between objects, thus simplifying the process of assembling applications. Taligent has taken the idea of frameworks a step further to include everyday working environments, giving users the ability to customize and/or design new frameworks. The Taligent Application Environment has the support of Apple and Hewlett-Packard in addition to IBM.

The convergence of a number of new products based on object technology is creating the energized end-user environment of the future, which builds upon advances in hardware and software. This environment requires cooperation among programs so that people can concentrate on their tasks rather than spending time manipulating applications. It empowers users and enhances their decision-making capability by enabling them to access and analyze data and transform it into actionable information. This new user environment is comprised of distributed information and reusable components. It extends what users can do individually, across collaborative workgroups and even enterprise-wide.

Users will move from today's application-centered environment to a task-centered computing world based on object technology—becoming more productive than ever. IT shops will be able to respond to user needs with lightening speed, and organizations will have a new competitive edge. Object technology is not only IBM's future, but the future of computing as well. ■

CHOOSING AN OO METHODOLOGY?



NO MATTER WHAT YOUR APPROACH, OBJECTTEAMTM GIVES YOU A MASTERPIECE.

Most Complete Rumbaugh Automation

- Full support for the Object, Dynamic, and Functional models
- Automatic import and reuse of C++ class libraries and custom subsystems
- Advanced diagramming features including color, multiple fonts, free-form graphics, and user-configurable navigation
- Customizable generation of C++, Ada, SQL, and OODBMS
- Direct output and document generation to Interleaf, FrameMaker, and PostScript files
- Multi-user support for Windows and UNIX platforms

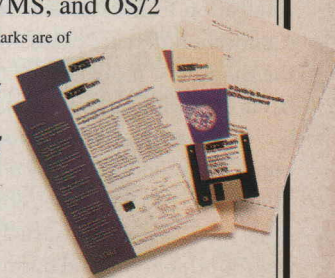
Includes Paradigm Plus/Cadre Edition, licensed from ProtoSoft Inc.

Most Complete Shlaer-Mellor Automation

- Automation of the entire methodology including support for Recursive Design and all graphics and notations
- Full semantic support for Domains, Subsystems, and OODLE design notation
- Comprehensive checking for identifying defects in analysis
- Code generation for C++ and Ada
- Automatic generation of Shlaer-Mellor Subsystem Notebooks to Interleaf, FrameMaker, and PostScript files
- Built-in versioning of all Shlaer-Mellor work products
- Multi-user support for UNIX, VMS, and OS/2

ObjectTeam is a trademark of Cadre. All other trademarks are of their respective holders.

ObjectTeam from Cadre includes the tools, educational services, and award winning support necessary for you to be successful in OO development. Our products and services have been proven successful in over 30 countries. Call today for your FREE OO Success Kit. In North America: (800) 743-2273.



CADRE

FIRST CHOICE IN SOFTWARE DEVELOPMENT AUTOMATION

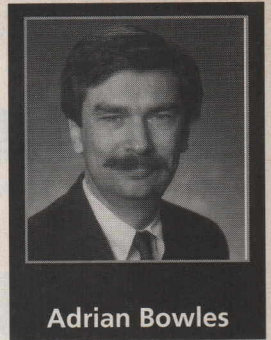
North American Operations

Cadre Technologies Inc., Providence, RI Tel: (401) 351-5950 Fax: (401) 455-6800

International Operations

Cadre Technologies SA, Nyon, Switzerland Tel: [41] (22) 362-2251 Fax: [41] (22) 361-0785

An integrated approach to modeling



ANALYSIS AND DESIGN methods guide the construction of models and systems. Current methods typically use a number of models that, taken together, give a complete description of the system under construction. Most method vendors decompose analysis and design models into some combination of high-level, detailed user interface design, database modeling, etc. With CASE support it is not unusual to see multiple models whose various representations are linked by an underlying repository or data dictionary. In an attempt to hide the complexity of the real world being modeled, we have created another level of complexity that requires software for model management.

This is similar to the modeling approach used by architects in which numerous representations are prepared to communicate with different types of users. These include various elevations, site plans, HVAC (heating, ventilation, AC) diagrams, etc., of interest to everyone from the future occupant to the plumbing contractor. The complexity of these models has made CAD indispensable for large construction projects.

However, although we frequently draw inspiration from architecture, at least one significant difference exists. In architecture, the maps and models have to be distinct from the building to be constructed. Nobody can progress from a model of a building to the building itself by adding form or functionality to the model. Software, on the other hand, presents just such an opportunity. Information-based businesses also pre-

sent this opportunity when a change to the model can directly change the behavior of the business. If our task is to build logical systems, we should look more carefully at areas where we have been successful with logical systems. Rather than focusing on the implementation or physical aspects of system building, perhaps it is time to look for a more logical approach to modeling throughout the lifecycle.

DEVELOPMENT AS PROBLEM SOLVING

One way to look at the development process is to compare it to problem-solving be-

**Rather than focusing
on the implementation
or physical aspects of
system building, perhaps it
is time to look for a
more logical approach to
modeling throughout
the lifecycle.**

havior, as studied by psychologists and AI researchers. A simplified approach to problem solving has three steps: intelligence, design, and choice (see Simón¹ for a more detailed account of problem solving). Intelligence is the gathering of information relevant to the problem to be solved. Design is a process of generating and evaluating alternative courses of action (COAs). Choice is the selection and execution of the COA

that scores highest according to some objective criteria.

Business process reengineering models, like formal models of problem solving, focus on representation of the current state and one or more goal states that represent solutions. Automating problem solving requires a system that can generate and evaluate intermediate states, working forward to a final state or backward to the start state. If there are several possible paths to several possible goals, the problem may change to finding the optimal path that can be determined given some time or computation constraint, rather than finding a true optimal path. This is "satisficing," rather than best fit, but it is better than the first fit approach, which is still the general rule in software development where time constraints prevent designers from evaluating many options.

A well-understood specialization of problem solving is game theory. If we examine chess programs, we find that they are based on the fact that chess is a two person, zero sum, perfect information game. That is, the problem for each player is to identify and execute the best move given that any advantage for one player comes at a cost to the opponent, and both players have complete information at their disposal. There are complete, consistent, and unambiguous rules that govern the game, and both players see the current state and can theoretically derive all possible future states of the game board.

The difficulty arises from the fact that in practice a player has a limited number of future moves that can be evaluated.

Some aspect of a chess game, like some aspects of an evolving application, are easy to model. The arrangement of the pieces

DON'T GAMBLE with other CASE support on your next Object-Oriented software project!

NEW!
• Object Interaction Diagrams
• CD-ROM "Overview of OO Methods"
• Requirements, test cases, bug reports...

Your odds without ICONIX PowerTools™

- 20 to 1 with tools that claim to be O-O but are really structured tools in disguise...
- 25 to 1 with CASE vendors that don't offer O-O methodology training...
- 40 to 1 with single-method tools that support a limited portion of the lifecycle...
- 50 to 1 with single-user (toy) tools that don't support server-based development...
- 100 to 1 with hard-to-use tools that take a team to make it work...
- 200 to 1 *just hack the code!* Don't bother with this methodology and CASE stuff.

Beat the odds with ICONIX PowerTools™

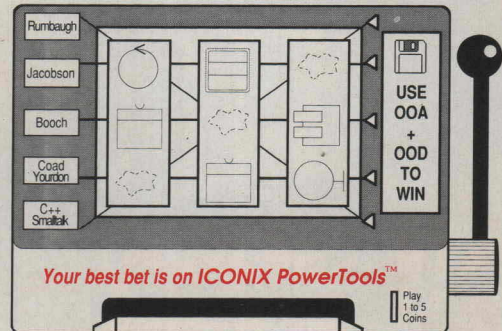
ICONIX PowerTools supports virtually all major OOA and OOD methods.

ICONIX provides training for methods, tools and languages including guidelines for choosing the best methods for your project and details of individual methods.

ICONIX PowerTools has full-lifecycle capabilities, including Class, Object, Use-Case, State, Process and Module Architecture Diagrams, Language Sensitive Editors for C++, Smalltalk, SQL, Ada and others, and Integrated Requirements Traceability.

ICONIX PowerTools are multi-user tools that support projects of any size.

ICONIX PowerTools features "standard Mac" ease-of-use.



We've been providing affordable, industrial-strength, easy-to-use CASE products on the Macintosh since 1985, and have been leading the way in Object-Oriented methods since 1988. Call us today at **(310) 458-0092** and find out why the smart money is on **ICONIX!**

ICONIX Software Engineering, Inc. • 2800 28th St. Suite 320 • Santa Monica, CA 90405 • FAX (310) 396-3454 • Applelink: ICONIX

©1993 ICONIX Software Engineering, Inc. ICONIX PowerTools™ is a trademark of ICONIX. All rights reserved. Macintosh is a trademark of Apple Computer, Inc.

ICONIX
Leadership in Object Technology™

Circle 10 on Reader Service Card

and the mechanics of verifying legal moves are simple. Assigning a meaningful value to a possible future state and optimizing the generation and evaluation of moves are more difficult tasks. For example, in chess there are point values associated with the loss or capture of a piece, but a strategy that loses a piece on the current move and ensures a victory in a subsequent move will only be seen if the player has sufficient look-ahead power. Current programs—and human players—have limited look-ahead ability and must therefore have a good algorithm for evaluation under uncertainty.

We have solved the simple representation problems for application development. Method and tool vendors should now take advantage of research on knowledge acquisition (the KADS method from the European Esprit project, for example) and problem-solving behavior. It is time for application development methods and tools to attack the difficult problems.

APPLYING THE LESSONS

Business process modelers solve problems that are very similar to building chess games. They must define the current state of the business and define and evaluate multiple "what-if" future states. The basis for evaluation is usually some combination of cost, quality, and time to market considerations. This means that any BPR modeling technique should include representation and evaluation criteria for these three dimensions.

It would be sensible to adopt a modeling approach for information systems development that uses the basic problem-solving concepts we have outlined here and includes evaluation of the business criteria as well as the technical criteria most frequently associated with systems design. The same underlying model could then be used from the earliest planning through post-implementation enhancement. Technical decisions made during systems design have cost consequences that are usu-

ally ignored by current design methods and tools.

An integrated model processor should provide the same type of look-ahead with evaluation that game programs use. Analysis and design should also include simulation capabilities and costing of alternative COAs. Unless we make simulation and costing an integral part of the analysis and design methods, we will continue to develop and deliver extremely suboptimal solutions on a first fit—rather than best fit—basis. When it is infeasible to model and evaluate every contingency, simulation should always be part of the evaluation process.

It is important to develop appropriate scoring algorithms. Today the state of the practice for planning, analyzing, and designing information systems is more art than science or engineering. Some areas of design—network optimization, for example—are understood well enough that algorithms can be embedded in tools. The



The Only Solution
is
**REUSABLE
SOLUTIONS**

- Client-Server Applications
- Multimedia
- Object-Oriented Design/Mentoring
- Smalltalk Programming
- OODBMS, Relational DB Systems
- Operations Research Projects

Call us for SOLUTIONS to your object-oriented challenges.

**REUSABLE
SOLUTIONS**

1130 S.W. Morrison St. • Suite 318 • Portland, Oregon • 97205
(503)223-6892 • FAX (503)223-4498

Circle 11 on Reader Service Card

average software developer with a CASE tool has little or no time to evaluate more than one or two design alternatives before making a commitment to one approach and moving on to the next decision. In chess, this would correspond to a very shallow look-ahead, which in turn corresponds to a losing strategy.

We don't need more models, we need more intelligence in some of the models we already have. Adding basic evaluation should be commonplace in a future generation of application development environments. Rather than completely automating and hiding the evaluation from a user, however, we must make it visible. The alternatives to be considered are more complex than chess moves, and for some time we will want an experienced user to make the final decisions. Simulation will be required to help the user select the appropriate move or design alternative.

GETTING STARTED

The key to integrating problem-solving

techniques successfully with the simulation and costing/evaluation benefits of BPR methods/tools and the state of the practice leaders in object modeling for analysis and design is to carefully select complementary methods and tools. As no single vendor is likely to provide a complete solution in the immediate future, leading users must plan to integrate tools and methods themselves. I expect some organizations to adopt a mix-and-match strategy tailored to the preferences of the business analysts or domain experts.

A number of vendors are offering BPR modeling tools that fit my criteria. Three of the best are FirstStep from Interfacing Technologies (Quebec), iThink from High Performance Systems (Hanover, NH), and Analyst from Action Technologies (Berkeley, CA). Each has a unique approach to representing the current and to-be scenarios. Of the three, FirstStep will probably be the easiest to integrate with an object-oriented application development environment because it was implemented

using object technology (VisualWorks from ParcPlace).

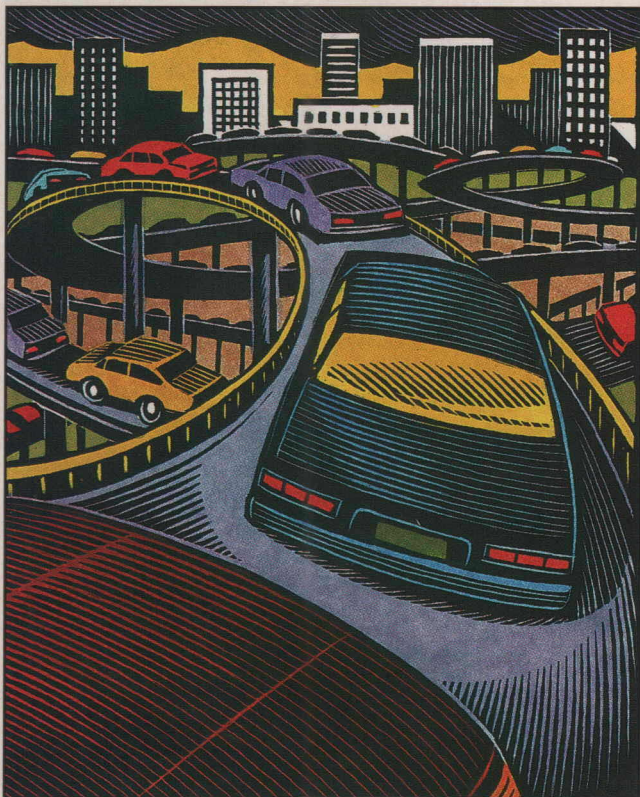
The best approaches to object modeling that I have seen—from the standpoint of integration potential—come from Enterprise Engines and Objectory (David Taylor and Ivar Jacobson, respectively). Both have carefully considered the role of object models in business analysis. In addition, Taylor has incorporated costing information and simulation directly in his modeling tool.

Users will have to define their process in problem-solving terms before selecting candidate products, which will require research or acquisition of new skills for developers. The integration of tools and techniques from different worlds will not be inexpensive or simple, but the results can be dramatic. A judicious investment today should yield returns long before a comparable solution is commercially available. ■

References

1. H Simon, *THE SCIENCES OF THE ARTIFICIAL*, 2ND ED., MIT Press, Cambridge, MA, 1981.

Software Reuse



Before ResourceCenter.

Until now, trying to implement a software reuse program has felt like one long winding road to nowhere, right? Fortunately, all that has just changed.

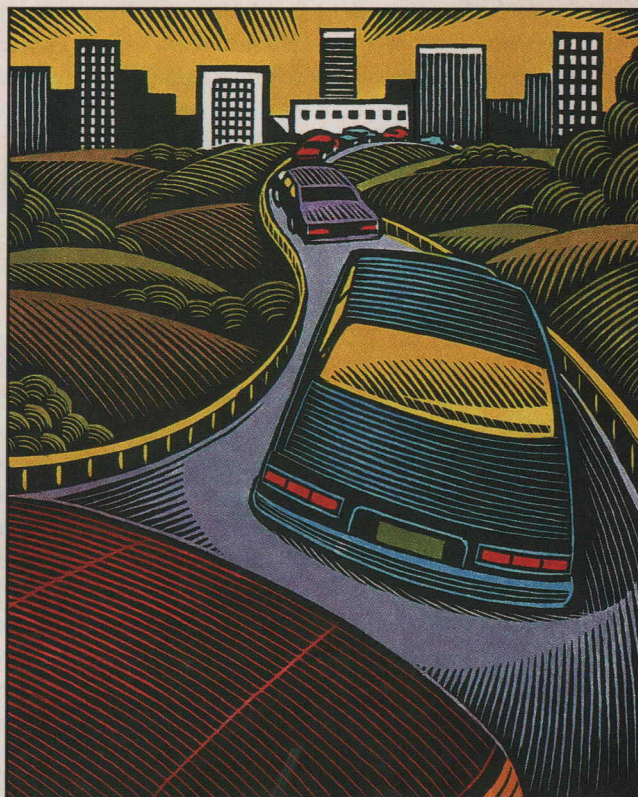
Thanks to new **ResourceCenter™** from CenterLine® Software, taking advantage of software reuse — the ultimate promise of OOP — is finally practical. With **ResourceCenter**, UNIX® programmers can now easily locate and retrieve your reusable software assets. And, since there are no tedious startup, ongoing maintenance, or classification scheme hassles, **ResourceCenter** will deliver results you can see immediately.



CENTER LINE]™

Delivering on the promise of software reuse.

1-800-NOW-CNTR



After ResourceCenter.

In addition, **ResourceCenter's** client-server architecture ensures sharing, scalability and performance. It also comes with built-in descriptions of leading class libraries, and a powerful lexicon-assisted searching technology. All of which means you can finally realize the full potential of reuse: shorter development cycles and higher quality applications.

So, if you're looking for the most practical route to implementing software reuse, it's not only here, it's a lot less complicated than you thought. For a free Software Reuse Implementation Kit, call us today at 1-800-NOW-CNTR.

The challenge of change



Ted Shelton

FOR A DECADE, cultural commentators like Alvin Toffler, Tom Peters, and Peter Drucker have been predicting major changes in the way individuals, companies, and countries compete for resources and markets. In the past few years we have all become front row observers to the real effects of these changes. Even the conservative business people now admit the ongoing shift in the way institutions are being organized. This shift will be as important as the agricultural revolution (which made civilization possible,) and the industrial revolution (which brought about mass society). At stake for business is how human beings will work and interact in our post-industrial, post-sustenance economy.

Some call this business shift an *information revolution*, and claim the next phase of this revolution will be oriented around objects. This new "object orientation" is based on how information can be encapsulated into robust and maintainable software objects as dependable and commodified as hardware is today.

OBJECTS AND BUSINESS

For over 20 years object technology (OT) has been primarily an academic pursuit. While some military and business applications were developed with OT, these efforts were largely ignored by the business marketplace. Suddenly, in the past five years there has been a growing interest in objects from the business community. This growth in interest has been matched by a growing maturity of object-oriented tools; the efforts of a few companies shipping object development environments such as NeXT and ParcPlace, and the

efforts of the IBM-Apple joint venture Taligent and Microsoft's Cairo environment.

Object technology is critical for business in two ways. First, objects greatly reduce the lifecycle in software development projects. Second, and more importantly, objects are capable of modeling business activities. Taken together, objects offer a practical means to revise software in rapidly changing business environments.

To understand why objects are so important, consider the fact that software has not yet reached the reliability and cost effectiveness of hardware. Moore's law, formulated by one of the founders of Intel, predicted that computer hardware technology would follow an exponential improvement curve in such areas as disk capacity, memory, and processor speed. Hardware performance has indeed doubled every two or so years. But software has not grown so quickly in friendliness or functionality.

Software will not reach this threshold with the old methodology, that is, by simply adding new features on demand for each release. But with objects, business constructs can be modeled and then revised as conditions change. These revisions can be systematically released throughout an enterprise on a weekly basis. Object technology thus offers the only plausible hope of software systems keeping up with fast-breaking changes in work and competition.

The financial industry has been first to grasp the degree to which objects drive the organization of their business. At Swiss Bank, for example, new financial instruments called derivatives are routinely created during the normal course of the trading day, right on the trading floor. All of the underlying elements of derivatives are well

understood by financial engineers before being objectified into the system. Traders are able to distill these "exotic" instruments to the reduced form of potential or actual cash flows. Time to market of new instruments is now the critical driving criteria for success for Swiss Bank, and object technology has proved a reliable answer for them. No other software approach comes close.

Firms like Swiss Bank are so successful because they understand which elements of their business domain are fundamental, and how these elements should be objectified. But how do objects affect the task work of the human beings themselves?

AUTOMATED VS INFORMED

Shoshana Zuboff, in her 1988 book *IN THE AGE OF THE SMART MACHINE*, described two paths that technology follows in transforming business activities. In the first of these paths, technology can automate and "replace the human body with a technology that enables the same processes to be performed with more continuity and control."

In the second, technology "... provides a deeper level of transparency to activities that had been either partially or completely opaque." Here information technology supersedes the traditional logic of automation. Shoshana coins a new verb for the second transformation: "informate."

To informate a task is to increase the discretionary activity of workers by increasing their power of decision over tangible information, while freeing them from the drudgery of needing to process such information.

Object technology today delivers rapid prototyping, shorter and more efficient product lifecycles, and low upkeep, which frees the discretionary powers of workers for the emerging changes in work and com-

Ted Shelton is President of IT Solutions, Chicago, IL.

OBJECT TECHNOLOGY TRAINING IS A MOVING TARGET. KSC WILL MAKE SURE YOU HIT IT.

With software design taking on new dimensions almost daily, you can't afford to make your transition to object technology a hit or miss operation. Choose the training specialists who can help you keep up with the changes as they happen — Knowledge Systems Corporation.

Knowledge Systems Corporation offers the most comprehensive classroom education program available, including Smalltalk courses for ParcPlace, IBM and Digitalk dialects. Our materials are based on over eight years experience in applying object technology. All KSC instructors undergo a rigorous certification process and have practical experience in real-world application development.

KSC CLASSROOM

EDUCATION COURSES

- *Manager's Introduction to Object-Oriented Technology*
- *Object-Oriented Analysis and Design*
- *Introduction to Object-Oriented Programming in Smalltalk/V*
- *Introduction to Object-Oriented Programming in VisualWorks*
- *Introduction to Object-Oriented Programming in IBM Smalltalk*

- *Advanced Programming in Smalltalk/V*
- *Advanced Programming in IBM Smalltalk*
- *Bridge Course from Smalltalk/V to VisualWorks*

All KSC courses are offered at the client site or our new training

center in Cary, North Carolina. Courses are structured so that students spend more than half their classroom time in hands-on programming activities. To maximize actual programming time, all classes have a ratio of one student per machine.

KSC has a proven history with companies such as American Management Systems, GE Capital Corporation, IBM, Northern Telecom, The Prudential, Southern California Edison, and Sprint. Our classroom education, apprentice programs and development services offer our customers a total solution to real business problems.

To put your object technology

transition on target, call the Smalltalk training specialists at Knowledge Systems Corporation, 919-481-4000. Or email: salesinfo@ksccary.com. 4001 Weston Parkway, Cary, North Carolina 27513.



KNOWLEDGE SYSTEMS CORPORATION

919-481-4000

petition. The power to combine and recombine information as well as to store, broadcast, and retrieve information is key to this new informed work.

The bottom line: Objects handle the process-intensive activities of the business—so that human beings can freely handle discretion-intensive activities. In the Swiss Bank example, the traders are involved with such discretion-intensive activities as deciding what instruments will be created, how they will be priced, and when they will be traded. Other process-intensive activities, such as derivatives valuation, can be handled by the objects themselves, rather than by the traders.

Or consider the world of customer service, where objects handle the complex process of approving automobile loan applications. Chrysler Financial decided that they needed a far more flexible technology for delivering loan approval procedures to their 85 loan approval centers in the US. With objects, Chrysler accurately modeled the business process for loan approval, and distributed this system nationwide. On a day-to-day basis, they can quickly modify objects as procedures change. Customer service representatives are no longer required to actually process the loans, but are liberated to assist the customers with their special needs.

These examples are anecdotal, but there are parallel examples all around us. Automation has indeed eliminated thousands of unneeded and tedious processing jobs from corporate America at a time when we are said to be in an economic upturn. And yet independent contractors with discretionary powers are filling every nook and cranny of "lean" companies. We are moving from a 1040A job force to a 1099 miscellaneous.

OT ANSWERS THE CHALLENGE OF CHANGE

Information technology managers do not often see themselves as revolutionaries. But we are now thrust into the center of the winds of change. Tom Peters in his latest book, *A TOM PETERS SEMINAR* writes that "... we are in the midst of a once every two centuries cataclysm." Peters suggests that we have arrived at a point where the ways in which we have organized ourselves in the past will no longer help us achieve our goals. A new model for orga-

nizing economic activity must emerge to respond to the new conditions that have caused our old models to become obsolete. Understanding these economic trends is crucial if we are to understand our jobs as technologists.

It is exactly because we don't understand the ways in which our industries, the nature of work, and the organization of our economy are changing that we need object technology. We as managers need tools at our disposal that allow us to model our enterprise and the way in which our enterprise relates to our industry and the larger economy. We need those tools to be sufficiently flexible to allow for rapid and complete changes in our models, so that our software can change as rapidly as our markets change.

OBJECTS IN BUSINESS COLUMN

In upcoming columns I will be focusing on the business considerations of objects—why object technology is important to managers, the issues and difficulties of introducing object technology to managers, making OT succeed in corporations, using OOA&D to model the enterprise, and understanding how objects are being used in firms around the world to transform the way they interact with vendors, customers, and competitors in our global economy.

As technology managers we need to understand the ways in which software development will have to shift. No more monolithic systems: everything must be broken down into component interactive parts. Every system must develop hand-in-hand with the needs of the customer or constituent user. Constant improvement in every modular component must be an ongoing assumption of software development—shifting project teams into object teams that remain dedicated to a particular line of business development over time.

These changes and methods are ones that we will all be experiencing in our daily lives of managing information technology in companies large and small. My mission is to provide you with a few grains of knowledge, refined information, and tangible insight, so to help you manage objects in your business more easily. I look forward to writing on these topics in the months and years ahead. ■

HOW TO CONTACT SIGS PUBLICATIONS

To submit materials for publication

Article proposals, outlines, and manuscripts; industry news; press briefings; letters to the editor—send to
Marie Lenzi, Editor, OBJECT MAGAZINE,
236 West 26th Street, Suite 7SW
New York, NY 10010
Phone: 212.229.0320 Fax: 212.807.9111
email: 70754.11112compuserve.com

Product Announcements

Jane Grau
218 Garden Street
Hoboken, NJ 07030
Fax: 201.963.2059

To order a subscription, renew, or change the address of an existing subscription

In the US—
SIGS Publications
P.O. Box 2027
Langhorne, PA 19047
Phone: 215.785.5996 Fax: 215.785.6073
email: p00976@psilink.com

In Europe—
SIGS Publications
Brocus House, Parkgate Rd.
Newdygate, Surrey RH5 5AH, UK
Phone: 44.0306.631.331 Fax: 44.0306.631.696

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
1111 Plaza Dr.
Schaumburg, IL 60173
Phone: 708.619.9800 Fax: 708.619.0149

For information on reprints of published material, contact:

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:

Shirley Sax, 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

For Writing Object-Oriented Applications To Access Relational Databases, Here's One Class Library.

DBtools.h++ is portable across databases and operating systems. In fact, it supports simultaneous connections to multiple data sources. Now you can face new platforms and databases with confidence.

Choose dynamic linking, and you can substitute other RDBMSs without rewriting, recompiling or relinking. Also, you can deploy your application as a DLL or shared library.

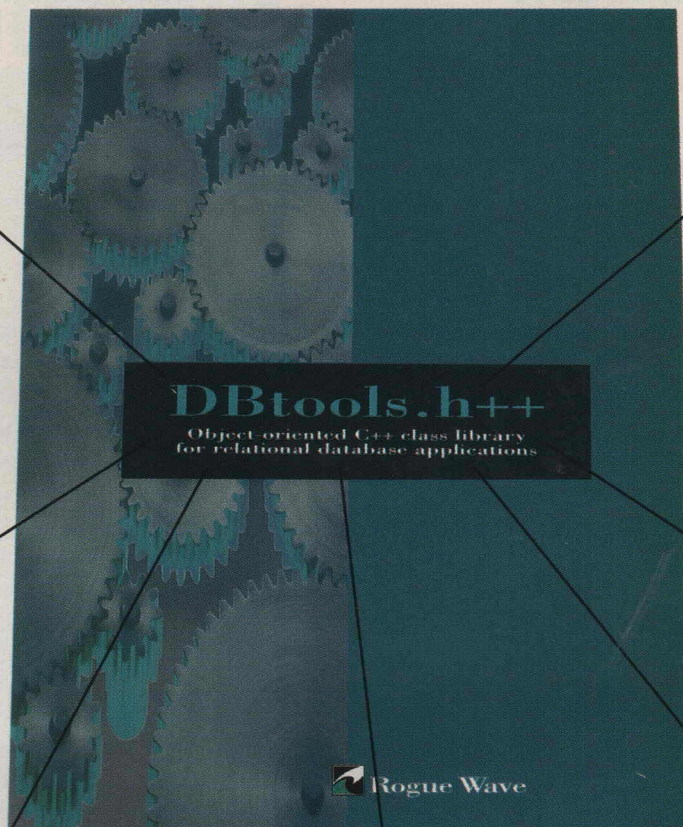
Includes Tools.h++, the de facto standard among foundation libraries. Here's everything you need for fast, safe OO programming.

The elegant design of this C++ class library encourages smart OO programming.

DBtools.h++ encapsulates all the objects and operations you need. Plus it supports stored procedures and accommodates triggers. You get real objects, not just a wrapper.

Shipping now for ORACLE7, Sybase and Microsoft SQL Servers, and ODBC drivers.

Besides being fully compliant with ANSI SQL 92 DML, the encapsulated SQL operations are easily extensible.

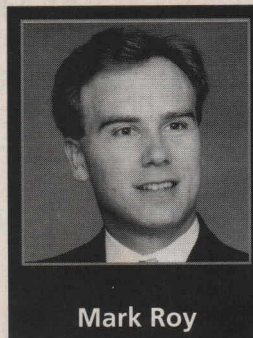


From its intuitive interface to its advanced, flexible error handling, DBtools.h++ provides everything you need to construct robust object-oriented applications for relational databases. Even so, prices start at just \$395 for Windows and \$495 for Unix. To put some class in your database applications, contact Rogue Wave Software today at (800) 487-3217 or (503) 754-3010, or send e-mail to dbtools@roguewave.com. In Germany, call PSItools at 49-6021-366-602.

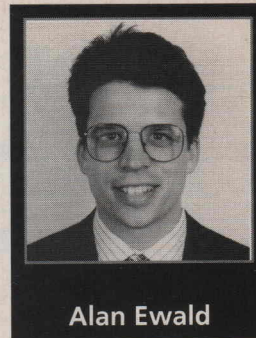


Rogue Wave

Building reliable ORB applications



Mark Roy



Alan Ewald

A difficult problem facing developers of distributed applications is the reliability of communication between clients and servers. There are environments where server availability is of paramount importance. If a service is not accessible, the latest stock market or process control or battlefield information may not be available—with potentially disastrous results. In non-distributed applications, reliability is often achieved by replication of hardware and/or software to guarantee availability in the face of partial failure.

A developer attempting to build a fault-tolerant ORB application today will have some difficult problems to solve. The CORBA specification does not mandate fault-tolerant features in a compliant ORB implementation. As with many aspects of CORBA, individual ORB vendors can provide different levels of service in their products. For the most part, client programs using an ORB have to be prepared to handle failures such as lost communication links, etc.

At the recent Object World conference in San Francisco, IONA Technologies Ltd and Isis Distributed Systems Inc presented a cooperative effort to layer IONA's ORB product, Orbix, on top of Isis's fault-tolerant programming environment. The result is an ORB that is much more resilient to failure conditions in a distributed environment. Meanwhile, IBM's ORB, SOMobjects, includes a replication framework that addresses some of the same issues as Orbix+Isis. We'll look at these two products in the following sections.

HOW TO HANDLE FAILURES

Figure 1 shows an ORB application with a client program communicating with a server to obtain up-to-date information about a

manufacturing process for an operator's workstation. It is critical for the operator to be able to react to changes in the process in a timely fashion.

Several errors could occur that would limit the availability of information to the client program.

- The communication link between the client and server may fail.
- The server itself may fail.
- The server may be too busy to give adequate response time to the client.

A developer attempting to build a fault-tolerant ORB application today will have some difficult problems to solve.

When any of these events occurs, the performance of the overall application is affected. With all the available ORB products, ignoring the extensions discussed below, the client program has to take steps to recover. If the ORB supports dynamic binding, the client may attempt to rebind to another server for the same object or objects. If not, the client may also attempt to find another object or objects that can satisfy the requested operations and are managed by a different server.

If the server is simply too busy to respond, the client could use deferred synchronous invocations or threads to avoid being blocked waiting for a request to com-

plete. This doesn't address the need for timely information, however. It simply allows the client program to continue working while waiting for a response.

The difficulty in recovering from a failure is determining the proper point to restart operations that were in progress prior to the error. If the client and server were in the middle of a series of operations when the server failed, then the client must be able to roll-back to some previous point in its execution when rebinding to a new server. This may not be feasible if the client was performing other related operations at the same time. For example, the client may have been displaying information to a user or communicating with other servers that have not failed.

Another possibility is to write the server program to be able to recover from a failure using transaction management. The server could potentially recreate its internal state at the time of failure (or at some checkpoint prior to the failure). Even so, resynchronizing with other clients or servers that were communicating with the server when it failed may be a difficult programming task.

Rather than attempt to recover, the client program could notify the user that a failure has occurred and involve them in determining the best course of action. However, if the client program is not meant to be interactive then this option is not feasible either.

ORBIX + ISIS

The Orbix + Isis product addresses the potential failure of servers by extending the ORB platform with fault-tolerant features such as replicated servers and data. Client programs written for Orbix + Isis are exactly the same as those written for standard Orbix. On the server side, however, the differences of Orbix + Isis become apparent.

A Clear Development Path Is Obvious From The Right Perspective.



Business managers demand one set of requirements, users want another, and the industry offers countless, often conflicting suggestions. Small wonder it's getting hard to find the right development path.

At Select Software Tools we stepped back from the problems and listened carefully to our users. They told us all about the growing pressure to deliver tangible business benefits faster than ever before, the need to integrate new systems with existing ones, while preparing for new technology just around the corner. They explained their need to scale-up existing department applications for enterprise-wide use. Our users realize that it doesn't always make sense to expect a single method or technology to deliver all the answers. On the other hand you can't ignore dominant industry trends. Client-server architecture offers the best flexibility for distributing applications

THE SELECT PERSPECTIVE

**A clear direction with
methods, techniques and
tools for large-scale
software development.**



over a network. Object Oriented and RAD techniques appear to hold the key to greater productivity. Yet these advances do not offer development solutions in their own right.

That's why Select identified a development solution that brings together these ideas in a clear path based on sound principles and tried and tested techniques.

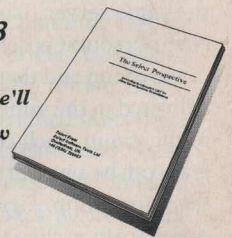
The result is the Select Perspective - a practical open approach to software development incorporating methods, tools, technologies and training.

Select pioneered affordable and accessible CASE tools. We know where CASE has succeeded, and we know its shortcomings. The Select Perspective goes further, to provide a path that leads the way to effective large-scale software development.

Call us today and we'll send you more information

1-800-577-6633

The very least we'll show you is a new and different point of view.



Microsoft
SOLUTION PROVIDER

Circle 15 on Reader Service Card

SELECT
Software Tools

Select Software Tools Inc, 1526 Brookhollow Dr, Suite 84, Santa Ana, CA 92705

Tel: (714) 957-6633 Fax: (714) 957-6219

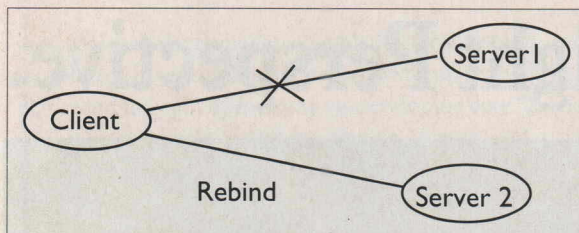


Figure 1. Communication failure.

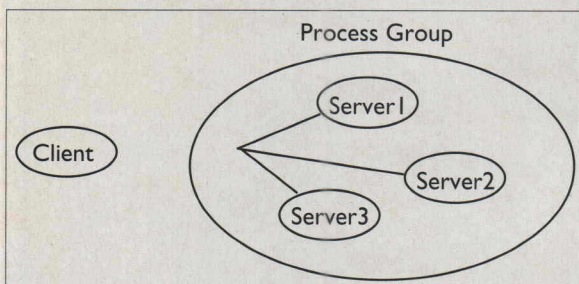


Figure 2. Orbix + Isis.

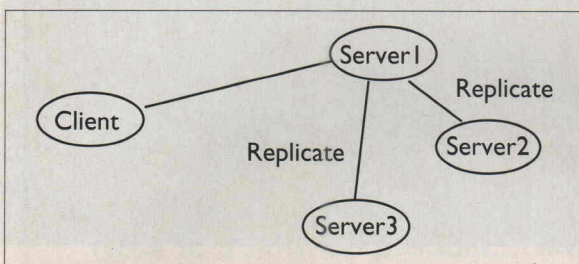


Figure 3. SOMObjects replication framework.

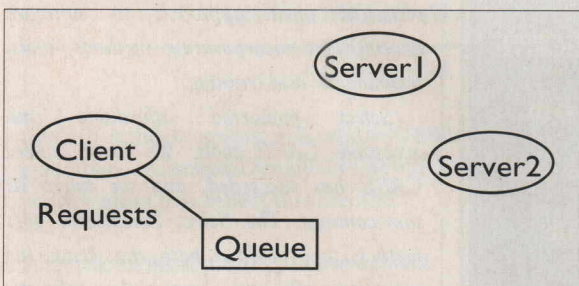


Figure 4. Queueing requests for future delivery.

The Isis programming environment introduces the concept of process groups that can contain redundant running copies of the same server. Isis guarantees that any client's request is delivered to all members of the group and that multiple requests are delivered in the same order to all members of the group. Each copy of the server program can be aware of the other copies within the same group. Figure 2 shows how a request from a client is dispatched to the members of a process group.

When a client's request is delivered to each server copy, the server program can be written to allocate the request to a par-

ticular copy with all of the others standing by in case the chosen copy fails. Alternatively, all the servers may execute the request simultaneously with the client accepting the first result returned. Finally, the client's request may be split up into several pieces that are each handled by a different server with the results combined to be returned to the client.

Writing a server program that can determine how to schedule and dispatch a request to multiple copies will be more complex than writing a standard Orbix server. That is one of the trade-offs for increased reliability. To simplify the task, Isis handles replication of data across all servers in a process group.

Orbix + Isis has the potential to greatly increase reliability in a manner that is seamless to a client program. If one server in a group fails, another can pick up where it left off before responding to the client. A developer could also implement load balancing by writing the server program to selectively assign a request to a free server copy.

THE SOMObjects REPLICATION FRAMEWORK

Like Orbix + Isis, the SOMObjects product from IBM includes a framework that provides replication of data between servers managing the same object(s). When one server makes a change to an object's state, the framework will automatically notify all other servers of the same object that its state has changed. A change can be "logged" by sending a new copy of the object's state to all relevant servers. The framework can also tell each server in a group to perform the requested operation thereby updating each copy of the replicated object. Figure 3 shows a client talking to a server that is managing a replicated object.

The SOMObjects replication framework allows a client program to respond to a server failure by binding to another server that manages the same object. Because the object's state is replicated, the client should be able to continue its operation talking to the new server. What the replication framework doesn't handle is transparent failover. The decision about how to recover is still up to the client program. In the Orbix + Isis case, the failover can be handled by the ORB and the server program without the client program being involved.

AN ISOLATED CLIENT

One error that could still occur with either Orbix + Isis or SOMObjects is communication failure from a client to all available servers in a process group. In such a case, the client machine is most likely isolated on the network and requires a different approach to recovery.

How could the ORB respond when communication is cut off? One solution provided by some messaging products is to queue outgoing requests from a client while waiting for a server to become available. Figure 4 shows a client program that is making requests that will be delivered at some point in the future when the network connection to the server is restored. To queue requests for future execution by an ORB, the client program would have to make each request using one-way or deferred synchronous invocations or with multiple threads.

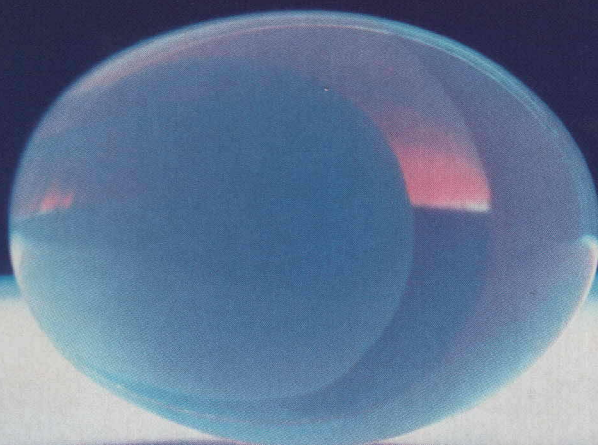
It will be interesting to see if any vendors decide to build an ORB layered on a messaging platform that provides persistent queuing of requests to address this problem. It would be particularly useful for distributed applications where clients need to send one-way requests (or notifications) to a server with guaranteed delivery.

SUMMARY

Orbix + Isis and the SOMObjects Replication Framework each address aspects of developing reliable ORB applications. With these tools, developers can rely on built-in functionality rather than hand-crafting algorithms for replication and redundancy. Look for other vendors to experiment with and deliver fault-tolerant ORB products. ■

Discover the True Power of Objects with ObjectIQ Series™

■ When it comes to developing knowledge-intensive applications, most programming environments simply don't cut the mustard. That's why Hitachi developed the ObjectIQ series, the world's most powerful set of object-oriented application development tools. ■ The ObjectIQ series includes the ObjectIQ™ software development system, the ObjectReuser™ object reuse librarian, and the ObjectIQ-DF™ distributed applications development tools. Each redefines the state-of-the-art in object-oriented application development.



ObjectIQ

Think Objectively ObjectIQ's application development language provides all the features you need. Messages and methods. Multiple class inheritance. Slots with default values, allowable values, and when-asked and when-changed demons. The application language also includes a full set of procedural constructs.

Play by the Rules ObjectIQ infers solutions to problems from rules and facts that you specify. The result is dramatically shortened development time.

Multiple Platforms ObjectIQ lets you develop and deliver applications on Sun SPARC™, HP 9000™, and IBM RS-6000™ workstations. You can also deliver applications on low-cost PCs running Microsoft Windows™.

Efficient Delivery ObjectIQ compiles source code to the language of the target system for the fastest possible execution. You can interpret or compile applications

incrementally to speed changes during development.

Team Player ObjectIQ's C/C++ interface lets you embed ObjectIQ applications in other applications (or vice-versa) preserving your software investment. The DBMS interfaces let your applications access information stored in industry-standard and custom databases.

Powerful Development Tools ObjectIQ comes with a complete suite of application development tools, including a visual Motif GUI editor, tailor, translator, browser/debugger, a foundation class library with more than 400 methods, and an on-line help system.

Field-Proven For developing knowledge-intensive applications, ObjectIQ is the tool of choice in Europe and Japan, where mission-critical ObjectIQ applications are running on more than 5000 workstations in major companies.

ObjectIQ-DF

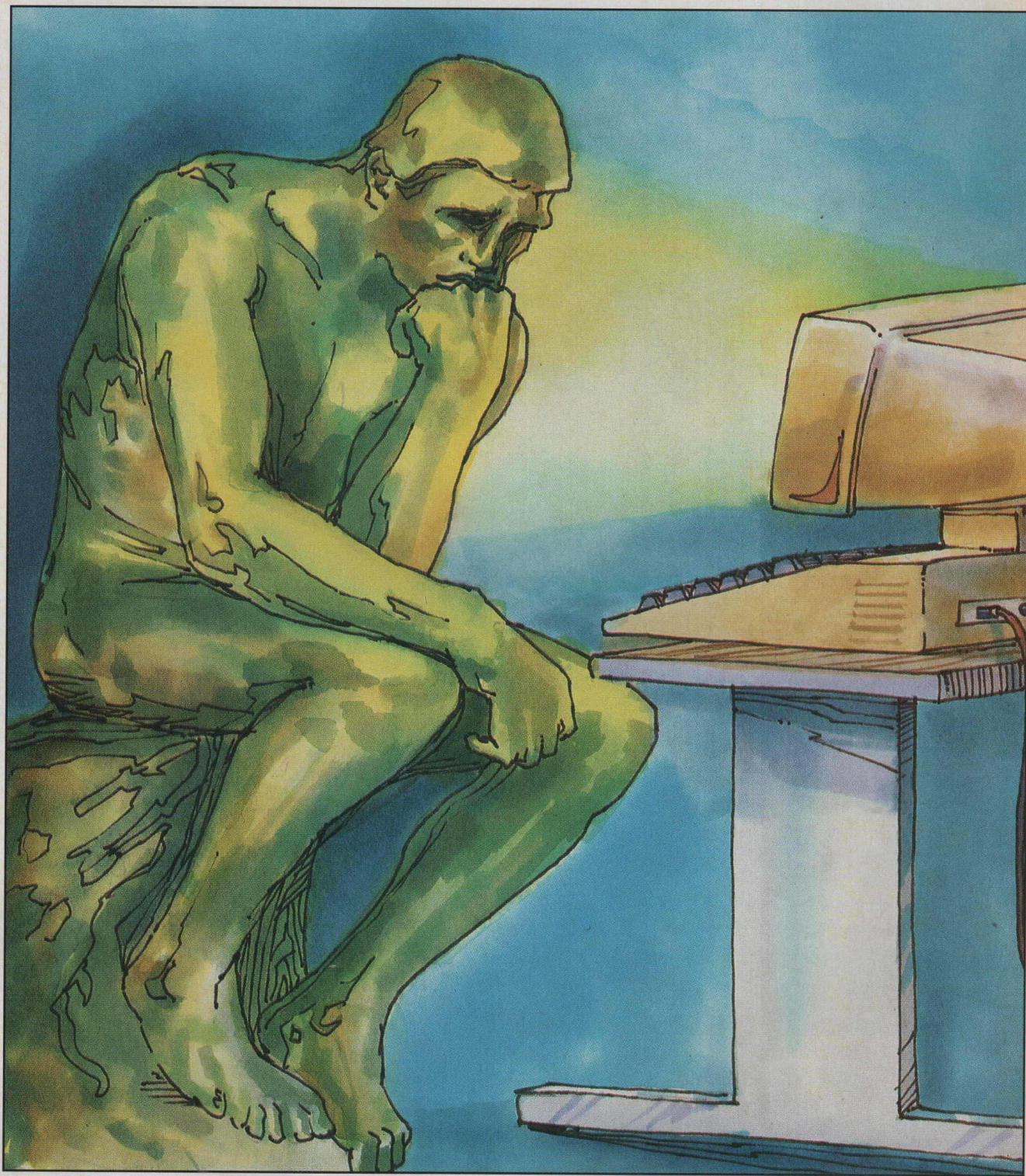
ObjectIQ-DF extends the power of object-oriented programming to heterogeneous networks. Since it handles the details of interaction between remote and local objects, ObjectIQ-DF eliminates the need for low-level network programming, speeding development of client-server and other distributed processing applications.

ObjectReuser

ObjectReuser is the first object reuse librarian specifically designed to handle C/C++ and ObjectIQ code. It facilitates software reuse by letting developers quickly locate and retrieve applicable C/C++ and ObjectIQ objects.

■ To discover the real power of ObjectIQ series, call (212) 751-6302.

HITACHI



Andrew Shiftsis

MODELLING RULES in object-oriented analysis

A rule is the codification of an organisation's policy or condition that must be satisfied in order for the organisation to function.

Sanjiv Gossain

OBJECT MODELLING IS increasingly being used to model an organisation's business. With this increase comes a need to model the rules of an organisation that place constraints upon the way that business functions. These rules are essential aspects of modelling the business because they reflect the rules and policies of that organisation.

We offer the following loose definition for a rule based on Odell¹:

A rule is the codification of an organisation's policy or condition that must be satisfied in order for the organisation to function.

Rules are therefore essential to the requirements of modelling a problem domain.

A rule can be modelled in a number of different ways. Simple rules, for example, may be captured as:

- **A relationship on the object model.** For example, "Landlord leases an apartment to a Tenant" is a rule that can modelled by a relationship between Landlord and Tenant.
- **A state model of an object.** For example, "the tank must be emptied before it is filled" can be modelled through the state model of a tank.

However, not all rules can be adequately captured in this way. Some are combinations of simpler rules that span objects, relationships, and processes, and others are reflected as constraints upon objects and relationships. Rules are often identified in the

latter part of analysis, after the objects have already been identified and the relationships formalised. It is therefore usually not possible to incorporate these rules into an existing model merely by adding an object or a relationship. Changing the model may also result in the exclusion of some rules. Some other ways to think about rules are required.

In this article we will look at a few other techniques in modelling rules that afford the object developer some advantages during both analysis and design.

In the next section we describe how to model rules that place constraints across objects and relationships, followed by an examination of how best to capture rules that affect attribute values of an object. The notation used herein is that of Shlaer-Mellor OOA,² although these techniques can be applied to any flavour of object-oriented analysis. As an example, a couple of diagrams have also been shown using the notation of OMT.³

RULES FOR OBJECTS AND RELATIONSHIPS

To illustrate the type of rule that affects objects and their relationships, consider the object information model extract of Figure 1. The model represents a possible example at a university or college where students major in a particular subject and must choose a professor as their adviser. It also shows that each professor must belong to one department.

Simple rules

In this triangle of relationships, however, there is a rule that must be adhered to:

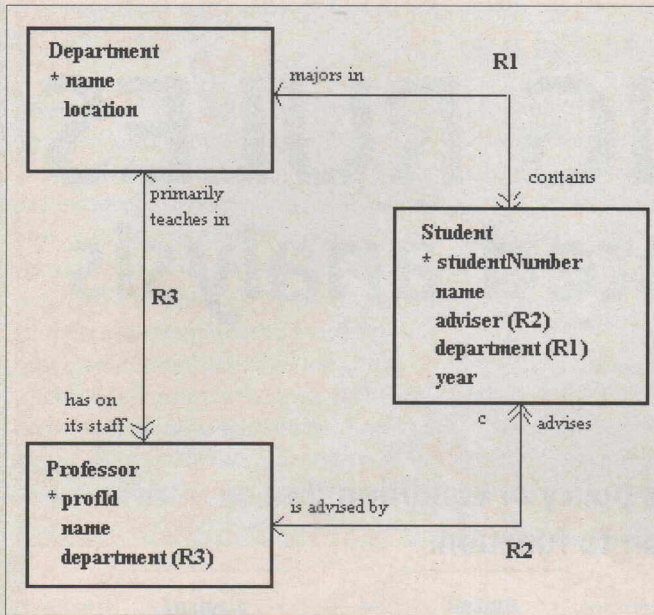


Figure 1. Object information model extract.

A student must be advised by a professor from the same department that he/she is majoring in.

For example, if a student is majoring in Physics, then that student's adviser must be a professor in the Physics department.

This rule is a key element of our understanding of the domain and should be captured in our analysis in some way. Figure 1, as it stands, doesn't indicate this rule in any way, and there are apparently no obvious ways in which to model this constraint. Of course, we can always add some natural language description to our model to capture the constraint (see Fig 2).⁴ However, as the model grows and the number of rules grow, the adding of natural descriptions will soon make the model cluttered and unreadable. Such a scheme for describing rules is limiting.

An alternative representation is to indicate that the relationship R2 is constrained, as shown in Figure 3. The relationship line is marked as constrained and the attribute adviser is also marked as being constrained by the label (R2c). This constraint can be documented in the description of relationship R2, stored as part of the data dictionary.* For example:

Relationship: R2—Student is advised by Professor; Professor advises Student (1:M)

Description: Each student must be advised by one professor. The student may choose which professor they wish to have as their adviser. However, this relationship is constrained.

Rule: (2) The college policy states that a student can only be advised by a professor from the department that the student is majoring in.

This annotation of the object model indicates that there is a constraint upon the relationship, but does not indicate how this constraint is met. Ensuring the rule is enforced can be modelled as part of the analysis of dynamic behaviour. If a state model is being used

to represent the dynamic behaviour of an object then the rule execution can be codified into the state model of one of the objects participating in the relationship. As a Student is responsible for selecting a Professor, it would seem more suitable to do this in the state model of the Student object, as shown in the partial state model of Figure 4.

We can see from the state model that the logic associated with the rule is in state 2 of Student. The trigger for this rule firing is either the event S2:selectAdviser or S4: changeAdviser.

When we move to implementation the rule would be enforced by invocation of the member function Student::selectAdviser or Student::changeAdviser,[†] thus causing the block of code that carries out the rule to be executed.

Complex Rules

Let us now consider a more complex rule or set of rules. The rules to be modelled now are:

1. A student must be advised by a professor from the same department that he/she is majoring in.
2. No professor can be adviser to more than five students
3. No professor can be adviser to the same student for more than one year.

Our set of rules here are more complex, and they also apply to more than just one Student-Professor pair whereas in the previous section the rule applied only to one instance of the relationship. There is also an issue of contention here: how do we cope with the situation where more than five students wish to be advised by the same professor? If the rule execution is in the instance of Student, it is possible to have more than five students selecting the same adviser. It is no longer possible to have the Student object responsible for managing its own choice of adviser, and so the firing of rules must be abstracted elsewhere.

To deal with this situation, we could encapsulate these rules within an object, in this case the Department object. The Department object will encapsulate the policy and decision-making logic responsible for ensuring that the above rules for the assignment of students to advisers are met in that particular department. The Department's role in this situation is very similar to the *assigner* concept described in Shlaer-Mellor,² where an Assigner object is responsible for handling contention issues in the problem domain.

The rules that the object Department captures can form part of the description of the object and the object may even abstract attributes such as maxNumStudents and maxAdvisingPeriod that may be used in enforcing the rules. Our object model only changes by the addition of some attributes to Department.

Let us now suppose that the rules for assigning students to advisers are the same across all departments. Now, the rules should be fired by another object, say a University object. University will be responsible for assigning students to advisers and will contain any attributes necessary for this task. Our object model now becomes that in Figure 5(a), with Figure 5(b) representing the equiv-

* A object model is not complete unless textual descriptions of objects, attributes, and relationships have been documented.

[†] This is assuming that the object is implemented as a class Student. This is certainly not the only implementation; it is merely the most direct.

alent in OMT notation.

Figure 6 depicts a sample object communication scenario showing how the rules may be executed with the University object.

Abstracting the rules into an object in this way means that any changes to the rules will be localised to one object, be it Department or University. So, if next year the rule is that a professor can have at most ten advisees, this change can easily be incorporated into the model. Fowler⁵ has also recognised the importance of incorporating rules as objects in analysis.

RULES FOR ATTRIBUTES

Now consider the situation where we have rules that are not defined across objects and relationships but seem to be more centered around attributes of the same object.

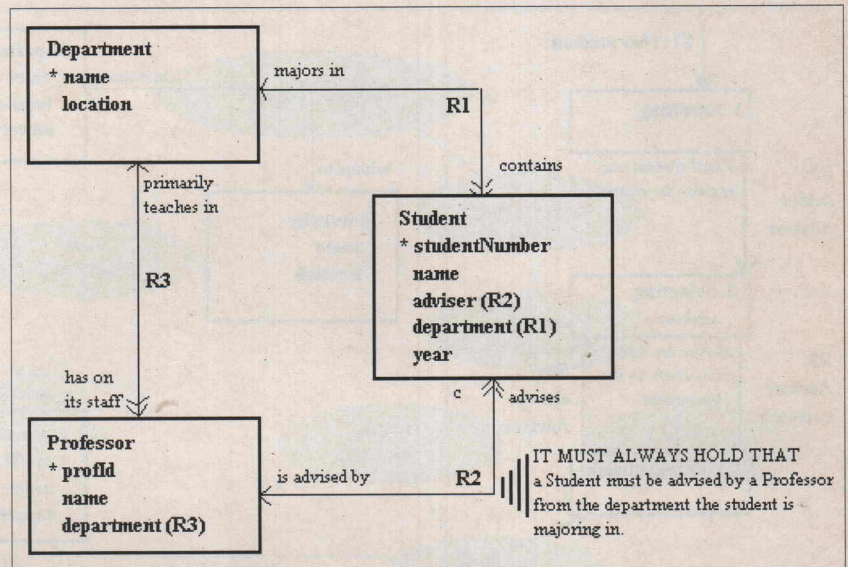


Figure 2. Capturing a constraint with a natural language description.

One rule across all instances

First, consider the situation where one rule applies to attributes across all instances of the same object. For example, consider Figure 7. This object information model extract from a bank shows one SavingsAccountType specifying the policies for many different instances of SavingsAccount. Each instance of SavingsAccount reflects an actual account opened by an account holder (eg, Account number 123456, opened by John Smith, etc.), whereas each instance of SavingsAccountType reflects the type of savings account it conforms to (eg, high interest, easy access, etc.).

The rule we wish to capture here is the dependency between the interestRate of the account type and the noticePeriod (the length of notice an account holder must give before making a withdrawal from the account so as not to lose interest). The rule is:

If interestRate $\leq$ 5% then noticePeriod = 30 days
 If interestRate > 5% and $\leq$ 10% then noticePeriod = 60 days
 If interestRate > 10% then noticePeriod = 90 days

We observe that the rules dictating the value of noticePeriod are constrained by interestRate. One can show this constraint by marking the attribute noticePeriod with a c on the object information model, and by going into more detail in the textual attribute description. The trigger for the execution of this rule is any change in the value of interestRate. Therefore, we also need to augment the description of interestRate to indicate that it is a trigger for the firing of the rule. The example descriptions accompanying the object information model could be:

interestRate: the rate at which interest is accrued on the type of account. Usually calculated monthly. Note that this attribute is a trigger for Rule 2.

Domain of this attribute: a percentage figure.

noticePeriod: the period of notice required to be given by the account holder before making a withdrawal to ensure that no interest is lost. Note that this attribute is constrained by Rule 2.

Domain of this attribute: a whole number of days.

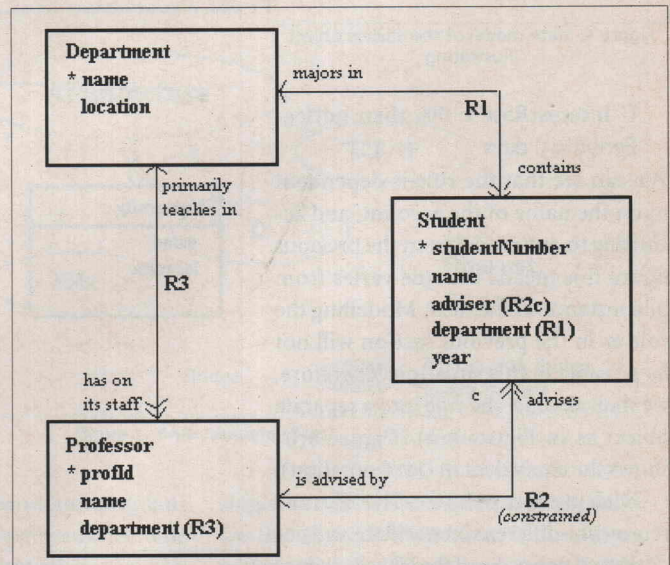


Figure 3. Modelling the example constraint.

This rule can be enforced by the dynamics of the object to which it applies. Because the rule applies to the same attributes across all instances, this is a reasonably straightforward task.

Rules varying by instance

The previous rule applies to attributes across all instances of SavingsAccountType. However, the situation may be a little more complex when the rule can vary from one instance of SavingsAccountType to another depending upon some conditions.

For example, a bank typically has a number of different types of account and there may be different rules according to different types. The "Plus" account type may have the following rule:

If interestRate $\leq$ 5% then noticePeriod = 0 days
 If interestRate > 5% then noticePeriod = 30 days

and the "Easy Access" account type may have the following rule:

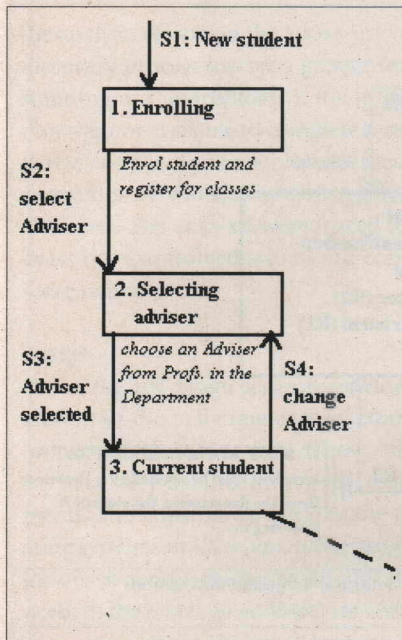


Figure 4. State model of the Student object illustrating

If $\text{interestRate} \geq 0\%$ then notice-Period = 0 days

We can see that the rule is dependent upon the name of the account, and according to our model from the previous figure this means the rule varies from one instance to the next. Modelling the rule as in the previous section will not be possible in this situation. Therefore, we shall abstract the rule into a separate object as in Figure 8(a) (Figure 8(b) shows the equivalent in OMT notation).

Each instance of InterestRateTier will represent a different interestRate and noticePeriod policy. So, if the "Plus" account type has its rule defined as above, and the "Easy Access" and "Instant Access" account types have the same rule, then we shall have three instances of AccountType, and only two instances of InterestRateTier. This is because the same instance of the rule can be involved in multiple relationships with account types. As we can see, the object InterestRateTier is treated in the same way as any other object on the object information model.

As before, the description of SavingsAccountType will indicate which rule the attribute noticePeriod is constrained by, and the attribute interestRate is triggering.

The execution of the rule may be handled by the instances of InterestRateTier. Figure 9 shows a possible object communication scenario showing how a SavingsAccountType checks to ensure its rules are met.

The advantages afforded by encapsulating rules as objects, again, is the localisation of changes due to any modifications of an organisation's policies.

Looking ahead to design and implementation, one possibility

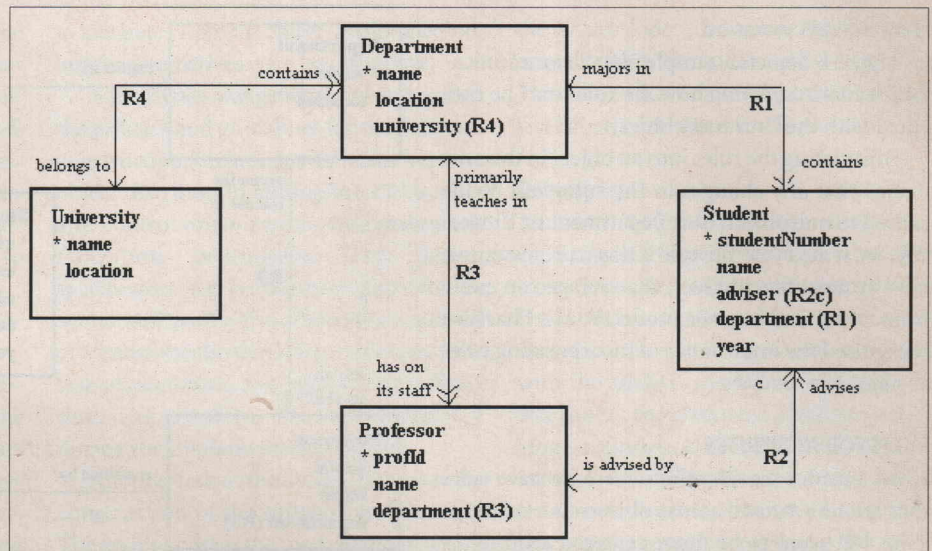


Figure 5(a). University object added to the model (Shlaer-Mellor notation).

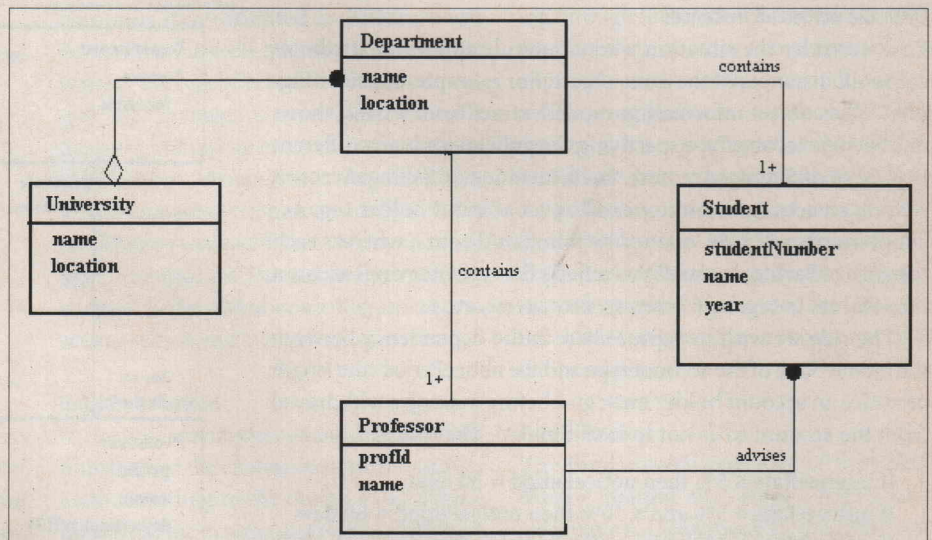


Figure 5(b). University object added to the model (OMT notation).

would be to have each rule implemented as a separate instance of a class, say in C++, with the rule execution being implemented by a member function fireRule(). An attempt to change the value of a triggering attribute in an instance of SavingsAccountType would invoke fireRule of the instance of SavingsAccountRule associated with it, passing it any necessary data (eg, a handle to itself). The InterestRateTier instance then can process the rule using the data it received. This instance of InterestRateTier can execute the same rule for multiple account types.

Following such a scheme segregates the logic of the rule execution from the account type it is operating upon. An organisation's policy is likely to change more frequently than its account types, and so modifications to the rules can be easily incorporated without changing the account types.

SUMMARY

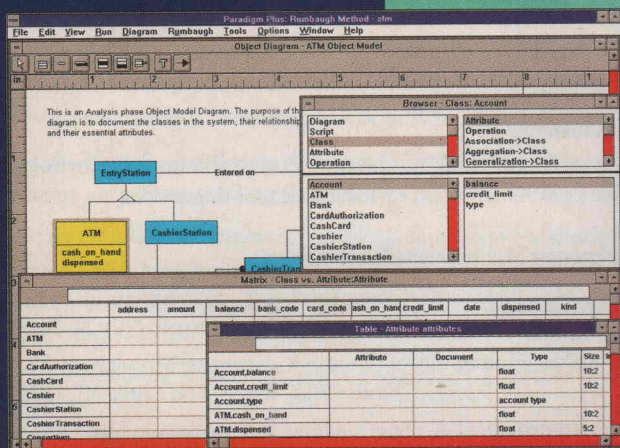
We have shown that business rules can be modelled in object-ori-

Paradigm Plus[®]

NEW
v 2.0

The Award Winning OO-CASE Tool

The Most Advanced,
Complete, and Easiest
to Use OO-CASE Tool



ProtoSoft[®]

17629 El Camino Real, Suite 202
Houston, Texas 77058
(713)480-3233 Fax (713) 480-6606

England 44 276 692269, Germany 49 721 25911, Israel 972 8 404898
Holland 31 4130 50330, Australia 61 2 360 6999, Brasil 55 11 287 5899
South Africa 27 12 4202361, Scandinavian Countries 46 40 70930

- **Methodologies:**
Rumbaugh, Booch, HP Fusion, Coad/Yourdon, Shlaer/Mellor
- **Languages:**
C, C++, Smalltalk, Ada code generation
C/C++ Reverse Engineering
- **Platforms:**
Windows, SunOS, Solaris, HP-UX, AIX
- **Database Systems:**
ObjectStore, Versant, Objectivity, Oracle, Sybase, Informix schema generation
- **Networks:**
Novell, LAN Manager, TCP/IP, NFS
- **Multi-User**

* Object World, January 1994

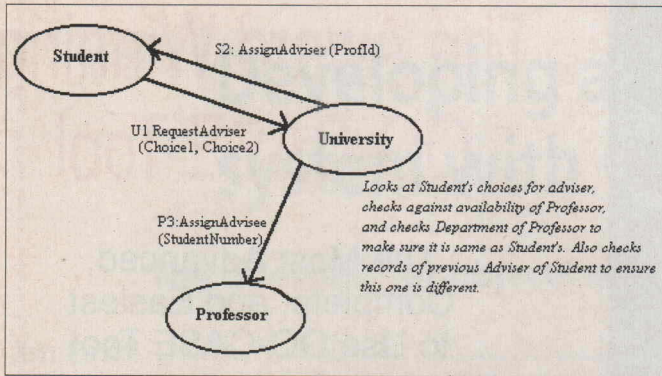


Figure 6. Rules are executed by the University object.

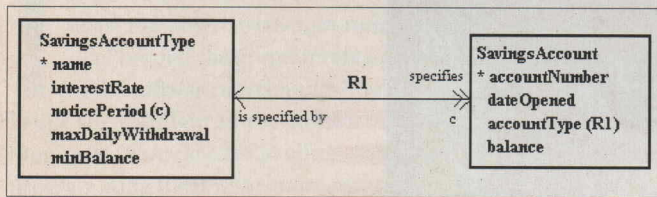


Figure 7. Object information model for SavingsAccountType.

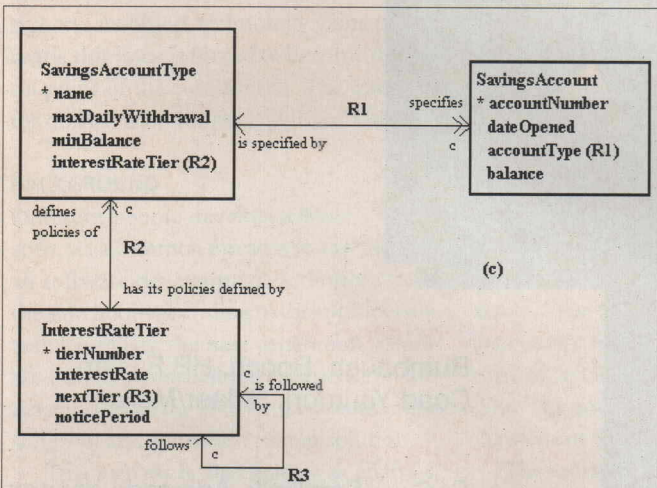


Figure 8(a). Modelling rules varying by instance (Shlaer-Mellor notation).

ented analysis in a number of different ways. The decision of how best to represent a rule—as a constraint on a relationship or attribute, or as an object—varies from case to case. However, we can offer some basic guidelines:

- Simple rules are best expressed as constraints to attributes and can be checked easily at design time.
- Rules that affect multiple attributes in the same object can be contained within that object.
- More complex rules and those varying from instance to instance of the same object can be abstracted as an object outright. Each instance of this object will then be a rule.

Abstracting rules as objects affords the object developer some advantages

- Clearer capture of the semantics of the problem.

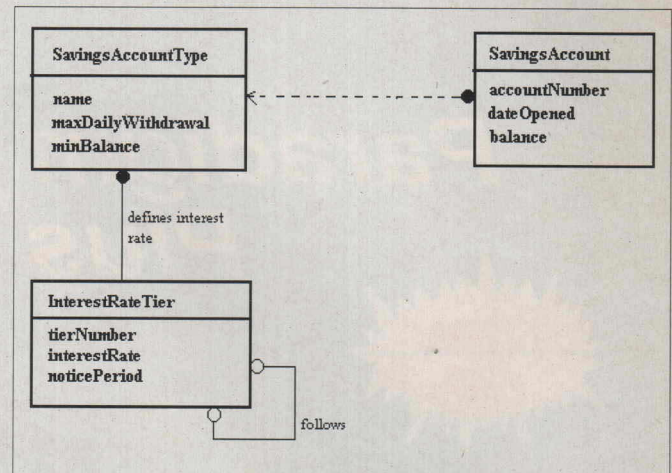


Figure 8(b). Modelling rules varying by instance (OMT notation).

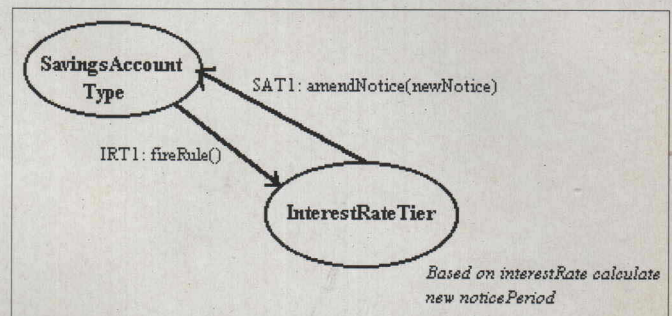


Figure 9. Possible object communication scenario for SavingsAccountType.

- Decoupling of organisational policy from the object(s) it affects.
- Localisation of change.

Acknowledgment

The author thanks Neil Lang of Project Technology for his valuable comments on an earlier version of this article.

References and suggested reading

1. J Odell. Analysis & design: Specifying requirements using rules, JOURNAL OF OBJECT-ORIENTED PROGRAMMING 6(2):20–24, 1993.
2. S Shlaer and SJ Mellor. OBJECT LIFECYCLES: MODELING THE WORLD IN STATES, Prentice Hall, Englewood Cliffs, NJ, 1990.
3. J Rumbaugh et al. OBJECT-ORIENTED MODELING AND DESIGN, Prentice Hall, Englewood Cliffs, NJ, 1991.
4. J Odell. Analysis & design: Using business rules with diagrams, JOURNAL OF OBJECT-ORIENTED PROGRAMMING 6(4):10–17, 1993.
5. M Fowler. Lessons in OO analysis and design from three commercial projects, PROCEEDINGS OF OBJECT DEVELOPMENT EXPERIENCES, UNICOM Seminars, London, UK, May 1994.
6. S Gossain. Practical techniques for object-oriented analysis, PROCEEDINGS OF OBJECT DEVELOPMENT EXPERIENCES, UNICOM Seminars, London, UK, May 1994.

Dr. Sanjiv Gossain is a senior member of the technical staff at Cambridge Technology Partners, UK.

Is a deep class hierarchy considered harmful?

The bottom line is: if your class library cannot be comfortably reused by your colleagues because of the deep hierarchy, then it is probably too deep.

..... Barbara Lin Wang & Jin Wang

Recruiter: Could you describe your experience in object-oriented design and programming?

Interviewee: I have designed a class library and one class hierarchy has twelve levels.

Recruiter: Wow...

Inheritance, a key feature of object technology, provides a basis for code reuse and concept abstraction. However, the class hierarchy created by inheritance could be harmful, in practice, if it is too deep. This article discusses reuse and maintenance difficulties associated with a deep class hierarchy, and provides suggestions on designing software for ease of reuse.

WHEN AND WHY COULD A DEEP HIERARCHY BE HARMFUL?

A major advantage of object technology over conventional ones is its lower cost of software maintenance over the long run and easier extension via component reuse. A deep class hierarchy, however, could discourage reuse, especially in the context of software extension and maintenance.

The complexity of reuse is twofold: one must understand class interfaces as well as the relationships between parent and derived classes. To gain such an understanding, reusers may need to go through the entire hierarchy even when reuse is for extension or maintenance. Because the middle of the class hierarchy normally provides many useful concepts, component reuse usually occurs in that feature-rich part. As a class hierarchy goes deeper, reusable components are further away from the root, and multiple inheritance and polymorphism are more likely to be explored. The subtlety and complexity of the two (inheritance and polymorphism) when combined could grow drastically, making reuse difficult.

Breaking up a large functionality into smaller pieces is considered a promotion of reuse.¹ We argue that understanding class relationships could be much more complex for reusers than understanding class interfaces themselves if the hierarchy is too deep. Fine class granularity promotes flexibility, but it could easily lead to deep hierarchies. For example, a pure object world, where even an integer is modeled as an object, may be theoretically small and beautiful, but the practicality and performance of such a model might be in serious question. On the other hand, an overly coarse class granularity could flatten out problem structures and remove useful relationship information inherent to the problem.

If software is designed with future growth in mind, a deep class hierarchy introduces barriers to extension for the same reasons as

described earlier. Maintaining software products with deep class hierarchies is costly, too. When new features are requested from customers, one can either derive new classes or modify existing intermediate ones. If the modification approach is adopted for some reasons, the tight coupling in the chain of classes involved in the hierarchy makes this approach difficult.

CASE STUDY: WHAT'S WRONG?

The extended relational model RM/T (or semantic-oriented modeling) uses *subtype* and *supertype* concepts to construct the type hierarchy.² To help explain the type hierarchy, Date introduced *is-a* relationships for relational database design.³ These terms (subtype, supertype, and *is-a*) are also commonly used in object training workshops to illustrate the meaning of "subclass" and "superclass." As a result, some people believe there is no difference between the type hierarchy of semantic-oriented modeling and a class hierarchy. They tend to directly convert their type hierarchy to their class hierarchy. This is even more true when companies migrate their applications to object technology, but still want to continue to use RDBMSs. Such imitation may result in a deep class hierarchy for their applications when they deal with a deep type hierarchy (see Fig 1 for an example). There is nothing wrong from the data modeling point of view, but the deep hierarchy so introduced could mean high maintenance costs and difficulty in extension if no redesign/consolidation is carried out.

HOW DEEP IS TOO DEEP?

Unfortunately, there is no absolute answer to this question. While a three-level hierarchy is considered deep,⁴ another example has nine levels.¹ Factors to be considered include level of depths, complexity of multiple inheritances and virtual functions, total functionality of a hierarchy provided, experience of reusers, and development tools. The bottom line is: if your class library cannot be comfortably reused by your colleagues because of the deep hierarchy, then it is probably too deep.

Software needs to be designed with *ease of reuse* as an important objective, and a deep hierarchy should be avoided at least in the initial design. For instance, you should use class containment for *has-a* memberships rather than inheritance, which handles *is-a* relationships.⁵ If your class hierarchy is still too deep due to fine class granularity, you may want to consider reexamining the design and consolidating the hierarchy. In Figure 1, *System_programmer* could be combined with *Network_programmer* during the

sixty six satellites can't be wrong

When Motorola needed a CORBA conformant Object Request Broker to develop software controlling 66 IRIDIUM® system satellites, it chose IONA Technologies' Orbix®.

Orbix® is a lightweight, easy to use, extensible implementation of the Object Management Group's Common Object Request Broker Architecture. With Orbix®, software developers can create robust distributed C++ applications, following a consistent and standards based model.

The development tool is currently available on a wide variety of platforms including Solaris, SunOS, IRIX, HP-UX, OSF/1, UnixWare, Windows 3.1 and Windows NT.

...Objects prefer Orbix®

For further information or an evaluation copy of Orbix contact:
IONA Technologies Ltd., 8-34 Percy Place, Dublin 4, Ireland.
Email: info@iona.ie Tel: +353-1-668 6522 Fax: +353-1-668 6573
In the USA call: 1-800-orbix4u (672 4948)

Orbix® is a registered trademark of IONA Technologies Ltd. IRIDIUM® is a registered trademark and service mark of Iridium, Inc. All other trademarks registered.



Circle 18 on Reader Service Card

design effort of object-oriented migration. A relatively shallow structure not only makes your class library more understandable to potential reusers, but also leaves more room for future extension (eg, by deriving new classes).

As modification requests and new features are incorporated into an existing software product, new classes and patches are derived from the original hierarchy to meet these needs. The hierarchy may gradually get deeper, making its maintenance more costly.

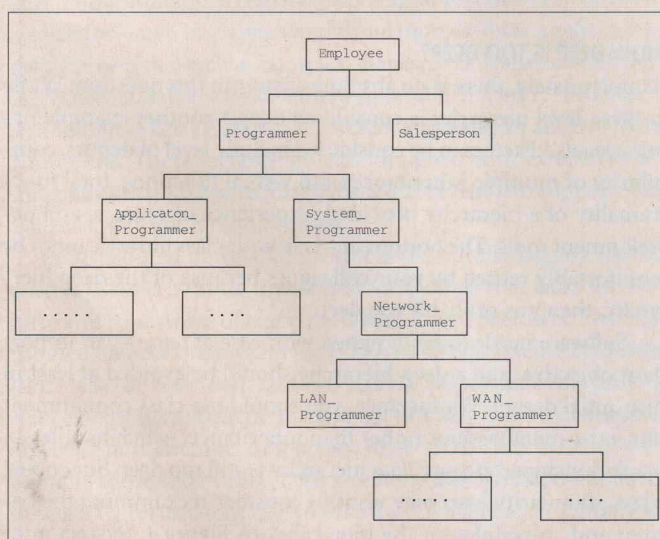


Figure 1. Example of a type hierarchy in the extended relational model.

At some point, it may be well worth the effort to consolidate the hierarchy. This consolidation would also improve ease of reuse.

Class designers should have no incentive to create deep hierarchies. The depth of a hierarchy has nothing to do with the degree of object orientation. What really matters is the ease of reuse and maintenance. ■

Acknowledgments

The authors wish to thank Wes Ague, Dennis Beasley, Youngmee Kim, Bruce Rowland, Ron Watson, Don Benkeser, and Mike Ryder for their encouragement and support in applying object technology to the LifeKeeper Fault Resilient Systems (FRS) development.

References

1. B Kennedy. Design for object-oriented reuse in the OATH library, JOURNAL OF OBJECT-ORIENTED PROGRAMMING 5(4):51-57, 1992.
2. C J Date. AN INTRODUCTION TO DATABASE SYSTEMS, Addison-Wesley, Reading, MA, 1987.
3. C J Date. RELATIONAL DATABASE: SELECTED WRITINGS, Addison-Wesley, Reading, MA, 1988.
4. L Stump. Writing reusable classes: Five top tips on C++ class design, EXE: THE SOFTWARE DEVELOPER'S MAGAZINE 6(9):59-61, 1992.
5. B Stroustrup. THE C++ PROGRAMMING LANGUAGE, 2ND ED., Addison-Wesley, Reading, MA, 1992.

Barbara Lin Wang & Jin Wang (Ph.D.) are software engineers with AT&T Global Information Solutions in Naperville, IL.

DEMYSTIFYING OBJECTS

Uncover the potential of object technology—increased productivity, faster development time, ease of maintenance and reusability of software designs. When it comes to object technology, ICON Computing delivers comprehensive, rigorous training to bring your project team up to speed.

Lay down a supporting framework for the OO paradigm with ICON's managerial and technical overview courses. Solidify it with a strong conceptual foundation of effective modeling methods and practical, proven techniques in our analysis, design and OMT/FUSION courses. Implement the technology using C++ or Smalltalk on a truly object-oriented modeling and design base.

Highlights of our curriculum include:

- OMT with complete integration of models
- The FUSION methodology
- Design Patterns
- Basic through advanced C++ and Smalltalk
- ... and much more!

Come to our open enrollment seminars, or we'll come to you and customize our curriculum to your project needs and domain. Call on ICON's consulting services to support every phase of your project development, both before and after courses.

ICON's extensive curriculum and experienced educators provide a clear, precise path to self-sufficiency and resilient object systems.

Design your training plans with our advisors and meet the object-oriented future today. Call for our brochure and a free copy of our informative article on Object Education and for our technical white-paper selection.



"Of the several courses I have taken on OOA&D, this is definitely the most comprehensive."

Joe Suttles, NCR

•

"Outstanding! Responses to queries were sharp, clear, and concise."

Michael Aden, Siemen's Quantum

•

"An excellent course."

A must for any group that is moving to object-oriented technologies."

Michael Diamond, BNR

ICON COMPUTING, INC.

Quality in Object Education and Consulting

12343 Hymeadow Dr., Suite #3C
Austin, TX 78750
(512) 258-8437 * (512) 258-0086
info@iconcomp.com

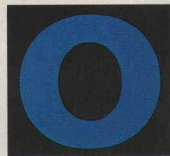




Choosing between OLE/COM and CORBA

An important factor to consider is the availability of the technologies over the lifecycle of the target system.

Thomas J Mowbray



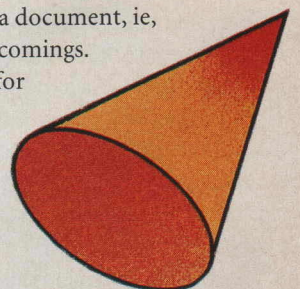
ONE OF THE MOST important technical and business questions of our time is the choice between the proprietary and standard distributed object technologies, OLE/COM and CORBA.¹ OLE/COM is a new technology from DEC and Microsoft, the industry leader in platform volume.² CORBA is a multi-vendor standard technology from one of the world's largest consortia, the Object Management Group (OMG).³ This article will compare and contrast these technologies, and provide some key considerations affecting the choice for corporate developers and system integrators.

OLE/COM

One can most easily characterize OLE/COM as an incremental extension of Microsoft's technology base. It is the first step in the migration to distributed objects for the series of integration technologies that includes DDE, OLE1, OLE2, and OLE/COM (Fig 1).

Object Linking and Embedding (OLE) is primarily a compound document framework. Initially, it was a proprietary integration solution for Microsoft's Office products. Every supplier of integrated office automation software has an analogous framework. Dynamic Data Exchange (DDE) was the precursor technology on Windows and it is still the predominant technology on Windows platforms. Many developers found DDE to be too low level and difficult to use. OLE Version 1 was implemented as a layer over DDE that provided a new capability: the capability to activate an embedded application object within a document, ie, a compound document. But it had many shortcomings. Linking was found to be too complex to use for the average end user. OLE1 linking was also a brittle mechanism; Its links were easily broken by any changes to the file pathname of a linked object.

OLE Version 2 provides some additional user benefits, but adds a great deal more



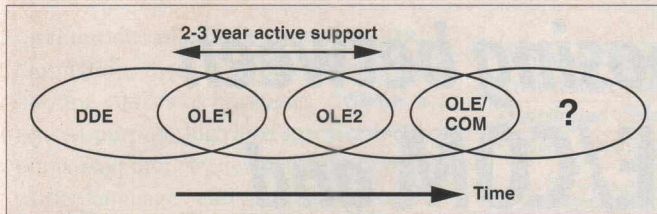


Figure 1. Incremental technology evolution and obsolescence.

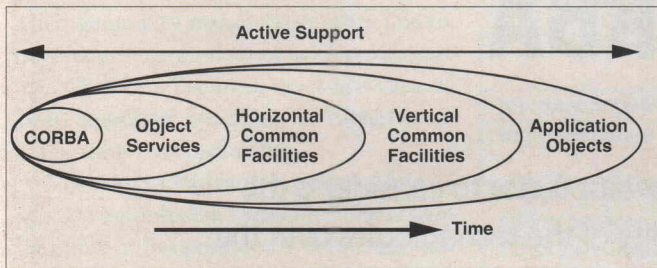


Figure 2. CORBA standards and technology evolution.

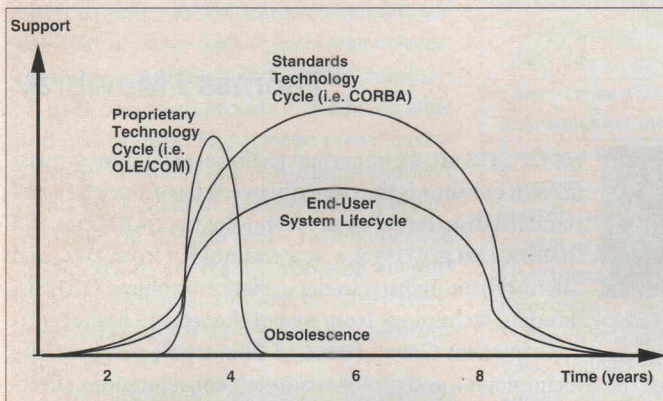


Figure 3. Commercial technology cycles vs end-user system lifecycle.

complexity for the developer. Its primary innovation for the end user is in-place activation. Embedded document objects can be edited in-place through changes to the container's menus and tool bar. In-place activation was found to be readily accepted by end-users. OLE2 does not support distributed objects nor multi-user environments. For example, OLE2 compound documents cannot be moved between machines without breaking most of the file and application linkages. This situation is essentially unchanged by The Common Object Model (COM).

COM is a distributed implementation of a small subset of OLE2. Its primary innovation is a distributed version of OLE2's Component Object Model. The COM specification has been released publicly, but the technology will not be available until late 1995.² The COM subset of OLE2 does not include the OLE2 compound document framework. OLE compound documents will continue to be a non-distributed, single-user technology.

COM contains object-based extensions to the remote procedure calls (RPCs) in the Open Software Foundation's Distributed Computing Environment (DCE). COM can be described as an object-based version of DCE RPC, including more than a dozen new proprietary extensions.

Many DCE extensions appear in the COM Interface Definition

Language (COM IDL), a relatively low-level language based on C. COM IDL is only a convenience; the real COM APIs are defined at the binary level. At this level, COM exposes the binary API of Microsoft's C++ runtime system. Theoretically, the binary API eliminates language bias, but COM's APIs are heavily dependent upon Microsoft's C++ implementation.

The realization of OLE/COM's innovations assumes universal adoption. End users must upgrade the installed base to Windows 4.0 to utilize distributed objects—Windows 3.1 will not be included in Microsoft's distributed object technology. Like Windows NT, Windows 4.0 is a new operating system that is not based upon DOS. OLE support will be required of all vendors intending to display the Windows 4.0 logo. This support can be as minimal as uniform data transfer, an extended clipboard facility. OLE's real advantages for the end user reside in the compound document framework, which involves the most complicated and unstable of the OLE APIs. Many future additions and changes are planned for these APIs.⁴

Incremental technology advances are a recurring pattern in Microsoft's technology evolution. This pattern includes single-generation backwards compatibility (Fig 1). Technology more than one generation old is effectively obsolete. Being a major independent software vendor (ISV) in its own right, Microsoft can push this evolution at a pace of its own choosing. Currently, the technology cycle is operating about 2 to 3 years per increment. The internal Microsoft application developers have about 6 months lead time on their external ISV competitors. Given this trend, one can reasonably expect new Microsoft technologies to arrive at frequent intervals with predictable obsolescence of previous technologies (Fig 1).

This situation is driving many groups to invent their own value-added integration technologies such as Lotus' Vendor Independent Messaging, WordPerfect's OpenDoc alliance, and Taligent. With these new technologies, vendors hope to achieve some technology discriminators, but, more importantly, these groups want to achieve independence from the Microsoft-controlled technology cycle. Corporate developers and system integrators have similar concerns because the constant technology change drives up operations and maintenance (O&M) costs.

CORBA

The Common Object Request Broker Architecture (CORBA) is the foundation of a standards and technology initiative coordinated by the OMG. The OMG has more than 385 member organizations, including virtually all computer platform vendors and major software vendors. The OMG has close alliances with other industry standards groups, such as the ISO ODP committee, X/Open, X Consortium, and OSF. Recently, CORBA was upgraded to join X/Open's Common Application Environment, indicating a strong worldwide consensus in support of the CORBA standard.

The CORBA specification defines distributed object facilities that provide the enabling infrastructure for all other OMG standards. These standards build upon the CORBA foundation with service definitions called Object Services, Common Facilities, and Application Objects (Fig 2).⁵ These specifications define the essential software boundaries needed to provide interoperability between independent distributed objects. Because the OMG's

OO development.
Reusable components instead of raw code. It sounds too good to be true. And it is. The culprits are the hundreds of classes and thousands of methods there are to keep track of. Fact is, a class is simply too small a unit of reuse to be practical in a changing business development environment. But now there's Synchronicity, the world's first business object management tool from Easel Corporation. "Hold on," you say. "What is a business object?" Glad you asked. Business objects combine many low-level objects into a single, recognizable business element—an invoice, for example. And with Synchronicity, you can move from design to assembly and back, to ensure that the business, system and application remain

A MESSAGE TO THOSE WHO HAVE JUMPED INTO THE OO SANDBOX ONLY TO FIND IT FILLED WITH QUICKSAND.



Object Studio 1-800-OBJECTS

synchronized at all times. Which means that, Synchronicity, the world's first business object management tool, gives you control over minutia. So you're less likely to be buried by it. Synchronicity is part of a powerful OO family, Object Studio. Featuring our popular Smalltalk environment, ENFIN™. Get an independent white paper on business objects, by calling Easel in the US: 1-800-OBJECTS (1-800-625-3287). Germany: 04101-5401-0. United Kingdom: 0344-304611. Japan: 03-3355-7195. Or anywhere else: 617 221-3000. But act quick.

Circle 20 on Reader Service Card

Table 1. Feature by Feature comparison of CORBA and OLE/COM

Common Object Request Broker Architecture (CORBA)	Object Linking and Embedding Common Object Model (OLE/COM)
<i>Distributed Object Infrastructure</i>	
CORBA Method Invocation	OLE Local Interface Calls and COM Remote Procedure Calls
OMG Interface Definition Language (OMG IDL)	Object Definition Language (ODL) and COM IDL
Inheritance	Aggregation and Containment
OMG IDL Static Interfaces	COM IDL Custom Marshaling Code
Dynamic Invocation Interface (DII)	OLE Automation/IDispatch (not distributed)
Interface Repository	Type Libraries
Implementation Repository	Dynamically Linked Library (DLL) Objects and Server-Based Objects
Basic Object Adapter (BOA)	Object Creation APIs
<i>Object Services</i>	
COSS Persistence Object Service using replaceable protocols such as ODMG-93, UNIX IO, and Bento	Structured Storage Specification using OLE/COM Compound Files
COSS Event Notification Object Service	IAdviseSink, IOleAdviseHolder
COSS Naming Object Service	Instance Creation APIs, QueryInterface
COSS Life Cycle Object Service	Instance Creation APIs
Query Object Service	Open Data Base Connectivity (ODBC)
Relationship Object Service	OLE/COM Monikers
Transaction Object Service	-
Externalization Object Service	-
Security and Time Object Services	-
<i>Common Facilities</i>	
User Interface Facilities	WIN32 API
Information Management Facilities	-
System Management Facilities	Hermes
Task Management Facilities	OLE Automation (not distributed)
Object Services Specializations	-
Vertical Common Facilities	Windows Open Services Architecture
Compound Document Technologies: Fresco, OpenDoc, Taligent	OLE2 Compound Documents (not distributed)

specifications are all based upon the CORBA foundation, there is no obsolescence built into the OMG's technologies. CORBA is a general-purpose, high level specification that can be supported by new generations of networks, languages, and platforms as they become available. The Object Services comprise globally applicable service definitions. Their generality makes them adaptable to new requirements and changes in technology. Common Facilities are a new area of OMG standards providing for higher-level application interoperability.

The CORBA standard was adopted 3 years ago and published in December 1991. Since that time, it has provided a stable basis for development and further service definition.⁶ There are nearly a dozen CORBA Object Request Broker (ORB) products available today that provide distributed object development environments.^{7,8} The OMG conducts an open consensus process that is predictable and easy to track. This process provides stability for suppliers and consumers of object technology. All draft OMG specifications are available on the Internet. OMG's adopted specifications are readily available in computer bookstores, and can be freely commercialized by both members and non-members.^{2,5,9}

The most important specification within CORBA is the OMG's Interface Definition Language (OMG IDL). OMG IDL and CORBA's Dynamic Invocation Interface (DII) are completely stable parts of CORBA

that will be unaffected by future revisions, such as future interoperability specifications (ie, CORBA 2.0 and COM/CORBA). These specifications are used to define the majority of APIs on which application software will depend.

OMG IDL is a standard for writing new standards. It is a higher level, general purpose notation for specifying object-oriented APIs. Because it contains no implementation information, OMG IDL has been found to be useful for developers because it provides very good object encapsulation. Because the OMG IDL is language independent, a single OMG IDL specification defines multiple language bindings for a set of user defined APIs. OMG IDL is a concise and rigorous notation that simplifies the specification of software architectures and new standards. Along with the ORB, OMG IDL simplifies programming and facilitates reusability by decoupling client code from dependence on object implementations.

COMPARING OLE/COM AND CORBA

Both OLE/COM and CORBA provide general capabilities for distributed objects. Table 1 is a point by point comparison of OLE/COM and CORBA. This comparison is partitioned into three categories:

distributed object infrastructure, object services, and common facilities. These categories correspond to the industry standard reference model for distributed objects, defined in the Object Management Architecture Guide.⁵ This mapping does not imply coverage of capabilities in any particular category. The table represents a nearest match of some key features of these different technologies.

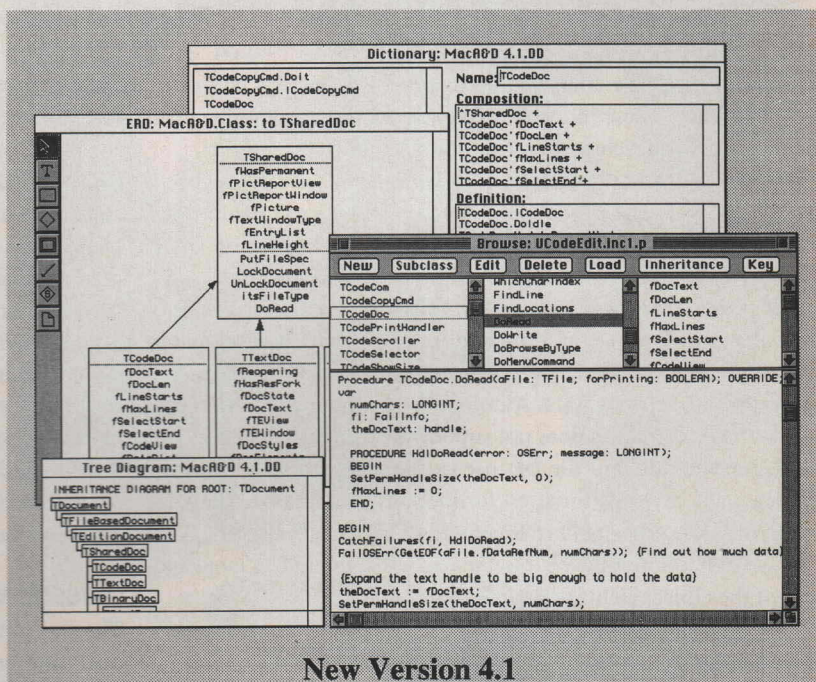
Both infrastructures support object RPC capabilities with automatic server startup, but they differ in their specification level. CORBA is defined in terms of high level language bindings and OLE/COM is defined as a binary API. High-level language bindings to CORBA can be uniquely standardized, so that source code is portable. OLE/COM's binary API does not guarantee portable language bindings; there is no standards process in place to establish unique OLE/COM bindings. Using the Microsoft C++ compiler, the binding is reasonably seamless between C++ objects and OLE/COM objects. The C binding to OLE/COM is quite complex and unnatural requiring explicit pointer dereferencing through the OLE/COM function tables:

(* Example of C function call to OLE/COM: *)
pInterface->lpVtbl->Function(pInterface, [parameters]);

MacAnalyst and MacDesigner

Computer Aided Software Engineering For The Macintosh

Structured Analysis
Structured Design
Real-Time Extensions
Data Modeling
Screen Prototyping
Object-Oriented Analysis
Object-Oriented Design
Task Design
Data Dictionary
Requirement Database
Code Browsing
Reengineering



New Version 4.1

Why choose the standard in Macintosh CASE?

You'll appreciate a CASE tool with all the capabilities you need, for even the most demanding project. And works like other Macintosh tools.

While some tools help with structured analysis, structured design and data modeling, few have the breadth to handle all your development projects.

For object-oriented analysis and design we support popular methods including Coad/Yourdon, Booch, Shlaer/Mellor, and OMT. And back each diagram with a data dictionary that understands inheritance.

The browser uses dictionary information for easy navigation and editing of object classes, attributes, and operations. Double-click to code anywhere on the network from any type of diagram.

Our reengineering support includes C++ and Object Pascal. From source code, you can populate

the data dictionary and automatically generate class diagrams using your favorite notation.

Real-time designers will be glad to know we haven't overlooked their special needs. Task diagrams express multi-threaded design with services like event flags, semaphores, and queues of a real-time operating system or scheduler.

Use our requirement database to ensure traceability of requirements throughout the development life cycle. So when changes do occur, the impact is quickly determined.

These are a few of the reasons why thousands of developers worldwide have made MacAnalyst and MacDesigner the standard in Macintosh CASE tools.

We invite your call today to bring out the very best in your development team.



Excel Software

P.O. Box 1414 • Marshalltown, IA 50158 • Phone: 515-752-5359 • Fax: 515-752-2435

MacAnalyst and MacDesigner are trademarks of Excel Software. © 1994. All rights reserved.


```
(* Equivalent C++ function call to OLE/COM: *)
pInterface->Function( [parameters] );
```

CORBA language bindings are designed to be well matched and natural for each language. Today, commercial products support CORBA bindings for C, C++, and Smalltalk. Other CORBA language bindings are in development, such as Ada, COBOL, and Common Lisp. The OMG standards process is actively adopting new bindings. These industry standards enhance portability and reduce risk for developers and compiler vendors.

Both OLE2 and CORBA support dynamic method invocation, ie, the ability to call new object interfaces discovered at runtime. The CORBA DII was the predominant capability in early ORB products, before OMG IDL compilers were commonplace. Although the DII is present in today's CORBA ORB products, the use of static interfaces are much more popular with developers, given the convenience of less code and the natural language mappings from OMG IDL. OLE Automation is similar to the DII, except OLE Automation does not support distributed objects. Both OLE Automation and the DII use runtime argument structures and a single `invoke()` function for method invocations. The DII uses the OMG IDL standard for interface specifications; OLE Automation has a unique, proprietary interface definition language called the Object Definition Language.

In the available OLE and CORBA technologies, there is a dramatic difference in user extensibility. CORBA provides comprehensive capabilities for user-defined static interfaces in today's ORB products. Because COM will not be available for another year, OLE Automation is really the only feasible way to call user-defined methods through OLE2. The alternative is for programmers to write their own marshaling code, a process that is not adequately documented in the OLE2 reference manuals.¹⁰

In the Object Services area, OLE/COM and CORBA differ principally on their generality. CORBA Object Services are very general-purpose APIs that can be subtyped to support specialized application needs. For example, the OMG's Event Notification Service is a general framework for event posting and delivery of events, providing all major forms of process interaction (such as blocking, callbacks, and polling).⁹ In contrast, OLE APIs are rather narrow in applicability. For example, the OLE analogy of the event service, `IAdviseSink`, is a special purpose interface. `IAdviseSink` is limited to a small set of events of interest to OLE compound documents, such as, closing an embedded object window.

Object services such as naming and relationships make extensive use of object identifiers (OIDs). Object identification is one of the most fundamental concepts in object orientation.⁵ CORBA provides a globally unique object identifier for all objects, including object classes and object instances. CORBA's OIDs are opaque references that hide differences in object location, programming language, and operating systems. Users of CORBA OIDs are supported by a standard naming service and a future trading service with directory services for objects.⁹

In OLE/COM, object identification comprises an ad hoc combination of three mechanisms: class identifiers, interface pointers, and monikers. Class identifiers (CSLIDS) are globally unique identifiers

for an object type. Object instances are referenced only by interface pointers, which are not persistent. The third mechanism, monikers, comprises an application-defined linking service used to encapsulate file references and other relationships. File monikers store absolute and relative file pathnames to provide two ways to find a file. This is an incremental improvement upon OLE1, but still lacks robustness. Item monikers can be used in an application-dependent manner to provide intelligent naming. Monikers are an extensible mechanism, but there is no standards process in place to provide consistent handling of monikers. Few moniker conventions exist today. OLE/COM's lack of a unified, robust object identification mechanism is problematic when used for distributed objects. For example, location-dependent names (such as pathnames in file monikers) do not translate well across machine boundaries.

Common Facilities is an important new area of OMG standards. These define higher-level APIs for distributed objects. Like all other OMG standards, Common Facilities utilize the uniform service access model standardized in CORBA. The OLE/COM services analogous to Common Facilities use a mixture of access models: some distributed (WOSA, Hermes), some local (WIN32 API), and some object-based (OLE Compound Documents and OLE Automation). Today, none of these OLE/COM services utilize distributed objects.

CHOOSING BETWEEN OLE/COM AND CORBA

The choice between OLE/COM and CORBA for corporate developers and system integrators should be made using a rational decision-making technique.¹¹ The decision process should involve some independent investigation of the facts behind vendor claims. There is a great deal of marketing evangelism on both sides of this issue. Claims can be confusing and sometimes misleading. OLE/COM and CORBA are both complicated technologies, and there are many organizations that have made premature choices for OLE/COM and for CORBA.

An important factor to consider is the availability of the technologies over the lifecycle of the target system (Fig 3). End-user system lifecycles can extend upwards of 10 years. Typically, 70% of the software cost will be expended for operations and maintenance (O&M) after the system is built.¹² Any software dependent upon a proprietary technology cycle (ie, OLE/COM) will need to be upgraded and maintained frequently. Software based upon a standards technology cycle (ie, CORBA) is more likely to resist obsolescence throughout the end-user system lifecycle.

A key factor in your end-user environment is the dependence upon custom software. If your end-user environment uses 100% commercial off-the-shelf (COTS) software on Windows platforms, then OLE/COM may be a logical choice. In this case, the burden of OLE/COM development will be the responsibility of the COTS software vendors. It is important to maintain up-to-date maintenance contracts on the COTS software and to update promptly when new releases are available. Some end-user organizations are forced to repurchase COTS software when it becomes more than one release out of date. Because the COTS vendors are releasing upgrades all the time, it is a continuous O&M responsibility to keep the installed versions current.

If your end-user environment uses custom software, then you

*It all
adds
up...*

8
+ 35

+ 20,000

years in the
business of object
technology
training and *only*
object technology
training

on-site and
open enrollment
introductory,
intermediate, and
advanced courses

satisfied attendees
who gained OT
skills they could
immediately apply
to their projects

Call 1-800-937-8080

Call Semaphore
for more
information about
OT training

 **SEMAPHORE**

800 Turnpike Street, North Andover, MA 01845 • (508) 794-3366 • Toll Free: (800) 937-8080 • Fax: (508) 794-3427

Circle 22 on Reader Service Card

should consider adopting CORBA. CORBA can support a user-defined software architecture written in OMG IDL that meets the technology cost and stability needs of your organization.

It is possible to have a combined approach supporting both OLE/COM and CORBA. ORB vendors such as DEC and IONA are developing CORBA gateways to OLE/COM. These gateways enable CORBA-based software to interoperate with shrink-wrapped OLE/COM applications.

A key factor in your choice between OLE/COM and CORBA should be the complexity of the APIs. CORBA allows developers to control complexity through OMG IDL encapsulations. Using OMG IDL, developers can expose what they want to expose and hide what they want to hide. One of the first facts one discovers about OLE/COM is that it is a very complicated API.⁴ Excess complexity is not a desirable quality for corporate developers and system integrators. OLE1 had more than 50 API functions; OLE2 is an order of magnitude more complex, comprising nearly 400 API functions. Big API libraries are difficult to learn and expensive to utilize. Each API adds to the complexity of application software; it also makes it more difficult to upgrade the code for new features (or new releases of OLE). OLE2 comprises a large number of special-purpose APIs that make application software complexity unavoidable. For example, all programmers are familiar with the input-output functions open, close, read, and write. OLE2's persistence scheme replaces these functions with over 70 API functions. The advantages of the new persistence scheme for the end user (such as enhanced browsing and spell checking) may not be worth the order of magnitude of additional cost due to complexity.

CONCLUSION

One of the key problems with Microsoft's incremental technology approach is that many short-term limitations are built into current APIs. These limitations are used to justify the obsolescence of legacy Microsoft APIs. For example, current OLE2 embedded objects are limited to one page in size. OLE2 objects are also restricted to an opaque, 2-D rectangular shape that cannot overlap other objects. These limitations restrict the potential uses of OLE2 for many applications such as graphics and simulation. In these applications, object properties such as complex shapes, 3-D, transparency, and object movement are not readily supportable in OLE2. Alternative technologies such as OpenDoc, Fresco, and Taligent address these limitations. These technologies are designed with extensible architectures that do not require substantial revision to support specialized application requirements.

In distributed objects, Microsoft is technically lagging the industry. COM is a future unreleased technology that lacks the features and robustness of current CORBA-based products. To a distributed object expert, COM looks like an ad hoc extension to DCE RPC. It is too low level, and it is heavily populated with proprietary extensions to OSF DCE. COM is a first step toward distributed objects, but it attempts to reinvent many of the mechanisms successfully addressed by the CORBA vendors and the OMG standards process. Unfortunately, due to Microsoft's influence, COM represents a major fragmentation of distributed object technology.

The recent initiative to provide interoperability between COM and

CORBA is a step in the right direction. At a recent OMG meeting, Microsoft joined with DEC and Candle Corporation (Santa Monica, CA) to propose standardized interoperability between COM and CORBA. Once standardized, Microsoft and ORB vendors could provide interoperability between COM and CORBA in a consistent way. Discussions are continuing at OMG regarding this potential standard.

Another approach to unify distributed object technology would occur if COM migrates toward the OMG IDL. Because CORBA has the maturity that COM lacks, this makes more sense than further COM divergence from the standard object model. Because the CORBA specification does not mandate the underlying mechanism, COM's version of Object DCE could be utilized. These and other flexibilities built into the CORBA specification make it feasible to utilize the OMG IDL as COM's interface definition language, instead of COM IDL. For corporate developers and system integrators, COM's migration to OMG IDL would provide a ubiquitous distributed object infrastructure supporting portability and interoperability.

Without taking this step, Microsoft must accelerate the rate of iterative technology evolution and obsolescence to catch up to the industry in distributed objects. This acceleration is contrary to the interests of corporate developers and system integrators who need to design stable software architectures for long system lifecycles. A technology and standards process, such as OMG/CORBA, gives corporate developers and system integrators an additional level of stability and control of technology essential for managing costs in large-scale distributed object systems. ■

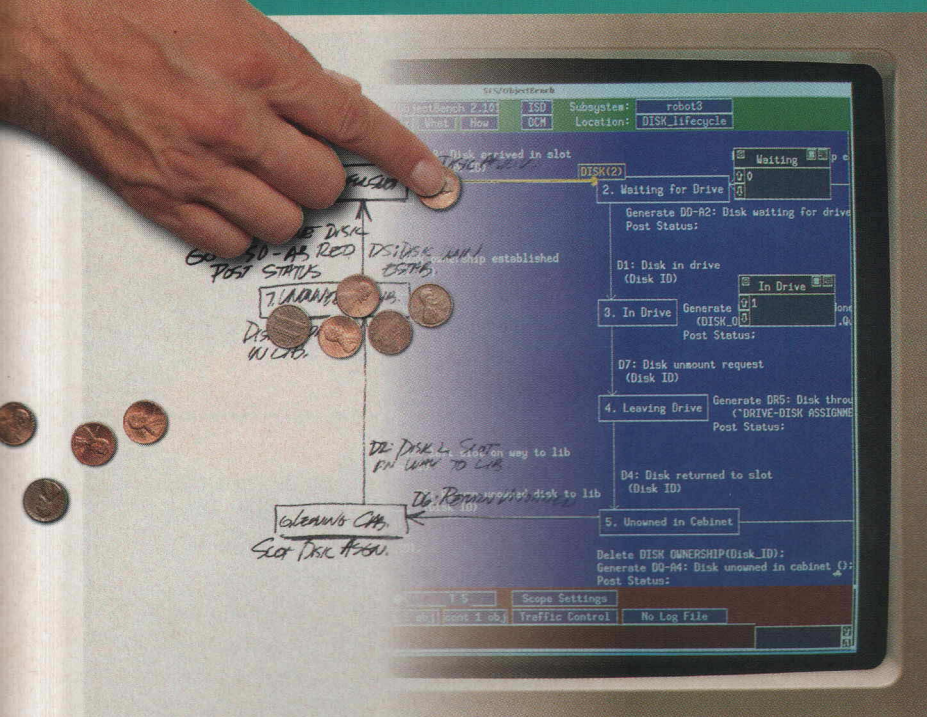
References

1. MS Cromer. OLE vs ORB: A tale of two technologies, *OBJECT MAGAZINE* 1(6):35-38, 1992.
2. COMMON OBJECT MODEL SPECIFICATION, Draft Version 1.0, Digital Equipment Corporation and Microsoft Corporation, 1994.
3. Object Management Group. COMMON OBJECT REQUEST BROKER ARCHITECTURE AND SPECIFICATION, John Wiley, New York, 1992.
4. K Brockschmidt. INSIDE OLE2, Microsoft Press, Redmond, WA, 1994.
5. Object Management Group. OBJECT MANAGEMENT ARCHITECTURE GUIDE, John Wiley, New York, 1993.
6. TJ Mowbray and T Brando. Interoperability and CORBA-based open systems, *OBJECT MAGAZINE* 3(3):50-54, 1993.
7. CORBA PRODUCTS DIRECTORY, Object Management Group, Framingham, MA, 1994.
8. TJ Mowbray. Distributed Objects Everywhere: A preview of Sunsoft's DOE, *OBJECT MAGAZINE* 4(1):64-69, 1994.
9. Object Management Group. COMMON OBJECT SERVICES SPECIFICATION, Vol I, John Wiley, New York, 1994.
10. OLE2 PROGRAMMERS REFERENCE, Vol 1 and 2, Microsoft Press, Redmond, WA, 1994.
11. CH Kepner and BB Tregoe. THE NEW RATIONAL MANAGER, Princeton Research Press, Princeton, NJ, 1981.
12. BM Horowitz. STRATEGIC BUYING FOR THE FUTURE, Libey Publishing, Washington DC, 1993.

Dr Thomas J Mowbray is Lead Scientist at The MITRE Corporation, McLean, VA.

"The [SES/objectbench] animator is a milestone in the evolution of software development."

Gene Forte of CASE Outlook, Sun World, February 1993



INTRODUCING A WHOLE NEW MOVEMENT IN OBJECT-ORIENTED ANALYSIS.

SES/objectbench

The CASE tool that animates your object-oriented models.

Simulation is an integral part of the object-oriented analysis process. Unfortunately, performing in-depth manual "walkthroughs" to simulate complex systems is cumbersome, time consuming, and accident prone. SES/objectbench is the first product on the market that visualizes, animates and allows you to interact with your OOA models to validate your analysis. Now you can be certain your applications function correctly.

Shlaer-Mellor OOA method.

Nobody supports it better than SES.

Accurate simulations require a complete, executable methodology. That is why we designed SES/objectbench for the Shlaer-Mellor method. SES/objectbench combines the most comprehensive graphical capture tool for the Shlaer-Mellor method with a well proven, user friendly, animating simulator. SES/objectbench streamlines the analysis process by making it easier to develop models and perform system validation. Well validated analysis models shorten your implementation and debugging time and help assure correct, robust systems.

The right tool for the task at hand.

Finally, an OO tool exists that fulfills the promise that CASE methodologies will help you create better systems quicker, with more reusable components. Don't spend time pushing pennies - make the move to SES/objectbench.

Recent enhancements and extensions to SES/objectbench.

SES/objectbench now includes many new features to support software development by large teams. These include:

- Domain and multiple sub-system support
- Concurrent usage and automatic versioning
- Architecture & performance modeling via SES/workbench® integration &
- C++ code frame generation.

Discover the benefits of automated OOA support today.



Call for a free
SES/objectbench booklet.
You'll learn how easy it is
to create complete, correct,
verified OOA models using
simulation and animation.

800 759 6333
512 328 5544



4301 Westbank Drive
Austin, Texas 78746

512 327-6646 Fax

Circle 23 on Reader Service Card

Developing a medical diagnostic system with OOA/RD

The question for objects is no longer "Why?" but "How?"

Jim Ladd

TO REALIZE THE full benefits of objects, the object paradigm must extend beyond the programming phase of the software development process.

While the spotlight of the software community has been directed at object-oriented programming (OOP), the more important issue of object orientation as a development philosophy remained on the collective periphery. As object technology gains popularity, this issue is forced to become the focal point of the community. The question for objects is no longer Why? but How?

BACKGROUND

Different people develop software differently, yet a common thread is woven through all software development activities. From the solo hobbyist coding to the multiperson teams building the next generation securities trading system, all will experience an understanding of the problem, a selection of a solution, and a creation of that solution.

This process is also known as analysis, design, and implementation:

- **Analysis is the study of a problem.** The analysis of a problem concentrates on what must be done and not how it is to be accomplished. The result of the analysis efforts is a set of descriptions of the problem.
- **Design is the selection of a particular solution for a given problem.** A solution specifies exactly how the problem is to be solved. The results of the design work are specifications and policies that guide the construction of the solution.
- **Implementation is the creation of the chosen solution for the problem.** During

the implementation phase, the source code is produced according to the design specifications and guidelines.

Although all software constructors (individual and organizational) perform analysis, design, and implementation, far fewer use formal methods, techniques, and tools to do so. The question should not be whether such things are advantageous. The question is how to forge the methods, techniques, and tools into an effective and efficient software development paradigm.

YET ANOTHER BLOOD ANALYZER—YABA

One of our more recent projects to follow the Shlaer-Mellor method is one I will refer to as the YABA (yet another blood analyzer). The YABA is a medical diagnostic instrument for determining the characteristics of certain components of white blood cells. The goal of the YABA is to automate the steps of the testing protocol.

The testing protocol is a controlled series of operations performed on the cells. Chemicals are added to the blood sample and the mixture is incubated. Depending on the type of test, additional mixing and incubation cycles may be performed. When the chemical reactions are completed, the sample is exposed to a detection system that gathers raw data. Finally, this information is processed and reduced to a set of final results.

The automation of the testing procedure is accomplished through the intelligent coordination of electro-mechanical devices. The major components of the YABA include a fluidics system for liquid chemical management and distribution, an incubator, an image processing module for assay detection, and two 3-axis robotic platforms for transporting the samples within the instrument. Small electric

motors and solenoids actuate the mechanisms of the components.

The requirements for the YABA extend beyond the real-time control of the instrument. The YABA provides the user with test protocol configuration, sample and patient data management, results analysis, report generation, and diagnostics capabilities.

Analysis

The first step of the analysis was to partition the problem into domains. The latest version of the YABA domain chart is shown in Figure 1. At the top of the chart is the application domain that was the most specific to the project. After the initial analysis was completed, the application domain was divided into two subsystems. The instrument subsystem was concerned with the processing of the sample and contained objects of the instrument such as the Input Load Station, the Pipettor, and the Reader Robot. The other application subsystem contained the objects that were of more interest to the user. Objects within this subsystem included the Patient, Results Report, and Test List. A few objects, such as Sample and Test, were shared by the two subsystems.

The application domain required the services of other domains. These domains consisted of the subject matter that was less specific than the application domain. The service domains of the YABA include the following:

- **Scheduler:** This domain reserves the instrument's components according to the resource and timing requirements of the test protocols. It calculates and maintains the timelines for the different resources (pipettor, optics system, etc.) within the instrument. Objects for this domain include the Protocol, the Activity, and the Resource objects.

Jim Ladd is Software Manager at Abbott Laboratories, Irving TX.

- **Command sequence:** Once a Protocol reserves a Resource object, that Resource must perform an Activity at the scheduled time. The Activity object is a sequence of commands. This domain parses the Activity into the underlying commands and monitors their execution. The Aspirate Command, Level Sense Command, and Get Tray Command are a few of the objects of this domain.
- **Motor controller:** The Motor Controller domain accepts, queues, and executes motor-specific instructions for the custom-built motor controller. This domain includes the Move Command and the Stop Command objects.
- **Image processing:** The Image Processing domain performs the operations needed to acquire, manipulate, and analyze images. Object-oriented analysis was not used for this domain because the algorithms were "discovered" through theoretical and experimental work and modeled with standard flow charts.
- **Process I/O:** This domain interfaces with the outside world. It is concerned with interface entities such as hardware interrupts and I/O ports.
- **User interface:** This domain deals with the human interface screens and the corresponding user interaction. The Patient Demographics Screen and the Test Ordering Screen are two objects within this domain.
- **GUI:** The GUI domain provides the window management and display functions. The GUI is a commercial package and provides the functions common to products of this type.

The architecture domain is concerned with the organization of data and control of the software. It was specified during the design phase.

The domains at the lowest level of the chart involve the implementation entities. These domains include the operating systems and programming language. The network interface was a commercially available package and did not require analysis. These domains were also selected during the design phase.

A benefit of domain analysis is the potential for reuse of work. The lower domains have the greatest potential. Obviously, a

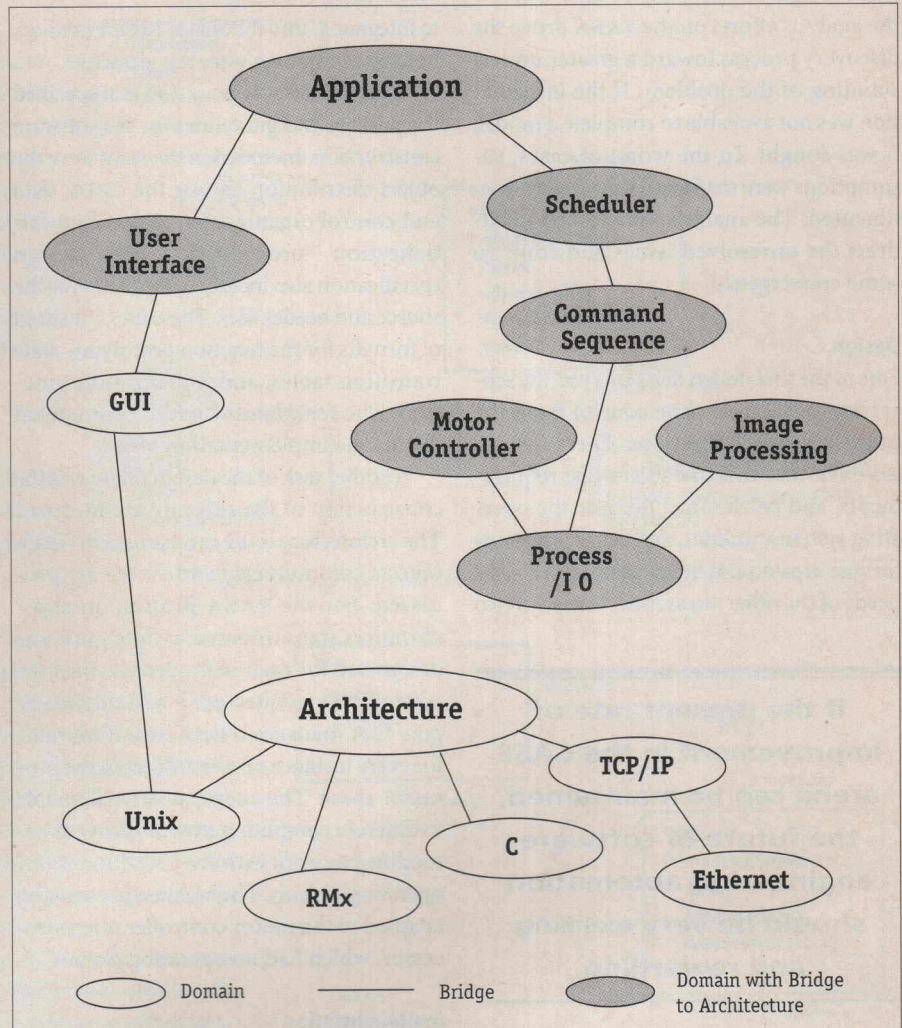


Figure 1. YABA domain chart.

operating system and programming language domains are extremely reusable. The service domains also provide opportunities for reuse. For example, the objects of the motor controller domain are "unaware" of the YABA system; they operate in a contained world. The analysis models could be easily applied to other projects requiring the service of a motor controller. The application domain offers the least reusability due to its project specificity. However, objects within this domain would provide an excellent starting point for the application domain of a similar project.

After the domains were identified, the analysis within the domains began. A small group (>5) of analysts (in our case, engineers with their analysis hats on) would meet to define a domain's mission statement and brainstorm for obvious objects. At the conclusion of the initial meeting, a

single analyst was assigned responsibility for the domain. Over time, if the subject matter of a domain was too much for the solo analyst, the domain was partitioned into subsystems. The boundary of the individual was reduced from the domain to the subsystem level. Review meetings were used to update the other analysts on the progress of the models.

The actual act of analysis can be imposing even at a subsystem level. Several techniques were adopted by the software group to simplify the task. One technique was to model the normal behavior of the objects in the initial analysis. The required failure analysis and error handling were modeled after the normal behavior of the objects became stable.

A major doctrine of the Shlaer-Mellor method is that the analysis models should be exact and complete descriptions of the problem. By embracing this philosophy,

the analysis efforts on the YABA drove the discovery process toward a greater understanding of the problem. If the information was not available to complete a model, it was sought. In the worst of cases, assumptions were made and thoroughly documented. The analysts were forced to address the unresolved issues and come to some convergence.

Design

One of the first design decisions was the separation of the real-time control from the human interface functions. These two areas have very distinct objectives, requirements, and constraints. Because the operating systems, utilities, and other programs for one area do not adequately address the needs of the other, an architecture with two

to integrate C into the Shlaer-Mellor method, but the alternatives were less attractive.

A set of rules was generated that specified the policies and guidelines for the software construction. Included in the rules were the object distribution among the CPUs, data and control organization, and system initialization procedures. The design specification also included templates for the source and header files. These files consisted of formats for the function prototypes, state transition tables, and initialization function. The templates would be completed during the implementation phase.

Another task of the design phase was the construction of the software architecture. The architecture is the mechanism in which objects communicate and events are processed. For the YABA project, an asynchronous state-oriented architecture was designated for each microprocessor in the system. This architecture is basically an engine that manages a state transition table for every instance of every object in the processor space. The engine is very adaptable to different computing environments. It was modified to work with the UNIX and iRMX operating systems. The architecture was also adapted to the motor controller microprocessor, which had no operating system.

Implementation

In the implementation phase, programmers transformed the analysis models into source code according to the design rules. Deviations from the design rules were not allowed. If a question or issue surfaced, it was addressed by modifying the design rules or clarifying the analysis models.

The code generation was very straightforward and mechanical. The source and header templates were completed for each object using information from the analysis models. Each process of the process model became a function in the C programming language. The task of code generation was extremely scalable. The time required for coding was dependent on the number of analysis models and the amount of programming resources available.

On previous projects, there has been much debate on the type and amount of comments to be placed in the source code. This argument was finally settled on the YABA project. No comments were to be in

the source code. The valuable information is in the analysis models and design rules. The source code is now relegated to the same level as assembly and machine code.

Debugging

The software produced with this process encourages debugging strategies that differ from those used with traditional software. If a problem arose during program execution, first the source code was compared with the analysis models. If a discrepancy was found, the error was fixed and debugging continued. If no deviations were discovered, the analysis models were inspected for errors. Typically, the events of the object were manually traced and the corresponding transitions of the objects observed.

For very evasive bugs, manual event tracing through the models proved difficult and alternative approaches were examined. One approach was to use the executable code as a debugging platform even though the system might be incomplete. With object-based software, attention can be directed to the behavior of a single object or the objects within a domain with minimal effort. Only the events that cause the objects to transition must be provided for this activity; it does not require involvement of the entire system.

CASE Tools

The fundamental concepts of the Shlaer-Mellor method encourage a high level of automation. Unfortunately, potential is one issue and realization is another. When the YABA project was in development, few CASE tools supported OOA/RD. The products that did were somewhat overly optimistic in their claims. Still, while the early tools were limited in features and were at times difficult to use, the software group continuously embraced CASE technologies because (1) while imperfect, the available CASE support added value to the process, and (2) all advancements provided by the tools would improve the analysis and design efforts of the group.

Recently, after a number of years of inactivity, the CASE vendors are now taking notice of the potential for automating the OOA/RD processes. The field is becoming populated with tools offering automation in different parts of the process. Simula-

If the present rate of improvement in the CASE arena can be maintained, the future of software engineering automation should be very exciting and rewarding.

complete computing environments was chosen. An off-the-shelf personal computer was selected for the human interface and configured with the standard accessories. UNIX was chosen for the operating system on the human interface computer.

A passive-backplane industry-standard architecture (ISA) computer was selected to execute the control software. The control computer contained a motor controller board, image processing board, and input/output board. The control computer would run Intel's iRMX real-time operating system. An Ethernet connection with the TCP/IP protocols provided the communications channel between the two computers.

The C programming language was chosen over C++ because, at the time of the decision, our group had (1) no experience with C++, (2) little confidence in the C++ compilers for the embedded processors, (3) a strong desire for homogeneity across the processor boundaries. Additional work was necessary

Yes, it's for real. Just listen to what people are saying about...

Together/C++

continuously up-to-date
object modeling and C++ programming

To: Peter Coad, Object Intl., Inc.
From: Ken Gonzalez, ACT Technologies (c-serve 71501,2563)
Re: Together/C++

I just had to write you. I have never felt so strongly about any other product. Together/C++ is now one of my most valuable software development tools. I would never think of doing C++ development without it (that would just be far too painful!).

I have been programming in C++ for years. The hardest things I have to deal with are not the language features. Far from it. The hardest things I have to deal with are the complexities of the class libraries (from many sources: major vendors, third parties, and other developers here). I used to keep piles of paper, with crudely-drawn sketches of classes, with their variables and functions. This complexity problem is compounded by poor-quality vendor documentation. Not that they were poorly written. It's just that I couldn't keep up with their documentation updates. By the time I got the update into the binder, I was already using the next release. What a pain.

I soon realized that to be productive (using C++ and a multitude of class libraries to their full potential), I needed an automated tool to help me manage the complexities that I was forced to deal with. I surveyed many tools. Frankly, I was "underwhelmed" — none of them did what I needed.

At that point, I decided to do what any truly frustrated developer would do: build my own. Yet shortly after I got started, I saw the ad for Together/C++. I immediately called for more information. I talked with one of your tech support engineers. I thought to myself that this was too good to be true. I received the video brochure, watched it, called in and asked some more questions, and then ordered the product. (By the way, I am no longer building my own tool — I don't need to!)

The impact? Together/C++ has increased my rate of production exponentially. I can now catch design flaws and errors in my class libraries — before they become a problem.

I believe that Together/C++ is the key to fulfilling the promise of the object paradigm — and is truly in a class by itself. Congratulations!

Hearty thanks to the Object Intl. team. You have done a superb job, supporting me from initial contact all the way through to successful day-to-day use.

"Amazing and very cool! This is how a development tool should work!"

— Keith Zenow,
Z-Tech

"You've really hit the nail on the head when it comes to re-engineering existing C++ code. No other tool comes close to the power and capability of Together/C++."

— Russell Rudduck,
Perot Systems

"The ability to change either code or diagram and have the changes reflected in the other is a clear advantage...and the ability to import existing source code into the tool and have Together/C++ create a visual map of the library."

— David Moskowitz,
Productivity Solutions

"Together/C++ is the kind of product opening completely new possibilities for programming."

— Michael Inglot,
Electrohome

Software Development Magazine

"Together/C++ is a new tool from Peter Coad and the folks at Object International...it provides support for object-oriented analysis and object-oriented design, and code generation, and reverse engineering for object-oriented programming in C++..."

"Easy to use and has lots of feedback and features like automatic, semi-automatic and manual layout...other graphical features I appreciated included drag-and-drop..."

"Taken together (pun intended) with your favorite C++ compiler, these components provide a comprehensive software development environment...can save a whole boatload of labor hours... It looks from here like Together/C++ is a winner."

Windows Tech Journal

"Together/C++...supports dual views of C++ classes: an object-oriented design view and a source code view. Any change to the data in one view is immediately and automatically reflected in the other... The multiple-view approach is well-suited to the complexities of software development."

Object-Oriented Strategies Report

"Together/C++ is an OO CASE tool that combines OOA&D and a C++ programming environment. The tool generates C++ code frames from object models created within the tool."

"The main feature of the tool is that it allows developers to work in an object modeling window or in a C++ programming window side-by-side while keeping the two continuously in-sync with each other. Hence, the name Together/C++. Other OOA&D tools (such as Rational's Rose or Martin Marietta's OMTTool) generate code frames only (from object diagrams), just in one direction. In other words, changes in code are not reflected in the object model."

"Together/C++ allows developers to work with the object model or the code at any moment, and code and model are kept up-to-date with each other automatically at the developer's choosing."

Together/C++ comes with a full 30-day money-back guarantee. For more information (including a free special report from Peter Coad) or to order a copy so you can try it out for yourself, send e-mail to info@oi.com. Or contact Object Intl. at 1-512-795-0202 fax 1-512-795-0332. In Europe, Africa, and Asia, send e-mail to 100034.1370@compuserve.com. Or contact Object Intl. S/W Ltd. at ++49-711-225-740 fax 299-1032.

Together™ is a trademark of Object Intl., Inc. © 1994 Object Intl., Inc.

SHLAER-MELLOR METHOD

Although it is best known for its approach to object-oriented analysis, the Shlaer-Mellor method defines a complete paradigm for the analysis, design, and implementation of software systems. The method combines Object-Oriented Analysis (OOA) and Recursive Design (RD) techniques in a comprehensive approach for software development. OOA describes the notation, models, and procedures necessary to discover and model objects in the problem space. RD provides the transition policies and guidelines from the analysis to im-

plementation. The Shlaer-Mellor method also includes the process for applying OOA and RD to a system.

The OOA method uses three primary models to describe the problem space:

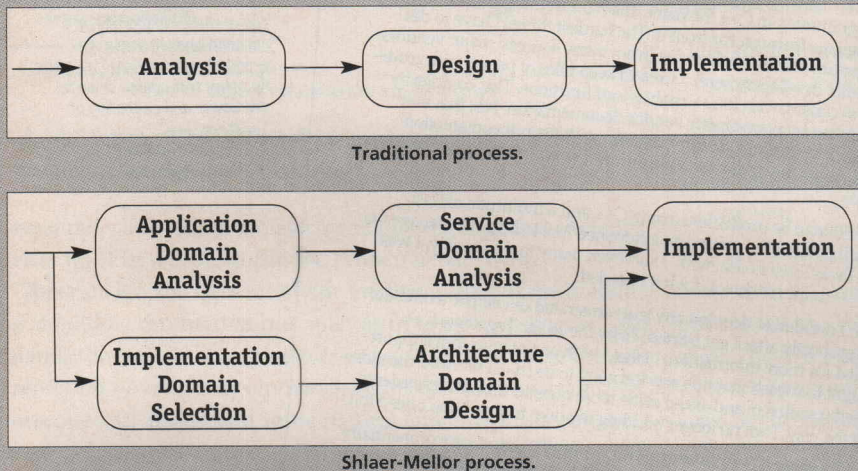
- Information models
- State models
- Process models

Integrated together in a comprehensive method, the models produced represent a complete and consistent description of

the problem. The models are so complete that they are, in fact, executable and offer the opportunity to verify the correctness of the analysis.

The cornerstone of recursive design is the design-by-translation concept. The analysis models are translated directly into code by applying design rules. The design phase focuses on the selection and creation of the rules and not on another set of models. The design rules guide the construction of the solution. During the implementation phase, the rules are applied to the analysis models to generate the source code.

The Shlaer-Mellor development process relies heavily on the concept of domains and bridges. Domains consist of specific subject matter. In the object world, domains are composed of closely related objects. Bridges are the connections between the domains. Recursive design defines four types of domains. The application domain consists of the material that concerns the user. Service domains provide generic functions. The architecture domain dictates the management of data and control of the system. The implementation domains are the programming languages and operating systems.



tion and animation of the state models and analysis-to-code translation are just two of the critical technologies offered by the latest versions of CASE tools. The products are also improving in other, more standard ways including enhanced report generation, better static checking of the models, more intuitive human interfaces, and open databases. If the present rate of improvement in the CASE arena can be maintained, the future of software engineering automation should be very exciting and rewarding.

RECOMMENDATIONS

The paradigm shift to object-oriented technology can be a confusing and frustrating passage. When coupled with the shift to the Shlaer-Mellor method, the task can be overwhelming at times. To accelerate the learning process for OOA/RD, the following recommendations are offered:

1. Acquire formal training whenever possible. Learn the fundamentals of the technology from the masters.
2. Target a microproject (< 6 objects) and take it through the analysis, design, and implementation phases.
3. Document everything. The trail will be valuable in post-project analysis and studies.
4. Avoid the temptation to "just code it." Follow the process by completing the analysis models and design rules before starting the implementation.
5. Don't get discouraged. Rigorous analysis is difficult and requires effort and time to master.

SUMMARY

The results of our efforts with the Shlaer-Mellor analysis and design method were bet-

ter documentation, improved understanding of the problem, increased communication among the team members, freedom from implementation technologies, and an increased potential of reusable work products. All team members were impressed by the OOA/RD methods and wanted to apply the paradigm to other projects.

The Shlaer-Mellor Method, however, is not for everyone. Those comfortable with traditional development practices might have difficulty adapting to the new paradigm. The shift from a code- or design-based process to an analysis-centric one can be as challenging as the shift to object technologies. To realize the potential benefits, the policies and guidelines of the process must be followed faithfully. For those who wish to migrate from software crafting to software manufacturing, the Shlaer-Mellor development paradigm is an excellent alternative. ■



The Users Conference and Exhibition

C++ from A to Z

November 14-18, 1994
The Stouffer Austin Hotel
Austin, Texas

Providing
the

Building Blocks

Join more than 2500 of your colleagues at the world's largest C++ training event of the year. Learn C++ techniques from the most prestigious faculty of C++ experts ever assembled under one roof at one time. Whether you are a C++ beginner, experienced user or managing a C++ development team, **C++ World's** technical training provides you with the strongest building blocks for advancing your C++ knowledge and skills.

Comprehensive Technical Program

This year's technical program has been expanded to include an even wider variety of topics. Top-notch faculty will offer instruction on:

- Modeling & Design with C++
 - Designing C++ Libraries for Evolution
 - CORBA and C++ Mapping
 - Virtual Functions in C++
 - Making the Transition from C to C++
 - Object Models: Strategies, Patterns, and Applications
 - Progress on Patterns
 - Mastering Pointers & Complexity in C++
 - Guidelines for Exception Handling
 - Writing Effective Versions of new & delete with C++
 - Inside the C++ Object Model
- and many more! 40 classes in all!

Circle 24 on Reader Service Card

for Innovation.

Keynote Presentations

Hear exciting keynote speakers, including **Bjarne Stroustrup** of AT&T Bell Labs, the originator of the C++ language; **Paul Gross** of Borland; **Gene Wang** of Symantec, and **Daniel Sabbah** of IBM.

Specialty Sessions

Attend **Walk-In Clinics** where your C++-related questions will be answered personally by the leading C++ experts and book authors; **Technical Paper Presentations** where you will hear the latest innovative techniques and unique problem-solving solutions in papers carefully selected by a panel of industry experts; **Product Education Sessions**, where you will receive customized, in-depth training on the latest products and services; **"Birds-of-a-Feather" Sessions**, where you will benefit from sharing innovative ideas and experiences with your peers; and the new **C++ World Job Fair**.

C++ World Cup Challenge

Don't miss the First Annual **C++ World Cup Challenge**, hosted by Jon Hopkins, President of Palladio Software. Challenge your knowledge of C++ with the leading industry experts and vendors!

Partial List of Exhibitors:

ACI
Alsys, Inc
C++ Report
C/C++ USERS JOURNAL
Cadre Technologies
CenterLine
DevelopMentor
ERI
ExperSoft
IBM
ICON Computing
JOOP
McCabe & Associates
Neuron Data
Object Design, Inc
Object International
OBJECT MAGAZINE
Objects ++
ObjectSpace, Inc.
Open Objects
POET Software
Prentice Hall
ProtoSoft, Inc.
ROAD
Rogue Wave Corp.
Select Software Tools
Semaphore
THE SMALLTALK REPORT
Software Emancipation
Symantec
Veritas Software
THE X JOURNAL
Zinc Software

SPONSORED BY:

C++REPORT

PRESENTED BY:

SIGS
CONFERENCES



YES!

Please send me more information on
C++ World '94

- ☐ Attending the Conference ☐ Exhibiting ☐ Receiving a Free Exhibits Pass

Name _____

Title _____

Company _____

Address _____

City _____ State _____

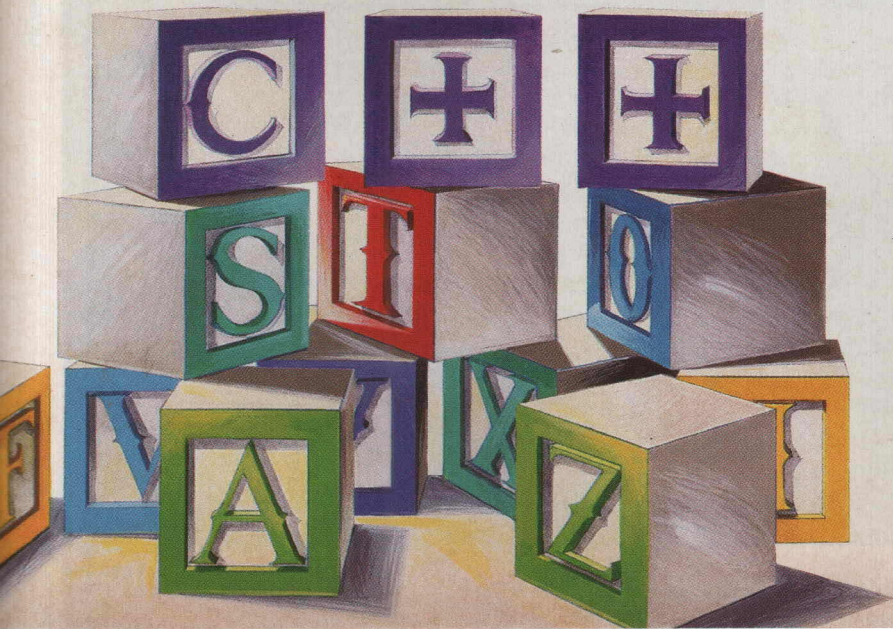
Country _____ Zip/Postal Code _____

Day Phone _____

Fax _____

Mail or Fax coupon to: **C++ World '94**

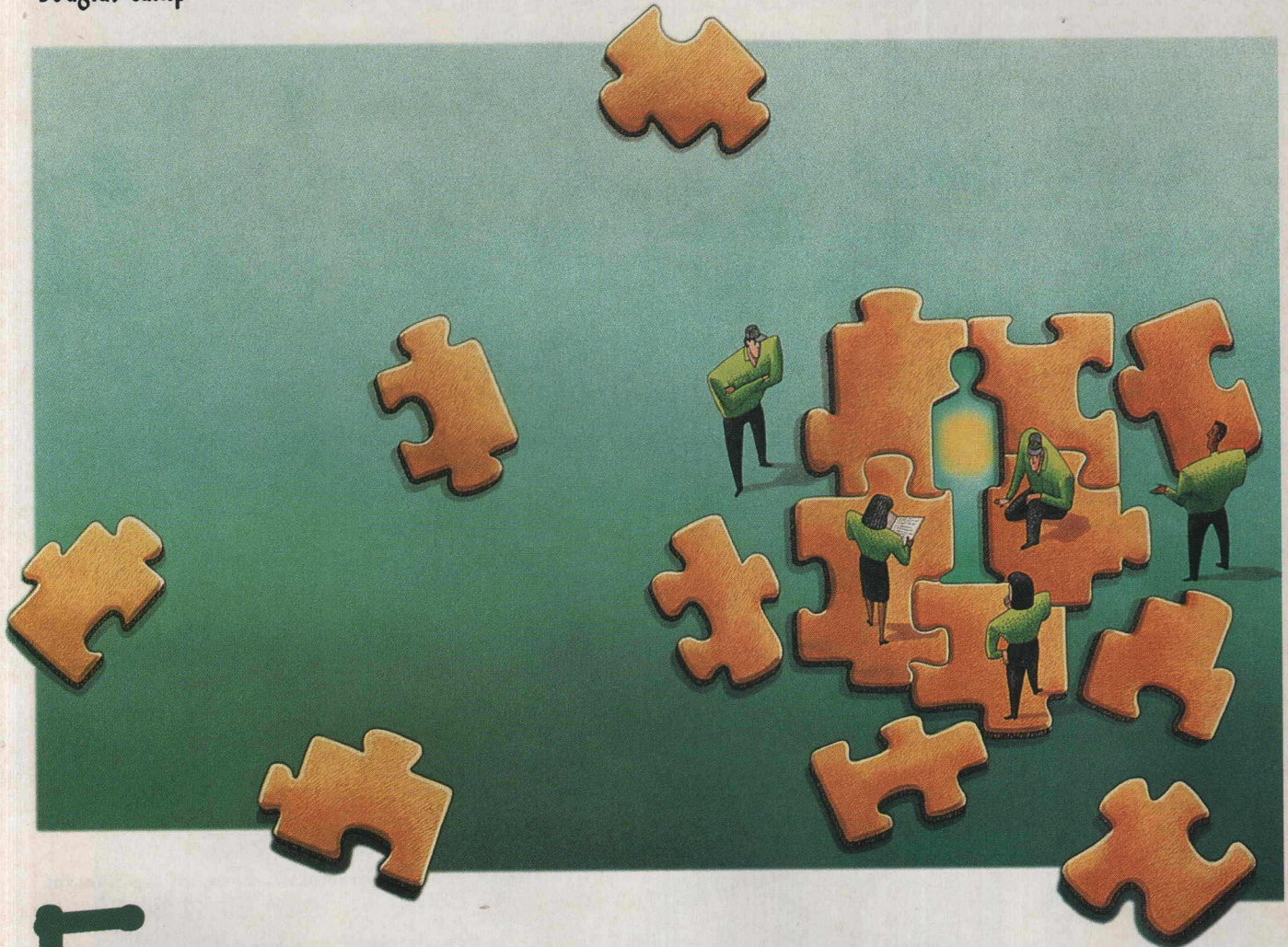
71 West 23rd Street, New York, NY 10010
Phone: 212/242-7515 Fax: 212/242-7578



Managing customer expectations in Smalltalk development

Douglas Camp

The task of the project manager is to understand what is possible, and align expectations accordingly.



David Chen/SIS

EXPERIENCED MANAGERS OF corporate software development projects know the importance of both understanding and managing the expectations of their customers, from senior managers all the way to end users. They know that if they fail to adequately *understand* customer expectations, any system they deliver very probably won't be what the customer wanted. And if they don't actively *manage* customer expectations, the system may be deemed a failure simply because the customer may have had unrealistic expectations of functionality, delivery time, or costs.

What do we mean by "managing customer expectations"? It means making certain that, during all phases of project develop-

ment, what your customers *expect* from the project is very close to what will be *delivered*. If your customers expect a Ferrari, and you deliver a station wagon, however capable, your project will have failed—even though the delivered system may have been the best that could be done given the technology and resources available to the project. The task of the project manager is to understand what is possible, given the constraints of the project, and closely align customer expectations with this possibility.

HOW IS SMALLTALK DIFFERENT?

The need to actively manage the expectations surrounding a software development project certainly



isn't new or peculiar only to Smalltalk projects. It doesn't matter if your project is done in assembler, COBOL, or an OOP language such as Smalltalk, the project manager will still be faced with the necessity of managing customer expectations. However, because Smalltalk is quite different from more traditional languages, and can dramatically change the way you do development, using Smalltalk does add some new wrinkles to the problem. This article discusses some of the ways in which Smalltalk changes the task of managing customer expectations, and suggests some general prescriptive strategies for better managing expectations on Smalltalk projects.

What are the differences when using Smalltalk? And how do they change the task of managing expectations?

Expectations are very high for Smalltalk

Smalltalk is very popular right now. Organizations of all sizes, in a wide variety of industries, are rushing into object technology (OT) in search of higher productivity, lower development costs, and other benefits promised by object orientation. Although these organizations may have no real depth of understanding of object technology, they may have very high *a priori* expectations of both OT and Smalltalk.

Many of these organizations may perceive Smalltalk to be the software equivalent of a "magic silver bullet"—a new technology that will enable them to deliver all their projects on time, under budget, with superior functionality.¹ CASE tools and AI had this reputation in the mid 1980s. Unfortunately, both these technologies were subsequently overhyped and oversold, and industry was ultimately disappointed.

Strategy. Smalltalk project managers should make certain their customers have realistic initial expectations. They should understand that, used properly, Smalltalk definitely can deliver on the promises of object technology. However, Smalltalk is only a programming language and development environment. Smalltalk alone definitely isn't a silver bullet—and simply using Smalltalk won't solve all the problems found in any software development project.

Very early in the life of a Smalltalk project the project manager should do a reality check with key project customers. Do they understand the capabilities of Smalltalk? Or do they have unrealistic, silver bullet expectations? If so, correct these misconceptions as early as possible. Don't allow unrealistic expectations of Smalltalk to persist, and be very careful not to oversell Smalltalk to your organization. It's critical that customer expectations of Smalltalk are brought in line with reality at this early stage rather than in the middle of the project.

Smalltalk is unknown

Smalltalk has only recently emerged from its status as the tool for revolutionary, limited audience, guerilla warfare projects led by early adapters of new technology. Even if your customers don't initially have unrealistic expectations of Smalltalk, they simply may not understand the technology sufficiently to form any appropriate expectations. Unless your organization has significant experience in object technology and Smalltalk, it's likely that your cus-

STRATEGIES FOR SUCCESSFULLY MANAGING CUSTOMER EXPECTATIONS ON SMALLTALK PROJECTS

- Dodge the "silver bullet"—don't allow unrealistic expectations to persist.
- Educate your customers about Smalltalk and object technology.
- Prepare your customers for Smalltalk's iterative development style.
- Clearly delineate between early prototypes and deliverable products.
- Plan for Smalltalk's *sticking points*.

tomers won't know much about either. Many of your customers may have extensive experience in non-object languages and development methods, and may strongly resist the transition to the relatively unknown waters of OT and Smalltalk. In this case, they may be unable to form appropriate expectations, or to communicate those expectations to the project manager.

Strategy. Educate your organization about object technology and Smalltalk. This can take many forms: in-house presentations, circulating journal articles, or providing introductory texts.* You may want to send key customers to outside classes to give them an overview of the language, and a working knowledge of the semantics of Smalltalk project development. The goal isn't to turn customers into Smalltalk developers. However, if the project manager can raise the general level of understanding of Smalltalk among the customer population, customer expectations are much more likely to be in line with the true capabilities of the language.

Smalltalk project planning and development is iterative

Most software development projects begin with a plan describing in some detail the tasks that must be completed to develop the system, and forecasting target completion dates for those tasks. The accuracy, and even the structure and appearance of this plan can have a profound impact on customer expectations. Good Smalltalk projects also start with a project plan; however, Smalltalk project plans may be quite different from plans with which your customers are familiar.

In the "old world" of pre-object software development methods, one of the benefits of the waterfall development model was that each stage of the project lifecycle was well-defined and independent of other stages. This style of project planning lent itself quite well to highly detailed and structured project task plans, often represented by impressive GANTT charts. These charts might forecast the development of fairly fine-grained tasks (perhaps corresponding even to a single program level function) several months into the future. As the project progressed linearly through each stage, it was straightforward to track performance against the plan.

Smalltalk project plans are different in that good Smalltalk pro-

* A good starting point is OBJECT-ORIENTED TECHNOLOGY: A MANAGER'S GUIDE by David Taylor, (Addison-Wesley, 1990.)

jects are often developed in a highly iterative fashion, and the project plans will reflect this iterative development method. Smalltalk plans tend to begin by focusing on developing core "kernel" classes, which model initial problem domain classes, and a simple user interface (UI) used to interact with those classes. Later stages of the project plan focus on refining and enhancing this core model, and extending its behavior. Although the project may repeat exactly the same stages as found in the waterfall model, much less time is spent in each stage, and the project iterates through the entire development cycle many times. Each complete trip through the development cycle may culminate in the delivery of a functional prototype, each of which builds on and adds behavior to its predecessors.

If your organization is new to object technology and Smalltalk, this style of development and project planning may lead to uncertainty among your customers and poorly formed expectations of delivery schedules and functionality.

Strategy. Smalltalk project managers should exercise caution when developing and presenting project plans to their customers, par-

**...to customers unfamiliar with
Smalltalk and GUI-building tools,
early prototypes may appear to be very
close to a finished product....their
expectations of delivery time may be
quite unrealistic.**

ticularly if the organization is new to Smalltalk. Make certain your customers understand Smalltalk's iterative development style and project plans that reflect this method.

Smalltalk excels at prototype development

As noted above, good Smalltalk projects are developed iteratively, and the result of each complete iteration through the development cycle will typically be an enhanced version of the system produced in the previous iteration. Smalltalk excels at this type of development because it contains both the necessary semantics to quickly and easily model a problem domain, and excellent GUI-building tools to rapidly develop graphical front-ends to the domain model.

However, problems can arise if your customers mistake the results of early iterations for deliverable products. Although early prototypes may give every exterior indication of stability and completeness, particularly when they are shown in a carefully scripted demonstration, it's very probable that much of the behavior is only stubbed out, and the system will require many further development iterations to refine it to production deliverable status. However, to customers unfamiliar with Smalltalk and GUI-building tools, early prototypes may appear to be very close to a finished

product. If this mistake occurs, their expectations of the current functional state of the system, and of delivery time for a finished system, may be quite unrealistic.

Strategy. Smalltalk project managers should be careful to communicate to their customers the difference between early prototypes and deliverable projects. Be especially cautious when demonstrating early prototypes to senior managers and end users who are unfamiliar with Smalltalk and GUI-building tools.

Smalltalk projects have different sticking points

Software development projects tend to have "sticking points"—places where, for some reason, the project slows down and progress lags, if only temporarily. Sometimes these sticking points can be traced to organizational issues such as political problems or resource availability. Other sticking points may be inherent in the technology used in the project.

Although Smalltalk is an extremely productive development environment, Smalltalk projects also have their own sticking points. Smalltalk project managers should understand and anticipate these potential slow points, and communicate them to their customers so they can factor these considerations into their overall project expectations. Three of the most common sticking points peculiar to Smalltalk projects are:

The learning curve. The Smalltalk language is deceptively simple in appearance. It has very few complex constructs, and is easy to learn and teach. However, although the language may be simple, the value of experienced Smalltalk developers is not in their understanding of the language but in their extensive knowledge of the class libraries. Their ability to quickly locate appropriate classes for extension and reuse, and their understanding of the intricate inheritance hierarchies found in all Smalltalk dialects make them much more valuable than newly minted Smalltalk developers. It can take an investment of several months to develop a working knowledge of a class library (which may contain hundreds or even thousands of classes) and longer still to understand the library in depth.

Smalltalk project managers should be careful not to underestimate the time required to train new developers, and to communicate this potential sticking point to their customers.

Persistence. Although Smalltalk has been described as an inherently persistent programming language,² progress on Smalltalk projects may slow when the need arises to make objects persistent and *shareable*—usually accomplished by storing them in a relational database (RDBMS). This is the much discussed *impedance mismatch* problem.[†] Unless you are using an object database (ODBMS), planning on storing your objects in the image, or developing an application that simply doesn't require persistent objects, you'll probably use a relational database to ensure that your objects survive across invocations of your Smalltalk image. This

[†] For a good explanation of the term *impedance mismatch*, see the April issue of BYTE.⁴

A SIGS Conference

February 21-24, 1995

Omni Park Central
New York, NY

SMALLTALK SOLUTIONS

95

Where All the Talk
is Smalltalk

Corporate Sponsors

- Easel Corp.
- IBM
- Knowledge Systems Corp.
- The Object People
- Servio

A Sampling of the In-Depth Schedule Includes:

- Writing High Performance Smalltalk Programs • Meta-Level Programming • Smalltalk: Past, Present, and Future • Metrics for Smalltalk • Prototyping in Smalltalk for Architecture, Design, Verification and Communication • Patterns • PANEL: Team Programming • Coding for Reuse • Building and Using Distributed Smalltalk • Smalltalk in the Large: Scaling Up Without Losing the Benefits • Implementing Custom Graphical Views in VisualWorks • Building and Using Distributed Smalltalk • Smalltalk and Teams: A Synergistic Solution ...and more!

Finally, a vendor-independent conference dedicated to all Smalltalk users. Focusing on the practical application of Smalltalk in its dialects, **Smalltalk Solutions '95** is an opportunity for the entire Smalltalk community to network, share innovative strategies and programming tips, and stay up-to-date on the latest tools and techniques.

Learn from the Smalltalk Experts

The educational program has been designed in conjunction with the Technical Conference Chair John Pugh, editor of *The Smalltalk Report*. The 4-day conference offers over 30 intensive classes ranging from beginner to advanced, all taught by experienced and well-respected Smalltalk experts.

You'll come away with new insights on language advances, usage tips, project management, analysis and design techniques, and insightful, practical applications. Specific class tracks focus on Team Programming, Analysis and Design, User Experiences/Case Studies, Technical Features, and Management Issues.

The latest Smalltalk products will be displayed in the **Smalltalk Solutions '95** Exhibit Hall, where you'll have a chance to demo the leading Smalltalk products, and receive an up-close, hands-on comparison. Don't miss this chance to see Smalltalk in action.

Smalltalk Solutions '95 is presented by SIGS Conferences, sponsor of over 7 conferences world-wide, including **Object Expo**, **Object Expo Europe**, and **C++ World**.



Circle 25 on Reader Service Card

For information on attending
Smalltalk Solutions '95, please
contact the SIGS Conferences Registrar:

PHONE: 212.242.7515

FAX: 212.242.7578

email: sigsconf@ix.netcom.com

**DON'T MISS
THIS UNIQUE
SMALLTALK EVENT!**

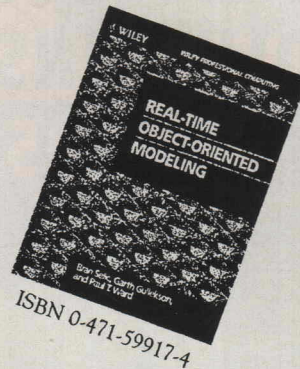
QUICK!

THINK OF A TOOL THAT GETS YOUR REAL-TIME PRODUCT TO MARKET FASTER

Having trouble? So were we. That's why we, as real-time developers, created the ObjecTime™ toolset to reduce time-to-market by automating key development activities. With ObjecTime™, to rapidly develop complex event-driven software you can:

- Create reusable components and component frameworks
- Execute and validate graphical requirements and design models throughout the development cycle
- Automatically generate complete production-quality code from design models. These are not just code "headers" or skeletons, but include the hardest parts to get right—complex component behavior and component interconnection and interaction. Target operating systems include VRTX, pSOS+, VxWorks, HP-RT, HP-UX, AIX, and Solaris.

While other companies are debating the merits of academic methods and traditional CASE diagramming tools, your real-time product is shipping!



ObjecTime supports the ROOM method authored by Bran Selic, Garth Gullekson, and Paul Ward (co-developer of the Ward-Mellor method)



OBJEC TIME™ ObjecTime Limited
1-800-567-TIME

Visit us at OOPSLA'94, Portland Ore., Oct 23-27

problem is inherent in all object languages, not just Smalltalk, and has the potential to slow your project while you develop the necessary interface to the RDBMS.³ Although the general problem is becoming much better understood and more tractable, and Smalltalk vendors are providing more support to solve this problem, it still should be factored into your project plan, and into your customers' expectations.

Paradigm. Smalltalk is a completely object-oriented language. In Smalltalk, *everything* is an object, from the simplest primitive data types (such as integers) to the most complex, behavior-rich application classes. When using Smalltalk, it's important that all members of the organization directly involved in the development project understand the central ideas behind object-oriented programming.

If your organization is transitioning to Smalltalk from a non-OOP language, it's likely that some personnel on initial projects will not yet be fully versant with object-oriented programming. They may try to graft Smalltalk onto legacy development frameworks with which they are more familiar rather than adopt Smalltalk's iterative development style. Left alone, they have the potential to dramatically slow overall progress on the project.

Smalltalk project managers, particularly on initial projects in an organization just adopting Smalltalk, must be careful to identify staff members who may not have sufficient training in object

development methods and Smalltalk, and provide them with the necessary training. The project manager should make certain that customers understand the potential setbacks that may occur as the organization makes the required initial investment in object technology training.

CONCLUSION

Using Smalltalk changes the general problem of managing customer expectations on software development projects. Smalltalk project managers who understand these changes, anticipate their impact, and actively manage their customer's expectations are more likely to complete their projects successfully. ■

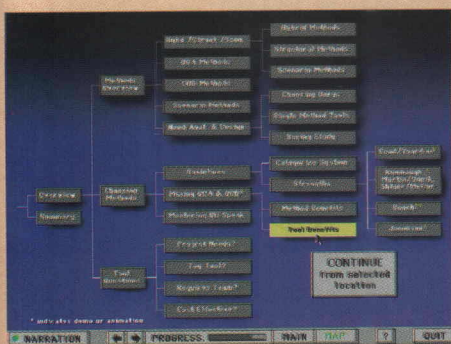
References

1. R Gabriel. Productivity: Is there a silver bullet? *JOURNAL OF OBJECT-ORIENTED PROGRAMMING* 7(1):89-92, 1994.
2. M Loomis. Making objects persistent, *JOURNAL OF OBJECT-ORIENTED PROGRAMMING* 6(6):25-28, 1993.
3. M Loomis. Hitting the relational wall, *JOURNAL OF OBJECT-ORIENTED PROGRAMMING* 6(8):56-59, 71, 1993.
4. C Mullins. The great debate, *BYTE*, April 1994.

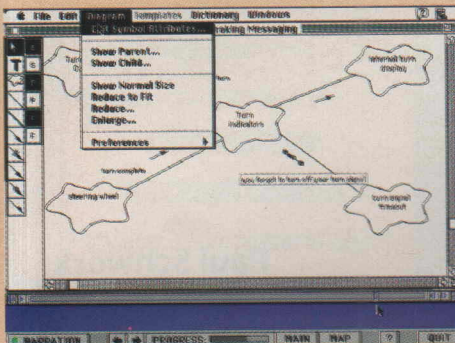
Douglas Camp, MS/MBA, is a senior member of the technical staff at Knowledge Systems Corporation.

THE NEW WAY TO LEARN O-O METHODS FAST!

Interactive Methodology Training on a CD-ROM



A navigational map is available at the touch of a button so you can quickly jump to any part of the presentation.



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.

HOW TO ORDER

Send order form to:
SIGS Books, Inc.
P.O. Box 99425
Collingsworth, NJ 08108-9970

FOR FASTER SERVICE:

Fax to: 609.858.2007
Phone: 212.242.7447

Circle 26 on Reader Service Card

**SIGS
BOOKS**

QuickTime and the
QuickTime Logo are
trademarks of Apple
Computer, Inc. used
under license.

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

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.

**15-Day
Money-Back
Guarantee**

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

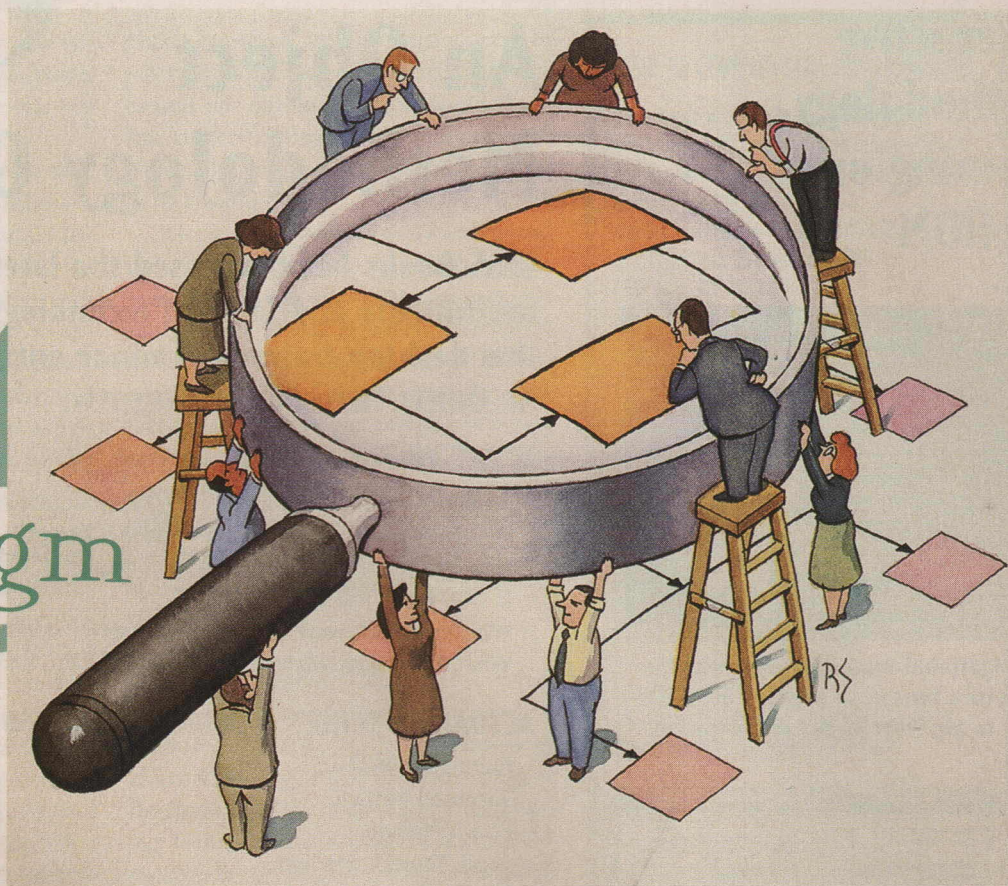
- ☐ Check Enclosed (Payable to SIGS BOOKS)
☐ Macintosh Version ☐ PC Version*
☐ Charge My: ☐ Visa ☐ Mastercard ☐ AMEX
CardNo _____ Exp _____
Signature _____

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

Name _____
Title _____
Company _____
Address _____
City _____
State _____ Zip _____
Country/Postal Code _____
Phone _____
Fax _____

About the paradigm



Rob Saunders/SIS

The paradigm provides a highly effective way of "seeing" the enterprise in a different light.

Paul Schwork

IN HIS BOOK, *THE 7 HABITS OF HIGHLY EFFECTIVE PEOPLE*,¹ Stephen Covey offers the following observation:

... The word *paradigm* comes from the Greek. It was originally a scientific term, and is more commonly used today to mean a model, theory, perception, assumption, or a frame of reference. In the more general sense, it's the way we "see" the world—not in terms of our visual sense of sight, but in terms of perceiving, understanding, interpreting.

For our purposes, a simple way to understand paradigms is to see them as maps. We all know that "the map is not the territory." A map is simply an explanation of certain aspects of the territory. That's exactly what a paradigm is. It is a theory, an explanation, or model of something else.

Covey is quoted here at length because his explanation of a paradigm, although given in a different context, is a highly effective way to view the object-oriented paradigm, especially as it applies to the process of enterprise analysis and modeling.

Viewed and used as a "frame of reference," a way of "seeing," a method of "understanding," a type of "map," the object-oriented

paradigm provides a different yet powerful means with which to examine and analyze the business environment we are targeting for automation. It is clearly different from the traditional linear-driven, process-focused, and back-end-centered (ie, produce the code) methodology of systems development used in the past. One could even argue that, because of the fact that the object-oriented paradigm introduces a different type of systems development approach, it serves a highly useful purpose in that it forces us to shake off some very old and very bad habits. One type of bad habit resulted from the attitude of "let's implement now since we can always finetune (ie, fix) later." It is powerful because the object-oriented paradigm focuses on the identification, definition, and classification of the objects that constitute the real substance of an enterprise. As such, the paradigm provides a highly effective way of "seeing" that enterprise in a different light, a light that enables one to uncover the truly material aspects of that environment.

Unfortunately, there are too many instances where the impact and effectiveness of the difference and power of the object-oriented paradigm are weakened when the technology is introduced into a real-world business environment. The technology itself (the lan-

guage selected, CASE tool chosen, etc.) often overshadows the *paradigm* (the frame of reference, way of seeing, process of understanding, method of mapping). Picking an object-oriented language or CASE tool first and then working backwards to find the business need that fits the given solution is no different from the way systems were developed and implemented when punch cards were still used. The solution was already there (usually COBOL, always a mainframe, probably CICS for the more sophisticated applications), and, in one way or another, the business need was force-fitted into the given solution. Moreover, the end users of the application, although stuck with the technological solution, were expected to be happy and appreciative. After all, the programmers worked nights and weekends to meet the project's end date.

Another unfortunate result of introducing the object-oriented paradigm into a business environment using a 1960s mentality is that the iterative process of validating the identification, definition, and classification of objects tends to be compromised considerably. The linear, cascading style of past systems development was driven by the need to produce code as soon as possible in order to implement the predetermined solution as rapidly as humanly possible. On the other hand, the iterative approach fostered by the object-oriented paradigm provides a very effective means for validating and verifying the objects that constitute the true composition of an enterprise. This validation and verification process, in turn, serves as a way to ensure that the system implemented is not only one of high quality but also one that satisfies the needs of the end user(s). In other words, if the model of the enterprise resulting from the application of the object-oriented paradigm is true, and the following design and programming faithfully follow that model, then the resulting automated system will reflect the reality of that enterprise (as opposed to the reality of a predetermined solution) and will provide what the end users really need to accomplish their day-to-day business tasks.

A final unfortunate situation is the tendency of the proponents of object-oriented technology (OT) to sell it as a tremendous cost savings because of reusability. Undoubtedly, reusability does offer the potential for significant cost reduction over the long run, assuming that the application of the paradigm to the enterprise environment results in a true, validated model. However, the cost savings argument is hard to sell for a couple of reasons. First, the introduction of object-oriented technology requires a significant learning curve, which translates into person-hours and actual expense dollars; and, today, many corporations, to improve the bottom line, are reluctant to expend the person-hours and dollars on another "new" type of technology. In this regard, Mr. Vorwerk's admonition, in his recent article "Toward a true DBMS," is a good one: "Let us talk about solutions rather than technology" (OBJECT MAGAZINE 3[5]). Secondly, what makes reusability as a cost savings an especially hard sell is the sins we, in this profession, have committed in the past: we have had the propensity to over-sell the benefits of new technology to our end users so that they would foot the bill, but we have not always delivered those benefits.

We told them, for example, that CICS is a sophisticated means of satisfying their business needs, but kind of forgot to mention that

this was true only insofar as their needs did in fact fit this particular solution. Then, we went back to them and said, no, a relational database is what is needed, but kind of forgot to mention that, at that time, a relational database was incapable of handling a high volume, online transaction application. Then, we went back to them and said, no, a distributed processing environment is what you need, but kind of forgot to mention the cost and reliability of a micro-computer/minicomputer/mainframe/fileserver/LAN type of architecture. Is it any wonder, then, that when we try to sell reusability as a cost savings feature, the end user reacts with skepticism?

The point here is let's not try to sell object-oriented technology on the basis of what it can do for an enterprise in terms of dollar savings. Based on our prior track record, the door to that argument is probably closed. Instead, let's try selling the paradigm aspect of the technology as a means for the end users of automated systems to have a better frame of reference, a better way of seeing, a better process for understanding, a better form of mapping their environment. So what if the application of this paradigm to the business environment does not lead to code (and who says it has to?); just maybe the business enterprise will be the better for it. And maybe, once sufficiently instructed, the end users on their own can, with the application of the object-oriented paradigm, better articulate the way in which their enterprise actually functions. Should the implementation of OT result in real costs savings through reusability, then "surprise"! We can then say truthfully, "look at the value-added service we have provided."

One final point is the need to address the people, cultural, and political implications of the introduction of the object-oriented paradigm and technology into an enterprise. In terms of people, the trauma of facing fairly radical change, both in thought processes and habits, must be softened through training, understanding, and intelligent goal setting. Having management read and apply McGregor's *THE HUMAN SIDE OF ENTERPRISE* or even a little bit of Maslow might possibly help. In terms of the cultural and political implications, an honest and objective assessment must be made regarding whether receptiveness on the part of the management to the paradigm/technology is worth the effort to try and make the hard sell. As Mr. Loveland points out, in his article "The growth of objects: A natural evolution": "In my experience it is less frequently the programmer and more so the management structure that disallows proper adoption circumstances" (OBJECT MAGAZINE 3[5]). In situations such as this it might be better to retreat in order to return and fight another day.

As an aside, it is interesting, but obviously coincidental, that Covey mentions that the word *paradigm* is of Greek origin. The gentleman mentioned in "Collected thoughts on Aristotle, objects, and business" (OBJECT MAGAZINE 3[6]) was also Greek. ■

References

1. S Covey. *THE 7 HABITS OF HIGHLY EFFECTIVE PEOPLE*, Simon & Schuster, New York, 1989, p 23.

Paul Schwork is president of Object Oriented Preparation Services, Robbinsville, NJ.

IF YOU MISSED Object Expo Europe '94...

...you missed over 2,500 of your colleagues at Europe's most respected OT conference and exhibition. The latest breakthroughs and advances in object technology were presented during this powerful, week-long event. **Object Expo Europe 1994** included 63 technical classes taught by a faculty of experts focusing on such topics as designing with patterns, distributed OT, and O-O business engineering. The exhibit hall was filled with over 50 leading vendors, and special events included keynote speeches presented by Rational Software Corp., Taligent, Microsoft and IBM. Throughout the week were Product Education Sessions conducted by top OT companies, and the debut of the 'sold-out' Executive Symposium.

Attendance rose over 50% compared to 1993! Programmers, developers, analysts, managers and administrators from 14 countries took the opportunity to attend the most up-to-date technical classes and tutorials presented by the masters and innovators in the industry. By offering beginner through advanced sessions, the curriculum was designed to help individuals gear the program to their own needs. And with expert speakers such as Tom Atwood, Grady Booch, Peter Coad, Ian Graham, Ivar Jacobson, Andrew Koenig, Stanley Lippman and Scott Meyers, it's no wonder these technical courses received such rave reviews. The OpenStep/NEXTSTEP Developers Day was overflowing with delegates eager to harness this award-winning object development and deployment environment.

Top vendors of the latest products and services in object technology were on display at the largest OT exhibition in Europe. Hands-on demonstrations of newly developed products allowing the user to sample product features, highlighted this showcase of events. Through product announcements and presentations, those making serious purchasing decisions had all their questions answered on the show floor.

A popular addition to the program this year was the Executive Symposium. A capacity of more than 90 pre-registrants attended this truly unique forum focusing solely on senior-level management. Sponsored by KPMG Management Consulting, companies such as British Telecom, Reuters, Thomas Cook, Ford of Europe, and Rolls Royce Plc, revealed how they benefit from the use of OT. Project strategies, implementation and results were discussed, proving to the audience of senior level executives with 'real' information that object technology is 'mature' for mission critical applications.

Object Expo Europe was the event of the year for anyone involved in the OT industry. By offering an intense conference program, an impressive array of exhibiting companies, seasoned with a variety of educational special events designed to help successfully infuse object technology into current software projects, this was truly a valuable experience for all those who attended.

So if you missed '94...

DON'T MISS Object E X P O europe '95

[When] 25-29 September, 1995

[Where] QEII Conference Centre, London

Be sure to receive information early - contact Jane Major at +44 306 631 331

Circle 28 on Reader Service Card



"A good mix of educational and commercial issues - a good value for your money."

A. SALAZAR, UNIVERSITY OF MANCHESTER



"The best UK show! Very well organized and professionally done - an exciting, comfortable environment."

HOWARD RICKETTS, SOFTCASE CONSULTING



"The quality and response of people is very good here. An equal balance of time is given to both the conference and exhibits."

IAN PRICE, HITACHI EUROPE



"It was our first year at Object Expo Europe and we found the audience of programmers we needed here."

DAN GILL, WROX PRESS, LTD.



"Well organized - good quality speakers and exhibits."

MICHAEL CHRISTIANSSON,
KOCKUMS COMPUTER SYSTEMS, AB



CLIENTSERVER

APPLICATIONS

Tour

Register and receive
a FREE copy of Borland's
new Delphi product—
a \$495 value.

January 12-13 NEW YORK
January 17-18 WASHINGTON, DC
January 19-20 CHICAGO
January 23-24 SAN FRANCISCO

**Two Days of Training—
A New Era of Information
Management**

Schedule of Events

DAY I			DAY II		
8:00-9:00	Registration/Breakfast		8:00-9:30	Breakfast	
9:00-9:30	Welcome Address		9:45-11:45	Morning Classes: C/S in the Real World	
9:45-11:45	Morning Classes: C/S Overview			TECHNICAL TRACK	MANAGEMENT TRACK
	What is C/S: The Technical Perspective	What is C/S: The Management Perspective			
11:45-1:45	Luncheon & Borland Presentation		11:45-1:45	Luncheon & Borland Presentation	
2:00-4:00	Afternoon Classes: Making the Change to Client/Server		2:00-4:00	Afternoon Classes: Strategic Considerations in the Transition to C/S	
	TECHNICAL TRACK	MANAGEMENT TRACK	TECHNICAL TRACK	MANAGEMENT TRACK	
	Design & Development of C/S Apps.	Managing C/S Development: Myth vs. Reality			Merging O-O and C/S: The Technical Transition
4:00-5:00	Q&A		4:00-5:00	Q&A	

The **Client/Server Applications Tour** is a 2-day training and education event making stops in 4 cities across the U.S. All classes are objective and product neutral. The tour is designed to provide attendees with practical guidelines to assess and develop realistic goals for their own client/server projects.

Divided into a technical and management track, this seminar offers practical solutions for implementing a client/server architecture within corporate environments. Whether you are a software manager or part of the technical development team, you'll walk away from these two days of intense training with a complete framework for implementing and managing client/server solutions which meet your particular business needs.

Who Should Attend

Anyone exploring the corporate benefits of client/server technology, including:

- Programmers
- Developers
- Designers
- Software Engineers
- System Analysts
- Project Managers
- Project Leaders
- CIOs
- MIS

Speakers

Jim Stikeleather



Charles Bowman



Management Track Speaker Jim Stikeleather is the Product Review Editor and C/S Workbench Columnist for *Client/Server Developer* Magazine. He is currently the Technical Partner of the Technical Resource Connection, a client/server and O-O technology implementation firm.

Technical Track Speaker Charles Bowman is an independent consultant, author and lecturer specializing in GUI, relational and object-oriented technologies. Mr. Bowman is a regular columnist for *Client/Server Developer* magazine and *The X Journal*.

Presented by:



Co-sponsored by:

Borland
CLIENTSERVER
DEVELOPER

For more information and a detailed brochure contact:

Phone: 212.242.7515 Fax: 212.242.7578

Mail: SIGS Conferences, 71 West 23rd Street, NY, NY 10010

Lessons from the Experts

Thoughts drawn from the August, 1994 Conference on Object-Oriented Software Development, held at St. Thomas University, St. Paul, Minnesota.

Richard W Koontz

Preface: Should you care about object engineering? This is one of those Rubik's cube kind of things. It is tempting to pick it up and "make it happen." But it isn't simple. A company is either going to get all the pieces going the right way, or it won't. And if it doesn't, it can try again and again without succeeding. The solution is not fundamentally obvious.

But again, should we care?

It is an interlocking puzzle. But it is anything but just a child's toy, and seems to me to be more of a genetic engineering kind of puzzle. I accept (and I'm a cynic) that, from a business standpoint, *ignoring this puzzle* will offer the same kind of future as sticking with the IBM 360 computer series. You and your company will be maintaining an increasingly rare skillset with decreasing commercial value.

But more than that, professional use of the entire pantheon of object-oriented concepts and tools will be just an entry-level requirement for corporations in the coming decade. It isn't a matter of choosing the right technology for the right job. It looks like it is simply a matter of choosing to stay with the computer industry and with world industry, or being left in the 1800s of business practices.

Please understand my normal bias against new technology. I will almost always choose an old, outdated but reliable tool over some new gadget. I love and respect COBOL as much as I do Assembler and C. I trust PowerBuilder, VBASIC, et al. Those are all tools, and we have all mastered their use, along with dozens of other tools that were appropriate. *But they are all the same kind of tool.* Even with excellent debugging aids and fast compilers, they are all hand tools suitable for carving wood, cutting leather, hewing stone, and digging trenches.

What is happening here is *not* a revolution. The ideas really aren't new and unique. It appears to be more a simple stepping out of the era of handicrafts, and into the use of heavily automated and abstracted *n*th-generation development technologies. Object orientation is not a technology—it is a *group of technologies that spans human behavior, management practice, and computerized solution development.* Taken one by one, they are no more than the same kind of hand tools we've used before. Take them together, though, and things happen that were unfeasible before. There are

things we have conceived of for years that will now happen, very rapidly, and with greater benefit than we expect.

To paraphrase Dr Wells, the eloquent leader of IBM's O-O team, who also doubles as the drumbeater to beat all drumbeaters: "In the previous 10 to 15 years, American business cleaned up its systems, but did little significant new information systems work. This has changed. American business is moving rapidly, driven by world competition, and is re-engineering itself and its information systems. Among the firms who have committed to do business in the new ways, there is consensus. You either keep your systems pretty much the same, and do business the way it used to be done—or you meet the competitive challenges, and start doing business the way it is going to be done in the future."

This is a Rubik's cube kind of thing, but it affects all of us, whether we like it or not. We can pick it up and put it down, but sooner or later we must master it. Or be content that our good old reliable System/360s are still doing exactly what they were intended to do back when they were first designed. (And there is nothing wrong with a system doing just what it was supposed to do, is there?)

The consensus is that competitive factors, not technological ones, will determine when each of us makes the jump. Or, as the techno-nerds like to call it, the "Paradigm Shift."

Sort of reminds one of the jump to hyperspace, what?

The heart of the matter: Is this a revolutionary technology that will solve our problems?

Emphatically NO.

It is not revolutionary.

It is not one technology.

It will not solve our problems. It may well create more than we imagine are possible.

It is just one more incremental step in a long development process. Almost everything involved has been thought of before, and used by most of us at some time in our careers. But the hardware technology that supports it is new. The developers are using that hardware technology to give us tools that encompass all those old ideas. The hardware is adapting to the tools, and being redefined to support the tools. And the behavioralist developers are finding new integrated approaches to finally bridge the gap between the users and the developers.

And the tools are being used to build tools, which are building

Richard W Koontz is the President of Object Technologies Group in Eagan, MN.

tools. And, more wonderful to me: the users get excited that they finally understand what the techno-nerds are saying, sometimes even better than the techno-crats themselves.

Is the situation getting out of hand when users understand developers?

Again, I have to repeat that the object world is a coherent *collection* of technologies, all interdependent. Analysis tools and methods. Design tools and methods. Management tools and methods. Monitoring and feedback tools and methods. Conceptual frameworks. Visualizations. Realizations. The "Ahah" Principle in 3D and in living color.

If you try to pick one of these and use it to the exclusion of the rest, you will gain some benefits. If you try an object language and forgo the supporting technologies and mental frameworks, you stand an excellent chance of failing. And if you want an opportunity to be immortalized in legend, try a major mission-critical project for your first baby step.

As to problems, well, we just have to look back over our shoulder. Every major solution I've seen has brought along its host of unexpected problems. And promised cost savings often vanished with the "Turnpike Effect" of new systems. Whether unforeseen events are challenges or problems is semantics. Sears Roebuck, a massive user of computers, once did a corporate study and concluded that, in retrospect, they had not saved any money by computerizing. However, they concluded they had no choice. It was computerize or disappear as a business.

In summary, the experts indicated that if you choose to enter the object-oriented world, you must step into it carefully and firmly. The benefits are there, both in the near and long term. With the sole exception of code reuse (some people's holy grail), the realities match the serious promises.

And the future will be an either/or proposition: either go object oriented, or stay with the old and continue the current methods. They won't go away, but they themselves will either adapt and be cobbled together with object interfaces, or they will simply continue providing good, solid, dependable stepwise code development.

If we go object-oriented, can I pretty much do it with any of the tools billing themselves as object-oriented? By accident or intent, I did not mention a computer language as one of the key technologies in the previous paragraphs. I'm anchored in the old view that a language is just a language is just a language, and the easiest or fastest one to use is often the best one. Being anchored means I don't move particularly fast in changing habits. Fortunately for my company, I keep these views to myself, and have not stopped my technical people from keeping up with the leaders.

The speakers, in general, tried not to slant their discussion for or against any one implementation language. One specifically stated that we'll simply push a button in the future and the language will be selected for us, and the code will be created as we drink our coffee. Sure.

I enjoyed the sharp disagreement among the audience and speakers. There is a strong C++ contingent that views the rest of the mortal world as, shall we say, just a shade less eligible for godhood. I've heard this view expressed on the Internet fairly frequently, when-

ever COBOL is referred to as something like a dead language suitable only for brain-dead zombies.

An opposing view, perhaps held mainly for its ability to get a strong reaction, was that the most eligible object developers come from COBOL backgrounds, and a long and in-depth knowledge of C++ is a warning flag for a manager.

Regardless of a person's religious beliefs, there was agreement on one thing: The coherent use of all the technologies is demanding on an implementation language. It requires more than *any* of them offer today.

The computer language most often chosen by the Fortune 500 to pull all of this together is Smalltalk, according to Dr Wells. This is often difficult for a hand-tool user like me to accept, but I do accept it. It is the only language I know that is practical from a total business viewpoint. I would do throw-away (cringe?) systems in PowerBuilder, COBOL, C, or VBasic. But if I intended to stay in a business after I had thrown away a few systems, and if I intended to keep the business competitive, and keep changing to fit the market, then I too would pick Smalltalk. (As an aside, we *have* chosen Smalltalk for all our major projects.)

For your reference, some cost estimates were thrown around in the conference. They expect a first project to be about 30% more expensive than a solution using a "get it done fast" tool. The second project will be about the same cost, and the third and subsequent projects will be inexpensive relative to other implementations.

Dr Wells stated that subsequent projects would be "orders of magnitude" less expensive, and gave some examples to demonstrate what he meant. In a nutshell, a small team of 3 to 5 people could do as much as a team of 10 to 12, and could do it faster.

Aside from C++ and Smalltalk, there was very little discussion of the other tools such as COBOL with object-oriented extensions, or VBasic, or PowerBuilder, etc. This is partly because no vendors of those products were speakers, and partly because they are object-oriented with limitations. Or because a few have such limited commercial support that they are unlikely to be selected by the market.

Other object tools will be developed. We will surely see variations of C++ and Smalltalk emerge. And the market, creative as ever, will toss us new solutions to new problems.

But for now, I felt the conference supported the selection of Smalltalk as the language of choice.

Last item: Team size, supporting hardware, and the mentoring equation. This is a knowledge-intensive path. And also hardware-intensive. Small teams are the preferred way to go—teams of 3 to 5 people, heavily supported by team development library management systems, working on richly outfitted workstations, and working closely with users.

Your development workstations should, at minimum, be 66 Mhz 486DX2 workstations with 24 Mb of memory. The library management system will prove its value within three weeks after coding starts.

In the first project, pick an important but non-critical system. Staff it with your own management and team leaders, but go extremely heavy on mentors. For every one of your people, use one outside expert developer who has done it again and again. Take your time; and take your normal time budget and increase it by 30%.

Take On Responsibilities

Many developers are discovering Responsibility Driven Design (RDD) as a powerful means of identifying the required classes and behaviors of an object-oriented system.

Introducing...

Object
System/CRC™

Used alone, Object System/CRC™ supports RDD activities described by Wirfs-Brock et al., such as creating CRC cards, building subsystems and defining collaborations.

and...

Object System/CRC™ links smoothly to Rational Rose® for additional design and construction detail.

Object System/CRC™

- Defines responsibilities and relationships of key abstractions
- Is convenient
 - easy navigation between classes
 - easy retrieval of classes
- Integrates with Rational Rose®
- Facilitates group analysis & design development
- Comes with comprehensive on-line tutorial

FREE FULL-FEATURED DEMO DISK!

Contact Palladio today to order your FREE full-featured demonstration diskette.

800 437-0019 (US) • 414 789-5253

FAX 414 789-5191

EMAIL info@palladio.com



16535 West Bluemound Road • Suite 360 • Brookfield, WI 53005

Circle 30 on Reader Service Card

STATEMENT OF OWNERSHIP

MANAGEMENT AND CIRCULATION
(Required by U.S.C. 3685)

Object Magazine (1B) Publication No. 008-612 (2) Date of Filing 10/3/94 (3) Frequency of Issue: Monthly except March/April, July/August, November/December (3A) Number of Issues Published Annually: 9 (3B) Annual Subscription Price \$39 (4) Complete Mailing Address of Known Office of Publication: 71 West 23rd, New York, NY 10010 (5) Complete Mailing Address of the Headquarters of the General Business Offices of the Publisher: 71 West 23rd Street, New York, NY 10010; Richard P. Friedman, Publisher, 71 West 23rd Street, New York, NY 10010; Marie A. Lenzi, Editor, 71 West 23rd Street, New York, NY 10010 (7) Owner: Richard P. Friedman, 71 West 23rd, New York, NY 10010, Margherita Monck, 71 West 23rd Street, New York, NY 10010, Richard Wiener, 71 West 23rd, New York, NY 10010: Average No. Copies Each Issue During Preceding 12 Months: (A) Total No. of Copies: 41952 (B) Paid and/or Requested Circulation: (1) Sales through dealers and carrier, street vendors and counter sales: 6128 (2) Mail Subscriptions: 20725 (C) Total Paid and/or Requested Circulation, 26853 (D) Free Distribution by Mail, Carrier, or Other Means, Samples, Complimentary, Other Free Copies: 7872; Free Distribution Outside Mail: 2728 (E) Total Distribution: 37453 (F) Copies Not Distributed: (1) Office use, left over, unaccounted, spoiled after printing: 900 (2) Returns from News Agents: 3599 (G) TOTAL: 41952 Actual No. Copies of Single Issue Published Nearest to Filing Date: (A) Total No. Copies 43532 (B) Paid and/or Requested Circulation: (1) Sales through dealers and carriers, street vendors and counter sales: 6175 (2) Mail Subscriptions: 23092 (C) Total Paid and/or Requested Circulation: 29267 (D) Free Distribution by Mail, Carrier, or Other Means, Samples, Complimentary and Other Free Copies: 7588 Free Distribution Outside Mail: 2300 (E) Total Distribution: 39155 (F) Copies Not Distributed: (1) Office use, left over, unaccounted, spoiled after printing: 750 (2) Returns from News Agents: 3627 (G) TOTAL: 43532 I certify that the statements made by me above are correct and complete: Richard P. Friedman, Owner and Publisher

I repeat: *one expert for every novice*. Experience indicates that full immersion is the *least* expensive and most successful route. Full immersion requires a very knowledge-rich environment.

This is hard to justify in some companies. If you can't obtain it all, consider not doing it. Bide your time, and do it later when you *can* obtain the resources to succeed. Don't shortchange yourself. This is not the time "to leap the chasm in two leaps."

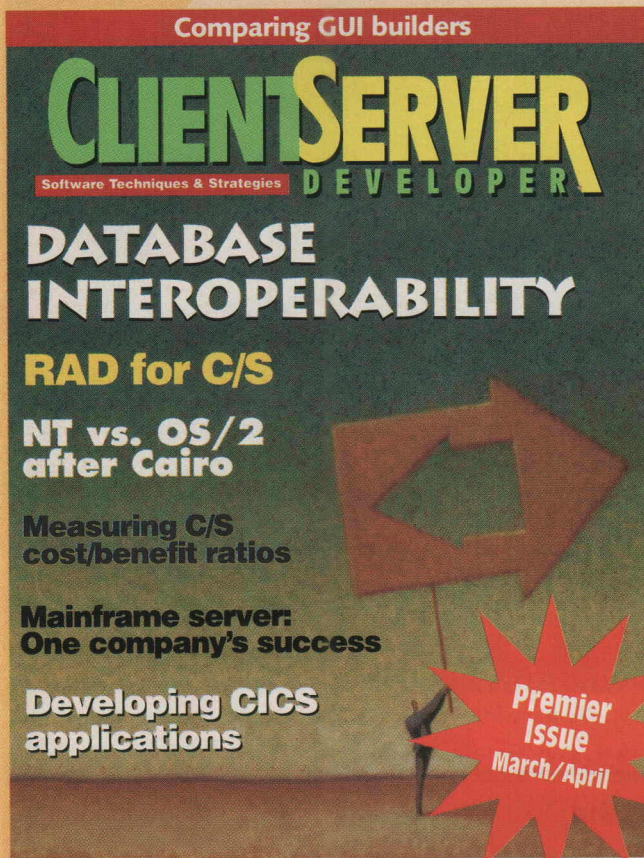
In the second project, scale the ratio back to one expert for every two novices. That is, once management and team leaders know where and why they're doing what they're doing, and your users have been through the drill and know what is expected of them, then and only then can you successfully scale back on outside knowledge.

According to the speakers, the surefire recipe for failure is a standard management approach using the waterfall analyze/design/code/test, using a standard roomful of developers, led by skilled team leaders and analysts, all coming from a data-oriented procedural background. You are guaranteed to create a standard, procedural system that happens to be written in an object language, with none of the benefits but all the pitfalls of object orientation. You will find it doesn't maintain well, doesn't compare well on management yardsticks, and ends up being a waste of money, time, and effort.

The third project, the experts promise, will surprise you with its low cost and rapid completion using in-house talent.

It sounds like every other management fad we've seen come and go. But I think we're going to see this one stick around for a couple of decades. And by then, it will have metamorphosed beyond recognition. ■

A Revolutionary Information Server



**Don't miss a single issue...
Subscribe today!**

Circle 31 on Reader Service Card

CLIENT/SERVER DEVELOPER is a unique new magazine devoted to every aspect of C/S software development. Written for—and by—sophisticated developers and technical managers. **CLIENT/SERVER DEVELOPER** focuses on the software engineering topics required to build successful client/server systems.

By reading CLIENT/SERVER DEVELOPER you will:

- Learn the software strategies companies are devising to take advantage of C/S technology
- Objectively compare the leading languages and methods to determine which is best for your C/S architecture
- Examine the latest advances in DB management systems
- Discover the realistic productivity gains from converting to a C/S environment

**If you're building C/S systems,
you should be reading CLIENT/SERVER DEVELOPER.**

Upcoming, insightful articles include:

- Performance tuning in a mixed C/S database environment
 - Optimizing database sorting across distributed servers
 - What you should know about voice/data servers
 - Operating environments: their benefits & drawbacks
 - Project management tools that work
 - How to accurately measure soft costs and soft benefits in C/S implementations
 - Data modeling techniques you should know
 - C/S debugging tips
 - C/S metrics and testing techniques
- Plus: Candid Product Reviews, Product News, the Client/Server Insider Interview...And more!*

PUBLISHED BY  **SIGS**
PUBLICATIONS

CLIENT/SERVER
DEVELOPER

Charter Subscription Offer

**SAVE
OVER
13%**

☐ **YES!** Enter my subscription to **CLIENT/SERVER DEVELOPER** at the special Charter Subscription rate checked below.

☐ **1 Year (6 issues) ~~\$29~~ NOW \$25**
SAVE OVER 13% OFF THE BASIC SUBSCRIPTION RATE!

☐ **2 Year (12 issues) ~~\$58~~ NOW \$50**
EXTEND CHARTER SAVINGS AN ADDITIONAL YEAR!

METHOD OF PAYMENT

- ☐ Check Enclosed
☐ Bill Me
☐ Charge My: ☐ Visa ☐ Mastercard ☐ Amex

Card #

Exp. Date

Signature

HCSD4

For faster service, fax to: 212.242.7574
Mail to: 71 W. 23rd St., New York, NY 10010

First Name _____ Last Name _____

Title _____

Company _____

Address _____

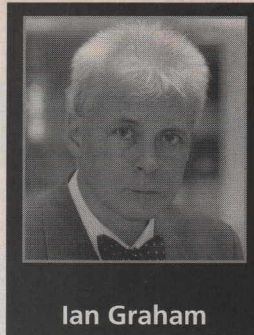
City _____ State/Province _____ Zip/Postal Code _____

Country _____

Business Phone _____ Business Fax _____

Non-U.S. orders must be prepaid. Add \$25 for Canadian and Mexican orders. Add \$40 for all non-North American orders. Checks must be in U.S. dollars and drawn on a U.S. bank.

You was my object... but I done you wrong



Ian Graham

IN MY LAST CONTRIBUTION to this column I promised to begin discussing how a reuse program should start at the specification level. This is still my intention but, on thinking about the problems associated with reuse programs, my attention was drawn to the fact that most companies are failing miserably at achieving significant levels of reuse and indeed at migrating to object technology in general. Thus, in this issue I want to share some of my thoughts and experiences relating to the reasons projects fail before returning to the reuse issue in my next column.

Management generally wants to migrate to OT for one or more of the following reasons:

1. they are moving to distributed platforms as part of a right-sizing exercise;
2. they need to provide more sophisticated interfaces (typically graphical ones) or manage more complex applications;
3. they have heard that object technology can provide significant productivity gains due to reuse and reduced maintenance costs;
4. they think that systems can be built faster with object-oriented programming languages;
5. OT is new and they want to explore it, hoping to position themselves for the next step in application development.

Often there is confusion about the relative weightings given to these objectives. Unfortunately, the fifth objective seems to be rather more important than it should be in

many instances. As Mae West once put it: "Between two evils, I always pick the one I never tried before." Perhaps this attitude is shared by many IT managers. Further, many managers have not fully realized the extent of the cultural and organizational changes required to implement object technology successfully. For these, and possibly other, reasons the stories of failure are becoming increasingly common, although they are seldom published.

My organization has been successful in its migration strategy so far. We have in-

**In my view, the
whole business process of
software development
itself needs a radical
rethinking.**

creased our productivity by 50% annually through the use of object technology and intend to continue to do so. We are totally committed to object technology and to a distributed client/server architecture. We are not committed to any particular platform or language, but are currently extensive users of *inter alia* NEXTSTEP, Objective-C, and C++. We have developed our own, novel approach to reuse, migration, object-oriented rapid application development, requirements capture, and object-oriented analysis that has been successful in terms of quality and time to market. We are rated by independent auditors in the top 10% of IT shops in terms of productivity and quality and should soon be in the top 1%. Information technology, and especially object technology, has been a key part of our

success in the market, where we have achieved very high ratings in EUROMONEY and other surveys and been able to bring innovative products to market in a very short time. This success needs to be built upon and is not guaranteed to continue automatically. It is nevertheless tangible and real. We know that we have to keep on working at it but we also build on a firm base.

Other organizations have not been so fortunate. Before I worked for the Swiss Bank Corporation I worked for various consulting organizations. During this period, the following experiences brought home to me that some of the commonest reasons for failure have little to do with technology. One afternoon some of my colleagues returned from a meeting with a very large commercial organization, at which this client had hinted at a need to train over 100 developers in object-oriented techniques. This was potentially a very large consultancy and training contract. As the resident expert, I was asked to call them and clarify the requirement. The time was about 5:15 pm when I began to telephone the list of names I had been given. I started with the project manager. He had gone home. In fact everyone on the list had gone home except for a very knowledgeable member of the technical staff right at the bottom of the list. We chatted for a while and I began to drop in searching questions such as: "What benefits do you expect to achieve as a result of this initiative?"; "What business problem is being addressed?"; etc. The picture began to emerge that some pilot systems were being contemplated to build the skills base pending work on anything business critical or visible. Finally I asked: "What senior management support is behind this project?" The answer this time was unequivocal: "We

Ian Graham is an object technology specialist with Swiss Bank Corporation in London, UK.

NO Hype

We're not exaggerating! This is really IT. After all, when you invest valuable time you want solid, usable information — expertly presented. You'll hear

over 150 hours of *commercial-free education*, including eighteen full and half day tutorials, seven in-depth conferences, covering the key issues in Client/Server Application Development,

Join 25,000 Colleagues in New York —

at the most important DB/EXPO ever.

Over 150 hours of commercial-free education!

Data Warehousing, Parallel Processing, Interoperability, Enterprise and Workgroup Computing, plus Interactive Information, Object and Evolving Technologies.

And, on our show floor there's even more: 500 exhibitors, a full menu of special events, *plus* thought provoking keynotes from the industry leaders who are shaping your professional future.

Register now and save \$\$\$!

Want Proof? Call: 1-800-2DB-EXPO

The proof is in the program! Call today for your copy and see for yourself. Hurry! Conferences are filling up fast.

Yes, Give me the Proof!

- ☐ Send me my 36 page DB/EXPO '94 Conference Brochure
- ☐ _____ Complimentary Exhibit Passes
- ☐ Information on becoming an exhibitor

6411NY

24 HR FAX FORM (415) 966-8934

Name: _____ Title: _____
E-Mail Address: _____
Company: _____
Street: _____ Mail Stop: _____
City: _____ State: _____ Zip: _____
Telephone: _____ Fax: _____

Exposition • 6-8 December 1994
Jacob K. Javits Convention Center
Conference • 5-9 December 1994


BLENHEIM®



**The World's Largest Database, Client/Server,
& Information Technology Exposition & Conference**

Blenheim NDI, Inc. is a subsidiary of Bruno Blenheim Inc.
444 Castro Street - Suite 1101 - Mountain View - California 94041
Tel: (415) 966-8440 - Fax: (415) 966-8934

Objects Everywhere!

Why settle for hybrid implementations when you can have the real thing? JumpStart is the leading provider of solutions and training programs for pure object systems using Smalltalk and the GemStone^(tm) ODBMS. We also specialize in deploying IBM Smalltalk^(tm) and VisualAge^(tm) applications.

Ask about our Corporate Educators Program.



919.460.1583

**Manufacturing
Process Control
Network Management
Pharmaceutical
Client-Server IS Systems**

Certified Service Partners with:



Copyright 1994, © JumpStart Systems, Inc.

Circle 33 on Reader Service Card

- The idea that there is a simple formula (eg, learn Smalltalk) that, if applied correctly and consistently, will lead to success.
- The depth of the reporting hierarchy or management structure.
- The lack of vision of most IT managers.
- The lack of the courage to take on really high profile and visible applications early on in migration.
- The view that object-oriented programming alone provides reuse.

Successful organizations seem to be characterized by strong and imaginative leadership and a high-risk/high-reward approach. Flat management structures and motivated, empowered developers are another key indicator. The languages used do not seem to matter although the development methods do have some impact. Skilled (and therefore expensive, at present) developers are a prerequisite.

In a survey of European C++ projects conducted by Michael Chua and Nigel Backhurst (and so far unpublished) it was found that the seven most frequent contributory factors leading to failure were, in order of descending frequency:

- poor class library control
- problems with the development method
- management underestimation of costs and time
- poor version control
- lack of specification
- skills shortages

haven't really told anyone in the upper management layers. If we did, they'd probably stop us from doing it." We didn't get the consultancy and I, for one, was relieved to hear it.

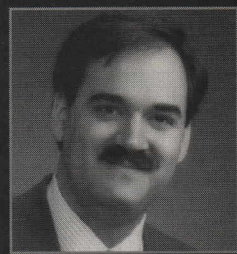
About 18 months previous to this incident, I had called on one of this organization's main competitors. They too were starting some small pilot projects and seemed to have a knowledgeable and enthusiastic team. They had decided that a single supplier of object-oriented services was what was needed. I thought, at that stage, that such an organization—covering languages, tools, lifecycle, training, and methods—probably didn't exist and I certainly couldn't offer that in all honesty. This is beginning to look like a history of my failures as a salesman because, once again, we didn't get the business. However, after about a year I was invited to bid again. The "single supplier" it seemed had not been able to wear his underpants on the outside of his tights after all and a new one-stop shop was being sought. Once again we didn't offer the right kind of solution and failed. Latterly, I hear rumours that this very organization is searching again for consultancy support for their project and I have not seen any reports of successful, mission-critical applications from either one of them. One is tempted to conclude that most large organizations are stumbling along badly in their attempts to move to objects.

What are these companies doing wrong? In both cases discussed above, the developers were knowledgeable about OT, skilled, enthusiastic, and suitably cautious. It seems to me that the success or failure of object-oriented projects is related to the following issues:

These seem to be technical reasons but behind them all lurk deep management and infrastructural issues. Reuse teams and library management need serious funding. Object-oriented programming needs a method that takes projects from requirements capture and estimating through maintenance, as few of the current object methods do. The confidence of users must be won. Training and education must be paid for. Delays due to long chains of command must be overcome. In my view, the whole business process of software development itself needs a radical rethinking. The organizations that are succeeding are those that measure their results, get close to their users, test everything as it is produced rather than after development, emphasize quality, and shorten time-to-market. They are not bound by existing "structured" methods or rigid development "phases," nor do they zealously document everything unless the documentation is sure to be useful. Object technology is really a quite secondary enabling technology behind these issues. It may contribute 80% to the technical aims of a project but it probably has less than 5% influence on the likelihood of success.

Above all, the main fault lies with management, and senior management at that. Asking small R&D teams to coordinate such an important task as the Great Leap Forward to object technology is reminiscent of asking every Chinese peasant to make steel in his backyard to industrialize the nation. Organizations migrating to object technology should make sure that the leaders of the organization are 100% committed to OT and understand both its potential and its risks clearly. If they are not, and cannot be converted (or replaced), then it is probably best to give up or go elsewhere. ■

Defining interfaces



Richard Soley

If you've ever read anything about client/server, distributed object computing, remote procedure calls, or other schemes for simplifying network services programming, you have read something about interface definition languages (IDLs). And you probably have decided that there's got to be something wrong with a simplifying scheme that requires you to learn another language. In this article I will convince you of two things: (1) IDLs are a great idea (how did we live without 'em?); and (2) the learning curve for IDLs is remarkably short (days, in fact).

First, let me explode a myth you probably hold dear: IDLs have nothing to do with distribution, client/server, or networking per se. There are perfectly good occasions to use an IDL on a single computer, or even within a single process or address space. So, what is an IDL?

Whether the "D" is for "description" or "definition," an IDL is used to *specify public interfaces of services*. A basic IDL will include at least the ability to define functions or procedures of service interfaces, and the parameters of those functions. For example, we might define a bank account interface as follows (don't ask me why everybody uses bank account examples, I'm certainly not going to buck the trend):

```
; Define datatype for transaction amounts to
; be fixed point, 10 digits before and 2 digits
; after the decimal point.
TYPE amount IS fixed_point (10, 2)
```

```
; Define datatype for account identifiers as a
; string.
TYPE account IS string
```

```
; Interface for retrieving a bank account
; balance. Takes one input (an account
; identifier) and returns the balance.
amount balance (IN account)
```

```
; Interface for withdrawing from a bank
; account. Takes two inputs (account identifier
; and amount to withdraw) and returns
; nothing. Raises the overdraft exception if
; the amount is greater than the current
; balance in the account.
void withdraw (IN account, IN amount)
    raises overdraft
```

In other words, "code" in IDL looks just like declarations in PL/1, COBOL, C, Ada, Algol, or any other modern programming language. In fact, IDL specifications can be automatically generated from various analysis & design or CASE tools, or, in fact, from certain programming languages (such as Lisp).

If you've been writing programs for some time, you'll notice a curious absence from that "code" above. No conditionals, no control statements (gotos, returns, and the like). Not even computational statements (arithmetic). Just specification of interface. That's what an IDL is about: specification of interface.

So why would you want one? You want an IDL whenever you need a written record of an interface that must be shared by multiple people, languages, networks or operating systems.

Well, haven't we seen other languages for specifying interfaces? Certainly, in fact, every language including declarations (like those mentioned above) could be used for defining a public interface. But which one?

The problem is that many languages and system architectures take idiosyncratic (or at least *different*) positions on various data representation questions. Floating point representation, list representation, byte order, even integer length differences abound between programming languages, pro-

gramming models, operating systems, and hardware architectures. What you need is a way to bridge the gap, to describe interfaces in a common language everybody *maps* (understands). An Esperanto, if you will.

The next step involves having *mappings* between these "generic" types used in an IDL, and the declaration style and language of your programming language of choice. For example, in the IDL I used above, the "code"

```
TYPE amount IS fixed_point (10, 2)
TYPE account IS string
amount balance (IN account)
```

would likely map to something like the C code:

```
typedef amount long;
typedef account char *;
amount balance (const account identifier);
```

and furthermore might map into Lisp as:

```
(defun balance (identifier-p)
  (let ((identifier (coerce 'integer
                           identifier-p)))
    (coerce 'string
            ... )))
```

Note the power of this approach: by declaring an interface *once*, in a generic way, we can automate the "appearance" of that interface in many different languages. Even better, a "compiler" or similar tool can automatically take care of byte-ordering, network coding, or any other issues that might be necessary to glue together code written in multiple languages. Or for multiple operating systems. Or running on multiple hardware architectures. Or *connecting across networks*.

That last point, of course—the application of IDLs to network programming (call it client/server or distributed object computing, the role of an IDL in network programming is the same)—is the one that gets all the press. Remote procedure call schemes (RPCs) such as that of the Open Software

Richard Soley is Vice President and Technical Director at the Object Management Group, Framingham, MA.

ADVERTISER INDEX

	PAGE	RSC #		PAGE	RSC #
American Management Systems	79	Rec.	Price Waterhouse	79	Rec.
Borland International	5	4	Protosoft	33	17
Blenheim NDN, Inc.	69	32	Rational Software Corporation	13	8
Cadre Technologies	15	9	Reusable Solutions	18	11
Centerline Software, Inc.	19	12	Rogue Wave Software	23	14
Easel Corporation	41	20	Select Software Tools	25	15
Excel Software	43	21	Semaphore	45	22
Hitachi	27	16	Servio Corporation	C4	3
IBM	7	5	SES	47	23
ICON Computing	37	19	SIGS Books: CD-ROM	59	26
ICONIX Software	17	10	SIGS Conferences: C++ WORLD	53	24
IONA Technologies, Ltd.	36	18	SIGS Conferences: CLIENT/SERVER TOUR	61	27
JumpStart Systems, Inc.	70	33	SIGS Conferences: OBJECT EXPO LONDON	63	28
Knowledge Systems Corporation	21	13	SIGS Conferences: OOP '95	73	29
Martin Marietta	C2	1	SIGS Conferences: SMALLTALK SOLUTIONS	57	25
Object Design	1		SIGS Publications: CLIENT/SERVER DEVELOPER	67	31
Object International	51		US West	78	Rec.
ObjecTime, Limited	58		Versant Object Technology	C3	2
Palladio Software	66	30	Versant Object Technology	11	7
Poet Software	9	6			

Foundation (DCE-RPC) use IDL mechanisms to represent interface, and compilers to automate the generation of network code to link application programs transparently across network boundaries. (The DCE-RPC IDL is named, of course, "IDL"—everybody seems to choose that great name).

There are quite a few other other things we can do with IDLs, however. The IBM System Object Model (SOM) uses an IDL to represent complex interfaces and relationships, enough to allow users to link together programs written in different languages into an executable image—even *without the original source code*. There are software development tools (such as Rational's ROSE Viewer) for managing software projects that automate the creation and management of IDL descriptions to help multiple-programmer software projects run smoothly. There are even academic research programs (at the University of Campinas, Brazil, and the Russian Institute for Problems of Informatics, for example) to encode not only generic *interfaces* but even generic *semantics* in IDLs. And DARPA-funded projects to automate the synthesis of software systems from disparate components stored in a huge United States government code repository.

Oops, we got this far and I haven't yet mentioned objects. No, I didn't forget the name of the magazine. There's obviously a strong connection between the expressive power of an IDL and its utility to programmers. And what better way to improve the expressive power of a language—even just an interface language—than by adding object orientation?

The IDL found in the OMG Common Object Request Broker Architecture (CORBA) specification (called OMG IDL), for example, is *object oriented*. That is, instead of simply allowing the declaration of types and procedures as above, OMG IDL groups types, exceptions, structures, and *operations* (methods) into sets

called *interfaces* (classes). And of course, new *interfaces* may be created by inheriting the components of another *interface*:

```
interface bank_account {
    typedef amount long int;
    typedef account string;

    attribute amount current_balance;

    exception overdraft;

    void withdraw (in account x, in amount y) raises overdraft;
}

interface bank_account_with_overdraft_protection : bank_account {
    attribute amount protection_amount;

    exception overdraft_warning;

    void withdraw (in account x, in amount y)
        raises overdraft_warning, overdraft;
}
```

In this example, the class `bank_account_with_overdraft_protection` inherits from (specializes) the `bank_account` class, adding a new attribute (the amount of protection the customer gets) and a new exception (a warning to be sent to the customer that he is in danger of overdraft); in addition, the new account type changes the interface of the `withdraw` method, adding the new exception as a possible outcome.

OMG IDL in fact allows multiple inheritance, which allows the synthesis of new classes from multiple existing classes. This provides an IDL—which, remember, provides no programming expressiveness—to be an invaluable tool for code development, networking support, client/server development, automatic type-correctness checking of programs.

And, not surprisingly, distributed object computing. ■

For fast, FREE product information, fax this card to 215/785-6073!

PRODUCT INFORMATION CARD

OBJECT magazine

Want a subscription to OBJECT MAGAZINE?

Circle number 999. We'll begin your one year subscription immediately (9 ISSUES @ \$39) & bill you later.

NAME _____
TITLE _____
COMPANY _____
ADDRESS _____
CITY _____
STATE _____ ZIP _____
POSTAL CODE _____
COUNTRY _____
PHONE (_____) _____
FAX (_____) _____

▼ Please complete the following questions

1. My primary function is:

- 1 ☐ Corporate Management
- 2 ☐ Systems Integration
- 3 ☐ Systems Architect, Business/Syst. Analyst
- 4 ☐ Technical/Group/Project Mgr.
- 5 ☐ Programming, Hardware/Software Development
- 6 ☐ Engineering, Scientific, R&D, Educ.
- 7 ☐ Consulting
- 8 ☐ Other

2. My company's primary business is:

- A ☐ Manufacturing (computer or data

systems hardware & peripherals)

- B ☐ Manufacturing (other than computer)
- C ☐ Systems Integration, VAR, Computer Service Bureau
- D ☐ Software Development
- E ☐ Financial, Banking, Insurance, Real Estate
- F ☐ Retail, Wholesale, Distribution
- G ☐ Utilities, Communication Services, Transportation
- H ☐ Government and/or Military
- I ☐ Health Services/Institutions

- J ☐ Education, Consulting
- K ☐ Agriculture, Mining, Construction
- L ☐ R&D, Testing, Evaluation Labs
- M ☐ Other

3. The language I am most interested in is:

- 1 ☐ Eiffel
- 2 ☐ Smalltalk
- 3 ☐ Objective C
- 4 ☐ Object Pascal
- 5 ☐ C++
- 6 ☐ CLOS
- 7 ☐ Proprietary
- 8 ☐ Other

4. The platform on which I work is:

- A ☐ Mainframe
- B ☐ Mini
- D ☐ Workstation
- E ☐ Macintosh

- C ☐ PC
- F ☐ Other

5. My involvement in computer purchasing decisions is (check all that apply):

- 1 ☐ Determine Need
- 2 ☐ Select Vendor/Source
- 3 ☐ Approve Acquisition
- 4 ☐ Not involved in purch. process

6. I expect to purchase the products about which I am inquiring—

- A ☐ Within 3 months
- B ☐ Within 6 months
- C ☐ Within 1 year

BOSL4

To receive **FREE** detailed information on an advertised product or service, circle the number below which matches the one appearing at the bottom of each ad.

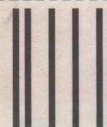
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	999

PRODUCT INFORMATION SERVICE

Send me information on each of the following product categories:

- 801 ODBMS
- 802 Training/Consulting
- 803 CASE Tools
- 804 Develop. Tools
- 805 GUI Tools
- 806 Testing/Debugging Tools
- 807 Development Env.
- 808 DLLs (Dynamic Link Libraries)
- 809 Multi-Media Application Tools
- 810 C/C++ Compilers
- 811 Class Libraries
- 812 Client-Server

NEED FASTER SERVICE? FAX TO: (215) 785-6073



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

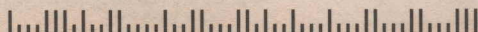
FIRST CLASS MAIL PERMIT NO. 2034 LANGHORNE, PA

POSTAGE WILL BE PAID BY ADDRESSEE

OBJECT MAGAZINE

PO BOX 2034

LANGHORNE, PA 19047-9711



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

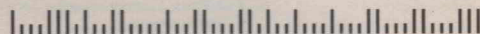
FIRST CLASS MAIL PERMIT NO. 2034 LANGHORNE, PA

POSTAGE WILL BE PAID BY ADDRESSEE

OBJECT MAGAZINE

PO BOX 2034

LANGHORNE, PA 19047-9711



PRODUCT INFORMATION CARD

OBJECT
magazine

Want a subscription to OBJECT MAGAZINE?

Circle number 999. We'll begin your one year subscription immediately (9 issues @ \$39) & bill you later.

NAME _____

TITLE _____

COMPANY _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____

POSTAL CODE _____

COUNTRY _____

PHONE (_____) _____

FAX (_____) _____



To receive **FREE** detailed information on an advertised product or service, circle the number below which matches the one appearing at the bottom of each ad.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	999

PRODUCT INFORMATION SERVICE

Send me information on each of the following product categories:

- | | | |
|-------------------------|-----------------------------------|-----------------------------------|
| 801 ODBMS | 806 Testing/Debugging Tools | 809 Multi-Media Application Tools |
| 802 Training/Consulting | 807 Development Env. | 810 C/C++ Compilers |
| 803 CASE Tools | 808 DLLs (Dynamic Link Libraries) | 811 Class Libraries |
| 804 Develop. Tools | | 812 Client-Server |
| 805 GUI Tools | | |

▼ **Please complete the following questions**

1. My primary function is:

- ☐ Corporate Management
☐ Systems Integration
☐ Systems Architect, Business/Syst. Analyst
☐ Technical/Group/Project Mgr.
☐ Programming, Hardware/Software Development
☐ Engineering, Scientific, R&D, Educ.
☐ Consulting
☐ Other _____

2. My company's primary business is:

- ☐ Manufacturing (computer or data

systems hardware & peripherals)

- ☐ Manufacturing (other than computer)
☐ Systems Integration, VAR, Computer Service Bureau
☐ Software Development
☐ Financial, Banking, Insurance, Real Estate
☐ Retail, Wholesale, Distribution
☐ Utilities, Communication Services, Transportation
☐ Government and/or Military
☐ Health Services/Institutions

3. The language I am most interested in:

- ☐ Eiffel
☐ Smalltalk
☐ Objective C
☐ Object Pascal
☐ Other _____

4. The platform on which I work is:

- ☐ Mainframe
☐ Mini
☐ Education, Consulting
☐ Agriculture, Mining, Construction
☐ R&D, Testing, Evaluation Labs
☐ Other _____

5. My involvement in computer purchasing decisions is (check all that apply):

- ☐ Determine Need
☐ Select Vendor/Source
☐ Approve Acquisition
☐ Not involved in purch. process

6. I expect to purchase the products about which I am inquiring—

- ☐ Within 3 months
☐ Within 6 months
☐ Within 1 year

BOSL4

For fast, FREE product information, fax this card to 215/785-6073—TODAY!

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 a impressive floor of exhibits.

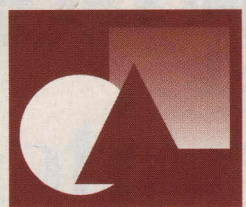
No other conference in Germany provides such an educational range of classes, in-depth advice and hands-on demonstrations in 1995!

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.

Circle 29 on Reader Service Card

January 30 - February 3, 1995
Munich Sheraton, Germany



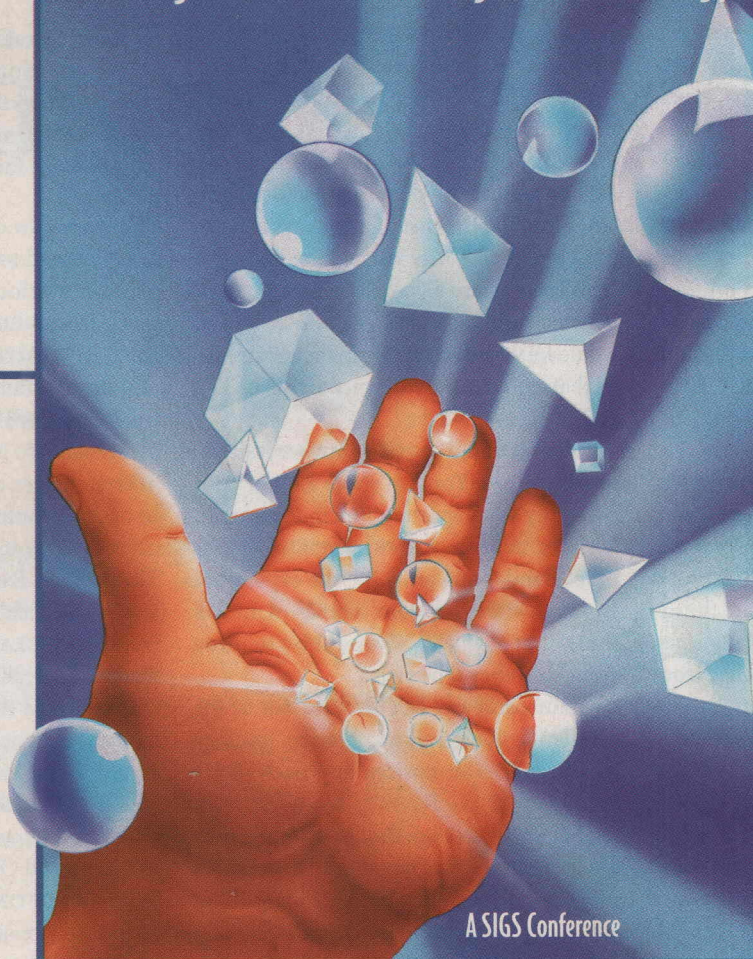
Objekt-orientiertes Programmieren

OOP '95

M Ü N C H E N

Featuring C++ World

Moving Forward with Object Technology



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

Exhibit Sales

Nadine Tillack
SIGS Conferences GmbH
Birkenhohenweg 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 Franczyck, 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 Rösch, 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

Making the most of object technology conferences



THE BROCHURES SHOW up in your mail. Glossy and colorful brochures, with optimistic-sounding advertising slogans—"Bringing Object-Based Solutions to Light" boasts one; another proclaims, "It's Time to Get Real." These brochures are not selling the latest, greatest CASE tools, compilers, or object libraries. What they are selling is an *opportunity*—an opportunity to hear from (and meet!) the leaders in object technology; to attend tutorials that go way beyond "What is an object?"; to get your hands on tools you've only read about; and to get together with peers who share your interest in object technology.

The glossy brochures are advertising conferences dedicated to object technology. The list of speakers at recent object conferences reads like a "who's who" of the object world; if I start to list the names, I'll use up my whole column. Most of your favorite name-brand methodologists are on the list, as are the inventors of various object-oriented languages. The tutorials and presentations range from "Introduction to OOP" through a plethora of technical topics—advanced object modeling, distributed object computing, object-oriented database systems, C++ debugging techniques, application development in Smalltalk, and so on. Increasingly, there are offerings that focus on the business issues in object technology—using objects to reengineer your business, technology transition, training your organization, etc. The conference exhibit halls feature vendors of every imaginable object-oriented tool, as well as providers of education and consulting services, books, magazines, and other

publications. Something for everyone—at least, everyone who is interested in objects.

TEMPTING? YOU BET!

The glossy brochures are very tempting, at least until you get to the registration form, where (in small print) the prices are listed. Basic three-day conference: *A thousand bucks!* Pre- and post-conference tutorials: Add on another three or four hundred dollars. Now add the cost of a few days in a hotel in a place like New York or San Francisco (conferences are always in places with expensive hotels), airfare, taxi, meals. . . It would be very easy to run up the bill to a total that would make your manager blanch—and then say, "No way."

Is it worth the cost? The answer is: It depends on you. You can waste a lot of your company's money attending industry conferences, trade shows, and symposia. On the other hand, you can make the experience so worthwhile that your management will think that they got a bargain by sending you—and they'll be right!

Your company can get a lot out of the excellent conferences on object technology by taking a more *strategic* approach in deciding which conferences to attend, who should go, what they should do while they are there, and what they should do after it's over. In the rest of this column, I'll talk about some ways to be more strategic *before* you go to a conference on object technology; in my next column, I'll address how to get more out of the event while you're there, and what comes afterward.

BEFORE THE CONFERENCE...

Typical "ad hoc" scenario: Tech-minded employee receives glossy brochure for

cool-sounding conference, begs management to send him/her, and gets permission to go, but too late for the "early bird" registration discount.

A more strategic approach: Tech-minded organization identifies the upcoming conferences that are of interest to its business, determines who should go to each, and looks for ways to lower the cost.

There are now quite a few conferences totally devoted to object technology, sponsored by various industry and academic groups. In addition, many other computer-industry events have tracks focusing on objects. Most of these events are held on an annual cycle, with advertising for months beforehand, so it's easy to put together a list of conferences to choose from. Here are a few of the conferences that are devoted to object technology—Object Expo, held in June (Object Expo Europe in the fall); Object World, held in mid-summer and mid-winter; OOPSLA, held in the fall. These conferences are really growing; Object Expo and Object World each had 700–800 "full conference" attendees this year, with thousands of additional visitors to the exhibit halls.

Which conferences are right for you? One thing to consider is the sponsor. Object Expo and Object World are sponsored by SIGS and OMG, respectively, and have an industry/business focus, while OOPSLA, which is sponsored by the ACM, has a more academic slant mixed in.

By treating conference participation in a more strategic way, as part of your continuing education or technical vitality plan, you are more likely to select the best events for your company—best timing, best location, best focus. You can also avoid one of the most common problems that results

Susan Lilly works for IBM's Object-Oriented University, Gaithersburg, MD.

from ad hoc choices—using up all of the available funding early in the year, and missing a later event that might be more important to your business.

WHO SHOULD GO?

Picking the right people to send is one of the main decisions in a strategic approach to conference participation. In general, novices to object technology may not get much out of a conference, beyond the “Intro to OO” tutorial the first day; much of the rest will go over their heads. Someone with more experience will have a better framework for assimilating new ideas, and can take advantage of the excellent opportunity for education on “advanced” topics.

Many organizations have an unfortunate tendency to send the same person (eg, the tech lead) to every event. Good technical employees usually feel rewarded by the opportunity to attend an industry conference; they may be highly motivated to bring back information to the rest of their team. Consider the communications skills of the people who will be attending—what will they bring back home, besides the proceedings? (More on this topic in my next column.)

Project managers will also benefit from attending an object technology conference—the number of sessions devoted to topics of technical management is on the rise. One exception to the “don’t send a novice” rule: I think it would be worthwhile to send an influential manager, who is both cynical (“it’s just a fad”) and unknowledgeable about object technology. The impressive array of speakers and vendors—not to mention the presence of business competitors—may inspire a change in attitude.

Sending more than one person, especially when the conference is large and has multiple tracks, is often worthwhile. However, sending a whole team can be counterproductive; besides the expense, the team members may tend to stick together, missing out on the opportunity to interact with people from the rest of the industry.

WHAT TO DO ABOUT THE COST

Recently, I heard a local discount jeweler’s radio ad, which ended by asking: “Nobody pays retail anymore—why should you?” I suspect that lots of people pay “retail” (full

attendance price) for conferences, but there are often ways to cut the cost—if you’re taking a strategic approach.

If you register on-site, you will pay the full price—and maybe even a penalty for late registration. However, registering on-site is not a strategic approach. Most conferences offer a discount for “early birds”—you can get \$100–200 off just by registering at least a month before the event. In addition, many conferences have discount rates that apply if more than, say, three or four people from the same group register. Generally, the definition of “group” is fairly liberal; if you work for a large company, you might be able to make up a “group” of em-

**You can get more
mileage out of your “paid”
attendee if other coworkers
are made responsible for
visiting the vendors at the
exhibit hall.**

ployees from disparate organizations. I’ve used internal company electronic bulletin boards to locate other people who were planning to go to an event, just to be able to get the group rate. (If you really want to be strategic, consider that speakers often get in for free.)

If the conference is local, check out whether free passes are available for the exhibits. You can get more mileage out of your “paid” attendee if other coworkers are made responsible for visiting the vendors at the exhibit hall.

PLANNING WHAT TO DO WHEN YOU GET THERE

Not long ago I attended what seemed to be an excellent conference on object technology. Yet, returning home, I had the nagging feeling that I hadn’t gotten as much as I could have out of my attendance. At first I couldn’t figure out why—the speakers were good, the topics were interesting, there were lots of vendors in the exhibit hall. After a while I realized that the problem wasn’t the conference; it was me. I was attending in the week between leaving one

job assignment and starting another in a different part of my company. Since I hadn’t yet joined my new group, I really didn’t have a strong sense of what its needs were. Thus, my choices of sessions to attend, vendors to visit, and people to meet were based on my own personal interests, but didn’t focus directly on the business needs of my new organization.

This, obviously, is not an example of a strategic approach; however, it’s probably not an unusual story. A better approach is to plan your conference participation before the event takes place. The pre-conference program, which is typically sent to every registrant, will generally have a detailed schedule of tutorials, presentations, demos, exhibit hours, and so on. Some conferences require you to sign up for every session on the registration form, in order to facilitate their own planning. While this may seem annoying while you’re filling out the form, the conference organizers are probably doing you a favor, by making you plan your own participation.

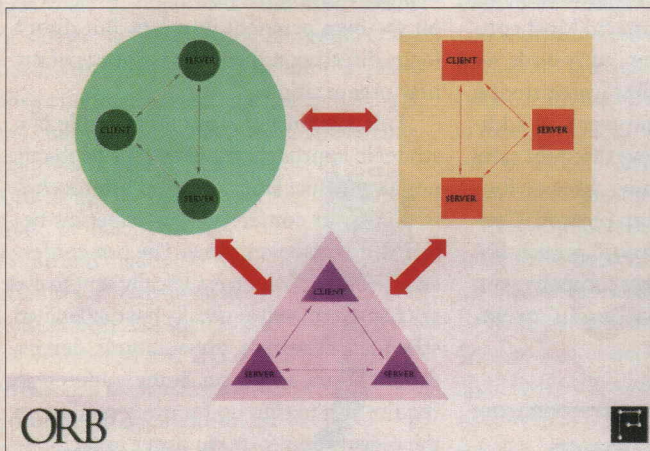
Your plan can be very informal and flexible, but should at least identify the conference events that you must attend, and others that can be traded off to spend more time at the exhibit hall. The exhibition hours typically overlap with the speakers, so be prepared to sacrifice some presentations if you want to visit with the vendors. If more than one person from your group is attending, you can coordinate your plans to get the best coverage of conference events, especially when there are multiple concurrent tracks.

In my next column I’ll talk about getting more than your money’s worth during and after attending an object technology conference. I’ll admit that none of the ideas in this column are rocket science; when you think about it, they are just good business sense. However, I find that when it comes to sporadic events such as conferences, many technical organizations use an ad hoc approach to participation, rather than a strategic one. And that’s just not good business sense. ■

Acknowledgment

The author thanks Sharon Smith of Loral Federal Systems, Gaithersburg, MD, for sharing her ideas on conference strategy.

News and first looks at object-related products



Object Request Broker (ORB) from PostModern Computing Technologies

OBJECT REQUEST BROKER

PostModern Computing Technologies Inc announced ORBeline, its smart Object Request Broker (ORB). ORBeline allows for communication between all classes of computers ranging from a Cray supercomputer, to workstations, to personal computers, and embedded systems. ORBeline's SMART Binding allows two foreign ORBs to talk to each other in a general-purpose way.

PostModern Computing Technologies Inc, 1897 Landing Dr, Mountain View, CA 94043, 415.967.6169, f: 415.967.6212

OBJECTS ASSIGNED TO OSF/MOTIF

Siemens Nixdorf's DiaViews++ class library combines the functions of the OSF/Motif toolkit with the features of C++, including strict type checking and virtual functions. By assigning a C++ class to all OSF/Motif objects such as widgets and gadgets, DiaView++ allows fully object-oriented software development for OSF/Motif.

Siemens Nixdorf USA, 200 Wheeler Rd, Burlington, MA 01803, 617.273.0480, f: 617.221.0220

DEVELOPMENT ENVIRONMENT FOR CREATION OF ENTERPRISE-WIDE WORKFLOW SYSTEMS

Template Software announced Workflow Template (WFT), its object-based development environment for the rapid creation of enterprise-wide workflow systems. WFT allows parallel development of various aspects of a workflow system by separating the activities of task development, abstract workflow design, work item definition, and application integration/instantiation.

Template Software Inc., 13100 Worldgate Dr, Ste 340, Herndon, VA 22070-4382, 703.318.1000, f: 703.318.7378

C++ LIBRARY OF NEURAL NETWORKS

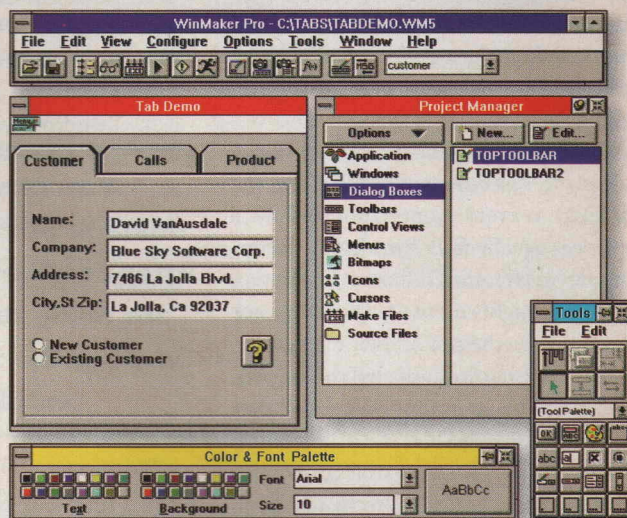
Advanced Computing Labs announced Neural++, a collection of neural networks packages as an easy-to-use C++ library. Neural++ contains neural networks: BP, CPN, Kohonen, and Outstar. Data Scaling and Z-score data preprocessing is transparently automated.

Advanced Computing Labs, PO Box 1547, West Chester, OH, 513.779.2716

PROTOTYPER & C++ CODE GENERATOR

Blue Sky Software announced WinMaker Pro 6.0, its new version of WindowsMAKER Professional, Prototyper, and C/C++ Code Generator for Windows, Win32, and Windows NT. WinMaker Pro's improvements include its new Project Manager, a Property Sheet, a Toolbox, and a Color & Font Palette.

Blue Sky Software Corp, 7486 La Jolla Blvd, Ste 3, La Jolla, CA 92037, 619.459.6365, 800.677.4WIN, f: 619.459.6366.



WinMaker Pro 6.0 from Blue Sky Software

REUSE ENVIRONMENT

CenterLine Software Inc announced ResourceCenter, a reuse environment designed to help development organizations using object-oriented programming to deliver on the promises of software reuse. Built on a client/server architecture, ResourceCenter scales across an organization; its open, documented interface allows users to easily add new software assets to meet their organization's specific needs.

CenterLine Software Inc, 10 Fawcett St, Cambridge, MA 02138-1110, 617.498.30-00, f: 617.868.6655



BOOK WATCH

SYMANTEC C++ PROGRAMMING FOR THE MACINTOSH, VOLUME 1, by John C May & Judith B Whittle, explains the important concepts of the C++ programming language, and provides realistic, easy to follow programming examples designed to work with the latest Symantec C++ compiler. ISBN 0-12-480621-X, \$37.95

AP Professional, 525 B St, Ste 1900, San Diego, CA 92101-4495, 619.699.6735, f: 619.699.6380

Information hiding was an extension to top-down design theory. The idea was to create a module that would provide exclusive access to a physical data structure. All accesses to that structure would be provided by the information-hiding module. An information-hiding module for a stack would initialize the stack and support push and pop operations. A database management system is a gigantic information-hiding module that provides exclusive access to a variety of physical data structures.

Layered software architectures have been used for some time in pre-object software. Each layer of software hides a cohesive set of technological characteristics from the layer above it. The seven-layer OSI model for telecommunication protocols is a well-documented example of this sort of implementation hiding.

End-user computing also predates encapsulation. Many tools are available today that eliminate the need for programming, object-oriented or otherwise. These tools provide a semi-natural language interface so that a user can create software without understanding software technology. Instance-based programming is the latest entry in this category of computer technology-hiding technologies.

If all these concepts were fully exploited, it is very doubtful that object technologies would be anywhere near as attractive as they are. The reality is that none of these technologies was used enough. Existing software is riddled with technological bias and is unreliable and hard to maintain/extend as a result. The lesson is that the concept of encapsulation is not enough to result in software that hides technology.

FAMILIARITY BREEDS ENCAPSULATION

As Tom Petty says, "Good love is hard to find." The same is true with encapsulation. Each new technology seems to go through a phase where the users need to know a lot about the encapsulated technology. Drivers of the first cars had to be car mechanics as well. Users of the first personal computers were often hardware engineers. End-user developers still need to be programmers. Users of relational database class libraries—otherwise known as object-oriented programmers—still need to be relational

database administrators.

The difficulty of achieving encapsulation should sober some object sycophants. Good encapsulation seems to evolve rather than arise spontaneously. Having the elements that allow encapsulation is only the start. Having a curtains, scrims, invisible wires can hide theatrical effects. A director still needs to decide which effects to include in the production, which elements of stage craft can produce those effects and how much hiding to use. The idea of getting encapsulation right the first time and all else is reuse seems pretty dubious. The history of GUI and relational database class libraries illustrates this difficulty.

Early object software developers had to create their own software to interface ap-

**Good encapsulation
seems to evolve rather
than arise spontaneously.
Having the elements that
allow encapsulation is
only the start.**

plication objects with GUIs and relational databases. With limited resources, especially project time, many application objects were bound either to a particular GUI and/or relational database technology. A few people, after a lot of head scratching, started to identify abstractions capable of encapsulating GUIs and relational databases.

Objects are capable of encapsulation. However, people still have to have insight as to where technology begins and ends in order to encapsulate it. Those sorts of insights depend on two very rare abilities. Developers have to be able to separate the logical from the physical, and to identify generic logical requirements.

Separating logical from physical

Encapsulation hides the mechanisms used to implement the abstractions that make up the problem domain. Object-oriented

analysis is the process of completely and correctly identifying those abstractions. Unfortunately, many analysts violate encapsulation because they cannot distinguish abstraction from implementation. The result is a technological bias either toward existing or new technology. In any case, the benefits of encapsulation are compromised.

Consider the developers of OOA CASE tools. They should understand the difference between abstraction and implementation. They should also understand whatever OOA method they're supporting. Unfortunately, their software outrageously violates encapsulation. OOA CASE tools—like their structured/information engineering predecessors—concentrate on the creation and maintenance of diagrams. These diagrams are the basis for the organization of both the analysis and design models. They are nothing more than electronic versions of what we draw with paper and pencil. The problems of maintaining analysis diagrams as complex as a VLSI chip designs and the frustrations of electronic drafting have caused many CASE tools to become shelfware.

Diagram drawing is not the essence of OOA/D methods. Instead, it is part of the implementation of those methods. Analysts and designers shouldn't have to become draftspersons any more than a car driver should have to be an engine mechanic. CASE tools are slowly abandoning the diagram to focus on the essence of analysis and design. As they do, they provide further evidence of the difficulty of getting encapsulation right the first time.

Identifying generic logical requirements

Great software technologies recognize the great degree of commonality among seemingly diverse abstractions. Relational technology is based on the belief that information a system must remember can be organized into tables. Object technology advocates contend that all system information and behavior can be implemented with objects. The creators of both technologies are rightly acclaimed because they have identified a major insight about abstraction commonality. The few heroes of software technology serve as statistical evidence to the difficulty of identifying generic logical requirements.

Those who develop software technology deal with a different problem domain than those who develop applications. Technology developers want to build a product that can work with any problem domain. Application developers want to build an application that can be implemented with a variety of technologies. It makes sense that technology developers see a lot more commonality among applications than application developers do. Unfortunately, those who develop technology usually fail to identify all the commonality that exists among logical requirements. Instead, it is the application developers who must complete the job. Without the experience across a wide range of applications and the imperative to develop a product with broad market appeal, it is no surprise that most application developers fail to extend implementation technology.

The result is insidious infiltration of implementation technology into an application. Much of the volume of application software is really unrecognized application generality that is implemented uniquely. The bottom line is that way more technology could be hidden by encapsulation. However, before it can be hidden, it must be separated from application requirements.

TO BE CONTINUED...

Only the most primitive software technologies have been unable to hide technology. While object technology has some features that help make it a better technology hider, that alone will not guarantee encapsulation.

Far more than a new technology, developers need to have some rare and difficult-to-acquire skills. Then, they have to be given the time to use those skills. Even then, good encapsulation may not result the first time around.

Encapsulation is clearly critical to reuse. It is the quality that allows technology to be reused in different applications and applications to be ported to different technologies. It is so powerful an idea that many in the object world want objects to hide more than technology—they want objects to hide application data and procedures as well. Exposing the hype associated with this aspect of encapsulation will be the subject of my next column. ■

Come build the future, one object at a time.

Object-Oriented Software Design and Development

Be a part of the exciting future in Telecommunications with U S WEST Technologies. Make the vision of the information highway become a reality. We need experienced LEAD SOFTWARE ARCHITECTS, DESIGNERS, AND DEVELOPERS to help build the next generation of U S WEST network services operations systems. State-of-the-art distributed object systems and applications will be developed to support U S WEST's digital public switched voice and data networks, AIN-based services, broadband video/multimedia services and wireless/PCS network services. The applications for these network services will support key network and business processes: network planning, forecasting, engineering and design, construction and installation; network surveillance and performance monitoring, self-healing service restoration, network management, automated non-intrusive testing, and service repair. Employ the latest object-oriented design and development tool technology to build these UNIX-based distributed client-server systems, which will be the computing infrastructure of the future in Telecommunications.

If you want to help build the future of object-oriented software, you belong at U S WEST Technologies in Denver.

To be part of our team, you must possess both a B.S. and M.S. in technical disciplines (Ph.D.'s with practical mindset also sought), 3+ years' relevant industry R&D experience, and extensive experience in object-oriented software design and development technology, including experience in: UNIX-based/C++ client-server and manager-agent systems; CORBA and/or DCE-based distributed object applications; CASE technology for object-oriented design and development; object reuse and reusable platforms; X-based window builders and application generators; format software development processes and tools; and software and document management systems. You must also have experience implementing and executing distributed applications and servers for user-layer, application-layer and database-layer over various data network protocols: X.25, TCP/IP, Frame Relay and other LAN/MAN/WAN interfaces. In addition, knowledge and experience in telecommunications and network operations systems domains for switching, signaling, transport and adjuncts for various network technologies and services are required.

U S WEST Technologies, Inc. is an equal opportunity employer. Any offer of employment is contingent upon the applicant's undergoing and passing a pre-employment drug test, subject to applicable federal, state and local law. **Send your resume to: U S WEST Technologies, 931 14th St., Suite 1420-HM, Denver, CO 80202, REF. #OM1112/PC.**

USWEST
TECHNOLOGIES, INC.

Give your ideas more bandwidth.

To place an ad, please call Michael Peck at
212 242-7447

Computer

SMALLTALK/OOP Birmingham, AL

For more than 20 years, American Management Systems, Inc. (AMS) has led the way in the application of leading-edge information technology for our client base of Fortune 500 companies and public sector organizations. Today, our reputation as an industry innovator is stronger than ever.

Continued growth has created immediate opportunities in Birmingham, Alabama for qualified object oriented programmers, designers, and architects.

Requirements include 1+ years of experience in one or more of the following:

- Smalltalk
- Envy/Developer
- ODBMS
- C++
- OOA/OOD

Experience with Windows or UNIX is a plus.

We offer a competitive compensation and benefits package and career growth opportunities. For immediate consideration, please send your resume in confidence to: Dept. OM-9411, AMS, Two Chase Corporate Drive, Suite 105, Birmingham, AL 35244. An Equal Opportunity Employer M/F/D/V.



AMERICAN MANAGEMENT SYSTEMS, INC.



DOES YOUR MIND'S EYE SEE THE FUTURE

AS **OBJECT ORIENTED?**

If you share our vision that object oriented methodologies are dramatically impacting application development and information systems design, **Price Waterhouse** wants to discuss your future career plans.

Our Object Oriented Methods Practice is expanding. We need **object oriented architects, analysts, designers and programmers** experienced with the following tools and methodologies:

C++	SMALLTALK	NEXTSTEP	OBJECT DATABASES
BOOCH	RUMBAUGH	JACOBSON	SHLAER-MELLOR

If you are among the best and the brightest and have 5 years or more of information systems technical experience and 2 years of object oriented tools or methodologies;

Please fax your resume to (301) 897-4408 or send to:

Price Waterhouse

Suite 500

6500 Rock Spring Drive

Bethesda, Maryland 20817

Attn: NPd

Price Waterhouse LLP



Authors Wanted For Two Innovative Book Series

Managing Object Technology

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

For more information please contact:

Dr. Richard S. Wiener

135 Rugely Court

Colorado Springs, CO 80906

(phone & fax) 719-579-9616



CLIENT SERVER DEVELOPER

Call for Papers!

CLIENT/SERVER DEVELOPER is a new publication committed to helping programmers, developers and technical managers understand C/S technology. We are now actively seeking manuscripts on the following:

Operating Systems • Databases • Programming Languages • Object Technology and Reuse • C/S Application Design Methodologies and Tools • Software Engineering Methodologies • Pre-Packaged C/S Applications • Business Process Re-Engineering • Project Management in a C/S Environment • Metrics and Testing • Multimedia

To submit an article or request author guidelines, contact:

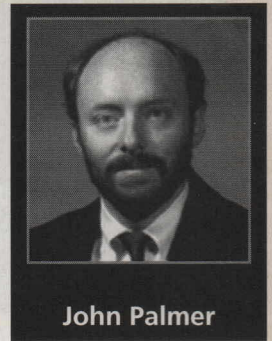
Thomas O'Flaherty, Editor

411 West End Avenue, Suite 2B

New York, NY 10024

Phone: 201.801.0050 Fax: 201.801.0441

Published by SIGS PUBLICATIONS, Inc.



John Palmer

Violating encapsulation

THE OBJECT PARADIGM is defined as encapsulation, inheritance, and polymorphism/dynamic binding. In previous columns, I've taken aim at some of the hype associated with inheritance and polymorphism. I suppose that I'm enough of an object convert to have been blind to hype associated with encapsulation. As a COBOL programmer, I learned the hard way about the problems with common data. As a PL/I programmer, I learned the beauties of variable scoping. Most recently, as a Smalltalk programmer, I've realized my most obsessive data hiding dreams.

As a practitioner of structured design, I know that communicational cohesion—creating a module that contains all of the functions associated with a data structure—is a pretty good rationale for composing a module. I've long considered information-hiding modules to be a great addition to a structure chart. But my honeymoon with encapsulation is over. The gloves are off. The hype associated with encapsulation will be revealed.

ENCAPSULATION:

A NEW WORD FOR TECHNOLOGY HIDING

All gurus essentially agree: encapsulation is a quality associated with the knowledge of how to use an object. Ideally, understanding what an object can do should be sufficient to understand how to use it. It should not be necessary to understand how the object works. Encapsulation is the concept of hiding the implementation of an object from view. Encapsulation therefore reduces, drastically in many cases, the amount of knowledge necessary to get

benefit from an object. A telephone is a decent real-world example of encapsulation. Using a phone requires only minimal knowledge of how a phone works. Most people don't know about the analog to digital to analog conversions that takes place with a modern phone call. They are also oblivious to the parsing of the call into packets routed by a variety of means from one location to another. By hiding an object's mechanism, the design reduces the impact of evolving technology. New technology can be substituted for old without redesigning the entire object. Over the years, the completely analog phone system has become more and more digital. The general public has been largely oblivious to this technology transformation. In the same way, if a software object implements its functions with a link list one day and a relational database the next, consumers of that object's services should be unaffected.

By hiding the mechanism that makes an object work, the object usually becomes easier to use. When an object is easy to use, it is easier to see when it could be used in different situations. All sorts of new uses for phones have been invented. The telephone is now used to check voice mail messages and process banking transactions. In each case, from the customer's point of view, knowledge of what the object could do was pretty much sufficient to use the object in a new context. In the software world, work gets done with data structures and algorithms. As a result, definitions of encapsulation often mention hiding the data structures and algorithms used by a computerized object. This is especially true if the internal workings of the object reflect technology constraints. If a designer knows only what a software object will do, that object (or class

from which it is instantiated) can be easily used and potentially reused. The more a programmer needs to know about an object's data structures and algorithms, the more difficult that object/class is to use or reuse.

THE LATEST NAME FOR AN OLD IDEA

Encapsulation is a new name for a very old idea. Back in the top-down, structured programming days, we talked about *black boxes*. Such a module could be understood by knowing its inputs and outputs and the name of the function it performed. By creating a hierarchy of black boxes, top-down design promised to hide implementation technology.

Structured design added the idea of transform-centered design. This pattern for a module hierarchy proposed the creation of a standard set of subhierarchies. For each form of input technology, a subhierarchy would be created to extract logical data from its physical format. For each type of output technology, a subhierarchy would be created to add physical format to the logical data. The remainder of the module hierarchy—called the transform center—could then deal exclusively with technology-neutral data. If the technology changed, it would only affect the input/output subhierarchies.

Data independence is another old term associated with hiding implementation technology. Back in the pre-database days, programs were coded to deal with the specific physical file structures used to store data. Any time these physical structures were changed, the programs that referenced them had to be changed. Database management systems attempt to hide the physical data storage structures. Programs access a logical organization and the DBMS translates in real time between logical and physical structures.

continued on page 77

John Palmer is a Principal of the Atlantic Systems Guild.

How To Build High-Performance Object Applications Faster With VERSANT

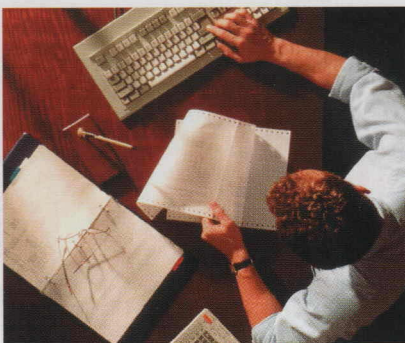
IF YOU'RE programming in C++ or Smalltalk and storing your objects in a relational database, you've probably encountered some nasty problems that hinder application development and decrease run-time performance.

Problems like having to write "mapping code" to map programmatic objects to tables. Executing that mapping code at run-time, which slows down performance. And waiting for your relational database to perform costly "join" operations to reassemble the structure of your objects.

To solve these problems, we've built the VERSANT Object Database Management System (ODBMS). By storing objects directly *as objects*, VERSANT eliminates wasteful mapping code and costly relational joins, making it faster and easier to build high-performance object applications.

Transparent Interfaces Eliminate Mapping Code

Ernesto Herrera, President of Miramar Technology, says: "It took our ARGOS development team six days to write a sales analysis prototype using an RDBMS. With VERSANT, we finished the same prototype application in just one day — because VERSANT eliminates the need to map Smalltalk objects to tables."



Mark Woyna, Section Manager at Argonne National Labs, relates a similar experience: "In building our VIEWS workbench, VERSANT allowed us to eliminate literally thousands of lines of C++ mapping code. This not only accelerated development to help meet our deadline, but improved performance and made our workbench product more robust."

Navigation Eliminates Costly Relational Joins

VERSANT eliminates costly relational joins by modeling relationships between objects with its unique logical object identifiers. By traversing these links, VERSANT lets you navigate directly between objects — at rather astounding speeds. One of our customers, for example, benchmarked VERSANT at 100 times faster than a leading relational database on the same set of complex data.

Production Ready

VERSANT includes expected production features such as DBA utilities, server-enforced data integrity, object-level locking, and transparent data distribution. And VERSANT is in production use, today, at large sites.

Argonne's VIEWS Workbench, for example, was a finalist in Computerworld's recent Object Technology Awards in the demanding category of "Best Use of Object Technology In an Enterprise or Large Systems Environment."

To build your high-performance object database applications faster, learn more about VERSANT. Call us at **1-800-VERSANT, ext. 403**. Email us at info@versant.com. Or, fill in the coupon and FAX or mail it back to us.

You'll be on the way to building applications faster. And building faster applications as well.

FAX or Mail Today!

☐ YES, please send me more information on VERSANT.



Name _____		
Title _____		
Company _____		
Address _____		
City _____	State _____	Zip _____
Telephone () _____		

Fax to: 415-325-2380

Or Mail to: Versant Object Technology,
1380 Willow Road, Menlo Park, CA 94025
USA

(403)

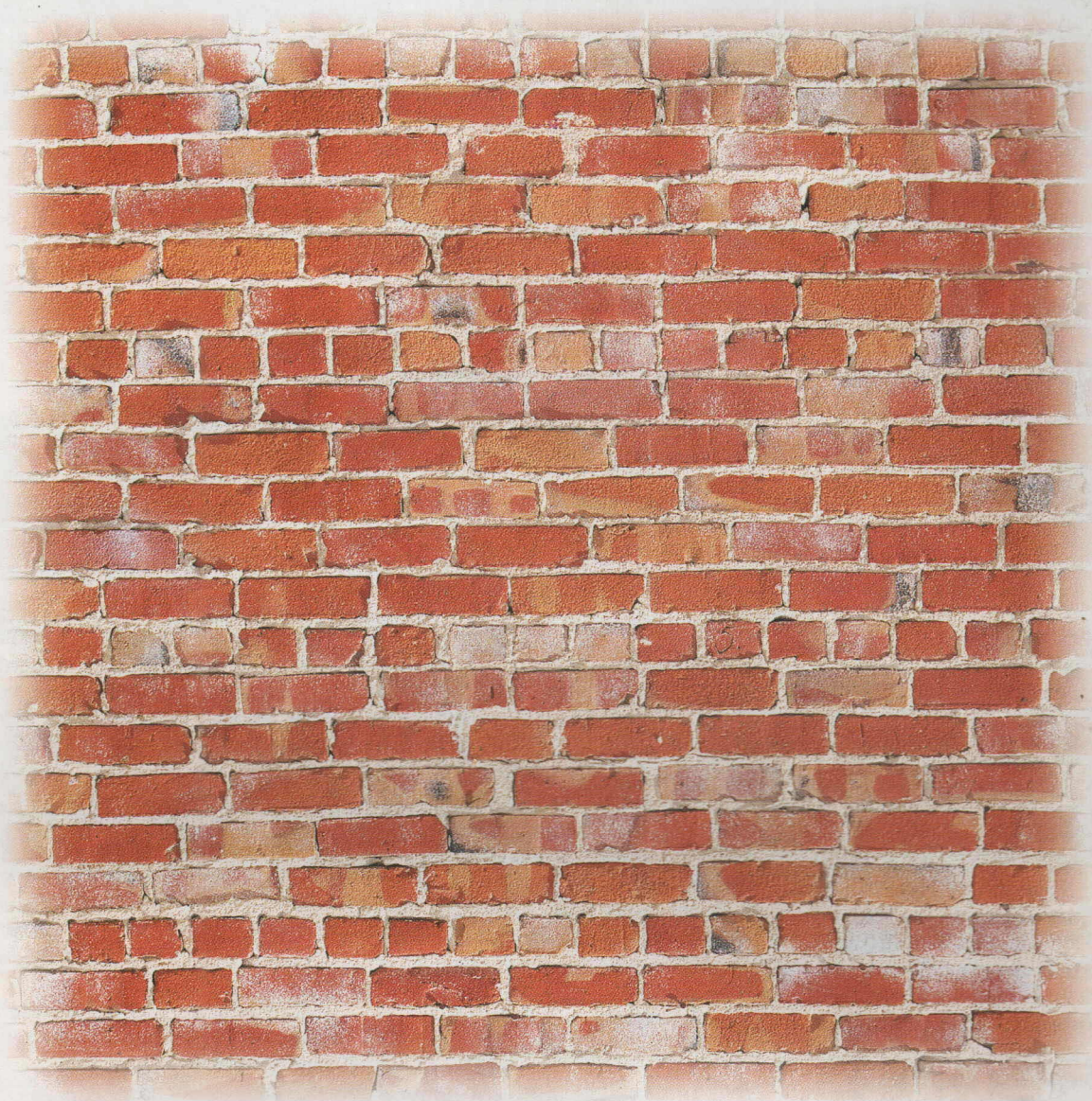
VERSANT™

The Database For Objects™

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

© 1993 by Versant Object Technology. VERSANT and The Database For Objects are trademarks of Versant Object Technology Corporation. All other company names and logos are registered trademarks of the individual companies.

Circle 2 on Reader Service Card



FORCE-FIT RELATIONAL TECHNOLOGY AND YOU COULD REALLY HIT IT BIG.

Maybe you're beating your head against the relational development wall – trying to integrate your Smalltalk or C++ applications with an RDBMS. Maybe you're spending all your time debugging SQL calls instead of building great applications. Or maybe you've hit the relational performance wall because you're wasting too much processing time on object decomposition and recomposition.

Servio™ has a better way. With our high-performance GemStone® object database management system, you

can store Smalltalk and C++ objects directly in the database. We make your development time more productive and your object applications more efficient.

Learn for yourself by calling us today for a copy of "Object or Relational? A Guide for Selecting Database Technology." After all, the best way to deal with an obstacle is to avoid it in the first place.

SERVIO
**OBJECT TECHNOLOGY
FOR THE REAL WORLD**

Call 1 800-243-9369 for a free copy of "Object or Relational? A Guide for Selecting Database Technology."

Servio is a trademark and GemStone is a registered trademark of Servio Corporation.

Circle 3 on Reader Service Card