DEVELOPER'S
PowerBuilder Journal

Editor-in-Chief: John Olson

Art Director: Jim Morgan
Executive Editor: M'lou Pinkham
Managing Editor: Hollis K. Osher
Production Editor: Brian Christensen
Internet Editor: Andy Blum
Technical Editor: Bernie Metzger
Watcom SQL Editor: Joe Celko
DataWindows Editor: Richard Brooks

WRITERS IN THIS ISSUE

Michael Barlotta, Andy Blum, Aleksandr Godik,
Bill Green, Bernie Metzger, John Olson, Tony Pow,
Chris Roller, Taonan "Peter" Tsai

SUBSCRIPTIONS

For subscriptions and requests for bulk orders,
please send your letters to the Subscription Department

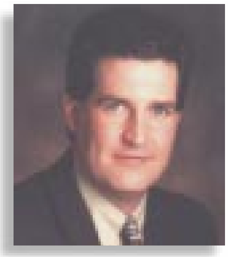
Subscription Hotline: 800 513-7111
Cover Price: $14/issue.
Domestic: $149/yr. (12 issues)
Canada/Mexico: $169/yr.
All other countries: $179/yr.
(International subscriptions by U.S. Banks or Money Orders only)
Back Issues: $12 each

Publisher, President and CEO: Fuat A. Kircaali
Vice President, Production: Jim Morgan
Vice President, Marketing: Carmen Gonzalez
Accounting: Ignacio Arellano
Graphic Designers: Robin Groves
Alex Botero
WebMaster: Robert Diamond
Senior Web Designer: Chad Sitler
Advertising Account Manager: Robyn Forma
Advertising Assistant: Megan Ring
Circulation Manager: Mary Ann McBride
Customer Service: Sian O'Gorman
Paula Horowitz
Ann Marie Milillo
Online Customer Service: Mitchell Low

EDITORIAL OFFICES

SYS-CON Publications, Inc.
39 E. Central Avenue, Pearl River, NY 10965
Telephone: 914 735-7300
Fax: 914 735-6547

POWERBUILDER DEVELOPER'S JOURNAL ISSN # 1078-1889 is published monthly (12 times a year) for $149.00 by SYS-CON Publications, Inc., 39 E. Central Ave., Pearl River, NY 10965-2306. Periodicals Postage rates are paid at Pearl River, NY 10965 *and additional USPS offices.* POSTMASTER: Send address changes to: POWERBUILDER DEVELOPER'S JOURNAL, SYS-CON Publications Inc., 39 E. Central Ave, Pearl River, NY 10965-2306.



By John Olson, Editor-in-Chief

Getting on the Right Track

Setting the Standard

There's clearly a trend in software development toward component-based architecture, distributed objects and Java as a central language. Each of these lead to quicker development through reuse as well as greater portability through platform independence. Unfortunately, there are several "standards" being promoted in these areas and therefore there's no guarantee that any two of the hundreds of tools will create objects that will work together.

Many standards have emerged over the last few years, but we've reached a point where only a few remain. Even so, there's still a dizzying number of technical domains, and within each one there are one or more standards. Until we achieve a single final standard in each domain, or at least achieve a point where all tools can support the remaining standards, there's no guarantee that the objects you develop will support the final standard.

In the world of distributed objects, the main contenders for a standard component model are Microsoft's DCOM (Distributed Component Object Model) and the Object Management Group's CORBA (Common Object Request Broker Architecture), though EJB (Enterprise JavaBeans) – the component model for Java – is the model of choice for that language.

These dueling standards remind me of the "narrow," "standard" and "wide" gauge railroads that are used in various locations around the world, including the U.S. In the 1800s narrow gauge railroads were the standard in some mountainous areas of the U.S. and Canada, including parts of the Pacific Northwest and Alaska. Narrow gauge was used because it was more economical and easier to install. The problem was that these regional railroads were incompatible with the major transcontinental railroads.

Recently, my wife and I traveled around Scandinavia and took a side trip to St. Petersburg, Russia. Once in Russia, it becomes necessary to change trains because the tracks in Russia are wide gauge while most of the bordering countries use standard gauge. Within Russia all train cars have wide gauge axles because all tracks are wide gauge. However, when attempting to transport goods via train to other countries, the incompatibility of wide gauge axles to standard gauge tracks requires moving the goods to standard gauge cars. In the end, there's an extra cost associated with transporting goods to and from Russia.

The U.S., on the other hand, chose to standardize on the European standard gauge. Subsequently most narrow gauge tracks were torn up and replaced. This standard is used not only throughout North America but also in South America and in much of the rest of the world. To transportation companies in the U.S. this means that equipment won't restrict where the trains can go. Because of this standard, goods can travel on the same train car 10,000 miles from the northernmost point of Alaska to the southern tip of Chile.

So why don't we have a single accepted standard for these software technologies? Standards evolve as technologies evolve, and numerous possible standards get developed. Some are even put into practice, but many are weeded out by superior standards as well as by the politics of large corporations like Microsoft, IBM and Sun. Recent history has shown that software standards often narrow down to two major standards: the Microsoft standard and the "everybody else" standard. Historically, Microsoft hasn't shown itself to be a leading innovator, but rather a very good follower. This is shown by the fact that early generations of its tools are often inferior to the competition. However, it uses its powerful distribution pipeline and broad customer base to gain market share, then uses closed technology to avoid losing customers to direct competition. By creating its own "standard," its tools are effectively proprietary and incompatible with the competition. While the real standards are being haggled over, it uses its massive brain power to catch up with the competition. By the time an industry standard is adopted, the Microsoft tools are successfully competing and Microsoft already has substantial market share. Microsoft has been very successful with this strategy.

New Feature

Our goal is to give you the information you need to understand the dizzying array of standards so you can choose the right tracks on which to run your business enterprise. For that purpose we've created a new feature section for distributed technologies and Michael Barlotta will lead the way as our section editor.

Over the last several years Sybase has redirected its energies toward developing features in the three technology areas of component-based architecture, distributed objects and Java. As the technology evolves, so do the Sybase development tools and servers. Watch for a steady stream of articles reviewing these features and showing you how to use them to solve real-world problems.

Next month: PowerBuilder 7.0! ◆