IBM Systems Journal Abstracts 1962–1994

Volume 1, September, 1962

A program for optimal control of nonlinear processes *by R. A. Mugele, p. 2.* At present, there are many industrial processes of a nonlinear character for which it is difficult to develop an effective industrial process control system because no efficient mathematical method is known to carry out the optimization procedure.

This paper presents a flow chart description of a computer program incorporating a new optimization technique which will resolve many such problems. Although the mathematical basis for the technique is suggested, details and proofs are omitted—these will appear in a subsequent paper.

The technique has been successfully tested on a number of problems. Testing was conducted using a control system (IBM 1710) as well as both small and large computers (IBM 1620, 7090).

A general purpose systems simulator *by G. Gordon, p. 18.* Systems engineers have come to recognize simulation as a valuable tool in their work. However, writing simulation programs can be a difficult, time consuming task requiring intricate and extensive programming. For simulation to be most useful, it must be possible to carry out a simulation quickly and be possible to change the simulation easily as the system design proceeds.

This paper describes a general purpose simulation program designed to simplify the task of simulating systems. It is applicable to a wide variety of important problems. The program features a simple block diagram language with which to describe the system to be simulated. Given this description, the program will automatically simulate the system.

Simulation in systems engineering *by E. C. Smith, Jr., p. 33.* The author assumes that the reader is familiar with the content of the preceding paper, "A General Purpose Systems Simulator," by G. Gordon. Two dissimilar examples are provided to illustrate various aspects of simulation in the systems engineering process. One example involves the study of an IBM 7040—IBM 7090 computer complex for scientific applications. The other concerns an IBM 1410 Tele-processing system for a stock brokerage house.

In addition to illustrating the paper, both examples are of intrinsic interest. The first presents a new philosophy of multiprocessing. The second examines a method of integrating communication facilities with an information processor.

Tables, flow charts, and program logic *by M. Montalbano, p. 51.* "Decision" tables are introduced with reference to business data processing. A method of verifying both the completeness and consistency of a problem description is given.

The conversion of tables to computer programs is considered and a technique of obtaining a computer program which minimizes the branching requirements with respect to both memory and execute time is included. Program debugging and program modification are also discussed.

A multiprocessing approach to a large computer system *by F. R. Baldwin, W. B. Gibson, and C. B. Poland, p. 64.* This paper examines the machine utilization and job turnaround problems of a large computer center by analyzing the information handling and queuing problems occurring between jobs.

A system designed to overcome these difficulties is described and the results of simulating the system are reported. The system design includes the interconnection of input-output computers with large scale processors by means of commonly shared disk files.

Although this paper deals with a study which is not yet completed, the techniques developed and the results obtained to date are of general interest.

Note—The trim problem *by R. E. Gomory, p. 77.*

Note—On modifying the 1620 ADD table *by G. Gerson, p. 82.*

Volume 2, March, 1963

Economic evaluation of management information systems *by D. F. Boyd and H. S. Krasnow, p. 2.* A method of representing the gross characteristics of an information system within a dynamic model of the firm is presented.

The performance of the firm and, indirectly, that of the information system is measured in accordance with usual financial accounting practice.

The procedure is demonstrated by simulations (programmed using a general purpose simulator) conducted with a specific model of a hypothetical manufacturing firm.

Computer construction of minimal project networks *by B. Dimsdale, p. 24.* Computer techniques, now employed by management in the planning, scheduling, and control of projects, generally rely on the formulation of project "networks" as input to the computer programs.

This paper discusses a computer procedure for improving the input by obtaining networks with certain minimal properties.

With this improvement in input, the overall efficiency in using existing programs can be increased.

Sequential data processing design *by V. P. Turnburke, Jr., p. 37.* This paper outlines a systematic method of designing a data processing tape system utilizing currently available types of equipment.

Primary effort was devoted to obtaining a procedure which would approach an "optimal" system design.

The method presented is an iterative procedure which tends to focus special attention on the critical system functions and the critical relations between functions.

Optimum response analysis *by C. F. Kossack, p. 49.* This is the first of a series of expository papers, to appear periodically, dealing with selected statistical techniques which can be conveniently applied with the use of a digital computer to a substantial range of practical problems.

The present paper discusses a technique, developed by G. E. P. Box and K. B. Wilson, which permits solution of the problem of finding an optimum (or minimum) value without first finding the underlying mathematical model.

Applications of the method and the accompanying programming problem are also considered.

Programming considerations for the 7750 *by Nicholas Sternad, p. 57.* The design of real-time commercial data processing systems includes special computers serving as communications control devices.

These special purpose stored-program "data exchange" computers, used in conjunction with standard data processors, permit significant simplification of the system's programs as well as an increase in the overall system efficiency.

This paper considers the new concepts involved in programming a particular data exchange computer.

Recovery for computer switchover in a real-time system *by Harry Nagler, p. 76.* A programming technique is presented which permits switching from central to stand-by computer in case of failure. Switchover is accomplished automatically and without loss of data or interruption in service.

The technique is applicable to a large class of commercial real-time systems which must function in an uninterrupted manner.

During the normal periods when both computers are operable, the programming system permits the second computer to be utilized independently for other data processing.

Volume 2, June, 1963

File organization and addressing *by W. Buchholz, p. 86.* The principal approaches to random-access file organization and addressing are reviewed in this paper. The review is general, in the sense that it is relatively independent of specific equipment. In the case of a number of unsettled questions, the author's evaluations of alternatives are included.

The relation between sorting and random-access file addressing is clarified by reviewing both as belonging to a common class of ordering operations. Basic considerations of both sequential and random-access approaches, arithmetical key-to-address transformation methods with their overflow problems, and table lookup methods are discussed.

Results of an experimental analysis of key transformation techniques are presented.

Note on random addressing techniques *by W. P. Heising, p. 112.* Formulas are derived for the average number of record references required to retrieve a record from a file (a) in case the records are loaded without regard for relative frequency of reference and (b) in case each set of records with a common home address is arranged in order of decreasing frequency of reference.

The formulas are first derived under the assumption that the mapping from keys to addresses is "random." Finally, an informal argument is given which suggests the formulas will also hold under a familiar "pseudo-random" mapping based on the use of division, provided the keys have a certain property commonly encountered in practice.

Programming notation in systems design *by K. E. Iverson, p. 117.* The function of programming nota-

tion in systems design and the characteristics of a suitable language are discussed.

A brief introduction is given to a particular language (developed by the author and detailed elsewhere) which has many of the desired properties.

Application of the language is illustrated by the use of familiar examples.

On the location of supply points to minimize transportation costs *by F. E. Maranzana, p. 129.* An algorithm applicable to the problem of locating supply points optimally with respect to transportation costs is given.

Although the algorithm may fail to converge to an optimal solution, repeated application with judicious selections of alternative starting values will assure a good, if not optimal, solution.

The algorithm has been tested and some sample results are included.

Statistical classification techniques *by C. F. Kossack, p. 136.* This paper reviews the procedure of evolving statistical classification rules.

Selection of variables, methods of classification, selection of a decision rule, and the problem of analyzing effectiveness of the technique are considered.

The procedure is demonstrated computationally by means of an example.

Design of an integrated programming and operating system, Part I: System considerations and the monitor *by A. S. Noble, Jr., p. 153.* The present paper considers the underlying design concepts of IBSYS/IBJOB, an integrated programming and operating system.

The historical background and over-all structure of the system are discussed.

Flow of jobs through the IBJOB processor, as controlled by the monitor, is also described.

Design of an integrated programming and operating system, Part II: The assembly program and its language *by R. B. Talmadge, p. 162.* Integrated system design leads to the inclusion of certain features in the assembly language for the convenience of compilers, and others for the convenience of program segmentation.

This paper discusses motivation for the inclusion of these features, and traces their influence upon the internal structure of the assembly program.

Volume 2, September–December, 1963

An intrinsically addressed processing system *by J. E. Griffith, p. 182.* This paper, motivated by the classical work of Bush, discusses the possibilities of designing an information processing system based on intrinsic addressing techniques.

The primary design objective is to develop a system with increased capability for non-numerical information processing.

Suggestions for the physical and programming system logic are outlined from a macroscopic point of view and some applications of the system are indicated.

Project evaluation and selection *by B. Dimsdale and H. P. Flatt, p. 200.* A criterion is formulated which will permit project selection corresponding to management's statement of objectives and their relative importance.

An algorithm is developed to implement the criterion. The accompanying programming problem is examined and experience gained in executing the algorithm is described.

Application of the algorithm is demonstrated by detailing the solution of a problem.

A directly coupled multiprocessing system *by E. C. Smith, Jr., p. 218.* Interconnecting processors is one approach to organizing a computer facility to better serve its users.

The objective of such system organization is to reduce the elapsed time a job resides in the system (turnaround time) while simultaneously increasing the workload the equipment can handle (throughput).

Alternative philosophies of multiprocessing are discussed and, in particular, a concept which enables coupling an IBM 7090 and an IBM 7040 to meet this objective. In this system the smaller machine performs supervisory and input-output functions while the larger one performs program assembly and computation.

Dynamic storage allocation for a real-time system *by B. I. Witt, p. 230.* An algorithm for dynamic storage allocation of variable-sized programs and records is described. The algorithm is designed for real-time systems in which core is assigned for data and programs in a completely unscheduled manner as, for example, in reservation systems.

The objective is to make efficient re-use of available core with minimal movement of data or programs after entry.

The procedure given depends on the frequency distributions of program usage and of data block sizes. However, the distributions need not be specified since the system will adapt to these distributions and, equally important, to any changes in them that may occur.

The algorithm has yet to be simulated or tested within an operating system.

A computer-operated laboratory data-taking system *by H. Cole, Y. Okaya, and F. W. Chambers, p. 240.* This paper discusses the use of a computer to control data-taking in the laboratory—including the case of "closed-loop" control.

Illustration is provided by describing a particular system involving computer control of an x-ray diffractometer.

A pattern identification system using linear decision functions *by J. S. Griffin, Jr., J. H. King, Jr.,*

and C. J. Tunis, p. 248. This paper is concerned with application of linear decision functions to the pattern identification problem and describes an experimental pattern recognition system for the magnetic ink character font now used in the banking industry.

The system is based on a linear decision function determined by means of a variant of an "adaptive training" technique due to Rosenblatt.

The system has been partially implemented (in part, through simulation with aid of a digital computer and, in part, by hardware) and experimental results in using the system are reported.

Requirements generation, explosions, and bills of material *by F. L. Church, p. 268.* This paper introduces the principal data processing procedures now applied within many manufacturing industries to the "requirements generation" problem.

The procedures discussed take into consideration certain related problems in production planning and inventory control.

The nature of the various problems is illustrated and flow charts for the principal procedures are included.

Generation of input data for simulations *by S. Yagil, p. 288.* An algorithm is given to generate additional input data for simulations when some, but insufficient, historical data are available. The additional data generated are statistically "similar" to the historical.

Motivation and application of the algorithm are demonstrated by means of a problem related to the monthly water inflow to Lake Tiberias which had to be resolved in connection with the "Israeli Integrated Water Supply" project now under construction.

The algorithm is a variant of a method previously used by Thomas and Fiering in hydrological studies.

Design of an integrated programming and operating system, Part III: The expanded function of the loader *by R. Hedberg, p. 298.* This paper outlines the structure and operation of the system's loader.

The new system functions which affect the loader are related to the additional functions which the loader performs.

Descriptions of the algorithms employed by the loader for symbolic unit assignment and buffer allocation are included.

Design of an integrated programming and operating system, Part IV: The system's FORTRAN compiler *by R. Larner, p. 311.* This paper describes the system's 7090/94 FORTRAN compiler. Comment is made on the design problem and objectives.

The general structure and operation of the compiler are examined.

Indexing procedures for array reference and iteration control within the object programs produced by the compiler are detailed.

Design of an integrated programming and operating system, Part V: The system's COBOL compiler *by R. T. Dorrance, p. 322.* The general considerations underlying the design of the system's COBOL compiler are discussed.

A brief outline of the operation and structure of the compiler is included.

Finally, attention is focused on certain techniques which are incorporated within the compiler.

Volume 3, Number 1, 1964

Storage requirements for a data exchange *by I. Delgalvis and G. Davison, p. 2.* A computational procedure is derived analytically to evaluate the input/output buffer storage requirements in a data exchange.

Validity of the analysis is substantiated by comparing—in a typical instance—the analytical results with those obtained by simulation.

Systems simulation with digital computers *by K. Blake and G. Gordon, p. 14.* The general nature of digital simulation of a system is discussed.

A machine-independent examination of the associated programming problem is conducted and illustrated by means of an example.

Finally, the nature and application of simulation languages are noted.

A general purpose digital simulator and examples of its application, Part I: Description of the simulator *by R. Efron and G. Gordon, p. 22.* This part of the paper describes GPSS II, a general purpose digital systems simulation program based on a block diagram language.

The program is a result of incorporating improvements dictated by extensive experience in the application of an earlier version. However, this article is self-contained.

Development and application of the language are illustrated by means of an example.

A general purpose digital simulator and examples of its application, Part II: Simulation of a telephone intercept system *by C. R. Velasco, p. 35.* The simulator is employed to determine a set of parameters defining a telephone "intercept" system with appropriate characteristics.

The intercept system is automatic and involves standard data processing components—data exchange, disk files, and audio response units.

A general purpose digital simulator and examples of its application, Part III: Digital simulation of urban traffic *by A. M. Blum, p. 41.* This part discusses use of the simulator for problems associated with urban traffic studies.

Included are simulation methods for intersections and networks, vehicular characteristics and input, and the network traffic control mechanism.

The general purpose simulator is used to write a general traffic program which is used with data cards specifying the geometry, signal settings, statistical

distributions, and other details of the particular network selected for simulation.

A general purpose digital simulator and examples of its application, Part IV: Simulation of an integrated steel mill *by D. F. Boyd, H. S. Krasnow, and A. C. R. Petit, p. 51.* An approach to the simulation of an industrial enterprise for the purpose of evaluating alternative decision algorithms is illustrated.

As an example, simulation of an integrated steel mill is discussed in sufficient detail to display programming techniques.

A description of the SIMSCRIPT language *by B. Dimsdale and H. M. Markowitz, p. 57.* This paper describes the SIMSCRIPT system simulation language and the philosophy of system structure on which it is based.

Application of the language to programming both discrete and continuous models is indicated and illustrated with examples.

SIMSCRIPT processing is described and statistics regarding operating characteristics are given.

The SIMSCRIPT system was developed at The RAND Corporation by a group including the second author.

A character computer for high-level language interpretation *by J. E. Meggitt, p. 68.* This paper discusses the design of an experimental character-processing computer for the interpretive execution of higher-level language programs.

The design specifies a 100-nsec instruction cycle for the microprogram instructions stored in a read-only memory, a fast memory for intermediate "scratch pad" computation, and input/output through a conventional computer coupled to a 2-μsec main memory. The object program to be interpreted is stored in the main memory.

As a part of the research, the design was simulated on standard equipment.

Design of an integrated programming and operating system, Part VI: Implementation on the 7040/44 data processing system *by B. White and J. Trimble, p. 79.* This paper compares factors governing the implementation of the IBSYS/IBJOB operating systems for the 7090/94 and 7040/44 processing systems.

Operation of the 7040/44 system's monitors and the means of communication between them are described.

Familiarity with the content of the previous five papers of this series is assumed, though for the most part, the paper can be read independently.

Algorithm for a gear-train problem *by H. G. ApSimon, p. 95.* An algorithm for the numeric solution of a common gear-train problem is developed.

A number-theory approach, relatively novel in current engineering practice, is used in deriving the algorithm.

The form of the algorithm obtained is suitable for programming on a digital computer.

A concordance generator *by K. F. Scharfenberg, P. H. Smith, Jr., and R. D. Villani, p. 104.* The structural design of a general purpose program for concordance preparation is described.

Options in the input format, the operating mode, and the output edit provide wide flexibility in organizing data in a form convenient for many analytical purposes.

An experimental program was written, and some results obtained in testing the program are included.

Volume 3, Numbers 2/3, 1964

The structure of SYSTEM/360, Part I: Outline of the logical structure *by G. A. Blaauw and F. P. Brooks, Jr., p. 119.* A general introductory description of the logical structure of SYSTEM/360 is given in preparation for the more detailed analyses occurring in the other parts of the paper.

The functional units, the principal registers and formats, and the basic addressing and sequencing principles of the system are indicated.

The structure of SYSTEM/360, Part II: System implementations *by W. Y. Stevens, p. 136.* The performance range desired of SYSTEM/360 is obtained by variations in the storage, processing, control, and channel functions of the several models.

The systematic variations in speed, size, and degree of simultaneity that characterize the functional components and elements of each model are discussed.

The structure of SYSTEM/360, Part III: Processing unit design considerations *by G. M. Amdahl, p. 144.* Considerations underlying the design of the central processing unit are discussed.

Particular emphasis is placed on addressing, sequencing, and monitor control functions as well as provisions for arithmetic and logical operations.

The structure of SYSTEM/360, Part IV: Channel design considerations *by A. Padegs, p. 165.* The organization of the input/output section and the control of input/output operations in SYSTEM/360 are described.

Emphasis is on the philosophy of control and on the reasons for choosing the particular logical and physical organization.

For each machine feature, the types of tasks requiring the facility are outlined and the significance of the solution is shown.

The structure of SYSTEM/360, Part V: Multisystem organization *by G. A. Blaauw, p. 181.* Operation of several systems as one multisystem to obtain increased availability, improved cost/performance, or both, is considered.

System requirements for various applications are formulated, and the multisystem capabilities of SYSTEM/360 are discussed in context.

A formal description of SYSTEM/360 *by A. D. Falkoff, K. E. Iverson, and E. H. Sussenguth, p. 198.* All SYSTEM/360 functional characteristics having programming significance are completely and concisely described.

The description, which is formal rather than verbal, is accomplished by a set of programs, interacting through common variables, used in conjunction with auxiliary tables.

The language used in the programs involves operators and notation selected from mathematics and logic, together with additional operators and conventions defined to facilitate system description.

Although the formal description is complete and self-contained, text is provided as an aid to initial study.

Examples to illustrate the application of the formal description are given in an appendix.

Volume 4, Number 1, 1965

An interpretive program for matrix arithmetic *by F. H. Branin, Jr., L. V. Hall, J. Suez, R. M. Carlitz, and T. C. Chen, p. 2.* The structure and use of an interpretive program for matrix operations is treated.

The discussion emphasizes the nature of the programming language and the method of storage allocation. The system provides automatic storage allocation for external disk storage as well as for core memory.

Algorithm for computer control of a digital plotter *by J. E. Bresenham, p. 25.* An algorithm is given for computer control of a digital plotter.

The algorithm may be programmed without multiplication or division instructions and is efficient with respect to speed of execution and memory utilization.

An analysis of floating-point addition *by D. W. Sweeney, p. 31.* This paper analyzes the addition operation of floating-point systems.

The analysis of a million executed floating-point additions is presented as an aid in optimizing design and measuring performance.

The frequency of the various shifts for floating-point additions with different radices was derived from the basic data so that designs with various radices may be evaluated.

On the reliability of polymorphic systems *by P. D. Welch, p. 43.* The reliability aspects of polymorphic systems are examined within the confines of a simple failure and repair model.

Emphasized are the derivation and use of easy-to-calculate approximations to the unavailabilities of system capacity levels.

A technique to control waiting time in a queue *by S. Shapiro, p. 53.* This paper describes a control technique for regulating the waiting times of jobs in a discrete manufacturing process.

The technique is based on the second method of Lyapunov, which has been extensively used for deterministic processes. Two illustrations of the method are included.

Experimental evidence of the effectiveness of the technique is indicated.

Notes on testing real-time system programs *by M. G. Ginzberg, p. 58.* Procedures for program testing associated with implementation of a large complex real-time system are discussed step by step.

The discussion includes testing both in a simulated environment and in real time.

Final testing and monitoring of system performance are also briefly considered.

Serial compilation and the 1401 FORTRAN compiler *by L. H. Haines, p. 73.* This paper discusses a compiler organization in which phases act sequentially on a source program held in core storage.

A brief description of each phase of the 1401 FORTRAN compiler is given to illustrate the general scheme.

Volume 4, Number 2, 1965

Fabrication and assembly operations, Part I: The outlines of a control system *by C. T. Baker, p. 87.* The structure of a comprehensive control system for the fabrication and assembly industries is outlined.

The system consists of several major functional components that are interconnected as an integrated whole.

Component functions are described and relevant methodologies mentioned.

Fabrication and assembly operations, Part II: Long-range planning techniques *by A. B. Calica, p. 94.* An approach to the preparation and evaluation of preliminary plans for a discrete manufacturing enterprise is outlined.

Some major data processing problems arise in this type of long-range planning. Mathematical techniques applicable to the solution of these problems are discussed.

Fabrication and assembly operations, Part III: Matrix methods for processing configuration data *by P. G. Loewner, p. 105.* This discussion presents a unified method for organizing the configuration data of manufacturing files, and for generating and retrieving essential quantities from the files. The required processing operations, which include various requirements and engineering-change computations, are explained with the aid of matrix algebra.

An important objective of the method is to permit a reasonably optimal balance between the bulk-storage requirements and the amount of time required for processing.

Fabrication and assembly operations, Part IV: Linear programming in production planning *by B. P. Dzielinski, p. 122.* In many industries, production

planning involves the allocation of various resources in the joint production of similar products. Inventory levels, labor decisions, and an economic choice of lot sizes are all influential in the planning process.

Formulated in terms of mathematical programming, the economic and mathematical facets of production planning are discussed. A feasible computation technique is suggested.

The construction of discrete dynamic programming algorithms *by M. Held and R. M. Karp, p. 136.* Certain sequencing and scheduling problems are formulated as shortest-route problems and treated in a uniform manner by dynamic programming. Computational considerations are discussed.

Algorithms for traffic-signal control *by L. A. Yardeni, p. 148.* Algorithms for the design of traffic-signal progressions for fixed-time control are described.

Least-squares and minimax fits are used to derive solutions for given volume requirements within specified limits of speed and cycle time.

The algorithms have been programmed for processing on a digital computer, thus reducing the initial design time considerably and leading to solutions that are superior to manually derived designs.

Computer channel interference analysis *by W. Chang and D. J. Wong, p. 162.* This paper develops a queuing model that analyzes the capabilities of a low speed data channel for real-time data inputs. The model yields estimates of waiting, service, and overall transit times.

Low speed channel throughput can be increased by multiplexing low speed devices. It is assumed that the multiplexing operation employs common registers, the contents of which are saved at initiation and restored at completion of service for all outstanding requests. The model takes into account preemptive interference from high speed data channels.

Volume 4, Number 3, 1965

GPSS III—an expanded general purpose simulator *by H. Herscovitch and T. H. Schneider, p. 174.* Significant improvements in the modeling capability and storage flexibility of the General Purpose Systems Simulator are described in this paper.

Increased versatility and ease of use as well as new debugging aids are also discussed.

The additions and changes to the simulator are illustrated by examples.

On dynamic program relocation *by W. C. McGee, p. 184.* A general statement of the problem of dynamic program relocation is presented as an aid in describing specific relocation principles.

The main purpose of the paper is to review a number of typical methods of meeting the expanding need for dynamic program relocation. Although no attempt is made at evaluation, the methods are discussed in the context of selected computer systems for tutorial concreteness.

A computer-aided linkage analysis system *by F. Bitonti, D. W. Cooper, D. N. Frayne, and H. H. Hansen, p. 200.* An experimental system for the kinematic analysis of two- and three-dimensional mechanical linkages is outlined.

The structure of the programmed system, the input language, and the method of storage allocation are described.

The class of problems treated by the system is discussed in brief, as are the basic vector equations used in obtaining solutions for position, velocity, acceleration, and force of linkage elements.

Fabrication and assembly operations, Part V: Production order sequencing *by A. B. Calica, p. 225.* The sequencing of several project networks on limited facilities is discussed under the assumption that the projects and resources have already been specified by a higher scheduling function.

A priority function is proposed which uses both the local and global properties of the project network.

The resulting schedule is then converted into a network on which useful alterations can be made.

Fabrication and assembly operations, Part VI: Parameter values for sequencing control *by S. Gorenstein, p. 241.* This document adapts the sequencing control reported in Part V of this paper to individual plant requirements and goals.

A regression model is used to relate measures of plant performance to certain control parameters. This relationship is periodically recomputed using statistical analysis of operational data.

A pertinent decision rule is derived by optimal control theory.

Fabrication and assembly operations, Part VII: Adaptive control in production planning *by S. Shapiro, p. 250.* This paper discusses a control method for reducing the operating costs of a production system by continual modification of the planning operations.

The method improves resource allocations by adjusting the mathematical model of the production system to actual system performance.

The results of some preliminary experimental work with a simulated fabrication shop are presented.

Volume 5, Number 1, 1966

The functional structure of OS/360, Part I: Introductory survey *by G. H. Mealy, p. 3.* A brief outline of the structural elements of OS/360 is given in preparation for the subsequent sections on control-program functions.

Emphasis is placed on the functional scope of the system, on the motivating objectives and basic design concepts, and on the design approach to modularity.

The functional structure of OS/360, Part II: Job and task management *by B. I. Witt, p. 12.* This part of the paper discusses the control-program functions most closely related to job and task management.

Emphasized are design features that facilitate diversity in application environments as well as those that support multitask operation.

The functional structure of OS/360, Part III: Data management *by W. A. Clark, p. 30.* Concepts underlying the data-management capabilities of OS/360 are introduced; distinctive features of the access methods, catalog, and relevant system macroinstructions are discussed.

To illustrate the way in which the control program adapts to actual input/output requirements, a real operation is examined in considerable detail.

Volume 5, Number 2, 1966

Macro language design for SYSTEM/360 *by D. N. Freeman, p. 62.* The macro language design discussed in this paper provides a systematic means by which the SYSTEM/360 assembler-language programmer can develop macroinstructions, thereby expanding the set of machine-oriented instructions that serve as the basis of the assembler language.

Also treated is the format of macro definitions, the design of a macro generator, and the principal considerations that governed the design of the system as a whole.

A study of replacement algorithms for a virtual-storage computer *by L. A. Belady, p. 78.* This study is based on a virtual-storage concept that provides for automatic memory allocation.

Several algorithms for the replacement of current information in memory are evaluated.

Discussed is the simulation of a number of typical program runs using differing replacement algorithms with varying memory size and block size. The results are compared with each other and with a theoretical optimum.

Computation of e^x with the use of large tables *by K. Spielberg, p. 102.* A procedure is given for computation of e^x using tables of coefficients of the economized approximating polynomial over a range of positive and negative x. A related procedure that uses continued fractions is also discussed.

The exponential function was selected to test the effectiveness of table lookup methods in the computation of elementary functions. The number of multiplications or divisions required of standard methods is compared with the number required when table lookup is employed.

A queuing model for a simple case of time sharing *by W. Chang, p. 115.* This paper discusses a queuing model for a non-priority time-sharing environment in which all active tasks fit in a homogeneous main storage. Design parameters such as queue length and response time, as well as their distributions, can be estimated with the aid of the model. The model provides a basic frame of reference for the development of more complicated models.

Volume 5, Number 3, 1966

On teleprocessing system design, Part I: Characteristic problems *by W. P. Margopoulos and R. J. Williams, p. 134.* For analytical purposes, a teleprocessing system can be characterized as a digital computer with unscheduled inputs from a number of remote points. In the design of such a system, various queuing problems arise as a consequence of the unscheduled inputs, and the necessity of linking remote points to the central computer leads to a problem in combinatorial mathematics.

To show the origin of these problems, a functional classification of teleprocessing applications is given, a schematic of a basic teleprocessing system is introduced, and the relative merits of mathematical analysis and digital simulation are discussed.

On teleprocessing system design, Part II: A method for approximating the optimal network *by L. R. Esau and K. C. Williams, p. 142.* A teleprocessing system may include many low-speed terminals at great distances from the computing center. Specification of a communication network for connecting the remote terminals to the central computer constitutes an important design problem.

An iterative method for obtaining an approximate solution to an optimum network is presented. The method assumes that an acceptable line utilization factor is given.

On teleprocessing system design, Part III: An analysis of a request-queued buffer pool *by J. P. Bricault and I. Delgalvis, p. 148.* The problem of allocating main storage for a message-segment buffer pool is considered. A queuing model that approximates the most typical mode of operation is formulated. Solutions for the number of buffers by the pool are obtained. Although the solutions require iterative computational methods, they are not difficult to program.

Inasmuch as the model and solution methods both involve approximations, the validity of the approach was checked by simulating a typical set of operating conditions. Although the computational results were found to be conservatively biased, the method is clearly adequate for most design purposes.

On teleprocessing system design, Part IV: An analysis of auxiliary-storage activity *by P. H. Seaman, R. A. Lind, and T. L. Wilson, p. 158.* Queues of requests for access to auxiliary storage play a major role in every teleprocessing application. Assuming that access requests are randomly distributed, a queuing model is formulated; formulas are obtained for the mean and variance of the response-time distribution, as well as for the utilization factors of the access channel and the storage modules.

Samples of analytical and simulation results are given.

On teleprocessing system design, Part V: A technique for estimating channel interference *by T. W. Gay, Jr., p. 171.* In typical teleprocessing applications, a large number of terminals communicate with the main-storage unit of a centrally located computer. The communication activities may reduce the processing capacity of the computer by claiming a significant number of main-storage access cycles.

If the number of cycles to be claimed is partially dependent upon channel-busy and request-pending conditions, the problem involves a probability analysis. Using a simple queuing model as a starting point, a usefully accurate solution method is developed. The SYSTEM/360 multiplexor channel is analyzed as an example.

On teleprocessing system design, Part VI: The role of digital simulation *by P. H. Seaman, p. 175.* As a tool for quantitative investigation, digital simulation is especially suited to the study of stochastic processes having many interdependent variables. Not only can a simulation model be modified to reflect structural changes in a process, but it can be used to gain insights during the design of the process. These properties recommend the use of digital simulators in the design of complex teleprocessing systems.

This paper comments on the main considerations involved in choosing between general-purpose and special-purpose simulators.

Volume 5, Number 4, 1966

A computer program for the statistical analysis of series of events *by P. A. W. Lewis, p. 202.* This paper discusses general considerations that arise in the statistical analysis of point stochastic processes (series of events) and a computer program called SASE designed to implement such an analysis.

The program is written as a sequence of independent subroutines. The computations performed in each subroutine are described and an example of an analysis of a series of events is presented and discussed.

Merge-sort analysis by matrix techniques *by C. E. Radke, p. 226.* Previous work, which analyzed certain merge-sorting methods with the aid of difference equations, is extended to include a wider range of methods. Matrices are introduced to represent the set or sets of difference equations associated with a merge-sort. Two or more matrices are required to define a Class II method, whereas a Class I method can be defined with one matrix. The merge-sorts of most interest fall into a special subclass called Class Ia.

It is shown that an asymptotic solution to the set of difference equations for a Class Ia merge-sort is readily obtainable. Carter's analysis of cascade and polyphase merge-sorts is generalized and extended to include, among other things, the compromise merges. Various properties of the Class Ia merge-sorts, including relative performance measures and explicit merge patterns, are shown to be obtainable by matrix multiplication. Although the analysis emphasizes Class Ia merges, suggestions are given for applying the matrix technique to other merge-sorts of Classes I and II.

Kernel analysis of elliptic partial differential equations *by S. G. Hahn and E. V. Hankam, p. 248.* The extrapolated Liebmann method for solving partial differential equations is selected for study. With typical computer characteristics in mind, several schemes for organizing the requisite data flow are discussed.

To show the potentialities of timing formulas, as well as their limitations and the problems encountered in their construction, one of the data-flow schemes is treated at length. Kernel programs are included, and timing formulas needed in making comparisons of various configurations of two computers are developed.

Volume 6, Number 1, 1967

Function and design of DOS/360 and TOS/360 *by G. Bender, D. N. Freeman, and J. D. Smith, p. 2.* The functions of disk and tape operating systems for SYSTEM/360 configurations with as little as sixteen thousand bytes of main storage are discussed. The two related systems are designed to provide a range of services that include input/output control, stacked-job control, symbolic device assignments, and library maintenance. A set of language translators, a set of sort/merge programs, and various other programs go far toward minimizing the effort required of program preparation.

Design objectives, system definitions, and functional capabilities are stressed. The design of the control program is discussed in some detail.

Data management concepts for DOS/360 and TOS/360 *by A. R. Cenfetelli, p. 22.* The data management function is discussed in the specific context of DOS/360 and TOS/360, the disk and tape operating systems for intermediate SYSTEM/360 configurations.

Explained are the processing routines of the data management facilities, collectively referred to as the input/output control system.

Techniques that keep the routines small in size, efficient in operation, and simple to use are emphasized.

Internal data management techniques for DOS/360 *by D. H. Ricour and V. Mei, p. 38.* A technique for individual preassembly and linkage of input/output program sections, which reduces overall assembly time, is described.

Also discussed are two techniques used in generating channel programs for direct-access devices. One of the techniques is designed for random addressing of records, the other for indexed sequential addressing.

The developmental work that led to these techniques was heavily influenced by the objective of effectively minimizing the amount of main storage required for

the input/output control functions in DOS/360, the disk operating system for SYSTEM/360 configurations with intermediate amounts of main storage.

Real-time systems in perspective *by J. D. Aron, p. 49.* The more important characteristics of real-time systems are listed, discussed in an historical context, and illustrated by remarking on relevant features of typical applications. Because the intent of the paper is to provide a general survey, no attempt is made at a truly rigorous definition of the term "real-time." The point of view taken is a functional one, viz., that the distinguishing properties of most real-time systems stem directly from the distinctive needs of five different classes of applications: control, command and management information, time-shared computing, remote batch computing, and data acquisition.

A number of general references are included for the reader who is interested in more detail on the various aspects of real-time systems.

Volume 6, Number 2, 1967

An application-oriented multiprocessing system, Part I: Introduction *by J. F. Keeley, p. 78.* The purpose of this short introduction is to provide background for a discussion of an application-oriented multiprocessing system.

An application-oriented multiprocessing system, Part II: Design characteristics of the 9020 system *by G. R. Blakeney, L. F. Cudney, and C. R. Eickhorn, p. 80.* The equipment that comprises the IBM 9020 multiprocessing system is described, emphasizing capabilities not appearing in the standard SYSTEM/360 line.

System characteristics are related to availability requirements for program-controlled reconfiguration capabilities. With these capabilities, subsystems can be formed from system elements as the need arises.

Other discussed functional capabilities center on recovery by program control, shared storage, and malfunction alerting.

An application-oriented multiprocessing system, Part III: Control program features *by J. A. Devereaux, p. 95.* The described control program dynamically schedules the operational activities performed by the IBM 9020 multiprocessing system. Scheduling is based on program execution requirements and allows dynamic switching of Computing Element assignments.

Storage resources are dynamically allocated by the control program to guard against the mutual interference of concurrent operations.

The trace capability of the control program is described because of its importance to the checkout and evaluation of multiprocessor systems.

An application-oriented multiprocessing system, Part IV: The operational error analysis program *by D. C. Lancto and R. L. Rockefeller, p. 103.* The described program analyzes and isolates equipment faults concurrently with regular processing.

If necessary, the program replaces system elements by realigning communication and control paths.

Dependence of the program's replacement decisions upon the recording of extensive error statistics is also discussed.

An application-oriented multiprocessing system, Part V: The diagnostic monitor *by R. Suda, p. 116.* Off-line diagnostic programs for system test, acceptance test, and field maintenance of the IBM 9020 multiprocessing system are executed under control of the monitor under discussion.

The structure of the monitor is based on seven functional components which are examined in detail.

A discussion of the debugging experience during the development of the monitor is included.

An application-oriented multiprocessing system, Part VI: Programs for the intended application *by F. K. Seward, p. 124.* The multiprocessing system is viewed within the frame of the intended air-traffic-control application.

Functions of the five application-oriented programs, as well as the main features of the input/output environment, are discussed.

Volume 6, Number 3, 1967

Evaluation of redundancy in a parallel algorithm *by G. S. Shedler and M. M. Lehman, p. 142.* Computers with parallel computing capabilities may become generally available in the future. Some implications for the field of numerical analysis are indicated. An analogue of the bisection algorithm for root determination, employing redundancy in computation as a means of developing parallelism, is presented. An evaluation of the effect of redundant parallel computation on the speed and efficiency with which results are obtained is given for this algorithm.

Aspects of the Gemini real-time operating system *by J. H. Mueller, p. 150.* This paper describes the major elements of a programmed operating system for a complex of five computers employed at the Gemini Mission Control Center. The system was designed for an application environment that includes real-time space missions, simulated real-time exercises, and extensive job-shop operations. Relationships among programs, input/output control, and various operational techniques are described. The characteristics of a statistics-gathering routine are outlined.

High-speed calculation of the critical paths of large networks *by M. Montalbano, p. 163.* Discussed is the design, implementation, and detailed operation of an experimental IBM 7090 program that calculates the critical path as well as early and late start times for a project network.

The program accepts suitably coded information about activity durations and precedence relationships

and constructs an internal representation of the network. Results are printed out in either one of two formats: one for the consistent case and the other for the inconsistent case.

A formal descriptive language is used to describe the basic algorithm. The specific storage and indexing techniques employed in the program are useful in a wide class of directed-graph applications.

An automatic dictionary and the verification of machine-readable text *by E. J. Galli and H. Yamada, p. 192.* The possibility of applying an experimental dictionary and a digital computer to a proofreading application was investigated. Because technical abstracts yield a high concentration of proofreading difficulties, a sample of such text was used for study purposes.

The general features of the dictionary, as well as the main algorithms used for dictionary search and text processing, are discussed. Methods for classifying input words and flagging output words are described. Approximately nineteen thousand words of keypunched abstracts were experimentally processed, with results that are discussed. The verification algorithms are evaluated in light of the results obtained, and recommendations for additional improvements and refinement are then presented.

Volume 6, Number 4, 1967

Microprogram control for SYSTEM/360 *by S. G. Tucker, p. 222.* This paper describes the kind of microprogram control that has been used in several models of SYSTEM/360. A microprogramming language, as well as some of the main techniques used in "assembling" and testing microprograms, are discussed. Applications of microprogram control to the design of emulators, to compatibility features, and to special modifications are summarized.

Two continuous system modeling programs *by R. D. Brennan and M. Y. Silberberg, p. 242.* The motivation, history, and basic concepts of user-oriented languages for digital simulation of continuous systems are presented. Reference is made to two illustrative programs, the IBM 1130 and SYSTEM/360 Continuous System Modeling Programs (CSMP).

Both programs accept user-oriented input statements for constructing simulation models and controlling simulation runs. The 1130 CSMP also allows on-line interaction by the user. An engineer or scientist at the console can alter the model or change run conditions based on direct observation of simulation outputs. The SYSTEM/360 CSMP is intended for batch-mode operation. It has extended facilities for describing the model and for obtaining automatic program control of successive simulation runs.

Conventions for digital data communication link design *by J. L. Eisenbies, p. 267.* A definitive set of conventions has been established for the automatic, synchronous transmission of digital data over half-duplex (nonsimultaneous) communication links. Provision has been made for communication between different device types and between computer processing units. Although one transmission code must be used on a given data communication link, a special feature permits digital data in any form to be transmitted, including encrypted data and compiled computer programs.

This paper describes the Binary Synchronous Communication (BSC) conventions, which prescribe the encoding of data, the procedures for synchronizing stations, the methods for controlling the data links, transmission message formats, and error detection and correction methods. The presentation is sufficiently detailed to indicate the kinds of design decisions that are involved in setting up automatic data communication links.

Volume 7, Number 1, 1968

Structural aspects of the System/360 Model 85, Part I: General organization *by C. J. Conti, D. H. Gibson, and S. H. Pitkowsky, p. 2.* A basic design objective for the Model 85 was to add a computer to the SYSTEM/360 line that offers high performance over a wide range of job types. Simulation studies indicate that the Model 85 will provide an average three- to five-fold increase in internal performance with main storage capacities of up to four million bytes.

This part of the paper discusses the major elements of the Model 85 within the architectural context of SYSTEM/360, including the addition of a high-speed buffer, called a cache.

Also summarized are the simulation studies that led to use of the cache, selection of its parameters, and verification of internal performance of the system.

Structural aspects of the System/360 Model 85, Part II: The cache *by J. S. Liptay, p. 15.* The cache, a high-speed buffer establishing a storage hierarchy in the Model 85, is discussed in depth in this part, since it represents the basic organizational departure from other SYSTEM/360 computers.

Discussed are organization and operation of the cache, including the mechanisms used to locate and retrieve data needed by the processor.

The internal performance studies that led to use of the cache are described, and simulated performance of the chosen configuration is compared with that of a theoretical system having an entire 80-nanosecond main storage. Finally, the effects of varying cache parameters are discussed and tabulated.

Structural aspects of the System/360 Model 85, Part III: Extensions to floating-point architecture *by A. Padegs, p. 22.* The repertoire of SYSTEM/360 instructions has been expanded in the Model 85 by introducing facilities for extended-precision floating-point arithmetic. This part describes the new instructions, discusses their need, and considers the design factors that influenced their choice.

An economic lot-sizing technique, Part I: The part-period algorithm *by J. J. DeMatteis, p. 30.* As a lot-sizing technique for minimizing the sum of ordering and inventory costs, the algorithm described is based on some simple dimensions. By dividing the ordering and setup costs by the inventory holding costs per part per time period, the ordering costs are expressed in part-periods. This value is used to determine lot size.

First a simplified version is shown for demand sets that do not vary widely between periods. For large variations in demand, significantly greater overall accuracy is achieved with simple look-ahead and look-back tests which are also discussed.

Two of the more important economic lot-sizing algorithms are compared with the Part Period Algorithm.

An economic lot-sizing technique, Part II: Mathematical analysis of the part-period algorithm *by A. G. Mendoza, p. 39.* The Part Period Algorithm discussed in Part I is compared with optimal solutions, determining the maximum deviations possible. Optimality of the algorithm is established for the case of constant demand.

Performance characteristics are compared with those of the Least Unit Cost algorithm.

A multi-item economic lot-sizing problem *by J. F. Pierce, p. 47.* A multi-item economic lot-sizing problem is considered wherein, as a consequence of a joint ordering or production setup cost, the ordering policies for individual items are interdependent.

The problem is to determine an optimal ordering plan in which the sum of the costs of carrying inventories and the costs of ordering are minimized and in which the known demands for each item in each time period are satisfied.

Two algorithms are presented for solving such problems. The first is a direct algorithm which yields periodic solutions and applies to problems in which demand occurs uniformly over time. The second is a dynamic programming algorithm which yields optimal solutions, whether periodic or aperiodic, and which applies to dynamic problems as well as to problems with constant demands.

Volume 7, Number 2, 1968

Avoiding deadlock in multitasking systems *by J. W. Havender, p. 74.* Designers and users of multitasking operating systems must be alert to the problem of task deadlock, which prevents the affected tasks from being completed.

This paper describes the conditions that can result in task deadlock in any multitasking systems. Also discussed are techniques for avoiding deadlock in both operating system and application program design. Finally, it is shown how these techniques were applied in the design of the SYSTEM/360 Operating System job initiator, the part of the system that allocates major resources needed to execute jobs.

Statistics gathering and simulation for the Apollo real-time operating system *by W. I. Stanley and H. F. Hertel, p. 85.* Since each Apollo manned space flight makes new demands on the computer configuration and operating system, data processing efficiency is tested before each flight by simulation described in this paper.

Discussed is the dynamic gathering of operating system performance data during real-time simulation, achieved by incorporating appropriate routines in the Apollo control program. The data thus collected is used as input to improved system models.

The effect of the statistics gathering routine on systems performance can be measured.

Turnaround time for messages of differing priorities *by C. Hauth, p. 103.* Queuing theory and statistical methods are used in this paper to derive formulas for determining average turnaround time in teleprocessing systems that handle messages on a priority basis. This information is needed to ensure, for example, that messages are processed within acceptable time limits and that efficient use is made of system resources.

Factors considered in arriving at the formulas described here include waiting time in message queues, message processing time, I/O waiting time, and delays for higher priority processing. Results conform closely with those obtained from simulation studies.

Hierarchical control programs for systems evaluation *by D. D. Keefe, p. 123.* Today's complex operating and computing systems make systems testing a difficult task. Major problems arise when one attempts to measure the performance of a system in a multiprogram environment and to evaluate the interfaces between computer elements, programs, and operator.

Historically, testing devices were first developed to monitor system activity and to produce test data. Separate computer systems were next used to permit on-line data reduction and generation of appropriate test conditions.

The purpose of this paper is to describe a hierarchical control program design which incorporates the major capabilities of the previous solutions without requiring a separate computer. This low-cost, flexible technique has been applied in the measurement of various performance characteristics, in generating simulated error conditions, and in simulating machine devices and features.

Volume 7, Numbers 3/4, 1968

Interactive Graphics in Data Processing: Principles of interactive systems *by C. I. Johnson, p. 147.* Present considerations in interactive display activity are brought into perspective.

Emphasizing programming aspects, varying approaches are considered with respect to system concepts and peripheral graphics processors, as well as

complex data structures, high-level languages, and image-generation techniques. Many of the problems discussed are not unique to graphics systems but are common to interactive systems in general.

The requirements for a conversational system to support programmers and application users are listed in the Appendix. An extensive bibliography has been added as basic reference for this issue.

Interactive Graphics in Data Processing: Aspects of display technology *by A. Appel, T. P. Dankowski, and R. L. Dougherty, p. 176.* In providing background information on display technology, this paper discusses cathode-ray tube characteristics and interactive devices as they affect the user.

Described are the display functions of the IBM 2250 display console, which is used in many applications.

Some elementary aspects of image generation are presented, and the current and potential capability of displays is discussed.

Interactive Graphics in Data Processing: Geometry for construction and display *by D. V. Ahuja and S. A. Coons, p. 188.* Matrix notation is used to develop geometric concepts for computer-controlled graphics.

This natural form of geometric expression leads to homogeneous coordinates which form the basis of an algorithm used for geometric construction. In this way, three-dimensional objects and other pictures can be displayed on a graphics console.

The paper also briefly discusses the notation and development of functions for the construction of surfaces.

Interactive Graphics in Data Processing: An algorithm for generating spline-like curves *by D. V. Ahuja, p. 206.* Discussed is a method of drawing curves of arbitrary shape on a graphic display screen.

An algorithm for the design of free-form curves is developed by using rational polynomials and the notation of homogeneous coordinates. With this algorithm, spline-like curves can be generated through arbitrarily placed points in a plane or in space.

Interactive Graphics in Data Processing: A multilevel modeling structure for interactive graphic design *by H. B. Baskin and S. P. Morse, p. 218.* In the past, a common data structure, or model, has been used for all phases of a computer graphics design system. This has meant that the model used during conversational interaction was the same as the model used in the subsequent analysis operations, usually resulting in poor overall performance. This paper suggests the use of separate models for each phase, providing a general model for the conversational drawing phase which is suitable as a front end in many different application areas.

Described are data structures and programs for both phases: a conversational display image manipulation program (DIM) and its interconnections with an existing analysis application program (IBM 1130 Continuous System Modeling Program). Examples of the use of this particular multilevel modeling design facility are included.

Interactive Graphics in Data Processing: Auxiliary-storage associative data structure for PL/I *by A. J. Symonds, p. 229.* A recent approach to representing relations between entities in a graphics data structure has been to store information as triples in the form Attribute (Object) = Value.

This paper describes an associative technique for holding a universe of triples on auxiliary storage and then accessing a triple in response to an inquiry.

The paper also shows how relational operations have been performed—on an experimental basis—with PL/I as the language for the controlling program, using machine-language subroutines to perform only the basic functions on associative storage.

Interactive Graphics in Data Processing: A subroutine package for FORTRAN *by A. D. Rully, p. 248.* A basic objective in the design of a package of general-purpose graphics subroutines was to make them accessible to FORTRAN programmers while circumventing some of the limitations of that language for graphics applications.

This paper discusses how observation of graphics applications led to the establishment of design criteria for a subroutine package to facilitate the generation of interactive displays. It outlines how application programmers would use the package, including provisions for communication between the console and the program. Many of the fundamental concepts that characterize the package are described, including provisions for display modification and animation.

Interactive Graphics in Data Processing: A system for implementing interactive applications *by F. C. Chen and R. L. Dougherty, p. 257.* This paper discusses a system designed for interactive problem solving by use of a graphic display console. Existing application programs can readily be modified for use with a graphic display device, and the graphics programming can usually be done in a higher-level language. Based on intermediate results, the order of execution of application modules can be controlled from the console.

The system description emphasizes the structure and generation of display formats for displaying output, for accepting user-defined commands, and for accepting data that is made accessible to the application modules. Also described is a generalized data structure and a set of experimental routines designed to adapt the structure to particular needs.

Interactive Graphics in Data Processing: Conversational job control *by S. H. Brown, p. 271.* A graphic job processor enables nonprogrammers to introduce application programs conversationally from a display console. Although not restricted to graphics applications, the processor makes the same display console available for both job definition and interaction with a graphics application program.

This paper discusses some of the factors considered in designing displays to elicit information from the user. The structure of the processor is then described, including its interface to the operating system under which it functions. It also discusses communication among the system operator, the console user, and the application programmer.

Interactive Graphics in Data Processing: A conversational display capability *by F. W. Gagliano, H. W. Thombs, and R. E. Cornish, p. 281.* This paper discusses a system called DISPLAYTRAN that interpretively executes FORTRAN statements entered at a display console, allowing graphics users to perform unanticipated computations and to more easily debug graphics application programs.

The relationships among the operating system, the display terminal, and the computing system are discussed, and the major components of this system are described. A command language, the FORTRAN IV subset, and the graphics language provided for users are presented. Internal operation of the graphic facility is outlined.

Interactive Graphics in Data Processing: A language for three-dimensional geometry *by P. G. Comba, p. 292.* This paper argues that there is a need for a problem-oriented language to handle three-dimensional geometric information, and proposes a set of language facilities that illustrate how this need should be met.

The emphasis is on the facilities needed for describing solid objects and their placement in space, and for defining and operating on configurations of objects.

Interactive Graphics in Data Processing: Modeling in three dimensions *by A. Appel, p. 310.* Discussed are two computer programs for generating and realistically plotting any view of a three-dimensional object from the same object description, thereby simulating the viewpoints of a person moving around the object. Although the programs have been implemented—on an experimental basis—for digital plotting, the use of the underlying concepts for graphic display is contemplated.

Involved in the SIGHT program are approaches to some of the most difficult problems in three-dimensional graphics—the hidden-line problem, approximating curved solids by polyhedra, and simulating degrees of surface transparency.

The description of the program LEGER emphasizes the design of data storage for the object description. This scheme allows the use of the same data for generating all views of the object. The data structure can be modified to adjust the dimensions of the scene and the relative orientations of the component parts.

Interactive Graphics in Data Processing: Interactive aspects of crystal structure analysis *by Y. Okaya, p. 322.* In demonstrating the advantageous use of graphics in a heuristic approach to problem solving, the deciphering of unknown crystal structures is used as an environment. The interactive graphic console lends concreteness to the crystallographer's abstract perception of three-dimensional structures.

This paper describes how the crystallographer uses graphics in determining crystal structure by matching a known spatial molecular model with an experimental crystallographic model and by direct theoretical procedures when no model is available. Also presented are computer rendering techniques that make the man-machine relationship more productive.

In conclusion, the author ponders the possibilities of more efficient and productive scientific research through the evolution of computer-aided data libraries.

Interactive Graphics in Data Processing: Geometric relationships for retrieval of geographic information *by J. D. Jacobsen, p. 331.* Reported is an experimental technique that has been developed for retrieving, by geometrical means, information related to city maps.

This paper emphasizes the analysis needed to translate a retrieval query into relationships among points, vectors, and polygons. An illustration of the technique is given.

Interactive Graphics in Data Processing: Analysis and display of physics data *by W. C. McGee, H. R. Penafiel, and S. K. Howry, p. 342.* A group of programs, called SUMX, is used for statistically analyzing high-energy physics data by batch-processing techniques.

Against this background, the paper discusses the first phase of a project directed toward placing the SUMX user on-line via a display console. On-Line SUMX provides a helpful interactive mode of computer use in an inherently difficult application area.

The experimental environment of the data source is discussed. Presented are functions of component programs as well as the types of statistical analyses performed.

Interactive Graphics in Data Processing: Neutron cross-section evaluation *by R. J. Creasy, p. 355.* Evaluation, storage, and retrieval of neutron cross-section data are of major concern to the international community of low-energy physicists. Discussed is an experimentally evolving program, called SCORE, designed to perform these services.

The overall concept of the SCORE program and the programming environment are presented. Cross-section data entry, evaluation, and curve generation are discussed.

Interactive Graphics in Data Processing: Cam design on a graphics console *by J. M. Lafuente, p. 365.* This paper describes a generalized program for the analysis and design of plate cam and follower mechanisms using a graphic display console. The experimental program was developed to study the use of interactive graphics systems for solving mechanical design problems.

The program can handle almost all types of plate cam and follower mechanisms. A wide selection of motion curves permits the designer to specify any desired motion of a point of interest by synthesizing several curves into a displacement diagram.

Interactive Graphics in Data Processing: Implementation and usage *by C. W. Day and L. L. Zimmerman, p. 373.* The present status of interactive graphic displays in the application environment is reviewed.

Although graphics data processing is still largely experimental, several applications have come into productive use—especially in the areas of data and design analysis. The primary benefit from enhancing such applications with graphic displays is the savings in calendar time.

Before surveying several application areas, user aspects and application characteristics are discussed.

Volume 8, Number 1, 1969

Hierarchical structure for data management *by W. R. Henry, p. 2.* This paper describes an approach to data management that is based on a hierarchical organization of the data management control function and makes use of list processing concepts.

Discussed are the separation of the logical and physical control functions as well as the data-element and operating-system controls. This hierarchical approach establishes a common basis for the creation, maintenance, and retrieval of data in direct-access storage. Logical functions express the control and management of generalized physical data structures; the physical level typically includes strings for data retrieval and maintenance.

Undirected graphs, and matrices derived from them, illustrate the data management relationships within the physical level. The same type of analysis may be used to show relationships between the hierarchical levels.

GPSS/360—an improved general purpose simulator *by R. L. Gould, p. 16.* Increased adaptability, versatility, and ease of use are significant advantages offered to the user of the General Purpose Simulation System in this latest version as described in this paper.

Modeling capability is especially enhanced by such improvements as new entities and block types and extended features.

Also discussed are improved input specifications and an output editor.

A teleprocessing approach using standard equipment *by R. D. Wade, G. P. Cawsey, and R. A. K. Veber, p. 28.* This paper describes an operational teleprocessing system that allows both low-speed conversational data entry and high-speed remote job entry and output. At the same time, the multiprocessing system provides conventional batch processing.

The total system configuration, which consists of standard equipment, is briefly introduced. Then the teleprocessing control program, the major programming support developed for the system, is described in detail. This program functions as the interface between the teleprocessing lines and the input/output streams of the batch processing system. The final topic is the terminal program, which is provided for the central processing units at the terminals.

Coding for error control *by D. T. Tang and R. T. Chien, p. 48.* Tutorially presented are theoretical and practical concepts that underlie error-control coding for data computing, storage, and transmission systems.

Emphasis is on cyclic codes, the most deeply studied and widely used of the many available codes. Operations of typical binary shift registers illustrate the encoding and decoding processes.

Strategic considerations for applying coding to computer-communication systems are discussed. Actual applications further exemplify the basis for code selection.

Volume 8, Number 2, 1969

Some principles of time-sharing scheduler strategies *by H. Hellerman, p. 94.* Fundamental considerations in time and space scheduling for time-sharing systems are reviewed. Workload components are classified as trivial and nontrivial foreground, and background. Each has certain resource-use and required response properties. A central issue in scheduling is the degree of advance knowledge available to the scheduler about calls on system resources. This provides a theme for classifying several algorithms.

A response figure of merit believed to be helpful in understanding time-sharing schedulers is defined. Simulation results using a very simple workload and system model are included in the discussion. A summary is given of some major issues in scheduling for time-sharing and virtual systems.

An auxiliary processing system for array calculations *by J. F. Ruggiero and D. A. Coryell, p. 118.* The IBM 2938 Array Processor discussed enhances the processing power of SYSTEM/360 Models 44, 65, and 75 for operations on vectors and matrices. As an integrated channel-I/O device, the array processor responds to standard I/O instructions.

Discussing the overall flow of control between the central processing unit, main storage, and the array processor, the relationship between the SYSTEM/360 Operating System and the array processor's programming support is emphasized. The twelve mathematical processing operations embodied in the 2938 are described in terms of their algorithms.

Several methods for measuring array processor performance are shown together with actual timing results.

A pseudo-random number generator for the System/360 *by P. A. W. Lewis, A. S. Goodman, and J. M. Miller, p. 136.* A particular pseudo-random number generator is described that uses the full 31-bit capacity of the registers in the IBM SYSTEM/360 computers.

Experience with the generator in obtaining random permutations of sequences is discussed, and results of statistical tests applied to evaluate the generator are given. The generator has been found to be highly satisfactory.

An assembler language program of the generator is included.

A network algorithm for empty freight car allocation *by W. W. White and A. M. Bomberault, p. 147.* Distributing empty freight cars throughout a railroad system in anticipation of future requirements is an allocation problem. The actual movement of cars can be examined in terms of a space-time diagram. An inductive network flow algorithm for solving this problem utilizing the network underlying the space-time diagram is developed and illustrated by an example.

A computer program implementing this algorithm is discussed, along with the context in which it might be used. Possible extensions are also presented.

Volume 8, Number 3, 1969

A three-value computer design verification system *by J. S. Jephson, R. P. McQuarrie, and R. E. Vogelsberg, p. 178.* Described is an experimental system for verifying logic designs in the development of a computer before a commitment to produce the computer is made.

The system simulates logic activity with both known (0,1) and unknown (X) values. The use of the third value facilitates the generation of tests and the detection of circuit hazards.

Internal sorting with minimal comparing *by L. J. Woodrum, p. 189.* An ordering operator that leads to the development of an algorithm for internal sorting is described. An analysis of the algorithm is presented, together with a discussion of the number of comparisons necessary for sorting.

It is shown that the number of comparisons is close to the theoretically obtainable number. The sorting algorithm is a variant of the two-way merge.

Problem formulation using APL *by H. G. Kolsky, p. 204.* Much of the arbitrariness of conventional program solutions to large-scale scientific problems can be removed by the approach presented in this paper. The logical formulation of such problems can be improved by using the programming language APL, which is mathematically compact and explicit. The language also allows the system much freedom in producing computed results.

A meteorological problem is discussed as an illustration of APL-augmented programming. Two approaches to programming the solution are compared.

Determining economic sampling plans *by E. W. Stacy, L. E. Hunsinger, and J. F. Price, p. 220.* Each stage in multistage manufacturing processes raises a question of how much inspection is appropriate for quality assurance. Sampling procedures usually provide the least expensive way to maintain quality.

In this paper, a method for use on a computer is developed for evaluating single sampling plans on the basis of economics.

Volume 8, Number 4, 1969

A perspective on system performance evaluation *by M. E. Drummond, Jr., p. 252.* A historical summary of the growing complexities in computing systems and the effect of these complexities on system performance evaluation are presented.

The paper outlines basic approaches. It considers the application of test results and the test data itself. Two general approaches to gathering performance data are discussed.

Simulating operating systems *by P. H. Seaman and R. C. Soucy, p. 264.* A simulator is discussed that provides a language and a structure specifically designed for modeling computer systems to evaluate their performance.

The simulator provides general equipment models, and the authors discuss their experience in developing a general submodel of a multiprogramming operating system. The user assembles a system from the equipment models, specifies parameters to allow simulation of his operating system functions, and provides models of his application programs.

Trace-driven system modeling *by P. S. Cheng, p. 280.* A trace-driven modeling technique for computer system evaluation is discussed. This approach intends to solve the inherent dilemma in computer systems modeling of too many simplifying assumptions or of too much detail.

An experimental model based on this technique is also discussed. The objective of the model is to study the effect on performance of various multiprogramming and multiprocessing system design choices. Model implementation is focused on those aspects of the system contributing substantially to total system performance. While the operating system is conceptually modeled, detailed logic and timing are supplied in the job-trace profile, reducing modeling effort and improving model flexibility.

Using system monitor output to improve performance *by A. J. Bonner, p. 290.* Computing systems are analyzed from a system performance profile, which shows usage of the different system components, such as channels and central processing units. Component usage data are obtained by a system monitor.

The profile indicates where a system configuration or a program might be modified to improve performance. The profile also suggests areas where more detailed monitoring and analysis appear promising.

Measurement of system operational statistics *by W. I. Stanley, p. 299.* This paper discusses programming techniques for continuously gathering performance data in a complex multijobbing environment. The program takes advantage of computing system and support program characteristics to count and time data processing activities.

The data gathered by the program enables installation managers to measure the effects of modifications to equipment and programming support configurations and to define work loads.

A synthetic job for measuring system performance *by W. Buchholz, p. 309.* A technique for measuring and comparing the performance of existing computer systems is to devise a synthetic job that is simple enough to be programmed with a modest effort in different languages and on dissimilar machines, so as to be run and timed on each of the systems.

The job described here is a greatly simplified file maintenance procedure, which exercises both the central processing unit and major input/output devices, with activity parameters being specified in a manner independent of the system. A complete PL/I version is shown as an example. It is conjectured that such a synthetic job may evolve into a practical standard of performance.

Effects of storage contention on system performance *by C. E. Skinner and J. R. Asher, p. 319.* This paper presents a mathematical model to measure the amount by which a computer's speed is reduced when it time-shares storage with other computers and I/O channels. The method can be applied to any number of processors and/or channels and storage units, although the complexity of the solution does increase rapidly as the number of processors increases. Explicit formulas and numerical results are given for several special cases.

The results of a simulation of a shared-memory multiprocessor are presented, showing how closely the mathematical model fits the operation of a simulated system.

Volume 9, Number 1, 1970

On-line inquiry under a small-system operating system *by K. Darga, p. 2.* Operating systems for small computers are more restricted in their capabilities than those used on larger computers and are generally not designed for multiprogramming. However, some facility is needed to permit such actions as inquiries.

This paper discusses an option to a small operating system that permits interruptions of a job, execution of a program to handle an inquiry, and return to the interrupted job. A description of the operating system serves as a basis in discussing the characteristics of the option.

Trajectory control programs in support of Apollo missions *by D. R. Quarles, p. 12.* Once the trajectory of a spacecraft in flight has been predicted, control measures are required to ensure that the trajectory data is applied in a consistent manner when calculating the many trajectory-related parameters required by the flight controllers of the space flight.

This paper describes the queue control techniques used in generating the predicted trajectory and the subsequent use of the trajectory data. The queue control logic that is discussed requires a minimum amount of main storage while a task is waiting to be performed.

A structure for real-time Stenotype transcription *by J. W. Newitt and A. Odarchenko, p. 24.* Computer transcription of Stenotype code offers the possibility of producing English text from speech via the Stenotype keyboard in real time. Reported are experiments directed toward designing a time-shared Stenotype transcription system that makes use of earlier work in Stenotype dictionaries and language processing.

A content-addressing algorithm for direct-access storage, requiring a single access per retrieval, is presented. An existing experimental Stenotype dictionary program is used to implement on System/360 this algorithm and a dictionary-compaction technique. Transcription analysis indicates that the experimental design can reduce the average transcription error rate to six percent.

Single-server queuing processes in computing systems *by W. Chang, p. 36.* Reviewed are applications of queuing models that may be economically useful in computing system analysis.

With emphasis on terminal-oriented systems with priorities, methods are given for estimating such average quantities as service time, waiting time, and response time.

Examples illustrate these methods and their ranges of efficiency, beyond which simulation techniques may be preferable.

Volume 9, Number 2, 1970

Evaluation techniques for storage hierarchies *by R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, p. 78.* The design of efficient storage hierarchies generally involves the repeated running of "typical" program address traces through a simulated storage system while various hierarchy design parameters are adjusted.

This paper describes a new and efficient method of determining, in one pass of an address trace, performance measures for a large class of demand-paged, multilevel storage systems utilizing a variety of mapping schemes and replacement algorithms.

The technique depends on an algorithm classification, called "stack algorithms," examples of which are "least frequently used," "least recently used," "opti-

mal," and "random replacement" algorithms. The techniques yield the exact access frequency to each storage device, which can be used to estimate the overall performance of actual storage hierarchies.

A model of floating buffering *by L. J. Woodrum, p. 118.* Discussed in this paper is the effect of floating buffering on the execution time of a program.

An analytic model of floating buffering is developed and discussed. It is shown that with use of the model it is possible to compute the run time of a program as a function of the number of floating buffers it uses.

Interactive Saturn flight program simulator *by J. H. Jacobs and T. J. Dillon, p. 145.* Space vehicle control, guidance, and navigation require onboard computers. Mission safety and success demand high program reliability without preliminary in-flight testing.

Interactive Saturn flight simulation discussed in this paper tests all normal and perturbed launch vehicle interactions with the mission computer and programs to find and correct programming problems. Using a graphics console, flight analysts execute mission programs, make programming changes, and observe and document the simulated reactions of the launch vehicle.

Volume 9, Number 3, 1970

Code-generation technique for large-language compilers *by M. Elson and S. T. Rake, p. 166.* A solution is proposed to the problem of optimizing code generation by a large-language compiler.

A high-level definitional language is used to define the code mappings, and an interpreter executes the routines in this language during the one-pass, text-driven code-generation phase.

The technique might also be applied to extendable languages and shared-component compilers.

A heuristic approach to task dispatching *by K. D. Ryder, p. 189.* This paper describes an experimental algorithm for allocating use of a central processing unit to perform separate data processing tasks in a multitasking system. The algorithm, which may control only a subset of the tasks being performed by the system, appears to improve run time for some work loads.

Tasks with a recent history of using input/output facilities are given preference. This heuristic treatment of tasks is carried over to the algorithm itself, which modifies its own characteristics based on its overall effectiveness in handling the tasks under its control.

A virtual machine time-sharing system *by R. A. Meyer and L. H. Seawright, p. 199.* Time-sharing has resulted in the development of methods to increase the utilization of computers. In this paper, one such method employing the concept of the virtual machine is discussed.

Described are the design objectives of CP-67/CMS, a multi-access system that manages the resources of a computer set up for time-sharing such that each user appears to have a complete, dedicated computer at his disposal. Also discussed are the system operation and some of its applications.

Interactive aeronautical charting *by J. H. Luetje, p. 219.* Discussed is an interactive graphic system for compiling air navigational data on aeronautical charts.

Eliminating many tedious manual updating operations, chart manuscripts can be interactively created and revised at a graphic console. Emphasized are design details of the graphic files, which are specific digital compilations of aeronautical data for individual charts.

Volume 9, Number 4, 1970

Automatic generation of test cases *by K. V. Hanford, p. 242.* Discussed is the "syntax machine," a program for automatically generating syntactically correct programs (test cases) for checking compiler front ends.

The notion of "dynamic grammar" is introduced and is used in a syntax-defining notation that provides for context-sensitivity.

Examples demonstrate use of the syntax machine.

The authorization problem in shared files *by T. D. Friedman, p. 258.* The problem of sharing information while protecting proprietary data in large computer files is reviewed.

The author suggests certain guidelines for data protection in general-purpose, time-sharing systems, and develops a model of a secured shared file. Operation of the system based on these guidelines is discussed.

Compiler assignment of data items to registers *by W. H. E. Day, p. 281.* This paper formulates as integer programming problems three methods for assigning data items to registers in the compilation process—the one-one, many-one, and many-few global assignment methods.

Three algorithms are described for obtaining feasible solutions to the many-one and many-few global assignment problems. One provides an optimal solution. The others, which provide good approximations, appear to be sufficiently fast for inclusion in an optimizing compiler.

Volume 10, Number 1, 1971

The application of formal logic to programs and programming *by C. D. Allen, p. 2.* The use of first-order predicate calculus in proving correctness and other properties of programs is shown to be possible in practical situations.

The necessary concepts and theory are explained, and some practical examples worked through.

FORTRAN extended-precision library *by H. Kuki and J. Ascoly, p. 39.* This paper discusses a FORTRAN subprogram library developed primarily to support extended-precision floating-point arith-

metic. The general strategy, which makes limited use of guard digits, is developed to achieve high accuracy with reasonable execution time and storage space.

In addition to describing some previously unpublished algorithms, the authors present subprograms for simulating extended-precision arithmetic and for input and output conversion.

Interactive scheduling system *by A. C. Brewer, p. 62.* Discussed are design principles, file structures, and programming techniques of a scheduling system that approximates the overall magnitude and complexity of an airline scheduling system.

Used worldwide by the National Aeronautics and Space Administration to schedule its manned and unmanned space flight missions up to one year in advance, the system operates in either batch or interactive modes to produce, modify, and observe actual and simulated schedules.

Volume 10, Number 2, 1971

The formal description of programming languages *by E. J. Neuhold, p. 86.* This paper presents a formal method for describing programming languages independently of machine architectures and compiler implementations. The method, which was developed to describe PL/1, is being applied to other programming languages and to compilers and operating systems.

The definitional techniques are demonstrated using a simple programming language (SPL). The paper has been written so that little knowledge of mathematics or formal logic is required.

Simulation of a model of paging system performance *by G. S. Shedler and S. C. Yang, p. 113.* Explored by simulation is the performance of a probabilistic model of a multiprogrammed single-processor computing system operating under demand paging.

Results of experiments on statistical methods for improving the efficiency of the simulation are presented.

Estimates of the response variables in the simulation are reported for a variety of conditions of system overhead, queuing delays, and transient response. Sensitivity of these factors to the assumptions of the model are discussed.

An analysis of the machine interference model *by A. E. Ferdinand, p. 129.* Discussed in this paper are asymptotic properties of the classical machine interference model, the simplest of queuing models. In systems analysis, the judicious use of such asymptotic properties can result in significant savings in time and effort.

Included in the paper is the solution of the generalized machine interference model.

A computer graphics system for block diagram problems *by L. A. Belady, M. W. Blasgen, C. J. Evangelisti, and R. D. Tennison, p. 143.* An experimental on-line network design system is proposed. Called DESIGNPAD, it consists of a small computer with graphic display equipment connected to a time-sharing computer and includes the necessary programming support for the equipment.

The system is designed to accept problems covering a broad spectrum of applications in the form of labeled block diagrams. The input/output medium, the man-machine interface, and the supporting data structures, particularly the cellular structure, are discussed.

Volume 10, Number 3, 1971

Program restructuring for virtual memory *by D. J. Hatfield and J. Gerald, p. 168.* Program reference patterns can have a more profound effect on paging performance in a virtual memory system than page replacement algorithms.

This paper describes experimental techniques that can significantly reduce paging exceptions in existing, frequently executed programs. Automated procedures reorder relocatable program sectors, and computer displays of memory usage facilitate further optimization of program structure.

Performance criteria and measurement for a time-sharing system *by Y. Bard, p. 193.* The performance of a complex time-sharing was monitored under actual operating conditions during a period in which changes in system configuration (both hardware and software) took place. Various techniques for assessing the impact of those changes on performance are discussed.

Real-time traffic flow optimization *by B. C. Black and D. C. Gazis, p. 217.* Discussed are methods underlying the real-time monitoring and controlling system of a critical traffic link.

Algorithms were developed for recognizing the length patterns of vehicles passing over detectors and using these for precisely computing traffic density in several sections of the Lincoln Tunnel. Vehicle control was adjusted by the system to optimize tunnel throughput.

Programming for economic lot-sizes with precedences between items *by S. Gorenstein, p. 232.* Discussed is an optimal programming model for lot-size, inventory, and work-force planning over a finite planning horizon for assembly-type production. The object is to prepare a minimum cost lot-size and work-force plan that meets the deterministic demands within the resource constraints. Planning for end-items, components, and overtime is included.

The main feature of the model is the ability to plan for assembly production having precedences with nonlinear (set-up) costs using essentially linear programming computations.

A guided bibliography to sorting *by H. Lorin, p. 244.* This bibliography attempts to help the reader select from the rich body of sorting literature that

which is in accord with his interests, needs, and prior training.

Historical trends within the field are briefly outlined, and subspecialties are identified. Critical comments and classification of the cited works are intended to help the reader to avoid wasted effort.

Volume 10, Number 4, 1971

A large-scale interactive administrative system *by J. H. Wimbrow, p. 260.* Through a nationwide network of interactive terminals in a teleprocessing configuration, users perform over twenty major business functions by sharing a single large and varied data base.

Emphasized are system design principles of the central complex whereby terminal message processing and data-base management are independently yet cooperatively performed.

Also discussed is system security, which includes user authorization and data-base reconstruction and auditing.

Analysis of free-storage algorithms *by B. H. Margolin, R. P. Parmelee, and M. Schatzoff, p. 283.* Dynamic management of free storage in a time-sharing operating system was studied empirically by the techniques of monitoring, emulation, and on-line experimentation.

A new algorithm, based on observed usage patterns of different block sizes, was implemented and evaluated. On-line experiments demonstrated that supervisor time spent in free-storage management was reduced by seven or eight to one.

Modeling for computing center planning *by F. Hanssmann, W. Kistler, and H. Schulz, p. 305.* A probability-based, theoretical model of a multiprogrammed computing system is suggested for planning future computing center requirements.

Validation of the planning model is attempted with respect to the theoretical model and applications to short-range and long-range planning.

Volume 11, Number 1, 1972

Evaluation of an interactive-batch system network *by W. S. Hobgood, p. 2.* Discussed is a computer network experiment designed to study a method of making the computing power of a high-speed batch system available to the interactive terminal system user. Commands for effecting the intersystem linkage and network operations are presented. Emphasized are system measurements and system tuning techniques for increasing efficiency.

Readings in microprogramming *by P. M. Davies, p. 16.* This guide to the literature on microprogramming is preceded by an exposition intended for the less knowledgeable reader.

Microprogram control is seen as a form of simulation in which primitive operations are combined and sequenced so as to imitate the characteristics of a desired machine. Discussed are such design considerations as microword formats, performance, writable control stores, and the relationship between microprogramming and software reliability.

System aspects of large-problem computation and display *by J. E. Fromm and D. E. Schreiber, p. 41.* Techniques for using the System/360 Operating System for implementing a large fluid dynamics program are discussed as a prototype for the solution of other coupled nonlinear partial differential equations.

Presented also are methods for real-time interaction with the problem solution.

A graphic analysis program for producing computer animated motion pictures is outlined.

Chief programmer team management of production programming *by F. T. Baker, p. 56.* Seeking to demonstrate increased programmer productivity, a functional organization of specialists led by a chief programmer has combined and applied known techniques into a unified methodology.

Combined are a program production library, general-to-detail implementation, and structured programming. The overall methodology has been applied to an information storage and retrieval system.

Experimental results suggest significantly increased productivity and decreased system integration difficulties.

Accounting control of data processing *by R. C. Rettus and R. A. Smith, p. 74.* An objective of a corporate-wide data processing service is to distribute its costs equitably among its users.

Aimed toward more realistically meeting this objective is an accounting system—general ledger, budget, and data processing resource utilization system—based on a cost-center configuration.

Discussed are techniques for accurately measuring data processing resource utilization in a multiprogramming environment and coordinating cost center expenses with the general ledger and the budget.

Volume 11, Number 2, 1972

Virtual storage and virtual machine concepts *by R. P. Parmelee, T. I. Peterson, C. C. Tillman, and D. J. Hatfield, p. 99.* This paper reviews virtual storage and virtual machine concepts, consolidating and updating earlier discussions. The manner in which actual virtual storage and machine systems have been implemented, and certain problems of current implementations, are described. To better illustrate the material, the virtual machine system CP-67 for the IBM System/360 Model 67 is considered at some length. An annotated bibliography is included.

Virtual machine computing in an engineering environment *by M. McGrath, p. 131.* Enhancement of the computing in an engineering environment by the installation of a virtual machine time-sharing system

is discussed. This installation has been particularly useful in allowing the engineer to make the computer an integral part of a design cycle through the interactive use of graphic displays.

Described is a CP-67 system implementing the virtual machine concept. By using an operating system of his choice in his own virtual machine, the engineering user has great flexibility in the development of applications.

Numerical control for machining complex surfaces *by D. B. Almond, p. 150.* The art of machining complex, doubly curved surfaces has been advanced by experimental extensions to the Automatic Programmed Tool Language (APT) and its numerical control program.

Described are mathematical programming procedures and language extensions for directing the cutting path of a machine tool over a ship's propeller, which serves as an example.

Illustrative of an area for future development is the possible extension of interactively designed surfaces—using graphic displays—to interface with an APT processor.

A general management business simulation in APL *by P. N. Wahi, p. 169.* Described is a management game that is programmed for use on a computer. The game provides participants the opportunity to make decisions regarding production, marketing, finance, and planning in a competitive industry. This game, implemented in APL utilizes interactive computing, giving participants more flexibility in the use of one of the most recent techniques in management gaming.

Volume 11, Number 3, 1972

Channel and direct access device architecture *by D. T. Brown, R. L. Eibsen, and C. A. Thorn, p. 186.* System dependence on channel and direct access device architecture was addressed with the introduction of System/370. Discussed are the alternatives to this problem and their evaluation by a channel architecture program simulator.

Also presented is the solution, a block multiplexer channel and sector addressing in devices, which resulted in more efficient channel utilization and reduced programming overhead.

Uses of virtual storage systems in a scientific environment *by P. H. Callaway, J. P. Considine, and C. H. Thompson, p. 200.* Some ways in which the virtual storage systems TSS/360 and CP-67/CMS have been used in a research environment are described emphasizing the features of each of these operating systems found to be most useful. Descriptions of research projects employing the systems are given with the discussion centering on the reasons for choosing a particular system in each project.

Cost-benefit evaluation of scientific computing services *by D. N. Streeter, p. 219.* Discussed is an approach to evaluating and comparing system costs and benefits (value) to the user and to his employer in a scientific environment.

Necessarily a semiquantitative measure, value to the user implies a departure from usual system efficiency measures such as system throughput.

Evaluated are usage policies based on single-stream and dual-stream batch systems, and terminal-oriented time-sharing systems.

A guide to programming tools and techniques *by J. W. Pomeroy, p. 234.* Current programming tools and techniques facilitating program development and maintenance under Operating System/360 and /370 are collected and discussed.

These aids are categorized and defined according to their function. Abstracts of some of the available programs are also presented.

Queuing simulation using a random number generator *by R. N. Rechtschaffen, p. 255.* Among the system performance predictive techniques available to the systems engineer are those of theoretical analysis and simulation.

One method of simulation uses the random number generator to simulate the probability distribution of events.

Introduced are principles of random number generator simulation together with examples, the results of which are compared with theoretical results.

Volume 11, Number 4, 1972

Encoding verbal information as unique numbers *by W. D. Hagamen, D. J. Linden, H. S. Long, and J. C. Weber, p. 278.* The representation of verbal information as single numbers using APL functions can optimize main storage, peripheral storage, and data transmission.

Presented in tutorial form are the concepts of the encoding and decoding process. Applications including text processing and instructional systems are also discussed.

Techniques for developing analytic models *by A. L. Anthony and H. K. Watson, p. 316.* Techniques for developing analytic models of computer systems and subsystems relate to establishment of the level of system detail, to selection of significant parameters, to definition of analytic expressions, and to validation of model results.

This paper emphasizes the use of discrete-event models in the development of analytic models, particularly with respect to identification of key parameters and to correlation of results. Described are two analytic models of computer systems, the analytic techniques employed, their relationship to corresponding discrete-event models and their advantages as performance evaluation tools.

Design features of a real-time check-clearing system *by J. A. Banham and P. McClelland, p. 329.*

Banking operations often require complex facilities for their data processing. The application required a multiprocessor configuration controlled by a single job step running continuously for many hours a day. Discussed are the special access methods and recovery procedures designed for this environment. The paper also describes a particularly efficient sorting technique evolved for handling large volumes of paper documents.

Volume 12, Number 1, 1973

Net change material requirements planning *by Joseph A. Orlicky, p. 2.* Material requirements planning, a principal approach to manufacturing inventory management, is discussed.

Defined and evaluated are concepts and characteristics of net change, a method that facilitates replanning and provides timely response to change in a transaction-oriented system.

Data structures and accessing in data-base systems, Part I: Evolution of information systems *by M. E. Senko, E. B. Altman, M. M. Astrahan, and P. L. Fehder, p. 30.* Presented in three parts is a descriptive analysis of data-base information systems.

Part I reviews the evolution of data-base systems to reveal the direction of their growth and applications. Emphasized are the two primary functions of data-base systems: storage and maintenance of structured information; and presentation of structured output information.

Part II discusses the structuring of information, and introduces a new fundamental approach to this structuring. The approach provides a stable information oriented terminology for relating the conceptual frameworks of existing systems and future systems.

Part III presents a framework, the Data Independent Accessing Model (DIAM), for describing information and its stored representations. The generality of this framework allows the model to describe most stored representations of existing systems in detail. Over the long term, it can provide a conceptual basis for systematic migration to systems with new improved capabilities.

Data structures and accessing in data-base systems, Part II: Information organization *by M. E. Senko, E. B. Altman, M. M. Astrahan, and P. L. Fehder, p. 45.* A new approach to information structuring is presented.

Basic to the structure is the notion of an Entity—an object, concept, or event—and associations among Names for Entities.

Discussed on the basis of these concepts is an Entity Set Model for information structuring.

Data structures and accessing in data-base systems, Part III: Data representations and the data independent accessing model *by M. E. Senko, E. B. Altman, M. M. Astrahan, and P. L. Fehder, p. 64.* Presented is the Data Independent Accessing Model (DIAM)—a complete model for the representing, storing, and retrieving of structured information.

DIAM is a hierarchy of models formed by the Entity Set Model and three lower modeling levels—the String Model, the Encoding Model, and the Physical Device Level Model.

Protocol for a computer network *by D. B. McKay and D. P. Karp, p. 94.* Computing system networks hold promise of increasing system efficiency through the sharing of resources, programs, and files.

Required is a protocol that performs for the network a function analogous to that performed by control blocks for operating systems.

Described are basic message-handling concepts and a protocol that are compatible with a broad range of network designs.

Volume 12, Number 2, 1973

Concepts of financial models *by P. L. Kingston, p. 113.* The use of financial models can assist in company business planning processes. This paper presents introductory concepts and considerations of financial models with emphasis on their structure and general design methodology.

A guide to financial planning tools and techniques *by B. P. Dzielinski, p. 126.* Current financial planning tools and techniques facilitating development of financial planning systems are summarized. Also presented are abstracts of some available programs that employ these techniques.

Planning-data systems *by H. F. Lande, p. 145.* Factors inhibiting the development of planning-data systems are now being resolved in part by the availability of planning-oriented programming languages.

Discussed are types of planning and the processing of planning data. Emphasized is the use of a planning systems generator—a planning-oriented language facilitating data bank definition and data entry, logical computations, and formatting of statistical or graphical reports.

Financial modeling on small systems *by R. J. Gordon, p. 161.* The implementation of financial models on small systems is discussed.

Presented are methods for card systems and direct access (FORTRAN and non-FORTRAN) systems. Financial plans are produced, similar to those generated by large-system methods.

Interactive simulation for banking *by J. F. Brown and D. W. Low, p. 172.* Current bank modeling systems are generally based on sets of equations that place limitations on the flexibility of the applications and the predictive ability of the model.

The experimental system discussed in this paper uses a three-fold approach to the simulation of actual banking activities. A generalized bank modeling system is presented from which a subset is selected

for use. The user interacts with the modeling system via an interactive simulation language. The composition of the desired model is determined interactively via a terminal and an interactive model generator.

Research results indicate that representative models can be generated by using these techniques.

Forecasting techniques *by M. Aiso, p. 187.* Forecasting, the evaluation of effects of various strategies, is discussed. Emphasized are the quantitative techniques used in forecasting and the formulation of equations to represent functional relationships.

Also presented are two example forecasting applications—demand analysis of a consumer product and a financial forecasting model.

Volume 12, Number 3, 1973

User program performance in virtual storage systems *by J. E. Morrison, p. 216.* Factors that affect the paging characteristics of user programs in virtual storage systems are presented in tutorial form. Measures are suggested that can be taken to exploit the virtual storage concept at the source language level in assembler, COBOL, FORTRAN, and PL/I.

An interactive graphics system for analysis of business decisions *by J. Ravin and M. Schatzoff, p. 238.* Described is an experimental system that enables the user, through an intelligent graphics terminal, to construct, modify, analyze, and store decision trees. With this system, business decisions under uncertainty can be analyzed. This paper discusses the system and its capabilities. Included is a brief discussion of decision analysis, which represents an aspect of financial modeling.

Describing data in computer networks *by D. H. Fredericksen, p. 257.* Discussed are three major classes of data descriptor messages—for language processors, for input/output devices, and for resource allocation—required by computer networks where the details depend on the variety of systems in a network and on file complexity.

Three classes of networks are also discussed—remote job entry networks with compatible operating systems and computers, networks for transmitting arbitrary data sets between systems that have the same internal data representation but different hardware and operating systems, and networks for transmitting arbitrary data sets between arbitrary systems. The first two classes are illustrated by IBM networks and the third by an interuniversity network.

Requirements for the three classes of networks are compared. Further details of networks of systems with the same internal structures but different hardware and operating systems are given.

Centralization or dispersion of computing facilities *by D. N. Streeter, p. 283.* Cost factors involved in computing centers that tend to motivate the centralization as opposed to the decentralization of computing services are evaluated, and a cost-minimization solution is presented.

Proposed and evaluated is a strategy for linking large regional service centers that perform standard production services with satellite centers that perform local personalized services.

Emphasized are techniques, including user waiting, for evaluating the two characteristic service types.

Experimental evaluation of system performance *by Y. Bard, p. 302.* The performance of different features of system software can be compared efficiently by means of rapid, on-line switching between the versions. This technique of on-line switching has been applied to determine the effects of page replacement algorithm, time-slice length, and user priority setting in the CP-67 time-sharing system.

Design of a checkout compiler *by B. L. Marks, p. 315.* The PL/I Checkout Compiler was designed to emphasize programmer productivity in developing programs, even at the expense of consuming extra machine resources. We explain the choices in the design of the compiler that resulted from this emphasis. The design is constrained by the requirement that a subroutine developed using this checkout compiler should be capable of executing in conjunction with code generated by a more conventional compiler. The execution environment that supports this operation is described.

Volume 12, Number 4, 1973

Data Dictionary/Directories *by P. P. Uhrowczik, p. 332.* A Data Dictionary/Directory System can provide centralized control over data resources and data management. This paper presents introductory concepts of data dictionaries, their capabilities, and an example implementation approach.

Indexing design considerations *by R. E. Wagner, p. 351.* This paper deals with the structure and use of indexes that facilitate the retrieval and storage of records based on a specific value, value range, or value sequence of a given field of a record within one or more data sets. Specifically, it examines general index structures, maintenance, index entry compression, and complex indexes as considered in the basic design of VSAM (Virtual Storage Access Method). Under complex indexes, indirect secondary indexes and indexes to multiple data sets are considered.

Functional structure of IBM virtual storage operating systems, Part I: Influences of dynamic address translation on operating system technology *by M. A. Auslander and J. F. Jaffe, p. 368.* Presented are early developments of storage management techniques, particularly those used in OS/360. Innovations introduced by systems that use dynamic address translation are traced. The impact of these techniques on current IBM System/370 Operating Systems is described.

Functional structure of IBM virtual storage operating systems, Part II: OS/VS2-2 concepts and philosophies *by A. L. Scherr, p. 382.* The largest IBM programming effort since the introduction of OS/360, the virtual storage operating system OS/VS2-2 has been designed to integrate efficient support for interactive, data base, and data communications applications.

Discussed from the point of view of its designers are tradeoffs, options, and objectives of the architectural features related to system parallelism, main storage exploitation, system resource allocation, and system recovery.

Functional structure of IBM virtual storage operating systems, Part III: Architecture and design of DOS/VS *by J. P. Birch, p. 401.* An addressing space larger than main storage—virtual storage—in System/370 challenges the operating system architect to design a disk operating (DOS/VS) for superior program execution performance.

Compared are the bases for program execution by DOS/VS with those of the earlier Disk Operating System. Increased system versatility is presented in terms of storage management, channel management, and program management.

Volume 13, Number 1, 1974

User behavior on an interactive computer system *by S. J. Boies, p. 2.* Discussed are observations on the usage of an interactive computing system in a research environment. Empirical data on user behavior are discussed that concern the duration and frequency of terminal sessions, the use of language processors, user response time, and command usage.

Direct-access device simulation *by E. Nahouraii, p. 19.* Discussed is an approach to simulating direct-access devices. An experimental simulator provides the capability to test newly developed I/O supervisors and to test code for new or proposed devices without the benefit of the actual device.

This functional simulator performs all of the search, data-movement, and status-reporting functions of the device transparently to the user. Data-driven (table) techniques enable the user to simulate a number of direct-access devices (one at a time) and measure their would-be performance. Interactive options of the simulator enable the user to check for errors and test the various error routines.

Advanced function extended with tightly-coupled multiprocessing *by R. A. MacKinnon, p. 32.* Multiprocessing hardware as implemented on System/370 is presented. With emphasis on tightly-coupled systems configurations, topics include instructions and facilities, and a comparison with prior IBM multiprocessors.

Design of tightly-coupled multiprocessing programming *by J. S. Arnold, D. P. Casey, and R. H. McKinstry, p. 60.* Selected components of tightly-coupled multiprocessing programming support are presented. Included are design rationale and reference to prior multiprocessing systems to give additional perspective.

Volume 13, Number 2, 1974

Performance analysis for the Skylab terminal system *by R. J. Mancini, p. 94.* System performance analysis techniques have been applied to support the development of a data-communication, data-base system. These techniques have been applied continuously from the system planning phase through system testing.

Computer simulative models and computer measurement tools were used in this analysis.

Structured design *by W. P. Stevens, G. J. Myers, and L. L. Constantine, p. 115.* Considerations and techniques are proposed that reduce the complexity of programs by dividing them into functional modules. This can make it possible to create complex systems from simple, independent, reusable modules. Debugging and modifying programs, reconfiguring I/O devices, and managing large programming projects can all be greatly simplified. And, as the module library grows, increasingly sophisticated programs can be implemented using less and less new code.

Synchronous data link control: A perspective *by R. A. Donnan and J. R. Kersey, p. 140.* Data link control requirements are discussed and summarized. A generalized structure for a data link control capable of meeting those requirements is presented. Synchronous data link control (SDLC) is given as a solution, evolving from the generalized structure, to meet the requirements stated. Finally, the significant attributes of SDLC are discussed and summarized.

A model for the evaluation of storage hierarchies *by J. Gecsei and J. A. Lukes, p. 163.* The design of the storage component is essential to the achieving of a good overall cost-performance balance in a computing system.

A method is presented for quickly assessing many of the technological and structural possibilities that exist today for designing storage hierarchies.

The evaluation is based on a cycling queuing model of the computer system and its programming environment, which are taken into account by miss ratio curves.

Volume 13, Number 3, 1974

VSAM data set design parameters *by D. G. Keehn and J. O. Lacy, p. 186.* A general description of the Virtual Storage Access Method (VSAM) is followed by a qualitative discussion of performance expectations. VSAM data-set design parameters are discussed with respect to performance tradeoffs. Analytic techniques are developed for relating some of the VSAM performance sensitivities to data set design parameters.

OS/VS1 concepts and philosophies *by T. F. Wheeler, Jr., p. 213.* Dynamic address translation equipment is key to the design of System/370 central processing units, and dynamic relocation is key to the design of Operating System/Virtual Storage 1.

Discussed are the significance and implementation of these key facilities in the supervisor and job scheduler functions of virtual storage operating system.

Within the supervisor are presented system initiation, page management, input/output supervisor, and storage management. Within the job scheduler are discussed queue management, the job entry subsystems, and remote job entry services.

Operating system integrity in OS/VS2 *by W. S. McPhee, p. 230.* System integrity is a basic requirement for operating system security. Presented are types of system integrity problems and their general solutions. Techniques used in OS/VS2 Release 2 to solve these problems are highlighted.

The job entry subsystem of OS/VS1 *by J. H. Baily, J. A. Howard, and T. J. Szczygielski, p. 253.* Discussed is an extended facility of job management for OS/VSI. This facility provides spooling and scheduling in a virtual storage system. Its three major components are peripheral services, central services, and queue management.

Volume 13, Number 4, 1974

The OS/VS2 Release 2 System Resources Manager *by H. W. Lynch and J. B. Page, p. 274.* Discussed is a new subcomponent of the control program of the IBM OS/VS2 Release 2 operating system that has been designed to use the resources of the system to satisfy two distinct but potentially conflicting performance objectives, i.e., response and throughput.

Termed the System Resources Manager, the subcomponent controls performance by address space swapping through the use of a swapping analysis algorithm and a workload management algorithm.

Optimizing program placement in virtual systems *by K. D. Ryder, p. 292.* An experimental algorithm for optimizing program placement in virtual storage systems is described. Interprogram linkages are monitored and subsequently analyzed for frequency and proximity. The algorithm evaluates this information within the context of a paging environment. Program lists that define the optimum program placements are then generated. Performance gains are also discussed.

System/7 in a hierarchical laboratory automation system *by H. Cole, p. 307.* Described is an implementation of a three-level computer hierarchy that provides a high degree of performance, availability, and ease of use for laboratory automation applications.

Elements of probability for system design *by A. O. Allen, p. 325.* Basic notions of probability theory are applied to problems of performance analysis of on-line real-time systems.

Frequently used is APL as both an analytical tool and as a scratch pad in working out the examples.

Volume 14, Number 1, 1975

Overview of the Supermarket System and the Retail Store System *by P. V. McEnroe, H. T. Huth, E. A. Moore, and W. W. Morris III, p. 3.* An introduction to the concepts of the Supermarket and Retail Store Systems is presented.

Discussed are the objectives of the systems as they relate to the problems of the merchandiser and the solution to those problems in terms of specific system function and the structure chosen to implement that function. The system design philosophy pertaining to the terminals, store controllers, host processors, and specialized I/O devices is also discussed. Specific requirements of each system are described.

The characteristics and decodability of the Universal Product Code symbol *by D. Savir and G. J. Laurer, p. 16.* Described are the coding and symbol of the Universal Product Code. The symbol code structure, format, encodation technique, and characteristics with their technical tradeoffs are discussed.

The symbol is analyzed and evaluated. Decodability is shown to depend on the structure of the code and symbol, the size of the symbol, the precision with which the symbol is printed, the technique of scanning employed, the accuracy with which measurements are made, the decoding logic, and the physical operation of scanning. The relationship between the scan pattern of a fixed head scanner and symbol size is shown.

The role of the operator in the Supermarket and Retail Store Systems *by D. C. Antonelli, p. 35.* Some aspects of the role of the operator in the Supermarket and Retail Store Systems are presented, specifically with respect to the input of data to the system. Major differences between the data input requirements of a system for a supermarket and a system for a retail store include the volume of data and the rate of entry. These are discussed in terms of the system requirements and the alternative methods of implementation. Studies of wand entry for the Retail Store System and fixed optical scanning for the Supermarket System are also discussed.

Store performance studies for the Supermarket System *by W. C. Metz, Jr. and D. Savir, p. 46.* Performance of the Supermarket System is measured by throughput of the shoppers and the response time of the system to messages generated during checkout. This paper discusses some system design features adopted for the purpose of meeting a performance objective and two models developed for analyzing the throughput capacity of the system.

Design and performance considerations for the Retail Store System *by M. A. Berk, C. W. Dunbar, and G. C. Hobson, p. 64.* The transactions in a retail

system are a comprehensive set of store applications. These applications include point-of-sale operations and back-room activities. Store performance is described by the throughput rate of transactions and the delay in the processing of any particular transaction. Two techniques of performance analysis are discussed: analytic evaluation and store simulation. An important consideration is the annual operating cycle of the store that indicates the amount and types of demands on the system.

Reliability, availability, and serviceability design considerations for the Supermarket and Retail Store Systems *by R. O. Hippert, Jr., L. R. Palounek, J. Provetero, and R. O. Skatrud, p. 81.* This paper discusses system considerations of error recovery, prevention of data loss, and protection against loss of function. Back-up techniques, problem determination procedures, maintenance procedures, and system features provided to facilitate their respective uses are discussed. Avoidance of the interruption of store operation is emphasized.

Volume 14, Number 2, 1975

A program generator *by W. D. Hagamen, D. J. Linden, K. F. Mai, S. M. Newell, and J. C. Weber, p. 102.* A person-to-computer communication system for application program writing by nonprogrammers is discussed.

Called A Program Generator (APG), the interface system is built upon the authors' previous developments of a tutorial system that is briefly discussed.

Principles and applications of APG are presented and illustrated in terms of actual applications.

Performance measurement tools for VM/370 *by P. H. Callaway, p. 134.* To support the smooth running of a VM/370 installation, performance measurements of various types are desirable. This paper describes a range of measurement facilities that have been developed for VM/370 for use both on-line and off-line at the level of the users (general user, operator, and system analyst) and the installation management.

Elements of queuing theory for system design *by A. O. Allen, p. 161.* Reviewed are fundamental principles of queuing in terms that apply to computing systems.

After laying a foundation of a minimum number of definitions, the author provides a working familiarity with extended principles and applications to system performance estimation through the use of worked out examples.

Hierarchical approach to computer system integrity *by J. J. Donovan and S. E. Madnick, p. 188.* Security is an important factor if the programs of independent and possibly error-prone or malicious users are to coexist on the same computer system. In this paper, we show that a hierarchically structured operating system, such as produced by a virtual machine system, that combines a virtual machine monitor with several independent operating systems (VMM/OS), provides substantially better software security than a conventional two-level multiprogramming operating system approach. This added protection is derived from redundant security using independent mechanisms that are inherent in the design of most VMM/OS systems. Such a system can be obtained by exploiting existing software resources.

Volume 14, Number 3, 1975

An access control mechanism for computing resources *by H. M. Gladney, E. L. Worley, and J. J. Myers, p. 212.* The architecture of an access control mechanism for the resources of an OS/360 or VS/370 computer system is presented.

The use of this operating system component for data base security and integrity in a research and engineering environment is described.

The techniques described make possible controlled access to the system's processing power, controlled access to the database, decentralized authorization responsibility, measuring dataset usage, and event recording for automatic dataset migration, archiving, and staging.

Generalized audit trail requirements and concepts for data base applications *by L. A. Bjork, Jr., p. 229.* Discussed is a data base audit trail. It is defined here to be a generalized recording of "who did what to whom, when, and in what sequence." This information is to be used to satisfy system integrity, recovery, auditing, and security requirements of advanced integrated data base/data communications systems. This paper hypothesizes what information must be retained in the audit trail to permit recovery and audit later in time and a scheme of organizing the contents of the audit trail so as to provide the required functions at minimum overhead.

Introduced are the concepts of types of audit required, DB/DC audit assumptions, time domain addressing, time sequences required to support versions of data, what constitutes an audit trail, and implementation considerations.

Tuning a virtual storage system *by H. A. Anderson, Jr., M. Reiser, and G. L. Galati, p. 246.* A methodology for performance-tuning a virtual storage system is discussed. This methodology encompasses performance measurement, workload characterization, performance evaluation, and planned experimentation. Use of the methodology is illustrated by describing results of a case study involving the IBM Research Division's TSS/360 system.

The Power Profile—An installation management tool *by J. A. Laird, p. 264.* This paper describes a method used to estimate the electrical power required for a computer system and the heat generated by it. The method is a program that serves as an installation planning tool accessed principally by Installation Planning Representatives through communication terminals.

Computing center optimization by a pricing-priority policy *by S. B. Ghanem, p. 272.* Included in the user's price-priority decision is his cost of delay.

Briefly discussed for the general case, and shown in detail for the two-queue case, is the principle that the total cost to users of the computing service is minimized in a computing installation that has a price-priority service policy.

Productivity of computer-dependent workers *by D. N. Streeter, p. 292.* Beginning with a description of various degrees of computer dependency among workers, a model of the worker-computer process is constructed. The model demonstrates the characteristic forms of functional dependencies and suggests ways in which these dependencies can be evaluated.

Key among the many considerations discussed are such process characteristics as system congestion, needs and habits of users, and relative costs.

Volume 14, Number 4, 1975

Computer installation accounting *by H. M. Gladney, D. L. Johnson, and R. L. Stone, p. 314.* This paper examines the function and design of an accounting program package for a medium to large computer installation. A specific implementation is used to illustrate key points.

Evaluating system changes under uncontrolled workloads: a case study *by H. P. Friedman and G. Waldbaum, p. 340.* When a change to a computer system is evaluated under workload conditions that are not controlled, it is necessary to estimate to what extent system performance has been affected by the change and to what extent by variations in the workload. This paper describes a regression-analysis method by which such an estimate was made for a particular computer system.

Testing in a complex systems environment *by M. O. Duke, p. 353.* The testing problems in a complex systems environment are described with categorization by purpose of the tests and objectives to be achieved. The approach to testing and the methodology required to adequately test in the various testing categories are presented.

The methodology described, together with appropriate testing tools, can aid users in the migration to new operating systems and new versions of subsystems, as well as to accelerate their own application development.

Performance analysis of virtual memory time-sharing systems *by Y. Bard, p. 366.* The performance of VM/370 systems is analyzed in relation to the multiprogramming level and the user work load. Saturation conditions are examined, and methods for locating bottlenecks in the CPU, main storage, paging, and I/O subsystems are given. The paper also describes the data requirements, along with techniques for data collection and reduction. The techniques are illustrated with data from an actual case study.

Structured programming for virtual storage systems *by J. G. Rogers, p. 385.* The disciplines of structured programming and programming for virtual storage are examined to show how they affect each other.

Considerations and techniques are proposed that, when applied during the process of structured design and coding, produce programs that place fewer demands on the computer storage resources.

The techniques are illustrated by example programs.

Volume 15, Number 1, 1976

Systems Network Architecture: An overview *by J. H. McFadyen, p. 4.* Recent technological advances have allowed information processing and data storage capability to be distributed more easily from a central computer complex to remote user locations. Systems Network Architecture provides a unified systems structure for the contemporary teleprocessing environment that resulted from these advances. Using some current implementations as examples, this overview introduces the concepts on which the architecture is based and broadly describes the basic components of the structure. Specific architectural and implementation details can be found in the references.

The transmission subsystem in Systems Network Architecture *by P. G. Cullum, p. 24.* The components providing the means to transfer data from one end user to another within a system incorporating SNA comprise the transmission subsystem. This paper discusses the organization of the subsystem, its logical and physical aspects, and the components involved in its operation.

The role of the Network Control Program in Systems Network Architecture *by W. S. Hobgood, p. 39.* An integral part of SNA is the scheme for controlling the communications functions within a network. This scheme is the Network Control Program that resides within a communications controller of the network. Discussed are the operations of the major components of the NCP and their relationships.

The Virtual Telecommunications Access Method: A Systems Network Architecture perspective *by H. R. Albrecht and K. D. Ryder, p. 53.* As an access method, VTAM is influenced by many non-SNA considerations. However, this paper focuses on the SNA functions implemented by VTAM, discussing the components involved in some detail. Also included is a brief discussion of the historical factors that led to the conception of VTAM.

LABS/7—a distributed real-time operating system *by D. L. Raimondi, H. M. Gladney, G. Hochweller, R. W. Martin, and L. L. Spencer, p. 81.* A hierarchical distributed real-time computing system, LABS/7, provides facilities for attaching multiple IBM System/7s to a host System/360 or System/370. LABS/7 consists of a multiprogramming and multitasking supervisor for the System/7, a host commu-

nication facility that supports multiple satellite System/7s, and a high-level real-time language for user application development.

LABS/7 is operational in a variety of environments, including research, development, manufacturing, and clinical. The functional characteristics of the system are reviewed, performance is stated for well-defined situations, and experience with the system is reviewed with reference to some typical applications.

Penetrating an operating system: a study of VM/370 integrity *by C. R. Attanasio, P. W. Markstein, and R. J. Phillips, p. 102.* Discussed is a methodology for discovering operating system design flaws as an approach to learning system design techniques that may make possible greater data security.

Input/output has been found to be involved in most of the weaknesses discovered by a study team in a particular version of the system.

Relative design simplicity was found to be the source of greatest protection against penetration efforts.

Volume 15, Number 2, 1976

Experiments in line quality monitoring *by P. Bryant, F. W. Giesin, Jr., and R. M. Hayes, p. 124.* This paper describes an experimental line quality monitoring (LQM) system, which has been in regular operation in several IBM locations. The LQM system uses a System/7 to gather information about the analog phenomena that occur on telephone data transmission lines. It gathers the information without taking the lines out of service. Described are the ways in which the information from the LQM system is used in daily operations; for example, in one case, use of the LQM system contributed to an increase in circuit availability from 82 percent to 99 percent over a one-month period.

HIPO and integrated program design *by J. F. Stay, p. 143.* Discussed is a procedure of hierarchical functional design by which programming projects can be analyzed into system, program, and module levels. It is shown that program design is made more efficient by applying Hierarchy plus Input-Process-Output (HIPO) techniques at each level to form an integrated view of all levels.

Top-down development using a program design language *by P. Van Leer, p. 155.* Discussed is a program design language—a form of pseudocode—that has been developed and used to organize, teach, document, and develop software systems. An example of top-down program design illustrates the key steps in using the language: determining the requirements, abstracting the functions, expanding the functions, and verifying the functions.

Syntax and conventions of the language are given in an appendix.

Volume 15, Number 3, 1976

Design and code inspections to reduce errors in program development *by M. E. Fagan, p. 182.* Substantial net improvements in programming quality and productivity have been obtained through the use of formal inspections of design and of code. Improvements are made possible by a systematic and efficient design and code verification process, with well-defined roles for inspection participants. The manner in which inspection data is categorized and made suitable for process analysis is an important factor in attaining the improvements. It is shown that by using inspection results, a mechanism for initial error reduction followed by ever-improving error rates can be achieved.

Composite design facilities of six programming languages *by G. J. Myers, p. 212.* Examined are relationships between the methodology of composite design and six widely used programming languages. Strengths and weaknesses of composite design facilities of these languages are discussed. Based on this experience, language facilities for greater use of the potential of composite design are suggested.

A model of large program development *by L. A. Belady and M. M. Lehman, p. 225.* Discussed are observations made on the development of OS/360 and its subsequent enhancements and releases. Some modeling approaches to organizing these observations are also presented.

Volume 15, Number 4, 1976

The Peterlee Relational Test Vehicle—a system overview *by S. J. P. Todd, p. 285.* A high-level data-base system, the Peterlee Relational Test Vehicle (PRTV), provides flexible, interactive data-base support and functional extensibility. The user sees the system primarily through a programming language called ISBL, which is designed for manipulating bulk data held in relations. PRTV is not a full-fledged data-base system, but rather an evolving prototype which is expected to aid in solving some of the problems that have been encountered in using relational data bases. PRTV embodies research both in data-base language design and in efficient implementation techniques.

Interactive modeling of computer systems *by M. Reiser, p. 309.* This paper provides a nontechnical introduction to queuing network concepts, a short introduction to results of research into analytical modeling methodology, and a set of simple but typical examples that illustrate the application of that methodology to problems of computer performance.

Service levels: A concept for the user and the computer center *by L. J. Lewis, p. 328.* Service levels represent an important concept that can be applied toward the solution of difficult problems surrounding communications between users and providers of data processing services. In this paper, this concept is described in terms of the architecture, which defines

the scope and structure of service information. The paper further translates the architecture into data processing terminology by presenting the data-base structure and data elements related to service levels. The paper also addresses the post-processing of the data base, a step essential to properly communicating service-level information.

An APL emulator on System/370 *by A. Hassitt and L. E. Lyon, p. 358.* Emulation of an APL machine on a System/370 is exemplified by the APL Assist, a microprogram that enables APL expressions and defined functions to be executed at the hardware level. This paper discusses what the APL Assist does, how it works, and the way it interacts with System/370 software. Execution times for APL programs with and without the Assist are compared.

Volume 16, Number 1, 1977

A user-oriented data-base retrieval system *by A. U. Jones, p. 4.* Discussed is an experimental, specialized data-base system developed for users who do not require the sophisticated resources of the large data-base systems but do require many of the capabilities they provide. Data manipulation and retrieval within a data base are made available for the nonprogrammer user. The function and design attributes of the system are described including the reasons for basing the system on APL functions.

An APL interpreter and system for a small computer *by M. Alfonseca, M. L. Tavera, and R. Casajuana, p. 18.* The design and implementation of an experimental APL system on the small, sensor-based System/7 is described. Emphasis is placed on the solution to the problem of fitting a full APL system into a small computer.

The system has been extended through an I/O auxiliary processor to make it possible to use APL in the management and control of the System/7 sensor-based I/O operations.

The IBM 5100 and the Research Device Coupler—A personal laboratory automation system *by H. Cole and A. A. Guido, p. 41.* A small laboratory automation system has been developed by using the IBM 5100 Portable Computer in conjunction with the Research Device Coupler. This compact system provides a dedicated, high-level-language computer and a versatile data acquisition and control interface for experiments in which data rates do not exceed 9600 baud. Two experiments exemplify the use of the system.

The Research Device Coupler described in this paper is a prototype of the IBM 7406 Device Coupler.

A method of programming measurement and estimation *by C. E. Walston and C. P. Felix, p. 54.* Improvements in programming technology have paralleled improvements in computing system architecture and materials. Along with increasing knowledge of the system and program development processes, there has been some notable research into programming project measurement, estimation, and planning. Discussed is a method of programming project productivity estimation. Also presented are preliminary results of research into methods of measuring and estimating programming project duration, staff size, and computer cost.

Volume 16, Number 2, 1977

The information management system IMS/VS, Part I: General structure and operation *by W. C. McGee, p. 84.* The first of a five-part series of papers on IMS/VS, this paper discusses the architecture, goals, and objectives of that information management system, the purpose of which is to facilitate Data Base/Data Communication applications. Subsequent papers present data base facilities, batch processing, data communication, and transaction processing in greater depth.

The information management system IMS/VS, Part II: Data base facilities *by W. C. McGee, p. 96.* The structuring of a data base and its implementation through the use of several access methods are presented. A number of implementation methods are described and compared. Also presented are retrieval, updating, reorganization, and recovery. Other parts of this five-part series on IMS/VS include objectives and architecture, batch processing, data communication, and transaction processing.

The information management system IMS/VS, Part III: Batch processing facilities *by W. C. McGee, p. 123.* Batch processing is described from the application programmer point of view. Restart and recovery techniques are also discussed. Other parts of the series discuss objectives and architecture, data base structuring, data communication, and transaction processing facilities.

The information management system IMS/VS, Part IV: Data communication facilities *by W. C. McGee, p. 136.* Aspects of IMS/VS communication facilities discussed include networks, terminals, security facilities, editing, and formatting. Other parts in this series on IMS/VS include an introductory part on objectives and architecture, data base facilities, batch processing facilities, and transaction processing.

The information management system IMS/VS, Part V: Transaction processing facilities *by W. C. McGee, p. 148.* Transaction processing and a further discussion of communication facilities are presented. Also discussed are system operations, including startup and shutdown, restart, and system monitoring. Other parts in this series present IMS/VS objectives and architecture, data base facilities, batch processing, and data communication.

A high-performance DB/DC system *by J. E. Siwiec, p. 169.* Discussed is the evolution of a computerized airline reservation system from its early form up to the present version. Data base allocation, accessing techniques, and data communications of the system are described. The system consists of the Pro-

grammed Airline Reservation System (PARS) and its control program called ACP.

Volume 16, Number 3, 1977

Data structures and data accessing in data base systems past, present, future *by M. E. Senko, p. 208.* A broad range of commercial and research data base systems are analyzed. Common characteristics are discussed. These systems, which have roots in older filing systems and in punched card systems, are grouped into the three categories of hierarchic, network, and single-level models. Also presented is work on the standardization of data base systems. Research toward the discovery of new commonalities is also discussed. This paper is based on an extensive published literature.

CICS/VS and its role in Systems Network Architecture *by D. J. Eade, P. Homan, and J. H. Jones, p. 258.* Evolution of the Customer Information Control System/Virtual Storage (CICS/VS) is discussed, along with a description of how CICS/VS manages a pre-SNA teleprocessing network. Following a brief review of SNA (Systems Network Architecture), the role of CICS/VS within an SNA environment is described. The paper concludes by outlining SNA's advantages to CICS/VS.

Automated logical data base design: Concepts and applications *by N. Raver and G. U. Hubbard, p. 287.* This paper describes the design effort for an integrated data base and then develops techniques for automating significant portions of the labor. These techniques have been incorporated in a program to provide an effective data base design tool (Data Base Design Aid) in current use. The processes involved with this aid are discussed.

Volume 16, Number 4, 1977

Query-by-Example: A data base language *by M. M. Zloof, p. 324.* Discussed is a high-level data base management language that provides the user with a convenient and unified interface to query, update, define, and control a data base.

When the user performs an operation against the data base, he fills in an example of a solution to that operation in skeleton tables that can be associated with actual tables in the data base. The system is currently being used experimentally for various applications.

Design techniques for a user controlled DB/DC system *by G. F. Heyne and C. J. Daniel, p. 344.* The flexibility and usefulness of an integrated data base can be limited by constraints inherent in the data base management system. One such system, IBM's Information Management System (IMS), has been extended in terms of data independence, access control, data integrity, and user communication by a group of techniques integrated into an IMS application called the Product Development Communication and Control (PDCC) system. This paper describes the IMS extensions provided by PDCC and discusses the system's implementation.

PDCC is the prototype of the IMS Application Development Facility, an IBM Installed User Program.

Storage and access in relational data bases *by M. W. Blasgen and K. P. Eswaran, p. 363.* A model of storage and access to a relational data base is presented. Using this model, four techniques for evaluating a general relational query that involves the operations of projection, restriction, and join are compared on the basis of cost of accessing secondary storage. The techniques are compared numerically and analytically for various values of important parameters. Results indicate that physical clustering of logically adjacent items is a critical performance parameter. In the absence of such clustering, methods that depend on sorting the records themselves seem to be the algorithm of choice.

MARC: MVS archival storage and recovery program *by J. P. Considine and J. J. Myers, p. 378.* A newly designed and implemented automated storage hierarchy management system that operates under MVS is described. The needs for economical archival storage that at the same times makes possible the efficient retrieval of users' data are reviewed. Discussed in detail is the fulfillment of this requirement that is provided by the automated management of MVS/TSO on-line data storage space that includes the Mass Storage System (MSS) and other storage devices in a hierarchy. Experience with the system is summarized.

An input-output econometric model *by K. S. Sarma, p. 398.* A model of the type used to forecast the effects on an industry of changes in the national economy is described. The components representing input data and the type of output from such a model are discussed.

This paper is intended as a tutorial discussion of an advanced application in data processing.

Volume 17, Number 1, 1978

IBM's Santa Teresa Laboratory—Architectural design for program development *by Gerald M. McCue, p. 4.* The special needs of the computer programmer in terms of working space, furniture design, access to terminals and conference rooms, and overall working environment led IBM to construct a facility intended to enhance programmer productivity in a development environment. That facility is the Santa Teresa Laboratory in San Jose, California, designed by MBT Associates of San Francisco. This essay discusses the programmer's needs, how they were perceived, and the process by which they led to unique design concepts, as well as the architectural philosophy underlying the design process.

A method for the time analysis of programs *by S. L. de Freitas and P. J. Lavelle, p. 26.* Discussed is a technique for investigating the efficiency of compiled

programs. Based on research that uses FORTRAN as a test subject, the method is more widely applicable. Time analyses show programmers points at which efficiencies may be increased. Also discussed are uses of the technique for comparing the efficiencies of compilers and languages, and for making performance/cost analyses. Presented are validation data for the method under several sets of conditions.

Measuring programming quality and productivity *by T. C. Jones, p. 39.* Discussed is the unit-of-measure situation in programming. An analysis of common units of measure for assessing program quality and programmer productivity reveals that some standard measures are intrinsically paradoxical. Lines of code per programmer-month and cost per defect are in this category. Presented here are attempts to go beyond such paradoxical units as these. Also discussed is the usefulness of separating quality measurements into measures of defect removal efficiency and defect prevention, and the usefulness of separating productivity measurements into work units and cost units.

The Extended Control Language of MPSX/370 and possible applications *by L. Slate and K. Spielberg, p. 64.* Some large-scale linear and especially mixed-integer programming problems, and the underlying practical decision-making situations, have so far been solved with only limited success. A new control language for IBM's system MPSX-MIP/370 permits recursive use of the basic system and easy access to its elements, and therefore appears to offer great potential for new advances.

The paper first describes the facilities of the language, called the Extended Control Language, and the interfaces to the system and gives a number of representative illustrative uses. It then considers a number of basic applications of the system and possible heuristic and algorithmic approaches to difficult problems, often very large problems with structure, which may now become more easily solvable or tractable for the first time.

Solving the installation scheduling problem using mixed integer linear programming *by R. Chen, H. Crowder, and E. L. Johnson, p. 82.* The installation scheduling problem involves finding a program for installing a large number of sizes and types of items (e.g., machines) over time so as to optimize some measure (e.g., initial capital investment), subject to various resource constraints. Examples of this problem are scheduling the installation of point-of-sale terminals in supermarket and retail chains, and teller terminals in banks.

We have formulated the installation scheduling problem as a mixed integer linear program and developed a computer code for solving the model. By using techniques for exploiting the special structure of the model, our formulation allows rather quick solution times.

Volume 17, Number 2, 1978

A cryptographic key management scheme for implementing the Data Encryption Standard *by W. F. Ehrsam, S. M. Matyas, C. H. Meyer, and W. L. Tuchman, p. 106.* Data being transmitted through a communications network can be protected by cryptography. In a data processing environment, cryptography is implemented by an algorithm which utilizes a secret key, or sequence of bits. Any key-controlled cryptographic algorithm, such as the Data Encryption Standard, requires a protocol for the management of its cryptographic keys. The complexity of the key management protocol ultimately depends on the level of functional capability provided by the cryptographic system. This paper discusses a possible key management scheme that provides the support necessary to protect communications between individual end users (end-to-end encryption) and that also can be used to protect data stored or transported on removable media.

Generation, distribution, and installation of cryptographic keys *by S. M. Matyas and C. H. Meyer, p. 126.* A key controlled cryptographic system requires a mechanism for the safe and secure generation, distribution, and installation of its cryptographic keys. This paper discusses possible key generation, distribution, and installation procedures for the key management scheme presented in the preceding paper.

Cryptography architecture for information security *by R. E. Lennon, p. 138.* Information being transferred from point to point over a public communications carrier or stored on portable media can be protected, by the use of cryptography, from accidental or intentional disclosure. Control functions are required to ensure synchronization of the process. In a communications environment, the control functions become logically part of the network architecture. IBM's Systems Network Architecture (SNA) has been extended to allow the use of cryptography when sensitive information is being processed. Architectural similarities for the file environment are discussed.

Administrative control of computing service *by H. M. Gladney, p. 151.* The complexity of necessary administrative controls for computer service exceeds the capabilities of clerical methods. This paper presents a practical method for describing the external administrative environment in a data base which can be used by the operating system for dynamic enforcement of limits. An attempt is made to address consistently the different forms of data processing that may be concurrent in a single installation; included are general purpose time sharing, transaction oriented computing, and scientific computing. Described is the architecture of an operating system component that could be regarded as the interface between administrative security mechanisms and the security features of the system software.

Running prototypes exist. The long-range intention is to streamline such computer facility management functions as controlling access to specific services, processing power, and storage space; controlling access to the system data base; and gathering statistics needed for planning. Convenience to users is not degraded by the security mechanisms, but in fact is enhanced.

Data processing spheres of control *by C. T. Davies, Jr., p. 179.* There has long been a need for better definition of the audit and control aspects of data processing applications. This paper attempts to satisfy that need and thereby provide a framework for improving communication between systems analysts and computer scientists. It introduces the concept of spheres of control, which are logical boundaries that exist in all data processing systems, whether manual or automated. The paper describes their essential properties and portrays them as they relate to each other in the batch, on-line, and in-line processing environments. Included are spheres of control that define process bounding for such purposes as recovery, auditing, process commitment, and algorithm (procedure) replacement.

Volume 17, Number 3, 1978

Job networking *by R. P. Crabtree, p. 206.* This paper discusses the evolution of a facility, generally called job networking, that permits job-related information to be sent between programming system components operating on computing systems that are attached to a communications network. The capabilities of the programs that implement this facility are described along with the events that led to its development, the value of the facility to the user, and ways in which it can be extended to work with other forms of communication networking.

Network job entry facility for JES2 *by R. O. Simpson and G. H. Phillips, p. 221.* Job entry subsystems have been developed to provide operating systems with an interface for managing some of the workload of computing facilities. One of the job entry subsystems for OS/VS2 has been further enhanced with the addition of a network job entry facility that allows full access to a network of computers in a manner consistent with a local operation. This paper discusses the design objectives, implementation, and extensions of that facility.

Experiments in computer-aided graphic expression *by J. F. Musgrave, p. 241.* This paper presents a series of graphic design experiments using an experimental color graphic display system. Design principles and capabilities of the experimental color graphic display system are discussed from a graphic designer's point of view. The system allows a designer to choose freely among 128 different colors, various form modes, and collage capabilities, including image mixing. The designer need be neither a programmer nor one who understands the technical aspects of the system to use it creatively. Experimental results are shown visually here, some of which have been used as cover designs for IBM publications.

A time-sharing display terminal session manager *by J. M. McCrossin, R. P. O'Hara, and L. R. Koster, p. 260.* Display terminals, although faster than typewriter devices, do not implicitly create records of the user's interactive sessions. Based on the premise that a display terminal session facility that also has the record-keeping functions of typewriter terminals would increase productivity, a research project was undertaken that has resulted in the session manager discussed. Experience with the system is summarized.

Enhanced problem determination capability for teleprocessing *by J. B. Ford, p. 276.* Determining network problems is an important task that can be difficult and even tedious. This capability has been enhanced for teleprocessing users and service personnel through service aids developed for their use. The service aids gather error data and provide displays that assist in analyzing system quality and in determining specific problems. This paper describes the evolution and operation of these aids.

Performance tuning in OS/VS2 MVS *by T. Beretvas, p. 290.* A procedure of observation and correction is presented for the coarse tuning of an MVS system by analyzing it into its software and hardware components and increasing their efficiency successively. The method involves adjustments to swapping, the input-output load, the CPU load, main storage, and the system resources manager. Also discussed are performance measures necessary to characterize a system, tools to tune a system, and various aspects of data gathering and the effects of adjustments on system parameters. Two illustrative case histories are also given.

Volume 17, Number 4, 1978

Distributed data processing *by A. L. Scherr, p. 324.* Today there is a wide range of choice for configuring the data processing facilities of an organization—centralized systems, decentralized systems, small computers, and networks of communicating computers—for distributed data processing. This paper considers the factors that relate to organizations and their data processing requirements and to the various possible data processing configurations. Price-performance ratio, organizational needs, and other factors that recommend the flexibility of distributed data processing are discussed in detail. Also discussed are possible distributed data processing architectures, choice criteria, communications, and application and operating system design principles.

National Westminster Bank mass storage archiving *by C. M. Gravina, p. 344.* The problem of retrieving records at random from a very large archival data base has not previously been effectively soluble by data processing techniques. This paper describes

such an application, in which a large bank in the United Kingdom uses the IBM 3850 Mass Storage System for storing and retrieving customer account statements.

The development of software systems to aid in physical planning *by B. S. Smedley, p. 359.* Physical planning for geographic areas such as cities, counties, or regions can be greatly simplified if the planner can display the entity under consideration together with land-use and socioeconomic data and can interact easily with that data to modify the presentation and redisplay it. Presented in this paper are a system and a language to aid such physical planning and user experience with the system and language. Further research on graphic presentation, the incorporation of models and statistical routines are also discussed.

Data Stream Linkage Mechanism *by J. P. Morrison, p. 383.* Using a programming discipline called the Data Stream Linkage Mechanism (DSLM), a program can be built by linking program modules to form a network through which data passes. The network is specified by the program designer using a mixture of precoded and custom coded modules. This linkage technique and the capabilities that result from it constitute an approach to programming that is radically different from conventional techniques. It can increase the productivity of programmers and can result in programs that are easier to understand and to maintain.

This paper gives examples based on a specific implementation of DSLM and describes some of the experience gained from the implementation over the last six years.

Performance investigations with a DOS/VS-based operating system model *by W. Kraemer, p. 409.* This paper describes an operating system model that is based mainly on the DOS/VS supervisor but also reflects various design alternatives, providing a flexible tool for operating system design and tuning. The model is characterized by the subdivision of I/O activity into normal I/O, page I/O, and fetch I/O, corresponding to the different supervisor services involved. The model has been evaluated by analytical queuing methods in a set of APL functions that allow a flexible specification of the supervisor, the configuration, and the workload. Validation has been done by simulation and by benchmarking of a real system.

The main features of this performance tool are described, and its capabilities are illustrated by performance results that show the impact of workload and various supervisor changes on system performance.

A performance model of MVS *by Willy W. Chiu and We-Min Chow, p. 444.* Capacity planning, a major function of computer installation management, has the objective of determining cost-effective configurations to provide acceptable user service and system performance levels according to workload changes. The use of a performance predictive model is essential in the capacity planning process. This paper presents a research case study of the development of a performance model called PMOD, for the IBM OS/MVS operating system. The goal of the model is to predict user response times and system performance for different scheduling parameters, workloads, and configurations, with reasonably simple input requirements and fast run times. Both the validation and usage of the model for capacity planning and system tuning are discussed.

Volume 18, Number 1, 1979

VM/370—a study of multiplicity and usefulness *by L. H. Seawright and R. A. MacKinnon, p. 4.* This paper is an overview of IBM's Virtual Machine Facility/370. It describes the virtual machine concept and its capabilities and implementation in VM/370. Two components of VM/370 are discussed—the control program and the Conversational Monitor System. The usefulness of VM/370 in multiple and diverse environments is covered. New developments in VM/370 from hardware assists to system extensions, networking, and handshaking are briefly described as an introduction to the rest of the papers in this issue.

The changing virtual machine environment: Interfaces to real hardware, virtual hardware, and other virtual machines *by R. A. MacKinnon, p. 18.* This paper is a survey of changes to virtual machine interfaces, implementation, architecture, and simulation techniques as they affect IBM System/370 and 303X (3031, 3032, 3033) processors, the system control program to which virtual machines interface, and other virtual machines executing on the same real computing system or elsewhere. The paper seeks to summarize such changes and provide a perspective on the virtual machine environment. New uses of virtual machine subsystems are discussed as they relate to inter-virtual-machine communication.

VM/370 asymmetric multiprocessing *by L. H. Holley, R. P. Parmelee, C. A. Salisbury, and D. N. Saul, p. 47.* The design and implementation of VM/370 attached processor support is discussed from the point of view of adding radical new function to an existing operating system. Three major design decisions are described, and performance is analyzed as it relates to those decisions.

A formal approach for communication between logically isolated virtual machines *by R. M. Jensen, p. 71.* The growing use of the virtual machine concept has resulted in the necessity for communication between the virtual machines. The design and operation of the Virtual Machine Communication Facility is discussed as an approach to offering such communication. The facility is an interface allowing a logical connection between two or more virtual machines. Potential applications for this facility conclude the discussion.

Virtual Control Storage—security measures in VM/370 *by C. R. Attanasio, p. 93.* The architecture

of a virtual machine system has specific advantages over that of conventional operating systems because virtual machines are well separated from one another and from the control program. This structure requires that a protected, multi-user resource manager be placed in a distinct virtual machine because the protection domain and scheduling unit are one entity, the virtual machine. But cooperation between distinct virtual machines necessarily entails scheduling overhead and often delay.

This paper describes an experimental extension to VM/370 whereby a distinct execution and data domain (Virtual Control Storage) is made available to virtual machines that require access to a resource manager, without requiring a change in the scheduling unit. Thus scheduling overhead and delays are avoided when transition is made between user program and resource manager. A mechanism is described for exchanging data between execution domains by means of address-space mapping.

Evolution of a virtual machine subsystem *by E. C. Hendricks and T. C. Hartmann, p. 111.* Early investigation of virtual machine subsystem flexibility centered on telecommunications support and intercomputer networking and proceeded in two phases. The first phase focused on an experimental program for the virtual machine control program CP-67 that supported remote work stations and pioneered intercomputer spool communications. The results of that effort inspired a second effort in the same area with some significant redirection. This second phase ultimately led to the remote spooling communications subsystem component of VM/370, the VM/370 networking package (VNET), and a large network of interactive computer systems within IBM. These phases are discussed along with suggestions for several continuing lines of work based on current results.

Managing VM/CMS systems for user effectiveness *by W. J. Doherty and R. P. Kelisky, p. 143.* Discussed in this paper is a computing center management methodology based on the premise that the computer user's time and work product are valuable. Experience in the use of interactive systems in a research environment from 1965 to the present time is presented. Current user experience and management of VM/CMS are emphasized. The use of computers as tools for extending users' powers of memory and logic and the development of new methods of managing VM/CMS are discussed in detail.

State sampling of interactive VM/370 users *by W. H. Tetzlaff, p. 164.* Sampling the state of interactive computer users of hardware and programming in a time-sharing system leads to an understanding of delays to users caused by contention for resources. This paper discusses user state sampling by means of a program called VM/Monitor on the interactive time-sharing system, VM/370, although the methodology is applicable to other time-sharing systems. Also discussed are system bottleneck detection and secondary tuning after bottlenecks have been found. Possible extensions of the technique are also presented.

Volume 18, Number 2, 1979

Computing and communications—A perspective of the evolving environment *by L. M. Branscomb, p. 189.* Telecommunications regulation is an important public policy consideration and is presently the topic of much debate. The author presented a technical viewpoint of this topic, particularly with regard to possible applications and how the data processing industry could be involved, during the Keynote Panel Session at IEEE Compcon '78 held on September 6, 1978. His presentation is printed here.

An introduction to network architectures and protocols *by P. E. Green, p. 202.* This tutorial paper is intended for the reader who is unfamiliar with computer networks, to prepare him for reading the more detailed technical literature on the subject. The approach here is to start with an ordered list of the functions that any network must provide in tying two end users together, and then to indicate how this leads naturally to layered peer protocols out of which the architecture of a computer network is constructed. After a discussion of a few block diagrams of private (commercially provided) and public (common carrier) networks, the layer and header structures of SNA and DNA architectures and the X.25 interface are briefly described.

Public data networks: Their evolution, interfaces, and status *by J. R. Halsey, L. E. Hardy, and L. F. Powning, p. 223.* The service history of public data networks began in 1972. Since that time the number of such networks and the variety of services they offer has increased and continues to do so. In this paper, some of the networks, their characteristics, and the international network interface recommendations are briefly described.

SNA and emerging international standards *by F. P. Corr and D. H. Neal, p. 244.* Public data networks are now being designed and implemented to handle the expansion of data communications. The interaction of Systems Network Architecture (SNA) and the international standards now being developed is discussed. A provisional architecture model is used as the basis for discussion, and SNA is compared to each level of the model.

SNA multiple-system networking *by J. P. Gray and T. B. McNeill, p. 263.* Systems Network Architecture (SNA) has evolved from an architecture that supported implementation of tree networks rooted in a System/370 to an architecture that supports multiple-system networks with capabilities such as alternate paths and parallel links. This paper describes the major SNA enhancements that have been implemented for multiple-system networks. As network configurations have become more complex, the problems associated with network growth, change, failures, recovery, and flow control have required

solutions that permit continuous network operation. The SNA enhancements that address these problems are also discussed.

Routing and flow control in Systems Network Architecture *by V. Ahuja, p. 298.* Systems Network Architecture (SNA) has been enhanced to include features that address the topological, routing, congestion, reliability, and availability problems of networks. An important aspect of this new release of SNA is that it allows multiple active routes between network nodes. Multiple routing permits sessions between network users to use alternate routes in case of unexpected or planned route disruptions. In this paper, the multiple routing architecture of SNA is described.

An unrestricted data flow into the network can cause long delays and buffer depletion. Network congestion can be avoided by employing flow control mechanisms at both the local (node) and global (network) levels. This paper focuses on global flow control and describes the adaptive traffic-pacing "window" size algorithm that is the basis of the global flow control in SNA.

Evolution of a laboratory communication network *by R. S. Moore, p. 315.* This paper describes the IBM System Communications Division network and, using that network as an example, discusses some of the practical problems associated with providing computational and communication services to remote and local user communities.

Potential technology implications for computers and telecommunications in the 1980s *by W. D. Frazer, p. 333.* This essay looks at some future effects on computing and telecommunications of some countervailing technology trends. The projected time is the mid-1980s, and the technology trends considered are those of Large Scale Integration (LSI), related storage technologies such as charge coupled devices (CCDs) and magnetic bubbles, optical fiber transmission systems, and satellites. The principal focus is upon the implications of these trends for distributed processing and computer networks.

Volume 18, Number 3, 1979

Performance analysis of complex communications systems *by H. M. Stewart, p. 356.* This paper discusses the designing of complex teleprocessing systems using a discrete simulation modeling tool, the Systems Network Analysis Program/Simulated Host Overview Technique (internally and informally called SNAP/SHOT). This modeling tool aids in designing computer communications systems composed of local and remote terminals, teleprocessing lines, host processors that control the teleprocessing lines, and interconnected communications systems. The model is capable of analyzing both tree- and mesh-structured networks.

A distributed information system study *by K. Ziegler, Jr., p. 374.* This paper is a discussion of a methodology, a distributable information system model, and an experiment used to identify potential problems for supporting such a system. The experimental model was designed and implemented in an evolutionary manner for the purpose of studying the feasibility of a system with the postulated attributes. Incentives for distribution and design of the study introduce the two main topics—the study model itself and the implementation of the study model. Results of the study provide insights into such factors in distributed information system structural design as intercomponent communication, system control, and recovery philosophy.

An office communications system *by G. H. Engel, J. Groppuso, R. A. Lowenstein, and W. G. Traub, p. 402.* In developing a prototype of an office communications system, an office study was first done to specify requirements for the prototype. The study focused on the productivity of three groups of employees: principals, clerical personnel, and secretaries. With requirements set by the management of the office used as a framework, application requirements for end users of an office communications system were established. From a subset of these requirements the prototype was developed.

The prototype system was designed as an experimental learning system to provide managers and professionals with an easy, fast, and direct method for handling their business communications. The prototype was set up on IBM premises for testing and evaluation. Results of this operation are included in the discussion.

A research perspective on computer-assisted office work *by A. M. Gruhn and A. C. Hohl, p. 432.* The integrated office system of the future that relies upon computerized applications for most routine work will have to be a friendly system that can be used by individuals with a minimum of training and no previous computing experience. A discussion of the computer-assisted aids to office work that have evolved at the IBM Thomas J. Watson Research Center provides a preview of the possibilities for future office systems based on computers. We describe tasks that have been computerized, the environment in which automated office applications are used, and the reactions of people who use the computer for routine office work.

Automatic programming for energy management using sensor based computers *by M. J. Shah, p. 457.* An automatic programming approach has been developed for the use of sensor based computers (IBM System/7 and Series/1) for energy management in buildings. The purpose is to aid the facilities engineer who is unfamiliar with programming and who requires a system that can be defined by a sequence of questions and answers. Programmers can add or modify application source programs to extend the system to other user-defined functions.

Volume 18, Number 4, 1979

An integrated approach to centralized communications network management *by R. A. Weingarten, p. 484.* Recent technological advances have increased the size and number of teleprocessing networks as well as broadened their scope and complexity. This added size and complexity has magnified the need for communications network management. One approach to communications network management is to provide centralized control which is integrated into the Systems Network Architecture user network. This approach is described in this paper by focusing on two program products that provide centralized operator control and problem determination capabilities for a network.

An operating system for distributed processing—DPPX *by S. C. Kiely, p. 507.* The Distributed Processing Programming Executive (DPPX) is a new, full-function operating system designed to support distributed processing with the IBM 8100 Information System. The functional requirements of distributed processing and their solutions in DPPX are discussed. The structure of the operating system is outlined, and its advantages are analyzed. Highlighted are particular characteristics of the DPPX structure that uniquely support distributed processing.

I/O facilities of the Distributed Processing Programming Executive (DPPX) *by H. R. Albrecht and L. C. Thomason, p. 526.* This paper introduces the input/output facilities of DPPX, the Distributed Processing Programming Executive for the IBM 8100 Information Processing System. Design requirements and alternatives are discussed, as well as the general structure of the services that implement the I/O facilities. Services that support specific I/O resources, such as disk storage and communication devices, are related to the general structure. The paper considers some of the problems in designing a general structure to support a wide range of services, and it briefly describes the interface architecture used to solve these problems.

Data Management for the Distributed Processing Programming Executive (DPPX) *by A. K. Fitzgerald and B. F. Goodrich, p. 547.* The Data Management component of the new IBM 8100 Distributed Processing Programming Executive (DPPX) provides for the storage and retrieval of data on disk and tape. Its objectives are to support a broad range of functions and be easy to use, be easily extendible, and entail minimal cost for the user. The Data Management component is designed to meet those objectives by means of a layered structure, an improved concept of device independence, and the use of catalogs.

Design of the IBM 8100 Data Base and Transaction Management System—DTMS *by F. C. H. Waters, p. 565.* Transaction applications have specialized requirements for scheduling and data base support. This paper describes the procedure by which those requirements were identified during the design of a data base and transaction management program for the IBM 8100 Information System. It also provides an overview of the program structure that evolved to satisfy the functional requirements.

Distributed processing: An assessment *by H. Lorin, p. 582.* Highlighted in this technical essay are discussions of the nature of distributed systems, design processes associated with the distribution of processing, and the conditions under which benefits accrue. The essay concentrates on some of the major benefits expected from distributed systems so as to provide a context in which to judge particular designs and their benefits. Among the judgment-informing considerations are the following: centralized management, historical relationships with on-line systems, reliability and fail-soft, security and privacy, system growth and capacity limitations, and fitting the system to the organizational structure.

Volume 19, Number 1, 1980

Overview of the capacity planning process for production data processing *by L. Bronner, p. 4.* An overview of techniques available to address capacity planning in the production data processing environment is presented. The production data processing system is briefly described and its capacity is quantified. The measurement tools, reports, and data required to implement a capacity planning program are discussed. Modeling and prediction are placed in perspective with the overall objectives of the capacity planning process. Personnel (managerial and technical) and organization considerations are also discussed.

A capacity planning methodology *by J. C. Cooper, p. 28.* Discussed is a capacity planning procedure called USAGE. Various business elements to be individually measured and tracked are presented. Outlined are methods for estimating workload growth. Separate limits of capacity for on-line workload and batch workload demand are discussed. A simple graphic presentation procedure is included to communicate the results of a study to those who need the information for making business decisions.

System capacity and performance evaluation *by D. C. Schiller, p. 46.* The performance of MVS (Multiple Virtual Storage) systems can be predicted for changes in workload and environment by an IBM marketing aid informally called SCAPE (for System Capacity and Performance Evaluation). Written in FORTRAN, the programs use simple queuing formulas with empirical modifications. Response times for complex workloads (IMS, CICS, TSO, and batch) through the CPU and auxiliary storage are expressed as functions of application loads and other parameters that define the system's environment. SCAPE can predict the effect on performance of different CPU models, larger memory, additional channels, additional direct-access storage, larger block sizes, and alternate workload projections.

Modeling considerations for predicting performance of CICS/VS systems *by P. H. Seaman, p. 68.* Various aids and tools are used in capacity planning. One such aid, an analytic model, is discussed in this paper. Both the decisions made in the development of an aid and the way the aid is used are examined. Characteristics of a good planning aid are emphasized with the analytic model serving as the example.

The role of detailed simulation in capacity planning *by H. C. Nguyen, A. Ockene, R. Revell, and W. J. Skwish, p. 81.* A number of performance prediction methods are available to IBM marketing personnel. This paper describes one such method, which predicts the effects of changes in IBM 3790 and 8100 distributed processing systems and in teleprocessing networks. Such changes may involve system features (such as line protocols), the introduction of new applications, or volume growth in an otherwise static system. The technique makes use of a detailed simulator, informally called FIVE, in conjunction with a system monitor and data analysis program. Its use can make substantial performance information available at relatively low cost.

An MVS tuning approach *by R. M. Schardt, p. 102.* Experience contributes important knowledge in many procedures. From the experience gained in tuning the Multiple Virtual Storage (MVS) operating system, a set of guidelines to help MVS installations avoid performance problems are suggested, along with an approach to tuning an MVS system. The guidelines can help a performance analyst isolate the cause of a performance problem. Not all possible problems that might be encountered are included. It is found that most MVS performance problems are a result of poor workload management and are often related to I/O activities.

A sidestream approach using a small processor as a tool for managing communication systems *by J. R. Leach and R. D. Campenni, p. 120.* The term management implies the achievement of objectives through effective use of resources. Management style relates to the various approaches used in pursuit of those same objectives. There can be various management tools contributing to an effective management solution. In this article we will be discussing one such tool.

This management tool reflects an approach to communication systems management in which the management functions are physically separated from the host computers driving a communications network. These management functions are packaged on a small IBM processor base (i.e., sidestream processor) and designed for use in a centralized network management center environment. The management functions included in this sidestream processor tool relate to problem management, change management, project scheduling and tracking, network control, and network configuration.

Systems management *by R. A. Bird and C. A. Hofmann, p. 140.* Effective systems management is dependent on two factors—visibility of the data required for systems management and a structured, disciplined management system to effectively utilize this information. Discussed in this paper are two programs to assist in dealing with these factors. These host programs use VSAM data bases and are accessed via CICS/VS or IMS/VS. The programs provide applications to assist in problem management, change management, network configuration, and problem determination. These functional application tools are described in terms of their content and their relationships with the overall systems management tasks.

Volume 19, Number 2, 1980

Logical distribution of applications and data *by C. T. Baker, p. 171.* A distributed data processing system is composed of a set of nodes that are interdependent yet capable of operating autonomously. This paper describes a procedure for controlling the interdependencies and nodal autonomies with a logical distribution of applications and their data. The procedure is illustrated with data that were obtained from an on-line operations planning and control system at a steel mill.

Distributed processing communications software support for operation within an SNA network *by E. S. Harrison, p. 192.* The Distributed Processing Programming Executive (DPPX) operating system has network configuration requirements placed on it. This paper discusses those requirements and the way in which they are met, including those that result from the various configurations possible with a DPPX system. In addition, the unique way in which terminal resources are supported in DPPX and the dynamic approach to resource definition in the DPPX system are described. Finally, application definition and application usage of the network configuration capabilities of DPPX are discussed.

System contention analysis—An alternate approach to system tuning *by A. Yuval, p. 208.* Many existing monitors that are intended to assist in system tuning are based on the utilization approach which focuses on the active time of the system resources and activities and their users. This paper presents an alternative approach that is based primarily on the analysis of the contention in the system. The focus here is on the queuing delay time of the users and their activities when accessing the system resources.

Utilization and contention are two different ways of looking at the system. The two approaches complement each other, yet each may serve a different purpose or address different performance objectives. A prototype monitor was implemented on MVS (Multiple Virtual Storage) to produce the information necessary to continue investigations in contention analysis.

Data base security: requirements, policies, and models *by C. Wood, E. B. Fernandez, and R. C. Summers, p. 229.* This paper surveys some aspects

of data base security, with emphasis on basic principles and ways to express security requirements. Security policies and theoretical models are considered in detail, and the models are used to compare the security features of some data base management systems.

The IPS cryptographic programs *by A. G. Konheim, M. H. Mack, R. K. McNeill, B. Tuckerman, and G. Waldbaum, p. 253.* Cryptographic methods of data protection have taken on new importance as computers have become faster and as strong cryptographic algorithms, such as the Data Encryption Standard (DES), have become available. But a standard encipherment technique is only the first step in applying cryptography in a computing center. This paper discusses the Information Protection System (IPS), a set of cryptographic application programs designed to use the DES algorithm in a working computing center. In designing IPS, several important augmentations of DES were formulated. IPS was first implemented to help increase computing-center security at the IBM Thomas J. Watson Research Center and is now widely installed at other IBM locations. IPS is not an IBM product and is not available for use outside IBM, but many cryptographic techniques in IPS were incorporated into the IBM cryptographic products announced in 1977.

Volume 19, Number 3, 1980

Interactive graphics today *by R. S. Burchi, p. 292.* Mapped out is the field of interactive computer graphics technology. The author surveys the range of applications from the visual arts to the visualization of theoretical mathematical models to the simulation of aircraft and ship navigation. Hardware and software are explored. Also outlined are interactive graphics data bases, data structures, and proposed standards that apply to them.

Software architecture for graphical interaction *by D. L. Weller, E. D. Carlson, G. M. Giddings, F. P. Palermo, R. Williams, and S. N. Zilles, p. 314.* Pointing at items on a graphics display is one of the most useful methods of interacting with a system graphically. This paper examines existing graphical support and lists requirements for high-level support of graphical interaction. The architecture of a prototype system with high-level support for graphical interaction is presented. This includes database support for manipulating graphical data and device-independent graphical support based on a proposed standard for graphical interaction. Algorithms are presented for identifying items selected from a display by the user. Inclusion of a database management system in graphical software support is shown to be helpful in meeting the requirements of interactive graphical application programs.

Architecture of the IBM 3277 Graphics Attachment *by D. F. McManigal and D. A. Stevenson, p. 331.* The IBM 3277 Graphics Attachment is an interactive computer graphics workstation using a dual-screen concept. A storage display monitor is attached to an IBM 3277 Display Station, the combination providing low-cost, moderate-performance interactive graphics. This paper describes the architecture of the Graphics Attachment, both functional structure and rationale. Nonarchitectural characteristics are also considered.

Experimental page makeup of text with graphics on a raster printer *by B. J. Shepherd, p. 345.* The economic advantages of printing internal-use documents on a raster printer have usually been limited to purely numeric and text documents. This paper describes an experimental character-graphic art program that demonstrates the potential of the IBM 3800 for printing a restricted set of character-graphic art documents. A special character set is outlined, as well as an algorithm that selects those characters from the set that best approximate any straight line. This character-graphic algorithm permits line art to be included in formatted text documents. There is no manual artwork or paste-up in the document output. The artwork for this paper has been reproduced from material printed by the technique discussed, although the body text has been reset from the 3800 output.

A high-resolution computer graphics system *by S. W. Handelman, p. 356.* Discussed are a graphics system and a high-resolution printer that provide scientists with a means of producing camera-ready text and graphics. This paper describes techniques for producing three types of graphics: halftone pictures, line drawings, and solid-filled areas. An overview of the software system is also presented.

An APL approach to presentation graphics *by W. H. Niehoff and A. L. Jones, p. 367.* Producing data in pictorial form is a type of computer graphics application known as presentation graphics. One approach that has been used for this type of graphics is a graphic support package using APL as the command language. Here discussed is the evolution of this approach up to its currently available forms.

A graphic interactive application monitor *by J. H. Bleher, P. G. Caspers, H. H. Henn, and K. Maerker, p. 382.* The development of interactive graphic application programs, designed for fast response, high productivity, and moderate system load, is difficult and time-consuming. Therefore, a structured approach has to be employed using function distribution in system design and application support program development.

This paper describes a comprehensive interactive graphic system that provides an environment for development and execution of graphic applications. It features an interactive graphic command language, a hierarchical structure of system, semantics, and storage, a set-oriented data concept, and library facilities.

Volume 19, Number 4, 1980

The management of software engineering, Part I: Principles of software engineering *by H. D. Mills,*

p. 414. Software engineering may be defined as the systematic design and development of software products and the management of the software process. Software engineering has as one of its primary objectives the production of programs that meet specifications, and are demonstrably accurate, produced on time, and within budget. This paper in five parts discusses the principles and practices used by the IBM Federal Systems Division for the design, development, and management of software.

The general principles of software engineering are set forth in Part I, in which the author relates software engineering to the whole field of the system development process—system engineering, hardware engineering, software engineering, and system integration. Presented briefly are overviews of the major aspects of software engineering—design, development, and management.

The management of software engineering, Part II: Software engineering program *by D. O'Neill, p. 421.* Part II, on the software engineering program, deals with the architecture of the new discipline. Discussed is the underlying concept of the software development life cycle. Based upon this foundation are a series of formally documented practices that set forth the specifics of software design, development, and management methods, which are presented in this paper. Also presented is an educational program whereby this discipline with its principles and practices has been made teachable.

The management of software engineering, Part III: Software design practices *by R. C. Linger, p. 432.* Part III, on software engineering design practices, deals with activities bounded by requirements definition on one side and program implementation on the other. Three levels of design practices are defined, dealing with construction and verification of software systems, modules within systems, and individual programs. At each stage, a new level of mathematical rigor and precision for creating and evaluating software designs is introduced.

The management of software engineering, Part IV: Software development practices *by M. Dyer, p. 451.* Part IV, on software engineering development practices, discusses a methodology for translating designs into software products. The subject is treated under two main headings, code management and integration engineering. These are rigorous methods for building the parts and integrating them into the whole software product that meets the design specifications.

The management of software engineering, Part V: Software engineering management practices *by R. E. Quinnan, p. 466.* Part V deals with the management of software engineering, which is primarily the intellectual control of the whole software engineering process. Intellectual control is brought about by a technical review strategy, a cost management approach, and a project environment for effective software development.

Application development system: The software architecture of the IBM Health Care Support/DL/I-Patient Care System *by D. J. Mishelevich and D. Van Slyke, p. 478.* Application development productivity is a broad-based concern. A system answering this concern is the IBM Health Care Support/DL/I-Patient Care System announced by IBM in late 1977. The system is of general importance because its application development system architecture is not application specific and thus can be used for the rapid development of many types of on-line systems. It has an elegant simplicity, and it uses the standard facilities of such operating system components as CICS/VS and DL/I. The application productivity has been clearly and successfully demonstrated in the real working environment of the Dallas County Hospital District (Parkland Memorial Hospital) and other sites. This paper provides an architectural overview followed by a description with an example of CRT (cathode ray tube) screen and print format design and coding and an examination of a data collection list to demonstrate the power of that facility.

A system for constructing linear programming models *by S. Katz, L. J. Risman, and M. Rodeh, p. 505.* The use of linear programming is impeded by the effort required to express a model as a matrix and to collect and handle data. An experimental interactive system called LPMODEL simplifies the development of linear programming models. It frees the user from the necessity of expressing the model as a matrix. LPMODEL provides a nonprocedural language for constructing a model in terminology that is natural to the problem, using ordinary algebraic expressions. With this language, the user can express a model concisely by generic constraints which the system interprets in conjunction with a data base to generate a concrete model for optimization.

The design of the system and its terminology and data base subsystems are discussed. An informal description is given of the modeling language which involves both ordinary arithmetic operations and symbolic operations with associated semantics. Experience with the system in agricultural modeling is described.

The Modular Application Customizing System *by R. D. Gordon, p. 521.* A system for generating application program packages for use on small computers can produce both questionnaire-tailored packages for individual users and standard packages for general distribution.

GREENPRINT: A graphic representation of structured programs *by L. A. Belady, C. J. Evangelisti, and L. R. Power, p. 542.* To improve the readability of programs over existing techniques, a new program representation termed GREENPRINT has been developed and is discussed in this paper. GREENPRINTs (the name taken from the phosphor fluorescence of certain display terminals and paralleling the term blueprints) are tree-structured diagrams together with source code statements that

represent the control structure of programs. Discussed in this paper are the diagramming conventions, control flow methodology, presentation graphics, and practical experience with GREENPRINTs.

Volume 20, Number 1, 1981

Electronic information interchange in an office environment *by M. R. DeSousa, p. 4.* This paper describes an architectural approach that provides information interchange across a broad spectrum of user applications and office automation offerings. Some of the architectures described herein are currently implemented in existing IBM products. These and other architectures will provide the basis for document interchange capability between products such as the IBM 5520 Administrative System, the IBM System/370 Distributed Office Support System (DISOSS), and the IBM Displaywriter System. Specifically described is a document distribution architecture and its associated data streams. Transforms can be utilized to interchange between these data streams and others.

A general overview of the architectures as opposed to a detailed technical description is provided. The architectures described are protocols for interchange between application processes; they do not address the specific user interface. The document distribution architectures utilize SNA for data transmission and communications control facilities.

A primer on relational data base concepts *by G. Sandberg, p. 23.* Basic concepts of relational data base management systems are described. Characteristics of the relational approach are identified and compared with present implementations of hierarchical and network data base systems. Depending on the application, a user may experience one or more of the following benefits of relational systems described in this paper: ease of understanding, increased data independence, ease of use, sound theoretical basis, and generalized data definition. Types of applications most suited to hierarchical and network data base systems are also compared and contrasted.

System R: An architectural overview *by M. W. Blasgen, M. M. Astrahan, D. D. Chamberlin, J. N. Gray, W. F. King, B. G. Lindsay, R. A. Lorie, J. W. Mehl, T. G. Price, G. R. Putzolu, M. Schkolnick, P. G. Selinger, D. R. Slutz, H. R. Strong, I. L. Traiger, B. W. Wade, and R. A. Yost, p. 41.* System R is an experimental data base management system that was designed to be unusually easy to use. System R supports a high-level relational user language called SQL, which may be used by ad hoc users at terminals or by programmers as an imbedded data sublanguage in PL/I or COBOL. This paper describes the overall architecture of the system, including the Relational Data System (RDS) and the Research Storage System (RSS).

RDS is a data base language compiler. Host language programs with imbedded SQL statements are compiled by System R, which replaces the SQL statements with calls to a machine-language access module. The compilation approach removes much of the work of parsing, name binding, and optimization from the path of a running program, enabling highly efficient support for repetitive transactions. In contrast, the RSS is a low-level DBMS, supporting simple record-at-a-time operators, but with rather sophisticated transaction management, recovery, and concurrency control.

Processor, I/O path, and DASD configuration capacity *by J. B. Major, p. 63.* This paper extends a particular capacity planning approach to include usage accounting by business element of Input/Output path and Direct Access Storage Device resources. A simple nonlinear procedure is outlined to size host configurations that can process workloads at specified rates. An important feature of the procedure is to take account of a law of diminishing returns, which is that doubling the number of components does not double the amount of work done. Discussed are configuration relationships involving TSO and DB/DC subsystem sizing, tuning, workload variability, data considerations, and hardware and software considerations. Typical, but hypothetical, examples are presented.

User-definable software applied to a real-time ambient air quality monitoring system *by P. Halpern and J. W. Rettberg, p. 86.* With increased tightening of air quality regulations, more systems for monitoring air quality became necessary. The greater use of such systems further stimulated the development of sensor-based systems that require less programming effort. The goal was to have a practical, real-time system with the ability to change the configuration of the sensors without extensively modifying the associated software. Described is a prototypical system that avoids the necessity of reprogramming every time sensors are changed.

Volume 20, Number 2, 1981

Procedures of the Human Factors Center at San Jose *by R. S. Hirsch, p. 123.* The work performed at the Human Factors Center located at IBM's development facility in San Jose, California, is representative of human factors work being done by groups of human factors specialists throughout IBM. A few of the projects that the Center was involved with are described as examples to show how human factors concerns are studied in the development of products and systems. The examples were selected to indicate the broad nature of the problems studied and include hardware and software areas. The complete scientific techniques used in the projects are not discussed in this paper so that the focus of discussion will be on the nature, scope, and methodology of the human factors work. The computing and data collection systems used for human factors tests are briefly discussed.

Effects of manual style on performance in education and machine maintenance *by J. M. Judisch,*

B. A. Rupp, and R. A. Dassinger, p. 172. Discussed is a study of human factors that was designed to measure the time to perform maintenance using two types of manuals (format) presented in two media types, for a total of four conditions. The types of manuals were the then-current Field Engineering Maintenance Manual (FEMM) and Field Engineering Theory of Operation Manual (FETOM) in both hard copy and microfiche, and a new Graphic Integrated Manual (GIM) covering the same subject matter, also in hard copy and microfiche. The objective of the study was to compare performance in solving problems on an electromechanical machine, the IBM 5424 Multi-Function Card Unit, through the use of standard and graphic integrated manuals in both hard copy and microfiche for that machine. Test results are analyzed and conclusions are presented. The general conclusion is that the new graphic integrated manuals in hard copy format lead to better performance both in education and on the job.

Natural language programming: Styles, strategies, and contrasts *by L. A. Miller, p. 184.* College students who were not familiar with computers were asked to produce written natural language procedural instructions as directions for others to follow. These directions were solutions for six file-manipulation problems that also could reasonably be solved by writing computer programs. The written texts were examined from five points of view: solution correctness, preferences of expression, contextual referencing, word usage, and formal programming languages. The results provide insight both on the manner in which people express computer-like procedures "naturally" and on what features programming languages should include if they are to be more "natural-like."

Human factors in the development of a family of plant data communication terminals *by M. Ominsky, p. 216.* Developing a set of terminals for users who had no computer experience and whose normal jobs could not be subject to interference involved human factors. Most of the design work focused on the keyboard and display interfaces of the terminals. Studies were made, alternative designs were considered, and tests were performed to ensure that the equipment was easy to use and provided acceptable speed and accuracy.

Human factors in communication *by J. C. Thomas and J. M. Carroll, p. 237.* One way of conceptualizing many of the human factors issues in interactive computing is as issues in communication about computers. Presented are a framework for this conceptualization and a review of research addressed to several levels of the communication process. Communication as an ill-structured design process is analyzed and contrasted with a process of algorithmic encoding and decoding. The design framework is then applied to examinations of how people name and refer to entities, how people understand and express relations (quantifiers and other predicates) between entities, how more complex communications (business letters) are created, and how preprinted forms reflect previous knowledge.

Volume 20, Number 3, 1981

Software simulation as a tool for usable product design *by I. A. Clark, p. 272.* A design exercise performed by human factors specialists is described. In this exercise a front-of-screen simulation of the Interactive Chart Utility was written before a working prototype was available in order to draft and test a series of on-line instructional (HELP) panels for incorporation into the final product. Trials were run in which the keyboard activity and utterances of naive subjects were recorded for later action replay, before and after redrafting the simulation. Three objective measures to detect the resulting improvement are considered, and the most robust identified.

Improving system usability for business professionals *by G. A. Helander, p. 294.* As businesses increase their emphasis on productivity, data processing departments face rising demand for computer services from people with no data processing training or background. To be effective, these services must be easy to use. This article discusses some usability considerations and how they were applied in developing an end user system. It relates experiences and observations in developing a system that is marketed in Canada under the name Interactive Extension Facilities. This system is an extension to VM/370 and CMS and was developed by the IBM Canada Limited Laboratory (Toronto) to enhance the ability of business professionals to do their work without becoming data processing specialists.

Improving the usability of programming publications *by F. J. Bethke, W. M. Dean, P. H. Kaiser, E. Ort, and F. H. Pessin, p. 306.* This paper summarizes the work of a study group on ways to improve the usability of publications that support programming products. Task orientation, an approach to providing, organizing, and packaging information, is covered, together with innovations to improve the usability of programming publications: ease-of-use education, measurement of user opinion, and incorporating usability into the publications development process.

A system for the automated office environment *by P. C. Gardner, Jr., p. 321.* A review of the history of an office system application is presented, highlighting the learning process that took place during its evolutionary development. This office system has served as the basis for a PRPQ (customized program) recently announced by IBM and known as the Professional Office System (PROFS). The general application architecture is discussed, with a specific focus on the use of virtual machines. Functional details of the various components are described, and the key distinction between office systems and office automation is addressed. The paper also discusses usage of the system and points up some of the benefits being realized by current users of a prototype system in IBM. The application function review details the

electronic document distribution capabilities of the system.

Capacity analysis of the Mass Storage System *by P. N. Misra, p. 346.* Performance of the IBM 3850 Mass Storage System (MSS) is analyzed with a view toward workload planning. Simple analytical models are discussed. The notion of staging capacity of the MSS is defined and analyzed. The main result is a set of staging capacity curves that define the processing ability of the MSS to stage and destage data to support concurrent execution of the user programs.

Volume 20, Number 4, 1981

A perspective on software science *by K. Christensen, G. P. Fitsos, and C. P. Smith, p. 372.* This paper provides an overview of a new approach to the measurement of software. The measurements are based on the count of operators and operands contained in a program. The measurement methodologies are consistent across programming language barriers. Practical significance is discussed, and areas are identified for additional research and validation.

System Productivity Facility *by P. H. Joslin, p. 388.* This paper discusses the purpose and design of a program called the System Productivity Facility (SPF). Perspective is provided by means of a brief summary of the earlier Structured Programming Facility (also termed SPF) and the requirements that led to a transformation of the earlier program into a new cross-system dialog manager. The new control facilities are explained to illustrate how the dialog manager supports a wide variety of interactive applications. Ways in which application development is simplified in the areas of data handling and display processing are explored. The purpose of the new table and file tailoring services is explained, and the error recovery philosophy is described.

Interactive user productivity *by A. J. Thadhani, p. 407.* Interactive user productivity is a measure of effective communication between man and the computer. Explored in this paper is the relationship between computer response time and user performance, and the separation of user cost from system cost. Strategies for effectively managing installations are presented and discussed.

The VM/370 Resource Limiter *by D. M. Chess and G. Waldbaum, p. 424.* The VM/370 Resource Limiter (RESLIM), a facility available on the computer systems of the IBM Thomas J. Watson Research Center, enables users, user management, and the site's Computing Center to monitor and control usage of various computing resources. If a user's consumption of a particular resource exceeds a previously established limit, RESLIM takes actions designed to improve system performance and resource availability. Possible actions include degrading the user's priority, forcing the user off the system, or simply sending a warning message to the user and/or other VM users.

Volume 21, Number 1, 1982

Strategies for information requirements determination *by G. B. Davis, p. 4.* Correct and complete information requirements are key ingredients in planning organizational information systems and in implementing information systems applications. Yet, there has been relatively little research on information requirements determination, and there are relatively few practical, well-formulated procedures for obtaining complete, correct information requirements. Methods for obtaining and documenting information requirements are proposed, but they tend to be presented as general solutions rather than alternative methods for implementing a chosen strategy of requirements determination.

This paper identifies two major levels of requirements: the organizational information requirements reflected in a planned portfolio of applications and the detailed information requirements to be implemented in a specific application. The constraints on humans as information processors are described in order to explain why "asking" users for information requirements may not yield a complete, correct set. Various strategies for obtaining information requirements are explained. Examples are given of methods that fit each strategy. A contingency approach is then presented for selecting an information requirements determination strategy. The contingency approach is explained both for defining organizational information requirements and for defining specific, detailed requirements in the development of an application.

Business Systems Planning and Business Information Control Study: A comparison *by J. A. Zachman, p. 31.* Business Systems Planning (BSP) and Business Information Control Study (BICS) are two information system planning study methodologies that specifically employ enterprise analysis techniques in the course of their analyses. Underlying the BSP and BICS analyses are the data management problems that result from systems design approaches that optimize the management of technology at the expense of managing the data. In comparing BSP and BICS, five similarities and five differences are selected for discussion, and, finally, the strengths and weaknesses of each methodology are noted. The choice between using one or the other methodology is strongly influenced by the immediate intent of the study sponsor, tempered by the limiting factors currently surrounding the BICS methodology.

Supporting Business Systems Planning studies with the DB/DC Data Dictionary *by J. G. Sakamoto and F. W. Ball, p. 54.* Traditionally, Business Systems Planning (BSP) studies have been conducted using manual techniques. This paper describes one approach for computer assistance to such a study. The Extensibility facility of the IBM DB/DC Data Dictionary is shown to satisfy the requirements for the capturing of and subsequent reporting on BSP study data. The possibility of extending this approach to follow-on software development activities is dis-

cussed. General overviews of the Business Systems Planning methodology and the IBM DB/DC Data Dictionary are also provided.

Towards an integrated development environment *by P. S. Newman, p. 81.* Problems of application-system cost, control, and effectiveness can best be addressed by highly consistent development and execution environments. This paper examines some relevant new approaches (systems description languages, new data models, application generators, and very-high-level languages), discusses the need for additional integration, and outlines a particular integration direction. This direction is intended to illustrate both the kind of consolidation needed and some of the problems involved.

Enterprise information analysis: Cost-benefit analysis and the data-managed system *by M. M. Parker, p. 108.* Enterprise information analysis studies have highlighted a gradual change in the data and information processing environment—a change in systems design and implementation from stand-alone, application-oriented systems, supporting primarily the operational and functional management levels, to data-base-oriented, data-managed systems, supporting the total organization. This shift has made many of the "traditional" financial analysis techniques used to justify a proposed system inadequate. Although a management study team that is developing an information systems proposal can choose from a variety of enterprise information analysis methodologies to assist them in the analysis of information needs, no such choice of associated (and generally accepted) disciplines or methodologies exists to support the financial justification of what has been proposed in the study team report. This paper explores the problems associated with moving from a "traditional" (data processing) financial justification of a system that is based largely on measurable costs and benefits to a financial justification of a system based largely on an assessment of intangible costs and benefits, technological change, and risk and uncertainty. A taxonomy is provided which can be used to supplement the value analysis found in the Business Systems Planning methodology. Extensive references are included as a guide to supplementary reading.

Volume 21, Number 2, 1982

Management considerations for an Information Center *by L. W. Hammond, p. 131.* This paper discusses what should be done in setting up an Information Center as part of an Information Systems group within a business organization. The Information Center is defined, including a user's viewpoint. Three key areas—the mission, organization and position, and staffing—are addressed. A procedure on how to initiate the center is presented. In general, the paper shows what an Information Center environment can be and how it might fit into a business organization.

How data flow can improve application development productivity *by W. P. Stevens, p. 162.* This paper presents the technique of data flow and how it can substantially improve application development productivity. Flows of data are the only connections needed between functional components of a computer program. Components which pass only data are so independent that they can easily be shared and reused. Such components can be developed independently, which substantially reduces the complexity of development and makes them much easier and faster to design, implement, test, and change. Building programs in this way can yield substantial increases in productivity over developing monolithic programs or even structures of called modules. The compatibility of data flow to natural human views of applications and other parts of data processing, such as distributed processing and high-performance architectures, is also presented. Recommendations are included.

SNA flow control: Architecture and implementation *by F. D. George and G. E. Young, p. 179.* To allow better network utilization, Systems Network Architecture (SNA), the IBM data communications architecture, includes flow control procedures to guard against data overrun to devices and to prevent network congestion. The measurement of "congestion" used by SNA to regulate traffic flow is performed by various SNA products. This paper describes the flow control protocols in SNA and the implementation of these protocols in the Network Control Program (ACF/NCP/VS Release 3).

Technique for assessing external design of software *by R. J. Pearsall, p. 211.* Discussed is a methodology of creating and using scenarios to assess completeness, correctness, consistency, and usability of the external design of computer software. Scenarios are paper tests of the specifications of software being designed. The approach is an outside-in, user-oriented evaluation of programs. The technique requires no machine time to perform the evaluation. As a result, defects are identified and changes are recommended early in the design phase of software development, at the time when defect removal costs are lowest.

The Document Interchange Architecture: A member of a family of architectures in the SNA environment *by T. Schick and R. F. Brockish, p. 220.* A wide variety of products for the office is now available, permitting increased automation of office procedures. To realize their full potential, these products must be able to exchange information and control requests with one another. A family of architectures has been defined to satisfy this need. This paper provides an overview of this family of architectures, including their relationship to one another. One member of this family, the Document Interchange Architecture, is described in some detail. An example illustrates use of the family of architectures in an office environment.

Volume 21, Number 3, 1982

JANUS: An interactive document formatter based on declarative tags *by D. D. Chamberlin, O. P. Bertrand, M. J. Goodfellow, J. C. King, D. R. Slutz, S. J. P. Todd, and B. W. Wade, p. 250.* This paper describes the architecture of an experimental document composition system named JANUS, which is intended to support authors of complex documents containing mixtures of text and images. The JANUS system is highly interactive, providing authors with immediate feedback and direct electronic control over page layouts, using a special two-display workstation. Authors communicate with the system by marking up their documents with high-level descriptive "tags." A tag definition language is provided whereby new tags may be defined and the format of each tagged object may be controlled.

Office-by-Example: A business language that unifies data and word processing and electronic mail *by M. M. Zloof, p. 272.* The age of the nonprogrammer user of computing systems is at hand, bringing with it the special need of persons who are professionals in their own right to have easy ways to use a computing system. Through the programming language discussed in this paper, executives and other office personnel can perform data and word processing and communications via terminals. This language, called Office-by-Example, provides rich and powerful access to the computing system computation, data base, communication, and display facilities. Discussed and illustrated by examples are a two-dimensional screen editor, triggers, and data bases, as well as word processing, electronic mail, customized menus, and application development.

The EPISTLE text-critiquing system *by G. E. Heidorn, K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow, p. 305.* The experimental EPISTLE system is intended to provide "intelligent" functions for processing business correspondence and other texts in an office environment. This paper focuses on the initial objectives of the system: critiquing written material on points of grammar and style. The overall system is described, with some details of the implementation, the user interface, and the three levels of processing, especially the syntactic parsing of sentences with a computerized English grammar.

OPAS: An office procedure automation system *by V. Y. Lum, D. M. Choy, and N. C. Shu, p. 327.* This paper discusses an experimental system being developed to support office automation. The emphasis of the paper is on a technology that allows people to automate their office and business activities. Specifically, using forms as the interface, the authors propose a powerful data manipulation and restructuring facility that not only allows users to extract and manipulate data in the forms, but can be used to interface between new and existing applications as well.

Since business and office procedures are not discrete activities, but a structured sequence of activities, a means to define and execute procedures is required. Such a means is described in this paper along with its model and an example of its application.

A case study of office workstation use *by C. V. Bullen, J. L. Bennett, and E. D. Carlson, p. 351.* This paper describes the use of the Office Analysis Methodology to study a research office environment in order to determine requirements for an advanced office workstation. The research site environment is unique in providing an opportunity to observe a natural growth pattern in the use of advanced technology. Specific workstation requirements are identified and are being implemented. Interesting observations are reported in the following areas: categories of secretarial work, use of existing workstations, influence of a community of users, access to shared services, and effects on productivity and organizational behavior.

Volume 21, Number 4, 1982

The design rationale of the System/38 user interface *by J. H. Botterill, p. 384.* This paper is a discussion of the rationale behind the design of the software user interface of the System/38. It presents the design approaches used to produce a highly usable interactive system. The three primary system user interfaces are also presented, showing how the approaches were used in their design.

How a computer should talk to people *by M. Dean, p. 424.* This essay deals with a very down-to-earth topic: the things computer programs or systems say to people—in particular, computer messages for users. It is the product of the author's experience as a programmer and technical writer and editor. It puts together a lot of common-sense insights into the philosophy of creating good computer messages, how people think and feel around computers, how to analyze the situations in which people need a message, what to say in a message and how to say it, why imagination is invaluable for creating and evaluating messages, what technical questions must be answered in order to design and build a program or system that can talk effectively to people.

Analytic queuing model for CICS capacity planning *by M. Deitch, p. 454.* In recent years there has been a growing need to develop techniques and tools for computer installation capacity planning. This paper presents both a tool, in the form of an analytic queuing model, and a methodology for performance analysis and capacity planning. Although general in its approach, the model was developed specifically for the CICS/VS environment.

Modeling distributed processing across multiple CICS/VS sites *by R. D. Acker and P. H. Seaman, p. 471.* Modeling is a useful method to aid a planner in designing the interconnection of a number of systems for distributed data processing. In the implementation in this paper, a computer-based model for sites using CICS/VS is discussed. The model permits the system

definition to be adjusted, taking into account such aspects as the number of sites. their interconnections, and workloads, so that a satisfactory configuration can be obtained.

IMS/VS: An evolving system *by J. P. Strickland, P. P. Uhrowczik, and V. L. Watts, p. 490.* The Information Management System, IMS, began in the mid-1960s as a batch-only data base system that was known then as Data Language/I (DL/I). IMS was introduced in 1969 as IMS/360, a program product for the System/360. As the System/360 evolved into System/370, including support for virtual storage, the operating system evolved into OS/VSI, OS/VS2, and then MVS. At the same time, IMS evolved to become IMS/VS. The Information Management System has continued to be adapted to new requirements, especially those of interactive, on-line operations that require data communications. Recent advances in the following categories of IMS/VS functions are discussed in this paper: Fast Path, Data Sharing, System Logging, Data Base Recovery Control, on-line changes in system environment, Intersystem Communications, MVS Common Services Area usage, and architectural restructuring.

Volume 22, Numbers 1/2, 1983

A perspective on communications and computing *by A. L. Scherr, p. 5.* Presented is an essay on the dynamics of the relationships between communications and computing. Movement of computer applications from the back office to the front office, from batch to on-line data processing, is illustrated and conclusions are drawn regarding communications protocols, network management, application and data base design, and system generality. The influence and requirements of several new technologies are presented, including those of microprocessors and teleconferencing.

X.25 and related recommendations in IBM products *by G. A. Deaton, Jr. and R. O. Hippert, Jr., p. 11.* This paper describes IBM's use of Recommendation X.25 and related recommendations of the International Telegraph and Telephone Consultative Committee. After reviewing the development history of X.25 and some of the motivations for using it, the paper gives an overview of packet-switched data networks. The reader is then given a brief technical description of Recommendation X.25 and some other recommendations used in conjunction with X.25. The architectural relationships between X.25 and IBM's Systems Network Architecture (SNA) are described for packet-switched X.25 connections between SNA and non-SNA nodes. Specific elements of Recommendation X.25 used in SNA nodes are defined. After several IBM products that support X.25 and some of the related recommendations are described, IBM's equipment for testing the X.25 interface is discussed.

Teletex—A worldwide link among office systems for electronic document exchange *by D. J. Moore, p. 30.* Teletex is a new international telecommunication service that provides direct electronic document exchange between such office text machines as electronic typewriters and word processors that are equipped with transmitting and receiving storages. Teletex is an international standard aimed at integrating office products and worldwide telecommunication. It represents a major step in the development of the office of the future. This paper traces the development of Teletex, describes its characteristics, and looks at how this service may be extended in the future.

A token-ring network for local data communications *by R. C. Dixon, N. C. Strole, and J. D. Markov, p. 47.* Technical innovations such as large-scale integrated circuit technology and distributed operating systems have respectively reduced the cost of computing and provided a basis for large networks within the confines of a single building or cluster of buildings in close proximity to one another. Local area networks can provide a systematic approach for interconnecting personal workstations, control units, and central processing units, thereby providing a means for these machines to pass information from one to the other. This paper describes a local area network based on the fundamental concepts of a token-ring. Two main ideas are presented. The first idea concerns the physical topology of the wiring network and its star-ring organization. Next, the logical data flows are overlaid on the physical network to provide control procedures for exchanging data through the network. The resulting system has unique features that produce a local area network with good performance and reliability characteristics.

Reflections on VM/Pass-Through: A facility for interactive networking *by N. Mendelsohn, M. H. Linehan, and W. J. Anzick, p. 63.* VM/Pass-Through, an interactive networking facility, has gained widespread acceptance within IBM and with IBM customers. Pass-Through allows a single terminal access to many different computers, including those at distant locations. In building Pass-Through, and in observing its growing use, we have had an opportunity to study the practical implications of this facility and of our approach to its design.

This paper is divided into two parts. The first, an introduction to Pass-Through networking, describes features of the system, supported configurations, and use of Pass-Through within the IBM Corporation. A brief history of Pass-Through's development is also provided. In the second part of the paper, Pass-Through is used to motivate a technical discussion of interactive network technology and virtual machine subsystems. Topics covered include appropriate use of the virtual machine environment, choice of routing strategy, and performance considerations. Although the introductory portions of the paper presume no prior knowledge of computer network or operating system technology, the subsequent technical discussions do depend on a basic understanding of these areas.

A Satellite Communications Controller *by J. W. Fennel, Jr. and B. D. Gobioff, p. 81.* A satellite communications controller, which is a component of satellite earth stations, establishes a time division multiple access structure that provides flexibility and efficiency in the use of satellite transmission capacity. The controller blends digital, computer, and communications technology to establish precise timings required for system synchronization. It also performs signal conversion and switching necessary to transmit voice and business machine traffic over the satellite. This paper discusses a particular satellite communications system and the architecture and implementation of the satellite communications controller.

Series/1-based videoconferencing system *by D. Anastassiou, M. K. Brown, H. C. Jones, J. L. Mitchell, W. B. Pennebaker, and K. S. Pennington, p. 97.* Discussed is a new videoconferencing system that has been developed and deployed at several IBM locations. This system transmits high-quality monochrome, freeze-frame images over dial-up telephone lines between two (or three) dedicated videoconferencing rooms. There are two main system components. An IBM Series/1 provides control, communication, data compression, and storage, and Grinnell GMR-270 image processing display system implements image acquisition, processing, and video buffering functions. Conference participants may choose either a basically black and white rendering of an image for fast transmission or a continuous-tone rendering with a longer transmission time. Details are given regarding the system configuration, function, and operation.

NIL: A high-level language for distributed systems programming *by F. N. Parr and R. E. Strom, p. 111.* Network Implementation Language (NIL) is a high-level programming language currently being used for the implementation of prototype communication systems. NIL is designed for writing executable architecture which can be compiled into efficient code for the different machines and run-time environments of a family of communicating products. NIL's distinctive features include (1) high-level primitive type families supporting constructs needed for concurrent systems, (2) facilities for decomposition of a system into modules which can be dynamically installed and interconnected, (3) compile-time *typestate checking*—a mechanism for enhancing language security without incurring large execution-time overhead.

Communications Network Management enhancements for SNA networks: An overview *by T. P. Sullivan, p. 129.* Hierarchical growth and diversification of communications networks have imposed new requirements on Communications Network Management. This essay presents the evolution of support included in IBM's Systems Network Architecture to meet these needs. Four main provisions of network management are discussed: (1) the collection and presentation of downstream network data on behalf of network resources by a Threshold Analysis and Remote Access program; (2) centralized network operator terminal access to local and remote systems by a Terminal Access Facility; (3) unsolicited alerting of the network operator by the presentation of data from network resources by a Network Problem Determination Application; and (4) the collection and presentation of data for the logical network by a Network Logical Data Manager.

An application of network management at a large computing service *by R. D. Garrigues, p. 143.* Productive operation of a large computer network serving multiple user facilities in conjunction with several host sites requires the use of network management. As treated here, network management includes an array of such methods as communication systems management, problem management, change management, and inventory management. In this paper the use of some of the management techniques is described as they apply in an education support facility network. The setup of the facility is first described; then some of the techniques for managing its operation are presented.

Volume 22, Number 3, 1983

Abstract design and program translator: New tools for software design *by J. L. Archibald, B. M. Leavenworth, and L. R. Power, p. 170.* Abstract Design And Program Translator (ADAPT) is an integrated set of tools and approaches for the design and development of software systems. Together they include a module specification language and a system design language for specifying module interfaces and interconnections. This paper explains some of their major features and illustrates their use in the design of some examples—a set of reusable software components and a generalized editor system. Benefits of the ADAPT approach are discussed, emphasizing executable design and modifiability.

The system architecture of EAS-E: An integrated programming and data base language *by D. P. Pazel, A. Malhotra, and H. M. Markowitz, p. 188.* EAS-E is an application development system based on an entity-attribute-set view of system description. It consists of a procedural language for manipulating data base and main storage entities, and direct (nonprocedural) facilities for interrogating and updating data base entities. The EAS-E software itself was implemented with the entity-attribute-set view. This paper reviews some of the EAS-E features and considers some of its implementation details. This paper is both an introduction to the EAS-E software architecture and an example of the usefulness of the entity-attribute-set view.

A simple architecture for consistent application program design *by G. R. Rogers, p. 199.* This paper addresses the architectural design aspects of general business computer application programs written in high-level procedural programming languages. It puts forth design concepts for easily built, maintainable

programs and describes a unique approach to program decomposition.

The Project Automated Librarian *by J. M. Prager, p. 214.* The Project Automated Librarian (PAL) is a tool that has been created to manage the logistical problems inherent in a medium-sized software development project. The main goals of PAL are to eliminate the problems of simultaneous updates to software modules, while allowing programmers access to the latest possible versions of the software. PAL also seeks to prevent the software from getting into an inconsistent state that could prevent users from proceeding with software development because of someone else's errors. PAL is a general-purpose tool, in the sense that it does not care what language or languages the system is being written in. It makes backups, keeps version information, and maintains documentation of changes.

Automatic generation of random self-checking test cases *by D. L. Bird and C. U. Munoz, p. 229.* A technique of automatically generating random software test cases is described. The nature of such test cases ensures that they will execute to completion, and their execution is predicted at the time of generation. Wherever possible the test cases are self-checking. At run-time their execution is compared with the predicted execution. Also described are implementations of the technique that have been used to test various IBM programs—PL/I language processors, sort/merge programs, and Graphical Data Display Manager alphanumeric and graphics support.

Full-screen testing of interactive applications *by M. E. Maurer, p. 246.* This paper describes the dialog test functions of the Interactive System Productivity Facility/Program Development Facility program product, with emphasis on the full-screen design that makes it unique. Perspective is provided by a brief summary of the test facilities available in the predecessor System Productivity Facility program product (SPF) and the requirements that led to their enhancement.

Software reliability analysis *by P. N. Misra, p. 262.* Methods proposed for software reliability prediction are reviewed. A case study is then presented of the analysis of failure data from a Space Shuttle software project to predict the number of failures likely during a mission, and the subsequent verification of these predictions.

Design and use of a program execution analyzer *by L. R. Power, p. 271.* Execution analyzers are used to improve the performance of programs, operating systems, and hardware systems. This paper presents a general overview of these tools, especially those designed for use by application programmers. The design tradeoffs of a wide variety of execution analyzers are examined. In addition, the design and use of a new execution analyzer are presented; its purpose is to assist in the optimization of highly modular PL/I programs.

Volume 22, Number 4, 1983

Advanced program-to-program communication in SNA *by J. P. Gray, P. J. Hansen, P. Homan, M. A. Lerner, and M. Pozefsky, p. 298.* Systems Network Architecture (SNA) defines the behavior of networks of heterogeneous, loosely coupled processors. This paper describes the development of program-to-program communication services in SNA and introduces Advanced Program-to-Program Communication (APPC), the culmination of this development. It also discusses the use of APPC in the construction of distributed services and shows that SNA with APPC and other SNA services can be thought of as a distributed operating system.

SNA Distribution Services *by B. C. Housel and C. J. Scopinich, p. 319.* This paper describes the IBM SNA Distribution Services (SNADS). Heretofore, SNA has focused on synchronous data distribution. Along with the advent of office systems and other distributed applications has come the requirement to provide a common architecture for interchanging data asynchronously among diverse systems and products. SNA Distribution Services provides a general asynchronous (delayed delivery) data distribution facility for SNA applications. The initial implementations are for office systems applications. Discussed are objectives for an asynchronous data distribution service, key architectural concepts, the relationship between SNADS and SNA synchronous communication architecture, and the interface between the distribution service and application transaction programs.

Interconnecting SNA networks *by J. H. Benjamin, M. L. Hess, R. A. Weingarten, and W. R. Wheeler, p. 344.* Systems Network Architecture (SNA) allows terminals and application programs to communicate with one another using SNA entities called logical units. Until now, these logical units have had to be in the same network to communicate. This paper describes recently introduced SNA network interconnection functions that allow logical units in independent SNA networks to communicate with one another. Each network is configured, defined and managed separately. By using one or more facilities called gateways, networks can remain independent while their logical units initiate, use, and terminate internetwork sessions, without any changes to themselves. A communications user need not be aware that a session partner is in a separate network.

An experimental address space isolation technique for SNA networks *by K. D. Ryder, p. 367.* The integration of computer networks has led to increasingly large and complex configurations. This integration has resulted in concern for the availability of resources for the merged networks. An experimental technique called TRAP has been developed as a way to minimize the constraints on such networks. It allows address space isolation between interconnected networks. This paper discusses the technique and its fundamental process of network address translation.

Logical problem determination for SNA networks *by R. A. Weingarten and E. E. Iacobucci, p. 387.* Problem determination on a Systems Network Architecture network has dealt mostly with error detection on physical network components. Adequate logical error-detection mechanisms associated with the logical network (software-related) errors have been only recently provided with the announcement of a new on-line interactive package called the Network Logical Data Manager (NLDM). This paper discusses the physical and logical network environments, logical network problems, and functions provided by the two releases of NLDM for logical problem determination.

Performance and availability measurement of the IBM Information Network *by R. M. Bailey and R. C. Soucy, p. 404.* A key requirement of a network service is the management of specified service levels as perceived by the end user. A capability for measuring and reporting end user response time and availability is essential. This paper describes measurement techniques to track these key service-level attributes in the IBM Information Network (IBM/IN). These techniques apply to most complex SNA networks.

SNA routing: Past, present, and possible future *by J. M. Jaffe, F. H. Moss, and R. A. Weingarten, p. 417.* This paper reviews the evolution of routing mechanisms in IBM's Systems Network Architecture (SNA) since its inception in 1974 to the present. Routing mechanisms are related to changes in the application and communications environment. Also discussed are possible evolutionary paths that may be taken in the future to address the problems of large heterogeneous networks.

Defining routing tables for SNA networks *by K. Maruyama, p. 435.* This paper addresses three basic problems associated with the definition process for the routing tables of IBM's Systems Network Architecture (SNA). The paper then introduces a program called the Routing Table Generator (RTG) and describes how these problems were solved with RTG. Also discussed are some approaches on how to use RTG in managing routing tables for growing networks.

Windows in the sky—Flow control in SNA networks with satellite links *by G. A. Grover and K. Bharath-Kumar, p. 451.* Geosynchronous communications satellites provide a unique means of high-speed computer-to-computer transmission of large volumes of data over long distances. The physical distances involved in transmitting to and from the satellites cause relatively long propagation delays for messages. In order that the high bandwidth be used effectively, large quantities of data have to be transmitted before pausing for an acknowledgment. This condition creates a potential for some new and unique types of traffic jams. This paper discusses these situations in the context of Systems Network Architecture (SNA) networks. In particular, the issues related to SNA's flow control and traffic management facilities in the presence of satellite links are discussed, along with potential solutions to ensure efficient network operation.

Volume 23, Number 1, 1984

Architecture prototyping in the software engineering environment *by W. E. Beregi, p. 4.* This technical essay presents a perspective on the evolution and problems of the software development craft and how software engineering techniques show promise to solve these problems. It introduces architecture prototyping as a program development technique for improving software quality. Experience with large software systems shows that over half of the defects found after product release are traceable to errors in early product design. Furthermore, more than half the software life-cycle costs involve detecting and correcting design flaws. In this paper, we explore a disciplined approach to software development based on the use of formal specification techniques to express software requirements and system design. As a consequence, we can use techniques like rapid prototyping, static design analysis, design simulation, and dynamic behavior analysis to validate system design concepts prior to element design and implementation. We explore how these techniques might be organized in a software architecture prototyping facility that would be similar to the Computer-Aided Design and Manufacturing (CADAM) tools used in other engineering disciplines. We also examine the process by which software engineers might use these facilities to create more reliable systems.

Factors affecting programmer productivity during application development *by A. J. Thadhani, p. 19.* The effects of good computer services on programmer and project productivity during application program development are examined. Programmer's terminal activity and the nature of terminal work are analyzed. The discussion includes the effects of short response times, programmers' skills, and program complexity on productivity.

A comparative study of system response time on program developer productivity *by G. N. Lambert, p. 36.* Skilled programmer time and computer time and resources are valuable. Earlier studies had shown that added computer resources can decrease system response time and increase programmer productivity significantly. A controlled study has been made to determine whether that finding is true for the particular conditions in another program development organization. That study is reported here. Programmer productivity increased sixty-two percent with subsecond system response time. A new finding is that individual group project offices lead to greater efficiency than large open rooms.

Analysis of free-storage algorithms—revisited *by G. Bozman, W. Buco, T. P. Daly, and W. H. Tetzlaff, p. 44.* Most research in free-storage management has centered around strategies that search a linked list

and strategies that partition storage into predetermined sizes. Such algorithms are analyzed in terms of CPU efficiency and storage efficiency. The subject of this study is the free-storage management in the Virtual Machine/System Product (VM/SP) system control program. As a part of this study, simulations were done of established, and proposed, dynamic storage algorithms for the VM/SP operating system. Empirical evidence is given that simplifying statistical assumptions about the distribution of interarrival times and holding times has high predictive ability. Algorithms such as first-fit, modified first-fit, and best-fit are found to be CPU-inefficient. Buddy systems are found to be very fast but suffer from a high degree of internal fragmentation. A form of extended subpooling is shown to be as fast as buddy systems with improved storage efficiency. This algorithm was implemented for VM/SP, and then measured. Results for this algorithm are given for several production VM/SP systems.

Speech filing—An office system for principals *by J. D. Gould and S. J. Boies, p. 65.* Business people spend most of their time communicating, or attempting to communicate, with others. We briefly describe our ideas about these communication activities and their resulting problems, and then discuss an experimental tool we developed to help business people solve some of their communication problems. This tool, called the Speech Filing System, allows users to send messages to anybody in the world. The system offers powerful editing, filing, retrieval, and message distribution and control functions, using pushbutton telephones as the terminals.

Playback: A method for evaluating the usability of software and its documentation *by A. S. Neal and R. M. Simons, p. 82.* Human factors evaluations of software products and accompanying user publications must be conducted so that developers can be certain that the target user population can learn to use the product with a minimum of difficulty and be able to perform the intended tasks efficiently. A methodology is described for obtaining objective measures of product usability by collecting performance data on the user interface without affecting the user or the system being evaluated. The log of stored activity is later played back through the host system for analysis.

Volume 23, Number 2, 1984

An overview of three relational data base products *by S. Kahn, p. 100.* This issue of the *IBM Systems Journal* focuses on aspects of three recently announced IBM relational data base products. They are IBM Database 2, Query Management Facility, and Data Extract. This essay illustrates the requirements that these products were designed to address and gives a brief overview of their content and history. Its objective is to provide an introduction to the more specific and detailed papers that follow.

IBM Database 2 overview *by D. J. Haderle and R. D. Jackson, p. 112.* IBM Database 2 (DB2) is a data base management system that supports the relational model of data. This paper presents the major features of DB2 and discusses its architecture and the relationship of DB2 with the host operating system. These principles are illustrated by an example.

The Query Management Facility *by J. J. Sordi, p. 126.* Data from a relational data base can be displayed in reports, changed, and otherwise controlled using a program called Query Management Facility (QMF). An overview of this program is presented and is followed by a discussion comparing equivalent forms of various queries expressed in two distinctly different languages. Both languages are designed for use with relational data and are supported by QMF.

TSO Attach: A multipurpose communication channel to IBM Database 2 *by K. R. Hammond and M. R. Zimowski, p. 151.* TSO Attach provides IBM Database 2 capabilities in a productive work environment that appears as a natural extension of the Time Sharing Option (TSO) and the Interactive System Productivity Facility (ISPF). It was designed and built with careful consideration for the varied and complex user group for which it was intended. Ease of use and ease of development and maintenance were among the significant factors in the design. These factors and others are addressed in this paper, which discusses the basic design decisions made in building the TSO Attachment Facility.

IBM Database 2 in an Information Management System environment *by J. R. Dash and R. N. Ojala, p. 165.* Over the years, the IBM Information Management System (IMS/VS) has been developed to meet expanding user needs. During that time, a parallel development has taken place. The relational data model grew from Codd's original theory to a practical data base prototype. Now a new data base management system, IBM Database 2 (DB2), has been built on the relational model. This paper discusses the implementation and design considerations for the integration of IMS and DB2 from the user's viewpoint. It also presents the attachment facilities from a design perspective.

Data recovery in IBM Database 2 *by R. A. Crus, p. 178.* This paper presents the various forms of data recovery provided by IBM Database 2 (DB2). It describes the DB2 recovery log, introduces the notion of a unit of recovery, and discusses the two-phase commit protocol used by DB2. Furthermore, it describes what type of information is logged, the DB2 checkpoint process, what a compensation log record is, and how DB2 handles undo/redo processing, media recovery, restart after abnormal system termination, and data unavailability.

IBM Database 2 performance: Design, implementation, and tuning *by J. M. Cheng, C. R. Loosley, A. Shibamiya, and P. S. Worthington, p. 189.* The larger and more complex a relational data base, the more efficient the data base management system must

be to maintain an acceptable level of performance. The design and implementation of IBM Database 2 (DB2) have been aimed toward this objective. Techniques for achieving this key objective in DB2 are the subject of this paper. Presented are performance-related strategies in query processing and performance-related design tradeoffs. Data base and application design options and their resolution for optimum performance are also discussed. Also presented are techniques to maintain performance by application monitoring and tuning and DB2 system tuning.

Managing IBM Database 2 buffers to maximize performance *by J. Z. Teng and R. A. Gumaer, p. 211.* The relational data base system, IBM Database 2 (DB2), has a component that manages data buffering. This paper describes the design considerations of the Buffer Manager and the tradeoffs involved in managing the allocation of DB2 buffers to maximize performance.

Volume 23, Number 3, 1984

Ease of use: A system design challenge *by L. M. Branscomb and J. C. Thomas, p. 224.* While it is becoming increasingly obvious that the fundamental architecture of a system has a profound influence on the quality of its human factors, the vast majority of human factors studies concern the surface of hardware (keyboards, screens) or the very surface of the software (command names, menu formats). In this paper, we discuss human factors and system architecture. We offer best-guess guidelines for what a system should be like and how it should be developed. In addition, we suggest ways in which advances in research and education could result in systems with better human factors. This paper is based on an address by L. M. Branscomb and a publication by the authors in the Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19–23, 1983.

Directions in cooperative processing between workstations and hosts *by B. C. Goldstein, A. R. Heller, F. H. Moss, and I. Wladawsky-Berger, p. 236.* Advancements in technology have provided us with the availability of high-performance processors from the high end of computing to the personal computer. In addition, technology growth has enabled us to envision sixteen megabytes of real storage for a personal computer.

As a result, we have witnessed not only a tremendous growth at the high end of the computing spectrum, but also the development of sophisticated personal computers (e.g., the IBM PC XT/370) with real storage capacities approaching those of high-end computers of a decade ago.

This growth at both ends of the computing spectrum has given us a choice. We can either allow a clean separation to grow between personal computer and host or provide a means by which they cooperate in providing quality service to the user without the complexity normally associated with high-end systems. This paper explores what such a cooperation could mean.

System/370 capability in a desktop computer *by F. T. Kozuh, D. L. Livingston, and T. C. Spillman, p. 245.* A desktop computer with System/370 capability was produced by enhancing the IBM Personal Computer XT with additional hardware and developing software that provides a compatible interface. The computer, the IBM Personal Computer XT/370, and this software allow users to run most System/370 Conversational Monitor System application programs unaltered in a desktop environment. The evolution of the development and details of the function of the hardware and software are described.

A tight coupling of workstations *by D. M. Chess, p. 255.* This paper addresses the problem of situations in which people at physically distant locations must have access to essentially the same computing environment at the same time. That is, each user must be able to provide input to whatever application or system is active, and must be provided with all relevant output. Common examples of this situation are demonstrations, presentations, education, and troubleshooting.

A prototype system has been developed to study ways of solving this problem in the microcomputer workstation environment. The prototype allows users at two IBM Personal Computers to share access to the computing environment through the keyboard and the display screen by tightly coupling the computers.

Architecture implications in the design of microprocessors *by R. E. Matick and D. T. Ling, p. 264.* This paper examines how architecture, the definition of the instruction set and other facilities that are available to the user, can influence the implementation of a very large scale integration (VLSI) microsystem. The instruction set affects the system implementation in a number of direct ways. The instruction formats determine the complexity of instruction decoding. The addressing modes available determine not only the hardware needed (multiported register files or three-operand adders), but also the complexity of the overall machine pipeline as greater variability is introduced in the time it takes to obtain an operand. Naturally, the actual operations specified by the instructions determine the hardware needed by the execution unit. In a less direct way, the architecture also determines the memory bandwidth required. A few key parameters are introduced that characterize the architecture and can be simply obtained from a typical workload. These parameters are used to analyze the memory bandwidth required and indicate whether the system is CPU- or memory-limited at a given design point. The implications of caches and virtual memories are also briefly considered.

Use of images in commercial and office systems *by P. J. Somerville, p. 281.* This paper examines some of the simpler processing techniques that may usefully

be performed on bi-level (two-tone) images by a competent commercial applications programmer if a few basic (albeit complex internally) tools are provided. Such processes include storage, indexing, changing resolution, rotating, trimming, and then display and printing. These processes can provide the facilities for an enterprise to incorporate images and image data into its office systems and into its main line-of-business data processing applications.

Security considerations for personal computers *by W. H. Murray, p. 297.* The wide use of personal computers and general access to telecommunications links have intensified the need for computer security. Security practices as discussed in this paper relate to protecting an organization's personal computers as physical property, protecting the organization's data and applications, and protecting the organization itself. These matters are discussed from the point of view of protection from the improper use of personal computers.

Volume 23, Number 4, 1984

An overview of computer security *by R. C. Summers, p. 309.* Presented is an overview of computer security, including concepts, techniques, and measures relating to the protection of computing systems and the information they maintain against deliberate or accidental threats. Motivations for security measures are discussed. Security strategies are considered. Actions and events that threaten security are described, along with technical problems that can prevent the computer from adequately dealing with threats. Security models are surveyed. Specific technical and administrative measures for promoting security are described. Among the technical measures discussed are design of secure systems, hardware and operating systems, identification of users, encryption, and access control packages. Administrative measures include personnel, physical security of the computing system, and auditing. Also presented is the establishment of a security program. Reviewed are special problems and their solutions, including communications and networks, data base management systems, and statistical data bases. This paper is based on a paper by the author published in *The Handbook of Computers and Computing*, edited by Arthur H. Seidman and Ivan Flores, Van Nostrand Reinhold Company, Inc., New York (1984).

The design of the REXX language *by M. F. Cowlishaw, p. 326.* One way of classifying computer languages is by two classes: languages needing skilled programmers, and personal languages used by an expanding population of general users. REstructured eXtended eXecutor (REXX) is a flexible personal language designed with particular attention to feedback from its users. It has proved to be effective and easy to use, yet it is sufficiently general and powerful to fulfill the needs of many demanding professional applications. REXX is system and hardware independent, so that it has been possible to integrate it experimentally into several operating systems. Here REXX is used for such purposes as command and macro programming, prototyping, education, and personal programming. This paper introduces REXX and describes the basic design principles that were followed in developing it.

An application analyzer *by R. Ambrosetti, T. A. Ciriani, and R. Pennacchi, p. 336.* An interactive tool, aimed at supporting the application user/analyst in specifying and analyzing a business area, is presented. The features of the tool, named the Application Analyzer/Experimental, are described both in their theoretical foundations and their actual implementation. A brief description of the architecture of the tool and its internal structure is given. A review of the main concepts of the application development area is also included. The follow-on of the prototype described here is the program offering known as System A.

Performance issues in local-area networks *by W. Bux, p. 351.* This paper discusses several important performance problems in the design of local-area networks. The questions discussed relate to various aspects of architecture, design, and implementation: (1) the delay-throughput characteristics of the medium access protocols, (2) the performance of local-area networks on which a file server provides file storage and retrieval services to intelligent workstations, and (3) timing problems in local-area network adapters. Since the paper does not primarily address the performance analyst, it is descriptive in nature; analytic details are omitted in favor of a more intuitive explanation of the relevant effects.

VM/370, Attached Processor, and multiprocessor performance study *by W. H. Tetzlaff and W. M. Buco, p. 375.* This paper discusses performance studies of Attached Processors, multiprocessors, and VM/370. A methodology for evaluating performance is discussed. Performance improvements are explained and evaluated. These studies played a role in a new option to the VM/System Product control program that is called the High Performance Option (HPO).

Volume 24, Number 1, 1985

Standardized graphics on the IBM Personal Computer *by T. B. Clarkson III, p. 3.* Although acknowledged to be an effective means of communicating information, graphics has not progressed more rapidly in the burgeoning use of personal computers due to the lack of standards for both writing and running graphics applications. A graphics standard—the Virtual Device Interface (VDI)—has been proposed for national use and is in the process of being adopted. An implementation of the VDI is currently available for the IBM Personal Computer. This paper briefly traces the history of graphics as used with personal computers, explores the difficulties that standardization efforts have met, explains the VDI model, and shows how this model operates in the IBM Personal Computer environment to make graphics a natural extension of the operating system.

A professional graphics controller *by K. A. Duke and W. A. Wall, p. 14.* The IBM Professional Graphics Controller and Display were developed to meet the needs of engineers and scientists for an improved graphics capability in the Personal Computer environment. These units provide graphics systems with improved function, resolution, and color range, and at the same time they allow existing productivity software to be executed in an emulation mode. This paper describes the function and discusses the design of the Professional Graphics Controller.

Expanded personal computing power and capability *by P. A. Korn, J. P. McAdaragh, and C. L. Tondo, p. 26.* Discussed is the XENIX™ Operating System for the IBM Personal Computer AT. The operating system incorporates capabilities of a mainframe operating system—multiusage, multitasking, file management and security, program compilation, and networking. The XENIX shell structure is introduced. Pipes and pipelining are presented. The XENIX file structure is explained and illustrated with examples. Software development and text formatting are treated in detail. The ability to compile C program code developed under XENIX and run it on the IBM Personal Computer Disk Operating System is explained.

The C programming language and a C compiler *by R. R. Ryan and H. Spiller, p. 37.* In the last few years, the C programming language has become one of the most widely used languages for applications and systems software on microcomputer systems. This paper describes the C language and its history. It then presents a specific implementation of C, the Microsoft C Compiler, which runs on the IBM Personal Computer.

Design considerations for IBM Personal Computer Professional FORTRAN, an optimizing compiler *by M. L. Roberts and P. D. Griffiths, p. 49.* An optimizing FORTRAN compiler with power to handle large applications at execution speeds comparable to those of large computers has been implemented on the IBM Personal Computer. This implementation is described, with emphasis on the design decisions that were considered in the development of the compiler.

An APL system for the IBM Personal Computer *by M. L. Tavera, M. Alfonseca, and J. Rojas, p. 61.* This paper discusses the design and building of an APL interpreter for the IBM Personal Computer. Discussed is the writing of the interpreter itself, which required the use of an intermediate language designed by the authors. This machine-independent language also made possible the development of APL interpreters for two other systems—System/370 and Series/1. The particularizing of the interpreter required a compiler, which in the case of the Personal Computer produced Intel 8088 and 8087 assembly language code. The matching of the APL interpreter to the operating system (DOS) required an APL supervisor, which is also discussed in this paper. The provision of the APL character set presented problems, the solutions of which are also presented. Other topics discussed are the display, the keyboard, and the session manager.

Volume 24, Number 2, 1985

The IBM large-systems software development process: Objectives and direction *by W. S. Humphrey, p. 76.* This paper introduces a special issue of the *IBM Systems Journal* on the IBM large-systems software development process. The issue provides an overview of the subject and a summary of the key principles of the IBM software quality and productivity efforts in large-scale systems programming. The major topics addressed in this issue are the software development process, software development tools and methodologies, quality and productivity measurements, and programmer education.

A programming process architecture *by R. A. Radice, N. K. Roth, A. C. O'Hara, Jr., and W. A. Ciarfella, p. 79.* The Programming Process Architecture is a framework describing required activities for an operational process that can be used to develop system or application software. The architecture includes process management tasks, mechanisms for analysis and development of the process, and product quality reviews during the various stages of the development cycle. It requires explicit entry criteria, validation, and exit criteria for each task in the process, which combined form the "essence" of the architecture. The architecture describes requirements for a process needing no new invention, but rather using the best proven methodologies, techniques, and tools available today. This paper describes the Programming Process Architecture and its use, emphasizing the reasons for its development.

A programming process study *by R. A. Radice, J. T. Harding, P. E. Munnis, and R. W. Phillips, p. 91.* A programming Site Study group was convened to look at the work of eight large-system programming development locations within IBM and to evaluate them according to a set of process stages. Eleven attributes were applied to each process stage. The process of the Site Studies is directly transferable to software evaluations on any project in the software industry, and it is believed that the studies are the first step necessary in the evolution of a consistently repeatable and dynamically controllable process of improvement within the industry. The phases of these studies and implementation of the studies are described.

Automating the software development process *by G. F. Hoffnagle and W. E. Beregi, p. 102.* Demand for reliable software systems is stressing software production capability, and automation is seen as a practical approach to increasing productivity and quality. Discussed in this paper are an approach and an architecture for automating the software development process. The concepts are developed from the viewpoint of the needs of the software development

process, rather than that of established tools or technology. We discuss why automation of software development must be accomplished by evolutionary means. We define the architecture of a software engineering support facility to support long-term process experimentation, evolution, and automation. Such a facility would provide flexibility, tool portability, tool and process integration, and process automation for a wide range of methodologies and tools. We present the architectural concepts for such a facility and examine ways in which it can be used to foster software automation.

Quality emphasis at IBM's Software Engineering Institute *by M. B. Carpenter and H. K. Hallman, p. 121.* Improvements in quality and productivity in the development of programs can be obtained by instructing the programming development groups in the use of modern software engineering methodology. To provide this instruction for its employees, IBM has established a Software Engineering Institute. Currently training in the methodology is being offered through an education program of the Institute known as the Software Engineering Workshop. This paper describes the role of the Institute, its background and offerings, and some results obtained.

PDM: A requirements methodology for software system enhancements *by R. G. Mays, L. S. Orzech, W. A. Ciarfella, and R. W. Phillips, p. 134.* Traditional requirements processes often do not address the many problems encountered in the development of software products. Conventional processes begin with the structural definition of the proposed system, under the assumption that the raw requirements are understood. How this understanding is developed is not formally addressed. The IBM software development process requires a methodology to develop the rationale of the requirement, both in terms of its underlying problem and its business justification, prior to the development of the functional specification. Conventional requirements processes address a single software application intended for use by a uniform set of end users. The resulting system is usually a one-time replacement of some existing system. Many IBM software products, however, address requirements received from a large, diverse set of customers who use the products in a wide array of computing environments. Product releases are typically developed as incremental enhancements to an existing base product. This paper describes the Planning and Design Methodology (PDM), a requirements planning process that supports the collection, analysis, documentation, and tracking of software requirements. The process includes requirements collection, definition of the underlying problems, development of an external functional description that addresses the problems, and development of system and product designs from the external functional descriptions. PDM has been applied in three development areas with positive results.

A process-integrated approach to defect prevention *by C. L. Jones, p. 150.* Recent efforts to improve quality in software have concentrated on defect detection. This paper presents a programming process methodology for using causal analysis and feedback as a means for achieving quality improvements and ultimately defect prevention. The methodology emphasizes effective utilization of all error data to prevent the recurrence of defects.

Programming process productivity measurement system for System/370 *by M. J. Flaherty, p. 168.* Discussed in this paper are the underlying principles of a programmer productivity measuring system. The key measures (or metrics) are people and lines of code. Definitions of these metrics are refined and qualified, according to the conditions under which they are used. Presented also is a data base design for retaining and retrieving these metrics under a wide variety of applications and other circumstances. Depending on definitions, applications, and other circumstances, productivity measurements may differ widely. On the other hand, after suitable productivity metrics have been defined, consistency of application of the same metrics yields comparable results from project to project.

Volume 24, Numbers 3/4, 1985

Worldwide systems engineering *by T. G. Peck, p. 182.* IBM systems engineering celebrates its 25th anniversary in 1985. This paper provides a perspective of the part systems engineering has played in the success of IBM in the information processing business during that 25-year period. The history of systems engineering is briefly reviewed, and the similarities and differences in worldwide systems engineer functions are examined. The relationships among marketing, systems engineering, and customers are discussed. Also discussed are career paths for systems engineers. Expectations and challenges for systems engineering in the future are explored.

HONE: The IBM marketing support system *by W. Boos, p. 189.* The storage, retrieval, and dissemination of data pertaining to a large, complex product line is made possible by the Hands-On Network Environment (HONE) discussed in this paper. HONE provides on-line interactive support to marketing, systems, and administrative personnel, and, most recently, to customers. The evolution of HONE is presented. Discussed in detail are new HONE distributed processing capabilities now enabled under an advanced network architecture. In that environment, the processing power and data bases of HONE and other host systems will be interconnected and support the speed and processing autonomy of IBM Personal Computers as workstations.

Performance considerations for a distributed data processing system designed for high availability *by S. Agassi, p. 200.* The high-availability requirements of computerized systems that are needed to meet the objectives of the organization are being acknowledged more and more by the data processing community. The paper presents the planning process

for a distributed data processing system designed to meet high availability requirements. This process was performed as a systems engineering activity in order to assess the feasibility of the presented approach, which was proposed to a customer.

Information System Model and Architecture Generator *by K. P. Hein, p. 213.* The advent of integrated, shared-data systems has made it increasingly necessary to address the application development process from the architectural and manufacturing perspective rather than from a build-as-you-go job shop viewpoint. Although the Business Systems Planning (BSP) methodology provides an enterprise-wide strategic Information Systems plan, it is still at an abstraction level that leaves the traditional gap between "requirements" and implementations untouched. The Information System Model and Architecture Generator (ISMOD) tool complements and enhances BSP by mechanizing the planning process, thus providing a facility to narrow this gap by allowing orderly and consistent top-to-bottom architectural decomposition of the enterprise environment. It is an enterprise planning vehicle and not an implementation system, but it is the first critical component to support an integrated systems architecture effort. It automates and, to a large extent, formalizes a laborious requirements documentation process preceding code development, and it does this "top to bottom," from a global, enterprise-wide, information requirements viewpoint. This paper discusses the overall architectural concepts of integrated data systems development, the place of ISMOD within it, and the specific facilities, techniques, and information provided by the system.

A single-system interface using the IBM 3270-PC *by M. M. Ghiotti, p. 236.* Many businesses use a variety of terminal types connected to central host computers. Presented here is a rationale and the experience gained with a single terminal type—the IBM 3270-PC—interconnected with hosts via the Application Program Interface to achieve enhanced user efficiency.

Strategies for problem prevention *by J. Newton, p. 248.* A philosophy of preventing problems from occurring in a data processing installation rather than reacting to problems is becoming increasingly necessary. The institution of comprehensive and formally managed testing strategies is an important step in this direction. Such strategies are discussed, and it is shown that they also support disaster backup/recovery plans.

Customer Information Control System—An evolving system facility *by B. M. Yelavich, p. 264.* Presented is an overview of the present CICS architecture. Discussed is the evolution of that original design as a transaction management system that accommodates data base management, operating systems, and input and output devices as well as hardware of increasing numbers and complexity. User needs past and present are analyzed with a view toward understanding how CICS might evolve in the future.

An approach to high availability in high-transaction-rate systems *by R. C. Brooks, p. 279.* In business enterprises, it is important that high availability be maintained in the computer systems used by the enterprises, particularly in systems that have high transaction rates. A way of maintaining high availability is discussed, including the implementation that should be undertaken and the design issues involved. Some additional steps for further improvements are also offered.

The System Planning Grid: A model for building integrated information systems *by B. R. Buckelew, p. 294.* Information systems have evolved as a result of technological advances and the increasing demand for information. Over the past few years, systems that developed separately are being forced to merge. This paper describes a model for building a set of integrated architectural guidelines to ensure that a "system" is being built. The use of the System Planning Grid as a model for setting product standards and organization responsibilities will also be discussed.

An information technology architecture for change *by M. W. Mudie and D. J. Schafer, p. 307.* This paper defines a technology architecture for information processing in large corporations. It describes a matrix of processing environments consisting of three processing types: production, decision support, and office; three processing locations: centralized, departmental, and workstation; and a methodology for implementing applications in those environments. Key to the architecture is a supporting framework comprising the communications network, a data service function, an office services function, enabling software, and support organizations. This approach is designed to provide an integrated information system to support organizations whose business environment is changing, and where flexibility, responsiveness to change, and cost effectiveness are vital. The approach is representative of methods used by systems engineers in assisting customers to decide on a system configuration that best suits their needs.

Cache-DASD storage design for improving system performance *by C. P. Grossman, p. 316.* This paper discusses three examples of a cache-DASD storage design. Precursors and developments leading up to the IBM 3880 Storage Control Subsystems are presented. The development of storage hierarchies is discussed, and the role of cache control units in the storage hierarchy is reviewed. Design and implementations are presented. Other topics discussed are cache management, performance of the subsystem, and experience using the subsystem. It is shown that a cache as a high-speed intermediary between the processor and DASD is a major and effective step toward matching processor speed and DASD speed.

Volume 25, Number 1, 1986

The IBM 3090 system: An overview *by S. G. Tucker, p. 4.* The first part of this paper places the IBM 3090 system in historical perspective with respect to its predecessors. Treated briefly are the technology and the design process, both of which were critical to the development of the 3090. Presented in detail is the 3090 system itself, with emphasis on its features that differ from those of prior systems.

IBM 3090 performance: A balanced system approach *by Y. Singh, G. M. King, and J. W. Anderson, p. 20.* The IBM 3090 system represents the highest level of system performance offered by IBM to date. To realize the full performance potential of this system, it is essential to maintain a balance among its various components. The major components of the system are the processor(s), storage, I/O, and the software that manages the system resources. Their performance attributes are discussed and their effect on system performance illustrated by laboratory benchmark measurements for the MVS and VM operating systems.

Engineering and scientific processing on the IBM 3090 *by D. H. Gibson, D. W. Rain, and H. F. Walsh, p. 36.* The IBM 3090 processor implementation of the System/370 Vector Architecture represents a major new system design for engineering and scientific processing, featuring both scalar and vector capability in a uniprocessor and in a dyadic and four-way parallel processing environment. The history of large-scale scientific processing is reviewed, leading to a statement of current requirements. The design objectives for scalar, parallel, and vector capabilities are identified, followed by a summary of the resulting 3090 features. Selected highlights of the vector hardware are given, followed by a summary of the supporting software. The paper concludes with a discussion of performance, beginning with the identification of suitable applications. An example is given of one application utilizing each of the three capabilities: scalar, parallel, and vector. Several of the most important performance parameters are identified.

The IBM System/370 vector architecture *by W. Buchholz, p. 51.* Discussed is the instruction-set architecture of the IBM System/370 vector facility, a compatible extension of the System/370 architecture. Both the base system, which is a general-purpose System/370 processor, and the optional vector facility employ a register type of organization. Data formats are the same, arithmetic operations produce exactly the same results, arithmetic exceptions are handled in the same way, and instructions are precisely interruptible for page faults and other causes in the same manner as those of the base system. This approach permits substantially increased performance on vectorizable programs with only a modest increase in hardware and software, while retaining the ability to run existing nonvector programs unchanged.

Vector system performance of the IBM 3090 *by R. S. Clark and T. L. Wilson, p. 63.* Performance of the Vector Facility of the IBM 3090 processor is discussed. The paper has two parts, the first presenting factors affecting performance measurement of the Vector Facility and the criteria for its design. In the second part, use of the 3090 storage hierarchy to support the vector processing implementation is the main aspect of the discussion.

The System Usability Process for Network Management Products *by K. D. Gottschalk, p. 83.* This paper presents an overview of a process for system usability. The process is a systematic series of activities and procedures designed to improve the usability of software network management products. The elements of the process are given and future directions for evaluating usability described.

Network management software usability test design and implementation *by L. C. Percival and S. K. Johnson, p. 92.* The approach used at one of IBM's development sites for usability testing is somewhat different from methods used elsewhere. The approach was developed specifically for testing of software communications products as one aspect of the System Usability Process. The test design and implementation are described.

Improving availability of software subsystems through on-line error detection *by L. Koved and G. Waldbaum, p. 105.* A VM/370 program called Auditor detects faults in the operation of computer software subsystems and attempts to restore service as quickly as possible. Through a series of periodic tests, Auditor diagnoses whether these subsystems are operating properly. When faults are detected, service restoration procedures are automatically called, and the persons responsible for the subsystems are notified. The various types of faults are recorded for subsequent analysis.

Workstations and mainframe computers working together *by J. K. Kravitz, D. Lieber, F. H. Robbins, and J. M. Palermo, p. 116.* The history and design philosophy of a research project that produced a prototype software package are described. The package provides cooperation between large-scale mainframe computing systems and personal desktop workstations. Also discussed are the features of PC/VM Bond, the product that grew out of the research project.

Volume 25, Number 2, 1986

Introduction to IBM's knowledge-systems products *by A. J. Symonds, p. 134.* The industrialization of artificial intelligence is believed by many to be a technology that will contribute to a new generation of "smart" computer systems. Technical managers who are users or suppliers of computer systems are trying to understand how the technology can help them, and they are finding it an elusive subject to grasp. This introductory paper starts with a technol-

ogy overview that aims to address this need for understanding and provide a suitable background for the papers that follow.

Knowledge-based systems in the commercial environment *by E. D. Hodil, C. W. Butler, and G. L. Richardson, p. 147.* Knowledge-based systems are among the first applications of artificial intelligence to make the crossover from the laboratory to the real-world commercial environment. Typically, artificial intelligence systems have been implemented in the LISP programming language on specialized hardware. The experimental nature of early systems has allowed many of them the luxury of having little or no interface to existing hardware, software, or data. In this paper, arguments are presented to demonstrate the feasibility of implementing knowledge-based systems using traditional hardware and software. Also, an architecture is proposed for knowledge-based shell systems that is compatible with the software development environment of large commercial information systems organizations. To demonstrate these concepts, an example system is shown.

YES/MVS and the automation of operations for large computer complexes *by K. R. Milliken, A. V. Cruise, R. L. Ennis, A. J. Finkel, J. L. Hellerstein, D. J. Loeb, D. A. Klein, M. J. Masullo, H. M. Van Woerkom, and N. B. Waite, p. 159.* The Yorktown Expert System/MVS Manager (known as YES/MVS) is an experimental expert system that assists with the operation of a large computer complex. The first version of YES/MVS (called YES/MVS I) was used regularly in the computing center of IBM's Thomas J. Watson Research Center for most of a year. Based on the experience gained in developing and using YES/MVS I, a second version (YES/MVS II) is being developed for further experimentation. This paper discusses characteristics of the domain of large computing system operation that have been illuminated by the YES/MVS I experience, and it describes the modifications in the design of YES/MVS II that are an outgrowth of the YES/MVS I experience.

The genesis of a knowledge-based expert system *by J. A. Voelker and G. B. Ratica, p. 181.* This paper discusses the genesis, development, and testing of a knowledge-based expert system called the Contract Support Services Consultant. This system aids in the process of estimating, bidding, and preparing agreements for certain services required for the rearrangement, relocation, discontinuance, and reinstallation of IBM equipment. The goals in developing the prototype of the Contract Support Services Consultant were to capture the knowledge and experience of experts and make them available to nonexperts in a convenient, consistent way. The result is a system that improves the accuracy, consistency, and timeliness of the estimates and bids.

Prolog for applications programming *by W. G. Wilson, p. 190.* This paper discusses the problems and benefits of using the Prolog language to write application programs. Much attention is currently focused on expert-systems shells, which play a role in artificial intelligence (AI) systems similar to that of application generators in more conventional applications. The Prolog programming language embodies many of the features found in these shells, while providing a relatively general and complete programming language. MVS performance tuning is used as an application that typifies a broad class of applications suitable for implementation in this language. Some of the difficulties that had to be overcome to use the language are presented, with their solutions.

The numeric representation of knowledge and logic—Two artificial intelligence applications in medical education *by W. D. Hagamen and M. Gardy, p. 207.* MEDCAT (medical diagnosis, consultation, and teaching) is a program that makes diagnoses from empiric data stored in patient records, explains its reasoning in response to questions (consultant mode), and uses its logical and communicative skills to instruct medical students in the proper approach to medical diagnosis (student mode). MEDCAT's reasoning can be modified by free-format discussion with physicians. CATS (computerized anatomical teaching system) is an entirely separate program designed to teach gross anatomy. Like MEDCAT, it has a consultant mode that the student may use to explore the program's reasoning, and a student mode in which the program takes the initiative. A prominent feature of CATS is its ability to discover meaningful general principles that reduce the need for memorization. Despite important differences in the subject matter, the data structure and code are very similar in the two programs. Both use a powerful natural-language interface that parses the input and generates the output.

The Portable Inference Engine: Fitting significant expertise into small systems *by N. A. Burns, T. J. Ashford, C. T. Iwaskiw, R. P. Starbird, and R. L. Flagg, p. 236.* The Portable Inference Engine (PIE) is the nucleus of an expert system that allows the segmentation of rules in order to utilize large knowledge bases in limited memory. If the expert-systems rules can be divided into segments of highly related rules with little interaction among those segments, such knowledge-base segments can then be paged in and out of memory on demand. PIE gathers information by querying the user and executing external procedures in order to conclude goals.

Computer processing of dates outside the twentieth century *by B. G. Ohms, p. 244.* This paper presents practical solutions to problems envisioned in extending computer processing of dates beyond the twentieth century. Many data processing managers are concerned with processing cross-century dates, and in doing so using existing systems, with a minimum of disruption to normal operations. The use of existing date formats can eliminate the need for massive system modifications. Methods of using existing date formats across century boundaries are explained. The use of a format termed the Lilian date format in honor of Luigi Lilio, the inventor of the

Gregorian calendar, is introduced. The requirements for an effective date-processing algorithm are presented.

Volume 25, Numbers 3/4, 1986

Systems architecture in transition—An overview *by H. Lorin, p. 256.* Systems architecture refers to the distribution of function and control among elements of a system. It is primarily a structural concept that includes the original meaning of the word architecture in the form of "processor architecture." This paper undertakes to describe topics of current interest in the evolution of computing structures. It discusses various unit structures that may emerge as the economics and capabilities of technology relax more and more constraints. Of particular interest is the internal structure of a central computing complex, the relation of computing elements and I/O elements, and the maturity of the I/O elements. The paper also suggests that the structures found within a single computing unit may be realized across larger elements more widely dispersed. Hardware and software issues are addressed.

Impact of memory systems on computer architecture and system organization *by R. E. Matick, p. 274.* The largest part of computer architecture, in both the central processing unit and the overall system, has been and continues to be directly influenced in one way or another by the types of memory systems available. This is readily apparent in certain areas such as I/O architecture and memory hierarchies. However, the pervasiveness of this influence throughout the entire system is not so obvious. This paper demonstrates this relationship and shows how it has affected computer architecture over the years. Two approaches are used, the first being a direct look at how specific architectures attempt to circumvent the limitations of the associated memory system. This includes such topics as the internal architecture of CPUs: memory hierarchies and virtual memory, I/O architecture, file structuring, and data base architecture. Second, a *gedanken* (thought) experiment is used to predict future trends. It is assumed that very large-scale integration will evolve to the point at which we can have nearly any main memory system we desire, with some reasonable constraints. The architectural changes that might take place will be seen to be precisely related to the weaknesses in current memory systems which various architectures currently attempt to circumvent.

Computing as a tool for human augmentation *by W. J. Doherty and W. G. Pope, p. 306.* The IBM Thomas J. Watson Research Center in Yorktown, New York, has experienced a factor of twenty times increase in the past ten years in the amount of time its people spend using computers interactively in their work. This is twice the penetration rate of television in the 1950s. A similar degree of penetration is expected to happen in the rest of industry in the next ten years. That will mean a major departure from traditional data processing, with computers being used as tools to augment the users' abilities in all phases of their work. Examples of human augmentation, as seen in the work of several researchers, are offered in this paper. The integration of large numbers of personal workstations into this environment has given us new understanding of how to work. The causes, impediments, and consequences of these changes are described, with emphasis on human requirements for bandwidth, response time, tools, online storage, and computing capacity.

IBM small-system architecture and design—Past, present, and future *by G. G. Henry, p. 321.* Small computer systems have become widespread and important parts of the computer industry. In this paper, a selection of IBM's small-system architectures and design approaches are reviewed. Then current architectures are discussed, with an emphasis on the RT Personal Computer. A brief presentation of trends in small systems is provided.

Software engineering: An emerging discipline *by R. Goldberg, p. 334.* Software engineering is an emerging discipline whose goal is to produce reliable software products in a cost-effective manner. This discipline is evolving rapidly as the challenges faced by its practitioners keep extending their skills. This paper gives a quick tour of the main ideas and thrusts that have driven software engineering in its first 25 years and attempts to look ahead at the next set of advances.

Tools for building advanced user interfaces *by J. L. Bennett, p. 354.* System developers are noticing that their design decisions strongly affect computer usability. The design of the user interface has an important bearing on the knowledge users must have to accomplish work through the user-computer interface. Recognition of this fact is leading to the development of User Interface Management Systems (UIMSs). A UIMS is a design concept for separating the details of user interaction from the details of advanced applications. This paper shows how UIMS research and research into the representation of user process knowledge (i.e., user how-to-do-it skills) can help developers understand issues involving ease of learning and ease of use. This parallel progress in UIMS development and in user modeling makes it easier to build high-quality advanced user interfaces.

Open Systems Interconnection *by J. R. Aschenbrenner, p. 369.* The subject of Open Systems Interconnection (OSI) standardization is becoming increasingly important to the telecommunications and information processing communities. A number of OSI standards have been completed, others are near completion, and initial product offerings by vendors have begun. This paper briefly defines what OSI is, the interrelationships of the various standards bodies, and the goals and benefits to users, vendors, country post telephone and telegraph bodies, common carriers, and governments. The IBM view of OSI and how it relates to Systems Network Architecture is also discussed.

An advanced voice/data telephone switching system *by J. M. Kasson, p. 380.* This paper describes a voice/data circuit-switching system known as the ROLM CBX II. The paper first discusses what the ROLM CBX II family does, describing the most important functions and relating them to popular voice and data applications. The second section describes how the CBX II works, delineating the architecture and giving some details of the system implementation. The final section offers an assessment of what the CBX II and similar products might become in the future.

The evolution of printers and displays *by A. F. Mayadas, R. C. Durbeck, W. D. Hinsberg, and J. M. McCrossin, p. 399.* Printer and display technologies have undergone remarkable changes since the beginning of the computer era. In this paper we trace the evolution of these two types of I/O devices, from the middle of the 1940s to the present, and show how computer system evolution has influenced the designs and technologies of I/O devices.

Volume 26, Number 1, 1987

Structures for networks of systems *by A. L. Scherr, p. 4.* This paper describes how systems will be interconnected in the future, the roles that they will play, and the trade-offs that affect these roles. Starting with a general model for structuring a network of systems, general trade-offs in cost and performance are discussed relative to where functions are placed in the network. Several general principles for data and function placement in a network of systems are derived from these trade-offs. The optimal roles for each of several layers of a network of systems are discussed. Finally, conclusions are drawn regarding the design of future networks of systems.

SNA: Current requirements and direction *by R. J. Sundstrom, J. B. Staton III, G. D. Schultz, M. L. Hess, G. A. Deaton, Jr., L. J. Cole, and R. M. Amy, p. 13.* Since its announcement in 1974, Systems Network Architecture (SNA) has evolved in terms of its functional content, configurational flexibility, and network management services. This paper briefly traces this progress to the present and examines the more recent advances in greater detail. It then discusses known requirements for enhanced application and transaction services, for additional provisions for very large networks, for continuing adaptation of small-system and transmission media advances, for inclusion of additional management capabilities, and for further integration of network standards—all of which will shape future SNA developments.

Prospects and design choices for integrated private networks *by P. E. Green, Jr. and D. N. Godard, p. 37.* This paper reviews some of the choices that will be available in the next few years, as the much-discussed move toward implementing voice and data integration within a single wide-area integrated private network proceeds. After the term *wide-area integrated private network* has been defined, a discussion is given of requirements the network ought to satisfy for its users. Then two particularly promising approaches, fast packet switching (FPS) and hybrid switching (HS), are defined, and specimen design points for FPS and HS are postulated, so that the two can be compared. While a definitive comparison would require systematic cost and performance studies, much insight can be gained from the qualitative comparison that we present here. We assess some of the arguments that have been put forward in favor of FPS or HS and conclude that, although today both architectures have promise, and research on both should continue, FPS appears to be slightly simpler to implement and operate.

Robotics *by J. U. Korein and J. Ish-Shalom, p. 55.* This paper is a survey that is intended to give the reader an introduction to some issues and problems in the field of robotics today. The first section discusses industrial applications of robotics and the requirements they engender. A substantial section is included on robot programming, including programming languages, motion programming, and techniques. This is followed by a section on trajectory planning. Issues in both robot-level trajectory planning and task-level trajectory planning are discussed. The section on control is divided into three parts: controller objectives, the system model, and controller types. Very brief discussions of actuators, sensing, and end effectors are also included.

Database technology *by P. G. Selinger, p. 96.* Computers were originally invented and used to ease and automate the task of computation. As the word "computer" implies, these early machines were used for calculations, such as tabulating census data. As a side effect, the technology needed for storing data was also invented to provide the computational engine with input data and allow it to output results. This means of permanently storing data included punched cards, tape, and disks. Throughout the 1950s and most of the 1960s, the management of stored data was done as required; file systems stored data according to user-defined formats and kept a table of contents. Users shared data by equally *ad hoc* means, generally by taking turns accessing the same device. Over the years, database technology has evolved through at least three generations to a diverse and sophisticated set of data management tools, as discussed in this paper. This paper has three major sections. Presented first is an introduction to database technology. Presented next is a description of the evolution of database technology from early computing to the sophisticated systems of today. The third section presents a view of both the driving forces that will influence the database technology of the future and also the resulting new directions for the future.

A perspective on the 801/Reduced Instruction Set Computer *by M. E. Hopkins, p. 107.* From the earliest days of computers until the early 1970s, the trend in computer architecture was toward increasing complexity. This complexity revealed itself through the

introduction of new instructions that matched the application areas. Microcode was an implementation technique that greatly facilitated this trend; thus, most computers were implemented using microcode. In 1975, work began at the Thomas J. Watson Research Center on an experimental minicomputer. This project, termed the 801 project, questioned the trend toward complexity in computer architecture. It was observed that most of the complex instructions were seldom used. Thus, a computer could be designed with only simple instructions without drastically increasing the path length or number of instructions required to implement an application. This made it possible to implement a machine without resorting to microcode, which improved performance. This paper described the background and evolution of these ideas in the context of the 801 experimental minicomputer project.

Data communications: The implications of communication systems for protocol design *by B. C. Goldstein and J. M. Jaffe, p. 122.* The construction of a communication network architecture, specifying protocols by which systems communicate, is a complex art. Much has been written about the optimal protocols for theoretical models of systems. This paper points out that protocol design must depend on the "nuts and bolts" of the systems which implement the protocols. Numerous examples are provided to support this thesis. The paper also briefly discusses other issues that influence protocol design and draws lessons for standards activities.

A large-scale computer conferencing system *by D. M. Chess and M. F. Cowlishaw, p. 138.* This paper discusses the relationships between computer-mediated communications and other forms of communication and describes a particular computer conferencing system in use within IBM. The system described is quite large, with over three thousand contributors and over twenty thousand readers. We discuss the structure of the system, the actions that users can take, and the ways in which the system is being used. Neither the definitions presented nor the system described are intended to be the last, or only, word on the subject; as computer-mediated communications and distribution become more and more important in the business and professional communities, we will need more ways of thinking about communication systems and about information distribution in general.

Volume 26, Number 2, 1987

OSI-SNA interconnections *by K. K. Sy, M. O. Shiobara, M. Yamaguchi, Y. Kobayashi, S. Shukuya, and T. Tomatsu, p. 157.* As Open Systems Interconnection (OSI) becomes an international standard, it is gaining support in both industry and government agencies. One of the major applications of OSI is to act as an intermediary between heterogeneous networks. This paper discusses a scheme for interconnecting a Systems Network Architecture (SNA) network with OSI. This scheme is based on a joint study between IBM Japan and Nippon Telegraph and Telephone Corporation conducted during 1984. Fundamental relationships between OSI session and transport layers and SNA Logical Unit type 6.2 are explored. An OSI-SNA gateway structure is examined, and data units, address translation, and exception handling are discussed.

Visual interpretation of complex data *by E. J. Farrell, p. 174.* With increasingly complex digital simulations and computations, larger volumes of output are generated, and users must select a concise method of displaying the output and extracting relevant information. A set of imaging functions and display modes is developed to interpret data effectively for a wide range of applications. The imaging functions are complementary. Each function is useful for a different aspect of data interpretation. The relationships between variables and the global structures within the data are obtained with different display modes such as multiple windows and animation. With this set of complementary imaging functions and display modes, more information is obtained than with prior imaging methods. Also, more complex simulation studies are feasible since the results can now be visualized.

An incidence-matrix-driven panel system for the IBM PC *by P. Halpern, S. M. Roberts, and L. Lopez, p. 201.* A set of programs called TRYLON is discussed which permits the application developer to design and create a set of intelligent full-screen panels. These panels serve as a user interface for application programs. The panels and the linkages between them are uncoupled from the application code, thereby reducing programming effort and development time. An incidence matrix is used to describe the graph of a network composed of the panels. Because the paths among the panels are specified by entries in the incidence matrix at panel creation time, there is no need to program the logical constructs for the network.

A page-swapping prototype for VM/HPO *by W. H. Tetzlaff, T. Beretvas, W. M. Buco, J. Greenberg, D. R. Patterson, and G. A. Spivak, p. 215.* This paper discusses a series of changes that were made to a system running the Virtual Machine/System Product with the High Performance Option to enhance paging. The motivation and the background for these enhancements are discussed, and the design of a series of experimental paging subsystems is described and contrasted with the old design: specifically, the new algorithms for main memory management, block paging, working set identification, trimming, prepaging, page replacement, page-out device selection, and page-out slot selection. The performance impact of these changes is illustrated by results of benchmark measurements, which are then contrasted to measurements without the enhancements. Some things learned in running the prototype are discussed and conclusions drawn.

Volume 26, Number 3, 1987

Message-handling systems based on the CCITT X.400 recommendations *by T. E. Schütt, J. B. Staton III, and W. F. Racke, p. 235.* Message-handling systems allow the exchange of electronic mail between computers. The International Telegraph and Telephone Consultative Committee (CCITT) has proposed a standard for message-handling systems in the form of the X.400 series of recommendations that has been widely recognized by computer manufacturers and communications carriers. This paper provides a tutorial on the X.400 recommendations and then describes two prototypes developed by the IBM European Networking Center in Heidelberg, Germany, in cooperation with its research partners. The prototypes were demonstrated together with X.400 prototypes from other manufacturers at the CeBIT 86 trade fair in Hannover, Germany.

Specification and implementation of an ISO session layer *by A. Fleischmann, S. T. Chin, and W. Effelsberg, p. 255.* This paper describes a novel technique for the specification and implementation of layered communication software. The technique is called Parallel Activity Specification Scheme (PASS) and is based on an extended-state machine model of protocol automata. It allows a convenient description of the communication behavior of concurrent systems and semiautomatic generation of programming language code from the specification. The first large-scale experience gained with this technique was in the specification and implementation of an ISO session layer. The code generation process and the embedding of the session code into a portable OSI operating system environment are described in detail.

A framework for information systems architecture *by J. A. Zachman, p. 276.* With increasing size and complexity of the implementations of information systems, it is necessary to use some logical construct (or architecture) for defining and controlling the interfaces and the integration of all of the components of the system. This paper defines information systems architecture by creating a descriptive framework from disciplines quite independent of information systems, then by analogy specifies information systems architecture based upon the neutral, objective framework. Also, some preliminary conclusions about the implications of the resultant descriptive framework are drawn. The discussion is limited to architecture and does not include a strategic planning methodology.

The structure of System/88, a fault-tolerant computer *by E. S. Harrison and E. J. Schmitt, p. 293.* In recent years, there has been a growing requirement for continuous processing capability approaching 24 hours per day, 7 days per week. Industries such as finance, transportation, securities, and telecommunications have continuous-availability requirements that can approach downtimes of not more than three minutes per year. This paper describes configurations of the Stratus/32 continuous processing computer system that are marketed as the IBM System/88 through an agreement with Stratus Computer, Inc. The system achieves its fault tolerance via hardware duplexing coupled with a distributed operating system that allows system resources to be distributed over many separate computers while maintaining a single systems image to the end user. This single systems image may also be extended across a network of multiple systems. The way in which software makes this distribution possible and the way in which system resources are named to allow transparent distribution across the system are described in the paper. Also described are the transaction processing services that are part of the operating system and allow transaction programs to be written to operate effectively over the distributed system, by means of a requester-server structured approach.

Volume 26, Number 4, 1987

Advanced Interactive Executive (AIX) operating system overview *by L. K. Loucks and C. H. Sauer, p. 326.* The Advanced Interactive Executive (AIX) is the operating system used in the RT Personal Computer. It is a portable operating system architecture that is suitable for a wide range of computer architectures and customer requirements. Discussed in this paper are the structure and services of AIX.

The IBM RT PC ROMP processor and memory management unit architecture *by R. O. Simpson and P. D. Hester, p. 346.* The ROMP processor is the microprocessor used in the IBM RT PC. It is a 32-bit processor with an associated memory management unit implemented on two chips. ROMP is derived from the pioneering RISC project, the 801 Minicomputer at IBM Research. This paper describes some of the trade-offs which were made to turn the research project into a product. It gives an introduction to the architecture of ROMP, including the addressing model supported by ROMP's memory management unit. Some of the unique features of the programming model are explained, with high-level language coding examples which show how they can be exploited. ROMP's architecture is extensible, and the fact that almost all programming for the RT PC has been in high-level languages means that the RT PC hardware architecture can be extended as needed to meet future requirements while preserving the investment in existing software.

Advanced Interactive Executive program development environment *by R. Q. Cordell II, M. Misra, and R. F. Wolfe, p. 361.* The IBM RT Personal Computer uses the Advanced Interactive Executive as an operating system. This operating system provides a distinct environment for the development of programs. Some of the characteristics of application development with this operating system, some of its features that influence application design, and the basic program development tools are described.

AIX usability enhancements and human factors *by F. C. H. Waters, R. G. Bias, and P. L. Smith-Kerker, p. 383.* As microcomputers become capable of running increasingly large and complex operating systems, the question of the usability of those operating systems becomes critical. Most microcomputer users neither are nor want to be systems programmers, yet most of the existing large operating systems assume the existence of a dedicated systems programming organization to install and maintain system software. This paper describes the process by which a large existing operating system was modified to allow it to be installed, configured, maintained, and used by individuals with minimal programming knowledge. We describe the aspects that had to be changed, the kinds of modifications that were required, the reasoning behind those modifications, and the priorities that constrained our activity. We also describe the development process by which potential usability problems were identified and corrections were defined, implemented, and validated.

Box structured information systems *by H. D. Mills, R. C. Linger, and A. R. Hevner, p. 395.* The box structure methodology for information systems development is based on a usage hierarchy of data abstractions, in which each abstraction is defined in three distinct forms, called its *black box*, its *state machine*, and its *clear box*. Each of these three box structures defines identical external behavior, but with increasing internal visibility, to provide a hierarchical structure which supports the systems development principles of referential transparency, transaction closure, state migration, and common services. This hierarchy of box structures provides, in turn, a basis for orderly management of information systems development by a finite set of analysis and design tasks in a spiral development process. The methodology and its use are described.

A perspective on Advanced Peer-to-Peer Networking *by P. E. Green, Jr., R. J. Chappuis, J. D. Fisher, P. S. Frosch, and C. E. Wood, p. 414.* This paper is intended to familiarize the reader with the many reasons for undertaking the design and implementation of peer networking on small and intermediate business machines such as the IBM System/36 family. Such networking function was recently announced as IBM's Advanced Peer-to-Peer Networking (APPN) on Release 5 of the System/36. This paper sets the stage for a companion paper in this same issue, which discusses the implementation experience and details of the System/36 APPN product. In the present paper, the history of System/36 communication is first reviewed, and it is shown how APPN was a natural evolution from earlier function. Then an extensive study of user requirements that was started in 1982 is summarized. The paper concludes with a brief technical tutorial on the structure of the APPN design.

Implementing System/36 Advanced Peer-to-Peer Networking *by R. A. Sultan, P. Kermani, G. A. Grover, T. P. Barzilai, and A. E. Baratz, p. 429.* System/36 Advanced Peer-to-Peer Networking (APPN) provides highly dynamic, fully distributed peer networking for low-end processors. It is built upon existing SNA Logical Unit 6.2 and Node type 2.1 support. APPN presents System/36 users with a simplified model of communications. The structure of the APPN subsystem is outlined, with particular emphasis on the integration of APPN functions with existing SNA support. The authors describe how particular aspects of the APPN design have been tuned to the System/36 operating environment.

Volume 27, Number 1, 1988

SNA network management directions *by D. B. Rose and J. E. Munn, p. 3.* Network management is the process of monitoring and controlling the components of a communication-oriented network of information systems in the areas of configuration management, operational control, problem management, change management, and performance and accounting management. This paper discusses the evolution of the SNA network management architecture and products that implement that architecture, and describes their likely future direction.

Utilizing the SNA Alert in the management of multivendor networks *by R. E. Moore, p. 15.* Managing multivendor networks is one of the largest challenges facing vendors and customers in data processing and telecommunications. This paper focuses on one aspect of managing multivendor network environments: problem notification, isolation, and resolution, via Systems Network Architecture's Alert. It describes an extension to the SNA Alert function, termed the generic Alert, that makes it possible for various vendors' products, as well as customer-written applications, to send Alerts of the same type to a single Alert receiver. It also describes IBM's implementation of the Alert receiver for the System/370, the NetView™ program product. Among the facilities that the generic Alert architecture provides to an Alert sender are the following: (1) code points that index short descriptions of Alert conditions, probable causes of these conditions, and recommended operator actions; and (2) vehicles to carry product-unique text. This text can be used for further characterizing an Alert condition or specifying a particular operator action.

NetView/PC *by M. Ahmadi, J. H. Chou, and G. Gafka, p. 32.* NetView/PC™ is an IBM program offering that provides the first implementation of a Systems Network Architecture (SNA) service point on an IBM Personal Computer. It allows integration of non-SNA devices such as computerized branch exchanges or Token Ring local-area networks into a central SNA network management facility by forwarding Alert information from these devices to a host-based network management product such as NetView™. It also provides some focal point services, including logging and display of Alerts and problem creation and tracking. Non-IBM products

may use NetView/PC services via a vendor application programming interface (API).

An integrated network management product *by D. Kanyuh, p. 45.* The NetView™ program is the cornerstone of IBM's network management concept. It allows enterprises to consolidate their network operations at a centralized point and is a key element in providing the function to perform the major disciplines of network management. This paper describes how and why the NetView program originated, the components that make up the product, the network management functions provided by these components, such as operations, problem management, and performance management, and the contribution of the NetView program to the open network management direction.

An architecture for a business and information system *by B. A. Devlin and P. T. Murphy, p. 60.* The transaction-processing environment in which companies maintain their operational databases was the original target for computerization and is now well understood. On the other hand, access to company information on a large scale by an end user for reporting and data analysis is relatively new. Within IBM, the computerization of informational systems is progressing, driven by business needs and by the availability of improved tools for accessing the company data. It is now apparent that an architecture is needed to draw together the various strands of informational system activity within the company. IBM Europe, Middle East, and Africa (E/ME/A) has adopted an architecture called the E/ME/A Business Information System (EBIS) architecture as the strategic direction for informational systems. EBIS proposes an integrated warehouse of company data based firmly in the relational database environment. End-user access to this warehouse is simplified by a consistent set of tools provided by an end-user interface and supported by a business data directory that describes the information available in user terms. This paper describes the background and components of the architecture of EBIS.

Volume 27, Number 2, 1988

The design of Operating System/2 *by M. S. Kogan and F. L. Rawson III, p. 90.* The design of Operating System/2™ (OS/2™) is a result of matching the requirements of IBM and its customers for a new operating system for various models of the Personal System/2® with the need for continuity with a very large body of established DOS applications. The design of OS/2 represented a significant challenge both in meeting these requirements and in making efficient use of the hardware. In this paper, the design characteristics of OS/2 are discussed.

OS/2 EE Database Manager overview and technical highlights *by P. Y. Chang and W. W. Myre, p. 105.* Structured Query Language (SQL) has become an industry standard. It is supported by mainframe products. This paper describes the OS/2 EE Database Manager, which is based on the relational database model of E. F. Codd and on the SQL query language. A functional overview of the OS/2 EE Database Manager and OS/2 EE is provided; technology applied to different areas is highlighted.

OS/2 Query Manager overview and prompted interface *by S. L. Watson, p. 119.* Operating System/2™ (OS/2™) Query Manager provides a user interface for both novice and sophisticated database users of the OS/2 Database Services. It offers defaults and standard options for the novice user. Prompting provides access to the database without requiring extensive knowledge of Structured Query Language (SQL), yet it also allows the advanced user to completely customize screens and reports. Direct keying of SQL statements is allowed as a fast path for the knowledgeable SQL user. Functions of OS/2 Query Manager are described, including details of the user interface.

Writing an Operating System/2 application *by R. L. Cook, F. L. Rawson III, J. A. Tunkel, and R. L. Williams, p. 134.* This paper illustrates use of the key facilities of Operating System/2™ (OS/2™). It provides some guidance on how to use the interfaces and functions implemented by the system and then introduces the program development environment. Two examples demonstrate the use of some of the more interesting capabilities. The paper discusses many of the significant differences between the functions of OS/2 and those of the Disk Operating System (DOS).

COBOL/2: The next generation in applications programming *by R. Sales, p. 158.* IBM COBOL/2 is a new compiler and debugger system for the Personal System/2® product range developed by Micro Fo cus Group PLC of the United Kingdom. In this paper, Robert Sales, a software development manager for Micro Focus who was instrumental in creating the COBOL/2 system, describes how COBOL/2 breaks new ground in providing support for many disparate COBOL language dialects and standards, as well as in providing support for OS/2™ on the Personal Computer architecture.

Understanding device drivers in Operating System/2 *by A. M. Mizell, p. 170.* To meet its design goals for multitasking, Operating System/2™ requires a device driver architecture for interrupt-driven device management. A device driver in OS/2™ is affected by the new architecture both in its structure and in its relationship to the system. An OS/2 device driver contains components, such as the Strategy Routine and Hardware Interrupt Handler, which have well-defined responsibilities. The basic form of these components is a FAR CALL/FAR RETURN model. The operating system calls the device driver components to handle certain types of events, such as an application I/O request or a device interrupt. In responding to these events, an OS/2 device driver must cooperate with the operating system to preserve system responsiveness by helping to manage the multi-

tasking of concurrent activities. Since OS/2 uses both the real mode and the protected mode of the system processor to support DOS and OS/2 applications, respectively, the components of an OS/2 device driver must execute in both modes. In this manner, an OS/2 device driver can be viewed as an installable extension of the Operating System/2 kernel. Comparisons between IBM Personal Computer DOS and Operating System/2 are drawn to illustrate differences between device management and device driver architecture.

VGA—Design choices for a new video subsystem *by S. Thompson, p. 185.* The VGA (Video Graphics Array) video subsystem is provided as standard on the system boards of the IBM Personal System/2® Models 50 and above. VGA was designed to meet the objectives set for these new systems and to support compatibility with older IBM offerings, while at the same time providing greater performance and increased function. The IBM Enhanced Graphics Adapter (EGA) was chosen as the compatibility base for VGA, since EGA had become the video standard for IBM-compatible computer systems. Six new modes of operation were designed to meet the needs of new business and consumer applications and to improve the ergonomics of the systems. Higher-performance video presents several design problems, including electromagnetic interference, physical design size, and cost. These design problems were contained by implementing the VGA function in a single-gate array and by using an analog display interface. The use of a video digital-to-analog converter (DAC) allows the VGA subsystem to show any color from a choice of 256K colors when a color display is used, or 64 gray shades when a monochrome display is used. The VGA subsystem was designed to provide a uniform interface for color and monochrome that allows a color mode to be selected when a monochrome display is used, or a monochrome mode to be used on a color display. A color-summing algorithm was designed and implemented in the BIOS (Basic Input/Output System) software that will allow colors to be shown as shades of gray on the monochrome display.

The Realtime Interface Co-Processor Multiport/2 adapter *by S. C. G. Sykes, p. 198.* The Realtime Interface Co-Processor Multiport/2 is a programmable, multifunction adapter that extends the processing capabilities of the Personal System/2® and provides a solution to applications with unique communications requirements. Customized for speed and flexibility, the Multiport/2 is fully programmable and supports asynchronous, byte-synchronous, and bit-synchronous protocols on its eight communications ports. This powerful single-slot computer can handle functions that previously required processing by the PS/2®. Microcode on the Multiport/2 provides a real-time multitasking base on which custom applications can be built. This paper describes the Multiport/2, its microcode, system support software, and development tools.

An introduction to typographic fonts and digital font resources *by A. W. Griffee and C. A. Casey, p. 206.* Type has evolved from blocks of wood or metal bearing the raised character shape to the many and varied digitized representations of the character that are available through computer system technology. Typography is the art or technique of composing printed material from type. The evolution of digital type into the computers of today has opened the door of typography to people who have had little or no previous knowledge of the subject. It has also introduced a higher level of complexity to document composition and presentation service software than was previously required. Discussed in this paper are the art of composing printed material, the selection of an appropriate type design for a given application, the information required to create and manage a digital font resource, and the computer system's use of digital font resources to produce typographic-quality documents. These matters are examined in a way that introduces the reader to typographic fonts, the additional complexities involved, and the need for consistency in the definition and application of digital font resources.

Advanced Function Printing: A tutorial *by R. K. deBry and B. G. Platte, p. 219.* Advanced Function Printing (AFP) is an IBM product for printing mixed text, image, and graphics in a system-printing environment. Described is the AFP printing model. We demonstrate the way in which this model is used for existing printing applications, enhanced line printing, and full advanced-function printing.

Architectures of Advanced Function Printing *by R. K. deBry, B. G. Platte, C. L. Berinato, and J. W. Marlin, p. 234.* Discussed is the use of the capabilities of all-points-addressable laser (page) printers in applications involving pages composed of text, image, and vector data in a device-independent way. Also presented is the ability to describe and print complex documents composed of multiples of such pages. Provision is made for the migration of current line-printer applications to print using these new page printers. Three architectures are described that—along with an Advanced Function Printing (AFP) model—support these capabilities. Each of these architectures is described in the context of the current implementation of the Advanced Function Printing software.

Volume 27, Number 3, 1988

Introduction to Systems Application Architecture *by E. F. Wheeler and A. G. Ganek, p. 250.* 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.

Common Communications Support in Systems Application Architecture *by V. Ahuja, p. 264.* Application execution in a Systems Application Architecture (SAA) network depends on the underlying capability of the network to obtain reliable connectivity and orderly data exchange among its system components. The objectives of SAA are distributed applications, distributed processing, and distributed data, which are achieved through interconnected SAA systems supporting appropriate interfaces and architectures. The Common Communications Support of SAA affords this capability by utilizing a number of Systems Network Architecture communication architectures and international standards. These architectures provide useful data interchange within SAA components by providing services ranging from managing data links to specifying data streams for user applications. This paper discusses the role of Common Communications Support and the means for SAA users to access this support, and provides an overview of the functions and roles of various component architectures of Common Communications Support, along with their interrelationships.

Common User Access—A consistent and usable human-computer interface for the SAA environments *by R. E. Berry, p. 281.* Systems Application Architecture (SAA) will allow customers to apply their investments in their computer operations across IBM's three major computing environments that exhibit unique characteristics in terms of architecture, workstations, operating systems, and system services. User experience is one of these investments. The Common User Access (CUA) establishes a degree of standardization that is compatible with the differences in the three environments and that supports transfer of users' experiences. CUA is based on a user-interface architecture that identifies fundamental elements of structure. The intent is to provide a transfer of users' conceptual-level learning across different and evolving technologies. CUA specifies user-interface components and guidelines to be used by application designers, and it provides a basis for programming development tool specifications.

Application enabling in SAA *by D. E. Wolford, p. 301.* The Common Programming Interface (CPI), one of the four key elements of Systems Application Architecture, comprises a growing set of programming languages and services. The CPI indirectly offers end-user access through the Common User Access by providing the application developer with the necessary interfaces. The CPI addresses the application development requirement for portability of applications and programmer skills. As the CPI continues to expand, it addresses the requirements for access to host data through intelligent workstations and for transparent access to remote data and applications.

Enabling the user interface *by S. Uhlir, p. 306.* Presenting a consistent interface to the user is one of the objectives of Systems Application Architecture (SAA). The development of SAA applications is simplified by providing enabling interfaces which help an application developer support the SAA user interface. Rather than providing a single-level enabling interface, SAA offers a spectrum of levels spread over two interfaces: the SAA Presentation Interface and the SAA Dialog Interface. This gives the application developer the freedom to choose the appropriate level of interface for the application.

Integrating applications with SAA *by L. A. Buchwald, R. W. Davison, and W. P. Stevens, p. 315.* Advances in computing technology and reductions in development cost have greatly increased the number of people who use computers, and have expanded the number and types of applications available to them. People want their applications to share data and to be consistent with one another with respect to terminology and appearance. They also frequently need access to applications and data on computers in other locations; the computers may be models and types that these persons do not normally use. Integrating application functions in a seamless environment is an important step toward satisfying some of these requirements. This paper discusses what integrated applications are, why they are valuable, and how Systems Application Architecture (SAA) can make it easier to develop them.

Designing SAA applications and user interfaces *by W. P. Dunfee, J. D. McGehe, R. C. Rauf, and K. O. Shipp, p. 325.* This paper describes a framework for developing applications that conform to Systems Application Architecture (SAA). The paper shows a high-level approach to creating a design; it gives examples of early modeling work with the user interface; and it appraises SAA through the eyes of several system designers. The usability of user interfaces has been evaluated through the modeling of office tasks. That experience is described, showing the influence of the SAA Common User Access (CUA) on the model and the influence of the model on CUA. Discussed is a design for distributed applications that fit within the SAA framework and the influence of SAA on the design of integrated distributed applications.

Distributed files for SAA *by R. A. Demers, p. 348.* Files are still a major way of storing data in computer systems, and they are a significant part of the information to be handled by the distributed processing networks that are developing. Systems Application Architecture is supporting distributed files. In this paper, the goals, benefits, and problems of providing this support are discussed, along with the role of Distributed Data Management architecture.

Distributed database for SAA *by R. Reinsch, p. 362.* This paper describes, in general terms, distributed database and its relationship to Systems Application Architecture (SAA). It shows the importance to effective distribution of IBM's Structured Query Language (SQL), the database element of the Systems Application Architecture Common Programming

Interface (SAA CPI). The paper defines five levels of distribution, showing how each fits real-world application requirements. Finally, it outlines the magnitude of the task.

SAA distributed processing *by A. L. Scherr, p. 370.* Discussed are motivations for distributed versus centralized data processing, the relative advantages of each, and the trade-offs involved as they relate to Systems Application Architecture (SAA). Presented is a taxonomy of the various approaches to designing applications to operate in a distributed manner. SAA support for these modes is described. The management of an enterprise-wide network of systems is discussed.

The Cross System Product application generator: An evolution *by W. K. Haynes, M. E. Dewell, and P. J. Herman, p. 384.* An application generator is a generalized application development tool with which professional programmers develop applications using a fourth-generation language. This paper describes the requirements that led to the Cross System Product application generator, and how the product progressed from a single-environment product to the current multienvironment product. Also described are how the Cross System Product fits within Systems Application Architecture and how that may affect the future of the Cross System Product.

Volume 27, Number 4, 1988

IBM's directions in technical computing *by P. L. Prairie and A. H. Weis, p. 393.* Technical computing comprises hardware systems, software, tools, communications networks, and applications to significantly increase productivity and competitiveness. A Technical Computing Structure (TCS) is described which provides a framework and IBM's direction to integrate these elements, including both IBM offerings and industry standards to support the technical end user. Also discussed are some of the special studies undertaken to improve development programs for this environment.

Engineering and Scientific Subroutine Library for the IBM 3090 Vector Facility *by J. McComb and S. Schmidt, p. 404.* The Engineering and Scientific Subroutine Library (ESSL) provides FORTRAN, Assembler, and APL2 application programmers with a high-performance set of mathematical subroutines which take advantage of the performance gains offered by the IBM 3090 Vector Facility. This paper describes the contents of ESSL and presents some of the techniques that were used to develop high-performance vector subroutines. Other key design considerations such as accuracy, ease of use, and error handling are also discussed. This information should be useful to anyone developing programs for the IBM 3090 Vector Facility.

IBM Parallel FORTRAN *by L. J. Toomey, E. C. Plachy, R. G. Scarborough, R. J. Sahulka, J. F. Shaw, and A. W. Shannon, p. 416.* IBM Parallel FORTRAN is a compiler and library for writing and executing parallel programs. It provides language extensions for explicitly programming in parallel, and it also provides compiler enhancements for automatically generating both parallel and vector code. Parallel FORTRAN offers a language for parallel programming that is independent of the machine configuration and the operating system. The combination of Parallel FORTRAN and IBM 3090 multiprocessors can provide a significant reduction in turnaround time for applications.

Program locality of vectorized applications running on the IBM 3090 with Vector Facility *by K. So and V. Zecca, p. 436.* An instruction-level simulator is used to study the program locality of large scientific applications. The simulator, which models an IBM 3090 processor with Vector Facility and a cache, was developed to help a programmer improve the performance of an application through better understanding and use of the Vector Facility and the memory hierarchy of the IBM 3090 system. Our main observations on a set of scientific applications are as follows: (1) although the applications have different characteristics of memory accesses and vectorization, their program locality is high enough to take advantage of conventional cache structures; (2) the cache hit ratio of the vector execution can be quite different from (but not significantly lower than) that of the scalar execution of the same application; and (3) the application programs that are written to optimize the use of the memory hierarchy in the system generally result in higher cache hit ratios than the others. The cache performance of these applications with respect to various cache parameters is also presented. In particular, our study finds that the cache structure of the IBM 3090 is well suited for large scientific applications.

Programming style on the IBM 3090 Vector Facility considering both performance and flexibility *by H. Samukawa, p. 453.* To obtain high performance from the IBM 3090 Vector Facility, we must investigate vector instruction constructs in terms of the loop context of the application algorithm. We exemplify the method by linear algebra subroutines for basic matrix operations and a linear equation solver. In these examples, we clarify the mathematical meaning that each loop is computed by analyzing the loops in terms of a generic algorithm. This analysis helps us to achieve optimal loop selection. We then obtain additional performance gain by considering cache capacity. These procedures suggest that there are three levels of performance classification. They also show that program structure yields great benefits in terms of performance and generality of the program.

ICAP 3090: Parallel processing for large-scale scientific and engineering problems *by E. Clementi, D. Logan, and J. Saarinen, p. 475.* Described is the ICAP/3090 (for loosely coupled array of processors) parallel processing system. General parallel processing performance issues that determine the success

of all multiple-instruction/multiple-data-stream parallel computing systems are examined in the context of large-scale scientific and engineering problems. Experiments with previous ICAP parallel processing systems that have made possible the present design of ICAP/3090 are also described.

Seismic computations on the IBM 3090 Vector Multiprocessor *by A. Kamel, M. Kindelan, and P. Sguazzero, p. 510.* Computerized seismic prospecting is an echo-ranging technique usually targeted at accurate mapping of oil and gas reservoirs. In seismic surveys an impulsive source, often an explosive charge, located at the earth's surface generates elastic waves which propagate in the subsurface; these waves are scattered by the earth's geological discontinuities back to the surface, where an array of receivers registers the reflected signals. The data recorded are then processed in a complex sequence of steps. Among them, seismic migration and stacking velocity estimation represent two characteristic components of the process solving the inverse problem of recovering the structure and the physical parameters of the earth's geologic layers from echo measurements. A complementary tool in relating seismic data to the earth's inhomogeneities is provided by seismic numerical models, which assume a subsurface structure and compute the seismic data which would be collected in a field survey, by solving the direct problem of exploration geophysics. This paper describes a vectorized and parallelized implementation of a two-dimensional seismic elastic model on the IBM 3090 VF Vector Multiprocessor. An implementation of a parallel seismic migration algorithm is then described. The paper also reports performance data for a vector/parallel implementation on the IBM 3090 of some typical seismic velocity estimation algorithms. The three problems chosen are representative of a wide class of geophysical computations, and the results summarized in this paper show their suitability for efficient implementation on the IBM 3090 Vector Multiprocessor; combined vector/parallel speedups in the range 15–25 are in fact observed.

Effective utilization of IBM 3090 large virtual storage in the numerically intensive computations of *ab initio* molecular orbitals *by M. Sakaki, H. Samukawa, and N. Honjou, p. 528.* A new level of storage hierarchy, called Expanded Storage and available on the IBM 3090 system, is utilized by the MVS/XA™ operating system as high-speed paging equipment, allowing a user to hold application data in large virtual storage. To exploit the large virtual storage capability of the IBM 3090, a new application technique was developed for numerically intensive computations of *ab initio* molecular orbitals where high-speed transfer of a vast amount of intermediate data is a common requirement of most application programs. An application program running under MVS/XA was modified so that it could handle a vast amount of intermediate data in large virtual storage combined with Expanded Storage, achieving a 4- to 10-fold improvement in turnaround time at a CPU rate-determining step (SCF step) in medium-sized molecules.

PAM-CRASH on the IBM 3090/VF: An integrated environment for crash analysis *by P. Angeleri, D. F. Lozupone, F. Piccolo, and J. Clinckemaillie, p. 541.* PAM-CRASH® is an industrial code developed by Engineering Systems International (ESI) S.A. and designed specifically for automotive crashworthiness analysis. We discuss the problems encountered and describe the solutions provided for an efficient migration of the code on the IBM 3090/VF system. Runs on actual test cases have shown a vector/scalar speedup between 2.7 and 3.5. Moreover, we present the program modifications we have introduced in order to exploit parallel processing using the Multitasking Facility of the VS FORTRAN compiler. Performance results for 3090/VF systems, from the Model 200E to the Model 600E, are given. Finally, we describe the restructuring of the graphic processors, PRE-3D and DAISY, to allow an effective use of the IBM 5080 Graphics System capabilities in providing an integrated design environment for crash analysis.

Interactive computations and display of characteristics of the radiation scattered by a sphere: A demonstration for PS/2 Model 80 *by P. Halpern and J. V. Dave, p. 561.* The Personal System/2® (PS/2®) Model 80 with its math coprocessor provides a considerable amount of computing power which can be used with advantage to solve technical problems interactively at a stand-alone workstation. To demonstrate this capability, a scientific program routinely used in diverse disciplines requiring significant computing power was modified to run on the PS/2 Model 80. It computes and displays variations of the specific intensity and degree of polarization of the electromagnetic radiation scattered by a sphere of given refractive index. For a sphere of size parameter of 100, about 475 000 double-precision floating-point calculations are performed and the results displayed in graphic format in less than ten seconds. The selected algorithm is routinely used in several different disciplines such as astronomy, atmospheric optics, chemical engineering, colloidal chemistry, and remote sensing. Because it requires the evaluation of spherical Bessel functions with complex arguments, and derivatives of the Legendre polynomials, it was selected as a representative problem of numerically intensive computing.

Volume 28, Number 1, 1989

Large systems and Enterprise Systems Architecture *by B. R. Aken, Jr., p. 4.* A number of diverse factors have influenced the development of IBM's Enterprise Systems Architecture. They range from the compatibility and migration considerations so important for preserving customer investments in existing applications and data, to new functional and capacity requirements of our customers, to the implications of emerging technologies and the projection of these into the systems environments of the future.

This paper provides an introduction to a collection of technical papers in this issue describing the ESA/370™ facilities. Its purpose is to convey a broad perspective on important factors that will influence the large-systems environment of the future, and to relate those factors to the key elements of the ESA/370 architectural enhancements. It does not attempt to address all of the considerations leading to the development of ESA/370, nor discuss the new features in any depth. Detailed discussion of the specific features, facilities, and design considerations of ESA/370 will be found in other papers in this issue.

Enterprise Systems Architecture/370: An architecture for multiple virtual space access and authorization *by C. A. Scalzi, A. G. Ganek, and R. J. Schmalz, p. 15.* The Enterprise Systems Architecture/370™ provides a significant step in the IBM System/370 evolution by providing new capabilities for virtual addressing and program linkage across multiple address spaces. This paper reviews the evolution that led to this advance and illuminates the goals, such as eliminating growth constraints and improving security, integrity, reliability, and performance, that have guided it. The major architectural capabilities are discussed, along with the system environments in which they are useful. The rationale for design choices is presented and related to issues of performance, access authorization, and constraints relief.

Concepts of Enterprise Systems Architecture/370 *by K. E. Plambeck, p. 39.* Enterprise Systems Architecture/370™ (ESA/370™) is the next step in the architectural evolution of IBM's large processors from System/360 to System/370 to System/370 Extended Architecture (370-XA). ESA/370 includes all of the facilities of 370-XA and also significant new facilities. It greatly increases the amount of apparent main storage that is readily available for use. It provides for more efficient secure program linkage, with increased status saving and restoring, among hierarchically or nonhierarchically organized programs. It allows improved control program efficiency.

Storage hierarchies *by E. I. Cohen, G. M. King, and J. T. Brady, p. 62.* The storage hierarchy is a natural structure, given the set of available technologies and their price and performance characteristics. The physical structure of the storage subsystem is described, and the flow of data through the system is traced. The concept of a storage hierarchy is discussed, and the specific components of the IBM storage hierarchy from the processor high-speed buffer (HSB) to the on-line DASD configuration are described in detail. Trade-offs between technologies and the interactions among the levels of the hierarchy are discussed. In particular, the importance of the I/O boundary, processor storage volatility, and data sharing are highlighted. A continuous increase in virtual storage capacity can be seen in the evolution of large-scale operating systems, and MVS/ESA™ now provides the ultimate virtual capacity and function. New virtual structures available in MVS/ESA are discussed, and their relationship to the storage hierarchy is studied. The importance of storage to the performance and cost of a large processing system leads to a discussion of guidelines for storage configuration and data placement within the hierarchy.

System-managed storage *by J. P. Gelb, p. 77.* In early 1988, IBM announced the Data Facility Storage Management Subsystem (DFSMS™), comprising functions in MVS/DFP™, other products in the Data Facility family, and RACF. This announcement constituted a major step in the realization of system-managed storage. The need for system-managed storage was established in the late 1970s and early 1980s, through growing customer requirements in the management of external storage space, performance, availability, and device installability within and across systems in these customers' installations. The concept of system-managed storage is an evolutionary one, culminating in a resource manager for external storage that separates the logical view of data from physical device characteristics, simplifies interfaces for the use and administration of storage, integrates the functions of storage management products, and provides a synergy of hardware and software functions to effect complex-wide management of external storage resources, as discussed in this paper.

Multiple operating systems on one processor complex *by T. L. Borden, J. P. Hennessy, and J. W. Rymarczyk, p. 104.* As large computing systems continue to grow in capacity and to offer improved price/performance, there is an increasing requirement to consolidate systems onto one processor complex. This paper describes the reasons why users need to run multiple operating systems today, provides a brief history of IBM's partitioning products, and introduces the Processor Resource/Systems Manager™, a machine feature on the IBM 3090 Model E and ES/3090™ Model S processors that provides users with a flexible and efficient capability to run multiple operating systems on a single processor complex.

The facilities and evolution of MVS/ESA *by C. E. Clark, p. 124.* As new processors were developed with new capabilities, the Multiple Virtual Storage (MVS) operating system was modified and enhanced to utilize the latest advances. The most recently available processors are structured on Enterprise Systems Architecture, and MVS has evolved to be a part of this architecture as MVS/ESA™. This paper describes the changes that occurred in MVS and the facilities that are currently available to support users of the latest processors.

MVS data services *by K. G. Rubsam, p. 151.* The IBM Enterprise Systems Architecture/370™ vastly increases the potential virtual addressability available to both system and application programs. The I/O model and the application model of permanent data are discussed to illustrate how large virtual addressability can be used to simplify application programs

and improve performance. New MVS services that exploit the architecture are described. Also described are data window services, which are callable from high-level languages and provide the capability to manage very large permanent and temporary objects in virtual storage.

VM/XA SP2 minidisk cache *by G. P. Bozman, p. 165.* Given the growing disparity between CPU power and the speed of secondary storage, a data cache exploiting large processor storage has the potential to improve response time dramatically in many situations. The VM/XA SP2 minidisk cache facility, the result of research activity on the characteristics of interactive file-system activity, uses expanded storage to cache input/output to minidisks on the Conversational Monitor System. The size of the cache is dynamically adjusted by an arbitration process to optimize system performance. Several other functions improve the performance of the cache during periods of unusual I/O loads.

VM/XA storage management *by G. O. Blandy and S. R. Newson, p. 175.* The VM/XA System Product manages the vast amounts of real and expanded storage available on the new Enterprise Systems Architecture/370™ processors for both guest use and support of internal operating system functions. The management algorithms are examined, and the rationale for their selection is presented.

Volume 28, Number 2, 1989

Evolution of the DASD storage control *by C. P. Grossman, p. 196.* This paper identifies the major requirements and design points for storage controls and describes how these requirements have been met over time. It also describes the interplay of three critical components of a subsystem: hardware technology, microcode, and software.

Local-area distributed systems *by R. C. Summers, p. 227.* Advances in computing and networking have led to the use of local-area distributed systems. The following are example configurations: workstations and file servers, multiple computers that present the image of a single computer, and heterogeneous workstations and mainframes that cooperate loosely. The paper focuses on the system software. It first discusses the forces leading to distributed systems and the obstacles to realizing the full value of the systems. Discussed also are common current uses of local-area distributed systems. Concepts and models are introduced. Requirements for user and program interfaces and for administration are presented, as well as major design attributes and design issues. Systems that represent the main approaches are described.

System-independent file management and distribution services *by J. C. Ashfield and D. B. Cybrynski, p. 241.* Applications universally require files to move from one location to another. Although different applications may use files differently, many of the files are of the same type. The identifying, fetching, moving, and storing functions are the same for all applications and can be most efficiently provided by a common process. Various applications can invoke the common process, which performs the required operations independently and notifies the appropriate applications when they are completed. In Systems Network Architecture (SNA) networks, the common process is an SNA/Distribution Services (SNA/DS) server defined by SNA/File Services (SNA/FS). The invoking applications are SNA/DS agents of various types. This paper describes the role of the agent in invoking the file transfer and the role of the SNA/FS server in fetching and storing the file. It also describes the SNA/FS architecture for uniquely naming files and data objects. One example of an SNA/DS agent that uses the SNA/FS server is the change management category of SNA/Management Services, described in another paper in this issue.

Managing changes in SNA networks *by C. P. Ballard, L. Farfara, and B. J. Heldke, p. 260.* Systems Network Architecture/Management Services (SNA/MS) has been enhanced to give network users change management capabilities. The first IBM products implementing change management are NetView™ Distribution Manager R2 and the 3174 Control Unit with the Central Site Change Management microcode function. This paper describes the design selected and the functions provided: Retrieve, Send, Delete, Install, Send-and-Install, Remove, Accept, and Activate. It also describes how SNA/MS makes use of another new SNA component designed for it—SNA/File Services, described in another paper in this issue. (Although not strictly necessary, it is recommended that the other paper be read prior to reading this one.) SNA/File Services, in turn, uses an enhanced SNA/Distribution Services format to provide an architecture for file distribution in an SNA network.

REXX on TSO/E *by G. E. Hoernes, p. 274.* REXX is a programming language primarily designed for ease of use. First implemented on the Conversational Monitor System (CMS), REXX has been implemented on TSO Extensions (TSO/E) as a new command language, yet it contains all of the elements of a full-function language. After a brief definition of the main elements of the REXX language, the paper discusses why REXX was implemented on TSO/E, some alternative designs which were considered, and how the final design integrates the new language into the existing TSO/E structure, yet allows REXX programs to be interpreted in any Multiple Virtual Storage (MVS) address space, even outside the TSO/E environment. The paper also introduces the TSO/E "data stack," which is similar to the stack implemented in CMS, and describes how the definition of the CMS stack had to be extended to allow REXX programs executing concurrently on different MVS tasks to either share or not share the data stack. Throughout the paper, compatibility with other Systems Application Architecture environments, partic-

ularly CMS, and performance considerations are discussed.

Program understanding: Challenge for the 1990s *by T. A. Corbi, p. 294.* In the Program Understanding Project at IBM's Research Division, work began in late 1986 on tools which could help programmers in two key areas: static analysis (reading the code) and dynamic analysis (running the code). The work is reported in the companion papers by Cleveland and by Pazel in this issue. The history and background which motivated and which led to the start of this research on tools to assist programmers in understanding existing program code is reported here.

DS-Viewer—An interactive graphical data structure presentation facility *by D. P. Pazel, p. 307.* DS-Viewer is a tool that is the result of a research project in data structure presentation within a program state. This tool addresses two distinct issues in this area: (1) to effectively present data structures themselves for a given program state and (2) to present groups of data structures and their interrelationships as described by their pointer definitions. Graphical presentations were developed to address these issues. For the data structure presentation, the user is provided a display window for any single data structure instance formatted with its fields and field values. Flexibility in display is provided by allowing the user a choice from the various value formats for each field. For groups of data structure instances, a graphical drawing space is provided in which pictures of these data structure instances and their interrelationships are drawn as blocks and arrows. The computer assists the user in drawing such a picture by describing its components, allowing the user to choose which to draw and to construct as much of the picture as desired.

A program understanding support environment *by L. Cleveland, p. 324.* Software maintenance represents the largest cost element in the life of a software system, and the process of understanding the software system utilizes 50 percent of the time spent on software maintenance. Thus there is a need for tools to aid the program understanding task. The tool described in this paper—Program UNderstanding Support environment (PUNS)—provides the needed environment. Here the program understanding task is supported with multiple views of the program and a simple strategy for moving between views and exploring a particular view in depth. PUNS consists of a repository component that loads and manages a repository of information about the program to be understood and a user interface component that presents the information in the repository, utilizing graphics to emphasize the relationships and allowing the user to move among the pieces of information quickly and easily.

Technical note—Engineering and Scientific Subroutine Library Release 3 for IBM ES/3090 vector multiprocessors *by R. C. Agarwal, F. G. Gustavson, J. McComb, and S. Schmidt, p. 345.* This technical note should be read in conjunction with the paper by McComb and Schmidt which describes the Engineering and Scientific Subroutine Library through Release 2. In this technical note, which is an addendum to that paper, we briefly describe some of the new features in Release 3 and indicate some of the techniques used to optimize vector and parallel performance.

Volume 28, Number 3, 1989

System overview of the Application System/400 *by D. L. Schleicher and R. L. Taylor, p. 360.* This paper describes IBM's recently available general-purpose midrange computers—the Application System/400™, the basic intentions of the product, the significant factors setting forth system requirements, the primary design themes incorporated in the implementation of those requirements, and a description of some of the key system components. However, the paper is not intended to provide a complete system description.

Design, test, and validation of the Application System/400 through early user involvement *by B. J. Pine II, p. 376.* The Application System/400™ (AS/400™) is the culmination of a development effort requiring seven million lines of code. Key challenges to its development were those of ensuring that the system had been designed correctly and thoroughly tested, that IBM Business Partners were ready for its introduction together with their applications, and that IBM marketing representatives and systems engineers were trained and knowledgeable on the system. This paper discusses how these challenges were met through the involvement of customers, Business Partners, vendors, and systems engineers in the development of the AS/400 system so as to positively affect its design and quality.

A new development rhythm for AS/400 software *by R. A. Sulack, R. J. Lindner, and D. N. Dietz, p. 386.* Synchronizing the software development process with hardware development and user involvement programs yielded a product offering that met the user requirements with a significantly reduced development cycle. This paper emphasizes the key elements of Application System/400™ (AS/400™) software development that contributed to synchronization and project success. It is intended to produce an awareness of the elements that set this project apart from most others.

Application System/400 performance characteristics *by B. E. Clark and M. J. Corrigan, p. 407.* The operating system for Application System/400™ (AS/400™) provides an unprecedented breadth of function and system services in a single, integrated system. The majority of functions are implemented on top of an abstract, high-level machine interface in a hardware-independent manner, using many architectural characteristics normally associated with poor performance. Despite these architectural and functional traits of the operating system, the AS/400

exhibits excellent price and performance characteristics for commercial applications and is a competitive system in the mid-range commercial application arena. A number of design and optimization techniques, many of them unique or innovative, were incorporated into the AS/400 to achieve a combination of advanced design, function, and performance and are the main subjects discussed in this paper.

The Application System/400 help facility—design philosophy and considerations *by D. A. Charland, p. 424.* The design of the Application System/400™ (AS/400™) system help facility was based on the philosophy that users must be able to quickly access the specific information needed to complete their immediate task. This philosophy, reinforced by experience with help information on earlier IBM systems, resulted in a modular help facility that provides two major types of assistance discussed in this paper. Contextual information based on cursor position is provided for each panel. This contextual help is supplemented by an index of how-to-do-it and what-it-means information that can be searched by users in their own words.

Design rationale of the AS/400 user interface *by J. H. Botterill, p. 443.* This paper discusses the design rationale of the software user interface of the Application System/400™ (AS/400™). It presents the design approaches used to produce the interface of this interactive system.

Object-oriented programming *by R. P. Ten Dyke and J. C. Kunz, p. 465.* Object-oriented programming involves a new way of thinking about and programming applications. The thought process and techniques are introduced through a discussion of the language Smalltalk and through an illustrative example. These concepts are extended to a hybrid functional language, object-oriented system, KEE®, and illustrated through the use of knowledge-based system examples.

A message management system for personal computers *by L. d'Arielli, p. 479.* This paper presents a design for a message management system that reduces the coding effort for the application developer, gives the user greater control over the treatment of application messages, and eliminates many problems of translation. Any message may be directed to one or more devices (screen, printer) and/or files (log, activity), with the ability to exclude message elements (e.g., date and time) from being sent to any output destination, or to exclude a destination altogether. Because the message-handling code and message texts are separated from one another and from the application code, the developer need only issue a message identifier and set the associated variable values. The message identifier contains codes for both class and severity, and the developer provides default selection criteria that the user can modify. These tables of selection criteria provide a simple yet highly flexible means of determining where each message will be sent and in what form. The simplicity of including variable information and the separation of message texts into a master repository (where an information developer or translator can work on them) tend to improve the quality of messages to the user by making them more consistent and informative.

Volume 28, Number 4, 1989

History and contributions of the IBM Scientific Centers *by H. G. Kolsky and R. A. MacKinnon, p. 502.* The IBM Scientific Centers are celebrating their twenty-fifth anniversary. These worldwide Centers are autonomous organizations that provide IBM with the ability to respond rapidly to the evolution of computer technology for IBM and for its scientific customers. During the past quarter century these Centers have provided technical leadership in almost every branch of computer science. Today, the 17 individual Centers continue to explore new technical areas and provide significant contributions. This paper has three parts: an introduction to the mission, scope, and history of the Centers; a description of each Center's charter, history, and accomplishments; and an extended list of selected publications for each Center.

Visual programming: Perspectives and approaches *by N. C. Shu, p. 525.* Visual programming tackles the problem of bringing computing facilities to people who do not have extensive computer training by using visual (i.e., nonlinear) representations in the programming process. In this paper, we first define visual programming and briefly discuss its many facets. The purpose is to lay a conceptual background so that common understanding can be established and various aspects of visual programming can be focused on and examined. We then concentrate on visual programming languages, namely, languages that enable the users to "program" with visual expressions. Examples are used to illustrate three fundamentally different approaches: diagrammatic, iconic, and form based. Finally, we show that FORMAL, a system developed and implemented at the IBM Los Angeles Scientific Center, not only captures the spirit of visual programming languages but also has the capability to automate a wide variety of common data processing applications.

The WINSOM solid modeller and its application to data visualization *by J. M. Burridge, B. M. Collins, B. N. Galton, A. R. Halbert, T. R. Heywood, W. H. Latham, R. W. Phippen, P. Quarendon, P. Reilly, M. W. Ricketts, J. Simmons, S. J. P. Todd, A. G. N. Walter, and J. R. Woodwark, p. 548.* The IBM United Kingdom Scientific Centre's WINchester SOlid Modelling system (WINSOM) is a set-theoretic, constructive solid geometry (CSG) modeller based on recursive division techniques. It specializes in handling complex models and provides graphical facilities intended for engineering applications. This paper describes WINSOM and some of the many programs that are linked to it, and gives examples of their application to problems of data visualization.

Data visualization in archaeology *by P. Reilly, p. 569.* Archaeological field work produces vast amounts of three-dimensionally recorded data which can only be analysed using computers. Developments in data-visualization techniques are continually increasing the volume and complexity of data that can be studied meaningfully. In particular, three systems developed at the IBM United Kingdom Scientific Centre have been applied in a wide variety of archaeological situations: a graphics-database system called the Winchester Graphics System (WGS), IBM's IAX (Image Applications eXecutive) image processing system, and the WINchester SOlid Modelling system called WINSOM. It has been shown that these systems not only permit well-known problems to be answered in new and interesting ways but have freed archaeologists to explore previously undiscovered avenues of research. The techniques developed using these systems also have major implications for education and training.

GARDEN—An integrated and evolving environment for ULSI/VLSI CAD applications *by A. H. V. de Lima, R. C. B. Martins, R. Stern, and L. M. F. Carneiro, p. 580.* The design and specification of efficient and powerful Ultra Large Scale Integration/ Very Large Scale Integration (ULSI/VLSI) computer-aided design (CAD) systems to deal with the current integrated circuit manufacturing technology is beyond the capabilities of the usual software development methodologies. This paper presents GARDEN, an integrated ULSI/VLSI design environment conceived to cope with problems in the evolution of the computing environment. It also highlights the utilization of the Vienna Development Methodology (VDM) for the specification, design, implementation, and maintenance—in short, all of the software life cycle—of this CAD system, under development at the IBM Brazil Rio Scientific Center.

An Arabic morphological system *by T. A. El-Sadany and M. A. Hashish, p. 600.* Nowadays, computers are used in every field in the Arab countries of the middle east. Software systems developed for the European languages are not convenient for the use of Arabic because of the nature of the language and its writing system. Problems arise when trying to use existing software systems, such as spell-checkers and business and office systems, with the Arabic language. These problems are attributable to the fact that the difference between Arabic and the European languages exists not only in character shapes and direction of writing, but also in language structure. In order to successfully use Arabic in software systems, we must, then, analyze the Arabic language word structure—that is, carry out a morphological analysis. Most of the written Arabic texts are nonvowelized, which may lead to ambiguity in meaning or mispronunciation. Moreover, vowelization cannot be avoided in many applications, such as speech synthesis by machines and educational books for children. A two-way Arabic morphological system (analysis/generation) capable of dealing with vowelized, semivowelized, and nonvowelized Arabic words was developed at the IBM Cairo Scientific Center. The system also has the ability to vowelize nonvowelized words. This system consists of three separate modules: computational lexicon, Arabic grammar model module, and analyzer/generator module. The grammar module contains, among others, morphophonemic and morphographemic rules formulated using the conventional generative grammar. Moreover, the developed system covers all of the Arabic language.

Designing molecules and crystals by computer *by A. Koide, p. 613.* An in-depth overview of computer-aided chemical design is presented through a discussion of three systems that we developed: the Molecular Design Support System, MolWorld, and the Molecular Orbital Graphics System. The first is considered as an example of the kernel of a simulation system for industrial research and development. The chemical formula interpreter and three-dimensional molecular geometry generator of MolWorld are discussed as a compact realization of intelligence. Finally, the use of visualized molecular electronic structures in relation to chemical reactions is considered in the discussion of the Molecular Orbital Graphics System.

S*P*A*R*K: A knowledge-based system for identifying competitive uses of information technology *by P. Gongla, G. Sakamoto, A. Back-Hock, P. Goldweic, L. Ramos, R. C. Sprowls, and C.-K. Kim, p. 628.* The use of information and information technology (IIT) as a strategic tool to gain competitive advantage has become increasingly significant in recent years. Numerous examples of how firms are using IIT to improve their competitive positions are highlighted in both popular and academic literature. Although the potential competitive benefits of IIT are generally recognized by business and I/S executives, there is a great gap between recognizing such value and applying the technology effectively. To help bridge this gap, a group at the IBM Los Angeles Scientific Center has developed a knowledge-based system facilitator, called S*P*A*R*K. The system is designed to help business and I/S managers identify competitive applications of IIT to help them be creative in generating a range of IIT alternatives. This paper provides an overview of S*P*A*R*K, including the conceptual frameworks used for knowledge sources, the design philosophy, functions, and implementation approaches. Examples from a database of competitive applications of IIT are also presented to provide a flavor of the S*P*A*R*K facilitative processes.

Concurrent computing by sequential staging of tasks *by J. Gazdag and H.-H. Wang, p. 646.* Described is a new approach to parallel formulation of scientific problems on shared-memory multiprocessors such as the IBM ES/3090 system. The class of problems considered is characterized by repetitive operations applied over the computational domain D. In each such operation, some fields of interest are extrapolated or advanced by an amount of $\Delta\tau$. The integration variable τ may be time, distance, or iteration

sequence number, depending on the problem under consideration. An extensively studied approach to parallel formulation of such computational problems is based on domain decomposition, which attempts to partition the domain of integration into many pieces, then construct the global solution from these local solutions. Thus, domain decomposition methods are confined to D alone at a single τ level. An inquiry into the possibilities of formulating parallel tasks in τ, or more significantly in the $D \times \tau$ domain, opens up new horizons and untapped opportunities. The aim of this paper is to detail an approach to exploit this τ domain parallelism that will be referred to as sequential staging of tasks (SST). Concurrency is realized by means of ordering the tasks sequentially and executing them in a partially overlapped or pipelined manner. The SST approach can yield remarkable speedup for jobs requiring intensive paging I/O, even when a single processor is available for executing multiple tasks. Noteworthy features of the SST method are demonstrated and highlighted by using results obtained from computer experiments performed with a numerical solution method of the Poisson equation and migration of seismic reflection data.

Advanced Information Management (AIM): Advanced database technology for integrated applications *by P. Dadam and V. Linnemann, p. 661.* The Advanced Information Management (AIM) project is currently one of the main activities at the IBM Scientific Center in Heidelberg. The main purpose of the project is to understand the database requirements and respective solutions for advanced integrated applications such as computer-integrated manufacturing and computer-integrated office. These application areas require an advanced database technology which is able to manage a large variety of data of various types in a consistent and efficient way. The underlying database technology should support not only simple numbers and simple tables used in business administration, but also large complex structured objects, including text, image, and voice data, in a uniform way. This paper describes the background, goals, and accomplishments of the AIM project. It also provides an overview of the design goals, the implementation, and the underlying concepts of AIM-P, an experimental database management system under development in the AIM project.

Technical note—Computer sculpture *by W. H. Latham and S. J. P. Todd, p. 682.* This technical note illustrates the graphic techniques used to generate the cover of this issue. It should be read in conjunction with the paper on WINSOM which describes the computer program used to generate the computer sculptures.

Volume 29, Number 1, 1990

Experiences with Defect Prevention *by R. G. Mays, C. L. Jones, G. J. Holloway, and D. P. Studinski, p. 4.* Defect Prevention is the process of improving quality and productivity by preventing the injection of defects into a product. It consists of four elements integrated into the development process: (1) causal analysis meetings to identify the root cause of defects and suggest preventive actions; (2) an action team to implement the preventive actions; (3) kickoff meetings to increase awareness of quality issues specific to each development stage; and (4) data collection and tracking of associated data. The Defect Prevention Process has been successfully implemented in a variety of organizations within IBM, some for more than six years. This paper discusses the steps needed to implement this process and the results that may be obtained. Data on quality, process costs, benefits, and practical experiences are also presented. Insights into the nature of programming errors and the application of this process to a variety of working environments are discussed.

Implementing the Defect Prevention Process in the MVS Interactive programming organization *by J. L. Gale, J. R. Tirso, and C. A. Burchfield, p. 33.* A process for preventing defects has been gaining momentum in the IBM Corporation as a way to improve quality and increase productivity. The Communications Programming Laboratory in Research Triangle Park, North Carolina (near Raleigh), has been implementing the process for the past six years and has realized a 54 percent reduction in errors. This paper documents experiences at the IBM Myers Corners Laboratory MVS Interactive programming area in putting the Defect Prevention Process theories into practice. This paper begins with the proposal to adopt the Defect Prevention Process at the Myers Corners Laboratory in Poughkeepsie, New York, and our experiences thus far. It is our belief that other organizations can benefit from our experiences by understanding how the Defect Prevention Process can be adapted to best meet the needs of any organization.

Effective application development for Presentation Manager programs *by S. M. Franklin and A. M. Peters, p. 44.* The OS/2™ Presentation Manager™ provides an integrated graphical, windowing user interface to IBM's OS/2 operating system. This paper addresses a primary area of interest for Presentation Manager application developers: the use and development of user controls. A control in the Presentation Manager environment is a program object with a programming interface and application function. The structure and interfaces between controls and the system are described in order to provide an understanding of the correct procedure for programming the Presentation Manager efficiently.

Using box structures for definition of requirements specifications *by J. E. Odom, p. 59.* Box structures provide a stepwise refinement and verification methodology for information systems analysis and design. They are especially useful for recording and decomposing requirements specifications. The benefits of using the structures center around making the requirements clear to readers, helping to make the requirements complete, and providing an artifact that

will enhance the traceability of the requirements. This paper describes the methodology of applying box structures and presents an example of their use in the definition of requirements specifications.

Implementing tool support for box structures *by B. S. Tagg, p. 79.* This paper describes a feasibility study to implement partial tool support for the graphical component of the box structure methodology (BSM). By following the defined strategy and process, an existing computer-aided software engineering (CASE) environment has been extended with a customizer to provide support for the box definition graphics (BDG) component of BSM. The critical functions required from a CASE environment are also described to provide the reader with a background for selecting one of the various implementations available today.

Porting DPPX from the IBM 8100 to the IBM ES/9370: Feasibility and overview *by R. Abraham and B. F. Goodrich, p. 90.* The DPPX/SP operating system was converted from its original implementation on the IBM 8100 Information System architecture to a new implementation—DPPX/370—on the System/370 architecture of the ES/9370 Information System processors. Portability was not an original design objective for DPPX, and yet the conversion of the operating system was straightforward and successful. This paper investigates the design fundamentals and technical approaches that led to the successful porting of DPPX/SP to the ES/9370.

Porting DPPX from the IBM 8100 to the IBM ES/9370: Migration *by C. Goodrich and M. B. Loughlin, p. 106.* This paper explains the development of the migration process by which applications running on a network of DPPX/SP systems would migrate to a network of DPPX/370 systems. DPPX/SP is a centrally managed, distributed processing system designed to run on the IBM 8100 family of processors. DPPX/370 is the DPPX/SP system ported to the IBM ES/9370 family of processors. The paper outlines the strategies, the technical problems encountered and their solutions, customer participation, and testing. Finally, it provides recommendations on how the process might have been improved.

Porting DPPX from the IBM 8100 to the IBM ES/9370: Installation and testing *by G. E. Boehm, A. M. Palmiotti, and D. P. Zingaretti, p. 124.* This paper describes the software tools, testing activities, and testing methods that were used to port the DPPX/SP operating system from its original implementation on the IBM 8100 Information System to its new implementation on the IBM ES/9370 Information System.

REASON: An intelligent user assistant for interactive environments *by J. M. Prager, D. M. Lamberti, D. L. Gardner, and S. R. Balzac, p. 141.* The provision of intelligent user assistance has been an ongoing problem in designing computer interfaces. Interactive computing environments must support expert as well as novice users when providing advice for error correction and answers to questions directed to a system. To address these issues, we have investigated the application of fairly well-understood artificial intelligence techniques in novel ways to provide intelligent help. This paper describes the design methodology used to build REASON (Real-time Explanation and SuggestiON), an intelligent user-assistant prototype for a windowed, multitasking environment. REASON's central component is an inference engine that solves problems arising from a user's activity. When the user makes one of several different kinds of errors, the inference engine offers dynamically generated suggestions about what the user might have intended. The user can also query REASON using natural language. In addition to providing suggestions of corrected input or answers to questions, REASON can provide two complementary types of explanations of these responses, derived from the inferences that led to them.

Volume 29, Number 2, 1990

AD/Cycle strategy and architecture *by V. J. Mercurio, B. F. Meyers, A. M. Nisbet, and G. Radin, p. 170.* Over the years, IBM has made progress in resolving many of the issues that deal with improving application development (AD) productivity and quality. Systems Application Architecture™, together with IBM's recently announced AD/Cycle™ direction, provides a platform for even greater progress. This paper addresses the IBM strategy that supports AD/Cycle and gives an overview of the major components of the AD/Cycle architecture. This paper is an introduction to other papers that follow in this issue.

The role of work management in application development *by G. Chroust, H. Goldmann, and O. Gschwandtner, p. 189.* Quality is probably one of the most serious concerns of today's software community. For software applications exhibiting a certain complexity, the quality of a product can only be guaranteed by a methodological approach, using appropriate administration and tools. The methodology and administration must be manifested in a well-defined and well-observed application development process. The process must integrate the human activity, the tools, and the intermediate and final work products into a coherent flow of actions. In this regard, the development of applications follows patterns that are well established in other industries where an application development (AD) process model is defined and then executed via an interpretation mechanism. The complexity of the development process makes it necessary to support and integrate all of its aspects by means of on-line interactive computer support. Computer-aided process support in the general sense we call work management. This paper explains the concepts of an application development process model and of work management for application development under AD/Cycle™ and its relation to project management.

Repository Manager technology *by J. M. Sagawa, p. 209.* IBM's Repository Manager™ enables specifications involved in the program application development process to be managed. On the basis of the technology, the Repository Manager/MVS™ was developed as a product. The primary concepts and services of the technology are introduced, and specific aspects of the product and its operation are discussed. A discussion of what is involved in designing and implementing a tool is also included.

Data modeling for software development *by R. W. Matthews and W. C. McGee, p. 228.* One of the motivations for the use of a facility such as the Repository Manager™ in an information processing system is to centralize the information needed for the development of software. What this information is and how it is interrelated is defined in the underlying data model. This paper discusses the kinds of information required for software development and offers some suggestions on how the data model should be organized and implemented.

User interface services in AD/Cycle *by J. M. Artim, J. M. Hary, and F. J. Spickhoff, p. 236.* Significant progress has been made in the effort to separate programmers from the management of data storage. By comparison, the window of a workstation is still managed and controlled in great detail by the typical programmer. In AD/Cycle™ user interface services defined a set of services that assist in the management of the displays on the workstation. These services also help increase the productivity of the tool builder by enforcing Common User Access rules and guidelines, and raise the level of consistency of user displays of the tools in AD/Cycle.

DevelopMate: A new paradigm for information system enabling *by K. P. Hein, p. 250.* This paper discusses a new approach to the use of information systems that is based on enterprise information system modeling concepts. This approach is primarily oriented to the enterprise expert, who is considered to be the individual most familiar with the functioning of a particular area of the enterprise information system. The approach is not primarily oriented toward the data processing professional. The paper discusses the phases of the approach and how the DevelopMate™ software product supports some of those phases.

Cross System Product application generator: Application design *by M. E. Dewell, p. 265.* This paper describes some techniques that can be used for Cross System Product/Application Development (CSP/AD) application design. CSP/AD is an application development tool for professional programmers. A well-designed application is obtained by using proven principles of structured analysis, structured design, and structured programming. An understanding of these principles and the application definition constructs provided by Cross System Product/Application Development is necessary for the CSP/AD application designer. Application design for CSP/AD is accomplished by using a combination of techniques for data design, application design, and application program design. For each of these design techniques there exist formal, accepted practices, and methodologies that may be used. These techniques are described, and methods that have proven successful for designing CSP/AD applications are presented.

Knowledge-based systems in the AD/Cycle environment *by D. M. Hembry, p. 274.* Knowledge-based systems technology is a branch of artificial intelligence that deals with the processing of knowledge, as distinct from other branches of artificial intelligence that deal with topics such as robotics, vision systems, and speech recognition. This paper describes how, over the last decade, knowledge-based systems have evolved into a viable technology for building commercial data processing applications, and how increasing attention has been paid to incorporating these applications into commercial data processing environments. A logical conclusion of this direction is the capability to build knowledge-based applications that are full Systems Application Architecture™ (SAA™) applications. As this conclusion is approached, a requirement emerges that the knowledge-based development process be integrated with the application development environment provided by the other SAA language and service components. The integrated environment must provide high customer productivity in the development of applications that use knowledge-based technology, and must support a spectrum of development scenarios, ranging from the most basic to those involving complex applications and large development teams. This paper explores how knowledge-based products can address these requirements by integrating their development facilities with AD/Cycle™.

Segmenting discrete data representing continuous speech input *by R. D. Faulk and F. Goertzel Gustavson, p. 287.* A probabilistic method for segmenting continuous speech into lexical units is described. The algorithm assumes initial conversion of the continuous speech signal to a discrete representation over some suitable alphabet. The problem of determining such alphabets is not considered. Experiments used keyed input in English, French, German, and Russian. We hypothesize that the low error rates obtained in the experiments can also be achieved with data representing actual speech. The paper discusses an area of linguistic science, and outlines a method for investigating it.

Volume 29, Number 3, 1990

Operational image systems: A new opportunity *by L. C. Kingman III, R. E. Lambert, and R. P. Steen, p. 304.* Within the span of a few years, image processing has evolved from an esoteric, expensive technology to an indispensable tool used by modern businesses to manage the overwhelming flood of paperwork. Some of the background for this evolution and the development of the IBM system solution

for image processing, ImagePlus™, are described in this paper.

Introduction to image technology *by R. M. Helms, p. 313.* Business today is wrestling with mountains of paper that must be moved, filed, located, and moved again from person to person. Often the paper must be stored for extended periods of time, sometimes as long as seven or more years. The long-term storage of paper records is becoming more and more costly. An image system not only makes the document capture, retention, and retrieval process more cost efficient, but also makes it a faster service to the users. The purpose of this paper is to explain the basic concepts of image processing in business.

The Image Object Content Architecture *by Y. Hakeda, p. 333.* Technical advances to image processing and the availability of the resulting technologies at reasonable cost have helped to promote the use of images in office, engineering, and scientific environments. As evidence of this use, a wide variety of applications and products designed for image processing have been introduced into the market in recent years. In order to encompass different applications and products in a single image processing system and to allow image data to be exchanged and interpreted consistently throughout the system, IBM has introduced the Image Object Content Architecture (IOCA). This paper discusses requirements for the architecture, concepts of the architecture, use of the architecture in the different data stream environments used by image processing systems, and the IOCA function sets that have been defined for interchange within Systems Application Architecture™ environments.

Large-scale image systems: USAA case study *by C. A. Plesums and R. W. Bartels, p. 343.* A large-scale, optical disk-based, operational image system for office-size documents has been implemented to support an insurance customer service application. Images stored on optical disk can be displayed on any of the more than 1400 workstations or printed on any of 22 printers. This system was the prototype for the IBM MVS/ESA™ ImagePlus™ product. Each day over 25 000 pages of incoming mail are scanned, stored, and delivered to users for processing. In addition, computer-generated data (soon expected to reach one million pages per day) are stored for display or print on image-stored overlay forms. The system is described in the context of any large-scale office document application. The discussion includes some of the business factors that created an environment for success and the business issues that led to the development of the system. The paper discusses the discoveries and lessons learned from use of two pilot systems. The authors conclude that the present level of technology makes this a good time to move forward with the installation of large-scale operational document image systems.

ImagePlus as a model for application solution development *by C. D. Avers and R. E. Probst, p. 356.* An early effort by IBM to use system integration services to assist in solving complex problems for commercial customers involved developing an image system for USAA, a large financial services association. USAA had well-defined and stringent requirements for a policy services application that required enhancements to existing products to provide the necessary function and performance. Key problems solved included managing a storage hierarchy to handle image size objects, the use of optical storage as a low-cost storage medium, and the capability to compress and decompress images rapidly at a workstation to allow high-speed paging through documents. Additionally, the registering and indexing of documents and management of work flow and recovery issues were undertaken. The effort was a good example of the new role of application solution development in that the solution was developed in conjunction with a specific customer, but has developed into a product. The particular solution described in this paper became IBM's ImagePlus™ MVS/ESA™ product. Although the specific technical issues were different, the same methodology was used to develop ImagePlus for the System/36 and Application System/400®.

Image system communications *by H. M. Morris and R. H. Orth, p. 371.* This paper discusses the communication requirements to support the IBM ImagePlus™ system. The analysis and approach discussed are based on the experience gained in installing the initial ImagePlus systems at several business enterprises.

Object storage hierarchy management *by W. B. Harding, C. M. Clark, C. L. Gallo, and H. Tang, p. 384.* The Object Access Method (OAM) component of MVS/Data Facility Product is responsible for the storage, retrieval, and management of objects in IBM MVS/ESA™ ImagePlus™ systems and in other applications. The OAM Storage Management Component is the subcomponent of OAM that provides storage management for objects stored within an object storage hierarchy. Storage management is a cyclic procedure which assures that data are stored in conformity to a policy defined by the data processing storage administrator. During a storage management cycle, the OAM Storage Management Component (OSMC) selects objects for processing based on requirements for backup, expiration, or service level changes. This paper describes the concepts of object storage management using a storage hierarchy that contains DASD and optical disk storage.

ImagePlus Workstation Program *by G. B. Anderson, B. P. Gross, S. M. Lewis, and J. A. Reimer, p. 398.* IBM's ImagePlus™ system is designed to permit the capture, storage, management, and retrieval of documents through digital imaging. The ImagePlus Workstation Program is that portion of the ImagePlus system that controls the user's workstation. This paper describes the challenges that were posed in designing an ImagePlus Workstation, and the ap-

proach taken by the development group to solve them. The primary design goal was to deliver operational performance for image processing in a cost-effective workstation, while permitting the user maximal control in viewing and manipulating scanned documents. The solution chosen implemented a Personal System/2® workstation that operates with a System/36, Application System/400®, or System/370 Multiple Virtual Storage host.

Personal systems image application architecture: Lessons learned from the ImagEdit program *by A. Ryman, p. 408.* Image applications require complex processing on large amounts of data. The application designer is presented with difficult challenges that are exacerbated on personal systems which have limited processor speed and constrained memory. This paper discusses the problems relevant to personal systems image application architecture and how these problems were solved in the ImagEdit® program. A virtual array manager (VAM) consisting of a virtual memory manager (VMM) and an access scheduler was used to solve the data management problem. The VAM divided each image into segments and transferred them to the VMM for storage. These segments were swapped between memory and disk in response to a sequence of access requests, controlled by the access scheduler using performance-maximizing heuristics. Object-oriented design was used to address the functional complexity problem. The processing functions were divided into two classes. The data-stream class included scanning, printing, and filing, with each data-stream function decomposed into a series of demand-driven pipe objects. The editing class included cut and paste, textual and graphical annotation, and freehand drawing.

ImagePlus High Performance Transaction System *by R. F. Dinan, L. D. Painter, and R. R. Rodite, p. 421.* The need for a cost-effective method for handling, processing, storing, and retrieving transaction documents combined with the availability of hardware and software technologies capable of satisfying this need are the basis for the High Performance Transaction System discussed in this paper. The particular transactions that are the subject of this paper are bank checks, the volumes surpassing 50 billion per year and continuing to grow. Other transactions might be the handling of such documents as bill remittances, tax documents, mail-order forms, census forms, and many other similar applications. This paper discusses the system design, hardware and software architectures, and performance of the ImagePlus™ High Performance Transaction System.

Intelligent Forms Processing *by R. G. Casey and D. R. Ferguson, p. 435.* The automatic reading of optically scanned forms consists of two major components: extraction of the data image from the form and interpretation of the image as coded alphanumerics. The second component is also known as optical character recognition, or OCR. We have implemented a method for entry of a wide variety of forms that contain machine-printed data and that are often produced in business environments. The function, called Intelligent Forms Processing (IFP), accepts conventional forms that call for information to be printed in designated blank areas, but in which the information may exceed boundaries due to poor registration during printing. The human eye easily accommodates data that impinge on form boundaries or on background text; however, the same powers of discrimination applied to machine processing pose a technical challenge. The IFP system uses a setup phase to create a model of each form that is to be read. Scanned forms containing data are compared against the matching form model. Special algorithms are employed to extract data fields while removing background printing (e.g., form lines) intersecting the data. The extracted data images are interpreted by an OCR process that reads typical monospace fonts. New fonts may be added easily in a separate design mode. If the data are alphabetic, a lexicon may be assembled to define the possible entries.

AS/400 ImagePlus system view *by M. Addink and J. J. Mullen, p. 451.* Storage space, loss of documents, misfiled and misplaced documents, and document retrieval are just a few of the problems of processing large volumes of paper. High storage costs, lost productivity, personnel costs, and poor responsiveness are some of the results. Until recently, technology could not provide a cost-effective method for reducing this paper flow. IBM designed its midrange ImagePlus™ system to provide an effective and comprehensive solution to both the paper and the work-flow problems. The AS/400™ ImagePlus system provides an imaging environment for departmental installations of up to 256 workstations. This system combines the image-handling capabilities of the IBM ImagePlus Workstation Program, the user interface and case/folder processing of the AS/400 Workfolder Application Facility, with the image storage and retrieval capabilities of the optical storage support subsystem. All together, these facilities provide a high configurable system capable of handling high volumes of transactions.

Experience gained in implementing ImagePlus *by B. T. Perry, B. A. Wester, W. W. Baker, and J. F. Kemmis, p. 467.* An ImagePlus™ internal use program (IUP) was established in IBM to assist users inside the company with their initial application selection and their training, procedure definition, and prototype system installation, as well as their initial use of the system on a regular working basis. This paper discusses the experience gained from identifying, selecting, and preparing several areas within IBM for an ImagePlus system. The experience begins with establishment of IUP objectives and guidelines, continues through the account nomination and selection process used to identify and select the application, and ends with identification of the justification or business case process and some of the major elements considered in the justification for the purchase of a system. A detailed description of several internal operations that have installed an ImagePlus system

is given. In conclusion, ImagePlus as an application enabler is discussed.

Volume 29, Number 4, 1990

Multimedia presentation development using the Audio Visual Connection *by D. J. Moore, p. 494.* This paper describes the technology behind the creation of multimedia presentations using a new IBM program product, the Audio Visual Connection® (AVC™). The multimedia approach represents a major innovation in computer technology involving new concepts such as the digitization of audio and video, the involvement of the Musical Instrument Digital Interface (MIDI), and the creation of a multimedia story. Two hardware adapters support the AVC: the Audio Capture and Playback Adapter and the Video Capture Adapter. When used with these adapters, the AVC digitizes both stereo audio and color video, performs powerful edit and synchronization functions, defines an interactive user environment, and creates a multimedia presentation using standard IBM Personal System/2® hardware. Additional input is available from MIDI song files, screen capture, and non-AVC image systems. Final results range from passive presentations to interactive applications to sophisticated database front ends.

Business/enterprise modeling *by R. L. Katz, p. 509.* This paper reports on pertinent aspects of business/enterprise modeling studies that were conducted with nine IBM customers using what are now called computer-aided software engineering (CASE) tools. Coming shortly after the recent AD/Cycle™ announcement and the increased focus in IBM on tool-supported (CASE) business/enterprise modeling, this description of actual modeling studies should be especially germane. The model definitions (dimensions) used in the studies correspond exactly to many of the dimensions used by AD/Cycle, DevelopMate™, and the Repository Manager™. Compelling business reasons for conducting the studies are identified.

Exponentiation cryptosystems on the IBM PC *by P. G. Comba, p. 526.* Several cryptosystems based on exponentiation have been proposed in recent years. Some of these are of the public key variety and offer notable advantages in cryptographic key management, both for secret communication and for message authentication. The need for extensive arithmetic calculations with very large integers (hundreds of digits long) is a drawback of these systems.

This paper describes a set of experimental programs that were developed to demonstrate that exponentiation cryptosystems can be efficiently implemented on the IBM Personal Computer (PC). The programs are organized into four layers, comprising procedures for: multiple precision integer arithmetic, modular exponentiation, prime number generation and testing, and cryptographic key generation. The major emphasis of the paper is on methods and techniques for improving execution speed. The items discussed include: the use of a specialized squaring procedure; a recursive splitting method to speed up squaring and multiplication; the computation of residues by using multiplication instead of division; the efficient encoding of residue information; and the use of thresholds to select the most effective primality testing algorithm for a given size number. Timing results are presented and discussed. Finally, the paper discusses the advantages of a mixed system that combines the superior key management capabilities inherent in public key cryptosystems with the much higher bulk-encryption speed obtainable with the Data Encryption Algorithm.

Extension of the relational database semantic processing model *by T. Hirao, p. 539.* A data model consists of three parts: (1) a data definition that represents the information in an understandable manner; (2) a definition of the constraints that must hold for the information to be valid; and (3) a definition of operations that can be performed on the information. Current database management systems do not allow explicit specification of all three parts of the data model. This paper gives an approach that extends current database management systems through a technique called pre-precompilation.

Re-engineering software: A case study *by R. N. Britcher, p. 551.* In 1986, the Federal Aviation Administration formed a contract with three companies to re-engineer a major portion of the New York terminal approach control (TRACON) application software—the software that supports air traffic control in the New York City and Newark, New Jersey, area. This paper discusses the techniques used to successfully re-engineer the software to run on an IBM System/370™, illustrating that real-time software can be logically converted from one computer to another, reliably and cost-effectively.

Guidelines for authors of the IBM Systems Journal *by A. G. Davis, J. R. Friedman, M. J. Haims, and G. C. Stierhoff, p. 568.* It is important to have guidelines for the scope and breadth of papers being prepared for submission to a technical journal. This article provides information about the *IBM Systems Journal* and offers guidelines for prospective authors. The *Systems Journal* and its audience are described, and the processing of papers is discussed, along with suggestions for content and structure. To further aid the writer in preparing clear, complete papers of high quality, a bibliography of technical writing references is included.

Volume 30, Number 1, 1991

VM/ESA: A single system for centralized and distributed computing *by W. T. Fischofer, p. 4.* The rapid evolution of distributed and personal systems in recent years has not diminished the importance of centralized computing. Today, systems at all levels need to operate in networked configuration to allow users and applications to access and manipulate data from anywhere with full integrity and optimal per-

formance. Virtual Machine/Enterprise Systems Architecture™ (VM/ESA™) satisfies this requirement as a single VM product that has been designed for both centralized and distributed computing. This essay describes how VM/ESA builds on IBM's reputation for virtual machine performance, function, and flexibility to form an ideal solution base for the 1990s.

VM Data Spaces and ESA/XC facilities *by J. M. Gdaniec and J. P. Hennessy, p. 14.* Release 1.1 of the Virtual Machine/Enterprise Systems Architecture™ (VM/ESA™) operating system introduces a new function called VM Data Spaces, provided through a new virtual-machine architecture called Enterprise Systems Architecture/Extended Configuration (ESA/XC). ESA/XC is the strategic VM/ESA virtual-machine environment for Conversational Monitor System (CMS) users and service virtual machines requiring large amounts of storage or advanced data-sharing capabilities. ESA/XC includes all of the facilities of System/370 Extended Architecture (370-XA) that are used by CMS or server programs and is therefore upward compatible for CMS or server programs currently running in 370-XA virtual machines. To this 370-XA base, ESA/XC adds the data space and access-register addressing capabilities previously available only under the Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA™) operating system. These addressing extensions can be used to make additional storage available to large, storage-constrained applications and can also be used by servers as an efficient way of sharing data between service virtual machines and the users that access those servers. As an introduction to the VM Data Spaces function, this paper describes the ESA/XC virtual-machine architecture and presents an overview of the VM/ESA services provided in support of the ESA/XC architecture.

ESA/390 interpretive-execution architecture, foundation for VM/ESA *by D. L. Osisek, K. M. Jackson, and P. H. Gum, p. 34.* The interpretive-execution facility of Enterprise Systems Architecture/390™ (ESA/390™) provides an instruction for the execution of virtual machines. This instruction, called START INTERPRETIVE EXECUTION (SIE), was initially created for virtualizing either System/370™ or 370-XA architectures, and was used later for virtualizing ESA/370™ and ESA/390 architectures. SIE has evolved to provide capabilities for a number of specialized performance environments. Most recently it provides for the unique requirements of Enterprise Systems Architecture/Extended Configuration (ESA/XC) virtual-machine architecture. This comprehensive set of capabilities in the architecture serves as the platform for the ability of VM/ESA™ to provide functions in virtual machines for end users and system servers. This paper describes the evolution of SIE and outlines use of the various capabilities in VM/ESA.

VM/ESA CMS Shared File System *by R. L. Stone, T. S. Nettleship, and J. Curtiss, p. 52.* Discussed is work toward satisfying requirements on the Conversational Monitor System (CMS) in the areas of data sharing and physical DASD space sharing. This work advances the present CMS file system design that allows only active read sharing among users on a single VM system, where each user has a reserved, private allocation of DASD space for file data. Described in this paper is the CMS Shared File System (SFS), which was designed to satisfy the data sharing and physical DASD space sharing requirements by providing a pool of DASD space that is shared among multiple users. DASD space assigned to the pool is easily extended, and read/write sharing of individual files is allowed. Also discussed is SFS security, usage of Virtual Machine/Enterprise Systems Architecture™ (VM/ESA™) data spaces for single system performance, and coordinated resource recovery to provide file data integrity in the distributed environment.

Coordinated Resource Recovery in VM/ESA *by B. A. Maslak, J. M. Showalter, and T. J. Szczygielski, p. 72.* A system service for coordinated recovery of resources is a critical function needed for distributed processing environments because applications need to provide for data integrity while the location of the data and processes are transparent to the application. VM is the first IBM operating system to provide Coordinated Resource Recovery as a system service rather than as a service provided by unique environments running on the operating system, and the VM Common Programming Interface-Communications and Shared File System are the first subsystems to utilize the service. This paper is an overview of why and how VM provided Coordinated Resource Recovery (CRR). CRR is the implementation of the Systems Application Architecture™ (SAA™) resource recovery interface within Virtual Machine/Enterprise Systems Architecture™ (VM/ESA™). This coordinated sync point system service allows one or more applications or subsystems to update multiple resources and to request that all updates be committed or backed out together. The applications and their respective resources can be local or distributed. CRR either coordinates the request to commit or backout immediately, or supports automatic resource resynchronization in case a system or subsystem fails. When restart is not possible, CRR allows for intervention by a system operator or administrator.

Systems management for Coordinated Resource Recovery *by R. B. Bennett, W. J. Bitner, M. A. Musa, and M. K. Ainsworth, p. 90.* Coordinated Resource Recovery is a Virtual Machine/Enterprise Systems Architecture™ (VM/ESA™) function for providing consistency of changes to multiple resources in environments that include distributed applications. It provides a uniform solution for applications to the problem of resource consistency. Systems management of Coordinated Resource Recovery in VM/ESA (CRR) is the set of system services and interfaces that

support both automatic and manual procedures for managing CRR installation, performance, and recovery, as well as resource manager and application participation. Much of systems management is focused on application recovery from occasional failures of the procedures for coordinating consistent resource changes. This paper describes several key aspects of CRR systems management, including the CRR recovery log, facilities for minimizing manual intervention when failures occur, performance considerations, and application participation in recovery.

VM/ESA support for coordinated recovery of files *by C. C. Barnes, A. Coleman, J. M. Showalter, and M. L. Walker, p. 107.* This paper discusses the concepts and facilities of the Shared File System (SFS) support for Virtual Machine/Enterprise Systems Architecture™ (VM/ESA™) Coordinated Resource Recovery (CRR). It includes background information on limitations that lead to SFS support for coordination of file recovery functions. The level of support provided by the Virtual Machine/System Product (VM/SP) Release 6 SFS support is identified and contrasted with the support provided in VM/ESA. The paper contains an overview of the system structure and the rationale for the support and is a discussion from the overall perspective of the total system environment and system processing for resource recovery. After the concepts and structure of VM/ESA SFS support are introduced, the paper discusses the specific technology involved in providing SFS support for Coordinated Resource Recovery. This includes a discussion of specific facilities used by SFS and how SFS deals with certain conditions that can arise. In addition, this paper discusses the Conversational Monitor System (CMS) compatibility considerations that contributed to the design of SFS support for Coordinated Resource Recovery. This includes compatibility with prior releases and compatibility with the CMS file system support for minidisks. Finally, some of the future directions for file system support of resource recovery are identified along with some of the challenges that remain to be solved.

Volume 30, Number 2, 1991

Common Cryptographic Architecture Cryptographic Application Programming Interface *by D. B. Johnson, G. M. Dolan, M. J. Kelly, A. V. Le, and S. M. Matyas, p. 130.* Cryptography is considered by many users to be a complicated subject. An architecture for a cryptographic application programming interface simplifies customer use of cryptographic services by helping to ensure compliance with national and international standards and by providing intuitive high-level services that may be implemented on a broad range of operating systems and underlying hardware. This paper gives an overview of the design rationale of the recently announced Common Cryptographic Architecture Cryptographic Application Programming Interface and gives typical application scenarios showing methods of using the services described in the architecture to meet security requirements.

Key handling with control vectors *by S. M. Matyas, p. 151.* A method is presented for controlling cryptographic key usage based on control vectors. Each cryptographic key has an associated control vector that defines the permitted uses of the key within the cryptographic system. At key generation, the control vector is cryptographically coupled to the key via a special encryption process. Each encrypted key and control vector is stored and distributed within the cryptographic system as a single token. Decryption of a key requires respecification of the control vector. As part of the decryption process, the cryptographic hardware also verifies that the requested use of the key is authorized by the control vector. This paper focuses mainly on the use of control vectors in cryptosystems based on the Data Encryption Algorithm.

A key-management scheme based on control vectors *by S. M. Matyas, A. V. Le, and D. G. Abraham, p. 175.* This paper presents a cryptographic key-management scheme based on control vectors. This is a new concept that permits cryptographic keys belonging to a cryptographic system to be easily, securely, and efficiently controlled. The new key-management scheme—built on the cryptographic architecture and key management implemented in a prior set of IBM cryptographic products—has been implemented in the newly announced IBM Transaction Security System.

ESA/390 Integrated Cryptographic Facility: An overview *by P. C. Yeh and R. M. Smith, Sr., p. 192.* This paper reviews the objectives of the Enterprise Systems Architecture/390™ (ESA/390™) Integrated Cryptographic Facility. It presents the cryptographic key-management scheme, summarizes key elements and unique characteristics of the facility, and describes the physical security provided by the first ESA/390 implementation.

Transaction Security System *by D. G. Abraham, G. M. Dolan, G. P. Double, and J. V. Stevens, p. 206.* Components of previous security systems were designed independently from one another and were often difficult to integrate. Described is the recently available IBM Transaction Security System. It implements the Common Cryptographic Architecture and offers a comprehensive set of security products that allow users to implement end-to-end secure systems with IBM components. The system includes a mainframe host-attached Network Security Processor, high-performance encryption adapters for the IBM Personal Computer and Personal System/2® Micro Channel®, an RS-232 attached Security Interface Unit, and a credit-card size state-of-the-art Personal Security™ card containing a high-performance microprocessor. The application programming interface provides common programming in the host and the workstation and supports all of the Systems Application Architecture™ lan-

guages except REXX and RPG. Applications may be written to run on Multiple Virtual Storage (MVS) and PC DOS operating systems.

Transaction Security System extensions to the Common Cryptographic Architecture *by D. B. Johnson and G. M. Dolan, p. 230.* A well-designed application program interface for a line of cryptographic products simplifies customer use of cryptographic services by helping to ensure compliance with national and international standards and by providing intuitive high-level services that may be implemented on disparate systems. The Common Cryptographic Architecture is IBM's strategic cryptographic architecture. The Transaction Security System implements the Common Cryptographic Architecture in full. Furthermore, the Transaction Security System has implemented extensions to the architecture to address additional customer requirements. This paper gives the design rationale for some of the additional cryptographic functionality in the Transaction Security System beyond that mandated by the Common Cryptographic Architecture.

Volume 30, Number 3, 1991

SNA route generation using traffic patterns *by S. C. Baade, p. 250.* This paper describes a procedure used by the IBM Information Network to generate optimum routes for a complex Systems Network Architecture (SNA) network by utilizing communication traffic patterns. The Route Table Generator and an understanding of customer locations and available facilities had been the basis for route generation. However, this approach became overwhelming as the network grew. The lack of flexibility required an increasing need to manually override generated routes. The resulting approach could not ensure that network delay had been minimized. The Network Design and Analysis (NETDA) tool developed at the IBM Yorktown Research Center was used as a solution. NETDA orders routes based on static indicators such as number of hops, route distance, and speed of the path components. However, NETDA also selects optimal routes based on network traffic patterns. Traffic data were easily incorporated into NETDA, and the IBM Information Network has optimized its SNA routing using NETDA and actual traffic data. The process was challenging because of the number of network components involved and the difficulty in obtaining portions of the traffic data. The use of NETDA for route generation is discussed, and the data collection methodology is described. Network component utilization and network delay changes are reviewed as a means of showing the benefits of such optimizations.

A base for portable communications software *by S. H. Goldberg and J. A. Mouton, Jr., p. 259.* The emerging international standards for interconnecting computers will be important in IBM's future plans. The Open Systems Interconnection (OSI) protocols are already part of IBM's Systems Application Architecture® (SAA™), implying that they will be implemented across the dissimilar SAA operating systems. Building these complex OSI protocols is costly, and additional expense is involved in verifying conformance and interoperation with other systems. "Porting" a common implementation of these protocols to all SAA systems offers major cost savings, but the difference between systems and the need for high-performance, robust implementations poses problems. The OSI/Communications Subsystem Base solves many of these problems in a general way that may apply to other layered protocols and other systems. The Base provides all necessary operating system services to support the layered communications protocol machines of OSI and allows access to the I/O services of the native operating system as required. This paper discusses the sophisticated communications-oriented environment provided by the OSI/Communications Subsystem Base, which includes multiple threads, back-pressure flow control, resource monitoring, layer modularity, and steps to minimize process switches and data copying. The paper is addressed primarily to systems engineers and communications architects interested in OSI and portability in general.

Perspectives on multimedia systems in education *by S. Reisman and W. A. Carr, p. 280.* Although multimedia or interactive video seemed revolutionary in the early and mid-1980s, its application to individualized instruction followed clearly defined evolutionary paths. Forms of individualized instruction leading up to multimedia instruction are described, and a review of the integration of individualized instruction into a standard education curriculum is included. Discussed is a jointly defined effort between IBM and the California State University at Fullerton that demonstrated: (1) the benefits of a parallel course-development and course-implementation approach, (2) the superiority of multimedia over traditional instruction in the subject area tested, and (3) very low-cost development of quality multimedia courses. A projection by IBM for its own internal education program indicates that by the year 2000 not only will individualized instruction become fully integrated into IBM's education curriculum, but it will become the dominant approach, encompassing within it many aspects of traditional instruction. The continuing integration of individualized instruction with other technologies and advances in digital full-motion capabilities can help make multimedia instruction not only independent of time and place, but more engrossing and enjoyable as well.

FORTRAN for clusters of IBM ES/3090 multiprocessors *by R. J. Sahulka, E. C. Plachy, L. J. Scarborough, R. G. Scarborough, and S. W. White, p. 296.* IBM Clustered FORTRAN is a combination of software and hardware that allows two IBM Enterprise System/3090™ (ES/3090™) multiprocessors to be physically connected as a cluster and allows FORTRAN jobs to execute in parallel across all of the processors of the cluster. The FORTRAN compiler and library provided as part of Clustered FORTRAN are used for writing and executing the

parallel programs in this hybrid environment of distributed and shared-memory systems. The compiler provides language extensions for explicit programming in parallel, as well as the ability to automatically generate both parallel and vector code. The Clustered FORTRAN language allows users to write parallel applications that are independent of the machine configuration and operating system. This paper describes the execution environment, compiler, and library, gives some variations of programming matrix multiplication, and shows that performance of one GigaFLOPS can be achieved using Clustered FORTRAN.

Partial compilation of REXX *by R. Y. Pinter, P. Vortman, and Z. Weiss, p. 312.* A comprehensive set of compilation techniques for coping with various dynamic features of the REXX programming language are described. Among them are a novel symbol table structure, a multiple representation method for type-free objects, and a number of run-time acceleration techniques. Most of the work can be unified under the general principle of delayed execution, which is applicable in other situations as well. Significant performance gains were observed in an experimental setting, and these results led to the decision to develop IBM's recently announced REXX compiler product.

A C programming model for OS/2 device drivers *by D. T. Feriozi, p. 322.* The recent growth in the number of new and different types of devices for use with personal computers has challenged software engineers to plan new and better ways of developing software to run the devices. For Operating System/2® (OS/2®) device drivers, an improvement would be to code in a high-level language rather than to use assembly language. A practical and proven method of writing OS/2 device drivers in the C programming language is presented here. The C language was chosen because of its documented suitability as a systems programming language and because of its universal availability for use on small systems.

A knowledge-based system for MVS dump analysis *by N. G. Lenz and S. F. L. Saelens, p. 336.* A new domain in software problem determination can be automated by means of this knowledge-based system. The system imitates a human problem solver by using the same tools and the same diagnostic approach as the experts use, including the processing of human-readable data. The application is fully integrated within the target Multiple Virtual Storage (MVS) operating system to ensure user acceptance. A large variety of knowledge is contained in the system, ranging from pattern-recognition knowledge to basic MVS knowledge and problem-solving strategies. The diagnostic approach is based on a model of software problem situations and on diagnostic reasoning methods adopted from the medical application domain. The goal of the project was to solve a significant part of the problem resolution process automatically, rather than to build yet another tool for use in software problem determination. This system is a first step to further automation in this area.

Modeling and software development quality *by S. H. Kan, p. 351.* This paper summarizes the models used by a large software development organization for estimating software reliability and managing software development quality. The role of modeling in software quality improvement is illustrated. Implementation of the models, reliability of the estimates, predictive validity, and the nature of data are discussed.

Integrated hypertext and program understanding tools *by P. Brown, p. 363.* This paper describes some concepts and issues related to software tools integration. Questions regarding data integration and functional integration between tools are identified and discussed. Some techniques for handling large volumes of data are briefly described. A prototype tool is described in which hypertext links are automatically created between program analysis data and hypertext documentation. With this tool, end users can freely move between source code views and related documentation. A common annotation feature lets software developers and information developers share information and synchronize maintenance activities in a single tools environment.

Technical note—The WATINFO face server and associated utilities *by A. Appel, G. A. Cuomo, E. A. Overly, J. A. Walicki, and R. E. Yozzo, p. 393.* WATINFO is a TCP/IP server that operates at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, to provide VM-based information about services and personnel at the IBM Yorktown Research site. Client programs have been written for Operating System/2® (OS/2®) and Advanced Interactive Executive™ (AIX®). An advanced feature of WATINFO is the display of images, when appropriate, in combination with other data. The images are supported by a face server, which fetches image data from a library of face images. The format of the face images is small—256 by 256 by one bit—and is transmitted rapidly to a variety of displays. The face library can be used for other purposes: inclusion of images in documents, the preparation of image labels, and the construction of an image telephone book.

Volume 30, Number 4, 1991

The IBM family of APL systems *by A. D. Falkoff, p. 416.* The developmental history of IBM subfamilies of APL systems is traced in this paper, focusing on the inter-relationships among them and the methods of implementation used by the various groups involved. The language itself, and the way its evolution was managed, are also considered as factors influencing the development process. A chart is included that illustrates the evolution of mainframe and small machine programming products supporting APL, beginning in 1964 up to the present time.

APL2: Getting started *by J. A. Brown and H. P. Crowder, p. 433.* APL is a concise and economical notation for expressing computational algorithms and procedures. This paper introduces the main ideas of APL2, an IBM implementation of APL, and illustrates the programming style with some graphical examples.

Extending the domain of APL *by M. T. Wheatley, p. 446.* This paper explores connectivity mechanisms between APL and other languages and applications available on a modern computer system. The design, implementation, and application of APL facilities such as shared variables, auxiliary processors, external names, file subsystems, and namespaces, as they are implemented in IBM's APL2 product, are discussed and compared.

Storage management in IBM APL systems *by R. Trimble, p. 456.* APL systems have traditionally used specialized storage management schemes that avoid storage fragmentation by "garbage collection," moving live data as needed to collect unused storage into a single area. This was very effective on systems with a small amount of real storage addressed directly. It has become less effective on today's systems with virtual addressing and large amounts of virtual storage. Both traditional schemes of storage management and a recently implemented replacement for them are described. The focus is on implementations for IBM mainframe hardware.

Putting a new face on APL2 *by J. R. Jensen and K. A. Beaty, p. 469.* APL2/X is an interface between APL2 and the X Window System®, built at the IBM Cambridge Scientific Center. This interface enables the full set of the X Window System Xlib calls and the related data structures to be used directly from programs written in APL2, thereby providing APL2 with a true, full-function windowing environment. The interface also deals with the broader and more general issue of how to call C programs from APL2. The interface and the experience of building it are described in some detail in this paper.

The APL IL Interpreter Generator *by M. Alfonseca, D. Selby, and R. Wilks, p. 490.* The objective of the APL IL Interpreter Generator is to solve the problem of creating APL interpreters for different machines at a minimum cost. The objective has been accomplished by writing an APL interpreter in a specially designed programming language (IL) that has very low semantics but high-level syntax. The interpreter is translated to each target machine language by easily built compilers that produce high-performance code. The paper describes IL, the APL interpreters written in IL, and the final systems generated for seven different target machines and operating systems. Some of these systems have been generated in an extremely short time.

Parallel expression in the APL2 language *by R. G. Willhoft, p. 498.* This paper reports on an investigation of parallel expression and execution in the current APL2 language. The study covers a historical, theoretical, and empirical viewpoint. The parallel nature of APL is traced from its foundations in the Iverson notation to current problems in executing APL on parallel hardware. The paper discusses features of the APL language and its current implementations that limit taking advantage of parallel expressions. A survey of related topics from the work on APL compilers is also included. Each APL2 language construct is examined for potential parallel expression. The operations are grouped based on the possible parallelism exhibited by each operation, and the possible implementation of each group is discussed. Three APL2 applications are explored to determine the actual parallelism expressed in "real" APL2 code. These applications are chosen from distinct areas: graphics, database systems, and user interactive systems. The actual data passed as arguments to every operation are dynamically examined, and the information is collected for analysis. The data are summarized and results of the study are discussed.

The foundations of suitability of APL2 for music *by Stanley Jordan and Erik S. Friis, p. 513.* APL is commonly used in scientific and quantitative applications, such as engineering and finance, but there has been little acceptance so far in artistic and symbolic applications, such as music. This paper demonstrates the suitability of APL2, a dialect of APL, as a powerful tool for the building of music-oriented software. The interactive interpreter, flexible built-in primitive functions and operators, and the independence from the details of the hardware are attractive features for music programmers. With APL2, a user can interactively create and transform complex informational structures. Thus, it is not only a formidable language for implementing music software, but also a valuable notation for representing the music itself.

Verification of the IBM RISC System/6000 by a dynamic biased pseudo-random test program generator *by A. Aharon, A. Bar-David, B. Dorfman, E. Gofman, M. Leibowitz, and V. Schwartzburd, p. 527.* Verification of a computer that implements a new architecture is especially difficult since no approved functional test cases are available. The logic design of the IBM RISC System/6000™ was verified mainly by a specially developed random test program operator (RTPG), which was used from the early stages of the design until its successful completion. APL was chosen for the RISC System/6000 RTPG implementation after considering the suitability of this programming language for modeling computer architectures, the very tight schedule, and the highly changeable environment in which RTPG would operate.

APL2 as a specification language for statistics *by N. D. Thomson, p. 539.* APL has had a dedicated following for many years among some sections of the academic and industrial statistical communities. One of its greatest strengths is its value as a specification language. Not only can algorithms be described consistently and unambiguously, but also, given an

appropriate interpreter, the specifications can be immediately executed. A group of academic and industrial statisticians in the United Kingdom recognized these capabilities and embarked on a project called ASL (APL Statistics Library) with the support of the British APL Association. ASL aims to provide a collection of coherent APL functions for widely used statistical calculations, thereby creating standards for the unambiguous expression of statistical algorithms. A natural consequence of this is that discussions of more complex algorithms and methods can occur without the need to revisit and redefine basic functions and the ways in which they interpret data.

Advanced applications of APL: logic programming, neural networks, and hypertext *by M. Alfonseca, p. 543.* This paper reviews the work of the author on the application of the APL and APL2 programming languages to logic programming, emulation of neural networks, and the programming of hypertext applications.

Language as an intellectual tool: From hieroglyphics to APL *by D. B. McIntyre, p. 554.* We learn elementary mathematics before understanding the source of its symbols and procedures, which therefore appear, incorrectly, to have been decreed ready-made. Language and reason are intimately related, and the embodiment of an idea in a symbol may be essential to its comprehension. APL unifies algebra into a single consistent notation; it allows us to exploit the powerful concepts of functions and operators; and it helps us to escape from the tyranny of scalars by giving us the tools to think in terms of arrays, or multiple quantity, as J. J. Sylvester so eloquently urged us to do a century ago. APL has an intellectual consistency that is a source of satisfaction and pleasure. This paper traces the history of symbols from hieroglyphics to APL.

A personal view of APL *by K. E. Iverson, p. 582.* This essay portrays a personal view of the development of several influential dialects of APL: APL2 and J. The discussion traces the evolution of the treatment of arrays, functions, and operators, as well as function definition, grammar, terminology, and spelling.

Volume 31, Number 1, 1992 *G321-0106*

Introduction to the IBM Optimization Subroutine Library *by D. G. Wilson and B. D. Rudin, p. 4.* This essay introduces the IBM Optimization Subroutine Library (OSL) and seven OSL-related papers that appear in this issue. Developed as a result of a partnership between several IBM research and development groups, OSL provides a suite of tools for manipulating the models and solving the resulting minimization and maximization problems of mathematical optimization. The problems that OSL addresses include: linear, quadratic, mixed-integer, and pure network programming problems. OSL includes solvers based on the classical simplex method and on newer interior point methods. Because a user-supplied driver program coordinates the problem solution, and because of the "mix and match" philosophy of OSL, a user may, within rather wide limits, individually tailor a technique to solve a particular problem. We conclude that OSL is something new in optimization software.

Implementing the simplex method for the Optimization Subroutine Library *by J. J. H. Forrest and J. A. Tomlin, p. 11.* In this paper we describe the simplex algorithm and briefly discuss the interaction of the detailed implementation of the algorithm with the changes in computer hardware over the last 30 years. Then we give one example of the design changes needed to implement the method efficiently for the IBM 3090™ vector architecture. For the later RISC System/6000™ implementation, it was necessary to rethink this yet again. Finally we discuss the issue of robustness and the steps that were taken to give maximum reliability in the simplex algorithm in the IBM Optimization Subroutine Library.

Implementing interior point linear programming methods in the Optimization Subroutine Library *by J. J. H. Forrest and J. A. Tomlin, p. 26.* This paper discusses the implementation of interior point (barrier) methods for linear programming within the framework of the IBM Optimization Subroutine Library. This class of methods uses quite different computational kernels than the traditional simplex method. In particular, the matrices we must deal with are symmetric and, although still sparse, are considerably denser than those assumed in simplex implementations. Severe rank deficiency must also be accommodated, making it difficult to use off-the-shelf library routines. These features have particular implications for the exploitation of the newer IBM machine architectural features. In particular, interior methods can benefit greatly from use of vector architectures on the IBM 3090™ series computers and "super-scalar" processing on the RISC System/6000™ series.

A decomposition method for quadratic programming *by D. L. Jensen and A. J. King, p. 39.* We discuss the algorithms used in the Optimization Subroutine Library for the solution of convex quadratic programming problems. The basic simplex algorithm for convex quadratic programming is described. We then show how the simplex method for linear programming can be used in a decomposition crash procedure to obtain a good initial basic solution for the quadratic programming algorithm. We show how this solution may be used as a starting solution for the simplex-based algorithm.

Besides its ability to obtain good starting solutions, this procedure has several additional properties. It can be used directly to find an optimal solution to a quadratic program instead of simply finding a good initial solution; it provides both upper and lower bounds on the objective function value as the algorithm proceeds; it reduces the complexity of inter-

mediate calculations; it avoids certain numerical difficulties that arise in quadratic, but not linear programming.

A systematic approach to OSL application programming *by A. S. Minkoff, p. 49.* The Optimization Subroutine Library (OSL) provides powerful tools for solving mathematical programming problems, and permits the integration of these tools into larger applications. In order to access the computational power, an application must translate data between forms used in the rest of the application and the form in which the data can be manipulated by OSL. OSL does not currently offer tools to aid in this translation. The purpose of this paper is to provide a systematic approach for translating symbolic representations of mathematical programming problems into computer code that performs all necessary interactions with both OSL and the rest of the application.

Frontier: A graphical interface for portfolio optimization in a piecewise linear-quadratic risk framework *by D. L. Jensen and A. J. King, p. 62.* "Frontier" is a pilot graphical user interface for portfolio optimization built for the new IBM workstation, the RISC System/6000™, out of basic X-windows and OSL utilities. The program asks the user to select a piecewise linear-quadratic risk measure, draws a risk/reward efficient frontier, and permits the user to examine the efficient frontier using zoom and histogram display facilities. This paper describes the interfaces and discusses possible extensions.

A global approach to crew-pairing optimization *by R. Anbil, R. Tanga, and E. L. Johnson, p. 71.* The problem addressed in this paper is crew-pairing optimization in airline flight planning: finding tours of duty (pairings) that are legal and cover every flight leg at the least cost. The legal rules and cost of a pairing are determined by complex Federal Aviation Agency and contractual requirements. In addition, the problem is made more difficult by the hub-and-spoke system used by airlines that multiplies the possible ways a pairing can link flight legs. The state-of-the-art crew-pairing TRIP system of American Airlines uses subproblem optimization and, as is true for other crew-scheduling systems, may not be able to improve a solution even though a better one exists. We report on the methodology developed during a joint study by IBM and American Airlines Decision Technologies to use the IBM Optimization Subroutine Library in conjunction with TRIP to improve on crew-pairing solutions by taking a global approach. The resulting improvements have been a reduction of 5 to 11 percent in excess crew cost. Estimated total savings are five million dollars per year.

Recent developments and future directions in mathematical programming *by E. L. Johnson and G. L. Nemhauser, p. 79.* Recent advances in mathematical programming methodology have included: development of interior methods competing with the simplex method, improved simplex codes, vastly improved performance for mixed-integer programming using strong linear programming formulations, and a renewed interest in decomposition. In addition, use of vector and parallel processing has improved performance and influenced algorithmic developments. Application areas have been expanding from the traditional refinery planning and distribution models to include finance, scheduling, manufacturing, manpower planning, and many others. We see the acceleration of better methods and improved codes moving together with faster, lower-cost, and more interesting hardware into a variety of application areas, thereby opening up new demands for greater function of optimization codes. These new functions might include, for example, more powerful nonlinear codes, decomposition techniques taking advantage of network and other problem-dependent structures, and mixed-integer capability in quadratic and general nonlinear problems. Stochastic scenario programming and multitime-period problems are becoming solvable and open up applications and algorithmic challenges. The IBM Optimization Subroutine Library has helped to accelerate these changes but will have to continue to change and expand in ways that are touched upon in this paper.

Customized systems for engineering applications *by Y. Hazony and L. Zeidner, p. 94.* An APL™-based high-productivity software-development environment is shown to enable small teams of two or three persons to build complex engineering software systems. The productivity and flexibility of such small teams, equipped with this environment, enables them to build customized engineering application systems economically. These customized systems are far more useful for the particular applications they address than are the generic systems that are commonly produced by large software-development groups. A customized engineering application system is described, illustrating the productivity of the two APL2-based computer-aided software engineering (CASE) tools used for its implementation and long-term software maintenance. The system is presented in some detail, to demonstrate its sophistication and thus provide a measure of the productivity of the software-development environment. The two CASE tools that comprise this software-development environment are used to build interactive graphical application systems, and to build systems for applications that require or can benefit from distributed cooperative processing. A list of some customized application systems built using the described environment is provided, along with estimates of the implementation efforts. The features of APL2 that play a key role in the effectiveness of these tools are also discussed.

A split model for OS/2 SCSI device drivers *by D. T. Feriozi, p. 114.* The concept of splitting one logical device driver into two or more physical units is presented. The specific case of an Operating System/2® (OS/2®) SCSI (Small Computer System Interface) device driver is used as an example. The

primary reason for splitting the device driver is to reduce the development effort required to produce new SCSI device drivers. Common code is isolated in a separate driver in order to prevent its reinvention as each new SCSI device becomes available. Additional benefits are that the overall device driver size is reduced, and the performance of the SCSI subsystem is enhanced. The complete separation of the upper- and lower-level drivers provides the ability to replace one of the device drivers without affecting any of the other components of the system. This is particularly important because it enables backward compatibility for older device drivers, while allowing for the support of emerging technology.

Role of the DASD storage control in an Enterprise Systems Connection environment *by C. P. Grossman, p. 123.* This paper compares the Enterprise Systems Connection Architecture™ (ESCON™), with its use of fiber optic cables, to the parallel channel architecture introduced with System/360™. It also describes many of the reasons for the introduction of ESCON. The ESCON implementation for the IBM 3990 Storage Control is described in some detail, including a description of nonsynchronous operation. The paper concludes with a discussion of some of the benefits of ESCON for 3990 installation, performance considerations, and migration considerations.

Volume 31, Number 2, 1992 *G321-0107*

IBM network management strategy *by M. M. Szabat and G. E. Meyer, p. 154.* This essay describes some major directions at IBM with respect to network management. It describes the network management environment and discusses four key initiatives of IBM's SystemView™ network management strategy. This strategy provides integrated applications and services, broadens the scope of network management products, provides open access to multivendor products, and delivers and supports complete solutions for customers.

Evolution of an open communications architecture *by R. J. Cypser, p. 161.* An overview of the current IBM communications paradigm for interconnecting computer networks is presented. Emphasis is on the incorporation of multiprotocol, multivendor facilities in an integrated architecture. This paper presents an overview of key elements of the evolving communications architecture.

Management of multivendor networks *by J. G. Stevenson, p. 189.* Technical advances in multivendor network management capabilities allow customers to effectively manage their networks. Packaged offerings such as NetView® Extra simplify the ability to take advantage of these new capabilities. This paper describes the multivendor environment, customer network management requirements, IBM's initial approach to responding to these requirements, and enhancements needed to provide additional management offerings that automatically handle failures, including detection, bypass and recovery, vendor notification, and restoration of the repaired resource into service.

Network and system automation and remote system operation *by B. W. Irlbeck, p. 206.* The rapid growth in the size and complexity of today's information system networks highlights the importance of automation and remote system operation in managing these networks. Version 2 Release 2 of the NetView® program provides improved facilities in these two areas to assist enterprises in consolidating their operations staff in central locations and managing their information systems more efficiently and reliably. This paper describes the Systems Network Architecture (SNA) framework for remote system operation and the NetView remote operations platform, including the LU 6.2 data transport mechanisms and related management services applications. Extensions to the NetView automation platform that permit automation based directly on the receipt of SNA alerts or other structured data are described. In addition, enhancements that improve the performance, usability, and functional capabilities of the NetView automation table are discussed.

NetView Version 2 Release 3 Graphic Monitor Facility: Network management graphics support for the 1990s *by K. D. Gottschalk, p. 223.* The NetView® Version 2 Release 3 Graphic Monitor Facility provides an Operating System/2® (OS/2®) workstation-based graphics user interface for NetView that permits an operator to view graphically and control via generic commands both Systems Network Architecture (SNA) and non-SNA networks. This paper describes the major new capabilities of the NetView V2R3 Graphic Monitor Facility, discusses its structure at a high level, and describes in some detail the new support in this release for graphic views of various types of networks, simplified commands for controlling these networks, and the manner in which non-SNA networks are supported.

RODM: A control information base *by A. J. Finkel and S. B. Calo, p. 252.* Operational management of computers and computer networks was formerly performed exclusively by an operator or a team of operators equipped only with consoles for the display of status messages. Each system component independently determined its own set of such messages, identifying conditions needing attention. To meet future challenges, however, a structured approach to systems and network management and associated automation will be necessary. The amount and complexity of the status information needed for control and coordination will make it unlikely that operators will be able to keep up with such needs unaided. This control information must be made available to a family of systems and network management applications (including operator display programs). The NetView® Resource Object Data Manager (RODM) is designed to facilitate the storage and retrieval of control information. It provides services for defining a structured data model of a computer system. The control information is not kept

simply in the form of messages, but instead the data are organized into units called objects. This allows the model to effectively capture interrelationships and dependencies as well as status information.

AIX NetView/6000 *by J. H. Chou, C. R. Buckman, T. Hemp, A. Himwich, and F. Niemi, p. 270.* AIX® NetView®/6000 is a network management system that manages simple network management protocol (SNMP) devices developed by IBM and other vendors. It provides configuration, fault, and performance applications integrated into an advanced end-user interface (EUI), which incorporates a graphic display of network topology and performance as well as system management functions accessible from both graphic and character-based devices. An application builder and event configurator allow users to generate performance applications and provide automation of management tasks specific to their networks. In addition to providing stand-alone distributed network management, AIX NetView/6000 also provides a bidirectional connection to IBM's mainframe-based NetView product to enable central management of the enterprise network from System/370™ and System/390™ NetView.

Managing session performance using the NetView Performance Monitor *by L. Temoshenko, p. 286.* Managing the performance of network devices and their interaction with host applications is a complex task that entails the collection and reduction of information related to the underlying session. Depending on the specific management task at hand, differing types, correlations, and formattings of session performance measurements are required. The NetView™ Performance Monitor has a flexible set of facilities that can be used to provide the information needed to manage session performance. Its facilities to collect, correlate, and present session performance measurement are discussed in relation to typical network management tasks.

Estimating the fault rate function *by T. Jennings, p. 300.* Paging activity can be a major factor in determining whether a software workload will run on a given computer system. A program's paging behavior is difficult to predict because it depends not only on the workload processed by the program, but also on the level of storage contention of the processor. A program's fault rate function relates storage allocation to the page fault rate experienced while processing a given workload. Thus, with the workload defined, the fault rate function can be used to see how the program's storage allocation is affected by varying levels of storage contention, represented by varying fault rates. This paper presents a technique to represent program workloads and estimate the fault rate function, and describes how these results can be used in analyzing program performance.

Architectural directions for opening IBM networks: The case of OSI *by P. Janson, R. Molva, and S. Zatti, p. 313.* This paper discusses the results of a research project that developed an architectural framework for integrating non-IBM network architectures to the reference model and node structures of IBM's Systems Network Architecture (SNA). The unique features of the selected integration approach allow multiple protocol stacks to coexist and interoperate within the same computer, to share use of common physical network ports, links, and switching nodes, and to be accessed and managed through homogeneous interfaces. The architectural framework was developed for the specific purpose of integrating the Open Systems Interconnection (OSI) Reference Model to that of SNA, but its basic philosophy and key aspects turn out to be generally applicable to the integration of other network technologies as well, such as TCP/IP or NetBIOS.

SNA Management Services architecture for APPN networks *by M. O. Allen and S. L. Benedict, p. 336.* The introduction of Advanced Peer-to-Peer Networking (APPN) provides for a more flexible Systems Network Architecture (SNA) environment: end-user systems (physical units, or PUs) at the edge of a routing network no longer need a predefined relationship with a system services control point (SSCP) for network control purposes. This new flexibility creates challenges for SNA Management Services, however, since the SSCP-PU relationship provided a vehicle for network management as well as network control. To meet the needs of this peer-to-peer environment, the SNA Management Services architecture was extended to provide a management infrastructure that replaces the previous SSCP-PU relationship, and at the same time provides for much greater flexibility. This new infrastructure consists of a formalization of the focal-point/entry-point concept in the architecture and a transport technique for management services data that utilizes the facilities of Advanced Program-to-Program Communications (APPC) rather than the SSCP-PU session. Together this provides for a management structure in a peer network.

Naming and registration for IBM distributed systems *by S. Zatti, J. Ashfield, J. Baker, and E. Miller, p. 353.* Today's trends toward interconnection of networks expose limitations and deficiencies of traditional identification schemes. The need arises for a uniform naming solution that can accommodate the size and heterogeneity of worldwide domains, while still remaining understandable, usable, and manageable by human users.

This paper describes a proposal to name objects and resources in distributed environments; each object or resource can be located, accessed, communicated with, operated on, managed, or secured using the same, unique name. The solution proposed here includes registration mechanisms necessary to ensure name uniqueness. The scheme is based on existing standards, mainly Open Systems Interconnection (OSI) Distinguished Names; whenever standards disagree, the preference goes to the alternative that offers the widest usage across all protocols. Clear, consistent naming guidelines are given that would

enable IBM customers who have purchased IBM networking and system management products to name their resources so that their administrative processes and IBM's products and protocols can support those resources effectively. A method is suggested to encompass existing name spaces in a single, worldwide naming space, and a migration path is sketched. The interoperation of different protocols across network boundaries using the same naming constructs is shown by means of several scenarios. The naming and addressing scheme proposed here requires nothing new or different from the already defined standards, but allows interoperation among them by using a subset of each.

APPC/MVS distributed application support *by F. W. Voss, p. 381.* Advanced Program-to-Program Communication for Multiple Virtual Storage (APPC/MVS) is a major evolutionary change to MVS for applications that need to connect and communicate across the enterprise. APPC/MVS is an implementation of Systems Network Architecture (SNA) LU 6.2 session-defined protocol. This new MVS environment includes services to enhance the creation, execution, and management of MVS peer-to-peer and client/server applications. In addition to providing connectivity and communications services, APPC/MVS also has scheduling facilities for managing concurrent work originating from other systems in the enterprise network. The objective of this paper is to survey these facilities by examples of usage and by relationships to existing MVS services. Topics include various types of distributed models and approaches, design considerations, and characteristics that represent candidates for an APPC/MVS implementation. Relationships of the APPC/MVS environment with the Customer Information Control System/Enterprise Systems Architecture (CICS/ESA™), Information Management System/ Enterprise Systems Architecture Transaction Manager (IMS/ESA® TM), Time Sharing Option Extensions (TSO/E), and batch environments are included.

Volume 31, Number 3, 1992 *G321-0108*

The evolution of the Common User Access Workplace Model *by R. E. Berry and C. J. Reeves, p. 414.* This paper describes some of the influences contributing to and issues in dealing with the evolution of user interface guidelines over time. In particular, we focus on the evolution of IBM's user interface architecture, the Common User Access™ (CUA™) interface, over a period of six years. Discussed are the key architectural and design elements of the CUA Workplace Model, the fundamental shifts in computer-human interaction that have occurred since the first publication of the guidelines in 1987, and how user interface design, operating systems, and tools have interacted in the evolution of the guidelines.

The information should help designers of user interfaces and developers of user interface guidelines to appreciate some of the factors involved in the long-term evolution of a user interface style. The paper provides an introduction to the most recent evolutionary step in the CUA style (the Workplace Model) to help the reader place these factors in perspective relative to the degree of evolutionary change.

The designer's model of the CUA Workplace *by R. E. Berry, p. 429.* This paper discusses the details, insights, and rationale of the Operating System/2® (OS/2®) Version 2 Workplace Model, an implementation of the user interface defined by the IBM 1991 Common User Access™ (CUA™) guidelines. The Workplace Model is described as an object-oriented user interface where objects represented by icons are manipulated by selection and movement, copying and creation of other objects, and by defining their behavior to accomplish the user's desired task.

Inside IBM's Distributed Data Management architecture *by R. A. Demers, J. D. Fisher, S. S. Gaitonde, and R. R. Sanders, p. 459.* IBM's Distributed Data Management (DDM) architecture is an element of Systems Application Architecture™ that defines an open environment for sharing data in files and relational databases. DDM is a key element of IBM's Distributed Relational Database Architecture. DDM architecture enables programs to access and manage data stored on remote systems. It is a framework for a wide range of additional application services. Influenced by the concepts of object-oriented technology, DDM architecture is designed to be object-oriented. This paper examines DDM architecture from a number of viewpoints, considering why and how it was created, what it is, and how it has evolved.

Data description and conversion architecture *by R. A. Demers and K. Yamaguchi, p. 488.* A data description and conversion architecture has been defined by IBM to enhance data interchange among Systems Application Architecture™ (SAA™) programming languages and systems. Its components, described in this paper, are (1) A Data Language (ADL), a programming language for describing data and specifying what data conversions are to be performed, (2) an object-oriented method of encoding ADL for efficient machine storage, transmission, and processing, and (3) programs that translate the data declarations of other programming languages to or from ADL. Also discussed is the application of the architecture to record-oriented files for SAA Distributed File Management.

SAA distributed file access to the CICS environment *by K. Deinhart, p. 516.* IBM's Customer Information Control System (CICS) is the leading product family in the on-line transaction processing (OLTP) market. OLTP systems are being used by many enterprises to implement their daily business processes and manage operational data such as accounts, inventories, and orders. CICS/Distributed Data Management (CICS/DDM) implements the distributed file function of Systems Application Architecture®

(SAA™) Common Communications Support in the CICS environment on Multiple Virtual Storage (MVS) and Virtual Storage Extended (VSE) operating systems. Providing a DDM target server, CICS/DDM implements IBM's SAA protocol for access to distributed data, which are exploited by the SAA Common Programming Interface. CICS/DDM allows applications and their users to access and share the data managed through the OLTP environment provided by CICS.

The BiProcessor: A merger of two architectures *by C. Berggren, p. 535.* The BiProcessor consists of an IBM System/370™ and a Personal System/2® and merges these two IBM architectures into a synergistic relationship. The two processing environments are connected by an internal high-speed pipe that allows each system to take advantage of the other's strengths as well as developed products, both hardware and software. This paper describes this closely coupled heterogeneous multiprocessor and its capability of concurrent coprocessing. Also discussed are the implementation, coupling architecture, and design considerations of the BiProcessor and its development objectives. Some of the intended applications are host off-loading of communications protocol processing, use as an applications coprocessor, and service as a platform for future clustering technology.

Project Athena: Supporting distributed computing at MIT *by J. M. Arfman and P. Roden, p. 550.* Project Athena™ was an educational computing initiative at the Massachusetts Institute of Technology, undertaken in partnership with the IBM Corporation and Digital Equipment Corporation from 1983 to 1991. This paper gives an overview of the network-based distributed computing services, developed for a number of UNIX™-capable workstations. These services are extensions to the native operating systems of the workstations, and provide interoperability as well as systems administration facilities in a large heterogeneous workstation environment. Under Project Athena, a mature distributed computing environment was developed. Its organization and support structure may be used as a model when planning a new installation, whether on a university or commercial campus. A section of this paper deals with the support requirements for distributed computing environments, based on the Project Athena experience.

Design considerations for distributed applications *by J. J. Rofrano, Jr., p. 564.* Probably the hardest part about developing a distributed application is determining where to start. There are multiple hardware and software platforms to understand, network traffic implications, and numerous tools and technologies to consider. One question, however, transcends the importance of what platform to pick or what tool to use: that is, how do you design it? This paper represents the results of two years of work with customers regarding this question. The paper explores some of the implications of working in a distributed environment, reviews some rules for data and function placement, and introduces a methodology for distributed application design.

Extending and formalizing the framework for information systems architecture *by J. F. Sowa and J. A. Zachman, p. 590.* John Zachman introduced a framework for information systems architecture (ISA) that has been widely adopted by systems analysts and database designers. It provides a taxonomy for relating the concepts that describe the real work to the concepts that describe an information system and its implementation. The ISA framework has a simple elegance that makes it easy to remember, yet it draws attention to fundamental distinctions that are often overlooked in systems design. This paper presents the framework and its recent extensions and shows how it can be formalized in the notation of conceptual graphs.

Volume 31, Number 4, 1992 *G321-0109*

Interactive image segmentation for radiation treatment planning *by P. J. Elliott, J. M. Knapman, and W. Schlegel, p. 620.* COVIRA (COmputer VIsion in RAdiology) is a project in the European Community's Advanced Informatics in Medicine program. The goal is to improve the diagnosis and planning of treatment (radiotherapy) for patients with brain tumors and other diseases. The aim of radiotherapy is to provide a high dose of radiation to a tumor while sparing as much as possible of the surrounding healthy tissue. A necessary first step is defining the target volume and organs at risk by manually outlining the required contours on magnetic resonance or computed tomography scans. For a full three-dimensional plan this is time-consuming, as 40 or more scans are used. Computer image segmentation speeds up the process, and a method that combines information from edge and region detectors is described. Since this method is not able to completely meet the clinical requirements, an interactive image segmentation algorithm has been developed that enables the operator to employ clinical judgment. Probabilities are assigned to edges and regions and presented to the user as a hierarchy of segmentations. The approach is being subjected to extensive clinical evaluation, using pilot applications running on IBM RISC System/6000™ workstations.

Causal probabilistic network modeling—An illustration of its role in the management of chronic diseases *by R. Hovorka, S. Andreassen, J. J. Benn, K. G. Olesen, and E. R. Carson, p. 635.* This paper describes the role of the novel technique of causal probabilistic network (CPN) modeling as an approach to tackling control system problems typified by that of the administration of treatment to the patient suffering from a chronic disease such as diabetes. Three roles of a CPN are discussed. First, since diabetes arises as a consequence of impaired control of carbohydrate metabolism, the ability of a CPN to represent the uncertainty of a physiologically-based model is described. Second, its ability to make robust estimates of the parameters of the metabolic model

is presented, and finally, in conjunction with decision theory approaches, its ability to compare alternative therapies and advise on insulin therapy for patients with insulin-dependent diabetes mellitus is illustrated.

The European telecommunications research and development program RACE and its software project SPECS *by M. Dauphin, M. M. Marques, A. P. Mullery, and P. Rodier, p. 649.* This paper presents the RACE program and the objectives and achievements of SPECS, a representative RACE project. The European Commission has set up the research and development program RACE for the preparation and promotion of an integrated broadband communication system in Europe. The SPECS project develops methods and techniques for the development of the complex software needed by this communication system. Its approach is the use of formal methods and maximum automation. A unique feature of this approach is the support of multiple specification languages, including the ability to mix specification languages within a given system design.

A common compiler for LOTOS and SDL specifications *by C. Binding, W. Bouma, M. Dauphin, G. Karjoth, and Y. Yang, p. 668.* This paper describes a translation of LOTOS and SDL specification languages into executable code, as it was prototyped in the Specification and Programming Environment for Communication Software (SPECS) project under the Research and Development in Advanced Communications in Europe (RACE) program. Both languages are translated into a common intermediate representation in the form of a network of state machines with both synchronous and asynchronous communications. By a series of transformations that make full use of the equivalence relations defined on LOTOS processes, this translation solves unique problems stemming from the highly abstract nature of LOTOS. The common intermediate representation is mapped into C code that can be executed in a specific run-time environment, implemented on a UNIX®-like operating system. SPECS has also developed a pragmatic approach to represent implementable data types in the algebraic framework of LOTOS and SDL, based on a set of predefined type constructors.

The RACE Open Services Architecture project *by A. O. Oshisanwo, M. D. Chapman, M. Key, A. P. Mullery, and J. Saint-Blancat, p. 691.* The specification and implementation of current telecommunication services tend to be intimately bound to a specific network architecture. Moreover, within the service software, interactions between the logical modules are not always explicit, accessible, or uniform, and tend to be optimized for a particular service. This is exemplified by the difficulty experienced in integrating equipment from multiple vendors, and has resulted in telecommunication systems that cannot rapidly exploit the advantages of new technology or respond to changing customer requirements. In addition, current telecommunication services tend not to be viewed as an integral whole, whereby user, control, and management aspects of a service are developed independently from one another. Separate development can lead to problems of inconsistency if shared data are not updated correctly. The RACE Open Services Architecture (ROSA) project was established to address these problems. This paper presents an overview of the approach taken in the ROSA project.

Service and traffic management for IBCN *by K. Geihs, P. Francois, D. Griffin, C. Kaas-Petersen, and A. Mann, p. 711.* The future Integrated Broadband Communications Network (IBCN) will provide high-speed communication capabilities that support a variety of existing and new services. The management of such a complex environment requires innovative management systems. NEMESYS is a project within the European Commission's Research and Development in Advanced Communications in Europe (RACE) program. The project goals are to demonstrate and evaluate the use of advanced information processing techniques for quality-of-service and traffic management. To reach these goals, a series of experimental prototypes are being built. This paper describes the assumptions, objectives, and approach of NEMESYS, and in particular, the design and implementation of an experiment that investigates service and traffic management techniques in a simulated asynchronous transfer mode environment. Because the project is not yet finished, some preliminary results are presented.

The Open Document Architecture: From standardization to the market *by H. Fanderl, K. Fischer, and J. Kämper, p. 728.* The Open Document Architecture (ODA) was developed in the mid-1980s by several standardization bodies. It is now a stable set of international standards for the interchange of compound documents consisting of text, image, and graphic content. Since 1985 the standardization process has been accompanied by European industrial cooperation projects in order to get early experience with the standard and to develop technologies implementing the standard. IBM's European Networking Center has participated in the projects and has prototyped enhancements to OfficeVision™ platforms to allow the interchange of ODA documents between OfficeVision applications and applications running on other vendor platforms. In this paper, the ODA technology is described, experiences of interworking in heterogeneous environments are given, and the role of cooperating with project partners is outlined.

Prolog at IBM: An advanced and evolving application development technology *by M. Bénichou, H. Beringer, J.-M. Gauthier, and C. Beierle, p. 755.* Prolog is a powerful programming language, based on logic, that originated and matured in Europe. This paper aims to show that Prolog is becoming one of the key tools for the entire application development community. First explained is how the unique properties of Prolog give it many advantages over classical languages. Then we show that the language is sufficiently mature technically so that numerous industrial Prolog products are now available. In particular, IBM

offers the Systems Application Architecture® (SAA™) AD/Cycle™ Prolog product family, which provides a combination of logic programming and object programming facilities. Many industrial applications are written in Prolog. Examples of 15 outstanding operational applications, developed by IBM or major IBM customers, are presented. There is a potential for future growth in the types of applications enabled by improving the language. The simplicity and elegance of the theoretical basis of Prolog allow a number of extensions to be defined. Here, three European projects are briefly presented. In conclusion it is shown that Prolog, possibly extended in many directions, is one of the tools that could help solve the long-standing quality and cost problems in application development.

Internal combustion engine design on IBM platforms *by F. Papetti, S. Golini, M. Maggiore, S. Succi, P. Gaillard, and J.-M. Perez, p. 774.* Computer simulation of fluid flow and combustion in diesel engines is rapidly gaining an increasing popularity within the automotive industry, becoming recognized as a cost-effective tool for cutting design cycle time. This paper shows how an advanced computing environment for numerically intensive applications, entirely based upon IBM platforms, can be profitably exploited within the framework of a joint project with industrial partners, in this case Renault Vehicules Industriels. The computing environment has been applied to the code KIVA-II, a computer program for numerical combustion developed at Los Alamos National Laboratory. Numerical simulations have been performed to assess the capability of the code to correctly reproduce the experimental data. Several features, such as visualization of the fuel spray droplet formation and its evolution in time, and selected scalar fields (velocity components, temperature, and vapor concentration), have proved invaluable for a correct understanding of the various phenomena under examination. In particular, the scientific data visualizer, combined with the power of cooperative processing, has allowed a rapid identification of the most significant parameters that need to be tuned in order to recover good agreement between the simulation and the experimental data.

Numerical simulation of reactive flow on the IBM ES/3090 Vector Multiprocessor *by F. K. Hebeker, R. R. Maly, and S. U. Schoeffel, p. 788.* Prohibiting knock damage in internal combustion engines presents severe restrictions for engineers. Laboratory experiments are expensive or even impossible; nevertheless, numerical attempts that employ supercomputers have been rarely undertaken. The numerical approach described in this paper combines a recent shock-capturing finite-volume scheme for the compressible Navier Stokes equations, with semi-implicit treatment of the chemical source terms.

An algorithm is described and validated by experiment that is optimally adapted to vector and parallel computers. The algorithm has been implemented on the IBM Enterprise System/3090™ (ES/3090™) Vector Multiprocessor. Performance measurements are discussed. The potential of the code is illustrated by an example: formation of pseudo shock waves due to interaction of a shock wave with turbulent boundary layer flow.

A modeling study of the North Atlantic with emphasis on the Greenland-Iceland-Norwegian Sea *by T. Aukrust, J. M. Oberhuber, E. J. Farrell, and P. M. Haugan, p. 798.* This essay presents the results of a modeling study that addresses the circulation and convection of ocean currents. A possible change in the global climate due to human-induced increase of atmospheric CO_2 and other greenhouse gases is one of the major environmental challenges in our time. In order to get more insight into this problem, one needs to better understand the various components of the climate system and how they interact. Due to the large heat capacity of the global ocean, the magnitude, delay, and regional distribution of a potential global warming are to a large extent determined by exchanges of heat between the upper ocean and the world's deep ocean. An important process in this regard is deep water formation due to convection. The North Atlantic and the Greenland-Iceland-Norwegian Sea are particularly important regions for this process. Major parts of the circulation in this area are simulated by a coupled ice-ocean model that also includes the entire Arctic Ocean.

Volume 32, Number 1, 1993 *G321-0110*

Strategic alignment: Leveraging information technology for transforming organizations *by J. C. Henderson and N. Venkatraman, p. 4.* It is clear that even though information technology (I/T) has evolved from its traditional orientation of administrative support toward a more strategic role within an organization, there is still a glaring lack of fundamental frameworks within which to understand the potential of I/T for tomorrow's organizations. In this paper, we develop a model for conceptualizing and directing the emerging area of strategic management of information technology. This model, termed the *Strategic Alignment Model,* is defined in terms of four fundamental domains of strategic choice: business strategy, information technology strategy, organizational infrastructure and processes, and information technology infrastructure and processes—each with its own underlying dimensions. We illustrate the power of this model in terms of two fundamental characteristics of strategic management: strategic fit (the interrelationships between external and internal components) and functional integration (integration between business and functional domains). More specifically, we derive four perspectives of alignment with specific implications for guiding management practice in this important area.

Information technology and the management difference: A fusion map *by P. G. W. Keen, p. 17.* When every leading firm in an industry has access to the same information technology resource, the management difference determines competitive advantage

or disadvantage. The management challenge is to make sure that business processes, people, and technology are meshed, instead of being dealt with as separate elements in planning and implementation. This paper presents a framework for senior executives to use in order to lead the deployment of information technology (I/T) without having to know how it is managed and to ensure the fusion of business processes, people, and technology. The "fusion map" approach that focuses on the steps that precede and enable strategy, has been applied in a number of companies. Factors are identified that make I/T a frequent destabilizer of basic logistics in an industry.

New competitive strategies: Challenges to organizations and information technology *by A. C. Boynton, B. Victor, and B. J. Pine II, p. 40.* The old competitive strategies of invention and mass production no longer work in an increasingly turbulent business environment. Successful firms are implementing the new competitive strategies of continuous improvement (constant process improvement) and mass customization—a dynamic flow of goods and services via a stable set of processes. This paper provides a "lens" through which managers can assess their firm's current competitive position, build a vision for where they must be in the future, and craft a transformation strategy to turn that future vision into reality.

Beyond re-engineering: The three phases of business transformation *by W. H. Davidson, p. 65.* New information-technology-based capabilities make it possible to achieve systematic and dramatic gains in business performance. Re-engineering offers one method to access these gains, but a broader process of business transformation explored in this paper can give enterprises a greater range of benefits. This three-phase transformation process starts with structured automation and re-engineering efforts, builds on new infrastructure and capabilities to enhance and extend the original business, and then redefines it to create new businesses.

A new approach to business processes *by A. L. Scherr, p. 80.* This paper presents a methodology for analyzing and designing the processes that an enterprise uses to conduct its business. The methodology builds upon traditional approaches to business process definition by adding the dimension of people's accountabilities: their roles, relationships, and agreements. The approach presented allows for unique insights into customer satisfaction, employee empowerment, and quality. It also provides a basis for spanning the concerns of both business people and information technologists responsible for providing business process automation.

Measuring the value of information: The information-intensive organization *by R. Glazer, p. 99.* This paper suggests that firms that successfully integrate an information technology (I/T) strategy with their business strategies do so by focusing on the information itself, rather than on technology, as the real carrier of value and source of competitive advantage. A primary mechanism by which a firm becomes an information-intensive firm is the implementation of a procedure for measuring the value of its information assets V(I). This paper presents a methodology for measuring the value of information in the firm. This paper describes the application of that methodology in an actual case study and discusses some consequences of being able to compare organizations with respect to their relative levels of information intensity.

Strategic control in the extended enterprise *by B. R. Konsynski, p. 111.* The strategic role of information systems in "extending" the enterprise is examined. A number of issues emerge as essential considerations in the strategic alignment of the investment in information technology and business strategy. Information technologies transform organizational boundaries, interorganizational relations, and marketplace competitive and cooperative practice. The paper presents a framework of *strategic control* that guides the planning and execution of these investments in information technology for business transformation, seeking increased *understanding* and *influence.* Emerging information technologies change the limits of what is possible in the leverage of strategic control through transformation of boundaries, relations, and markets.

Global business drivers: Aligning information technology to global business strategy *by B. Ives, S. L. Jarvenpaa, and R. O. Mason, p. 143.* The alignment of worldwide computer-based information systems and integrated business strategies is critical to the success of multinational firms in a highly competitive global market. In this paper, information technology (I/T) solutions are explored that drive firms toward making economic decisions based on worldwide distributed knowledge. These solutions focus on a number of entities (or global business drivers) that identify where a firm can benefit most from the management and application of the technology. A variety of approaches for overcoming the barriers and risks of applying this technology are also discussed.

Improving business and information strategy alignment: Learning from the banking industry *by M. Broadbent and P. Weill, p. 162.* An empirical study that explored business and information strategy alignment in the information-intensive and competitive Australian banking industry is featured in this paper. The aim of the study was to identify organizational practices that contribute to and enhance such alignment. Multiple sources of information were used to collect data about business and information strategies from the major firms dominating Australian banking. Sources included written and interview-based information, strategic planning documentation, and annual reports. Evidence was sought for the alignment of business and information strategies through the use of information and information technology that provided a comparative advantage to

an organization over its competitors. The firm-wide strategy-formation processes of the banks, rather than their information systems (I/S) methodology, was central to the alignment of business and information strategies. The interdependence of firm-wide processes and I/S factors are emphasized in a strategic alignment model that summarizes the findings of the study. The paper concludes with a discussion of the management implications and requirements for action in both firm-wide strategy and I/S areas. The results of this study in the banking industry are pertinent to other industries where information technology and systems are playing an increasingly strategic role.

Quantitative techniques in strategic alignment *by P. V. Norden, p. 180.* There is increasing evidence in both the business and technical literature that the operations and strategy processes of many organizations have been aided materially by visualization and modeling techniques. Application of quantitative methods has progressed from relatively well-structured operations to the more speculative aspects of strategy and policy formation. In retrospect, however, the most valuable contribution of modeling has been greater insight: a clearer understanding of the situations and prospects at hand that the mere act of model formulation often provided the planner. This paper illustrates some characteristics of the modeling process, and explores the applicability of quantitative techniques to strategic alignment opportunities, such as current pressures to reduce the "cycle time" of many enterprise functions.

Transforming the enterprise: The alignment of business and information technology strategies *by J. N. Luftman, P. R. Lewis, and S. H. Oldach, p. 198.* The strategic use of information technology (I/T) is now and has been a fundamental issue for every business. In essence, I/T can alter the basic nature of an industry. The effective and efficient utilization of information technology requires the alignment of the I/T strategies with the business strategies, something that was not done successfully in the past with traditional approaches. New methods and approaches are now available. The strategic alignment framework applies the Strategic Alignment Model to reflect the view that business success depends on the linkage of business strategy, information technology strategy, organizational infrastructure and processes, and I/T infrastructure and processes. In this paper, we look at why it may not be sufficient to work on any one of these areas in isolation or to only harmonize business strategy and information technology. One reason is that, often, too much attention is placed on technology, rather than business, management, and organizational issues. The objective is to build an organizational structure and set of business processes that reflect the interdependence of enterprise strategy and information technology capabilities. The attention paid to the linkage of information technology to the enterprise can significantly affect the competitiveness and efficiency of the business. The essential issue is how information technology can enable the achievement of competitive and strategic advantage for the enterprise.

Volume 32, Number 2, 1993 *G321-0111*

Box-structured methods for systems development with objects *by A. R. Hevner and H. D. Mills, p. 232.* Box structures provide a rigorous and systematic process for performing systems development with objects. Box structures represent data abstractions as objects in three system views and combine the advantages of structured development with the advantages of object orientation. As data abstractions become more complex, the box structure usage hierarchy allows stepwise refinement of the system design with referential transparency and verification at every step. An integrated development environment based on box structures supports flexible object-based systems development patterns. We present a classic example of object-based systems development using box structures.

I/O subsystem configurations for ESA: New roles for processor storage *by B. McNutt, p. 252.* I/O subsystem configurations are dictated by the storage and I/O requirements of the specific applications that use the disk hardware. Treating the latter requirement as a given, however, draws a boundary at the channel interface that is not well-suited to the capabilities of the Enterprise Systems Architecture (ESA). This architecture allows hardware expenditures in the I/O subsystem to be managed, while at the same time improving transaction response time and system throughput capability, by a strategy of processor buffering coupled with storage control cache. The key is to control the aggregate time per transaction spent waiting for physical disk motion. This paper investigates how to think about and accomplish such an objective. A case study, based on data collected at a large Multiple Virtual Storage installation, is used to investigate the potential types and amounts of memory use by individual files, both in storage control cache and in processor buffers. The mechanism of interaction between the two memory types is then examined and modeled so as to develop broad guidelines for how best to deploy an overall memory budget. These guidelines tend to contradict the usual metrics of storage control cache effectiveness, underscoring the need for an adjustment in pre-ESA paradigms.

Introduction of the project management discipline in a software development organization *by T. Raz, p. 265.* An approach to the introduction of the project management discipline in a software development organization is presented, with emphasis on the aspects that can be generalized and adopted by other organizations under similar circumstances. The presentation includes the key elements of the approach taken, the maturity scale used to guide the introduction effort, a short description of the education program developed specifically for this case, a methodology for developing project models, and the staff and support structure put in place. The paper

concludes by reporting the initial experience and noting directions for future development.

Building business and application systems with the Retail Application Architecture *by P. Stecher, p. 278.* An industry application architecture is a framework for integrating applications and databases and can also be used for analyzing and re-engineering the business of an enterprise as a whole, provided it is structured correctly. This paper describes the motivation, structure, and possible uses of the Retail Application Architecture™ (RAA™). The core of RAA is a set of generic enterprise models for companies in the retail and wholesale distribution industry. RAA is oriented as much to the business expert as to the information systems (I/S) department. The goal of RAA is to contribute to the task of building sound business systems in a more efficient and effective manner.

System for the recognition of human faces *by M. S. Kamel, H. C. Shen, A. K. C. Wong, and R. I. Campeanu, p. 307.* This paper describes a system for content-based retrieval of facial images from an image database. The system includes feature extraction based on expert-assisted feature selection, spatial feature measurement, feature and shape representation, feature information compression and organization, search procedures, and pattern-matching techniques. The system uses novel data structures to represent the extracted information. These structures include attributed graphs for representing local features and their relationships, n-tuple of mixed mode data, and highly compressed feature codes. For the retrieval phase, a knowledge-directed search technique that uses a hypothesis refinement approach extracts specific features for candidate identification and retrieval. The overall system, the components, and the methodology are described. The system has been implemented on an IBM Personal System/2® running Operating System/2®. Examples demonstrating the performance of the system are included.

Technical note—Complementarity attacks and control vectors *by D. Longley and S. M. Matyas, p. 321.* A control vector is a data structure that specifies the nature and role of an associated cryptographic key. The control vector is checked by software and cryptographic hardware in order to limit the range of permissible operations to be undertaken with ciphertext produced with the key. The linking of the control vector and cryptographic key is such that attempts to modify, or substitute, control vectors will cause the subsequent processing to operate with a corrupted key, and hence ensure protection of data encrypted with the genuine key. A potential attack on the control vector approach is described in which the complement of the control vector is substituted. The manner in which such attacks are thwarted by the IBM implementation of control vectors is also described.

Process automation in software application development *by K. D. Saracelli and K. F. Bandat, p. 376.* Over the years, the field of application development (AD) has evolved from that of an art form to being more of a science, hence the emergence of concepts such as information engineering. In engineering and scientific fields, the value of process definition and management has long been known. This paper discusses the requirements for managing the AD process and establishes the need for automated assistance for these management activities. Considerations for an automated system to manage the process are presented, and the benefits to be realized by such an implementation are then discussed.

Rapid Delivery: An evolutionary approach for application development *by D. Hough, p. 397.* From a historical vantage point, large application development projects are frequently at risk of failure. Applications are typically developed using a monolithic development approach. Monolithic approaches generally feature business-user-defined requirements that are incorporated in the application but not evident until the resulting application has been implemented. To effectively produce new information systems, innovative methods must be utilized. This paper provides information about one of these, Rapid Delivery—a method for developing applications that can evolve over time. To fully understand the principles of Rapid Delivery, a discussion is included that illuminates a three-dimensional application model and its variations. The application model helps in understanding application segmentation, a technique used in Rapid Delivery to break applications into a variety of functional capabilities. After the development of each application segment has been completed, it is implemented to provide immediate benefit to the enterprise; each application segment is added to the evolving application and its ever-expanding capabilities. The result of using Rapid Delivery is an enhanced ability to build applications that better support the enterprise through a continuous stream of delivered requirements, a reduction in the possibility of project failure, and a diminished likelihood of runaway projects.

The impact of object-orientation on application development *by A. A. R. Cockburn, p. 420.* Object-orientation introduces new deliverables, notations, techniques, activities, and tools. Application development consists not only of these items but also of work segmentation, scheduling, and managing the sharing and evolution of deliverables. This paper breaks application development into three major components: construction, coordination, and evolution, with the topic of reuse receiving extra attention. It highlights four aspects of object-orientation having impact: encapsulation, anthropomorphic design, reuse with extensibility, and incremental and iterative development.

Measurement: The key to application development quality *by C. Walrad and E. Moss, p. 445.* Application development quality and productivity have been identified as being among the top ten concerns of information systems (I/S) executives in both 1991 and 1992. This paper discusses the role of measurement in pursuit of I/S application development quality and productivity. The relationships between productivity, quality, and measurement are described, classes of measures are identified, and "dominant measures" are grouped according to the maturity levels defined by the Software Engineering Institute's Capability Maturity Model for Software. Also discussed are the organizational and cultural issues associated with instituting a measurement process.

A public key extension to the Common Cryptographic Architecture *by A. V. Le, S. M. Matyas, D. B. Johnson, and J. D. Wilkins, p. 461.* A new method for extending the IBM Common Cryptographic Architecture (CCA) to include public key cryptography is presented. The public key extension provides nonrepudiation via digital signatures and an electronic means to distribute Data Encryption Algorithm (DEA) key-encrypting keys in a hybrid Data Encryption Algorithm-Public Key Algorithm (DEA-PKA) cryptographic system. The improvements are based on a novel method for extending the control vector concept used in the IBM Common Cryptographic Architecture. Four new key types that separate the public and private key pairs into four classes according to their broad uses within the cryptographic system are defined. The public key extension to the CCA is implemented in the IBM Transaction Security System (TSS). This paper discusses both the public key extension to the CCA and the TSS implementation of this architectural extension.

Morphologically based automatic phonetic transcription *by K. Wothke, p. 486.* A system is described that automatically generates phonetic transcriptions for German orthographic words. The entire generative process consists of two main steps. In the first step, the system segments the words into their morphs, or prefixes, stems, and suffixes. This segmentation is very important for the transcription of German words, because the pronunciation of the letters depends also on their morphological environment. In the second step, the system transcribes the morphologically segmented words. Several transcriptions can be generated per word, thus permitting the system to take pronunciation variants into account. This feature results from the application area of the system, which is the provision of phonetic reference units for an automatic large-vocabulary speech recognition system. Statistical evaluations show that the transcription system has an excellent linguistic performance: more than 99 percent of the segmented words obtain a correct segmentation in the first step, and more than 98 percent of the words receive a correct phonetic transcription in the second step.

A storage subsystem for image and records management *by H. M. Gladney, p. 512.* Digital storage and communications are becoming cost effective for massive collections of document images with access not only for nearby users but also for those who are hundreds of miles from their libraries. The Document Storage Subsystem (DocSS) provides generic library services such as searching, storage, and retrieval of document pages and sharing of objects with appropriate data security and integrity safeguards. A library session has three components: a manager of remote catalogs, a set of managers of large-object stores, and a manager of cache services. DocSS supports all kinds of page data—text, pictures, spreadsheets, graphics, programs—and can be extended to audio and video data. Document models can be built as DocSS applications; the paper describes a folder manager as an example. What differentiates DocSS among digital library projects is its approach to data distribution over wide area networks, its client-server approach to the heterogeneous environment, and its synergism with other components of evolving open systems.

Volume 32, Number 4, 1993 *G321-0114*

Software reuse: From library to factory *by M. L. Griss, p. 548.* Systematic software reuse is a key business strategy that software managers can employ to dramatically improve their software development processes, to decrease time-to-market and costs, and to improve product quality. Effective reuse requires much more than just code and library technology. We have learned that careful consideration must be given to people, process, and technology. One approach to the systematic integration of these three elements is the concept of the software factory. At Hewlett-Packard Co., we have initiated a multifaceted corporate reuse program to help introduce the best practices of systematic reuse into the company, complemented by multidisciplinary research to investigate and develop better methods for domain-specific, reuse-based software engineering. This essay discusses our experiences. Key aspects include domain-specific kits, business modeling, organization design, and technology infrastructure for a flexible software factory.

The business case for software reuse *by J. S. Poulin, J. M. Caruso, and D. R. Hancock, p. 567.* To remain competitive, software development organizations must reduce cycle time and cost, while at the same time adding function and improving quality. One potential solution lies in software reuse. Because software reuse is not free, we must weigh the potential benefits against the expenditures of time and resources required to identify and integrate reusable software into products. We first introduce software reuse concepts and examine the cost-benefit trade-offs of software reuse investments. We then provide a set of metrics used by IBM to accurately reflect the effort saved by reuse. We define reuse metrics that distinguish the savings and benefits from those already gained through accepted software engineering tech-

niques. When used with the return-on-investment (ROI) model described in this paper, these metrics can effectively establish a sound business justification for reuse and can help assess the success of organizational reuse programs.

Implementing Critical Success Factors in software reuse *by M. Wasmund, p. 595.* Software reuse is one of several technologies that can improve quality and effectiveness of software development. The introduction of a reuse infrastructure within an existing organization and the associated modification of employee behavior and processes is a complex interdisciplinary task. The structuring and monitoring of several coordinated activities is required in order to be successful. This paper describes a practical application of the Critical Success Factors method on reuse technology insertion into the software development process. The Critical Success Factors method has proved to be a useful means for the introduction of software reuse concepts. Application of the method and results are discussed in detail, concluding with lessons learned and recommendations for similar efforts.

Technical forum—Management of reuse at IBM *by J. R. Tirso and H. Gregorius, p. 612.*

Technical forum—Information reuse parallels software reuse *by K. P. Yglesias, p. 615.*

Technical forum—A reusable parts center *by D. Bauer, p. 620.*

Application reference designs for distributed systems *by J. J. Shedletsky and J. J. Rofrano, p. 625.* This paper is based on the findings and conclusions of a client/server work group that was commissioned in 1991 to report IBM's technical strategy for client/server computing. Although there are countless variations for designing applications and interconnecting components in a distributed environment, there seems to be a finite number of variations that represent what a large majority of customers want to build. The intent of the work group was to explore the possibility of defining a set of application "reference designs," which would represent the distributed designs that customers are building today or want to build in the near future. This paper documents the customer scenarios, the reference designs that represent them, and the requirements that were generated for the underlying system software. The work group concluded that the reference designs described herein represent our best working assumption about "where customers are going" with distributed application designs. The discussion should give those who have not yet begun to exploit distributed systems a starting point and considerations for their design work.

Advanced Function Printing—From print to presentation *by R. K. deBry and M. W. Munger, p. 647.* The strength of Advanced Function Printing™ (AFP™) is due largely to the architectures that form its foundation. The architectures on which AFP is based have been developed over the last 12 years and have influenced the development of standards, competitive architectures, and, most importantly, software inside and outside IBM. Customers are demanding a more comprehensive view of printing that includes easy creation, viewing, and even specialized editing of printable documents. These "next generation" requirements are now being satisfied by software products that are based on the existing architecture. This paper describes some of these products and how they use the architecture, and describes possible future directions for AFP and related technologies.

The continuing evolution of Advanced Function Printing *by R. J. Howarth and B. G. Platte, p. 665.* Advanced Function Printing™ (AFP™) has become one of the de facto printing standards. It is a broad architecture to support printing across an entire enterprise and encompasses IBM architectures as well as industry standards. AFP had its beginnings in the IBM System/370™ environment in 1984 and has since expanded to include midrange and local area network systems. Recently the capabilities of AFP have been extended beyond printing to include on-line viewing and management of presentation data. An overview of AFP capabilities was given in an earlier issue of the *IBM Systems Journal.* This paper traces the continuing evolution of AFP and its usage and how it is addressing the presentation requirements of businesses in the 1990s.

Volume 33, Number 1, 1994 G321-0115

Software quality: An overview from the perspective of total quality management *by S. H. Kan, V. R. Basili, and L. N. Shapiro, p. 4.* This essay presents a tutorial that discusses software quality in the context of total quality management (TQM). Beginning with a historical perspective of software engineering, the tutorial examines the definition of software quality and discusses TQM as a management philosophy along with its key elements: customer focus, process improvement, the human side of quality, and data, measurement, and analysis. It then focuses on the software-development specifics and the advancements made on many fronts that are related to each of the TQM elements. In conclusion, key directions for software quality improvements are summarized.

Forging a silver bullet from the essence of software *by R. G. Mays, p. 20.* Most improvements in software development technology have occurred by eliminating the accidental aspects of the technology. Further progress now depends on addressing the essence of software. Fred Brooks has characterized the essence of software as a complex construct of interlocking concepts. He concludes that no silver bullet will magically reduce the essential conceptual complexity of software. This paper expands on Brooks's definition to lay a foundation for forging a possible silver bullet. Discussed are the three essential attributes of software entities from which a number of consequences arise in software development: (1) conceptual content, (2) representation, and (3) multiple subdo-

mains. Four basic approaches to develop technologies are proposed that directly address the essential attributes. Although some of these technologies require additional development or testing, they present the most promise for forging a silver bullet. Among them, design reabstraction addresses the most difficult attribute, multiple subdomains, and the most difficult consequence, enhancing existing code, making it the best prospect.

Journey to a mature software process *by C. Billings, J. Clifton, B. Kolkhorst, E. Lee, and W. B. Wingert, p. 46.* Development process maturity is strongly linked to the success or failure of software projects. As the word "maturity" implies, time and effort are necessary to gain it. The Space Shuttle Onboard Software project has been in existence for nearly 20 years. In 1989 the project was rated at the highest level of the Software Engineering Institute's Capability Maturity Model. The high-quality software produced by the project is directly linked to its maturity. This paper focuses on the experiences of the Space Shuttle Onboard Software project in the journey to process maturity and the factors that have made it successful.

AS/400 software quality management *by S. H. Kan, S. D. Dull, D. N. Amundson, R. J. Lindner, and R. J. Hedger, p. 62.* This paper describes the software quality management system for the Application System/400® (AS/400®) computer system. Key elements of the quality management system such as customer satisfaction, product quality, continuous process improvement, and people are discussed. Based on empirical data, recent progress in several quality parameters of the AS/400 software system are examined. The quality action road map that describes the various quality actions that were deployed is presented, as are the other elements that enabled the implementation of the quality management system.

Adopting Cleanroom software engineering with a phased approach *by P. A. Hausler, R. C. Linger, and C. J. Trammell, p. 89.* Cleanroom software engineering is a theory-based, team-oriented engineering process for developing very high quality software under statistical quality control. The Cleanroom process combines formal methods of object-based box structure specification and design, function-theoretic correctness verification, and statistical usage testing for reliability certification to produce software approaching zero defects. Management of the Cleanroom process is based on a life cycle of development and certification of a pipeline of user-function increments that accumulate into the final product. Teams in IBM and other organizations that use the process are achieving remarkable quality results with high productivity. A phased implementation of the Cleanroom process enables quality and productivity improvements with an increased control of change. An introductory implementation involves the application of Cleanroom principles without the full formality of the process; full implementation involves the comprehensive use of formal Cleanroom methods; and advanced implementation optimizes the process through additional formal methods, reuse, and continual improvement. The AOEXPERT/MVS™ project, the largest IBM Cleanroom effort to date, successfully applied an introductory level of implementation. This paper presents both the implementation strategy and the project results.

RE-Analyzer: From source code to structured analysis *by A. B. O'Hare and E. W. Troan, p. 110.* The RE-Analyzer is an automated, reverse engineering system providing a high level of integration with a computer-aided software engineering (CASE) tool. Specifically, legacy code is transformed into abstractions within a structured analysis methodology. The abstractions are based on data flow diagrams, state transition diagrams, and entity-relationship data models. Since the resulting abstractions can be browsed and modified within a CASE tool environment, a broad range of software engineering activities are supported, including program understanding, reengineering, and redocumentation. In addition, diagram complexity is reduced through the application of *control partitioning:* an algorithmic technique for managing complexity by partitioning source code modules into smaller yet semantically coherent units. This approach also preserves the information content of the original source code. It is in contrast to other reverse engineering techniques that produce only structure charts and thus suffer from loss of information, unmanaged complexity, and a lack of correspondence to structured analysis abstractions. The RE-Analyzer has been implemented and currently supports the reverse engineering of software written in the C language. It has been integrated with a CASE tool based on the VIEWS method.

The impact of object-oriented technology on software quality: Three case histories *by N. P. Capper, R. J. Colgate, J. C. Hunter, and M. F. James, p. 131.* Techniques to obtain software quality are examined from the experiences of three very different object-oriented projects carried out by IBM Information Solutions Limited in 1991 and 1992. Object-oriented programming systems are sold on the promise of improved productivity from object reuse and a high level of code modularity. Yet it is precisely these aspects that also lead to their greatest benefit, namely improved software quality. In this paper, lessons learned from the three projects are described and compared, indicating approaches to consider in using object-oriented technology.

Deriving programs using generic algorithms *by V. R. Yakhnis, J. A. Farrell, and S. S. Shultz, p. 158.* We suggest a new approach to the derivation of programs from their specifications. The formal derivation and proof of programs as is practiced today is a very powerful tool for the development of high-quality software. However, its application by the software development community has been slowed by the amount of mathematical expertise needed to apply these formal methods to complex projects and

by the lack of reuse within the framework of program derivation. To address these problems, we have developed an approach to formal derivation that employs the new concept of generic algorithms. A generic algorithm is one that has (1) a formal specification, (2) a proof that it satisfies this specification, and (3) generic identifiers representing types and operations. It may have embedded program specifications or pseudocode instructions describing the next steps in the stepwise refinement process. Using generic algorithms, most software developers need to know only how to pick and adapt them, rather than perform more technically challenging tasks such as finding loop invariants and deriving loop programs. The adaptation consists of replacing the generic identifiers by concrete types and operations. Since each generic algorithm can be used in the derivation of many different programs, this new methodology provides the developer with a form of reuse of program derivation techniques, correctness proofs, and formal specifications.

In-process improvement through defect data interpretation *by I. Bhandari, M. J. Halliday, J. Chaar, R. Chillarege, K. Jones, J. S. Atkinson, C. Lepori-Costello, P. Y. Jasper, E. D. Tarver, C. C. Lewis, M. Yonezawa, p. 182.* An approach that involves both automatic and human interpretation to correct the software production process during development is becoming important in IBM as a means to improve quality and productivity. A key step of the approach is the interpretation of defect data by the project team. This paper uses examples of such correction to evaluate and evolve the approach, and to inform and teach those who will use the approach in software development. The methodology is shown to benefit different kinds of projects beyond what can be achieved by current practices, and the collection of examples discussed represents the experiences of using a model of correction.

Technical forum—Programming quality improvement in IBM *by D. L. Bencher p. 215.*

Technical note—On reliability modeling and software quality *by A. J. Watkins, p. 220.* This note continues the recent discussion on reliability modeling and its application in software development by Kan in a recent issue of the *IBM Systems Journal.* We focus on the initial stages of a reliability modeling process, as decisions here will often influence the later stages of an analysis.

Volume 33, Number 2, 1994 *G321-0116*

Decision support at Lands' End—An evolution *by G. G. Bustamente and K. Sorenson, p. 228.* A decision support system with over one billion rows of data has been developed at Lands' End using the IBM DATABASE 2™ (DB2®) relational database management system. This corporate database is a subset of an Information Warehouse™ framework and functions as both a decision support system server and an application enabler. The corporate database uses operational data gathered from order processing and customer mailing systems. Weekly processes reformat these real-time data for loading into the corporate database. This paper discusses some of the business requirements that guided the development of the corporate database, and also describes the database design process, tool selection, and implementation experiences.

The Business Object Management System *by M. Schlatter, R. Furegati, F. Jeger, H. Schneider, and H. Streckeisen, p. 239.* The Business Object Management System (BOMS) is a distributed resource manager that generalizes and extends the concepts of shared corporate information to include not only data that are structured such that the data can be held in relational tables but also generalized, complex business information objects. BOMS allows enterprises to store, manage, and query the totality of their documents, business transaction records, images, etc., in a uniform and consistent way. With this system, businesses can make more effective use of information that has in the past been inaccessible to thorough and systematic queries and that could not be integrated effectively into existing or new business processes. BOMS is targeted toward very large collections of information objects (on the order of a billion objects, equivalent to terabytes of data) and allows enterprises to unlock information treasures that would otherwise remain hidden in collections of that size. BOMS is influenced by theoretical concepts, such as object-orientation and hypermedia, but relies on proven relational database and transaction processing concepts. BOMS has been implemented with DATABASE 2™ (DB2®) and Customer Information Control System/Enterprise Systems Architecture (CICS/ESA™) and has been in productive use since 1991.

Extending relational database technology for new applications *by J. M. Cheng, N. M. Mattos, D. D. Chamberlin, and L. G. DeMichiel, p. 264.* Relational database systems have been very successful in meeting the needs of today's commercial applications. However, emerging applications in disciplines such as engineering design are now generating new requirements for database functionality and performance. This paper describes a set of extensions to relational database technology, designed to meet the requirements of the new generation of applications. These extensions include a rich and extensible type subsystem that is tightly integrated into the Structured Query Language (SQL), a rules subsystem to enforce global database semantics, and a variety of performance enhancements. Many of the extensions described here have been prototyped at the IBM Database Technology Institute and in research projects at the IBM Almaden Research Center in order to demonstrate their feasibility and to validate their design. Furthermore, many of these extensions are now under consideration as part of the evolving American National Standards Institute/International Organization for Standardization (ANSI/ISO) standard for the SQL database language.

Maximizing leverage from an object database *by C. Alfred, p. 280.* With increasing frequency, object database management systems (ODBMSs) are being used as a persistent storage framework for applications. This paper shows that ODBMS frameworks provide a natural repository for supporting object-oriented systems, because they store and manage objects as their atomic units. In addition, these frameworks can offer a great deal of leverage to the developers of applications with the integration of two distinct paradigm shifts: the object-oriented development model, and the direct-reference storage model. Software developers who understand the implications of both paradigm shifts are more likely to use the technology effectively and realize most or all of the potential leverage. Highlighted is ObjectStore™ from Object Design, Inc., which is available as part of the IBM object database solution.

Data access within the Information Warehouse framework *by J. P. Singleton and M. M. Schwartz, p. 300.* IBM's Information Warehouse™ framework provides a basis for satisfying enterprise requirements for effective use of business data resources. It includes an architecture that defines the structure and interfaces for integrated solutions and includes products and services that can be used to create solutions. This paper uses the Information Warehouse architecture as a context to describe software components that can be used for direct access to formatted business data in a heterogeneous systems environment. Concepts of independence between software components and how this independence can provide flexibility for change are discussed. The integration of software from multiple vendors to create effective solutions is a key emphasis of this paper.

Managing business processes as an information resource *by F. Leymann and W. Altenhuber, p. 326.* The relevance of business processes as a major asset of an enterprise is more and more accepted: Business processes prescribe the way in which the resources of an enterprise are used, i.e., they describe how an enterprise will achieve its business goals. Organizations typically prescribe how business processes have to be performed, and they seek information technology that supports these processes. We describe a system that supports the two fundamental aspects of business process management, namely the modeling of processes and their execution. The meta-model of our system deals with models of business processes as weighted, colored, directed graphs of activities; execution is performed by navigation through the graphs according to a well-defined set of rules. The architecture consists of a distributed system with a client/server structure, and stores its data in an object-oriented database system.

Parallelism in relational database management systems *by C. Mohan, H. Pirahesh, W. G. Tang, and Y. Wang, p. 349.* In order to provide real-time responses to complex queries involving large volumes of data, it has become necessary to exploit parallelism in query processing. This paper addresses the issues and solutions relating to intraquery parallelism in a relational database management system (DBMS). We provide a broad framework for the study of the numerous issues that need to be addressed in supporting parallelism efficiently and flexibly. The alternatives for a parallel architecture system are discussed, followed by the focus on how a query can be parallelized and how that affects load balancing of the different tasks created. The final part of the paper contains information about how the IBM DATABASE 2™ (DB2®) Version 3 product provides support for I/O parallelism to reduce response time for data-intensive queries.

Volume 33, Number 3, 1994 *G321-0117*

The Centre for Advanced Studies: A model for applied research and development *by J. Slonim, M. A. Bauer, P.-Å. Larson, J. Schwarz, C. Butler, E. B. Buss, and D. Sabbah, p. 382.* The Centre for Advanced Studies (CAS) is an applied research centre formed in 1990 within the IBM Toronto Software Solutions Laboratory. Its primary aim is to facilitate the transfer of research ideas into the various product groups of the laboratory. Although we are still learning how to make CAS operate more effectively, and it is too early to assess its long-term success, the model for CAS has proved to be workable. The primary partners, namely the IBM Toronto Software Solutions Laboratory, the IBM research community, universities in North America, and government agencies that support collaborative research, have found it a viable approach. As an overview, this essay provides some background to the formation of the centre, describes some of the challenges deemed important in defining the role of the centre, identifies a number of principles that are used to guide its formation and current operation, and reports on its progress. We conclude with a discussion of some lessons learned in the operation of the centre to date and identify future activities and directions for the centre.

A distributed system architecture for a distributed application environment *by M. A. Bauer, N. Coburn, D. L. Erickson, P. J. Finnigan, J. W. Hong, P.-Å. Larson, J. Pachl, J. Slonim, D. J. Taylor, and T. J. Teorey, p. 399.* Advances in communications technology, development of powerful desktop workstations, and increased user demands for sophisticated applications are rapidly changing computing from a traditional centralized model to a distributed one. The tools and services for supporting the design, development, deployment, and management of applications in such an environment must change as well. This paper is concerned with the architecture and framework of services required to support distributed applications through this evolution to new environments. In particular, the paper outlines our rationale for a peer-to-peer view of distributed systems, presents motivation for our research directions, describes an architecture, and reports on some preliminary experiences with a prototype system.

Reference architecture for distributed systems management *by M. A. Bauer, P. J. Finnigan, J. W. Hong, J. A. Rolia, T. J. Teorey, and G. A. Winters, p. 426.* Management of computing systems is needed to ensure efficient use of resources and provide reliable and timely service to users. Distributed systems are much more difficult to manage because of their size and complexity, and they require a new approach. A reference architecture for distributed systems management is proposed that integrates system monitoring, information management, and system modeling techniques. Three classes of system management—network services and devices, operating system services and resources, and user applications—are defined within this framework, and a detailed hospital application is presented to clarify the requirements for managing applications. It is argued that the performance management of distributed applications must be considered from all three perspectives. Several management prototypes under study within the COnsortium for Research on Distributed Systems (CORDS) are described to illustrate how such an architecture could be realized.

Evaluation of a predicate-based software testing strategy *by K.-C. Tai, M. A. Vouk, A. M. Paradkar, and P. Lu, p. 445.* In this paper, we report the results of four empirical studies for evaluating a predicate-based software testing strategy, called BOR (Boolean operator) testing. The BOR testing strategy focuses on the detection of Boolean operator faults in a predicate, including incorrect AND/OR operators and missing or extra NOT operators. Our empirical studies involved comparisons of BOR testing with several other predicate-based testing strategies, using Boolean expressions, a real-time control system, and a set of N-version programs. For program-based test generation, BOR testing was applied to predicates in a program. For specification-based test generation, BOR testing was applied to cause-effect graphs representing software specification. The results of our studies indicate that BOR testing is practical and effective for both specification- and program-based test generation.

Architecture and applications of the Hy$^+$ visualization system *by M. P. Consens, F. Ch. Eigler, M. Z. Hasan, A. O. Mendelzon, E. G. Noik, A. G. Ryman, and D. Vista, p. 458.* The Hy$^+$ system is a generic visualization tool that supports a novel visual query language called GraphLog. In Hy$^+$, visualizations are based on a graphical formalism that allows comprehensible representations of databases, queries, and query answers to be interactively manipulated. This paper describes the design, architecture, and features of Hy$^+$ with a number of applications in software engineering and network management.

Investigating reverse engineering technologies for the CAS program understanding project *by E. Buss, R. De Mori, W. M. Gentleman, J. Henshaw, H. Johnson, K. Kontogiannis, E. Merlo, H. A. Müller, J. Mylopoulos, S. Paul, A. Prakash, M. Stanley, S. R. Tilley, J. Troster, and K. Wong, p. 477.* Corporations face mounting maintenance and re-engineering costs for large legacy systems. Evolving over several years, these systems embody substantial corporate knowledge, including requirements, design decisions, and business rules. Such knowledge is difficult to recover after many years of operation, evolution, and personnel change. To address the problem of program understanding, software engineers are spending an ever-growing amount of effort on reverse engineering technologies. This paper describes the scope and results of an ongoing research project on program understanding undertaken by the IBM Toronto Software Solutions Laboratory Centre for Advanced Studies (CAS). The project involves a team from CAS and five research groups working cooperatively on complementary reverse engineering approaches. All the groups are using the source code of SQL/DS™ (a multimillion-line relational database system) as the reference legacy system. Also discussed is an approach adopted to integrate the various tools under a single reverse engineering environment.

Emerging technologies that support a software process life cycle *by G. T. Heineman, J. E. Botsford, G. Caldiera, G. E. Kaiser, M. I. Kellner, and N. H. Madhavji, p. 501.* The goal of developing quality software can be achieved by focusing on the improvement of both product quality and process quality. While the traditional focus has been on product quality, there is an increased awareness of the benefits of improving the quality of the processes used to develop and support those products. These processes are key elements in understanding and improving the practice of software engineering. In this paper, existing objectives for the development and application of models of software processes are restated, and current research sponsored by the IBM Centre for Advanced Studies (CAS) is discussed as it applies to furthering each of the objectives. A framework is also presented that relates the research work to the various sectors of a software process life cycle. The on-going research involves four universities, CAS, and collaboration with IBM Toronto Laboratory developers.

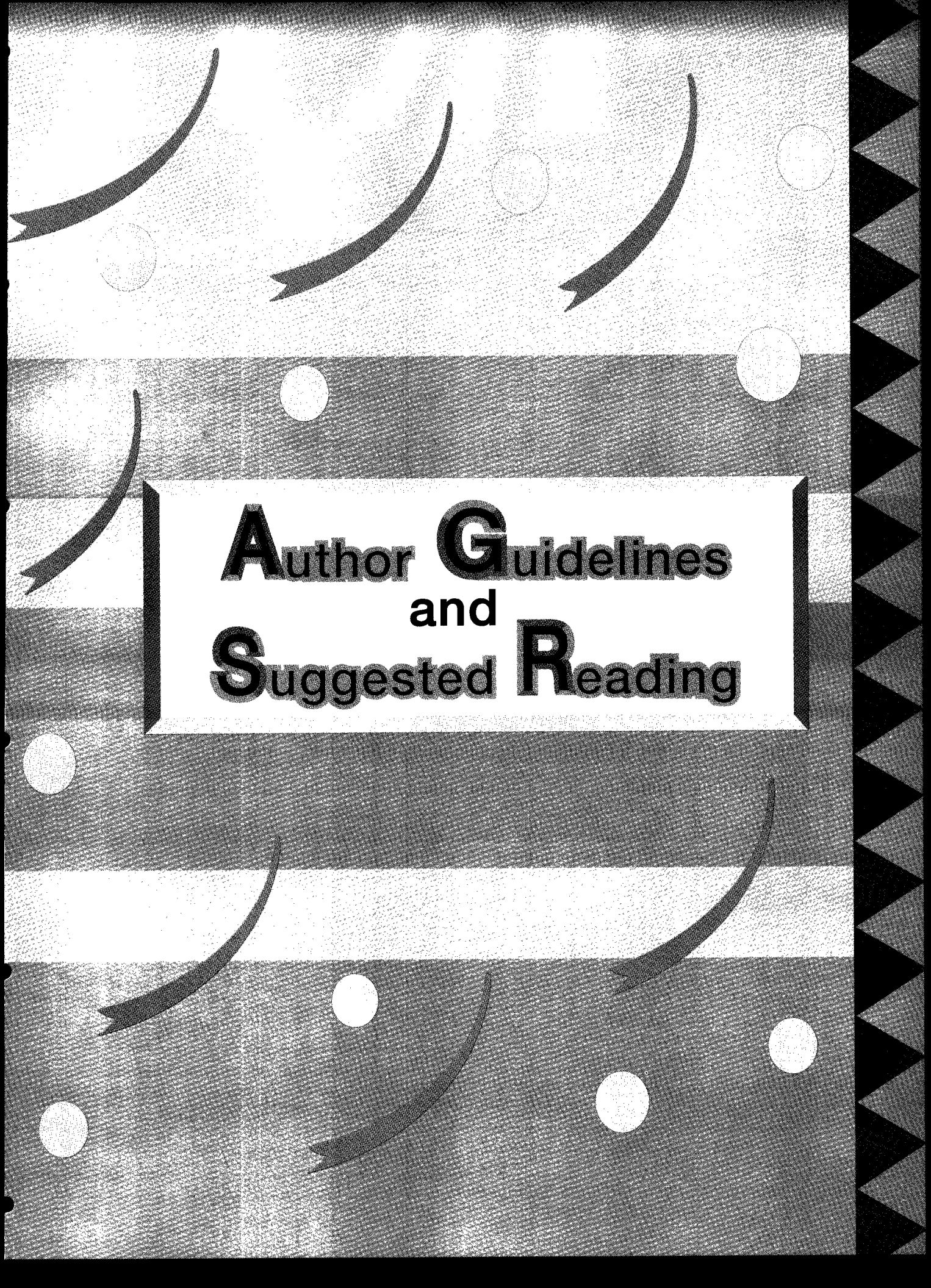
Volume 33, Number 4, 1994 *G321-0118*

IBM Systems Journal Cumulative Index 1962–1994, *p. 535.* This index includes author, subject, and abstract sections. All authors and complete titles of all published papers are listed, and each paper is categorized according to one or more topical areas. The complete abstracts of all papers are listed by volume number and issue.

Author guidelines for the IBM Systems Journal *by A. G. Davis, J. R. Friedman, and C. R. Seddon, p. 692.* Effective communication of technical work is the primary goal of the technical journal. This essay provides information about the *IBM Systems Journal* and offers guidelines for prospective authors. The *Systems Journal* and its audience are described, and

the processing of papers is discussed, along with suggestions for content and structure. To further aid the writer in preparing clear, complete papers of high quality, we include a bibliography of technical writing references.

[[Page 690 is blank]]


Author Guidelines
and
Suggested Reading