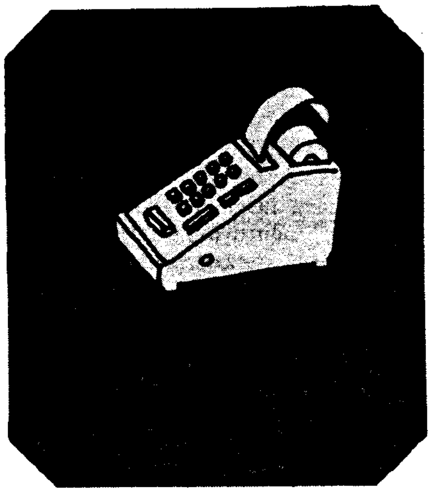
The law, as well as good business practice, requires adequate internal accounting controls. Modeling and machine analysis of office information systems can ensure compliance.

Internal Accounting Controls in the Office of the Future



Andrew D. Bailey, Jr.
University of Minnesota

James Gerlach, R. Preston McAfee, and Andrew B. Whinston
Purdue University

Although office-information-system prototypes and research articles tend to concentrate on the OIS's potential for enhancing worker productivity, the literature also expresses great concern that enriched job specifications and user decision-making accompany office automation. Both views are useful, interesting, and worthy of pursuit—as are many other perspectives on this topic.[1] This article, however, concentrates on a matter not explicitly discussed in most of the OIS literature. In addition to enhancing productivity and providing worker support, OIS designs must meet organizational control constraints. In particular, they must meet a set of conditions concerning the adequacy of internal accounting controls. These conditions are the result of both good business practice and legal requirements.

Satisfactory accountability—that is, a control structure which protects corporate assets from theft, misuse, and fraud—is an important aspect of office information system design. The OIS design goals of flexibility, efficiency, and modularity must not preclude the accountability needs of managers, stockholders, and auditors. This will require meticulous examination of the OIS system to ensure that it satisfies the multi-attribute control criteria used by auditors. Our approach is to model the firm's internal behavior, of which the OIS is a subset, using the computer-acceptable Internal Control Description Language. The model is then analyzed by machine to see that it satisfies auditing criteria. In this way, our system— called Ticom-II—incorporates both OIS theory and auditing criteria. Even though our work is not complete, we believe that early presentation is justified by the uniqueness—and importance—of considering the internal-control approach during OIS technology's current, formative stage.

Internal accounting controls

Objectives. Traditionally, accountants and auditors have been concerned about business firms' control characteristics. Several basic control objectives have been identified and codified by the American Institute of Certified Public Accountants in Section 320 of *Statements on Auditing Standards*,[2] and in drafting the Foreign Corrupt Practices Act of 1977 Congress adopted the same terminology. Organizations should

. . .make and keep books, records and accounts, which in reasonable detail, accurately and fairly reflect the transactions of the assets of the [organization] and [should]. . .

. . .Devise and maintain a system of internal control sufficient to provide reasonable assurance that—

(i) transactions are executed in accordance with management's general and specific authorizations;
(ii) transactions are recorded as necessary to permit preparation of financial statements in conformity with generally accepted accounting principles or other criteria applicable to such statements, and to maintain accountability for assets;
(iii) access to assets is permited only in accordance with management's general or specific authorization; and
(iv) the recorded accountability for assets is compared with existing assets at reasonable intervals and appropriate action is taken with respect to any differences.[3]

For companies registered with the Securities and Exchange Commission and traded in the open market, conformance to these objectives is not merely a matter of good business practice, but of the law. The Foreign Corrupt Practices Act of 1977 allows for fines and imprisonment for failure to maintain adequate internal accounting controls. The history of FCPA '77 is interesting in itself, but let us simply say that failure to develop adequate

0018-9162/81/0500-0059$00.75 © 1981 IEEE

system design, control, and analysis tools led to some highly visible frauds and firm failures. Congress reacted by passing FCPA '77.

However one views the above control objectives, anyone interested in OIS development must consider the control issue. There is little question that consideration of control conditions may mean less immediate operational efficiency, lower work-station productivity, and lower levels of work-station computer support in matters such as access to components of a distributed data base. Admittedly, the imposition of controls generally places a "drag" on system operations. But before OIS designers or operators despair of accountants' and auditors' failure to understand their needs and the organization's need for efficiency, productivity, and worker support, we suggest they step back and consider long-term organization objectives. Even without FCPA '77, no organization could maintain its competitive position long if it failed to protect its assets or could not provide accurate, reliable data to its investors and creditors. A wealth of experience has demonstrated that, without controls, assets are at risk and will be lost and accurate, reliable information will not be available.

Naturally, there are limits to the level or degree of control that can or should be enforced within an organization. At some point, costs of imposition become higher than expected benefits. Too great a decline in efficiency and productivity can slow organization response times to the point where a possible loss of assets becomes the better risk. This basic cost/benefit issue, long recognized by accountants and auditors, is given an ambiguous nod by the SEC, which demands only "reasonable assurance" for the adequacy of internal accounting controls.

Two problems remain even after accepting the cost/benefit argument. First, existing system design and evaluation tools frequently are not strong enough for adequate cost/benefit analysis. Second, it is not clear how the courts and the SEC will interpret "reasonable assurance." Accountants and auditors offer no pat answers for these problems, but a long history of practical experience has led them to a heuristically satisfying list of criteria worthy of consideration in designing OIS controls.

Criteria. The *Statements on Auditing Standards* provide some guidance for implementing basic control objectives. The categories of concern are paraphrased below.

(1) Establishment and supervision of internal control systems is a management responsibility.

(2) Absolute assurance of effectiveness is probably not cost effective; thus, reasonable assurance is acceptable.

(3) Concepts of internal control are independent of the data processing mechanism. (This applies to computer processing as well as manual processing of transactions.)

(4) Any system of control may be compromised by error, collusion, management override, or deterioration in compliance.

(5) Competent personnel of high integrity are essential to good internal control.

(6) Segregation of functions implies that those in a position to perpetrate "error" should not also perform functions enabling them to conceal those "errors." For instance, those who control assets should not also control the accounting for assets.

(7) There is a need to generate independent evidence supporting valid authorization, approval, and performance of actions.

(8) Proper documentation, recording, authorization, and approval of transactions must be maintained.

(9) Access to assets must be limited to authorized personnel.

(10) Periodic comparisons of recorded amounts to actual assets and follow-up on deviations are essential to good internal control.

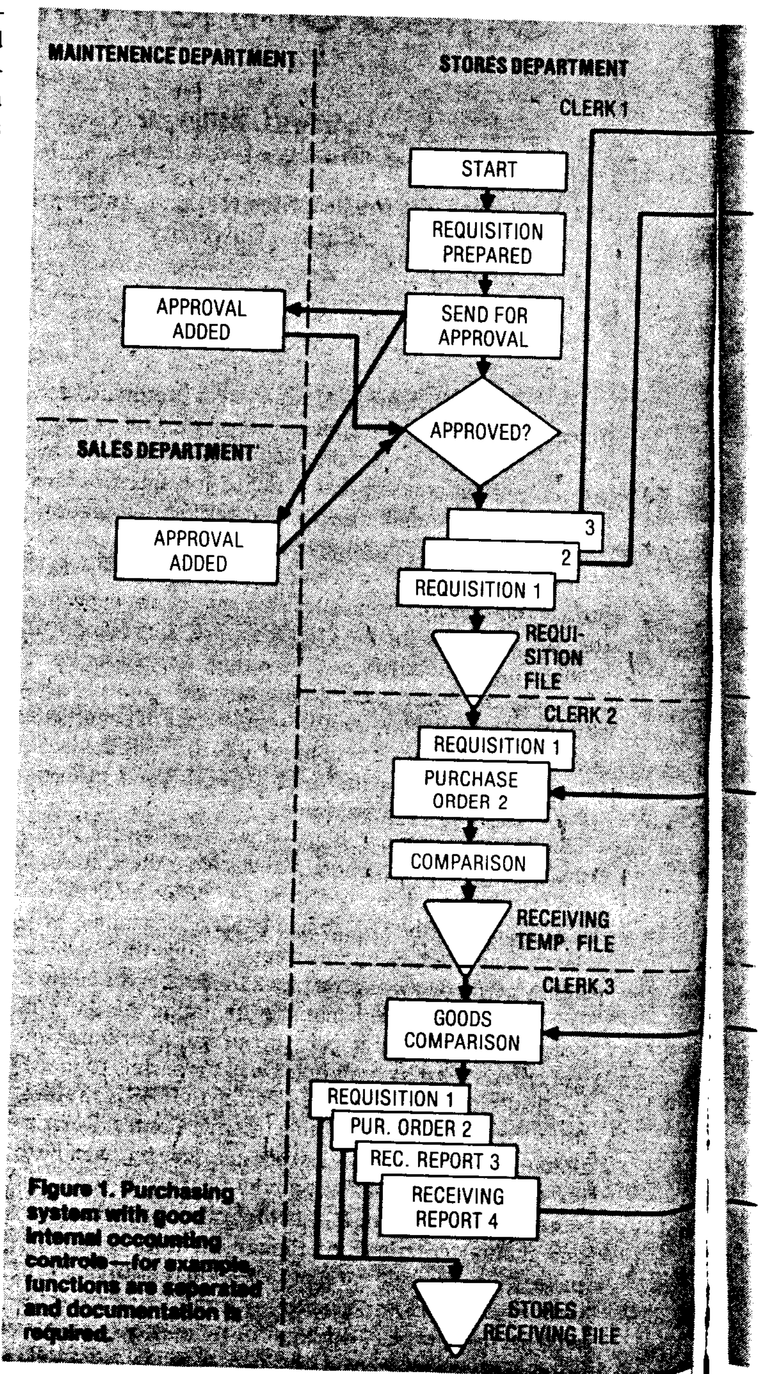
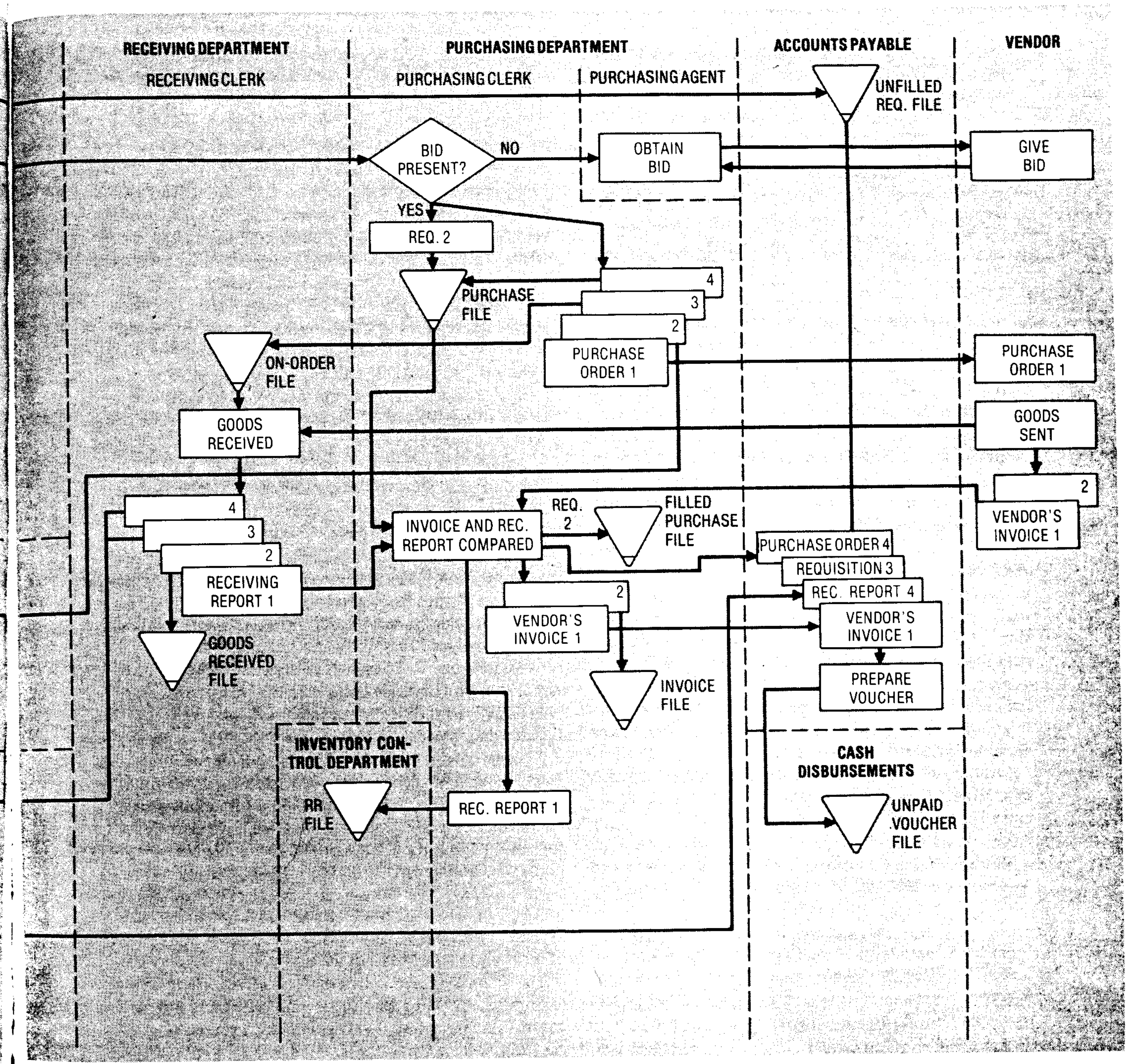


Figure 1. Purchasing system with good internal accounting controls—for example, functions are separated and documentation is required.

Items 6 through 10 are of particular interest to those involved with OIS design. To illustrate these concepts, we have selected a commonly used flowchart adapted from an auditing text by Robertson.[4] The system illustrated in Figure 1 is a manually operated purchasing activity, but each of the clerks identified in this example could perform the same tasks at a computerized work station.

Examples of criteria 6 and 7 are immediately evident: the organization has segregated such functions as the authority to requisition goods (Stores Department) from the authority to purchase goods (Purchasing Department). A further separation neatly illustrates criterion 9: the Purchasing Department has authority to purchase but lacks authority to handle assets. The Receiving Department takes delivery on the goods and subsequently turns them over to Stores, and access to assets can be further controlled by physical separation, e.g., locks and gates. At every stage of the process we find examples of criteria 7, 8, and 9.

Note that Accounts Payable, which maintains accounting control, does not have authority to requisition, purchase, or handle assets. Its segregated duties involve record maintenance based on documents (8) independently prepared at various points in the transactions pro-



cess (7). For example, a copy of the requisition is sent from Stores, a copy of the purchase order and invoice from Purchasing, and a copy of the receiving report from Receiving. In Accounts Payable, no payment will be authorized—and thus no voucher prepared—unless these independently prepared documents agree. If the documents agree, a voucher will be prepared and sent with the supporting documents to the Cash Disbursement Department for payment. Note an additional separation of duties: those who account for transactions do not handle assets, i.e., the Accounts Payable Department handles neither inventory or cash.

A post-audit of inventories illustrates criterion 10. Documents independently maintained by the Inventory Control Department, which accounts for but does not handle inventory, can be compared by the auditor to existing physical inventories maintained by the Stores Department. Any discrepancy must be explained if it exceeds normal wastage or loss, which is a cost/benefit analysis issue.

With combined Stores and Receiving Departments, a common weakness in smaller organizations, purchases cannot be as effectively controlled.

Failure to maintain adequate control creates exposures to loss. For example, if the Stores Department can prepare a requisition, authorize a purchase, and prepare a purchase order, the risk of loss through purchase of non-business goods increases. However, if the Receiving Department is independent of Stores, this compensating control assures that purchases are received and added to firm inventories. With combined Stores and Receiving Departments, a common weakness in smaller organizations, purchases cannot be as effectively controlled. While external locks and gates may preclude removal from the premises, many inappropriate acquisitions can be used or consumed on the premises. Even with control, the Stores Department might have purchases "drop-shipped" to other premises, thus bypassing the independent Receiving Department.

Examples of failure of control and the resultant loss of firm assets or data accuracy and reliability are legion. In the computer area, sources of such failures have been compiled and comprehensively documented by Parker.[5] An interesting aspect of these cases is that a "trivial" system weakness often permitted circumvention of intended controls by a determined defrauder.

Review and evaluation. The Generally Accepted Auditing Standards promulgated by the AICPA require

> . . .a proper study and evaluation of the existing internal control as a basis for reliance thereon and for the determination of the resultant extent of the tests to which auditing procedures are to be restricted.[2]

The FCPA '77 requires that such a system of internal control exist within a firm and that it be adequate for the purposes previously discussed. These requirements obviously suggest that means for the review and evaluation must either exist or be developed.

Currently, requirements are met by employing traditional flowchart descriptions such as that in Figure 1, questionnaires with a focus on potential weaknesses and exposures, narrative descriptions, decision tables, and other methods that seem appropriate to the individual auditor in each situation. The review and evaluation stage of this process is largely limited to human analytic capabilities, which are surprisingly deep, broad, and complex—but extremely varied and unreliable.

New techniques are needed to support the auditor in the review and evaluation process. Many of these techniques will take advantage of computer technology and its capacity to deal swiftly and accurately with highly complex systems of relationships. Research may ultimately lead to the application of artificial intelligence models to component activities or even to complete audits.

Current office modeling techniques

In a recent survey article on OIS, Ellis and Nutt emphasized the need for analytic modeling.[6] To substantiate their claim, they cited arguments concerning the formative state of OIS technology, the dynamic demands and requirements of the office environment, and the absence of a comprehensive OIS theory. Finally, they expressed the belief that offices of the future would rely heavily on formal models suitable for theoretical analysis.

Various OIS modeling techniques have been proposed. Three such models are Zisman's augmented Petri nets,[7] the Information Control Net model developed at Xerox PARC,[8] and Omega.[9]

The ICN model has been used to quantify information flows and to suggest reconfigurations for improving office performance. However, since OIS performance and internal control are not necessarily complementary design philosophies, modifications to optimize OIS task performance may compromise the internal control mechanisms. Before applying reorganization transformations to enhance efficiency, the benefits of these modifications should be weighed against the costs, if any, in the internal control structure. Ticom-II provides a method of establishing where these costs are sustained. In this way, proposed OIS modifications can be evaluated by both efficiency and internal control criteria.

The Omega approach has several important attributes. First, because it incorporates information of all finite higher-order logics, it is as general as a formal modeling method is ever likely to be. Second, Omega can encode information about itself, thus providing detailed information about both the office and the office description. The result is extreme flexibility in modeling the OIS and an extremely powerful data base method. Obviously there is an implicit cost to this generality: the astronomic complexity of information analysis within this system.

The Ticom system has different objectives. Ticom aims to minimize the cost associated with answering internal control issues; consequently, we have avoided adding generality where it was not justified by auditing needs.

Interestingly, one auditing firm—Peat, Marwick, and Mitchell—has developed an auditing procedure bearing a conceptual similarity to Ticom-II. PMM's SEADOC—Systems Evaluation Approach: Documentation of Controls[10]—is an entity- and exchange-based view of transaction processing systems. With its objects (entity), operations (exchange), and transactions processing capabilities, it adopts a position very similar to that of Ticom-II. The level of abstraction concepts that are an integral part of Ticom-II are also reflected in the PMM discussions and presentations of SEADOC. PMM, however, has not as yet proposed a computer-based description or evaluation process similar to that of Ticom-II.

The Ticom-II modeling and analysis system

The Ticom-II modeling and analysis approach to unifying OIS and auditing issues has four distinct components. The first, the Internal Control Description Language, is a modeling language for formally describing a firm's operations. ICDL is designed for use by general business personnel in describing their duties. The formal input it provides for the Ticom-II modeling process is consistent with the information collected by Deloitte, Haskins, and Sells[3] in their manual verification of internal control procedures.

This formal model is then mapped algorithmically into an internal representation, the second component of the system. As part of the mapping process, the ICDL description is checked for compliance with syntactic and semantic rules of the language to reduce the probability of misrepresentation errors. In addition, certain consistency issues concerning illogical construction sequences, e.g., documents transferred but not received, are examined.

The internal representation was designed to facilitate analysis, the third major component of Ticom-II. Because the auditor's internal control model corresponds to a directed graph, both graph theoretic methods and conventional auditing techniques are employed.

The analysis methods are controlled by the fourth component, a query processing system that permits the auditors to pose questions concerning the internal control model's behavior.

ICDL. The Internal Control Description Language was designed to support descriptive specifications of accounting internal control systems. Its constructs and terminology are closely related to the fundamental concepts and operations associated with internal control and systems design, and it is rigidly specified to facilitate mechanical recognition and unambiguous interpretation. We intended to develop a language rich enough to support system descriptions that are easily readable and intuitively appealing to business personnel, while avoiding the recognition and interpretation difficulties inherent to natural languages. Although further human engineering is required, our first experience in developing the language and employing it to model textbook internal control systems indicates that our formulation is sufficiently comprehensive.

In the ICDL, the internal control system description consists of definitions of system objects, repositories, and interrelated office tasks associated with the agent responsible for carrying out the tasks. The task descriptions focus on modeling the processing required to record single transactions. A transaction is defined as the action required to move a set of system objects from their source repositories to their destination repositories, destruction, or error-state detection. Knowledge of how single transactions are processed can then be related to situations concerning concurrent processing of transactions.

Abstractions. Since internal control systems are not limited to controlling documents and electronically stored records (i.e., cash, inventories, goods received, etc.), the ICDL incorporates a general mechanism for specifying system objects as abstract types. System objects are also abstractions since the internal control system model does not physically manipulate instances of the system objects, but only models the operations and their effects on the state of system objects.

Although further human engineering is required, our first experience in developing the language and employing it to model textbook internal control systems indicates that our formulation is sufficiently comprehensive.

The notion of abstraction is widely used. Programming languages use abstract data types to specify the functional properties of a data structure and the permissible operations on it; subsequent use of a type is done within the context of its specification.[11] Artificial intelligence applications use type hierarchies in conjunction with assertions to form a semantic network.[12] ICDL uses the inheritance mechanism of type hierarchies common to such implementations.

Since ICDL is a modeling language, references to real objects, agents, and repositories are modeled as references to abstract objects, agents, and repositories representative of their real-world counterparts. Thus, shipping clerks John and Mary can be collectively modeled in ICDL by the abstract agent CLERK associated with the Shipping Department. The task descriptions associated with CLERK are assumed to be the tasks performed by all shipping clerks. In a more concise way, we could say that John and Mary are of type SHIPPING.CLERK. The ICDL addresses this abstract level of collective representation and typing.

In ICDL, abstractions are defined by *labels* and *types*. The symbolic name of any abstraction is called a label. For example, let *object* PAYROLL-CHECK be representative of payroll checks in general. In addition, PAYROLL-CHECK is declared of *object type* CHECKS since it possesses the *attributes* necessarily belonging to check-like objects. The properties of checks are defined by the label CHECKS and a list of attributes. Two such attributes are PAYEE and DATE-PAID. To further complete the definition, PAYEE and DATE-PAID are typed to specify the "nature of the information" these attributes convey. PAYEE and DATE-PAID are declared to be of *attribute type* NAME and DATE, respectively. The definition is completed by declaring both attribute types, NAME and DATE, to be of scalar data types, in this case "character." This implies that character data can be used to represent instances of PAYEE and DATE-PAID, but because these attributes are of different attribute types, they convey dissimilar information. (Note, in general an attribute may represent a single entity, e.g., PAYEE, or a group of entities, e.g. ITEMIZED-ORDER-LIST.)

Thus, PAYROLL-CHECK is defined as an object of type CHECKS with attributes PAYEE and DATE-PAID. Attributes PAYEE and DATE-PAID convey NAME and DATE information, respectively. And this information can be represented using character data. This typing scheme is essential for validating comparisons (i.e., comparison of two objects or attributes is valid if they are of the same type), checking the consistency of other system operations, and establishing a knowledge base for future man-machine interaction.

In summary, an *object* is a labeled item with attributes indicated by its object type. An *object type* is a label denoting the attributes necessarily associated with objects of a given type. An *attribute* is a labeled characteristic with an associated attribute type. The *attribute type* specifies the data type that can be employed to represent an instance of an attribute. In essence, the attribute type labels the "nature of the information" an attribute conveys. The "nature of the information" conveyed by an object is the collective "natures" of its attributes.

Repository/agent declarations. The typing scheme is also extended to include repository and agent declarations. Suppose VAULT is declared a repository "holding" only objects of type CHECKS. Then the instruction GET PAYROLL-CHECK FROM VAULT is accurate, but GET CASH FROM VAULT would be inaccurate, assuming CASH is not declared to be of type CHECKS. Also, TRANSFER PAYROLL-CHECK TO SUPERVISOR is accurate only if SUPERVISOR is declared to be an agent with a matching statement denoting the reception of a payroll check. This type of consistency check is generally done by auditors in their internal control evaluation.

In addition to facilitating consistency checking (functional correctness) by type checking, this abstraction permits the system designer to choose a specification level that emphasizes the significant properties of the objects while ignoring other less significant aspects. The ICDL has provisions for allowing the system designer to define classifications for system objects, repositories, and agents. These classifications form a type hierarchy to represent membership relations. Objects, repositories, and agents may belong to many classifications which are not necessarily mutually exclusive.

The type hierarchy permits expression and interpretation of high-level queries. For instance, consider that object type CHECKS has been declared an element of the classification DOCUMENTS and also an element of the classification ASSETS. A query concerning the general use of indirect assets (objects that have no material value in themselves but can be used to acquire objects of value) can be referenced in the query as DOCUMENT (x) AND ASSETS (x), where x is a variable. An instance of x satisfying these restrictions is PAYROLL-CHECK; since PAYROLL-CHECK is of type CHECKS, it inherits the membership relations associated with CHECKS.

The depth of the type hierarchy is unrestricted, permitting various degrees of subset inclusion. Classifications of agents and repositories are handled in the same manner. Presently, no mechanism has been defined for checking the type hierarchy for contradictions, e.g., classification of an object as both an asset and a liability. Also, due to the level of standardized terminology and concepts in accounting, research is needed to formulate a skeleton type hierarchy as an initial base which can be tailored and expanded to fit the particular organization's needs.

COMMAND	DESCRIPTION
ASSIGN attribute-list OF object;	Specifies which attributes of a particular object are to be assigned values.
MODIFY attribute-list OF object;	Specifies which previously assigned attributes of a particular object are to be reassigned values.
DESTROY object-list;	Specifies which objects are to be destroyed.
IF boolean-expression THEN true instructions ELSE false instructions END IF;	Specifies a boolean expression, a simple or compound comparison, whose truth determines which disjoint set of instructions are to be performed next.
TRANSFER object-list TO agent;	Specifies that the data objects listed are to be transferred to another agent or organizational unit.
WAIT FOR object-list;	Specifies that the agent's processing is blocked until the objects listed are received.
PUT object-list INTO repository;	Specifies that the objects listed are to be placed into the designated repository.
GET object-list FROM repository;	Specifies that the objects listed are to be retrieved from the designated repository.
COPY target-object GIVING duplicate object-list;	Specifies that the target object is to be copied creating the designated duplicate objects.
END TASK;	Specifies the end of a task.
REVIEW;	Signifies the entrapment of an error or a discrepancy.

Figure 2. ICDL commands for describing individual office tasks.

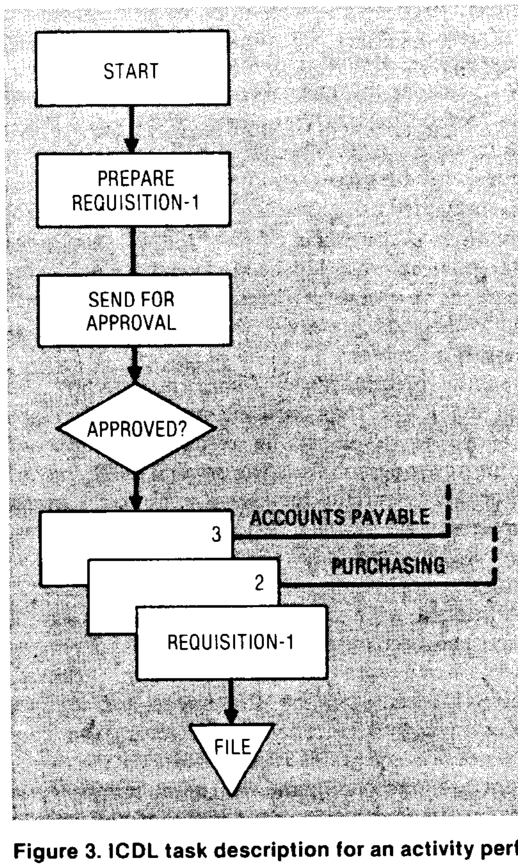
Task descriptions. Office activity is described by modularized descriptions of interrelated tasks. The individual tasks are associated with work-related positions that are classified within departments, divisions, and other organizational units. The functional capabilities of each organizational unit are defined as the sum total of the functional capabilities of its individual members. In this way, the system can be described and analyzed from several organizational perspectives.

The basic commands for describing individual office tasks are shown in Figure 2. Figure 3 illustrates a task description written in ICDL for describing the activity performed by clerk 1 in the Stores Department. To the left of the ICDL description is the flowchart representation of clerk 1's activity extracted from Figure 1. Note that the task description is not required to be deterministic. For instance, the send for approval activity stipulates only that REQUISITION-1 (i.e., copy 1) is to be sent to MANAGER (i.e., one manager (agent) for each department), and no stipulation is specified for determining which manager is to receive the requisition. Similarly, the transfer of REQUISITION-3 to Accounts Payable does not designate the agent who is to receive the document. Consequently, any agent in Accounts Payable with a wait command for the requisition could possibly receive it. Other cases of nondeterminism appear if a document is filed and several agents have the capability of extracting it from the file or one agent extracts it several times during processing.

Once the system description has been mechanically analyzed for inconsistencies and flaws, i.e., type checking, the description is interpreted into an internal representation expressing permissible orderings of the operations. The intertask precedence of the instructions is inferred through the matching of PUT/GET and WAIT FOR/TRANSFER instructions and the serial ordering of the instructions. Nondeterminism is removed by encoding mutually exclusive precedence constraints permitting the "execution" of all possible paths through the system.

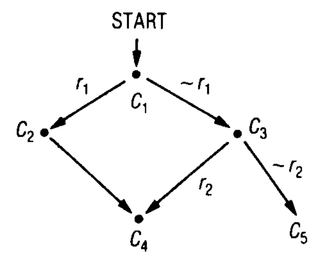
Internal representation. The Ticom-II ICDL is mapped into an internal representation to facilitate query processing and simplify system analysis. The internal representation is designed explicitly to contain information implicit in the ICDL description of duties.

Consider a command in the ICDL—for example, GET x FROM f. This command is identified by the term GET, two operands x and f, and the agent p who executes this command. In general these four pieces of information (executor, command, operand one, operand two) are sufficient to describe the computational information explicitly contained in the Ticom ICDL description. Implicit in this description, however, is the partial order that temporally relates the commands in the description. For example, in Figure 1, "prepare voucher" must follow "goods comparison." And any GET or WAIT FOR command must follow the corresponding PUT or



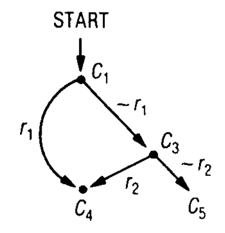
```
GET REQUISITION-1 FROM REQUISITION-SUPPLY;

ASSIGN SOURCE, DATE, ITEMIZED-DESCRIPTION
    OF REQUISITION-1;

TRANSFER REQUISITION-1 TO MANAGER;
WAIT FOR REQUISITION-1;

IF REQUISITION-1. APPROVAL NOT IN AUTHORIZATION-CODES THEN
    REVIEW

ELSE
    COPY REQUISITION-1 GIVING REQUISITION-2, REQUISITION-3;
    TRANSFER REQUISITION-3 TO ACCOUNTS-PAYABLE;
    TRANSFER REQUISITION-2 TO PURCHASING.CLERK;
    PUT REQUISITION-1 INTO REQUISITION-FILE;
    END IF;

END TASK.
```

Figure 3. ICDL task description for an activity performed by clerk 1 in the Stores Department.



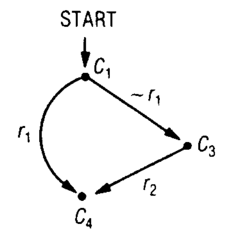
$PC_1 = \{ (\text{START}) \}$
$PC_2 = \{ ((1), r_1) \}$
$PC_3 = \{ ((1), \sim r_1) \}$
$PC_4 = \{ (2), ((3), r_2) \}$
$PC_5 = \{ ((3), \sim r_2) \}$

(a) A simple graph.



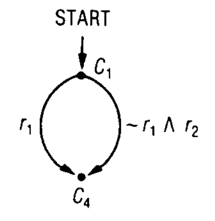
$PC_1 = \{ (\text{START}) \}$
$PC_3 = \{ ((1), \sim r_1) \}$
$PC_4 = \{ ((1), r_1), ((3), r_2) \}$
$PC_5 = \{ ((3), \sim r_2) \}$

(b) Elimination of C_2.



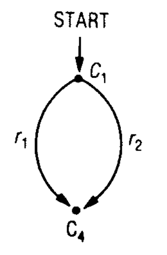
$PC_1 = \{ (\text{START}) \}$
$PC_3 = \{ ((1), \sim r_1) \}$
$PC_4 = \{ ((1), r_1), ((3), r_2) \}$

(c) Elimination of C_5.



$PC_1 = \{ (\text{START}) \}$
$PC_4 = \{ ((1), r_1), ((1), \sim r_1 \wedge r_2) \}$

(d) Elimination of C_3.



$PC_1 = \{ (\text{START}) \}$
$PC_4 = \{ ((1), r_1), ((1), r_2) \}$

(e) Logical simplification of PC_4.

Figure 4. Reduction of a simple graph to eliminate all nodes except those identified by the query—that is, the condition under which both C_1 and C_4 occur.

TRANSFER command, respectively. To encode this information explicitly, a precedence constraint set called a PC set is introduced. First, number each command with any arbitrary numbering scheme. To calculate the PC set for command i, denoted PC_i, let $((n_1, \ldots n_k), r_1 \wedge \ldots \wedge r_\ell) \in PC_i$ if and only if command i follows commands n_1 and $n_2, \ldots, n_k$ under the condition r_1 and r_2 and . . .r_ℓ. If $((n), r_1) \in PC_i$ and $((m), r_2) \in PC_i$, command i can follow command n given r_1 is true, or can follow command m given r_2 is true. Thus, for example, if a GET command i must unconditionally follow both the preceding command, n, and the PUT command, m, that permits the GET to find something in the repository; this is encoded by $((n, m)) \in PC_i$.

The conditions r_j can have two possible sources. First, any time an IF. . .THEN statement occurs, a conditional r_j is assigned if the condition is true, with $\sim r_j$ true otherwise. Second, when a TRANSFER or PUT can satisfy more than one WAIT or GET command, respectively, ficticious conditionals r_k are invented to describe which WAIT or GET is satisfied. For example, if a clerk processes a document twice, then when that document is transferred to him, he might use this transfer to satisfy either of his two WAIT commands. Thus, an r_k is created to mean he satisfies one of them under condition r_k and the other under condition $\sim r_k$. Graphical examples of these issues are provided in the next section.

Because the partial ordering of the ICDL description is explicitly encoded in the internal representation, analytic procedures can operate directly on the internal representation. In particular, commands can be eliminated and all relationships of the remaining commands preserved solely by operating on the PC sets.

Analyzing the internal representation. The procedure used for analyzing the internal control model is a hybrid analysis method derived from graph theory and traditional auditing methods. The auditors' flowchart models, analogous to the representation incorporated in ICDL, can be represented using graphs. Auditors generally proceed by dividing the model of the firm into subgraphs called accounting cycles,[10] which incorporate the information on the relationships of a subset of instructions, ignoring all others and preserving the graph theoretic relationships of the subset.

This motivates the approach Ticom uses to analyze the ICDL model. Rather than choosing particular subgraphs in advance, we developed an algorithm capable of generating any subgraph from the original ICDL model. The algorithm essentially removes a node in the graph while preserving the partial order of the remaining nodes. This permits generation of any subgraph. The proof that this algorithm works is a graph theoretic result, although, as we mentioned, it is analogous to the auditor's studying subcycles. The mechanism for choosing nodes for removal is based on queries to the system. Formally, of course, the analysis is executed on the ICDL model and not on a graphical model. Theoretically, however, the correspondence between graphical models and ICDL representations is one to one, permitting examination of the graphical model to serve as a foundation of the analytic procedure.

A simple example will serve to illustrate the reductions. In Figure 4(a), we have a very simple graph that might correspond to part of a control structure, where the nodes C_i are commands (instruction instances) and a split corresponds to a conditional IF. . .THEN, with the hypothesis satisfied under condition r_i and not satisfied under condition $\sim r_i$. Suppose the auditor wishes to know the condition under which both C_1 and C_4 occur. Eliminating all nodes except those identified by the query produces the graphic display of the solution, which is r_1 or r_2. The removal of a node from the graph, corresponding to Figure 4(a)-(d), is called a contraction. Figure 4(e) illustrates a simplification wherein ($\sim r_1$ and r_2) or r_1 is reduced to r_1 or r_2. Obviously, complex rules are necessary to deal with situations: (1) where cycles may exist, (2) where two commands (nodes) must be satisfied before a third can begin—for example, in the case of a WAIT statement embedded in a task description, or (3) where a given transfer can satisfy more than one WAIT. While these are more complex, they have been shown to be mechanically analyzable.[13]

This brings up an important methodological issue in decision support systems. The notion of an algorithm's complexity formalizes a "worst case" relationship between the algorithm's operation and the input size. Since most general problem-solving algorithms are extremely complex, in some situations they will be very expensive to operate. Thus other, more limited problem-solving is used. Ticom falls into this class. Its complexity, compared to more general problem-solvers, was substantially reduced by constructing the analysis algorithm to answer only questions regarding internal control, e.g., access to assets and separation of duties. In addition, the problem environment justifies certain simplification rules which, while preserving the internal control relationships, reduce the complexity of the subgraph being investigated. The formal statement of these algorithms, simplification rules, complexity results, and theoretical proofs are available elsewhere.[13]

Query processing. The essential feature of query processing is the linkage that controls the analytic procedures. A query can involve several individuals or types of individuals (e.g., clerk 1 or any member of Stores) and particular documents or types of documents. Using the type hierarchies and abstractions previously discussed, cited items can be linked to all instructions involving those items, thereby identifying instructions whose execution is irrelevant to the specific query. These instructions can then be eliminated via the contraction and simplification procedures described above.

The instructions of analytic interest (not to be contracted from the system) are identified by the query. Consider the following query concerning authorization to pay vendors by check:

WHEN (TRANSFER *y* TO VENDOR BY *q* AND CHECKS (*y*) AND NOT ASSIGN APPROVAL OF *v* BY *p* AND VOUCHERS (*v*))?

The critical commands are located by unifying system instructions against the query. Thus all transfer statements of the type transfer *y* to vendor where *y* is of type checks

Operating System Manager

Direct experienced professionals developing a new real-time distributed processing operating system at Bell Laboratories in Denver.

Your role will focus on planning the evolution of the operating system for use with new and existing applications and products while maintaining high standards for system quality and performance. You'll have ample opportunity to contribute your expertise while directing a highly motivated team.

The innovative professional we're seeking must possess an MS or PhD in Computer Science, Electrical Engineering or related discipline. Proven ability in managing operating system projects, expertise in state-of-the-art operating system concepts, and at least five years development experience in commercial operating systems is essential.

Bell Laboratories offers advanced computer facilities, excellent libraries and strong technical/staff support. Our stimulating environment promotes creative freedom and professional growth.

A written response is required. Please send resume, references, and other pertinent data to: Mr. R. A. Parker, Bell Laboratories, Center 7831-123, 600 Mountain Avenue, Murray Hill, NJ 07974.

Bell Laboratories

An equal opportunity employer

are identified as critical instructions. Instructions matching the query's approval instruction have their PC sets modified to FALSE. This prevents the approval from occurring on the path, as demanded by the query. After this, all the noncritical instructions are contracted from the model. Then, for each critical command, in turn, the other critical commands are contracted from the model. The PC set of the selected critical command gives the necessary conditions for that instruction to occur. This procedure is replicated for each critical command. The setting of the PC sets to FALSE for the approval instructions logically blocks all system paths that depend on their occurrence.

The result of this procedure is a substantially simplified subsystem which preserves the properties of the original system with respect to the query. Consequently, it is now only necessary to check whether, and under what conditions, the state mentioned by the query can occur. In this manner, the conditions necessary for the query to be true are generated.

Ticom's complexity, compared to more general problem-solvers, was substantially reduced by constructing the analysis algorithm to answer only questions regarding internal control.

In an extended example, we checked whether a particular asset could be released without management authorization,[13] a typical internal control issue providing an excellent test of the analytic procedures. The result was that, if a given clerk did not perform his task in the manner expected, the asset could be released. Significantly, this query involved only two commands—the command which released the asset and the command which authorized the asset. When all of the other commands were contracted out of the system, it was clear that the asset could be released with authorization under some conditions but without it under others. The latter constituted an internal control failure systematically deduced by Ticom-II type analysis. In addition, a solution to the failure, involving separation of a single clerk's duties, was self-evident from the information provided by the analysis.

The query processing system provides the last link in the Ticom-II approach. So far, we have concentrated on constructing necessary conditions for the query to be true, not on providing the most interpretable formulation of these conditions; further work is needed on the human engineering aspects of the answers Ticom provides. Besides the human engineering needed here and in the first component, there are other exciting research areas—on both the auditing and OIS ends of the system—that we are investigating. These are discussed in the next section.

Further research issues

Software engineering encompasses system definition, documentation, correctness and verification, and implementation and maintenance with some particular orientation. Our intended use of the Ticom model includes these goals, plus extensions addressing real-time controls and information retrieval capabilities for work in progress (i.e., documents being processed) as well as for completed work. The OIS we envision has five development stages, each one built on the foundation of previous stages and centered around the internal control model. The plan, comprising a complete top-down design, implementation, control, and management methodology, includes

(1) office model description,
(2) office analysis,
(3) implementation of office tasks,
(4) model-driven monitor, and
(5) control and management of work in progress and completed work.

This article has addressed the first two stages, considered theoretically implementable. The third stage concerns the implementation of the modeled system, which can be viewed as a high-level description of the actual implementation. The implementation would begin by defining the boundaries of the system. To become an integral part of the OIS, system components must be grounded to physical structures. That is, electronic forms must be specified and linked to the object definitions specified in the model. Likewise, repositories must be grounded to data base definitions and agents to individual employees. Once this is accomplished, office activities can be implemented using the task description of the model as a macro definition of the task with its primary inputs and outputs already defined. Interactively, the responsible clerical worker or office analyst would decompose each composite step into more primitive steps until each step is irreducible.

To clarify, the ASSIGN command only indicates what portion of the document is to be filled in. It remains to be stated how the required information is to be calculated, edited, and entered. The source of the information could come from another form associated with the transaction or from a local data base not originally a part of the system description. The final procedure would comply with the internal control model if it accessed only system resources available to the task, as indicated by the model, and modified these resources according to the task specification. This use of OIS resources not specified in the internal control model would be permissible but would not be immediately suspected of violating the internal control mechanism. A similar decomposition process specifying business procedures has been incorporated into the document-oriented Business Definition Language developed by IBM.[14]

The fourth stage involves the control and analysis of an operating OIS. A model-driven monitor, complete with implemented office tasks, could enforce compliance of operations with the internal control model. The monitor would trace the progress of each transaction and office procedure and automatically initiate or execute the procedures at the appropriate times. Thus, the computer and the work-station operator would interact in a joint undertaking, with the monitor enforcing compliance with the internal control model. The basic idea of a model-driven monitor was implemented in Zisman's SCOOP, System for Computerization of Office Processes, at the University of Pennsylvania's Wharton School.[7]

This fourth stage requires real-time operation of Ticom. Because firms are beginning the automation of many activities—e.g., inventories, purchasing, and payments—execution of stage four is quickly becoming necessary. Significantly, implementing this stage would create a continuous audit.

Stage five concerns development of a centralized control and information management facility incorporating the internal control model and actual definitions and formats of system resources, i.e., documents and data bases. The user, manager, or auditor would have data base inquiry capabilities to completed work and work in progress, thus enabling him to quickly locate and view critical work in progress, estimate completion times, and assign priorities to speed processing of particular tasks or classes of tasks. Auditors could spot check and trace live transactions, verifying office procedures without the knowledge of work-station operators. This auditing task is virtually impossible today.

By specifying a capability set of managerial actions, it is possible – in theory – to analyze the potential for control failure in the presence of management override.

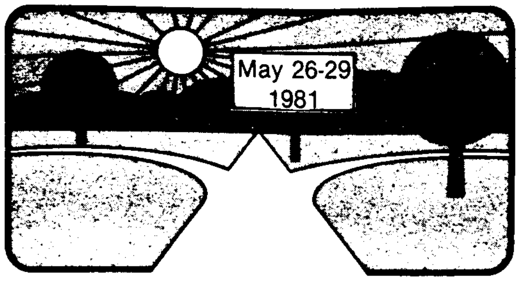
A final research topic is of possible interest to auditors, although it has not yet reached OIS developers. In most firms and in OIS proposals, management can alter system operation by edict. Management override presents a severe challenge to the internal control structure in that possible management alternations of the control structure are not limited. Thus, assessing the probability and significance of a control failure is difficult in the presence of management override.

By specifying a capability set of managerial actions, it is possible—in theory—to analyze the potential for control failure in the presence of management override. However, the combinatorial nature of this possibility set may rule out this approach. Additional information from auditors, establishing their concerns precisely, is necessary to focus the management override issue.

Alternately, the model-driven description of the system, part four of the plan, could deal with management override in a different fashion. It should be tractable to analyze the implications of management overrides as they occur and in this way avoid the combinatorial problem. As a result, the auditor need not be overly fearful of management override defeating the firm's internal controls; the analysis, based on preplanned queries provided by the auditor, could prevent this by alerting authorities as internal flaws arose.

We believe that OIS research possesses great potential in its likely impact on the design and operation of business. The unique controls-oriented perspective of

1981
THE ELEVENTH
INTERNATIONAL SYMPOSIUM ON
MULTIPLE-VALUED LOGIC



LINCOLN PLAZA INN
OKLAHOMA CITY, OK 73105

PROGRAM

May 26	
6:00- 8:00 p.m.	Registration and Social Hour
May 27	
8:00 a.m.	Registration
9:00 a.m.	Welcome Address
9:15 a.m.	Session I -Keynote address, Karpovsky, Hurst
10:30 a.m.	Session IIA -Hardware Applications Session IIB - Fuzzy Algebras
1:30- 3:30 p.m.	Session III - Invited Session on Applications and Circuits
3:45- 5:15 p.m.	Session IVA -Testing and Fault Diagnosis Session IVB - Applications of Fuzzy Logic
7:00 p.m.	Technical Committee on Multiple-Valued Logic
May 28	
8:30 a.m.	Session V - Invited Session on Fuzzy Logic
10:15-11:45 a.m.	Session VIA - Algebraic Topics Session VIB - Fuzzy Topics
1:00- 2:00 p.m.	Plenary Session
2:00- 4:00 p.m.	Session VIIA -Special Topics on Fuzzy Logic Session VIIB -Multiple-Valued Logic Circuits
May 29	
8:30-10:30 a.m.	Session VIII -Invited Session on Logic
10:45-12:45 p.m.	Session IXA -Algebraic Theory Session IXB -Circuits/Systems/Applications

ADVANCE REGISTRATION

Through May 12, IEEE members	$80.00
nonmembers	$90.00
After May 12, IEEE members	$90.00
nonmembers	$95.00
Students	$15.00

HOTEL RESERVATIONS DIRECT: LINCOLN PLAZA INN, Attn: Mr. Tom Ashinhurst, 4345 Lincoln Blvd., Oklahoma City, OK 73105 Telephone (405) 528-2741. Single $36.00, Double $44.00, through May 12.

IEEE *IEEE Computer Society* *ISMVL-81* *The University of Oklahoma*

Ticom-II contributes to these developments. We hope that OIS researchers will become aware of its existence and, when possible, incorporate it in their work. ■

Acknowledgments

The authors are indebted to Dr. Clarence Ellis of Xerox PARC for his support of the concept that auditing and internal control should play a key role in automated office design.

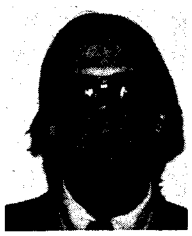
The research was supported in part by Army Contract No. DA 79C0154 and by a grant from the Peat, Marwick, Mitchell Foundation through its research opportunities in auditing program; however, the views expressed are those of the authors and do not necessarily reflect those of the sponsoring organizations.

References

1. A. D. Bailey, Jr., James Gerlach, R. P. McAfee, and A. B. Whinston, "Office Automation," *Handbook of Industrial Engineering*, Gavriel Salvendy, ed., (John Wiley & Sons, Inc., New York, forthcoming).
2. *Statements on Auditing Standards,* American Institute of Certified Public Accountants, AU Section 320, Nov. 1972.
3. Deloitte, Haskins & Sells, "Internal Accounting Control," *An Overview of the DH & S Study and Evaluation Techniques*, New York, N.Y., 1979.
4. J. C. Robertson, *Instructor's Guide to Accompany Auditing*, revised edition, Business Publications, Inc., Dallas, Tex., 1979, p. 213.
5. D. B. Parker, *Crime by Computer*, Charles Scribner's Sons, New York, 1976.
6. C. A. Ellis, and G. J. Nutt, "Office Information Systems and Computer Science," *ACM Computing Surveys*, Vol. 12, No. 1, Mar. 1980, pp. 27-60.
7. M. D. Zisman, "Representation, Specification and Automation of Office Procedures," PhD dissertation, Wharton School, University of Pennsylvania, Philadelphia, 1977.
8. C. A. Ellis, "Information Control Nets: A Mathematical Model of Office Information Flow," *ACM Proc. Conf. Simulation, Modeling and Measurement of Computer Systems,* Aug. 1979, pp. 225-240.
9. G. Barber and C. Hewitt, "Research in Workstation Network Semantics," Working Paper, MIT, Cambridge, Mass., 1980.
10. *System Evaluation Approach: Documentation of Controls,* Peat, Marwick, and Mitchell, New York, N.Y., 1980.
11. M. Shaw, "Abstraction, Data Types, and Models for Software," *ACM Proc. Workshop on Data Abstraction, Databases and Conceptual Modelling,* June 1980, pp. 189-191.
12. J. G. Carbonell, "Default Reasoning and Inheritance Mechanisms on Type Hierarchies," *ACM Proc. Workshop on Data Abstraction, Databases and Conceptual Modelling,* June 1980, pp. 107-109.
13. A. D. Bailey, Jr., James Gerlach, R. P. McAfee, and A. B. Whinston, "The Theoretic and Analytic Capabilities of TICOM-II," Working Paper, Purdue University, West Lafayette, Ind., revised Feb. 1981.
14. M. Hammer, W. G. Howe, V. J. Kruskal, and I. Wladawski, "A Very High Level Programming Language for Data Processing Applications," *Comm. ACM,* Vol. 20, No. 11, Nov. 1977, pp. 832-840.

Andrew D. Bailey, Jr., is a professor and chairman of the Department of Accounting in the College of Business Administration at the University of Minnesota. He has published in *The Accounting Review, Journal of Accounting Research, The Journal of Financial Management, Accounting and Finance,* and *Management Accounting.* Current teaching and research interests include auditing, computers in auditing, auditing and statistical sampling, and quantitative methods in management accounting. Bailey, who is a CPA and CMA, received his PhD degree from Ohio State University in 1971.

James H. Gerlach is a PhD student in management science at Purdue University. His major area of study is management information systems with a minor in accounting. His current interests include programming languages, artificial intelligence, data base, and auditing. Gerlach holds a BS in mathematics-computer science from St. Joseph's College and an MS in computer science with specialization in programming and operating systems from Purdue University.



R. Preston McAfee is a visiting assistant professor of economics at Purdue University. His research involves mathematical economics, decision theory, and information systems. He received the BA in economics at the University of Florida in 1976. He earned the MS in mathematics and the MS in economics, both in 1978, and the PhD in economics, 1980, at Purdue University. He is a member of Phi Beta Kappa.

Andrew B. Whinston is a professor of management, economics and computer science at Purdue University. He has published over 150 papers and five books on subjects ranging from auditing accounting systems and game theoretic models of resource allocation to data base management and decision support systems. Presently he serves on the board of directors of Micro Data Base Systems, Inc., Lafayette, Indiana. Whinston received a BA degree from the University of Michigan and a PhD in management from Carnegie-Mellon University.