Introduction to Systems Application Architecture

by E. F. Wheeler
A. G. Ganek

Systems Application Architecture is a framework in which applications are developed so that they run consistently on major IBM computing systems. This paper presents the motivation and requirements for this framework and describes the main elements of its structure. It also discusses the effect on current processing technologies and on application development.

In March 1987, IBM introduced Systems Application Architecture (SAA), a significant new direction for IBM software which provides the framework for the development of consistent applications across the major IBM computing environments—System/370, AS/400™, and Personal System/2®. This announcement is not about a specific product, but rather a pervasive software architecture that underlies the commitment to provide, in an evolutionary way, cross-systems consistency across a broad spectrum of hardware, architecture, and operating systems environments. It is a software-based approach to present the breadth of IBM's product line to its customers as a family of software systems—a family that will provide enterprise-wide solutions to business computing problems.

SAA is composed of three significant elements—Common User Access, Common Communications Support, and Common Programming Interface—that govern software interfaces, protocols, and conventions for human interaction with applications and system services, communication mechanisms that interconnect SAA systems, and programming interfaces for program development. Many of the specifications that describe these elements have already been published, and more will follow as SAA evolves and expands over time. The specifications are open and available to encourage customer and software-vendor participation in the development of applications that adhere to SAA. In addition, a fourth key element of the SAA strategy is IBM's own commitment to provide Common Applications that will execute across the SAA system environments. As customer, software-vendor, and IBM SAA applications become available, the new dimension of consistency among them will augment the end user's access to applications of all kinds.

SAA is a key strategic direction for IBM, analogous to the System/360 announcement in 1964 that introduced a family of hardware systems. SAA defines a family of software operating systems whose goal is to meet the following customer needs:

- Increased programmer and end-user productivity
- Enhanced ease of use and support by providing consistency among applications
- Improved communications capability and usability for enterprise-wide solutions
- Increased return on customers' information systems investment via greater leverage of programmer resources and user experience

These needs are to be met by providing consistent, durable interfaces in IBM software products.

This paper is intended to illuminate the concept of SAA, demonstrate the motivations for this direction, and explain its overall structure. Subsequent papers in this issue will enlarge upon selected aspects and components of SAA and highlight the technical challenges, trade-offs, and solutions involved in the implementation of this architecture.

Background

Perhaps the easiest way to understand how cross-systems consistency and Systems Application Architecture will meet customer requirements and remove constraints to their growth is to present an historical perspective on the evolution of IBM software.

Use of application enablers became widespread.

In the mid-1950s, the programs that existed were primarily application programs, executing the functions for which a computer was acquired. The hardware manufacturer provided the hardware, and the customer, with the aid of rudimentary operating systems and assemblers, wrote the software. Since the application programs ran on the hardware directly, they were tied to the instruction set or "architecture" of the hardware. By the late 1950s and early 1960s, hardware-oriented system software began to emerge. This software, which was code that could be shared by all customers, consisted primarily of input-output routines. The routines operated card readers, punches, printers, tape devices, and, toward the end of this period, disk storage. In addition, high-level languages, notably FORTRAN and COBOL, were introduced and offered the potential to ease greatly the task of programming. The net effect was to improve the productivity of application developers. But, for the most part, the applications of this period remained directly related to the specific hardware system on which they were running.

During the mid- to late 1960s, the function of operating system software increased as newer technology, such as larger memory, became more available and as the complexity of the hardware increased. These developments spawned the widespread use of the class of software we now call application enablers, including the high-level languages and a variety of other programming services. This software was a big step forward, since applications developed in high-level languages did not have to deal directly with the details of the hardware. But most large applications required additional, machine-specific support outside these languages and so were still directly dependent on the underlying hardware. The advances in application enablers, combined with the enhancement of operating system software to provide sophisticated "batch processing" for automated job scheduling, print spooling, and other aspects of shared resource management, comprised the first major stage in the evolution of operating system technology.

By the 1970s, the use of display terminals to access computing resources and data stored on line had become common, and technology had also made communication possible between systems. To support these changes, system software increased in scope and provided interconnect facilities to tie systems together. This, in turn, gave customers the opportunity to bring key aspects of their businesses on line and interconnect processors throughout their businesses. Database and data communication products were the leading technology advances of this second stage of operating system evolution. Together with interactive support for program development, this stage made the capabilities of computer systems directly available to terminal operators. Computer resource sharing now had an on-line, interactive interface that was extended to non-data-processing professionals as well as to industry personnel.

Today, technology has progressed to the point where the need for application-enabling software is recognized and provision has been made for it. Such enabling capability includes management of data in relational form, application-generation products, and display management, to name just a few. The result is that application programs have become largely insulated from the underlying software and hardware. With the use of this application-enabling technology, the effort expended in creating an application is, by and large, directed at solving the problem, rather than fitting the solution to a particular hardware architecture or operating system.

The structural changes just described occurred throughout IBM's product line. In the mid-1970s, each computer family and its operating system as a pair, along with the associated set of application enablers, were optimized to a particular purpose, in terms of criteria such as performance, capacity, and application environment (i.e., batch, interactive, or transaction processing). Each was designed and built to be the best-of-breed for its intended goals. Collectively, they allowed IBM to move forward to compete across a broad range of markets by satisfying customer requirements at each point in the range. This strategy proved to be successful and was well-accepted in the marketplace.

By the early 1980s, the span of computing power from the smallest to largest machine in the IBM product line had grown drastically. To provide competitive solutions across this increased range, several new environments were added. Today, in all of the major environments except the two smallest, personal systems and the System/36, the power of the hardware and the structure of the software have progressed to the point where application enablers provide levels of function that are roughly equal and that have, by and large, insulated the applications from the hardware, although they remain tied to the operating system on which they run.

In the case of the System/36, the structure of the software has reached the same point of evolution as the others, but the small size of the system has meant that the depth of function has certain limitations. As an example, a relational database management system is not available. In the IBM Personal Computer Disk Operating System (PC DOS), neither the structure of the software nor the depth of function is equivalent to what is found in the larger systems. Applications on PC DOS are still largely tied to the hardware, and some advanced application enablers are either not available or limited in function.

During the latter part of this evolution, the emergence of Systems Network Architecture (SNA) permitted the interconnection of any combination of these systems and communication with a variety of third-party equipment. SNA also brought the capability to more easily manage extremely large networks of interconnected systems. The benefits of interactive computing, coupled with the connectivity of these networks, provide significant capability for customers to bring together diverse aspects of their businesses for access from a single terminal anywhere within the enterprise.

In contrast, because the application enablers for each system have evolved independently and have been optimized for the system on which they run, applications that use the enablers have also become segmented by operating system. This condition limits the utility and increases the complexity of interconnecting heterogeneous systems because of the disparity of the application environments across systems. If the needs of an installation grow beyond the capacity of a particular system, this segmentation makes it difficult to migrate up the product line. The same barrier applies to the migration of applications that run on large systems when the need arises to replicate them on smaller, departmental systems. The result is that the IBM product line appears as a number of independent systems, each one offering advantages within its domain.

As we look forward to the end of this decade, hardware technology is expected to continue to improve at its historical rate of 20 to 25 percent per year. Clearly, it is IBM's goal to continue to position the product line to exploit fully this enormous range of computing capability. As a result of this growth, we have arrived at a key point in the evolution of systems software, one that affords the opportunity to integrate key systems across the entire spectrum of processor power.

Technology has progressed to the point where we can provide a full-function system, Operating System/2™ (OS/2™) Extended Edition, on the current line of intelligent workstations. This development will enable the graphics power of the intelligent workstation to be combined on one system with the full set of operating system functions such as database and communication facilities. The System/36 and System/38 are replaced by a new, unified system, the OS/400™ operating system and AS/400 hardware, that has the capability to support from four workstations to more than four hundred, depending, of course, on the workload and configuration. This system combines the ease of use of the System/36 and the advanced software technology of the System/38, together with state-of-the-art hardware for enhanced performance and capacity.

These developments are undeniably important for the small and intermediate computing environments. But it is equally important that, for the first time, there will be a full set of application enablers on all of our systems. By using the application enablers to mask the underlying hardware and operating system, the product line can be presented to cus-

tomers in a single coherent way from the smallest intelligent workstation to the largest System/370 machines as a single family. Application developers can begin to develop common applications that run on each system of the family, rather than being limited to a particular operating system or hardware architecture, thereby minimizing the effort required to build such applications and to migrate users from one system to another.

SAA defines a consistent set of application enablers.

To make this happen, SAA defines a consistent set of application enablers that span the systems and minimize the historical differences, making these application enablers the unifying force for the future.

The establishment of consistent application enablers across a family of operating systems goes well beyond the goal of portability of applications from one environment to another. When this consistency is combined with a rich set of communications capabilities and protocols for the interchange of data and for process synchronization, the family-of-operating-systems concept extends to a global environment in which interconnected systems concurrently participate in addressing the needs of the enterprise. The result is called an *Enterprise Information System*, which represents the third major stage of operating system evolution. Such an environment is a distributed system, and the programming functions that enable it are referred to as distributed services. The implication of such a capability is that any system in the family can interact with any other system by using the range of distributed services. Distributed applications are those written to exploit multiple system configurations to satisfy a variety of requirements, including specialized processing needs and those motivated by geography, security, capacity, availability, and organizational considerations. Within this context, the technological advances in power and improvements in price/performance of personal computers that enable a full-function operating system environment on a personal computer today also offer a full realization of the potential of cooperative processing. In cooperative processing, intelligent workstations are used in conjunction with host systems in an integrated way to provide the end user with the advantages of both personal and host-based computing. Cooperative processing allows the user to have a single, seamless view of the function of an application, whereas, in fact, the implementation is split between the workstation and the host. This implementation model is of key importance to SAA and will be addressed in more depth later in this article.

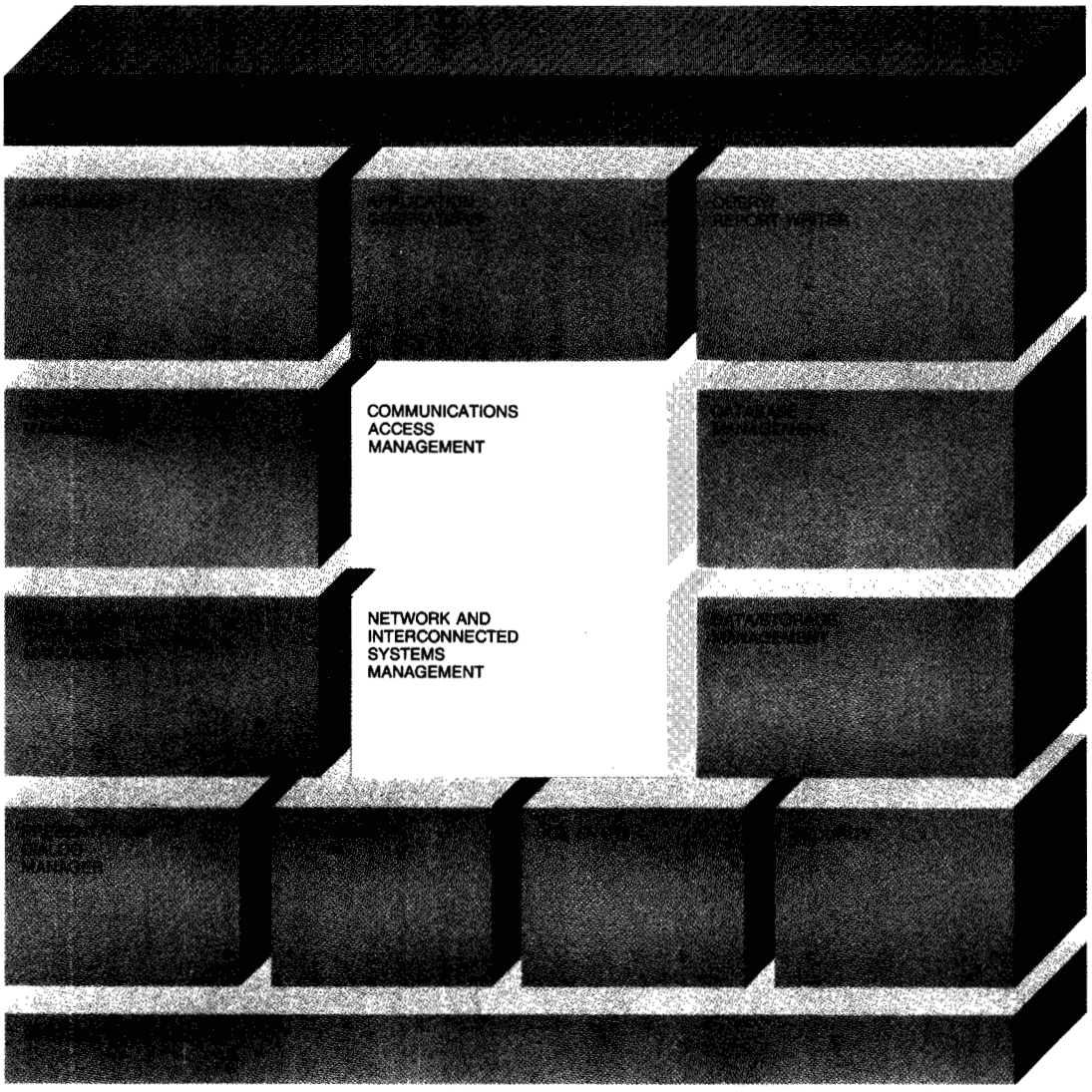
Requirements

To support a product line that spans an ever-increasing capacity range, more than one hardware architecture and operating system will be required. Exactly how many are required and what part of the range each can span may change over time as technology changes. Built on each of these architectures and operating systems will be a set of application enablers matched to its system. When one considers the IBM product line, the broadest in the industry, ranging from the IBM PC to the 3090 (which represents a factor of about 1000 in performance), it is evident that multiple hardware architectures and operating systems are necessary. They are necessary today to allow customers to exploit this wide range of capacity, and they will continue to be needed to exploit the expanded capacity of the future—capacity driven by technology improvements that, as indicated earlier, continue at rates in excess of 20 percent per year. Hardware architectures and operating systems have natural limits, both upward and downward, and when forced to operate beyond those limits, do so either poorly or with difficulty.

In IBM, the multiple hardware architectures and operating systems are well-positioned to take advantage of the progress of technology. We face a different problem: how to present that technology in a consistent way. Consistency across the product line can be achieved by making the interfaces of the software consistent across however many implementations are required to span the range. This mechanism works because software technology is now advanced enough to have the interfaces for application enabling independent of the hardware, even on the smallest and largest systems.

The guidance necessary for defining an architecture that meets the challenge of consistency across the

Figure 1 General software structure



product line can be obtained by identifying the key requirements for productive use of heterogeneous data processing systems. Numerous IBM studies, customer advisory councils, and user group meetings, as well as discussions with software vendors, have repeatedly determined three leading requirements in this area:

- Usability—the need for applications to be easier and simpler to use and to learn
- Productivity—the need for a straightforward and productive way to develop applications that operate across a variety of operating systems and thereby ensure a wider range of usefulness, obviating the necessity of rewriting applications to meet the demands of different environments
- Connectivity—the ability to connect systems and peripheral equipment in an easy and consistent way, supported with the tools necessary to manage the interconnected environment simply and efficiently

Structure

Systems Application Architecture provides consistency across dissimilar operating systems. The cornerstone of the definition of such an architecture is a conceptual model that describes software structure in a generic way and provides a coherent organization of function. This model, shown in Figure 1, consists of four layers, each of which represents a set or category of related functions. This structure provides a foundation for understanding the strategic elements of IBM software and how they fit together. The blue, or bottom, layer in the figure is the software uniquely related to the hardware. This software is geared to manage and exploit the capabilities of the specific hardware and architecture for which it is designed. Included in this layer are functions that manage physical system resources such as memory, disk storage, printers, and processor dispatching. The yellow layer represents communications software, which provides the connectivity to allow communication among systems and applications and the ability to manage the interconnection of systems. This software entails communication protocols and network management. The green layer represents the products directly related to writing and executing applications, also called the application-enabling products. Included in this category are services such as database, query and report writing, and transaction management, along with languages such as FORTRAN and COBOL. At the top, the red layer represents application software such as office systems, manufacturing automation, health-care systems, and thousands of others that deliver the solutions to problems for which computers are used.

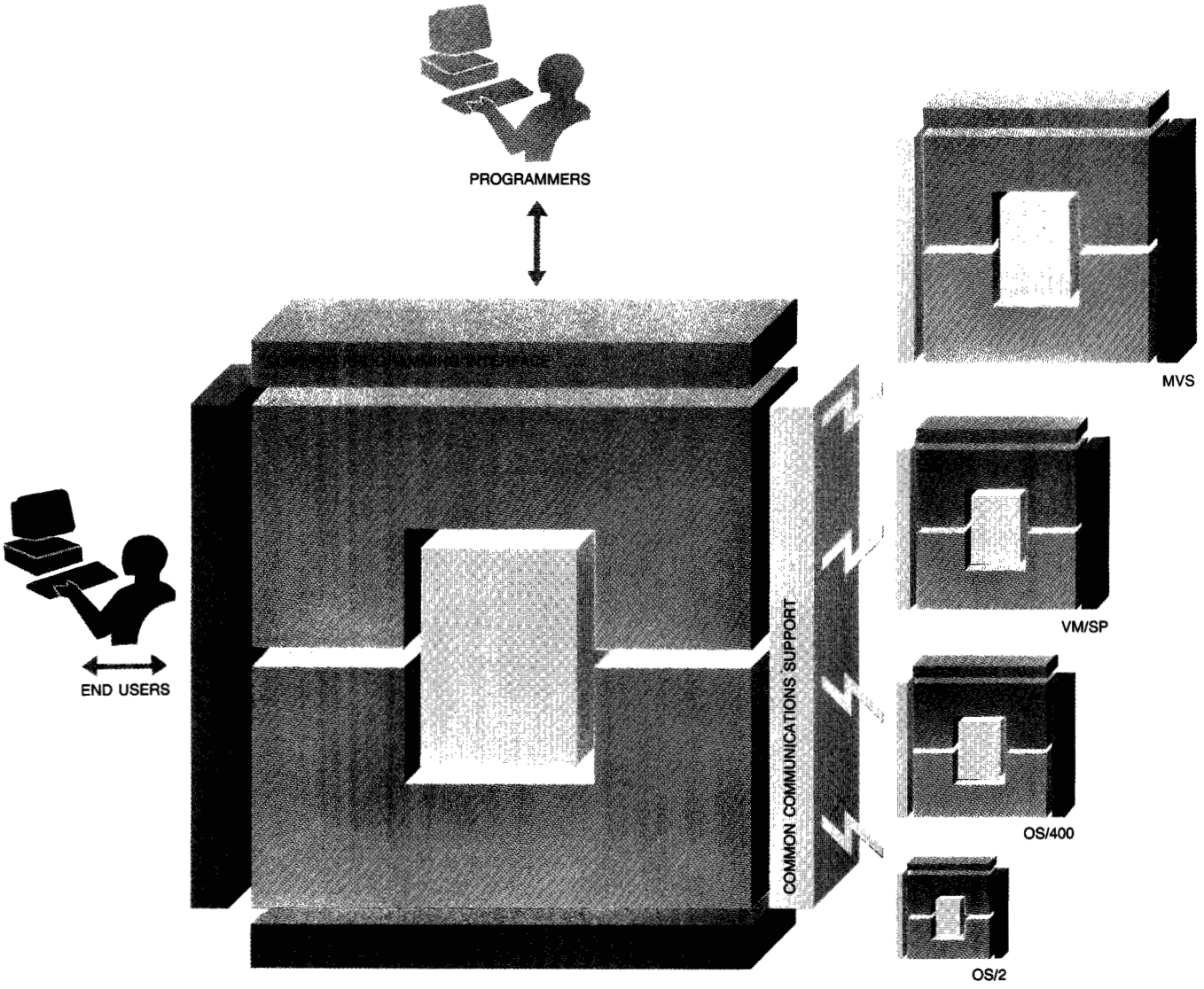
This software model, which began as an organizational tool for high-end systems, has been mapped against each of IBM's major operating systems: Multiple Virtual Storage (MVS), Virtual Machine/System Product (VM/SP), OS/400, and OS/2 Extended Edition, and found to be generally applicable. The host system environments of MVS, VM, and OS/400 support all of the key functions of the model, although in many cases with different interfaces. OS/2 supports most of these functions; however, some either are not yet supported at this time or are not applicable, such as Job Entry or Data Center Systems Management. Despite these omissions, the model still applies to OS/2 Extended Edition, since the majority of functions are present, and the components it does have can be structured according to the blue-, green-, and yellow-layer approach. The model not only simplifies the classification of products within the various systems, but provides a basis for defining common elements across these operating systems to yield a common set of interfaces for application programs. The idea of cross-systems consistency was developed from this concept of a common framework giving rise to common interfaces.

SAA uses the generic software structure model as a basic building block to address the three key requirements for cross-systems consistency. It does this by controlling three new specifications that relate to the requirements for consistency on a one-to-one basis and also by building directly on the system structure model, as shown in Figure 2. As can be seen in the figure, the SAA components surround the software structure in such a way as to provide consistent user, programming, and communications access to the operating systems functions of each system—MVS, VM/SP, OS/400, and OS/2 Extended Edition.

SAA is controlled by the specifications of the following three elements: Common User Access, Common Communications Support, and Common Programming Interface. They are discussed below.

Common User Access. The first component governs the end-user interface and is called Common User Access (CUA). This interface controls how the system, including the applications, interacts with a person at a workstation or terminal. It is this interface that ensures that the way things look to the user and the actions required of the person by the system are

Figure 2 Consistent interfaces built on system structure model



familiar no matter what tasks are performed. By providing a well-designed end-user interface for application programs, applications are made easier to learn and easier to use. By making the end-user interface consistent across applications and across systems, skill acquired with one application or on one system is usually transferable to other systems and applications. CUA provides the window through which the user views and accesses applications. The enhanced ease of use and the consistency it provides will augment the number of applications readily available to the user.

CUA is defined for the interaction between humans and computers. In dealing with how the machine communicates with people, it addresses what the computer presents to the workstation operator, including screen layout, use of color and highlighting, messages, and help information. Also important to CUA is how the user communicates with the machine, which involves the keyboard layout and usage, the use of a mouse, scrolling, and selection mechanisms. CUA is intended to provide a consistent and highly usable framework for the dialog between the person and the machine that will result in continuity among different applications and across the four SAA systems. The guidelines and rules are developed to enhance ease of use, and the continuity achieved by pervasive application of these conventions will enhance ease of learning. CUA is designed to take full advantage of the capabilities of the intelligent workstation, which has the greatest capacity for a highly interactive user interface. Dependent workstations are also supported in the specification; however, the limitations of this technology will restrict the use of some of the dynamic features available on intelligent workstations. CUA allows a framework of consistency to exist between these different kinds of workstations, while still encouraging full utilization of the potential of the intelligent workstation.

Common User Access is discussed in greater depth in the paper in this issue by Berry.[1]

Common Communications Support. The second SAA element controls the interconnect protocols and is called Common Communications Support (CCS). This interface deals with how systems work together to accomplish a job. For example, it controls how systems communicate with one another to store, retrieve, and move information through the communications network. With consistent interconnect implementations provided by CCS, customers can build networks of systems with vastly differing capacities and readily share and exchange data among them. These protocols will continue to evolve using portions of SNA and international standards.

Included in CCS are the following facilities:

- Data streams—Data streams govern the way in which data are handled and formatted when transmitted over a communications link. They are provided in SAA for support of display devices, document text, and printers.
- Application services—These services relate to services that enhance the function of the network. Examples include distribution services that enable asynchronous distribution of data throughout the network, protocols that define common office functions such as document interchange, and network management.
- Session services—These services reflect the SNA Logical Unit (LU) Type 6.2 Advanced Program-to-Program Communications protocol, which defines a rich set of communications services along with a consistent programming interface on each of the SAA systems.
- Network—The network facility applies to Low Entry Networking Nodes (Type 2.1 Nodes), which support peer-to-peer communication across nodes in the network.
- Data link controls—These controls provide link usage and management disciplines such as Synchronous Data Link Control (SDLC) for teleprocessing links, Token-Ring for local area networks, and X.25 for packet-switched networks.

The subject of Common Communications Support is explored in detail in the paper by Ahuja in this issue.[2]

Common Programming Interface. The third element with which cross-systems consistency is concerned is application enabling, addressed in SAA by the Common Programming Interface (CPI). It specifies how a programmer is to write and attach a new application to IBM's family of SAA systems. The application writer uses this interface to exploit the power of the system. Having such an interface allows applications to be independent of the underlying system and, therefore, to run on any system of the IBM SAA family, maximizing the return on investment in application code. It is also valuable because an application programmer can apply skill at using the CPI across the whole IBM SAA family—not just a single system—vastly increasing productivity. This component of SAA defines a set of application building blocks consisting

of languages and programming services for application programmers. It already contains a rich set of published interfaces that will expand over time to include an even broader spectrum of interfaces for application enabling.

The CPI language set provides consistent implementations of the most widely used languages that are applicable across the SAA system spectrum. The scope of the language set encompasses high-level languages, procedures languages, application generators, and expert systems. The number of products in this category is increasing as the SAA requirements broaden. Included at this time are the COBOL, C, FORTRAN, and RPG high-level programming languages; the REXX procedures language; and the Cross System Product (CSP) application generator. Although not supported at this time, expert systems technology is planned for inclusion in SAA in the future.

Cooperative processing can now be employed cost-effectively.

The services of the CPI provide key enablers for the development of portable and integrated applications. Central to the SAA direction is the relational database accessed via the Structured Query Language (SQL) standard database language. Complementary to the relational database is the Query Interface that is usable by end users and programs to access relational data. The Dialog Interface provides a high-level, programmable capability for defining and displaying terminal screens that conform to the Common User Access specification. Mechanisms are provided that control the relationship between variables in the program and fields and selections as portrayed on the screen. Navigation among multiple panels is supported. The Presentation Interface enables a lower-level control of screens and printers for text and graphic format display as well as multiple-window support; it is used in the implementation of the Dialog Interface. The Common Programming Interface for Communications (CI) provides a consistent, high-level programming interface for the LU Type 6.2 protocol for program-to-program communication.

The role played by the Common Programming Interface in SAA is elaborated upon in the paper by Wolford in this issue.[3]

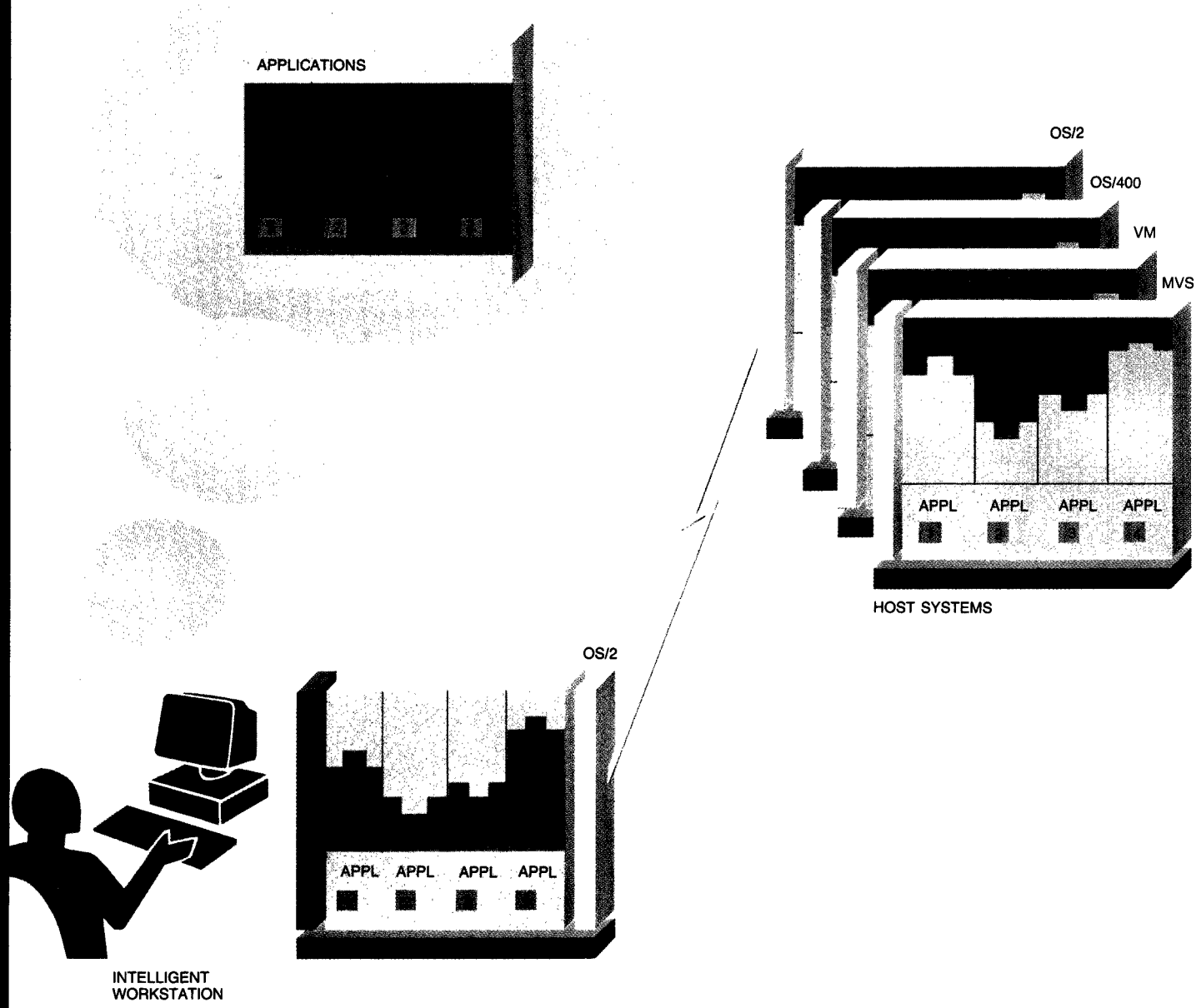
Cooperative processing

As indicated earlier, the advance of technology has now permitted the cooperative processing model to be employed cost-effectively for a wide variety of applications. Use of this model has many benefits and is a key aspect of the SAA direction. The use of an intelligent workstation as an integral part of an application makes it possible to provide the application user with the most advanced and usable human-to-machine interaction available in information systems technology. This capability includes advanced screen-display techniques and windows, keystroke-initiated processing interactions, and high-performance use of graphics. By using these capabilities in conjunction with host-system services, the application can afford to offer the user access to shared files, databases, transaction services, specialized computing functions, and peripheral equipment such as printers and plotters. Because the user perspective of the workstation/host interaction is made transparent, the image of a single application is preserved, as depicted in Figure 3. Advanced communication support, the Common Programming Interface for Communications, and multitasking in OS/2 Extended Edition allow workstation users to concurrently utilize local applications that are written for the personal computer and cooperative applications that exploit host-system services in a consistent way, with easy transitions from application to application.

Distributed processing

As we have seen, cooperative processing makes it possible to present an integrated and seamless view of intelligent workstation and host-system capabilities. Advanced Program-to-Program Communication (APPC) capabilities, as defined by the SNA LU 6.2 protocol and accessed via the CI, provide a consistent way to interconnect all of the SAA systems. This interconnection allows application function to be split across intelligent nodes in the network in a general way, providing for the distribution of function across multiple host systems as well as intelligent workstations. Consistent implementations of CI per-

Figure 3 Cooperative processing



mit distributed application functions to be written independently of the system on which they will execute, allowing function to be distributed according to the requirements of an enterprise and redistributed as necessary when those requirements change to accommodate growth, reorganization, and consolidation.

Built on the APPC base and the additional capabilities of the Distributed Data Management (DDM) architecture, generalized distribution of data throughout a network of heterogeneous systems can be achieved. Distributed data means that files and databases can be dispersed throughout the network, yet are accessible by any program as if located on the local system. The remote location of the data is thus transparent to the application program. The application program invokes the same interface for SQL or high-level language record I/O operation regardless of the location of the data, and the appropriate communications processing is implicitly generated in the case of remote data. The communications processing takes advantage of the CCS data formats, session protocols, network management, and data link control facilities.

Distributed relational data and distributed files are discussed in more depth in this issue in the papers by Reinsch[4] and Demers.[5]

Application development

The value of applications that comply with SAA is enhanced because those applications

- Are easier to learn and use as a result of the CUA
- Execute in a broader range of processing environments as enabled by the CPI
- Interconnect that range of systems as needed via the CCS
- Utilize advanced technologies such as relational data and cooperative and distributed processing

Attainment of these benefits depends on the creation of applications that take advantage of these features. A critical aspect of the strategy to support SAA is to provide a set of tools that will greatly improve productivity in the development process for such applications.

The program development process normally consists of a sequence of stages, including requirements gathering, analysis and design, code development, system integration and test, and maintenance. This sequence reflects more than just the construction of an application program; it is the entire life cycle from conception and definition of the requirements to production usage and ongoing maintenance. Major improvements in productivity necessitate not only enhancements to each phase of the process, but also a more comprehensive approach to the problem as a whole. This approach in SAA will emphasize an integrated family of tools which provide an application development environment to share programming objects throughout the life cycle. A key element in such an approach is a common data repository affording advanced features for storing and retrieving such objects, for describing their attributes, and for defining relationships among them. These capabilities would allow information related to an application to be created once and be shared throughout the product life cycle.

The application development environment is an excellent candidate for utilizing cooperative processing to achieve the best possible ease of use and to exploit the value of local processing power in an intelligent workstation wherever applicable. Host facilities would be likewise used where appropriate, such as for the data repository.

Common applications

The manifestation of the SAA concept is the set of applications that exploit its principles to achieve its objectives. The characteristics of these applications, called Common Applications, are directly derived from the principles of cross-systems consistency that we have explored in this paper. Such applications execute on all the SAA operating systems, providing consistent function across a vast range of computing power. They adhere to the CUA specification to interact with people in a highly usable and consistent way. Similarly, where applicable, they utilize the CCS to effect interactions from system to system. They are also based on the CPI, which enables consistent function and implementation. Foremost design considerations are integration and extensibility across the breadth of the enterprise they serve. Cooperative processing is emphasized wherever applicable to enhance the function, usability, and integration of the application.

IBM is committed to deliver many such Common Applications targeted to support specific industry requirements as well as cross-industry needs. An example of the latter is Office and Decision Support, which is analyzed in a case study in this issue by Dunfee et al.[6]

The objectives and value of Common Applications are as suitable for IBM's customers and independent software vendors as they are for IBM. The emphasis on an open, published set of interfaces, protocols, and conventions is designed to encourage customer and vendor participation.

As SAA evolves, the support of the Enterprise Information System will continue to expand.

Enterprise Information System

Cooperative processing, advanced communication facilities, distributed data, and Common Applications permit an end user, through the window of the intelligent workstation, to access all of the capabilities available in the network to which that workstation is attached. Not only are diverse capabilities accessible, but the complexity of the interconnections is not apparent. This environment, conceptually depicted in Figure 4, is the Enterprise Information System, a central theme in SAA. The objective is to provide enterprise-wide solutions to business problems by extending the scope of applications to span multiple systems, from workstations to midrange systems to mainframes, that may be geographically dispersed. This objective allows an enterprise to install and configure information processing systems according to the needs of the business, including organizational, geographic, and historical factors, while still achieving the integration so vital to productivity and effectiveness.

As SAA evolves, the support of the Enterprise Information System will continue to expand. The key to the success of this direction is the cross-systems consistency foundation:

- Consistent user interface as defined by CUA to allow consistent human interaction with information processing facilities across the entire scope of the enterprise, enhancing ease of use and knowledge transfer
- Flexible connectivity of heterogeneous systems as provided by CCS to allow the concurrent use of the variety of computing capabilities in the enterprise in an integrated and complementary way
- Consistent programming interfaces as provided by the CPI to allow application development to be independent of the system selection, programs to be portable from system to system, and programming skill expanded to be applicable across the systems in the enterprise

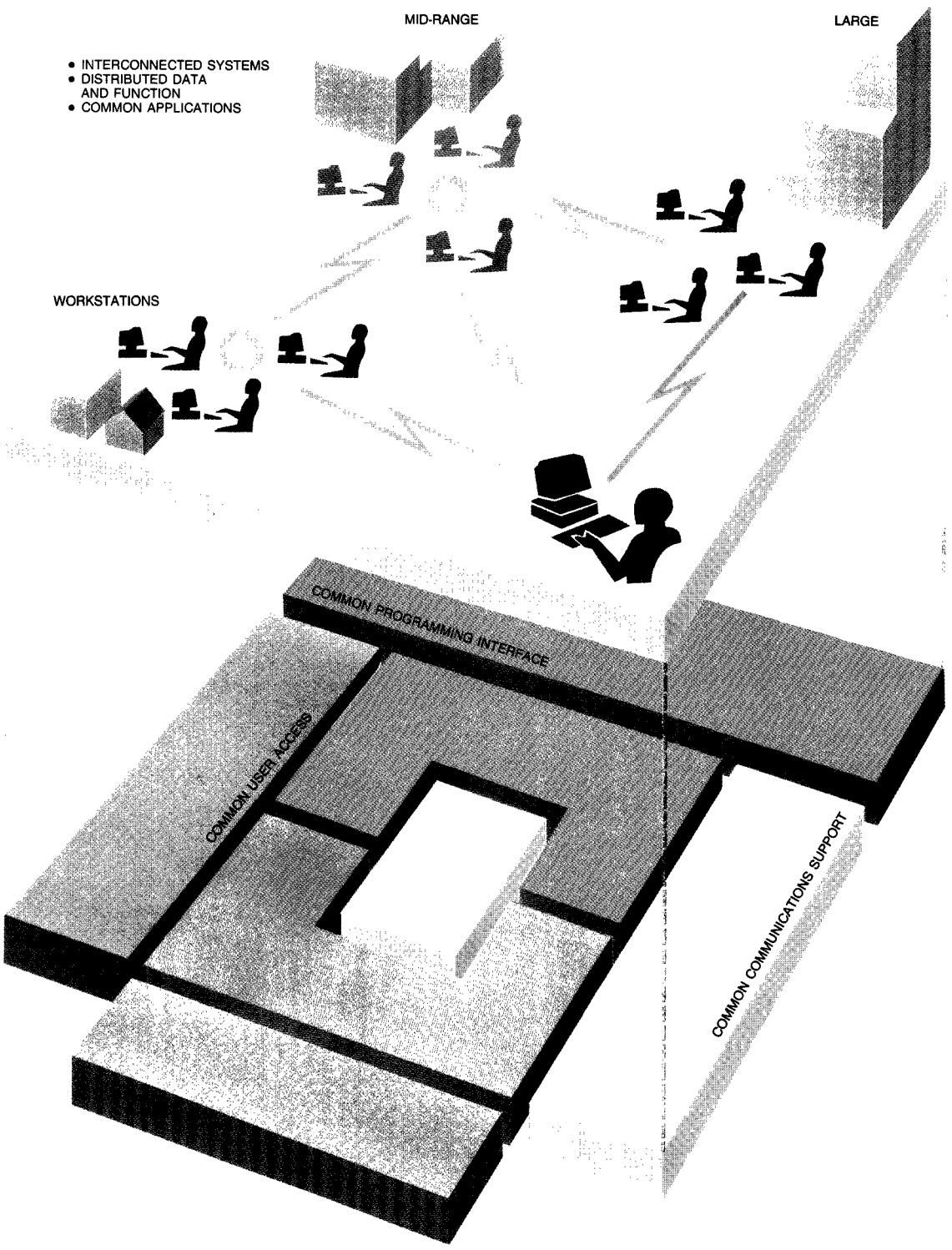
The Enterprise Information System support will be further enhanced by the advances in distributed processing that will facilitate transparency of location for data and function across the enterprise. This capability will help mask the complexity of the interconnected environment and will permit application programmers to focus more on the problem to be solved. Continued emphasis on network and system management tools as well as consistent security mechanisms across the scope of the Enterprise Information System will be required to minimize the effort to support and control the diverse systems in the enterprise.

Conclusion

In summary, Systems Application Architecture defines IBM's solution to achieve cross-systems consistency. It provides an architectural framework for a family of operating systems that spans the three orders of magnitude in the range of power of computing hardware now offered in IBM's product line. SAA achieves the framework via an extensive set of published protocols, interfaces, and conventions that give rise to Common Applications, which are portable across diverse environments; this set makes possible the interconnection and integration of these environments. SAA provides a consistent, state-of-the-art user interface to achieve the best possible usability.

SAA is evolving, continually encompassing a broader scope over time in order to keep pace with the advances of technology in both hardware and software. Its goal, however, remains the same—to provide the base of usability, consistency, and connectivity required to make use of programming resources and user experience most effectively, thereby increasing productivity and protecting software investment. SAA brings a comprehensive unification to IBM's family of systems, providing access to an enormous spectrum of processing power via an architecture that is available now and is the base for expansion in the future.

Figure 4 Enterprise Information System



Cited references

1. R. E. Berry, "Common User Access—A consistent and usable human–computer interface for the SAA environments," *IBM Systems Journal* **27**, No. 3, 281–300 (1988, this issue).
2. V. Ahuja, "Common Communications Support in Systems Application Architecture," *IBM Systems Journal* **27**, No. 3, 264–280 (1988, this issue).
3. D. E. Wolford, "Application enabling in SAA," *IBM Systems Journal* **27**, No. 3, 301–305 (1988, this issue).
4. R. Reinsch, "Distributed database for SAA," *IBM Systems Journal* **27**, No. 3, 362–369 (1988, this issue).
5. R. A. Demers, "Distributed files for SAA," *IBM Systems Journal* **27**, No. 3, 348–361 (1988, this issue).
6. W. P. Dunfee, J. D. McGehe, R. C. Rauf, and K. O. Shipp, "Designing SAA applications and user interfaces," *IBM Systems Journal* **27**, No. 3, 325–347 (1988, this issue).

Earl F. Wheeler *IBM United States, 2000 Purchase Street, Purchase, New York 10577.* Mr. Wheeler is IBM Vice President and General Manager, Programming Systems. He joined IBM as a junior engineer in 1957 in Endicott, New York. Throughout his career, he has played a significant role in the development of IBM's products. During the late 1960s, he was responsible for the systems management of intermediate processors such as the Model 40 and Model 50 of the IBM System/360 and the emerging System/370 Model 155. During his assignment as Laboratory Director in Kingston, New York, in the early 1970s, he directed the early formulation of Systems Network Architecture. As Systems Development Division Vice President of Industry Systems, and Vice President, Communications Systems Division, in the mid-1970s, Mr. Wheeler directed product management for all of IBM's communication products, including the 3270 displays, 370X multiplexors, and industry terminals. During this period, he was instrumental in managing the convergence of the IBM communications product line to support SNA. As Assistant Group Executive, Systems Development, Information Systems and Technology Group, in the early 1980s, he was responsible for the product strategy for all System/370 systems software. Mr. Wheeler came to Corporate Headquarters as IBM Director of Programming in 1984 and was elected IBM Vice President, Programming, in 1985. In his current position, he is responsible for directing the worldwide development of IBM's application-enabling software product offerings and is the chief architect of Systems Application Architecture.

Alan G. Ganek *IBM Data Systems Division, P.O. Box 100, Kingston, New York 12401.* Mr. Ganek is manager of VM/XA Advanced Systems. He received his M.S. in computer science from Rutgers University in 1981. He joined IBM as an associate programmer in 1978 in Poughkeepsie, New York. His first assignments involved the implementation of the MVS operating system software support for the cross-memory and 370/XA architectures. In 1981 he joined the MVS System Structure Technology department, where he contributed to the definition of the Enterprise System Architecture/370 leading to a first patent award. He later became team leader for the MVS software support design for ESA/370. In June 1983, Mr. Ganek was named development manager of the MVS System Design Department, involving work on a variety of advanced technology activities. In 1985, he was appointed manager of MVS Design and Performance Analysis, where he was responsible for the technical plan and content of the MVS base control program future releases and, in 1986, was promoted to program manager. Later that year, Mr. Ganek joined the Information Systems and Storage Group staff as a technical assistant. In 1987, he went to the Corporate Programming Staff, where he contributed to a number of efforts concerning IBM's programming strategy and Systems Application Architecture. He began his current assignment in June 1988 and is responsible for VM/XA Advanced Systems direction, design, planning, and product introduction support.

Reprint Order No. G321-5323.