FEATURE ARTICLE

The Business of Application Portability

David Rowley
MKS, Waterloo, Ontario

■ The focus of this issue of *StandardView* is on application portability. At its most basic level, portability is an *economic issue.* It centers on leveraging an existing investment and deploying it in new ways. It's about being able to run an application written for one platform on an entirely different platform. It's about saving time and money, and maintaining quality. Portability is fundamentally a *business issue,* not a technical one.

Portability is also an old issue. Portable Fortran in the late 1960s first gave rise to the need to move scientific applications from outdated equipment to the then brand-new. From the earliest days of application development, rapid advancement of computer hardware and fierce competition among platform vendors have created the need for moving applications. Business requirements change, integration requirements become more stringent, mergers and acquisitions force previously separate MIS groups to exchange data and applications, and to adopt new platform standards. All these factors create a landscape where application portability is key to competitiveness and to the abilities to innovate and provide high-quality application support to line areas.

Application Interface

The fundamental building block of software portability is the Application Programming Interface or API. It represents the contract between the supplier of a service (an SDK, OS, etc.) and the user of the service (the application). When multiple vendors agree to implement to the same API definitions, users of those definitions can move their software from one platform to another. Typically, APIs are described in terms of their C or C++ calling sequence. While applications are written in many different languages (assembler, COBOL, Pascal, Smalltalk, Delphi, PowerBuilder, etc.), portability discussions usually center around the 95% portable languages of C and C++. The non-C-based languages often provide their own portability environments. For example, applications written in ParcPlace Smalltalk are portable across all the platforms ParcPlace supports. Both C and C++ walk a middle road, providing a high degree of portability while at the same time providing a high degree of openness and variation.

The Value of Portability

The value of portability is not unique to computing. The need to move intellectual investment from one "platform" to another is commonplace. The canonical example is the VCR. VHS has become the de facto standard in prerecorded video cassettes. The Beta vs. VHS battle is long over. In a short time, the

Portability must be a design requirement from day one.

prerecorded videotape industry dropped support for the Beta format, in spite of its technical superiority. The battle was decided not by technical strength, but rather by market share. Through short-sighted licensing practices, Sony kept the penetration of the Beta format low. Portability is enhanced by working with the most popular standard, which is not necessarily the best one.

Timing Is Everything

As with Gallo wines, no standard should be defined before its time. The computing industry is advancing at a rapid pace, and users want to procure against standards as much as possible. This is the double-edged challenge of portability: standardizing enough functionality to leverage the investment while ensuring that innovation occurs in areas that are proving commercially useful. Standardizing too late causes arbitrary diversity and wasted investment. Standardizing too early entails the risk of entrenching an approach or technology that doesn't meet real-world needs. Creating standards that don't serve commercial interests slows down the growth of the entire industry. Striking a balance is the key to effective standardization.

Non-Portability

One leading electronics company is in the midst of an all-too-typical project: reengineering an outdated application written in a variant of Business BASIC that is no longer supported by the vendor. The application consists of hundreds of individual modules, comprising more than a million lines of code. Their best alternative, the company decided, was to take on the conversion effort of moving the product to ANSI C, one module at a time. The frustrating aspect of this project is that the primary deliverable is producing the exact same functionality as is offered now, but converted to a contemporary ANSI C-based development approach. This provides the users with no immediate improvement, other than the ability to more rapidly respond to user demands in the future and lower maintenance costs. By not taking a long-term view of the portability of the application, the company has incurred a huge redevelopment cost. Other companies can benefit from this lesson by treating the ability to maintain and port the application to future environments as a design requirement from day one.

Portability and Standards

Portability is closely related to the concept of standards. Application portability is enabled by multiple vendors creating separate implementations that conform to the same standards. Standards fall into two categories: de facto (determined by the marketplace) and de jure (determined by procurement bodies). Examples of de facto standards include Visual Basic, Win-32 and TCP/IP. Examples of de jure standards include POSIX.1, ANSI-C, and now TCP/IP. Note that TCP/IP falls into both categories. This networking standard grew in popularity through grass-roots adoption and investment, eventually leading to standards bodies defining a specific TCP/IP standard and creating policy to mandate TCP/IP-compliant networks.

Dictatorship vs. Democracy

Given the relationship between standards and portability, there are basically two approaches to developing APIs: the benevolent dictator (also called "the Microsoft approach"), and the open democracy (also called "Open Systems"). Each model has advantages and drawbacks.

The Microsoft approach fuels rapid development. So long as the vendor invests the time and effort to fully understand user requirements and takes innovative approaches, and so long as the basic paradigm remains intact, the approach can be very productive. The Win16 and Win32 de facto standards have rapidly created an enormously successful and profitable software market. Microsoft has responded well to the demands of this marketplace, and, for the most part, the market has benefited. If, however, the paradigm falls out of favor (such as happened to IBM's mainframes, and is perhaps happening now to Microsoft due to the Internet), then customers may jump ship to a supplier that can better promote the new opportunity. Certainly, Netscape is betting on this trend.

The open democracy approach has the benefit of putting the evolution of the technology into the hands of an open process, whereby direction is determined by consensus. This process is typically administered through any one of a number of standards bodies (formal or otherwise). Its advantages lie in the breadth of input that can be taken into account and the harmonization with other standards. The downside is the sometimes glacial pace of standards efforts, with their tedious review/edit/ballot cycle and the fact that some specification enhancements are academically oriented and create extensions that have not been validated in the marketplace.

For an interesting look at standardization and portability in the Windows world, specifically the APIW effort, see "Applications Programming Interface for Windows—A Timely Standard" by Rob Farnum in this issue.

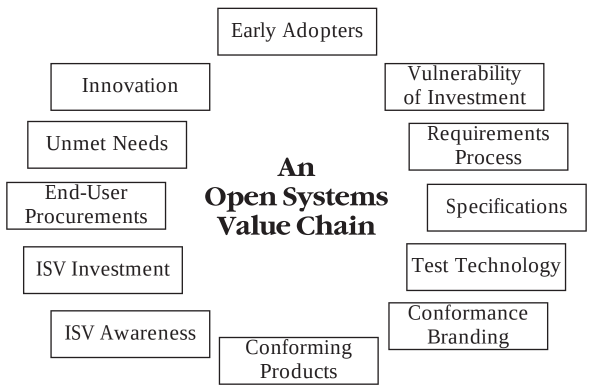
Innovation

The computer industry was built on innovation; its people are extremely creative and inventive. The problem is that often a new technology will become a standard before we have evidence that it is suitable for handling the mainstream, mature demands placed on it (CORBA and DCE, for example). Standards established and mandated before their underlying concepts have been fully proven in the marketplace can stifle creative solutions. Therefore, the best standards are those that deal in interfaces already accepted by the marketplace. The value of the standards process is in creating unambiguous definitions of those interfaces, and providing a forum for their ongoing evolution (SQL, ANSI C, ANSI C++, HTML 3.0, etc.). This process can be conceived as an "open systems value chain."

No Relevant Standards?

The above approach works well if there happen to be existing standards or specifications to draw from. But what if there aren't? For example, there is no cross-platform GUI standard that runs on all systems. There are a number of commercial offerings: VISIX Galaxy, XVT, Neuron Data all provide comprehensive portable GUI environments cross-platform. But they aren't standards (in fact, an effort to standardize a cross-platform GUI by the IEEE as P1202 failed). This means that an investment in any one offering may be riskier than is acceptable. Plus, you may want at some point to port your application to a platform that the suppliers don't (or don't want to) support. Suddenly you are faced with a complete rewrite just to target a single additional platform.

If source code for the product is available, you are on much firmer ground, especially if the product is written against lower-level industry standards (X-Windows, POSIX, etc.). If you are then faced with the need to port your application to an unsupported platform, you can do the port of the GUI layer yourself. Some companies (including MKS) have, for some projects, chosen TCL/TK for this reason. The benefits of TCL/TK are that it is available in source code, it is freely usable for commercial purposes, it is robust, and is well supported by the TCL community. The downside is that it assumes an X Windows environment (though Windows and Macintosh versions are forthcoming).



When source is not available, you as an ISV should carefully consider the interfaces being invested in. It is often practical to define a higher-level, more abstract, interface, and then write the application against that, effectively creating your own internal standard. This allows you to move to a new platform by writing a new "middleware" layer, preserving the bulk of your investment. MKS has successfully done this in our MKS Toolkit product, writing a POSIX.1 style layer for Windows, Windows NT, Windows 95 and OS/2, allowing us to achieve 95% portability of the core source code.

A Pragmatic Approach: Spec 1170

Standards efforts have started to take a more pragmatic approach. The first such example is the UNIX harmonization effort, Spec 1170, named for the 1,170 APIs that eventually made their way into the standard. Spec 1170 was an acknowledgment that it is more important to leverage existing investment than to invent new solutions to old problems.

In spite of UNIX applications generally being regarded as "portable," ISVs were frustrated by the extensive and arbitrary differences between the flavors of UNIX. Supporting a UNIX application commercially meant wrestling with a huge number of independent implementations, each offering slightly different functionality. This led to large amounts of platform-specific code, arcane collections of #ifdefs, and rapidly unmaintainable applications.

Finally, the UNIX vendors took practical steps towards resolving this situation and establishing a more competitive UNIX marketplace. In 1993, a group of system vendors (including IBM, HP, DEC, Sun) undertook a study of the top 50 UNIX applications. This initiative was called COSE, or the Common Operating System Environment. Among the applications analyzed were: Autocad, Cadence, Frame, Lotus, Island Graphics, Visix, Wingz, and SAS. The investigation determined the portability requirements of each application. This information was distilled into a requirements specification. These requirements, in harmony with existing open systems standards (POSIX, X/Open, OSF), led to the creation of Spec 1170, which includes all of the functionality in POSIX.1, POSIX.2, and the XPG4 Base.

In March 1994, the document was transferred to X/Open for "fast-tracking" and was renamed the Single UNIX specification (X/Open is annoyed if you still refer to it as Spec 1170.) X/Open has now launched a branding program that allows any system vendor to brand their offering as UNIX (specifically UNIX 95). Not only does this mean greater commonality between AIX, Solaris, HP/UX, and other historical UNIX implementations, it also allows other systems such as Windows NT, IBM MVS, DEC VMS, and Linux to also be branded as full UNIX platforms.

What Went Wrong?

The above story may sound like UNIX has finally been unified, once and forever. Unfortunately, the

An application doesn't become a solution until it is up and running.

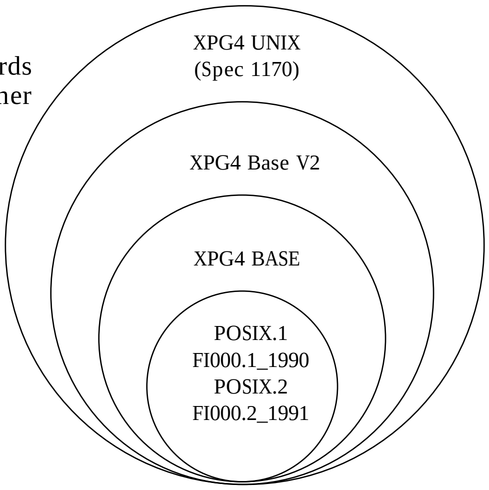
impact of Single UNIX has yet to be felt. System vendors are still working to incorporate full Single UNIX conformance into their next major OS release. Even when this happens, there will still be portability issues. The Single UNIX specification group decided that X-Windows and Motif should not be included in the final document, as this would disallow "server-oriented" systems from achieving UNIX branding. As a result, they are now included in the separate "Common Desktop Environment" standard, which not only includes the developer APIs, but also a standard "look and feel," standard desktop accessories (calendars, etc.) and a common desktop manager. It will be the end of 1996 or early 1997 before a critical mass of platforms support both the Single UNIX and CDE standards. Once this happens, ISVs will be able to write full graphical applications that are source-code portable to a large number of environments.

The Future

The key revelation in this project is the approach: standardizing APIs that are actually being used, leveraging existing investment, and harmonizing them with existing standards. The approach guards against the OSI syndrome of creating standards that nobody ends up using, and it can serve as a roadmap for future standards efforts.

Specifically, future standardization efforts could entail establishing a dialogue with as broad a subset of the ISV community as possible, and performing an ongoing analysis on their applications to determine the active areas of investment by the group. The APIs that draw the greatest investment would then be specified, standardized, and established as requirements for the system vendors to support. Previously, establishing the infrastructure required to support this process would have been prohibitively difficult. Fortunately, thanks to much better tool support and the Internet, implementing this approach is now feasible and cost-effective.

How Standards Work Together



HTML

Portability is a key attribute of many Internet-related standards, HTML being the obvious example. While there are a great number of vendor-specific extensions (Microsoft's scrolling banners and AVI widgets, Netscape's tables and frames, and JavaScript), the industry has recognized the importance of adopting broadly-supported and portable HTML standards. The HTML 3.0 standard, for example, defines the specific behavior of HTML tables. This ensures that any HTML 3.0-compliant page using tables will be accurately viewed by any HTML 3.0-compliant browser. The major vendors of both HTML authoring packages and browsers are supporting this standard.

This example again illustrates the important balance between innovation (non-portability) and standardization (portability). If nobody was enhancing the functionality of the web; if everyone was waiting for a standards body to dictate what is and is not valid, the web would not be experiencing the incredible growth (and investment) we see. Because developers are throwing new functionality into the ring and working to get users to adopt it and to recognize the benefits of the extensions, the web platform is rapidly evolving into a highly responsive and agile marketplace where the focus is on commercial value, not standards.

Portability and Testing

One valuable side-effect of application portability is the impact it has on quality. When an application runs on multiple platforms, defects that may only show up under unusual circumstances are more effectively detected and corrected. Even though an application may be 100% portable, there is no substitute for platform-specific regression testing. Users demand (reasonably) that the application perform properly on the platform desired. This means that portability of the test technology (harness, test cases, etc.) is also a requirement in defining a portable application. Test harness vendors (Mercury International, SQA, etc.) ensure that their products run on a variety of platforms and that the test cases can be used on multiple machines.

Portability Branding

Recently, there has been a great deal of discussion within the standards community on the issue of "application branding." This refers to a formal process whereby the portability requirements of a specific application are determined through the use of a tool, and are then fed into a "Good Housekeeping Seal of Approval"-style branding mechanism. This brand would allow buyers of the application to understand how readily the vendor may be able to deploy the application on additional platforms, and how well they may respond to future platforms.

While this aspect of branding has value, the real value of portability branding may lie in the outsourcing of development projects to third parties. A mandated portability policy in the development of the application could be mechanically verified to ensure that the development company adhered to the stated specifications. Where source is not supplied to the customer, the supplier could provide a standard set of reports with the deliverables to certify that they had followed the policy.

Application branding could also form part of a company's adherence to their ISO 9000 development policy, ensuring that their applications were designed to meet changing requirements in a volatile world.

For a closer look at the issues of application branding, Paul Tanner's "Software Portability: Still an Open Issue?" provides a good overview of X/Open's efforts and vision in this area.

Unintentional Non-Portability

Even though many development managers have portability as a standard objective, applications often become non-portable gradually and unintentionally. A few new developers who are not familiar with the coding policy can slowly compromise the application's portability. Unless the application is regularly built and tested on all the platforms supported or analyzed by portability verification tools, the problem can quickly get out of hand.

Key Standards[1]

So what is the state of the art when it comes to standards? Well, these should come as no great surprise, since they are, indeed, standards: POSIX System Interfaces, POSIX utilities, real-time extensions, sockets, shared memory, TCP/IP, ANSI-C, DCE, X11, Motif, Win32 APIs, SQL, ODBC, C++, COBOL, Fortran, NFS, CDE (Common Desktop Environment), CORBA, OLE, RPC, OpenDoc. Notice that no operating system names appear in the list (neither Windows nor UNIX nor MVS). Standards are available in many environments, therefore providing the alternatives required for true open systems.

That's fine in theory, but what about in practice? There is no value in referencing a standard unless it is actually available on the platforms you want to target (both now and in the future). Here's how a few of the common operating systems stack up against some of the items in the list:

The POSIX system interfaces include basic APIs to file I/O (open, read, write, close), process primitives (create a new process, job control), signals, etc. The POSIX utilities include the historical UNIX commands such as the Korn shell, awk, grep, the vi editor, tar, uuencode/uudecode, ls, etc. Emulators such as Bristol's Wind/U, Mainsoft, and Insignia Solutions all help to leverage the existing investment in Win16/Win32 APIs to new platforms. In fact, Bristol has recently announced they will be bringing the Wind/U environment to Digital's Open VMS. As the market demand for standards availability grows, expect to see much more standards availability for the ISV community.

Once ISVs have the ability to write portable applications, system vendors will continue to promote their own specific added value, encouraging ISVs to write custom versions of their applications to take advantage of the added functionality. ISVs must carefully weigh the tradeoffs entailed in following this path. Their competition may well be prepared to write for a specific platform (especially one with a good market share). If the custom development results in a more attractive product (performance, functionality, integration), then the effort is warranted. This scenario demonstrates how portability is not an all-or-nothing issue. ISVs can profit by ensuring that, say,

Standard	Microsoft NT	SOLARIS 2.3	AIX V4.1	IBM OS/2	HP-UX 9.04
POSIX System Interfaces P1003.1	Yes	Yes	Yes	No	Yes
POSIX Utilities P1003.2	No	Yes-O	Yes	No	Yes
XPG3 Base	No	Yes	No	No	Yes
XPG4 Base	No	Yes (V2.4)	Yes	No	Yes
ANSI C	Yes	Yes	Yes	Yes	Yes
DCE	Yes	Yes	Yes	Commit	Yes
X-Windows	Yes-O	Yes	Yes	Yes	Yes

1 Information courtesy of D. H. Brown Associates. Used with permission. From DHBA Report: "Update to Standards Conformance: An Ongoing Battle" June 1994.

Microsoft's dominance of the desktop is being challenged [by Java].

only 75% of their application source code is portable. The other 25%, clearly identified as platform specific, takes advantage of platform-specific features.

Beyond API Portability

While API portability is important, there is more to open systems. The tools available within the environment, the documentation, the quality assurance process, are all areas of great investment. MKS believes the following challenges must still be overcome:

—common documentation

—portable program construction tools

—portable test technology

—portable install tools

—unified software distribution channels

COMMON DOCUMENTATION

One of the impediments to application portability, especially in the UNIX community, is the lack of standard documentation. As a developer sits down to write code, he typically either consults the on-line documentation for a specific platform (Solaris, AIX, HP/UX, etc.) or the hard-copy versions of same. As a result, it is difficult for the developer to understand which APIs are based on standards and which represent vendor-specific value-adds.

Fortunately, X/Open is now in the process of creating user-level documentation, describing the functionality of the Single UNIX Specification. For the first time, the "Common Doc" project will enable a developer to consult documentation for a portable application development environment, with vendor-specific value-adds being clearly identified.

CONSTRUCTION TOOLS

The process used to build the actual application from the component source code, data files, etc. is often complex, and supported by a host of different tools. Compilers, linkers, profilers, debuggers, make engines, parser generators, SQL/4GL preprocessors, are all huge dependencies. ISVs must also weigh the portability of these tools when looking at platform-specific investment.

TEST TECHNOLOGY

Software is so complex these days that manual quality assurance and verification is almost completely obsolete. As a result, there is an increasing investment in test technology (frameworks, test plan tools, etc.) These tools must also run within each supported environment, often becoming a portability problem themselves. Common examples of problems include: variation in output formats from platform to platform (say, due to floating point representation) leading to bogus test failures, fragility of GUI test plans, and so on.

INSTALL TOOLS

An application doesn't become a solution until it is up and running on the customer's platform. The process whereby your application is installed into the user's environment is often complex. Handling dependencies related to other components, directory structure layouts, security, user administration, disk space, and license management are all items where platform-specific approaches add to the ISVs' expense.

UNIFIED SOFTWARE DISTRIBUTION

Finally, the cost of developing platform-specific distribution channels is often prohibitive. An ISV may well be able to service its Windows, OS/2, Solaris and AIX customers by working with standard, mass-market-oriented distribution partners. The cost of servicing DG/UX, MVS, VMS, MPE/iX and Linux customers often leaves these markets untapped. A more unified approach to software distribution would make a dramatic difference in leveling the playing field. Fortunately, Internet-based distribution seems to be positioned to fill this void.

How Do I Adopt a Portable Approach?

Understand your investment, and work to protect it. Try performing a Spec1170-style analysis on your own applications. Determine your dependencies on underlying build tools. Support the suppliers that provide the standards you depend upon. Tell them what you need and, more importantly, tell them what you don't need. Talk to other ISVs and users in similar situations. Compare notes. Develop your own consensus. Work with organizations like X/Open, OMG, UniForum, and OSF to ensure they are responding to your real-world needs.

At every turn, work to reduce the arbitrary differences between platforms that end up costing us all. Only when ISVs and users get involved will the potential of open systems be realized and result in a larger and more cost-effective market for your products.

Portability vs. Availability

One important aspect to keep in mind is the distinction between the portability of an application and the actual availability of that application. Even though the code base might be portable to 20 flavors of UNIX, unless the vendor builds, tests, sells, and supports the product on each of those platforms, the portability of the application is irrelevant to the customer. As a result, portability is primarily an issue of interest to application developers, both ISVs and internal MIS developers.

INDUSTRY GROUPS

Due to the specialized requirements of certain sectors of the economy, a number of industry groups have formed to serve the portability and standardization needs of their member companies. No vertically-oriented group stands out more than the Petro-technical Open Systems Consortia. Supported by the major oil and gas companies, POSC has established a leadership role in industry-oriented standardization efforts. Alan Doniger and Nigel Goodwin's article, "Standards: What's in it for Me?" clearly illustrates the business advantages of such an approach.

THE U.S. GOVERNMENT

One staunch supporter of application portability through standardization has been the U.S. government, with efforts led by both the National Institute of Standards and Technology (NIST) and the Defense Information Systems Agency (DISA). As Gary Fisher from NIST points out, their use of a disciplined approach in writing portable applications stems from their interest in reducing costs and improving deployment flexibility.

STANDARDS BODIES AND CONSORTIA

The best way to get a handle on the standards-based choices available is to familiarize yourself with the activities of the various consortia. The following is a selection of the more useful organizations and their respective mission statements:

X/OPEN COMPANY LIMITED

X/Open is a not-for-profit, vendor-independent, international consortium dedicated to the advancement of open systems throughout the world. It has become the integrator of standards within the industry, bringing together users, vendors and standards bodies working towards the proliferation of open systems.

Since its founding in 1984, X/Open Company Ltd. has demonstrated its unique ability to bring value to open systems application and systems developers and those who use them. Today the X/Open brand, used by hundreds of commercial and government organizations worldwide, is the most widely-used tool for making sense of the "standards confusion."

IEEE (POSIX)

The objectives of the society's standards activities are: "to provide an organizational framework and conducive environment within which to develop broadly accepted, sound, timely, and technically excellent standards that will advance the theory and practice of computing and information processing science and technology."

OBJECT MANAGEMENT GROUP

"The Object Management Group (OMG) is a nonprofit consortium dedicated to promoting the theory and practice of object technology (OT) for the development of distributed computing systems. OMG was formed to help reduce the complexity, lower the costs, and hasten the introduction of new software applications. Our goal is to provide a common architectural framework for object-oriented applications based on widely available interface specifications."

UNIFORUM

UniForum is a vendor-independent, not-for-profit professional association that helps individuals and their organizations increase their information system's effectiveness through the use of open systems, based on shared industry standards. Central to UniForum's mission is the delivery of high-quality educational programs, trade shows and conferences, publications, on-line services, and peer group interactions.

OPEN SOFTWARE FOUNDATION (OSF)

The Open Software Foundation is a not-for-profit research and development organization whose goal is to provide a software solution that enables computers from multiple vendors to work together in a true open systems computing environment. OSF uses an innovative open process for selecting and implementing technology.

The Future of Portability

The whole "open systems" approach is rapidly accelerating. The Internet in particular is fueling this acceleration. Let's examine why.

Issues of portability and standards become more important as equipment from different vendors must work together. The Internet is the ultimate example of this. Multiple hardware configurations, operating systems, databases, protocols, etc. are being thrust together as never before, necessitating a much greater focus on portability and interoperability standards.

A new approach to portability on the client side is being put forward by Sun with their Java interpreter. Java takes a portable bytecode approach, offering a standard set of services (networking, GUI, I/O) that can be used to write small, object-oriented applets which can be sent down the wire to a Java-enabled web-browser to implement full client-side interactive front-ends. While initial experimentation with Java seems limited to rather unimpressive embellishments to standard HTML pages, the full promise of Java is clear-full client-side portability. Microsoft's dominance on the desktop is challenged as the investment line moves from Win32 to the Java API.

As a result, Netscape and other browser suppliers are well-positioned to ensure that these new standards become thoroughly entrenched.

Summary

Application portability plays a central role in the cost-effectiveness of information technology. There is already a great investment being made in tools and SDKs that facilitate portability, and even more sophisticated products are coming on the market to make the process more manageable. An approach midway between "standards for standards sake" and "we'll stick with a single vendor who will take care of all of our needs" is the best prescription for taking commercial advantage of portability. All these issues point to the central tenet: understand the portability of your software investment, and design it into everything you do from day one. In this chaotic market, you never know when you're going to need it. **SV**

Where to Find More Information. Interested readers can get further details on portability, application programming interfaces, standards and standardization by contacting:

X/Open, http://www.xopen.org/; UniForum, http://www.uniforum.org/; ECMA, http://www.ecma.ch/; POSC, http://www.posc.org/; OSF, http://www.osf.org/.