

Information Systems Industry

Bay Colony Corporate Center
1100 Winter Street
Waltham, Massachusetts 02154

Telephone 617.487.3700
Telefax 617.487.5750

Prospects for Unix in a Standards World

Gordon Bell
Consultant to Decision Resources

Business Implications

- Users are driving vendors to adopt open system standards that achieve portability and interoperability. However, given vendors' propensity to create unique products, a single operating system running on multiple platforms, which achieves essentially the same results, may be the only feasible way to meet these demands.
- The past inability of Unix vendors to agree on a unified standard bodes ill for the newest call for a common interface standard: Spec 1170. The likelihood that vendors can and will stop providing uniqueness to differentiate their Unix dialects is almost nil.
- Windows NT poses the gravest threat to Unix thus far. Its compatibility with the huge base of PC applications, multiple platform operation, integration with SQL and network services, and high volume will challenge Unix's hegemony on workstations and servers, and precipitate a whittling down of Unix dialects over the remainder of this decade.
- Applications are the key to success of any computing environment. Higher volume and, thus, lower costs/prices can be achieved by minimizing the number of versions supported. Competitive pressures will favor the development of sole-sourced Windows NT applications over those for multiple dialects of Unix. This trend is already too far along to reverse.

The Development of Standards

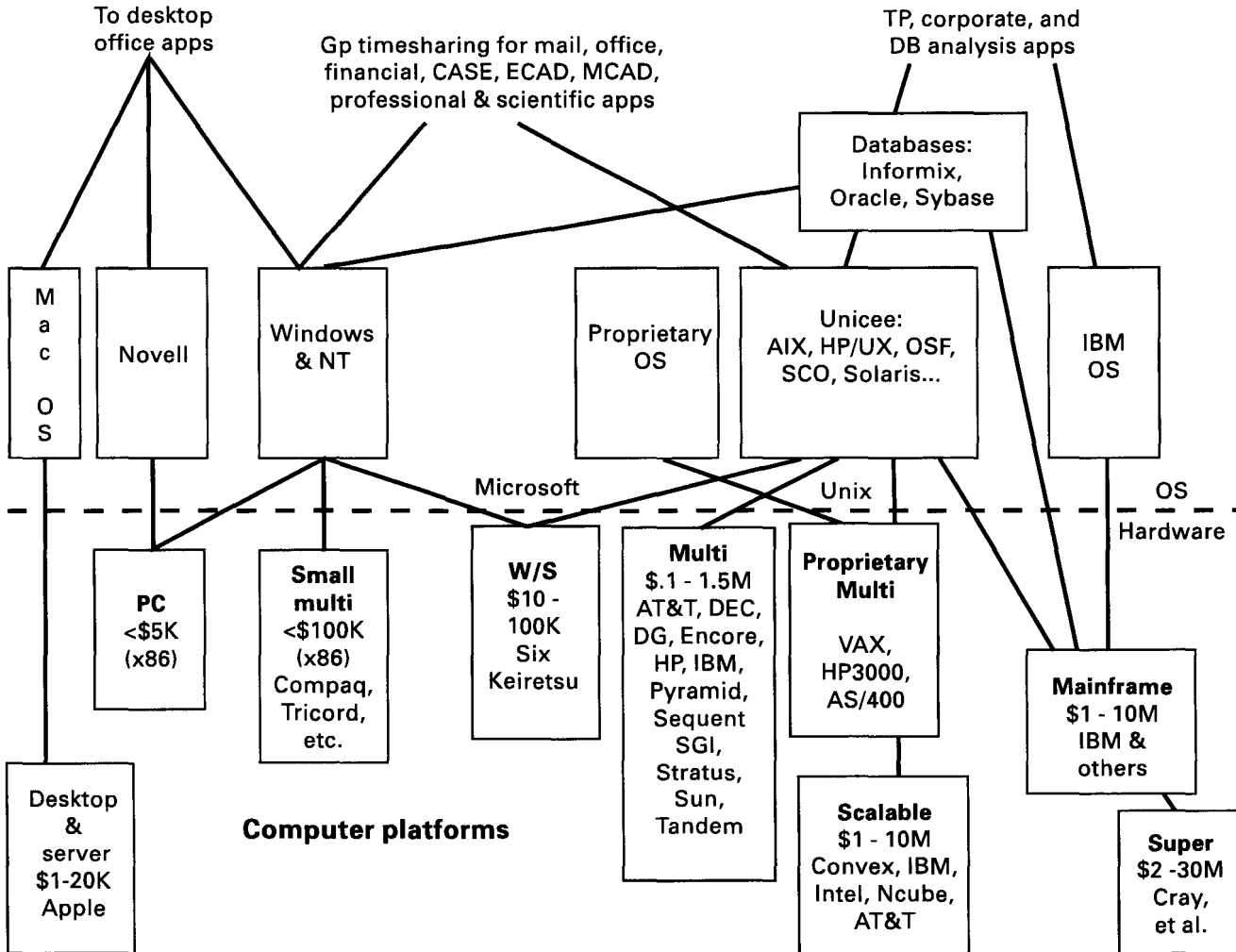
The computer industry can be viewed as a number of independent and competing subindustries that have created a series of "proprietary standards" and platforms (Figure 1). A proprietary standard may be established by a single company (e.g., Intel's X86 standard) or through cooperative arrangements among many companies (e.g., IBM/Motorola/Apple's PowerPC standard). Standards in the computer industry can develop in many other ways; more than a dozen types are shown in Table 1. These range from a single company achieving market dominance to a completely chaotic and democratic design process, such as the process that created the Internet. This latter process allowed Gopher, the World Wide Web, and the Mosaic browser to form despite the attempts of various standards bodies, forums, and consortia to establish standards, oftentimes without the benefit of any implementation knowledge or user input.

When one vendor becomes dominant in a particular market, we call its proprietary standard an *industry*, or *de facto*, standard. De facto standards can develop slowly or quickly, depending on the dynamics of a particular market. De facto standards are the most important and most widely used in the computer industry precisely because they result from market dominance.

Standards that are developed by government entities or standards bodies to foster interoperability are called *legal*, or *de jure*, standards (in other words, standards arrived at by process that is not market-based). For example, CCITT standards specify international telecommunications protocols for interoperability. POSIX is a de jure standard established by the IEEE to define the

Figure 1

"Proprietary Standards" and Platforms in the Computer Industry



Source: Gordon Bell.

language interface between application programs and the Unix operating system (OS). POSIX is also a de jure standard because it is a federal government requirement for products purchased by various governmental agencies. De jure standards are rarely useful to fast-changing markets because they take a long time to formulate. The concept of an open system based on de jure standards is unachievable—the fantasy of standards bodies and marketers pushing proprietary brands.

An *open* standard (as defined by IEEE P1003.0) implements sufficient open specifications for interfaces, services, and supporting formats to enable applications software to do the following:

- Be ported with minimal or no changes to a wide range of systems.
- Interoperate with other applications on local and remote systems.
- Interact with users in a style that facilitates user portability.

Open specifications are public, maintained by an “open” consensus process to accommodate new technology, and consistent with international standards. Unix is considered an open standard, although in reality there are many different Unix dialects. De facto standards arrived at through marketplace battles are critical for the development of APIs. The operating

Table 1
"Standards" Types

Standard Type	Examples
Industry (de facto)	Intel's X86 microprocessor for PCs
"Wannabe" de facto	Apple/IBM/Motorola's PowerPC
Open	None (Unix dialects are unique and thus not really "open.")
Proprietary	IBM's AIX
Trademark	Novell's UnixWare
PR standard	Spec 1170
Implicit cross-platform	SQL
Explicit cross-platform	Visix's Galaxy
De jure (legal)	POSIX, IEEE 802, OSI
Government-mandated	Ada, DES
Cross-industry forum	ATM, MPEG
Consortia	OSF, OMG
Company-centered consortia	PowerOpen, SPARC International
Chaotic	Internet, Mosaic

Source: Gordon Bell.

system with its file system, windows, user interface, and languages create the API for applications on a particular platform.

The Evolution of Systems

In the 1960s, the mainframe industry consisted of IBM, RCA, General Electric, and the BUNCH (Burroughs, Univac, NCR, Control Data, and Honeywell). By the mid 1970s, the IBM System 360 hardware and operating systems had become the de facto standard, with only System 360-compatible system vendors (e.g., Amdahl, Fujitsu, and Hitachi) seriously competing in this market.

Likewise, the minicomputer market, which was born in the 1970s, coalesced from many vendors to just three significant hardware and operating system platforms: Digital Equipment (DEC) VAX/VMS, Hewlett-Packard (HP) 3000/MPE, and IBM AS/400. However, none of these can be called a de facto standard because none dominates the market. Furthermore, none serves as a standard platform available from multiple suppliers for an independent software market.

The early workstation market, created by Apollo and Sun, quickly evolved to a Unix standard for the operating system. However, each vendor developed its

own Unix dialect (which originally ran on Motorola's 680X0 architecture) to run on one of six RISC-based microprocessors, essentially maintaining the same proprietary stance that evolved in the mainframe and minicomputer markets.

Virtually every type of application is available on the PC because of the large markets made possible by standardization.

The PC market has had a different evolution. With the DOS operating system freely licensed to all platform vendors, and a single microprocessor standard established by Intel, a large number of PC suppliers have grown and prospered around the Intel and Microsoft proprietary standards. Apple has been the only successful alternative with its Macintosh line of personal computers. However, Apple's refusal to license the operating system until 1994 stifled chances for it to grow market share much beyond its early success.

The mainframe and PC are both de facto standard platforms. The IBM 360 and Intel X86 hardware architectures plus IBM and Microsoft operating systems define these standards. In the PC market, however,

multiple, competing hardware platform suppliers have dramatically lowered the cost of computing as compared with other markets. Virtually every type of application is available on the PC because of the large markets made possible by standardization. The PC standard demonstrates the impact of high volume on costs, and competition on price.

Portability and Interoperability

Standards are aimed at achieving portability and interoperability. *Portability* is the ability to move applications and data among systems and have them operate in the same manner on the new system. *Interoperability* is the ability to exchange data among application programs such that the data can be used meaningfully by the receiving program. Interoperability is a property of a particular application or system based on whether it has external inputs or outputs that may interface to other programs.

It is clear that single file and data formats can be accomplished in practice only with a single operating system.

Many programs interoperate but not in a sufficiently detailed fashion to provide the complete functionality of the interoperable components. For complete interoperability, file formats must be identical. Proponents of object technology claim that interoperability is not possible except through the wide-scale adoption of object standards that handle intercommunication. However, we believe that a single OS running on multiple platforms provides the best environment—although it alone is not a sufficient condition—for ensuring that developers, encouraged by users, will support interoperability.

Users want freedom to choose the most cost-effective solution at each level of integration, or independent system, and then to have the various parts or systems interoperate. Open system standards, which are designed to enable portability, often require extensive modifications to programs and data, ignoring the fact that users' largest investment is in programs, data, and training. To be cost-effective, there can be only one file format such that program, data, and people can be moved among the systems at zero cost. Given the complexity of platforms, it is clear that single file and

data formats can be accomplished in practice only with a single operating system.

Tests for Applications Portability, Compatibility, and Openness

Ideally, users want applications that can be moved from platform to platform with the following characteristics:

- No additional training or work, such as recompiling.
- Effortless file and database portability.
- Compatibility with other applications and the operating system in style and substance.
- Interoperability with other systems and applications.

Users do not want to have to translate programs or data to operate with another system. Nor do they want to have a systems expert for each dialect of an operating system running on multiple platforms. For client/server environments, the ability to effortlessly operate or hold any part of an application on any platform by any manufacturer is key. To qualify for true platform portability, compatibility, and openness, the following four questions must be answered affirmatively:

1. Is there a single source file for all applications for all ports of an application across multiple platforms?
2. Is there a single user manual, system maintenance and reference manual, and training course for all ports of an application across multiple platforms?
3. Is there one format for the removable media and server for all ports of an application across multiple platforms?
4. Can an arbitrary client/server application interoperate across multiple vendor platforms running either as client or server?

To achieve these goals, the best and perhaps only feasible standard is a de facto, single operating system standard.

Architecture Keiretsu: Software Platform Standards

Table 2 shows the variety of software platform standards that exist around the six major microprocessor architectures and corresponding hardware platforms.

Table 2
Architecture *Keiretsu* for Software Platforms

Operating System	Microprocessor <i>Keiretsu</i>					
	Apple/IBM/ Motorola PowerPC	DEC Alpha	Intel X86	HP PA-RISC	MIPS R Series	Sun SPARC
Unicee	AIX Apple A/UX NextStep Solaris	OSF/1 Ultrix	NextStep SCO Unix Solaris UnixWare m-variants	HP-UX NextStep (future)	IRIX	Solaris SunOS NextStep (future)
Microsoft	NT	NT	NT Windows	NT	NT	NT via Intergraph
Other	MacOS OS/2 Taligent	VMS VAXELN	OS/2 CTOS NetWare Vines Teradata Taligent	MPE Taligent	—	1st Person

Source: Gordon Bell.

These architectures form both vertical or supplier-chain *keiretsus*¹ and horizontal *keiretsus* where all companies use the common architecture to build platforms. The largest platform *keiretsu* is built around Intel's X86 architecture and Microsoft's operating systems.

The variety of operating systems that create unique platform environments for each of the six microprocessors is large. For example, the PowerPC microprocessor can be controlled by at least six operating systems: Microsoft's Windows NT, Apple Computer's Macintosh OS, IBM's OS/2 and AIX, Motorola's Unix System V port, and Sun Microsystems's Solaris. However, a common hardware platform does not make an application portable among these six OSs. On the other hand, applications written for any one operating system, such as Windows NT or Solaris, are portable (with recompilation) among hardware platforms that run NT or Solaris. The situation for X86-based microprocessors is even more variable with dozens of operating environments—although only a few are significant to the industry: Windows, NT, NetWare, and perhaps SCO's Unix (V.3-based).

Sole-Source Proprietary Environments

Sole sourceness is the degree to which the operating system and hardware platform are available from a single

supplier. Table 3 shows the sole sourceness in terms of the availability of hardware from a single source and the degree of control a single manufacturer has over the operating system. The industry started at the upper left of the table where one company controlled the definition of the hardware architecture, hardware platform, and the operating system. The PC demonstrated the power of multiple vendors and a single, de facto standard. However, the most powerful combination is in the lower right corner, where a single de facto OS can run on multiple platforms, which permits common usage of data and files across platforms. Currently, only NT provides this environment.

If we look at DEC and HP minicomputers in the box in the upper left-hand corner (IBM, Data General, and other minicomputer vendors also fit into this box), we observe that each of their environments (hardware and software) is proprietary and that each of their operating systems runs solely on the respective vendor's platform. Macintosh systems prior to 1995 also fit into this definition. In essence, each vendor created a unique environment that requires specifically ported applications, which garner unique data and files. This "lock-in" to a sole-sourced hardware and software environment enabled companies to main-

1. *Keiretsu* is a Japanese word that describes a group of affiliated companies.

Table 3**Degree of Sole Sourcedness for Various Platforms**

Hardware/Software	Sole-Source Software Controlled by Hardware Vendor	Sole-Source Software Unix Dialect (Unicee)	De facto Standard Software
Sole-Source Hardware Platform	VAX / VMS, HP 3000 / MPE, Macintosh/MacOS (<1995)	Alpha / OSF/1, PA-RISC / HP-UX, Mips R Series / IRIX, others	None
Single-Architecture Platform, Multiple Hardware Vendors	IBM 360 / MVS, Macintosh/MacOS (>1994)	X86 / SCO Unix, UnixWare, others	X86 / DOS/Windows, NetWare
Multiple Architectures, Multiple Vendors	X86, PowerPC/ OS/2	SPARC, X86, PowerPC / Solaris	X86, Alpha, Mips R Series, PowerPC, PA-RISC / Windows NT

Source: Gordon Bell.

tain high margins on their products, but caused users to begin looking elsewhere for solutions that provided more flexibility. These users turned to Unix.

With the promotion of Unix as an open standard, the “illusion of open systems” was created, whereby every interface is defined openly and is available for use across the industry. However, users soon found that if a few parts of an operating system, which otherwise adheres to a well-defined set of public standards, are unique, “openness” is meaningless: the environment is still essentially proprietary. For instance, DEC and IBM have declared that the VMS and AS/400 operating systems, respectively, are open because they adhere to well-defined standard languages and interfaces. But specific lower-level standards, such as file formats, operating system call interfaces, and command languages, are unique (and possibly proprietary because they were maintained on a unique and closed basis by one vendor). Although all specifications are publicly available, users do not care because VMS and AS/400 are sole-source environments, entrapping applications developers and users once these operating systems are adopted.

The Unix Trap

By making the various Unix dialects (Unicee) unique in some aspect, the Unix vendors have not met users’ requirement for application portability across platforms. In essence, each has created its own pro-

prietary Unix environment—only slightly more beneficial to users than the proprietary minicomputer environment—not a truly open system.

By building a unique Unix dialect platform under the rubric of adding value for users, a vendor is able to lock-in users to its products, thus maintaining higher profit margins by inhibiting competition for subsequent business. This scheme has worked well for low-volume mainframes and minicomputers, but PCs based on the DOS/X86 standard demonstrate another market model that works well—this one based on cost, volume, and direct competition.

By modifying Unix, these companies have created unnecessary differentiation and caused millions of hours of make-work for downstream applications developers and users.

Starting with a common operating system such as Unix does not make a common applications programming interface (API); that is, an application on one Unix platform cannot simply be recompiled to run on another Unix platform. With six major microprocessor architectures forming multiple workstation, server, and PC platforms, the so-called “common” Unix operating system has been modified by each platform vendor. Not that these modifications have not produced added value; they have, but by doing so, these

companies have created unnecessary differentiation (from the users' perspective) and caused millions of hours of make-work for downstream applications developers and users who deal with intersystem incompatibilities.

The fear of Windows NT becoming a de facto, cross-platform standard has caused Unix providers to focus again on Unix standardization.

When users were mostly developing custom applications and CPU performance was the primary purchasing criteria, it made sense to choose a workstation or minicomputer with an operating system "tuned" to maximize that performance. The additional (hidden) cost of this uniqueness was offset by performance gains. Today, however, MIPS are cheap, users are buying shrink-wrapped software, and performance issues mostly center around interoperability. Thus, Unix, with its many unique dialects, has become the most unproductive part of the computer industry, and can be viewed as a giant social, public, and private make-work program. This uniqueness translates into higher costs throughout the system cycle—from development to support.

A Common Unix Environment

As the PC market matured into powerful business machines in the late 1980s, a new style of computing developed, which is now commonly referred to as client/server. PCs running DOS formed the client side of the equation, and workstations running more powerful Unices took on the server role. This scenario did not last long. Microsoft, mindful of the expanding server market and the advantages of controlling the operating system, introduced an operating system for the server: Windows NT. Although it is not in workstation vendors' interest to support NT because a freely licensed operating system will create a less differentiated, PC-like market, they have been forced to do so by users who welcome NT for its easier integration with Windows clients and less complex programming environment compared with Unix.

Thus, the fear of NT becoming a de facto, cross-platform standard has caused Unix providers to focus again on Unix standardization. This cry has been heard before. The last time was when AT&T and Sun declared in 1988 that they were standardizing Unix in

a closed fashion, causing the creation of the rival Open Software Foundation (OSF). This time, the standardization cry has manifested itself in an agreement between OSF and COSE (Common Open Systems Environment) on a common Unix environment, Spec 1170, which is an aggregation of all the past dialects and functions. We question whether this effort is likely to produce a standard Unix definition that includes all capabilities, rather than a subset like POSIX, which was created in the early 1980s.

Spec 1170 is aimed at providing an open environment such that a user can take a source application, recompile it, and run it on any other Spec 1170-compliant system. It is named after the 1,170 system calls and interfaces specified in the standard. Of course, many of the calls are redundant, which is acceptable. However, the standard must have 1,170 calls, not 1,171 or anything else that extends the standard without an open standards process.

By the year 2000, the major operating system alternatives in the client/server arena will be down to just a few Unix dialects and Microsoft's Windows and Windows NT.

It is not acceptable from a user's perspective to have a Unix dialect with 1,171 calls—or any superset/subset of calls—or a unique file system such as IBM's AIX version of Unix. Doing any of these renders the standard label meaningless and once again locks users into a platform and dialect. File formats must be identical across systems to avoid data conversion when an application and its data are moved to another system. Thus, file extensions, such as those that AIX uses, must be either removed or added to the standard. In addition, programs that use internal status information (e.g., counters and parameters) or reports must rely on identical names and formats because they can be used by, or piped, to create other programs.

Unfortunately, given the nature and ease of developing report formats (e.g., I/O status that later get "piped" to other programs), the likelihood of the various Unix vendors adhering to a single Unix standard is close to nil. The only way to have a single Unix standard is by agreeing to one source code that every platform vendor would use, and then appointing a standards body dedicated to its evolution.

Spec 1170 will allow Unix vendors to present a common interface standard that will enable common training and a large and profitable user applications market. But we believe that every Unix vendor will still want to maintain its own version (Unix dialect) in order to provide uniqueness. Unfortunately, maintaining multiple Unix dialects imposes higher costs in every part of the supply chain: Unix and platform vendors, applications vendors (particularly database vendors), and users.

The Spec 1170 initiative will not result in a "unified" Unix. However, it will—coupled with competition from Windows NT—begin to reduce the number of Unix dialects. As a result, we believe that by the year 2000, the major operating system alternatives in the client/server arena will be down to just a few Unix dialects and Microsoft's Windows and Windows NT, with a small legacy of Macintosh users. Thus, the days of many microprocessors and multiple Unix dialects to build proprietary computing environments are numbered.

The Unix "Tax"

Unix vendors have every right to fear Microsoft's encroachment into their traditional workstation server market. To understand why, we need look no further than Microsoft's \$700 million annual R&D budget, which includes development of not only its operating systems but also its applications, languages, CD-ROM products, and mice, as well as advanced technology and research. Contrast this with Unicee (not including compilers, etc.), which we estimate absorbs on the order of 10,000 programmers involved in purely engineering tasks at the 75 software and system suppliers. Thus, Unix vendors spend at least twice as much as Microsoft's total R&D budget (that is, more than \$1.4 billion) to maintain the many unique, proprietary dialects of Unix, resulting in an efficiency of the Unix development process of no more than 20%. This amounts to an extra R&D cost on platforms of at least \$1 billion, or a "sales price tax" of \$6-10 billion. In addition, each Unix vendor spends roughly the same amount as basic R&D on supporting Unix specific ports for transaction processing monitors, critical applications, and so on. Thus, R&D cost might be double the \$1.4 billion to reflect the true cost for Unicee uniqueness.

Unix is clearly one of the most significant, negative work amplifiers that the computer industry has invented. By attempting to maintain what fundamen-

tally *should* be an undifferentiated, standard product as a differentiated unique product across various hardware platforms, Unix vendors have caused downstream "make-work" in databases, networking applications, and higher-level applications. They have also created much of the middleware industry. The complexity this causes in the industry is graphically displayed in Figure 2. Each time a database, application, or user adopts another platform, a significant acquisition cost may occur depending on performance needs, the features required by the application, and the closeness to previous ports.

As we have shown, hardware providers maintain their own proprietary Unix to lock customers into a dialect and thus maintain their hardware margins. In most cases, having a unique dialect implies selling the software at or below cost to maintain hardware margins. This is the Unix "sales tax." However, the emergence of an operating system alternative to Unix will destroy this dynamic, and force hardware platforms to compete based on hardware alone, as is the case in the PC market. The pace of this evolution will depend on the strength of external pressures (e.g., Windows NT) and the rate of adoption of a single Unix dialect that runs on multiple platforms (e.g., Solaris).

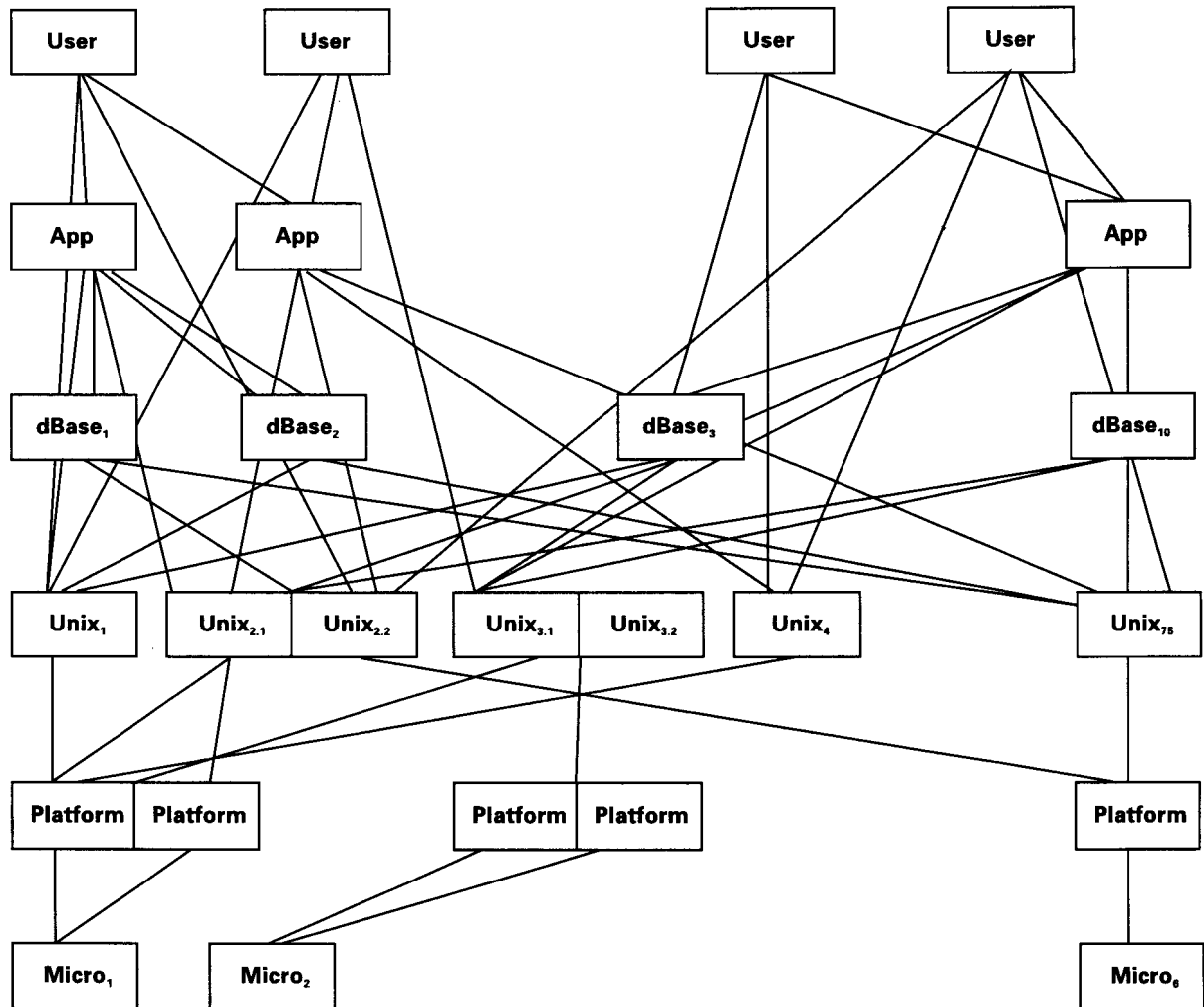
The fast-growing server market will become similar to the commodity-like PC market.

With a common version of Unix, it would be possible to significantly reduce the development cost and selling price, while supporting common platforms. A reasonable target for development of a common Unix operating system running across all platforms would be \$400 million, of which \$140 million is for the core development. We believe that a single source for Unix is the only way to technically achieve a true standard. Currently, the likelihood of this occurring is very low; however, as Intel-based products increase in performance to the point where servers are less distinct from multiple-processor PCs, the chances will increase.

A Proprietary Standard: Windows NT

An unconscious cartel of old line, "boutique" server vendors, made up of proprietary Unix platform and proprietary database vendors, has formed over the last decade. This cartel has been operating very profitably—underneath the IBM mainframe pricing um-

Figure 2
The Tangled Web of Unix Dialects, Platforms, Applications, and Users



Source: Gordon Bell.

bell—*as the downsizing market. However, Microsoft's Windows NT fundamentally destroys this market by enabling PC software to be "upsized" to various microprocessor platforms.*

Windows NT is a modern operating system supporting 64-bit addressing and 16-bit unicode data types while maintaining compatibility with PC hardware and Windows applications—probably the greatest challenge ever faced by software engineering. Furthermore, NT was designed for both real-time clients and multi-processor servers. Perhaps the most important aspect of NT is the users' ability to both install and operate it in a network environment that includes both NetWare and Unix.

Windows NT is a significant product in the computer industry and a direct challenge to Unix because it has

been ported to run on all six major PC/workstation hardware platforms, thus providing nearly all of the benefits of a single hardware architecture. As a result, the fast-growing server market will become similar to the commodity-like PC market. Utilizing NT, files on one platform can be brought directly to another platform because there is a single on-disk structure and little-endian byte orientation. This represents a major step over any of the Unices, even those that have been ported to several platforms and are bi-endian (can handle big- and little-endian byte orientation), because they are inherently not compatible among the on-disk structures.²

2. Unices do not have a common on-disk structure and, thus, must translate data rather than directly accessing records depending on the endianness of the processor.

Under Windows NT, users and independent software vendors can either run Windows applications directly with X86 emulation or recompile applications to run on any client or server environment. Thus, all applications are available on all platforms—applications for which developers need to maintain only one source code. NT provides a single API for both client and server, an essential requirement over the long term for computing environments. Additionally, NT is sufficiently kernel-based to support POSIX and the operation of any of the Unicee, which will enable it to run critical Unix applications that have been developed by users. Several companies support facilities to port applications from Unix dialects to NT.

Two other software approaches used to handle multiple platforms are worth mentioning. Cross-platform environments are created implicitly by porting databases to all common OS environments. Most server platforms are controlled by a proprietary database and a SQL dialect from Informix, Oracle, or Sybase. Each of these platforms, in turn, has applications tied to the database, thus creating more proprietary platforms.

The only meaningful standard is an industry one where the user can choose among hardware and software platforms from multiple vendors.

Microsoft SQL Server is designed specifically to run with Windows NT, thus giving it operational advantages over other SQL products that run on multiple OSs. This significantly lowers the cost of development, and simplifies installation and maintenance. In this fashion, both larger scale applications and a wide variety of applications can be provided that exploit the cost and performance of this directly supported relational database.

In contrast, explicit cross-platform environments, represented by Visix's Galaxy, Powersoft's Powerbuilder, and Next's OpenStep, allow users to develop distributed, client/server applications. They "cover up" the operating system by providing a common set of functions that can be called either as client or server, facilitating the construction of client/server applications. For example, Galaxy operates on the Macintosh, Windows, NT, the various Unicee, VMS, and OS/2. Taligent is also aimed at providing this functionality.

Another product (yet to be formally announced) that will provide cross-platform compatibility is TriTeal En-

terprise Desktop (TED). Backed by IBM, Sun, HP, and Silicon Graphics, TED is designed to allow developers to write to a common set of APIs for Unix. However, TED must be ported to each Unix environment and adds another layer to the operating system. Underneath TED will still be proprietary Unix dialects.

Surviving Open Systems

Based on the chaos that the Unicee have created, openness has become another name for proprietary entrapment and the resulting high cost. How do users today survive proprietary systems that are all called open? Is having an open or proprietary standard good or bad for platform suppliers, ISVs, or users?

The only meaningful standard is an industry one where the user can choose among hardware and software platforms from multiple vendors. The PC is currently the only platform that provides this environment by having many hardware providers and several operating systems that run Windows applications. This is possible only with a single source code.

Users can help enforce or set standards by applying the "power of two"; that is, by insisting on buying products from two vendors that adhere to specified standards. This ensures interoperability, compatibility, and a true standard. However, this strategy requires supporting and maintaining applications on two separate platforms, which will cost more initially. But it will save money over the life of the system as competition drives down prices. Furthermore, standards will converge rapidly. Applying the power of two can be done in a less costly fashion for large procurements by initially developing and testing the application in parallel on two platforms in order to characterize the vendors and their systems.

Applications Are the Key

The prospects for competing industry platforms ultimately depend upon the applications that are available on each. In turn, at least two factors affect the availability of applications, namely the cost to create and maintain unique versions, and the economics based on market size. For simple applications, it may be possible to maintain a single version. However, to optimize an application's utility, it is usually necessary to develop versions for each Unix dialect. (Cross-platform development environments for client and server, such as Galaxy, reduce this cost.)

Volume has a huge impact on software prices because the more units sold, the lower the amortization for fixed development and marketing costs.³ Developing multiple versions of a Unix-based application raises fixed costs and lowers the volume/version, resulting in higher prices. For example, the price of a relatively high-volume, document-preparation program, Frame, has come down from \$15,000 to \$1,500 on selective Unix platforms, but sells for \$400 on PCs.

Maintaining Multiple Platforms

The second factor that influences applications viability is the cost to maintain software on multiple platforms. This cost has two parts: design and customer support. For example, in the case of a database (e.g., DB2, Informix, Ingres, Oracle, Sybase), porting to multiple vendors' platforms (e.g., AT&T GIS, Compaq, DEC, IBM, HP, ICL, Olivetti, Pyramid, Silicon Graphics, SNI, Sun, Tandem, and Unisys) requires a group of 20 engineers (more or less) costing \$3 million/year. However, supporting the customer costs approximately \$10 million/year/platform. In addition, each company may be supplying other commercial applications that use their unique databases and, thus, have to interface to each of the Unix dialects.

Unix is being uprooted in universities by a combination of PC-literate incoming freshmen, a need for lower costs, and the reality that the commercial world has adopted PCs.

Databases are special applications because they are fundamentally just a complex file system consisting of three parts: the database engine (which manages data and represents approximately one-third of the code); an operating system (which schedules sub-processes and I/O requests and manages a cache); and an operating-system-specific interface. Because each Unix dialect has a unique internal structure, each database port is unique. As a result, database vendors spend most of their resources maintaining these versions instead of enhancing the database engine.

The benefits of having an application on a single operating system are clear (assuming volume is unaffected). For example, Microsoft SQL Server runs only on Windows NT, minimizing development costs. (Microsoft doesn't have to develop versions for other OSs.) There is a second advantage for Microsoft,

however: NT supports the database engine directly (in effect building it into the OS), which eliminates the operating system parts of the application, thus reducing the amount of code to maintain by a factor of two. It also reduces the amount of code executed in the critical path, resulting in significantly higher performance.

The single-platform advantages are vividly illustrated in Table 4. In a competitive benchmark set up by *Infoworld*, Oracle and Microsoft achieved comparable performance. However, the price of Oracle and SQL Net was nearly \$100,000 more than for Microsoft SQL Server, and SCO Unix was \$2,898 more than Windows NT Advanced Server. In addition, Oracle's hardware platform was more than \$10,000 more expensive because of additional memory, different SCSI controllers, and more disks. Thus, Microsoft's solution had an overwhelming cost advantage in both software and hardware.

Database companies spend approximately \$500 million on R&D. Maintenance of the various ports causes a development efficiency of about 35%, with more people working on porting than on the database. As Bob Epstein of Sybase has said: "If platform vendors delayed releases by 1 year, unprecedented productivity would occur in the software industry." If, on the other hand, database vendors had to support only one platform, prices could be reduced by a factor of 2.5.

Training: An Important Force

The final and perhaps most important factor that influences platform choice is the ability to acquire trained personnel. IBM, DEC, and Sun influenced future generations of users by placing computers in universities. Without the adoption of Unix by universities, system programmers and users could not have been trained.

Today, however, users are trained in homes, schools, and offices—mostly on PCs. This universal training combined with low-priced PCs has moved the universities from Unix platforms to the PC. Unix is being uprooted in universities by a combination of PC-literate incoming freshmen, a need for lower costs, and the reality that the commercial world has adopted PCs. A Fortune 500 CIO told us: "Why would I run Unix that requires a college graduate in computer science to support each version, when I can send a high school

3. For a more detailed discussion of the economics of software volume, see "Future Computing Environments: The Commodity Mainframe Era," *Spectrum, Information Systems Industry*, Issue 55, 1993.

Table 4
Comparative Costs: Microsoft SQL Server and Oracle

Oracle Implementation		Microsoft Implementation	
Oracle 7.1 for SCO	\$ 80,000	SQL Server 4.21A	\$ 777
SQL Net	25,000	50 client licenses	5,800
Pro C precompiler	240	Programmer's toolkit	449
SCO Unix	3,446	Windows NT Advancer Server 3.5	548
FTP PC/TCP2.2 for DOS	5,950	Compaq ProLiant 4000	<u>31,846</u>
Compaq ProLiant 4000	<u>44,255</u>		
Total	\$158,891		\$39,420

Source: InfoWorld.

graduate to a 2-week course to support NT?" In the years to come, new entrants to the workforce increasingly will be PC-literate. Concurrently, the supply of Unix programmers will diminish.

Unicee Versus NT

We have presented a future scenario based on a radically different standards situation than exists today, although the signs of this evolution are clear. Table 5 shows our perception of the major vendors' positions vis-à-vis Unix and Windows NT. Fundamentally, it is a race against time for the open standard Unicee against a proprietary standard Windows NT. The eco-

nomics of volume are almost certain to whittle down the number of Unicee survivors as users migrate to NT.

The transition from Unix is the inevitable result of the difficulty of implementing a single Unix with a single source code; the high cost and inefficiency to applications vendors and users of maintaining Unicee; and the sea of low-cost, PC servers with built-in SQL that allows users to avoid the database "tax."

With a prosperous downsizing market driving sales, server companies (e.g., AT&T GIS, DEC, HP, IBM, Pyramid, Sequent, Sun, Silicon Graphics, Stratus, Tandem) are not eager to go through the turmoil that this transition will cause. Yet companies predicating their

Table 5
Unicee Versus NT

Unicee Advocates	Fence Straddlers	Windows NT Advocates
Sell Unicee and other proprietary environments. Need NT to fail to maintain monopolies!	Sell both PCs & Unicee. Have to get along with Microsoft. Maintain status quo and margins as long as possible.	Have no platform software. Single standard reduces support, drives high volume, and users get lowest prices.
Amdahl, Apple, Auspex, Bull, Convex, Cray, IBM, Informix, Lotus, Ncube, Novell, Oracle, Pyramid, SGI, Sun, Stratus, Sybase, Tandem.	AT&T GIS, DEC, Fujitsu, Hitachi, HP, ICL, Motorola, NEC, Netframe, Olivetti, Sequent, SNI, Unisys	X86-based vendors such as Acer, Compaq, Dell, Gateway, Intel, and Netpower.
		Application developers who dislike proprietary databases.
		Users who want lower prices, a more robust system, and more applications.

Source: Gordon Bell.

future on PC-sized, standard servers (e.g., Compaq, Intel, Netframe, Tricord) and traditional PCs that every major vendor (e.g., DEC, HP, IBM) sells are certain to upset this status quo. The perception of the current immaturity of NT is the main factor holding back this change. But, by the time a single, competitive Unix is available, it is certain to be too late to stop the momentum to NT.

About the Author

Gordon Bell is a computer industry consultant at large. He spent 23 years at Digital Equipment Corporation as vice president of research and development, where he was the architect of various minicomputers and time-sharing computers and led the development of Digital's VAX and the VAX environment. Mr. Bell has been involved in, or responsible for, the design of many products at Digital, Encore, Ardent, and a score of other companies. He is on boards at Adaptive Solutions, Chronologic Simulation, Cirrus Logic, Kendall Square Research, Microsoft, Visix Software, University Video Communications, Sun Microsystems, and other firms.

Mr. Bell is a former professor of computer science and electrical engineering at Carnegie-Mellon University. His awards include the IEEE Von Neumann Medal, the AEA Inventor Award, and the 1991 National Medal of Technology for his "continuing intellectual and industrial achievements in the field of computer design." He has authored numerous books and papers, including High Tech Ventures: The Guide to Entrepreneurial Success, published in 1991 by Addison-Wesley. Mr. Bell is a founder and director of The Computer Museum in Boston, Massachusetts, and a member of many professional organizations, including AAAS (Fellow), ACM, IEEE (Fellow), and the National Academy of Engineering.

Eric P. Blum, Research Program Manager

95-11-70

About Decision Resources, Inc.

Decision Resources is an international publishing and consulting firm that evaluates worldwide markets, emerging technologies, and competitive forces in the information technology, life sciences, and process industries. Decision Resources links client companies with an extensive network of technology and business experts through consulting, subscription services, and reports. For additional information, please contact Marcia Falzone by phone at (617) 487-3749 or by fax at (617) 487-5750.

A DECISION RESOURCES publication © 1995 by Decision Resources, Inc.

SPECTRUM is a trademark of Decision Resources, Inc. DECISION RESOURCES is registered in the U.S. Patent and Trademark Office.

This material, prepared specifically for clients of Decision Resources, Inc., is furnished in confidence and is not to be duplicated outside of subscriber organizations in any form without our prior permission in writing. The opinions stated represent our interpretation and analysis of information generally available to the public or released by responsible individuals in the subject companies. We believe that the sources of information on which our material is based are reliable and we have applied our best professional judgment to the data obtained. We do not assume any liability for the accuracy, comprehensiveness, or use of the information presented.
