Data Base Engineering

Vol. 1 No. 3 Contents September 1977

A Quarterly Bulletin published by the IEEE Computer Society Technical Committee on Data Base Engineering

Data Base Engineering Bulletin

A Quarterly Publication of the IEEE Computer Society
Technical Committee on Data Base Engineering

Data Base Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Data Base Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, data base architecture, data base machines, intelligent front ends, mass storage for very large data bases, distributed data base problems and techniques, data base software design and implementation, data base utilities, etc.

Contribution to the Bulletin is hereby solicited. News items, letters, technical papers, book reviews, meeting previews and summaries, etc., should be sent to the Editor. All letters to the Editor will be considered for publication unless accompanied by a request to the contrary. Technical papers are unrefereed.

Opinions expressed in contributions are those of the individual author rather than the official position of the TC on Data Base Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in Data Base Engineering Technical Committee is open to IEEE Computer Society members, student members, and associate members. (Application form in this issue.)

FROM THE CHAIRMAN

On behalf of the members of the TC, I would like to express our thanks to our past chairman, David Hsiao of the Ohio State University. As a result of his dedication and hard work, the TC got off to a great start. Under his guidance, our TC has been very active in professional activities. It has organized sessions in the COMPCON and the forthcoming First International Computer Software & Application Conference. It has co-sponsored the Second and Third International Conferences on Very Large Data Bases (VLDB) and the Third Workshop on Computer Architecture for Non-Numeric Processing, and the Workshop on Operating and Data Management Systems.

Together with Jane Liu of the University of Illinois, they created the TC's Newsletter. Our Newsletter has been recently cited by Harlow Freitag of IBM, Chairman of TC activities, to be a model Newsletter for all TC's to follow. David and Jane are to be congratulated for having done such an excellent job. Jane particularly deserves our thanks for she has spent many hours to make this possible. I am very happy to say that Jane has agreed to remain as our Newsletter editor.

The course David has paved for our TC is an excellent one to follow. Our TC in the coming year will once again be very active in many professional activities. We will again organize sessions in the 78 COMPCON and COMPSAC conferences. In cooperation with other professional societies, we will co-sponsor the Fourth VLDB Conference to be held in West Berlin, the Fourth Workshop on Computer Architecture for Non-Numeric Processing, and a Workshop on Data Base Design being organized. We have also planned to bring out a special issue on DB Machines in the IEEE Transaction on Computers.

In addition, as our TC is maturing, David and I feel that the goal of our TC, its offices, and the rules of electing officers should be better defined. As a result, I have requested Bing Yao of Purdue University to organize and head the Roles Committee to deal with these issues. Moreover, because of the broad interest of our TC and the active role we play in the various professional activities, we need more members to be effective. Lorraine Duvall of Illinois Institute of Technology has agreed to organize and head the membership committee which will determine and implement plans for a membership drive.

As you see, our activities in the coming year are numerous. Our TC members will have ample opportunity to become professionally active. The extent that you would like to participate is your own choice. I urge you to let me know of your desire. My address is IBM, K55-282, 5600 Cottle Road, San Jose, CA 95193 and my phone is (408) 256-7654. With your help I am looking forward to another successful year.

Vincent Lum

A POST-MORTEM ON CASSM[*]

G. Jack Lipovski

University of Texas
Austin, TX 78712

From 1974 to 1976, Stanley Su and I labored at the University of Florida under an NSF grant to design, build and study an intelligent secondary memory called a context addressed segment sequential memory (CASSM). As the machine was being completed and the grant expired[**], I found a great opportunity, moved to the University of Texas, and left the project. That perspective gives me an opportunity to perform a slightly less passionate and defensive post-mortem on this machine than would be possible had my work on the project continued. That is the point of this paper.

Consistent with the framework of a National Science Foundation grant, CASSM was an experimental machine designed to explore concepts for data base management on a disc, as opposed to a prototype for commerical production. Some early work on this machine was done by Len Healy, myself and Keith Doty, and was published at the same time as Minsky's and Parhami's work in 1972. However, the design was considerably refined when NSF funding became available. Primarily, as Stan Su maintained and I accepted, the relational data base model was felt to be the best complete model for data base management, so it was realized in the instruction set of CASSM.

Although intelligent secondary memories had been proposed and built as early as 1956 by Hollander, and a trail-blazing effort that designed an associative processor on a kind of network data base model had been studied by Savitt, et. al. at Hughes Aircraft Company around 1966, CASSM contributed the principle that intelligent secondary memories should be designed around data base models. Just as the Burroughs computers were top-down designed to execute an accepted high level programming language, ALGOL, CASSM was the first to be top-down designed to implement an accepted data base model, the relational model. A number of other intelligent secondary memories, such as RAP and RARES, have since been designed on the same principle. Secondarily, CASSM included several non-relational techniques for comparison and evaluation, as is appropriate in an experimental machine. It implemented the hierarchical, network and associative net models. Incidentally, we found that all models could be effectively implemented with little hardware expense. It had several techniques for searching character strings and ordered sets. It stored and fetched its own instructions and operands. Its hardware automatically collected garbage. It provided for input and output that operated concurrently with query processing and for character string-code word translation on input and output. Finally, CASSM demonstrated that these varied features could be implemented in a small "microprocessor" (220 SSI circuit chips) for each disc track, and that adequate communication could be provided between these "microprocessors" through a few connections (pins) so that such a "microprocessor" could be economically realized by an LSI chip.

The following paragraphs discuss the features of CASSM, some remarks of them by others, and my own assessment of them.

[*] This paper was partially supported by NSF grant 6543225.

[**] An extension grant was subsequently made by NSF to Stanley Su, one of the original principal investigators.

CASSM was designed to be implementable in LSI. A typical system might contain a few dozen fixed head discs and a few hundred identical LSI chips, one on each head of the disc. (Incidentally, contrary to some claims, CASSM was designed to be implementable also on CCD or bubble memories, or IBM 3330 type moving head discs.) Each cell processed the data in serial fashion. Now, some investigators propose having specialized large modules. (One for directory lookup, one for sorting, etc.) We believe the identical cell approach is far superior. Look at how cheap microcomputers are, compared to maxicomputers. Microcomputers are cheap because the design cost, about $1 million per chip, can be amortized over several hundred thousand copies, and the copies cost in the order of $5.00 a piece. To design large specialized modules would cost several million dollars for each module. And, fewer copies of each copy could be made. Current LSI economics may well doom the approach of designing specialized large modules when an aggregate of small identical modules can do the job almost as well. Also, some investigators have proposed a parallel-by-word cell. CASSM was designed as serial-by-bit. There is no advantage to query processing in the parallel technique. Whichever way it is organized, the cell takes one disc revolution to execute an instruction. The only real difference appears to be input-output. It can be easier to design high speed parallel-to-parallel I/O interfaces than several high speed concurrent serial-to-parallel I/O interfaces. The serial approach uses smaller cells and fewer pins per cell; it is thus more suitable in the near future to LSI fabrication. Also, the word width in a serial organization can be changed merely by changing timing signals. However, since a parallel-by-word arrangement shares the control logic among several tracks, it may become more economical in the long run. But, hardware designers have no difficulty converting serial techniques to parallel techniques; it is naive to make a big issue out of this point.

CASSM used a random access memory to record search results on records (tuples) and to transfer pointers (we prefer now to call these tokens) from one record to another. This feature seems to be the most misunderstood of all. This RAM is one bit wide and has as many bits as there are records (tuples) in the file. It is not a big RAM, as other investigators have misrepresented it. Moreover, if you carefully study successful architectures like the standard von Neumann computer architecture, you should observe that a good architecture uses some hardware for more than one purpose. The computer, for instance, uses an adder to perform ADD instructions, calculate addresses, and increment the program counter. CASSM's RAM handles a number of problems, including backward marking (marking a word that has already passed by the write head as the disc revolves), implicit join (marking elements of the intersection of two sets A and B in one of the sets), pointer transfer, (marking records pointed to by selected pointer references) and, as the group at Edinbourough, Scotland has shown, deleting duplicates. Compared to the technique that stores an entire tuple to mark the beginning, this technique is less costly. CASSM used about half as many chips as RAP. Moreover, CASSM can do an implicit join in two revolutions using this technique, while other techniques take as many revolutions as there are elements in one of the sets. In my opinion, CASSM's RAM is like the accumulator in a standard computer; it is the best technique in an intelligent disc for all of the above mentioned capabilities because the same hardware is useful for several important functions.

CASSM utilized a technique for organizing hierarchical data in preorder form on the disc. Stan and I differ on the value of this technique. It required a fair amount of hardware, and a lot of interconnections between cells, to "walk down" to subtrees like CDR-CAR in LISP. While the SQ pair technique that George Copeland came up with is quite easy to implement, the technique of walking down was too expensive, in my opinion.

Garbage collection was correctly implemented in CASSM. Some investigators have claimed that the garbage collection technique is not useful because the hardware moves words from cell to cell to collect garbage, and the user does not know at which cell his data is currently located. But in CASSM, there is no need to know. We eliminate the directory and all the need for software garbage collection. Since we've solved this problem, I view any architecture that does not use this solution or that does not find a better solution as a step backwards.

Input and output is done concurrently with query processing. That is, as query i is being processed, query i-1 is being output and its codewords are being translated to strings while query i+1 is being input and its character strings are being translated to code words. Some investigators have claimed that this technique makes CASSM slower. I believe that they do not understand pipeline architectures. The purpose of pipelining is to increase throughput (this is the key performance factor in an intelligent disc), and to decouple the logic to simplify its design.

However, midway through the design of CASSM, we began to realize a subtle principle. We thought we would design the machine around the comparator that does the searching. But comparators are so easy to implement that we were putting half a dozen specialized comparators in the cell. The output channel is the key system component. I advise any future designers to very carefully design the output channel first and build the cell around it. All current designs, including CASSM, are weak in this key aspect.

One of the related issues is pipelining within the cell itself. CASSM was heavily pipelined. Garbage collection was pipelined with search hardware, which had to be split into two modules for effective pipelining. Query instructions were pipelined so that, although most instructions took two revolutions, the second revolution of one instruction was overlapped with the first revolution of the next instruction. This makes the machine much harder to describe and to understand, and may be the source of much of the misunderstanding about CASSM. Nevertheless, in a slow disc processor, pipelining contributes so much to the performance that it has to be considered. CASSM was demonstrated at the Second Workshop on Non Numeric Processing to prove that massive pipelining is feasible in a cell.

CASSM had several features that I felt were important but nobody has attempted to evaluate. One of the key questions is whether to store character strings or code words on the disc. Code words are far more compact and enable more efficient content addressing and token transfer techniques. However, the user inputs character strings and expects to have character strings output. I claim that the dictionary to convert between code words and character strings may be larger than the relational data base itself. CASSM stored both, and hardware automatically translated to and from code words. CASSM's ability to store and fetch its own instructions may lead to implementing large numbers of artificial intelligence demons on the disc. This may make possible far richer and more flexible query languages. However, the surface of this issue has hardly been scratched.

I have defended most of the techniques in CASSM, and have criticized a few. Please forgive my bias, for as they say, "only a mother could love it." I invite the reader to take issue with my remarks in a constructive way.

In closing, I would like to acknowledge the support of my colleague, Stan Su, who contributed background on data base management that was essential to CASSM's success, and of two outstanding graduate students, George Copeland who contributed a number of initial ideas (especially S-Q pairs) and helped us write the NSF proposal, and Ahmed Eman, who thoroughly dissected the machine and contributed key ideas (especially token transfers and the high level language CASDAL) near the end of the project. It is hard to say who can claim which idea, for we worked as a team and ideas flowed freely.

REVIEW OF THE ACM-SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA.

August 3-5, 1977
University of Toronto
Toronto, Canada

by

Carlo Zaniolo
Sperry Research Center
Sudbury, Massachusetts 01776

The University of Toronto and the surrounding city supplied the congenial setting for this year's conference attended by approximately two hundred and fifty participants, many of them from overseas. The conference was characterized by the stimulating contraposition of formal papers emphasizing database theory to panel discussions emphasizing practictioners' view and limits of current technology. Much to the credit of the organizers the formal paper presentations and the panel discussions were combined into a balanced program. An active participation by the audience contributed to the success of this conference.

A total of twenty one papers were selected from the seventy submitted. Four papers were further chosen for the September issue of the ACM Transactions on Database Systems. The remaining seventeen were included in the conference proceedings. Database theory and integrity constituted the preferred subjects of the papers. Also covered were the topics of database interfaces, applications, concurrency and performance evaluation.

The panel sessions dealt with general topics of practical relevance. Some interesting lessons learned in the business world were presented to the audience during the first panel entitled "Data Base Design Experiences". Randall Rustin reminded the attendees that general purpose database systems may not supply a cost-effective solution for an enterprise that is primarily concerned only with one dominant application. The need for thorough analysis and careful logical design at an early stage was unanimously stressed by the panelists; each of them nevertheless endorsed a different design methodology. Robert Curtice suggested a top-down approach to integrate the views of the various users into a single schema. Donna Rund proposed a design methodology based on enterprise procedures and information usage, while John Lyon advocated a data oriented approach which anticipates future growth.

A second panel discussed the emerging technologies for mass-storage. The pros and cons were debated for bubble memories, charge coupled devices, electron beam tubes and video disks (a high density inexpensive medium for archive applications.)

A panel session on "Distributed Databases" deserves a mention as the most popular event among the (technical) activities of the evening. Short reports were given on the work done by the Codasyl Systems Committee and by the Computer Corporation of America in Cambridge. A main objective of the Codasyl Systems Committee is generating a common definition basis for the field thru the analysis of applications characterized by a non-contrived need for data distribution. The panel titled "Database Committee Reports" concluded the conference. This panel focussed on the controversial issue of standards for database systems. The need for standards is acute since both commercial users and government agencies are asking for it. Yet the field is undergoing a rapid evolution and important problem areas remain unsolved. Thus further research results are needed in these areas and standardizing now would be premature. After unanimously recognizing this dilemma the panelists discussed possible strategies for its resolution.

The next ACM-SIGMOD International Conference will be held in June 1978 in Austin, Texas.

——— ● ——— ● ——— ● ——— ● ——— ● ———

TC/DBE MEMBERSHIP APPLICATION/RENEWAL FORM

To become a member of the TC/DBE and be on the mailing list for the Data Base Engineering Bulletin, please return this form or a copy of it to:

IEEE TC/DBE
Department of Computer Science
University of Illinois, Urbana, IL 61801

NAME ______________________________
(please print)

INSTITUTION ______________________________

ADDRESS ______________________________

____

Areas of Interest: ______________________________

USER INTERFACE MULTILEVEL SECURITY ISSUES IN A TRANSACTION-* ORIENTED DATA BASE MANAGEMENT SYSTEM

by

Stanley R. Ames, Jr.

THE MITRE CORPORATION

Bedford, Massachusetts

SECTION I

INTRODUCTION

The Department of Defense Advanced Research Projects Agency, the Navy and CINCPAC have agreed to carry out a Military Message Experiment to evaluate computer-aided message-handling systems in an operational military environment. Such systems are merely transaction-oriented data base management systems. Our role in the experiment is to investigate the security ramifications of such message-handling systems. This investigation includes the identification of primitives that are needed to ensure security, and the identification of the impacts that security imposes on the user interface. In this report we discuss several tradeoffs between secure implementation and the richness of the user interface.

BACKGROUND

In response to the need to process multiple levels of classified data, the Air Force Electronic Systems Division sponsored several research and development efforts to build an operating system that would satisfy security requirements in a technically verifiable way. Technical verification was required in order to demonstrate that the system met the security requirements of the Department of Defense.[1] We have borrowed many of the results of the ESD work in the message-handling experiment. Specifically, we are designing our systems to follow the rules of a mathematical model based on the concepts of a reference monitor – an abstract mechanism that controls the flow of information within a computer system by mediating every attempt by a subject (active system element) to access an object (information container).†[2] The hardware-software mechanism that implements the reference monitor is called a security kernel. The security kernel uses the rules of the mathematical model as a specific policy in mediating access requests. This incorporation of policy into the kernel allows for a proof which verifies that the kernel correctly applies the policy to the information it protects.

The mathematical model[3, 4] establishes an "inductive nature" of security by showing that the preservation of security from one state to another guarantees total system security. The model defines security with two rules: the simple security condition and the *-property.[5] The simple security condition states that a subject cannot observe an object unless the security level of the subject is greater than or equal to the security level of the object.†† The*-property further restricts possible

† In a computer system, subjects are users and processes, and objects include programs, data files, and peripheral devices.

†† A security level is composed of a classification and a set of compartments. One security level is considered greater than or equal to a second security level if: 1) the classification of the first is greater than or equal to the classification of the second, and 2) the set of compartments of the first is a superset of the set of compartments of the second.

* This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3006 Contract No. F19628-76-C-0001 MITRE Project No. 807B.

access by stipulating that a subject may not modify an object if that object has a security level lower than the security level of the subject.

At first the *-property may seem overly restrictive. In the manual paper/pencil system we trust all the tools we use not to disclose information. However, we cannot necessarily trust the tools provided by the computer utility, because disclosure of information can potentially happen with great speed, and there exists little chance for catching the violator. The *-property, if rigidly enforced, prevents a service operating on behalf of a user from reducing the sensitivity of any information, placing the responsibility for the classified data on the people who are using it.

To provide security, a kernel must: 1) mediate every access by a subject to an object, 2) be protected from unauthorized modification, and 3) correctly perform its functions. A kernel satisfies requirement (1) by creating an environment within which all non-kernel software is constrained to operate and by maintaining control over this environment. The kernel can be thought of as creating an abstract or virtual machine on a per process basis, i.e. the kernel creates a virtual environment in which a process thinks it is the only user on the machine. This machine performs instructions from the base hardware (i.e., the hardware on which the kernel software runs) and functions implemented by kernel software. If a security policy is correctly built into the abstract machine, programs running on it will not be able to perform operations that violate the policy. In practice, the abstract machine created by a security kernel will include all of the unprivileged machine instructions of the base hardware, constrained by a hardware-supported memory protection mechanism.

The requirement for protection against unauthorized modification is satisfied by isolating the security kernel software in one or more protection domains. As an example, a ring mechanism[6] can be used to provide a domain protected from unauthorized modification. Finally, the requirement that the kernel correctly perform its functions is satisfied by using a formal methodology to demonstrate its correctness. A suitable methodology was introduced by Bell and Burke[7]. Basically, there are two steps: 1) a proof that the kernel behavior enforces the desired security policy, and 2) a proof that the kernel is correctly implemented with respect to the description of its behavior used in the first step. Kernel behavior can be described with a non-procedural program specification. A method for proving that a kernel specification is a valid interpretation of a methematical model of a security policy has been developed by Ames[8], and Millen[9]. Techniques developed by a group at the Stanford Research Institute can be used to prove that the implementation of a kernel (or any other computer program) is correct with respect to its specification[10, 11]. These techniques involve the decomposition of the kernel into hierarchically structured levels of abstraction.

Presently, there are no plans to design and verify a security kernel for the Military Message Experiment. Rather, we are attempting to design the message systems in such a way that we could replace the identified security primitives with a verifiably secure security kernel. We will rigorously scrutinize the designs to ensure that the user interface provided remains unchanged should the message system be built on a security kernel. In addition, the security primitives will be evaluated to ensure that their usefulness warrants inclusion in a future security kernel.

RELATED RESEARCH

The designs of secure data base management systems have been investigated by several other projects, most notably Marvin Schaefer and Thomas Hinke of System Development Corporation and a team from I. P. Sharp Associates Ltd. SDC's work involved the designs of a secure data base management system built on a Multics Security Kernel. Schaefer and Hinke concluded that while all the problems of creation and deletion have not been resolved, a relational data base could comfortably exist within the confines imposed by a multilevel security kernel[12]. I.P. Sharp Associates Ltd. is currently investigating the design of kernel primitives that are related to data base management systems. Neither study has addressed the impact that security has on the user interface. The Military Message Experiment is unique in that the requirements for multilevel security and a usable operator interface are both present.

FACTORS

A primary requirement of the Military Message Experiment is the development of a multilevel secure operator interface. Usability can be judged in three areas: the features provided, the ease in which commands are entered, and the overall performance. Certainly an interface that does not provide needed functions, or is difficult to learn and use, will not be used. In addition, it is clear that a message system that does not exhibit secure behavior cannot be used by the military.

The types of features that we desire to support include the ability to interactively read, create, file, and annotate messages. With approximately a thousand messages being received a day, selective retrieval of a message on the basis of key words, date-time-groups, originators, and subjects is required. Given the necessary features that must be supported by a military message service, a message must be considered a multilevel object. The primary fields of a message are as follows:

- **Header:** The header of a message contains the address, destination, originator, date-time-group identifier, overall message classification, etc. The header is defined to be unclassified.
- **Subject:** The subject of a message may be of any classification less than or equal to the overall message classification.
- **References:** Most references contain a date-time-group which is unclassified; however, certain references may contain additional information which, unless specifically given a classification, must be treated at the overall classification of the message.
- **Text:** The text of the message has a classification usually equal to the overall message classification. Paragraphs of the text may be labeled at a classification that is lower than the overall classification of the text.
- **Annotations:** Although not specifically part of a message, annotations may be added by a user to a message, or message field, at any classification.
- **Key Words:** For filing or retrieval purposes, key words will be added by the user to the message. Key words may be of any classification.

PARTICIPANTS

In an effort to investigate alternative solutions to the problems encountered in message system development, DARPA and the Navy are sponsoring three separate message system development teams representing: Information Sciences Institute, Massachusetts Institute of Technology, and Bolt Beranek and Newman Inc. Throughout this paper, recommended solutions to the various security problems encountered at the user interface are presented. The three development teams have not necessarily selected the solutions given.

SECTION II

SECURITY TRADEOFFS AT THE USER INTERFACE

OBJECT SIZE AND FEATURES SUPPORTED

The size of an object, the basic protection unit that the hardware and software of the system can support, will significantly influence the types of features that can easily be supported. The smaller the size of the object, the larger the number of features that can be supported.

In the Military Message Experiment, we decided that we did not wish to be constrainted by the object size that TENEX, our host system, could support. We based this decision on the fact that TENEX is being used for convenience only. We believe that we can build a kernel that can support whatever object size that is shown to be most suitable to the needs of a transaction-oriented data base management system. By not constraining ourselves to TENEX, the basic protection unit can be determined by the size of object that the terminal can support.

The terminal being used is a modified HP 2645. This terminal can support three different object sizes: the entire screen, a window on the screen, or a domain within a window. Each window consists of one or more lines on the screen and scrolls independently of other windows. Each window is, in effect, a virtual terminal. A domain is a set of one of more contiguous characters within a window; a domain cannot be scrolled independently of other domains in the window.

If we consider the screen to be the fundamental size of the protection object, we achieve the simplest form of the user interface. Each user is able to work at only one security level at a time, which implies the need for only one process per user to be active at any time. Almost no verification of the terminal or the software operating within it is required. With the entire screen at a single security level, the user is informed of his working security level by a single set of security level indicators attached to the terminal.

The primary disadvantage with this approach is that it drastically limits the extent and richness of features that can be supported at the user interface. There are two features in particular that the user desires: the ability to generate a message at a lower security level while reading information at a higher security level, and the ability to have the header information of a message automatically copied on a reply, especially when the reply is done at a lower security level than the text of the message being replied to.[†]

Choosing the window as the basic protection unit allows increased richness of the user interface. If each window can have an independent security level, generation of messages at one level while reading at a higher level is easy to implement. In addition, having a smaller protection object allows the message systems to treat several parts of a single message at separate security levels. Treating the header of a message as unclassified allows the reply command to copy the header regardless of the security levels of the old and new message. In the Military Message Experiment system, which uses windows as the basic protection unit, the user is made aware of his working security level by a set of security level lights that are activated by the location of the screen cursor.

From a security standpoint, using windows as the basic protection unit increases the complexity of the kernel. For example, the terminal would have to verifiably protect the multiple levels of

[†]The second disadvantage can be solved for formal message traffic by keeping an unclassified copy of the message header to be used for replies. This solution requires, however, that the message header of newly created messages be reclassified when the message is released.

information within it. In addition, having multiple security levels on the screen implies either that significant sections of the message systems must be verified or that a process is required for every active security level used by the terminal. Since having security levels per window greatly increases the richness of the user interface, we have identified tradeoffs between verification difficulties, performance, and the richness of the user interface.

A window may still be too large an object size. For instance, if key words can occur at each of the different security levels, then each key word requires a separate window. Since windows must occupy at least one line on the screen, and the screen only has 24 lines available, one quickly runs out of lines if the message is broken up into fine-grained objects at different security levels. In addition, since each window scrolls independently, the message system has difficulty in reassigning portions of the screen if each line is a separate window. The solution is to make the domain the basic protection unit. If each domain has a separate security level, windows can be assigned as needed to group domains into logical scrolling groups. In addition, multiple security levels can be used on a single line.

While the advantage of security levels per domain may be great for the user, the effect on the security of the system can be staggering. For instance, instead of using lights driven by the window containing the cursor, the Military Message Experiment system that uses this approach labels the security level of a domain by using unforgeable characters at the beginning of each domain. A terminal-driving program must, therefore, interpret all commands to the terminal to ensure correct operation.

After investigating the three object sizes that our terminal can support, we believe that security levels per window offer the best long-range solution to the problem. Security levels per screen do not offer enough flexibility. Security levels per domain do not appear to offer enough added advantages to warrant the additional portions of the system that needs to be trusted.

COMMAND INPUT

There are several methods for entering commands into the system. These include treating all commands as unclassified, treating all commands at the user's maximum security level, requiring the user to enter commands in windows of different security levels, and having the command automatically transfer the user to the proper security level.

Treating all commands as unclassified has the advantage that the user state[†] can be controlled at the unclassified level. Each process at a higher security level can be made aware of the commands that are entered. There exist, however, commands that have classified arguments; an example of this type is a command that locates all messages that have a specific word in their subjects. We consider this word as a potentially classified argument. We have yet to determine how to handle classified arguments to unclassified commands.

Treating commands at the user's maximum security level eases the determination of the proper classification of the parameters, since the user can examine all of his messages at his maximum level. However, several commands must transmit arguments to the lowest level (unclassified). An example of this type of command is the transmit command. Transmission of messages must occur at the level of the message header, which is unclassified. Unless sizeable portions of the system are verified, or unless user reconformation is required for all commands that have lower-level arguments, the protection policy prohibits the transmission of arguments to the proper level.

[†] The user state consists of all information that needs to be preserved over security levels. It includes the current message pointer and the tools being used.

Multiple command windows eliminate the problem of determining the proper level of the command by forcing the user to enter the command at the proper level. The use of multiple command windows has two disadvantages: less screen space available to enter and display messages; and increased user interface complexity caused by requiring users to predetermine the proper security level for each command. Because we believe that security considerations should not unnecessarily restrict the user, we cannot recommend this solution.

The most desirable solution is to separate the set of commands into: commands that must be entered at the unclassified security level, commands whose parameters can potentially be at the users maximum security level, and commands that can be entered at whatever security level the user is working at. Command invocation would then automatically establish the security level of the command window so that parameters could be properly entered. Then, unless the user desires to specify that the command operate at a different security level from the predetermined security level, he need not be concerned with selecting the security level of the command or its parameters.

To eliminate the necessity of verifying the entire command processor, we suggest the use of command function keys for those functions that predetermine the security level of the command window. These function keys would transmit directly to the verified code that sets the security level of the command window. The use of function keys eliminates the need to verify user typing and command editing operations. Although a certain amount of additional code must be validated, this solution eliminates most of the security and user interface problems of the other methods of command input.

THE CASE FOR A LIMITED WRITE-DOWN CAPABILITY

The strict enforcement of the security model eliminates any possibility of a security compromise, a write-down path through the system that releases information of a higher security level to a lower security level. However, there are several situations in a transaction-oriented data base management system where the user, by following instructions given by the system, can inadvertently compromise small amounts of information.

Consider the following example: A user asks for a list of all his messages so that he can determine all the messages with a subject having word "x" in them. To perform this operation, the user must be at the highest security level of any of the messages he wants examined, which is normally his maximum security level. The enforcement of the *-property forces the results of this examination to be at the level at which the examination was performed, again his maximum security level. Should the user then decide to perform any modification to a message returned by this examination that has a security level lower than his maximum security level, he would have to issue commands at the security level of the message that he desires to modify, and tell the system the unique identification of the message[†]. However, the act of giving the system an identifier told to the user at a higher level is itself a formal *-property violation. It is conceivable that a maliciously written program could use this *-property violation to compromise information. We know of no way, however, to eliminate this type of *-property violation, and we believe that unless the entire system is validated, there is a risk that this type of violation will always be present.

Because *-property violations exist through actions of the user, a case can be made to simplify the user interface in situations where user *-property violations exist. The simplification takes the form of allowing the system, in violation of the *-property rule of the security model, to transmit

[†]The unique identification is required here because the system is unable to transmit the desired identifier due to the enforcement of the security model.

the unique identifier of the message that the user wants to modify. Although doing this increases the possibility that a security compromise can occur within the system, we believe that we can design sufficient controls, including auditing and restrictions on the amount and type of information passed, to limit the bandwidth of this type of *-property violation so that it is no larger than the violation that otherwise occurs through actions of the user. An important aspect of this limitation is the requirement that only a limited amount of fixed formatted information be transmitted to a lower security level for each user request. Allowing the system to transmit this information can, in some situations, greatly simplify the user interface.

MULTIPLE PROCESSES AND PERFORMANCE

Each different security level that is being used on the terminal requires a separate process on the host machine. The multitude of processes for security may greatly impact performance. On a traditional system such as TENEX, where process control is handled in software, process swapping takes a considerable amount of time. The solution to the security requirements for multiple processes is utilizing hardware that can efficiently support large numbers of small processes.

ERROR RECOVERY IN AN OPEN LOOP ENVIRONMENT

Several of the implementation designs that we have investigated include the concept of an unclassified process: receiving the majority of the commands, determining the proper security level needed to perform these commands, and then activating a process with that security level so that the commands can be performed. With these designs, much of the control of the message system will be done by an unclassified process. The disadvantage with this approach is that, should an error occur between the unclassified control process and the classified operational process, the classified process cannot ask for clarification[†]. Without clarification, error recovery is difficult. This problem is referred to as the open loop problem.

Presently we believe that the best solution to the open loop problem is to allow the system to close the loop when an error of this type is encountered. Closing the loop improves recovery but impacts security, since a *-property violation exists when the loop is closed. We are working on restricting the amount of information that is required to be transmitted and will employ some form of auditing to prevent over-use.

SECTION III

SUMMARY

We have reached several conclusions that impact the requirements for a usable secure interface. Among these are the realizations that: object size determines the types of features that can easily be supported, a limited type of write-down may be needed to ensure user acceptance, and, unless process control is handled by hardware, the security requirements for multiple processes may impact performance to an unacceptable degree. Among the problems that require additional effort are: the best way to securely enter commands, and identification of tradeoffs between verifying a portion of code and improved user interfaces.

The first implementation of the message systems will be completed in early 1977. These systems will include secure behavior at the user interface. In addition, each system will provide a detailed design for implementing the system in a verifiably secure fashion. This design will be implemented by late 1977, giving us a conceptually secure system. With these operational systems, we will be in a better position to answer questions regarding the impact security has on the user interface.

[†]The *-property rule prevents a process of one security level from requesting information from a process of a lower security level.

REFERENCES

1. "ESD 1974 Computer Security Developments Summary," MCI-75-1, Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, Massachusetts, December 1974.

2. J. P. Anderson, "Computer Security Technology Planning Study," ESD-TR-73-51, Volume I and II, James P. Anderson & Co., Fort Washington, Pennsylvania, October 1972.

3. D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations and Model," M74-244, The MITRE Corporation, Bedford, Massachusetts, October 1974.

4. K. G. Walter, W. F. Ogden, W. C. Rounds F. T., Bradshaw, S. R. Ames, Jr., and D. G. Shumway, "Primitive Models for Computer Security," ESD-TR-74-117, Case Western Reserve University, Cleveland, Ohio, January 1974.

5. D. E. Bell and L. J. LaPadula, "Secure Computer Systems," ESD-TR-73-278, Volume I—III, AFSC, Hanscom Air Force Base, Bedford, Mass., November 1973 — June 1974.

6. R. M. Graham, "Protection in an Information Processing Utility," *Communications of the ACM,* Volume 11, Number 5, May 1968, pp. 365-369.

7. D. E. Bell and E. L. Burke, "A Software Validation Technique for Certification, Part 1: The Methodology," ESD-TR-75-54, Volume I, AFSC, Hanscom Air Force Base, Bedford, Mass., April 1975.

8. S. R. Ames, "File Attributes and Their Relationship to Computer Security," ESD-TR-74-191, Masters' Thesis, Case Western Reserve University, June 1974 (AD A002159).

9. J. K. Millen, "Security Kernel Validation in Practice," *Communications of the ACM,* Volume 19, Number 5, May 1976, pp. 243-250.

10. L. Robinson and K. N. Levitt, "Proof Techniques for Hierarchically Structured Programs," Computer Science Group, Stanford Research Institute, Menlo Park, California, January 1975.

11. L. Robinson, P. G. Neumann, K. N. Levitt, and A. R. Saxena, "On Attaining Reliable Software for a Secure Operating System," *1975 International Conference on Reliable Software,* Los Angeles, California, April 1975, pp. 267-284.

12. T. H. Kinke and M. Schaefer, "Secure Data Management Systems," RADC-TR-75-266, System Development Corporation, Santa Monica, California, November 1975.

BOOK REVIEW

Data Base Design
Gio Wiederhold
Stanford University
McGraw-Hill Book Co., 1977

This is a comprehensive textbook (658 pages) on file structures and databases. After some introductory material in Chapter 1, the author discusses in Chapter 2 the characteristics and performance parameters of peripheral storage devices. Those parameters are then used in Chapters 3 and 4 in a discussion of the major types of file organizations. Chapters 5 and 6 contain additional material on file-system evaluation and analysis. The second part of the book, Chapters 7-10 describe database models, schemas, implementation aspects, and information retrieval from a database. The major approaches such as hierarchical, network, and relational database systems are described in detail. Chapters 11-14 deal with reliability, security, integrity, and coding. Chapter 15 serves both as a summary and as a place for a discussion of the operation and management of databases. There are three appendices and an excellent bibliography.

This reviewer is presently using the book in a senior-level course on File Structures. By the end of the quarter we will have covered Chapters 1-3, a part of Chapter 4, and Chapters 7-9. The book contains enough material for a second course. The writing is clear; the book generally is well-organized and has a wealth of excellent material. As preparation, the student should have a solid background in data structures, and some previous experience with files and databases would be very helpful.

Although it is a fine book, it does have some weaknesses. The instructor must provide meaningful problems. The author has a tendency to make references to later chapters. Chapter 3 seems to contain too many formulas some of which are difficult to justify. Occasionally material is written in a vague manner; perhaps Chapter 8 could be combined with Chapter 9 for a more coherent presentation. Several important ideas, such as the notion of set as used by CODASYL, do not seem to be clearly defined.

I plan to use this book again and would recommend its use for other instructors in similar courses. The author did a fine job, indeed, in writing such a comprehensive, up-to-date, readable, and usable textbook.

John Grant
University of Florida

CALL FOR ABSTRACTS IN DATA BASE ENGINEERING

★ ★

DBE, as a service to its readership, is now publishing short statements from members of the research and development community about what they and/or their groups are doing in Data Base Engineering and what they consider important topics for future research and development.

The purpose of this activity is to promote a general awareness within the community served by DBE of what is being done and/or what is planned, where it is being done, who is involved, and how to get additional information.

Individual project abstracts will be published on a continuing basis. Priority in publishing the abstracts will generally be determined by the date of receipt of the material, at the discretion of the Editor, and as space available within each issue of DBE allows.

Each individual Abstract should include at least the following information presented if possible within these named categories:

1.	<u>NAME OF PROJECT</u>:	Use an existing name. If it is not self-descriptive append more explanatory terms.
2.	<u>ORGANIZATION(S)</u>:	State the organization and/or sponsor of the project. Include mailing address.
3.	<u>PERSONNEL</u>:	Give the name of the Principal Investigator and/or other technical personnel, and include mailing addresses if different that above. Telephone numbers are optional.
4.	<u>KEYWORD(S)</u>:	Give short phrases or keywords that indicate subareas within Data Base Engineering.
5.	<u>DESCRIPTION</u>:	A paragraph or two describing the project, its objectives, novelity longevity, technical approach, outcomes, current status, etc.
6.	<u>IMPLICATIONS</u>:	A paragraph or two indicating what's expected, what the future may bring, what problems there might be,etc.
7.	<u>REFERENCES</u>:	A few key citations as appropriate to (5) and (6).

Each statement should be fitted into a 6.5" X 10.0" page frame minimizing white space from the bottom of the form upward. The photo-ready copy should be sent flat to the Editor of DBE: Dr. Jane Liu, Department of Computer Science, University of Illinois, Urbana, Illinois 61801.

Please circulate this Call for Abstracts in Data Base Engineering among your colleagues and others who should contribute.

PLEASE DUPLICATE AND POST

★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★

MEETINGS OF INTEREST

◆ The Third International Conference on Very Large Data Bases
October 6-8, 1977, Tokyo, Japan

Conference Chairman: Stuart Madnick, MIT,
Cambridge, MA 02139

◆ COMPSAC 77: First International Computer Software and Applications Conference
November 8-11, 1977, Chicago, Illinois
Sponsor: IEEE-CS

General Chairman: Stephen S. Yau, Dept. of Computer Science,
Northwestern University, Evanston, IL 60201
(312) 492-3741

◆ Sixth Texas Conference on Computing Systems
November 14-15, 1977, Austin, Texas
Sponsors: ACM and IEEE-CS
University of Texas
Department of CS and EE

Conference Chairman: Michael A. Duggan, Dept. of Computer Science
University of Texas, Austin, TX 78712
(512) 471-3326

◆ Sixth Symposium on Operating Systems Principles
November 16-18, 1977, West Lafayette, Indiana
Sponsor: ACM SIGOPS

General Chairman: Saul Rosen, Dept. of Computer Science
Purdue University, West Lafayette, IN 47907
(317) 494-8235

◆ ICPCI 78 International Conference on the Performance of Computer Installations
June 22-23, 1978, Gardone Riviera, Lake Garda, Italy

Call for Papers on: measurement and evaluation of data base systems, distributed system performance, virtual storage system improvement, performance monitoring technology, etc ...

Send working title and 300-600 word abstract by December 1, 1977 and full paper (not exceeding 7000 words) by January 10, 1978 to:

Prof. Domenico Ferrari
Computer Science Division
EECS Department
University of California
Berkeley, CA 94720