NCE & TECHNOLOGY:

PERFORMANCE ASSURANCE AND DATA INTEGRITY PRACTICES

NBS Special Publication 500-24
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Institute for Computer Sciences and Technology, the Office for Information Programs, and the Office of Experimental Technology Incentives Program.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of the Office of Measurement Services, and the following center and divisions:

Applied Mathematics — Electricity — Mechanics — Heat — Optical Physics — Center for Radiation Research — Laboratory Astrophysics[2] — Cryogenics[2] — Electromagnetics[2] — Time and Frequency[2].

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials, the Office of Air and Water Measurement, and the following divisions:

Analytical Chemistry — Polymers — Metallurgy — Inorganic Materials — Reactor Radiation — Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services developing and promoting the use of available technology; cooperates with public and private organizations in developing technological standards, codes, and test methods; and provides technical advice services, and information to Government agencies and the public. The Institute consists of the following divisions and centers:

Standards Application and Analysis — Electronic Technology — Center for Consumer Product Technology: Product Systems Analysis; Product Engineering — Center for Building Technology: Structures, Materials, and Safety; Building Environment; Technical Evaluation and Application — Center for Fire Research: Fire Science; Fire Safety Engineering.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus wthin the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Institute consist of the following divisions:

Computer Services — Systems and Software — Computer Systems Engineering — Information Technology.

THE OFFICE OF EXPERIMENTAL TECHNOLOGY INCENTIVES PROGRAM seeks to affect public policy and process to facilitate technological change in the private sector by examining and experimenting with Government policies and practices in order to identify and remove Government-related barriers and to correct inherent market imperfections that impede the innovation process.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world. The Office consists of the following organizational units:

Office of Standard Reference Data — Office of Information Activities — Office of Technical Publications — Library — Office of International Standards — Office of International Relations.

---

[1] Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.

[2] Located at Boulder, Colorado 80302.

COMPUTER SCIENCE & TECHNOLOGY:

Performance Assurance and Data Integrity Practices

Robert L. Patrick

Computer Specialist
9935 Donna Avenue
Northridge, California 91324

Robert P. Blanc, Editor

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

U.S. DEPARTMENT OF COMMERCE, Juanita M. Kreps, Secretary

Dr. Sidney Harman, Under Secretary

Jordan J. Baruch, Assistant Secretary for Science and Technology

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Acting Director

Issued January 1978

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-24
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-24, 53 pages (Jan. 1978)
CODEN: XNBSAV

Library of Congress Cataloging in Publication Data
Patrick, Robert L. 1929-
Performance assurance and data integrity practices.
(Computer science & technology) (NBS special publication; 500-24)
Supt. of Docs. no.: C13.10:500-24
1. Electronic digital computers--Reliability. 2. Electronic digital computers--Evaluation. I. Blanc, Robert P. II. Title. III. Series. IV. Series: United States. National Bureau of Standards. Special publication ; 500-24.
QC100.U57 no. 500-24 [QA76.5] 602'.1s [621.3819'58] 77-608309

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1978

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Price $2.20 Stock No. 003-003-01879-1

FOREWORD

The use of computers to automate the information handling and recordkeeping activities of Government and private organizations has brought the benefits of speed, efficiency, new capabilities, and greater flexibility to these operations. Because of these benefits, computers have become essential to recordkeeping activities in diverse areas such as banking, inventory control, payroll, and funds disbursement.

Government and private sector organizations have become dependent upon the benefits of computer utilization to such an extent that often it would be impossible to revert to manual methods and function without computers. This dependency on computers and, in particular, the need for them to perform their intended functions correctly, has made organizations vulnerable in a very real sense to the malfunction of computers. This vulnerability and its potential impact on a particular segment of Government are described in a General Accounting Office Report of April 1976 entitled, Improvements Needed in Managing Automated Decisionmaking by Computers Throughout the Federal Government.

The time has come, therefore, for the technology, as a primary goal, to focus on assuring the correct performance of computer systems. Techniques and mechanisms should be adopted to prevent or reduce the impact of system malfunctions.

The National Bureau of Standards has sponsored this study to identify a portion of those techniques and mechanisms which already exist and are used commonly in some sectors. This report, Performance Assurance and Data Integrity Practices, is the result of a research effort by Mr. Robert L. Patrick, a computer specialist. Many large Government and commercial data processing installations can quite likely benefit by selecting several of the techniques described by Mr. Patrick, adapting them to their needs and environments, and including them in their application systems.

Robert P. Blanc
Staff Assistant for Computer
Utilization Programs
Institute for Computer Sciences
and Technology

CONTENTS

PERFORMANCE ASSURANCE AND DATA INTEGRITY PRACTICES

Robert L. Patrick
Computer Specialist
Northridge, California

ABSTRACT

This report identifies the approaches and techniques now practiced for detecting, and when possible, correcting malperformance as it occurs in computer information systems.

This report is addressed to two audiences: to the systems designer using stock commercial hardware and software who is creating a system which will tax the available hardware, software, or staff skills; and to the manager who wishes to chronicle the deficiencies in an existing system prior to improvement. It enumerates 67 items of current practice which prevent computer malperformance.

Key Words: Best practice; data integrity; data processing; error correcting; high integrity; performance assurance.

PERFORMANCE ASSURANCE AND DATA INTEGRITY PRACTICES

I - INTRODUCTION

With increasing frequency, computer systems are becoming the control element in systems that initiate actions with little or no human intervention. Such automatically initiated actions are generally taken on the basis of data or information supplied to the computer systems and acted upon by the computer programming. Such systems may produce information products which feed into a given organizational framework, without human intervention, to test the correctness of the information.

Such computer information systems rarely have formalized systems processes in place to continually check for the correct performance of the system operation. As a result, system functional failures as caused by programming and data errors, for example, are discovered post-facto or in later post-mortems when the "damage" has been done and in-process correction is impossible.

This report identifies the approaches and techniques now practiced for detecting, and when possible, correcting malperformance as it occurs in computer information systems.

This report is addressed to the system designer using stock commercial hardware and software who is creating a system which will tax the available hardware, software, or staff skills. It enumerates 67 items of current practice which prevent computer malperformance. A separate study would be required to describe the proper analysis and development methods appropriate for large undertakings. A third study would be required to address the environmental and operational problems such as change control, regression testing, maintenance of large production program sets, and to balance the effort devoted to program design, program maintenance, data base administration, production control, and machine operations over the entire life cycle of a major application system.

In the conduct of this study, 65 persons known to be interested in the area were contacted for their inputs. Correspondence and followup phone calls elicited responses from 51 of them. Two automated literature searches gave access to both the NASA and Lockheed data bases. These actions yielded a batch of books, proceedings, reports, program documentation, and custom responses which were distilled to obtain the 67 proven techniques presented in Section IV of this report.

In reviewing the literature, two common themes reoccurred --

- Error detection is costly.
- Perfection, at least in data processing systems, cannot be attained for a reasonable price.

Therefore, the system designer should:

- Pursue the critical errors and be prepared to live with the rest.
- Design to limit the consequence of error.

These themes and actions can be seen underlying the ideas presented in Section IV. A list of especially pertinent references appears in Appendix A and an abbreviated glossary of the terminology used is in Appendix B.

II - STATE OF THE ART

(In Dynamic Error Checking and Correcting)

Dynamic performance assurance is the result of a hierarchical set of provisions built into a computer system in an attempt to guarantee that the entire system (hardware, software, applications programs, and manual procedures) performs properly or that prompt corrective action is initiated. The following framework identifies four stages of assurance:

The foundation of all such assurance must be the architecture of the computer hardware, specifically the checking that is built into each hardware unit and how those checking features are integrated into a checking subsystem.

The second stage is system software. What does it do with hardware error signals, and what additional actions does it take to supplement hardware error checks with software checks of its own? Further, are actions taken in a timely manner so errors do not propagate outside the system or cause secondary mistakes that are difficult to reverse?

The third stage relies on checks built into applications programs to weed out any context dependent errors, prevent propagation, and preserve the integrity of the entire system (including both programs and data).

The fourth and final stage is the man-machine interface. Since some errors cannot be automatically recognized (much less

automatically corrected), what information is made available to the human operators of the system, is it in a useful form, is it timely, and are manual procedures in place to respond quickly so processing can continue?

Since most information processing is conducted on stock commercial computers, without modifying the manufacturer's hardware and software, this investigation concentrated on stages 3 and 4 to enumerate what can be done with stock hardware and software. The passages that immediately follow discuss the state of checking in the hardware and software merely to define the environment in which most programmers/analysts/systems designers exist.

Current commercial computer architecture is blessed with several families of extremely reliable circuit technology. These circuits are assembled into computer systems with varying intrinsic reliabilities. Some years ago memory banks were the most likely subsystems to fail. Hence common practice, in large commercial machines, dictated that parity checking be included as a standard feature within each memory module. Similarly, since magnetic tapes and disks (and their media) were also error prone, manufacturers have generally included parity checking on tape and cyclic checking on disk. However, there the progress stops.

At one time high quality readers read cards at two stations to check for read errors, and even further back electromechanical printers used extra circuits to check that the character impressed on the paper was the character sent to the device. As printers and card readers got more reliable, these checks were discarded.

In the early 1960s at least one computer was designed to perform the following checks: checked parity established by redundant card reading stations; verify parity when the data reached the CPU; check parity performed in parallel with all arithmetic operations; check all transfer operations (register to memory, memory to register, etc.); check parity stored with data in memory; and check all of the CPU registers out through the channels to the tapes and disks and back again. (Checking also went out to the printer, but stopped short of the print hammers.)

Most computers recently built have been only partially checked. One senior designer informed me that a CPU could be fully checked by increasing the CPU circuit count only 10% to 15%. However, in the pursuit of lower prices, the manufacturers have convinced themselves that partial checking, i.e., memory but not CPU, is adequate and hence the machine with an unbroken checking trail is a thing of the past.

With the advent of on-line systems and minicomputers, the trail of hardware checks becomes far from continuous. A bare bones terminal without any checking is the cheapest to build and sell. To get the

price of minicomputer systems down to rock bottom, some users mix and match minicomputer components from several manufacturers. While the data and electronic control interfaces are, of necessity, compatible, steps must be taken to ensure the compatibility of the checking philosophy, the machine diagnostics, and the processing of errors.

The state of the art in computer software is worse. Many computer operating systems, compilers, and utility programs simply do not work as originally delivered. To be sure, the manufacturers successfully test the main line features of each software package and correct all of the errors those tests disclose. But the state of the testing art is still relatively primitive and the quality of the resulting product depends primarily on the quality of the tests devised by the manufacturer's product assurance personnel, the amount of money devoted to testing, and the time available for testing as the development program comes to a close.

Most manufacturers do an excellent job on small software packages of low to medium complexity. No manufacturer does an outstanding job on highly complex software systems which must support a variety of hardware configurations and carry a broad spectrum of work. Software testing is a cottage industry and the resulting unreliability of new installations is a direct result.

The manufacturer's software designers are pressured both by their customers and their competition. As a result they continuously seek to offer more function with less memory and faster execution speeds. To achieve needed code compression, it is not uncommon to strip out most of the software checking mechanisms that could provide dynamic checking and increased reliability*.

---

*Typically when a software module is under test at the manufacturer's location, temporary (driver) code is placed within and around the module to screen the inputs and outputs for errors, to provide path counts for performance estimates, and to provide dynamic options which will assist in diagnosis once an error is recognized. After the module is tested, this extra code is stripped out and discarded before the module is integrated with the evolving system. Consequently, the software finally installed in customer locations is configured presuming perfect software and perfect hardware operation. As a result, errors suspected at a user installation are hard to isolate, intermittent errors are almost impossible to diagnose, and problem determination becomes a chore to manufacturer and user alike.

A better scheme might be to retain some of the original test code in a form so it can be reinserted into a running system when required. Then a user could optionally add checking in small increments (and pay as much of a storage and performance penalty as was appropriate for his environment).

Against this bleak background, we attempt to run application systems in production. In that context, it is not surprising that we have undetected errors or that we have had to devise a series of application techniques to "plaster over" the lack of checking provided by the marketplace.

Since the machines do not fail very often and since the software works pretty well in most cases, designers of small application programs can proceed confidently since they seldom stretch either the hardware or the software to the limit. However, large information processing application systems tend to be long running, and to handle big files. They are frequently connected into large (unreliable) communications networks. Thus, the designers of large data base applications have become fairly adroit in getting satisfactory operation out of computer-based systems which are driven right to the limit of their reliability. (It should be noted that designers of little systems can get into the same troubles and need some of the same tools if their hardware and software is sufficiently unreliable or if their little application uses some uncommon I/O device or probes some unexplored corner of their operating systems.)

A trap awaits the application programmer who was raised on small, easily understood applications if he thinks that a 100 man-year application is merely ten times more effort than the familiar 10 man-year application. For example, consider building a large set of files using an application program set "thoroughly" tested on a sample file occupying a half a reel of tape.

If the operating system offers a feature which automatically sequences through a series of labeled tapes containing consecutive serial numbers, one might conclude all of the tools were in place to build and process a file which runs ten full tape reels. While the equipment reliability may not be taxed by the partial reel file, it is quite likely exceeded by the multi-reel file.

To reliably process the multi-reel file, the tape access method needs to provide for reel-to-reel duplicates of each volume, to pass those reel-to-reel duplicates in synchronism, to break the big sorts up, and to install restart in every process -- the basic approach must change to successfully work beyond the reliability limits of the installed system.

Today, due to the growth of the field and the lack of documentation on big systems, there are big users who have survived their learning years and can now routinely design for big files and long running programs; but there are many more inexperienced designers who have had small successes and have yet to have their first catastrophic failure. It is, therefore, useful to describe those techniques which have been successfully applied and make them available for more general application.

Even the big shops have trouble operating their systems. The techniques for staging, balancing, and controlling data are not adequate for the monster files we would like to process today. The operating systems required to control duplexed computers with 60 or more disk spindles and 100 or more remote terminals are so complex that common commercial use is far from routine. When an operating system exceeds a million instructions and the inventory of interconnected hardware units numbers in the hundreds, the system is too complex, using current techniques, for any single human to understand or intelligently troubleshoot.

Nevertheless, we do undertake and successfully accomplish big jobs. Are the errors excessive? Yes. Is the cost excessive? Probably, but compared to what? The best way to face complexity is to proceed slowly and deliberately. Some of the more successful managers religiously change only one thing at a time.

III - A CASE STUDY

An informal field survey was taken by contacting 65 individuals. Responses were received from the 51 persons listed in Appendix C. The responses ranged from a simple "no, we're not doing anything special" to a series of published pamphlets containing the descriptions of a mature concept which demonstrated dynamic editing and checking throughout an entire set of application systems.

From those who are successfully pursuing in-depth editing and checking (in some cases correction), it appears that no single grand solution to the problem of error control exists. Instead, each successful organization has carefully analyzed its environment and the error propensities peculiar to that environment. In parallel they have assembled a series of approaches and techniques for dealing with the errors peculiar to their installations. These techniques have been creatively woven into their systems designs so the resulting application becomes relatively insensitive to the errors it routinely encounters. Frequently, these approaches involve changes to environments and organizational responsibilities to support the building of error resistant systems. A real example will demonstrate such a systems approach.

> A credit clearinghouse in operation today receives credit card clearings from a nationwide network of 100 corresponding banks, sorts all of the detailed transactions received, and distributes the appropriate detailed information back through the same network to the 100 corresponding banks. The system is designed so that drafts from the retail credit transactions are converted to electronic form at the bank nearest the retail location providing the service. After the information is converted to machine readable form, the paper is retained at the originating bank and not transmitted, sorted, or combined with the customer's monthly statement.

Thus, once the information has been converted to machine readable form, the information from the draft can be transmitted electronically, sorted electronically, transmitted to the payer's bank, sorted in detail, and printed on the payer's monthly statement.

The system operates routinely every evening. One hundred banks transmit, a massive sort occurs at the clearinghouse, and 100 banks receive before the beginning of the following business day. There are computers from seven different manufacturers in the 100 bank bookkeeping centers, and on one recent peak day 1.6 million transactions were processed through the clearinghouse.

To process this large volume of work on a routine basis, system designers implemented several straightforward techniques which, when taken collectively, provide an error resistant system. The principal techniques used are:

A - Every account has a check digit appended to the account identifying number. This appears on the credit card, on the draft, is electronically scanned and verified, and from then on is checked at every transfer.

B - About 50 individual transactions are blocked into physical records for efficiency. There are hash totals on these records which are checked before and after every transmission.

C - Batches of records are collected on a tape prior to transmission. There are transaction counts and dollar totals on the tape batches. These are checked before and after transmission.

D - The central clearinghouse has defined the interface tape format in the minutest detail. Furthermore, as each corresponding bank comes onto the network, programmers from the central clearinghouse work with programmers at the new bank to adapt a standard edit program to their equipment and configuration. This edit program must be run each night prior to the transmission of the tape batch. It checks that the tape conforms to the defined standard interface, verifies the check digit on each account, the record hash totals, the item count for the batch (tape), and the dollar sum for the batch. The same program is used to check batches before transmission and to check cleared batches when they are received from the central site. As a result, only perfect batches

are released for transmission and all returned batches are verified.

E - As each batch is received at the central site, the standard checking program is run on that batch to check the hash totals, item counts, and dollar totals. Tapes which contain errors are rejected in their entirety and must be made perfect at the originating location and retransmitted.

F - Once the batches are verified correct at the central site, their item counts and dollar totals are accumulated into grand totals for the night's work. At each step of the clearing and distribution process, the central site zero balances to this grand total item count and dollar sum, thus guaranteeing all transactions received are processed and all money claimed on input becomes in turn a claim on a cleared output.

G - The same tape format is used for the cleared outputs and the process is mirror imaged as the clearings are distributed to the corresponding banks.

H - Since magnetic tape reels are being physically handled at the corresponding banks, an operator error can cause the same batch to be retransmitted and hence double cleared. To guard against this contingency, the tape labels and batch total records are stripped from each batch and compared with similar entries previously accumulated in a 45 day receiving log. Thus, duplicate tapes will be rejected. The log also allows missing tapes, late tapes and rejected batches to be operationally accounted for.

The system described above has been processing credit clearings routinely for several years. In isolation the amount of checking done may appear to some to be excessive. However, consider the enormous amount of work related to entering over a million transactions each night, sorting them, balancing them, and getting them correct. Also consider the total amount of electronics involved in 100 data processing installations, 100 sets of transmission/receiving equipment, the communications network, the set of communications receiving/transmitting equipment at the central site, and a several hour run on the central computer to balance, sort, rebalance, and batch the clearings. Then, the extra time spent to assure that batches are perfect before and after receipt seems to be well spent.

Note particularly how the batch verification program provided by the central site, but executed on the computers at the remote sites,

crisply defines the responsibility for data integrity and fully verifies that any changes made in the computer systems at the corresponding bank do not affect the standard interface.

IV - IDEAS FOR IMPROVEMENT

Overview

From the literature search and other contacts performed as part of this field study, the following 67 ideas have been distilled. Some of the ideas were explicitly contained in the materials sent to me, others were implied, and some are my own. Some are even published in the references (see Appendix A). No claim is made that they are unique or new.

However, this set of ideas does enjoy the following set of common properties: They are real; they have seen use in production systems; and they work on stock commercial equipment, using stock commercial software; and they were employed by creative systems designers devising systems to be implemented by average programmers and run by average operational personnel.

Some of the ideas that follow have been around for some time and hence can be termed common practice. Others, while they may have been around for some years, have not been broadcast and still exist more or less uniquely in their original environments. No installation can benefit from all of the ideas that follow, but large government and commercial data processing installations will quite likely benefit by selecting several ideas for inclusion in application systems adapted to their needs and their environments.

The ideas have been grouped into the following four categories:

A. Data processing systems analysis.

B. Scientific systems analysis.

C. Implementation.

D. Environment.

Each category begins with a description of the category itself and a rationalization for its inclusion in this report.

For completeness, one section contains ideas dealing with mathematical/scientific/engineering applications which were once common practices. These practices seem to be forgotten history that may be repeated by casual programmers using time-sharing systems, minicomputers, or private hand-held calculators.

A. DATA PROCESSING SYSTEMS ANALYSIS

The main thrust of this survey involved large, high volume commercial data processing installations and their problems with input errors, programming errors, operator errors, and equipment errors in the presence of large files. Some commercial installations, similar to large government installations, are devoted to single application systems. As these shops specialize, they develop tools and techniques to solve their problems. These tools and techniques are equally applicable to the similar problems in large government installations and are enumerated below.

Design Principles

1. Concentrate Change Activity: When updating a data record during a file processing application, locally structure the application code so it does not make any changes to the record until all tests have been passed and the transaction can be posted. Thus in the event of an error prohibiting update, the original record need not be reconstructed. Further, this technique minimizes the probability that an abort will leave the data base in an inconsistent status.

2. Usage Improves Quality: While item counts and hash totals are good, the primary controls over a data base should be selected from fields that are totaled and presented on the system's major reports, i.e., optimally, the totals should be meaningful numbers used by the system's human monitors for workload, trend, financial, or value analysis.

3. (Local) Standard Data Elements and Codes: Once the data base concept is understood and integrated files are contemplated, data processing management usually elects to establish the data base administrator's job. In establishing this job, some responsibilities are transferred from the application's programming and operations departments to the new data base administrator.

Shortly after the data base administrator is appointed, he routinely establishes procedures for the data file design and definition process, followed by standard file formats, record formats, data element formats, and local standards for encoding some data element values.

These local standard encodings eventually give rise to common tables of values shared among application systems. A single common utility program under the control of the data base administrator maintains these tables of standard values.

The library community has a century of experience establishing and maintaining standard value tables; they are called "authority lists" by librarians.

4. Context Edits: Edits on individual fields in isolation are less powerful than edits on fields in combination since the latter are performed using recent history from fields and records already processed in the current computer run. For complex edits dealing with a sequence of inputs, as in an on-line dialog, build a structured edit tree and make specific tests dependent on what has gone before. Thus, strings of fields containing a structure can be verified for reasonableness by using an edit structure.

For example, the response to the question "How many persons in your immediate family?" could be edited in isolation only as numeric and between 1 to 20. However, if a followup question reads "Name the persons in your immediate family," the two responses could be processed in combination. A higher confidence level would be achieved if the number of names matched the population count. A programmer might be uncertain whether a response of one included the person being interviewed. The programmer also would be suspicious of any large discrepancies between the number of names and the population count.

If default options are used in conjunction with a complex context edit, the defaults inserted can be conditioned by the point in the edit tree where the default is invoked. Thus, a blank field could receive different default values based on different contexts.

5. Staged Edits: A large system was designed which involved a high volume of complex high value input. The system design involved minicomputers controlling clusters of keyboards and batches of transactions sent on-line to the central data processing system. The edit concept called for four discrete stages of edit:

(1) The first edit was a field edit. This was performed by the minicomputer on-line and instantaneous to the keyboards. Following each end-of-field delimiter, the field was checked against the field spec for length, character set, and value range.

(2) Following the last field in a record, the record was checked by the minicomputer against the record specification to determine if the fields existed in legal combinations for the record type and, where possible,

whether the collection of fields in the record was a legal set of fields containing individually legal field values that constituted a legal value set.

(3) After a group of records was completed by an individual operator, the batch was checked for batch consistency against a batch spec using the background partition of the minicomputer (operating under the control of the keyboarding supervisor) before the batch was released for central processing.

(4) Finally, upon receipt, the central computer edited the batch in the presence of the data base for format, content, context, and verified all encoded variables against the appropriate authority list. Incorrect or questionable records were flagged to inhibit further processing, passed through the system, and eventually found their way to printed worksheets for human resolution. All of the errors found in the batch were kept together as an error batch and printed on error worksheets in the same sequence as they were received (to assist in matching errors and original documents during the resolution process).

6. Restart Log: It is a common practice to log input transactions during a data base update process. Further, with the advent of inexpensive file storage, the logs have been specified to contain the "before image" of the data fields destined for change, and the "after images" of those same fields. Thus, it is possible to rapidly recover the modified files by loading the data set produced during the last checkpoint dump and avoiding the cost and time penalties of reprocessing by simply replacing all "before images" with their companion set of "after images" from the log.

In addition, if a sequence of transactions ends up in failure and leaves the data base in an unusable state, it is possible to reverse the effect of processing and replace all "after images" with "before images" back to the point of the first transaction in the erroneous sequence.

If input transactions are sequentially numbered and logged on the same log with the before and after images, and if those sequence numbers appear in the before and after records logged as a by-product of the update process, it is then possible to tie a transaction to its related update(s), and in so doing it is possible to recover from a failure with transactions in

process. (Such recovery requires a selective backout of changes to all records affected by the transaction in process at the instant of failure.)

7. Default Options: Default options are a reasonable way to fill in missing data elements or to correct erroneous element values where the processing context allows the correct value to be known to the computer program.

Where life or severe catastrophe are at risk, default values should not be used. If the critical process is on-line, safety considerations should require that the terminal keyboard be locked and that the computer insist on perfect data with all exceptions positively acknowledged, i.e., no assumed values and no default data values.

8. Keyboard Factors: The design of keyboard entry applications requires a careful appreciation of the human factors involved. These in turn require the system designer to be familiar with the data and its various formats; the equipment to be used; and the keyboarding environment including work stations, lighting, and sound conditioning.

Professional full-time keyboarding personnel, properly supervised and in a good working environment, may use heavy abbreviating and short synonyms to cut the key stroking required. However, if the input volume from the specific application being designed is not sufficient to provide full-time employment for several operators, the designers should be wary of excessive abbreviations and any form of code compression at the input stations. Instead, the designer should provide alternate input sequences and varying codings for different levels of training, experience, and skill. These must be combined with a set of supervisory statistics which allow each operator's throughput to be measured against an appropriate skill classification. (Beware of simple statistical averages as these will push unskilled operators beyond their capability and error cycles will result.)

File Structure

9. Quality Flags: Data quality flags can be added to files, records, groups of fields, or individual fields in a data base and can be used to condition processing to avoid a mismatch between the quality of the data and its intended use.

Every record can carry a quality flag. This flag would indicate to each processing program which format and content checks the record has successfully passed. The quality flag should be the first field read by an application program, and the last field written by an update program.

10. Activity Dates: If each input transaction carries a date-of-event, and each record in the file carries a date-of-last-transaction-processed, these two dates will allow processing to be performed in the proper sequence (or manual intervention to be requested) in case of transportation or processing delays.

11. Reliability Through Batch Redundancy: Hash totals and item counts assure that batches retain their integrity through multiple transformations (i.e., editing, sorting, updating, and printing). Item counts should be carried completely through the process to assure that every item was processed. This has long been a common practice in some industries.

Ideally the primary key used in file maintenance would be a unique numeric identifier carrying a check digit. Where this is not possible, partial protection can be achieved by check-summing (hash totaling) numeric key fields so a batch of data maintains its initial integrity throughout all computer processes. It should be noted that check digits on I.D. fields, and checksums on both records and batches constitute specific practical examples of added redundancy in the input stream. They increase the character count to achieve improved data reliability.

12. Reliability Through File Redundancy: Some file systems carry checksums over into the data base to assure that data quality does not deteriorate over time. The more advanced systems carry separate checksums, one for the control information and the other for the data values.

13. Reliability Through Input Redundancy: Additional redundancy can be built into input messages (at the cost of increasing the input character count) to provide context information for edit programs, to allow meaningful correction (in some instances), and to allow meaningful operator dialogs in an on-line system. This redundant information takes several forms: check words, double entries, check digits, and batch totals.

14. Redundancy for Audit: Within the data base, additional information can be carried in the record header to provide the basic parameters for certain types of integrity audits. Specifically, a record header could contain the identifier associated with the last program to update the record, the date of that last update, the time of that update, and the transaction sequence number causing the update.

15. Date Checks on Sequence: When processing input transactions, the effective date of the transaction can be processed by the computer to alert the control group in the event over-age transactions are being processed and in extreme cases to inhibit processing of ancient transactions. When the effective date of the transaction is compared against the effective date of the last transaction to update a data record, out of time sequence transactions can be recognized and manual intervention requested.

16. Self-Defining Files for Integrity: One technique for handling very large files (in excess of a dozen full tape reels at 1600 bpi) involves a file design with a directory on the front of every individual record. The directory describes the structure of the record and contains pointers to every data element contained in the record. Not only does this record format give files independence from their application programs (and hence insensitivity to change), it provides extensive redundancy which allows the integrity of the large files to be audited and maintained.

Key features of the above techniques are: (a) a master dictionary defining all data elements in the system and awarding each data element a unique I.D. number. (b) A record format which provides for a directory on the front of every data record. (The directory would be variable length and contain one four byte entry for each data element present in the record. The first two bytes would be the data element I.D. number, the third byte would be a pointer to the beginning of the related data value in the record, and the fourth byte would be the length of the value field in the record.) (c) As each program was loaded, it would reference the dictionary and provide the names of the data elements it planned to process; in return it would receive current descriptions, edit specs, and I.D. numbers for those data elements. (d) Then when each individual record is read for processing, the data element I.D. out of the directory is used to locate the specific occurrence of the data element in question and to invoke the correct edit specification.

One variant of this scheme provides a structure where records always grow through the addition of new information and no information is ever deleted; only the pointers change. Thus if correction is ever required, the backout can be accomplished by fetching the old pointers from the log and restoring the pointers to their previous values (since the data still exists in the records). (Note: This technique is most appropriate for standing master files of data with low rates of change.)

17. Variable Names Enhance Edit Logic: One set of data management software stores the variable name adjacent to each occurrence of the variable. While the storage space is increased, the name/value pair provides valuable redundancy for editing and lays the foundation for two stages of table driven edit: one as the data is received, and the second immediately prior to the update. The update edits are used to test the validity of an input variable, to compare it to other input values, with stored values, or with literals; and derive a replacement, take appropriate cancellation action, or accept the value as input.

If variable names appear in the input stream or are appended during input processing, these names can be carried with the variable throughout the system. Then, whenever a variable reference is required, the names are matched and the machine address of the adjacent data value is computed at the instant of reference. This is an example of delayed binding. It provides a degree of independence between the program and its data. Further, if the names do not match, an error routine can be called before erroneous data is introduced for processing.

While the permanent storage of named variables may increase file sizes disproportionately and hence cannot be used in the data base proper, the use of named variables on input and during preliminary processing provides a degree of flexibility in free form input coupled with important editing enhancements for the price of some additional characters in the input stream. Contrast this scheme with a highly efficient but unforgiving input format where variable recognition in the input stream is based solely upon its position.

18. Advanced DMS Offers Restore and Reprocess Functions: One set of data management software has before and after images stored on the log tape. In addition, it has a RESTORE verb to allow applications programs to request that the files be restored to a previous checkpoint. The system also has a REPROCESS verb to allow application programs to request that a limited sequence of transactions be reflected in the file up to a transaction previously found to be in error.

19. Self-Defining Files for Audit: One other advantage of a file design which has a directory on the front of every record is the ability to validate the record after update by using the transaction type code from the input and the data base description including the data element dictionary and the resulting record structure. Thus, it can be determined that all required data elements are present and all present data elements are permitted.

Whenever a group of input fields routinely appears together the group can be given a single identifier or transaction type code. This code may explicitly appear in the input record along with the group of fields that it names. This group name can be used to enter the data base description so that the data editing specification can be obtained for the group. Then, the desired structure of the input record can be verified against the specification and the following determined.

A - Does each data element meet its spec?

B - Does the record match its structure?

C - Are all the required data elements present?

D - Are all present data elements permitted?

In a similar way, the file can be verified at the moment of update or any later time. Thus, the file record would have a type code, the record specification could be obtained from the data base description, and then each record could be individually verified and the collection of data elements in the record could be checked for structure, presence, and absence.

File Controls

20. Programmed Read-Only: When you have established the correctness of basic records in master files of essentially static data, set a flag bit in the record header to inhibit further update without special privilege. If an installation standard file access method is used, this bit will inhibit unauthorized update and maintain the integrity of the basic data.

21. Redundancy for Post-Audit: After a data base has been designed, review the design and where necessary, add redundancy to allow an audit program to verify the integrity of the file set. The audit program can then be run to prevent the data in the files from deteriorating unobserved.

Take as an example the simple payroll file. The first level of detail provides a record on each employee. A second level summary record is provided for each department and the summary record contains a count of the total number of employees assigned to the department (item count), the total of the base salaries for all employees assigned to that department (control total), and an arithmetic total of all employee telephone numbers (hash total). Then whenever the condition of the file were suspect, an audit program could verify that the detailed records summarize to the department

totals. Just these few redundant control fields provide some level of confidence in the integrity of the file. Additional control fields of redundant information can be added to give any desired degree of confidence.

22. Segmented Controls for Efficiency: Most data bases have two or more processing patterns over a full processing cycle. All data base programs should maintain control totals even though they may process transactions in random sequence. Any sequential processes should do a 100% verify over the parts of the file they process. Needless to say, the file controls should be segmented so any sequential process handling part of the file can have access to control totals for just the part of the file routinely processed.

Consider the example given in the Item 21 above. In a disk oriented data base system, the department summary records would be kept in a file separate from the detailed data. Then, when the pay file was referenced during an update, the appropriate department summary record could be updated also. If the data base design allowed individual summary records to be referenced, the maintenance of summary records would be inexpensive yet they would provide the necessary redundancy so an audit program could be called at any time to prove the file. It is also desirable that the summary records be designed to provide statistical data useful for management reports.

Processing Sequence

23. Operator Feedback: For on-line systems, the computer should positively acknowledge each transaction by assigning sequential message numbers to the transactions emanating from each individual terminal. A new message is awarded the next consecutive number and a positive confirmation of receipt displays that same number back to the terminal operator. This number provides transaction accountability and is the substance of a dialog between the computer and the terminal operator in the event of restart. Using numbered transactions, the operator knows what transactions to re-enter, if any.

Ideally messages would be hash totaled by an intelligent terminal prior to transmission. If the terminal does not have this capability, then the hash total can be appended by the first intelligent machine--controller, multiplexor, or host--that handles the message.

If messages originating in the host computer and directed to individual terminals are also numbered, the operator can assure that no messages were lost in the communications network.

Similarly, if the host develops a hash total for the outbound message and if that hash total is verified by an intelligent terminal as the message is displayed, the system can assure the operator that the messages are delivered with the content unchanged.

24. Immediate Edit, Deferred Update: One large data base system collects input transactions during the day, edits them, stores them in a batch, and reflects the entire batch of transactions in the file during one massive overnight processing run. However, this system attempts to guarantee that transactions entered one day will be reflected in the file that night by editing each transaction thoroughly on-line before accepting it for the overnight queue. Thus, unless the transaction is incompatible with the data base (e.g., attempts to change a master record which does not exist), the overnight processing will be successful.

25. Concentrate Update Activity: With a data base containing redundant information in several files, or in an integrated data base which has redundant information in secondary indexes, multiple accesses are required to complete a single transaction. It is useful to establish a programming standard to minimize the risk of creating an incompatible file set due to an abort when processing an update. At the risk of adding a few file accesses, all reading of all files can be accomplished before processing and all changes to all files can be collected into a single compact write sequence following processing, i.e., reads and writes should not be distributed uniformly across the processes. By concentrating the writes, the probability is reduced that a processing failure will prohibit the sequence of writes from being completed and hence leave the file set in an incompatible state.

Program Structure

26. Interpretive Change: One large industrial firm has found that some programs unintentionally modify more data than they should; in an integrated data base, storing a field one byte too long destroys the adjacent field. A table driven interface module can be appended to the record fetch routine supplied by the data base management system. Thus, a program can get any data it desires, but can only put back data it is authorized to change. Further, the data being placed in the record is edited to assure that it matches the data element specification for the element addressed.

Programming Technique

27. Workspace Integrity: Fill the unused space in a data record and the unused space in a memory allocation with a known pattern of bits. Some programs fill with a supervisor call to an error routine. Then, by checking the pattern, it can be determined that the unused space is in fact unused and stays that way.

28. Space Erase: After each message/input transaction has been processed, some large systems obliterate the residual data to keep previous values from being accidently used in subsequent processing. To get full benefit from this technique, the pattern used to obliterate data should not be a legal combination of characters (blanks, zeros, nines, etc.) because a program which fails to initialize its workspace could still run properly under some circumstances. Note that this is one of a set of techniques designed to keep a program within its allocated data and instruction space and to assure that the program so constrained controls its entire environment but uses only data that it controls.

One efficient way to set unused space to null values is to obtain the actual length of the record just read from the read routine. Then, store null values in any allocated but unused space set aside for the record. Assuming update-in-place, the original record length should continue to describe the length of the occupied space unless it is consciously adjusted by a program adding a field, deleting a field, or extending an existing field. After all processing is over, the updated record length can be used for two purposes: to write the modified record, and to check that any residual unused space is still filled by the null values stored after the read but before the processing sequence.

Editing

29. Classes in Edit Specs: Many financial systems have established credit classifications and adjust a credit rating dynamically based on the customer's basic rating and the current status of his account. After the adjusted rating is established, the editing system invokes dollar limits appropriate to the customer's adjusted rating.

Similarly, inventory systems establish classes for customers and edit order quantities for reasonableness against those customer classes.

For example, a utility might establish four classes of subscriber service: apartment, single family residence, large residence/light industry, and heavy industry. As usage meter readings are processed, edit criteria appropriate to the service classification would be invoked. Thus, erroneous readings, overdue bills, or unusual usage would all be recognized since the usage and the edit criteria would be compatibly mated due to the gradations established by the classification system.

30. Classes of Limits: Many data base systems keep an order quantity limit on the master file along with the product description and status. If a quantity requested exceeds this limit, manual intervention is requested to confirm the order. In more sophisticated systems, a series of limits are stored by customer class so the small customer is protected against over ordering by a low limit.

31. Normal Limits: When many independent transactions are received related to members of a homogeneous group (of customers, individuals, parts, or accounts), an aggregate average across all members of the group will allow the computer system to determine if any individual member of the group deviates significantly from the average. Furthermore, if the group average deviates unpredictably from the previous average for this same group, the entire batch of transactions can be held until the deviation is explained.

32. Trend Limits: In processing input transactions, the behavior of an individual can be compared to the aggregate behavior of all individuals within his population (as described above) or an individual's behavior can be compared against his own previous behavior. The latter requires a behavior history to be maintained by the computer for each account so previous account activity is retained for future comparison. Both methods are popular in the utility industry. In the latter case the history record is adjusted for seasonal and temperature effects before the definition of "normalcy" is determined.

33. Error Suspense Files: When input transactions are lengthy and when error experience shows corrections usually affect only a small portion of the original record, suspense files containing original records (complete with errors) are frequently maintained. The correction process then consists of reentering identifying information plus the fields in error. The computer then updates the queue of incorrect transactions, usually by simple field replacement, and then passes the modified transaction back to the main input editing module.

It is important that the entire corrected transaction be passed through all of the editing procedures and that no short cuts be taken assuming what was valid before is still valid.

When systems are designed which maintain queues of transactions pending correction, one good practice for uniquely identifying the error to be corrected is to cause the erroneous data to be reentered along with the transaction I.D. and the replacement data.

34. Maintenance of Limits: Several editing schemes require history data stored in a file to establish the editing limits and to adjust them based on seasonal, time, activity, or cyclic factors. Note that these systems require edit, update, and reporting modules to maintain the file of edit limits. Further, such systems require special data or additional programming to initialize the system at start-up prior to the time any real history is available.

Post Processing Verification

35. Dummy Processing: A frequently used practice places dummy master records on the file and routinely processes fictional transactions against those dummy master records to produce predictable outputs. The results of processing go to the control group which routinely checks that the item counts for all input items were processed and verifies the control totals for those items. The results from the processing of fictional data against the dummy accounts is also verified to assure that in these particular instances the actual detailed processing was correctly performed. In the case of a payables run, the check written for the dummy account must be captured and destroyed.

The above technique works on many types of files. Consider establishing a record for a dummy part number in an inventory file, and processing fictitious receipt and withdrawal transactions against that part number. If one understands the inventory algorithm imbedded in the processing program, input data can be prepared to test limit checks and a variety of transaction processes. In all cases the resulting inventory-on-hand is predictable and the processing procedures can be checked by verifying the computer produced results against the predicted values. Sets of dummy records and dummy transactions can exercise a computer program and provide the manager of controls with confidence that typical transactions are being processed correctly.

Trouble Symptoms

36. Analyze Exception Reports: As can be inferred from many of the previous comments, the computer can be made to reject obvious mistakes and to process suspicious transactions while listing their occurrence on an exceptions report for manual review. The very quantity of items requiring review is an excellent indicator of the general health of the system. In addition, the flavor of the exception reports provides an indication of whether the trouble is consistent or, in the absence of a pattern, merely related to individual transactions. Exception reports might be only big ticket items, all subscribers residing in California, just items manufactured by the XYZ Company, only items input on December 15, etc.

37. Performance Change, A Trouble Symptom: Performance changes are an indication of general systems health, provided the programs have not been modified since the previous run. It should be routine to notify the operations staff whenever a program is changed. Further, the operations staff should be provided with some indication of the processing pattern so they can sense performance changes during lengthy runs. For example, staff can be provided an audio amplifier for CPU activity, a CPU activity meter, a hardware measurement box, or a printout of the CPU time used per transaction.

38. Inform Operator of Performance Anomalies: In some cases estimated run time can be predicted by an equation based on a simple count of the input transactions. In other cases run time must be estimated by a more complex equation which operates on the number of transactions in each of several transaction classes. However, in yet other cases the transaction run time will vary unpredictably based on the parameters in the transaction and the constitution of the values contained in the files. Commands which select records from a data base can cause varying percentages of the entire data base to be read in response to an inquiry. In these cases operators should be provided with in-process progress reports (percentage of the transactions processed, or percent of the data base read) so they know that an unpredictable process is progressing satisfactorily.

The purpose of informing the operators whenever unpredictable transactions are being processed is to define, by implication, that the remainder of the processes are predictable and hence performance should conform to historical expectations.

A large file system should provide a communication path from the computer operator back to the system manager so unexpected activities can be reported for review.

39. Analyze Adjustments: As discussed in the previous section on editing, the computer can be programmed to recognize unusual transactions. In the utility industry it is traditional to keep historical usage in the customer's record, adjust it for seasonal and temperature variations, and use this to screen the customer's current usage for reasonableness. After all the out-of-character transactions are posted to an exception list, those exceptions must be manually reviewed to determine if they are errors or not. Sometimes these reviews require additional information and if that introduces delays in the system, a standard invoice is sent and an adjustment made in a later cycle after the necessary data has been acquired.

The volume, magnitude, and sense of these adjustments provides an indication of troubles in the flow of data to the computer. Some of these troubles may reside in the electronic data processing system itself, whereas others may reside in the upstream procedures, or in the techniques used for gathering the basic data.

40. Analyze Error Logs: All errors should be logged. In addition to recording the error and storing a unique error code, an error log should contain enough information to identify the discrete data path from the originator to the computer (source, observer, keyboard operator, batch, dates, etc.). The error log must contain the information necessary to identify the input transaction in its own local processing sequence. It should present the customer classification awarded by the input screening process, any classification associated with the master file, the type of error discovered, and should aggregate statistics on error trends during that processing run, day, week, and month; the entire context must be logged with the error.

Careful analysis of such an error log will pinpoint error prone processes or deficiencies in the system design which can be improved for economy and efficiency. Systematic improvements are usually accompanied by an extra bonus since a high volume of routine error may cause personnel to expect a high volume of system errors and to be less cautious in their own activities.

Optional Checking

41. Adaptive Checking: Computer operators learn the processing rhythm of a large job even when multiprocessing. In the past, systems have been constructed with optional checking features which can be invoked either by internal checks or by operator action. One early system had a series of expensive checks designed to guarantee the continuing integrity of the data base, but they were inhibited during normal operations

and invoked whenever the operator desired confidence that a suspicious processing pattern was not destroying the integrity of the data base.

42. Program Integrity: Some program designers structure their object program modules into read-only (re-entrant) areas and into read-write (serially reusable) areas. After loading, a simple checksum on the read-only areas will provide an initial value so that subsequent checks at breakpoints in the process can assure that the program and its tables remain as loaded and unmodified.

More sophisticated checks are required to verify the patterns in the read-write areas. If these are performed at major breakpoints in the process, the parameters and most work space usage can be verified, thus proving the program is behaving to the extent that the output areas are properly configured and formatted.

43. Automatic Error Analysis: One leading manufacturer recently enhanced his operating system software to provide increased availability. He determined that system crashes were accompanied by incorrect modifications to the control blocks or parameter strings which dynamically controlled the operating system. To minimize the effects of an error and allow the system to be restarted with the greatest portion of the workload surviving, a series of optional programs called Checker/Fixers were provided. When called during the restart process, each checker verifies the format and contents of a control block. If the block is incorrect, the checker optionally calls the fixer to rebuild the control block from other available information stored redundantly within the system.

Many commercial applications operate on several separate strings of data simultaneously and can benefit from an adaptation of the same technique. First determine the extent of the error and then restore operations as soon as possible for the healthy parts of the system. Massive system restarts provide opportunity for compound failures since restarts frequently violate system controls and usually cause all operational personnel to follow unfamiliar and sometimes untested procedures.

44. Verify As You Duplicate: Every data base has some built-in redundancy. A well designed data base may have considerable redundancy. As discussed earlier, a separate audit program can be created to check that redundancy and verify the integrity of the file set. If properly programmed, some sections of that audit routine can be imbedded in a file duplicate

program so the process of creating a backup file also checks that file for correctness.

A simple utility program provided by the manufacturer will copy records from one file to another treating each record only as a string of bits. A more sophisticated utility program would first obtain the file description from the data base dictionary. As each record was copied, the program would check that the specification for data elements, detail records, summary records, and the file structure was inviolate. Since 100% of the file is handled every time the file is duplicated, a wedding between the audit program and the file duplicate utility produces a major benefit at a small incremental cost. Note that if the file is self-defining, the audit/duplicate program need not reference the data base description. This procedure works equally well on current files and archival files even though the file definitions may have changed over time.

B. SCIENTIFIC SYSTEMS ANALYSIS

From the early to mid-1950s, the preponderance of information processing was scientific computing. Persons performing this work had degrees in mathematics, engineering, physics, or other hard sciences. However, even that schooling did not prepare these early programmers in numerical analysis. After considerable difficulty, programmers were trained to know the difference between accuracy and precision, to be cautious when scaling, and to be skeptical of all floating point calculations.

Twenty years later those hard-learned lessons have all but disappeared since many of the early practitioners have moved to non-technical assignments and the computer population has exploded. Even though the colleges today teach classic problems in numerical approximations, many statistical, engineering, or mathematical computations programmers are not aware of the possible pitfalls. A sampling of these problems follows:

45. Arithmetic Representations: Every computer has its idiosyncrasies that give rise to arithmetic singular points. Prior to performing extensive math on an unfamiliar computer, each programmer should discover the arithmetic singular points in the system to determine if these idiosyncrasies will affect the planned calculations. For example, if two negative numbers of the same magnitude are subtracted, some machines yield a zero with a negative sign. If this result is then presented as an argument to a subroutine which accepts only positive arguments, it will be rejected.

46. Finite Arithmetic: Arithmetic operations using the full range of a computer's arithmetic registers, emphasize the importance of round off, truncation, and the residue following arithmetic manipulations. Even though programmed double precision operations may not be contemplated, each numerical programmer should discover how the arithmetic behaves at the extremes of the numeric ranges, and whether the residual values are reliable or not.

Since some systems behave differently when performing fixed point, complement, and floating point arithmetic; the results from each of these types of computation should be reviewed.

47. Numerical Hazards: Programmers should be cautioned to beware of the singular points which are created by performing arithmetic on nearly equal floating point numbers in the middle of a complex expression, the possibility of floating point underflow and overflow, and the hazards of division by zero.

48. Calibrate Numerical Approximations: Many manufacturers offer subroutine libraries and in some cases built-in transcendental functions. Frequently, these are represented as primitive computer operations and treated routinely as an extension of the order list. Each user should calibrate the subroutine library to be used, and publish in his programmers' manual the accuracy achieved by the algorithms it contains. Considerable effort will be required to validate a library of subroutine approximations.

49. Parallel Check Calculations: In many scientific calculations, there are frequently two different sets of equations which produce the same result or, in some cases, an additional equation can be introduced which exploits some other physical property and provides a check calculation. By introducing such redundant calculations, the correct operation of the computer system can be verified and as a by-product, the accuracy of the computer solution can sometimes be estimated. For example, integrate an energy balance equation along with the equations of motion for a missile. The continuity of mass in the energy equation will guarantee that no gross errors have been made as the set of differential equations are integrated.

50. Direct Errors with Numerical Aggregates: In repetitive scientific calculations dealing with many observations of well behaved data, standard statistical techniques (mean, variance, skewness, kurtosis) will detect gross errors in the observations or subsequent processing. An example outside the fields of science would be the use of statistical measure to verify that stock market transactions were properly reported and processed.

C. IMPLEMENTATION

In the past few years the procedures for programming and the systems which support programming development have been given a lot of attention. Structured programming, structured analysis, modular design, chief programmer teams, code inspections, test case inspections, and documentation inspections are all techniques aimed at producing reliable programs true to the programming specification. Assuming that the programming process is well disciplined and that these modern management techniques have been adopted and adapted to local needs, there are still a few specific techniques which will lead to further improvements in large information processing systems. These techniques are:

Development Methods

51. Pilot Test Input: When designing a big high volume input system, program the input system first and pilot test it under actual or simulated conditions while the rest of the system is being completed. Use the statistics from these pilot tests to verify the design, its documentation, the planned training, and most importantly to check the assumptions on which the input system design was based.

52. Manage Keyboarding Errors: In big shops having a high volume of similar keyboarding, keep error statistics by machine and by employee. By trying various employee-machine combinations, the error statistics will isolate employees with the best and worst records. Analyze these statistics to benefit from the techniques of the superior employees and to retrain or otherwise modify the habits of the inferior employees.

System Testing

53. Data for Test: To provide a test environment without the risk of allowing an unchecked program access to real files, keep a set of test files whose structure is identical to the structure in the real live data base. Have the operating system select the test or real files depending on a control parameter which designates whether the run is test or production. Insist on seeing satisfactory runs on the test file before a modified system is allowed to run production.

54. Built-in Verification Tests: When running many data cases through a complex calculation, routinely run a standard case before, during, and after the massive set of live data. This common practice will verify that the program and the computer system were operating correctly (at least for the paths exercised by this standard case) throughout the production run.

55. Built-in Exercisers: In an interactive system, catalog a standard demonstration program and provide each remote terminal operator with a script that accesses standard data, exercises the system, and produces predictable results. The remote operator can then be assured of an operational terminal, communications network, central computer, operating system, and at least one operational path through the applications system and the data base management system. With experience, the remote operator will also be able to determine if the system response time is within normal limits.

56. Built-in Diagnostics: In the event a remote terminal operator invokes the standard exercise package and receives unexpected results, a separate input diagnostic package can be invoked so the operator can follow a predetermined input script and the input diagnostic package can determine if the character strings received are exactly those expected.

Similarly, a file of predetermined or canned output can be stored and an output diagnostic program can be invoked to display the canned outputs to the remote operator. The operator can then determine if the standard outputs are being received and properly displayed/printed. The canned outputs stored for such a diagnostic program are usually geometric in nature. Patterns are displayed on the CRT screen or printed at the hardcopy terminal to place known characters in the first and last display position of each and every line, cover the display field with a readily recognized pattern, and display every graphic character to assure the character generation circuitry is properly functioning. Similarly, all electromechanical functions of a hardcopy terminal would be exercised.

Given the results of the input diagnostic and the output diagnostic, the remote operator will then be able to intelligently call for assistance and indicate what the problem seems to be.

57. Communications Exercises: Communications lines which interconnect intelligent devices such as intelligent terminals to computers and computers to intelligent multiplexors can be directed to repeatedly exchange standard messages to calibrate the line. If the measured error rates are excessive, the line can be scheduled for repair. In the interim the line speed can be changed, alternate network paths can be used, or line capacity can be exchanged for optional checking.

58. On-Line Testing: In any system which involves elaborate error checking, provision must be made to declare entry into a "test mode" so the fault detection logic can be exercised by introducing faults into the system. The purpose of the test mode is to inhibit the sounding of alarms, the printing of

error messages, and the logging of test events as actual errors thereby destroying the integrity of the error statistics.

In the past some computer systems have been designed with hardware and software that automatically entered a test mode during periods of light workload so the system was self-exercising.

D. ENVIRONMENT

Most systems designers realize that a successful system design requires the designer to consider the staffing, the skills available, and the organizational structure in which the system is to be imbedded in addition to the obvious concerns of hardware, software functional requirements, and a statistical profile of the workload. Some systems have failed when the designer overestimated the ability of the system's operators to cope with the complexity involved. Other systems have failed to produce predicted savings due to the flow of errors recycling between organizational units.

Traditionally batch systems were set up around existing organizational lines with the system functions assigned in harmony with the organizational responsibilities. On-line systems with integrated files frequently alter the flow of basic information and require the organization structure to be tuned before the systems will operate satisfactorily*. This makes it appear that the organization must be changed to accommodate the computer system. However, the organization must be changed to achieve the desired goals, and the computer is part of the new system designed for the new goals.

---

*Consider the organizational changes required when the traditional keypunch/batch balancing section is eliminated by on-line input from terminals remotely located in user departments and operated by user personnel. Not only does the flow change the personnel, but the skills required and the responsibilities change too. Or consider the changes required to go from 100% manual inspection of computer output to table-driven computer output screening with manual review of the exceptions selected by the computer algorithm. In one case the volume was enough to keep several clerks busy and require written manual procedures; in the other case, no pattern ever emerges and the maintenance of the control table is a new responsibility levied on the systems administrator.

Unfortunately some senior managements do not recognize the need for organizational change until they have spent money implementing an on-line system and cannot get it to work without changing some responsibilities, some budgets, and training or replacing some key managers. The items that follow are organization related and probably would not appear until trouble was encountered.

Organizational Topics

59. Data Base Administrator: Even if all the files are related to a single system of application programs and even if no third party uses the data being created, the concept of a data base administrator, separate from either programming or operations, is extremely appealing. As systems get larger, it is natural to split data base administration from development programming, maintenance programming, and production operation. This pares the jobs down to reasonable size and concentrates a series of like functions in the hands of a neutral administrator.

The data base administrator and his staff should address:

definition,	data elements,
controls,	formats,
procedures,	records,
backup,	files, and
transitions,	file sets.

In addition, the data base administrator should control and be solely responsible for the use of file utility programs to:

audit,
verify,
duplicate,
reconstruct,
initialize, and
reorganize.

60. Organize for Integrity: Organizational responsibilities will need review, programming standards will need modification, and the workforce will need discipline if changes to a production system are to be controlled so they prohibit loss of data integrity over time.

Naturally, data stored and untouched in a perfect world does not deteriorate. However, machines do make mistakes, operators executing unfamiliar procedures make mistakes, programming mistakes go uncaught, and the correction of mistakes sometimes breeds other mistakes. Thus, in an imperfect world, data seems to decay over time even though no transactions have been processed against the master file which update specific individual records. Changes must be controlled and files must be audited

to maintain their integrity over long periods. The organizational responsibilities must be properly defined (and the files must be properly designed) or this will not happen.

61. Quality Control: Most production computer shops have a team of individuals called a control group through which all computer input and output must pass. These personnel check the control totals on input batches, and reconcile totals on output reports against the most recent input and the current file statistics. The assurance function requires that report control totals be reviewed before the information reports or other computer produced documents are distributed. In addition to input/output totals, statistics to check file integrity and histograms of the input transaction mix will allow deviations from the norm to be caught before errors are broadcast to the outside world. Unfortunately, such a control group must work when the production is being run, and that usually implies second shift and weekends.

62. Masterfile Verification: A properly designed data base will contain enough redundancy so its integrity can be independently verified without reference to other files, indexes, or supplementary information. Naturally a computer audit program would be used to perform this verification. Typically the file structure would be explicit and additional redundant data elements would be included which contain item counts, numerical sums, or hash totals. For more complex files, truth tables describing the logic and record sequencing would be placed in a descriptive record at the beginning of the file. The data base administrator should own the data base audit program and run it for the benefit of all users sharing data. If the program is expensive to run, samples of the files can be checked on consecutive runs. Runs can also be triggered by trouble symptoms.

The data base administrator and the data librarian must coordinate their efforts to assure that the release of grandfather files depends on both age and proven data integrity. Whenever a 100% sample of any file within a file set is checked, the audit program should verify all file controls and totals. Any errors found should be resolved by the previously mentioned control group so the responsibilities of the control group are not abrogated and so the baseline totals for the next set of production runs are automatically adjusted.

The relationship between audit activities and the file retention plan deserves further comment. When a computer center deals with a master file which is cyclically updated, it is traditional to establish a retention plan which defines how many consecutive obsolescent versions of the master file shall

be retained. In batch operations with weekly updates, the current file plus last week's file and the master file of two weeks ago are traditionally saved (current copy/son, father, and grandfather versions).

Now if the audit program is run every two weeks to verify the integrity of the master files, then a retention plan providing for three cycles is sufficient. However, the file set may be massive; or the audit program long-running; or the audit program set up to do 100% verify of only a portion of the file (a progressive audit so the entire file requires several audit cycles to completely verify). In these cases, the audit cycles must be considered when preparing the retention plan as it may be desirable to retain additional cycles of the obsolescent master files to accommodate the audit cycle, i.e., the audit cycle and the retention plan cannot be independent of each other.

63. Local Expertise: With the advent of user-oriented programming packages that are juxtaposed to programmer-oriented programming packages, a hazard appears to be mixed in with the blessings. Many of these packages are sold in a ready-to-use state, and little or no programming talent is required for their installation and use. As long as these packages are used for non-essential work, the hazard remains hidden. But whenever these packages enter the mainline production flow, or they are used to summarize important data regularly used for important decision making, the hazard is exposed.

Briefly stated, the packages are sold assuming no local programming expertise is required. Thus, in most cases, no local programming expertise is developed. An installation can be successful without having copies of the source programs, without knowing the details of the file layout, and without understanding the intricacies of the restart/reload procedures.

With luck, an installation will never need expertise on the internals of the data base package. However, power failures do occur in the middle of updates, computer circuits sometimes fail, and programs stored in a library have been known to deteriorate or become incompatible with the operating system over time. Thus, these systems could fail at the time you most need the results. While the vendor will assist in diagnosis, long distance problem determination is sometimes a chore and the service you received when the package was new is bound to deteriorate over time.

Thus, if a programming package is used for mainline work so production must be completed on schedule, the prudent manager would build some level of local expertise as insurance that he has the skills he needs when an emergency arises. This should

not be interpreted to mean that every installation needs a seasoned programmer who is thoroughly conversant in the internals of every package. More properly, each manager should assess his risks and develop the level of expertise he feels necessary to hold those risks to a reasonable level.

Operational Features

64. Pilot Production Batches: On large production runs, the control group can be given a running start and the probability of meeting deadlines can be increased if a small but representative sample of data is routinely run as pilot production. The control group can then check the output of this pilot run as the main run is processing. Note that sometimes this technique requires minor changes to the system design to be feasible.

65. Integrity in Names: Check digits on procedure, job, and program names can be used to prevent setup mistakes or operator type-ins from calling for incorrect procedures/jobs. While such incorrect requests are most likely to abort, some will partially execute and they can destroy a portion of any data base designated for output before the catastrophe is recognized.

66. On-Line Controls: While many information systems accept inputs in random sequence and proceed to process the input stream in "as received" order, some installations have found that controls, restart, and error resolution are greatly simplified if the input transactions are sorted into ascending sequence on the primary file key before the main update run is scheduled.

About 10 years ago, with the advent of reliable disk files, data base systems were promoted on the basis of on-line operation that would allow remote transactions to be entered immediately and to be instantaneously reflected in the data base. However, this on-line operation with instantaneous update from remote terminals bypassed all of the traditional checks and balances that had been set up first in manual systems, and later in batch systems.

Many systems are running today with on-line update from remote terminals because an adequate system of controls has been satisfactorily automated. However, in some very large and very complicated systems, on-line controls at the transaction level proved difficult to achieve and these systems found it necessary to batch on-line transactions, sort these transactions into some sequence, and supply a more traditional form of batch control procedure before the transactions are allowed to update the file.

In the old days, paper was physically moved and a batch might accumulate for a week before a file update was scheduled. Today, systems move transactions from remote locations electronically and batch them until a sufficient volume has been accumulated to allow processing to be effective and efficient, usually a period of several hours.

While this situation was originally recognized in a large data processing center handling big files, the same conditions could occur in other environments. This is particularly true when the batch controls require trend analysis or statistically significant samples to prove the batch before update is scheduled. Obviously a batch consisting of a single transaction is seldom statistically significant.

67. Remote Backup: If continuing operation from an online system is an absolute must, and if two separate remotely situated computer systems are justified, then maintain a duplicate file set and duplicate control tables on both systems so the basic data and the network context are available whenever the terminals are switched from the primary to the backup system. Note that some systems will also require the log file with its before and after images to be maintained dynamically at both sites.

V - SUMMARY

The designers of large application programs must work hard to do a creditable job in today's environment. If the applications system is to be long running, restart must be considered. If the program runs frequently, it must be tuned to efficiently exploit the operating environment. This report has shown that other tuning is also required.

Audit trails require careful design if they are to remain unbroken in all circumstances. Files must have controls designed in; it is hard to add them later. If the mathematics of processing exceeds simple arithmetic, it sometimes requires special attention, too. More and more it is becoming apparent that a big data processing system requires careful design attention to be given both to the computer processing and the manual processes such as data capture, balancing, error correction, reports distribution that support the computer system.

If a data processing system taxes the hardware, involves careful consideration of individual privacy, or is geographically distributed, the challenge is greater, the hazards more pronounced, and the possibility of failure quite real.

The designer's best defense is team work. Suppress your ego and request a peer review of your design, organize your experience into checklists so you avoid repeating past mistakes, exchange checklists with your colleagues so you can profit from their knowledge, but above all don't expect perfection. Plan for the unexpected and contain the errors as they occur.

A manager cursed with operating a troublesome, error-prone system has a different problem. An inadequate system design, a system carrying data volumes in excess of its design limits, or a system whose environment has changed drastically needs money and talent. No amount of exhortation or training will correct a system that is badly matched to its environment. Instead you need to measure the error frequencies and their impacts, review the basic design, and reaffirm the design goals. Given a preliminary review, an action plan (more money and talent) can be prepared, sold, and scheduled. If you have to rework an existing system, first collect ideas such as those presented in this report. Not all of them are expensive to implement.

Robert L. Patrick
Northridge, California
1 August 1977

APPENDIX A

REFERENCES

1. Comptroller General of the United States. Improvements Needed in Managing Automated Decisionmaking in Computers Throughout the Federal Government. GAO Report #FGMSD-76-5, (April 23, 1976), 72 p.

2. Guide for Reliability Assessment of Controls in Computerized Systems: Financial Statement Audits. U.S. General Accounting Office, Exposure Draft; (1976).

3. Spier, Michael J. A Pragmatic Proposal for the Improvement of Program Modularity and Reliability. International Journal of Computer and Information Sciences, 4:2, (June 1975), pp 133-149, 18 refs.

4. Gilb, Tom, and Weinberg, Gerald M. Humanized Input. Winthrop Computer Systems Series; (1976).

5. Martin, James. Security, Accuracy, and Privacy in Computer Systems. Prentice-Hall, Inc., Englewood Cliffs, NJ; (1973), 626 p., 102 refs.

6. Phister, Jr., Montgomery. Data Processing Technology and Economics. The Santa Monica Publishing Co., Santa Monica, CA; (1976), 573 p.

7. Miles, Lawrence D. Techiques of Value Analysis and Engineering. McGraw-Hill, New York, NY; (1972).

APPENDIX B

ABBREVIATED GLOSSARY

(Only Words Whose Usage May Be Ambiguous Appear Below)

Activity Ratio - A phrase used by file designers to express the proportion of records in a given file likely to be read for processing in a given run. Expressed as a percentage of the records read divided by the total number of records in the file.

Application - An adjective, as in application program or application system, describing custom written code to fill a specific local information processing need. (Contrast with Software.)

Binding - The process, performed during program preparation, where flexibility and ambiguity are systematically removed from a computer program until an executable object program results. Binding takes place in stages starting with the programmer's original conception of the problem solution through compilation, loading, and the dynamic setting of absolute actual machine addresses immediately prior to an instruction's execution. Different types of variables are bound at different stages along the spectrum of binding time. In general, earlier binding means less flexibility and more (machine) efficiency. Later binding means flexibility at the cost of machine cycles. (Note: Each instruction must be completely bound at the instant it is executed.)

Commercial - An adjective, as in commercial computer, meaning a system offered for sale and listed in the GSA catalog. Contrast with special purpose, custom hardware prepared for military, space, or special industrial purposes at negotiated prices. A commercial computer is capable of performing both business and engineering processes.

In some segments of the computer community, the word commercial is used as an adjective to modify the phrase "data processing". In this context commercial data processing is used synonymously with business data processing, and in juxtaposition to scientific data processing. Throughout this report, the unadorned phrase "data processing" is used to denote all of these contexts and the word commercial is used to denote the manufacturer's stock price book offering.

Context Edit - An input edit performed on a specific field within a logical framework formed by combining adjacent fields, information distilled from previous inputs processed in the same run, and data retained from fields processed in previous runs. Contrast with field edit where an individual field is processed in isolation, against only its own field specification (containing only the legal character set, length, numeric scale (if appropriate), and a list of (local) standard value encodings (if appropriate)).

Data Base - A collection of related files (usually including some redundant data values) organized for some processing purpose(s). Thus, a data base contains fields within records within files within the data base. In addition, the data base organization (structure) usually contains a collection of explicit control fields which enhance the processes of update and retrieval from the data base, and allow the system to provide security, integrity, problem determination, efficiency, or accounting enhancements not available in the raw file set.

Data Management System (aka Data Base Management System) - A set of highly integrated file handling programs which exploit the control structure of the data base to retrieve data from the file set and add/change data compatibly within the file set. In addition, a DMS maintains a log for restart purposes, provides for start, restart, orderly shutdown, access controls, security, integrity, and accounting. There are all shapes and sizes of data base management systems. Some are integrated with the operating system and provided at no additional cost by the computer vendor. Others are optional and priced separately. DM Systems are available from computer manufacturers and from independent software houses. They vary dramatically in function, sophistication, size, and performance.

Data Processing - A phrase intended to mean the totality of computer activities involving processing in the presence of standing data files. Synonyms sometimes used are automatic data processing (ADP), business data processing (BDP), and commercial data processing (CDP). The phrase is sometimes contrasted with scientific data processing where the form of the mathematics involved is likely to be somewhat higher. In this report, the phrase "data processing" is used to denote the common input-process-output sequence performed in the presence of organized files of data for accounting, bookkeeping, or the general processing and maintenance of records on people, parcels, and things.

Default Options - A programming technique used to arbitrarily insert a missing data value whenever the program can determine, with a high degree of certainty, what the missing value should be. The insertion of default values during the input processing greatly simplifies the downstream processing since downstream processes need not be programmed to accept records with missing data values.

Driver Code - A colloquial name for special temporary program modules used to artificially create an environment so a module of deliverable code can be tested separately, in isolation, from other deliverable modules in the main program set. In addition to its temporary purpose in support of program testing, driver code is usually informally designed and informally documented. Driver routines are usually discarded when the modules of the production system are integrated for shipment.

Hash Total - An arbitrary arithmetic sum of a series of numeric fields used solely to determine that the number of items summed and their data values remain intact through a series of processing sequences. If the data elements being summed have homogeneous field specifications (as in money amounts from cash transactions), the total would be called a control total. However, if the field specifications vary, yet it is desirable to have an arbitrary total to prove the integrity of the batch, then the fields are totaled without regard to their field specifications and a hash total results.

JCL - An abbreviation for Job Control Language (which in turn is a misnomer) which conveys a series of control parameters to the operating system at the time a computer program is submitted for execution. The JCL parameters describe the environment required for the processing of the applications program and further provide certain auxiliary information which the operating system uses in the event the program does not complete satisfactorily. Included with the JCL parameters are the names of all files the program uses and the names to be attached to all files the program creates. JCL exists even for on-line programs, but their environments (and hence their JCL) are usually pre-stored within the computer system and not entered explicitly as jobs are initiated.

Performance - A noun, as in system performance, or an adjective, as in performance assurance, meaning functional allegiance to a specification. In contrast with the usual narrow meaning operations per unit time.

Self-Cleansing - A property of some files whose processing causes the contents of every single data field to be verified during each processing cycle. The pay file is typically used as an example of a self-cleansing file. It enjoys an activity ratio of 100% when pay checks are being written and presumably each check is individually scrutinized by the payee. Contrast this with a file of library bibliographic records which may have an activity ratio of 30%, and even when a record is active, does not check all the contents nor provide information to a downstream process which checks all the contents. In the case of the pay file, erroneous records are likely to be quickly found and corrected. In the case of the bibliographic file, erroneous records are likely to go uncaught for long periods of time.

Software - A noun, as in computer software, meaning computer programs of a general utility nature that are fundamental to the operation of a computer system and usually provided by the hardware manufacturer or by a major systems development effort. (Contrast with Application.)

Stock Commercial (Computer) Equipment - Computer equipment installed by its original manufacturer and maintained in pristine condition according to that manufacturer's specifications. Specifically, equipment that is designed for a general purpose business marketplace,

operates in an office environment, and is not modified or redesigned in any way by the purchaser, i.e., stock as in stock car.

Value Analysis - An engineering analysis technique, developed shortly after World War II, which allows the value of an article to be independently assessed and then compared against its cost. When this comparison shows a major disparity, the costs, the manufacturing process, and eventually the engineering design are reviewed to get the actual cost consistent with the awarded value.

APPENDIX C

INDIVIDUALS CONTACTED

#	Name	Affiliation
1.	Leland H. Amaya	Securities Industry Automation
2.	Lowell Amdahl	Compata
3.	Frank Anzelmo	Columbus Dispatch
4.	George Armerding	Capital Data Systems
5.	Herb Asbury	WESRAC
6.	Lynn Beck	Hughes Aircraft
7.	Dr. Tom E. Bell	Peat, Marwick, Mitchell
8.	Mort Bernstein	System Development Corporation
9.	Erwin Book	System Development Corporation
10.	Dr. Fred Brooks	University of North Carolina
11.	Dr. Robert R. Brown	Hughes Aircraft
12.	Richard G. Canning	Canning Publications
13.	Richard P. Case	IBM Corporation
14.	Doug Climenson	Central Intelligence Agency
15.	Don Cohen	Bank of America
16.	Dr. C. G. Davis	BMD Advanced Technology Center
17.	Edwin Derman	BankAmericard
18.	Phil Dorn	Consultant, New York City
19.	Hank Epstein	Stanford University
20.	Larry Foster	IBM Corporation
21.	Joe Fox	IBM Corporation
22.	Dr. Bernie Galler	University of Michigan
23.	Tom Gilb	Consultant, Norway
24.	George Glaser	Consultant, San Mateo, Ca.
25.	Ted Glaser	System Development Corporation
26.	Dr. Joe Gloudeman	Rockwell International
27.	Milt Goldwasser	Los Angeles Times
28.	John Gosden	Equitable Life Assurance Society
29.	Irwin Greenwald	Honeywell
30.	Dr. Maury Halsted	Purdue University
31.	Don E. Hart	General Motors Research
32.	Herb Hecht	Aerospace Corporation
33.	Richard Hill	Honeywell
34.	Edwin L. Jacks	General Motors Corporation
35.	Phil J. Kiviat	Federal Simulation Center
36.	Jack Little	Planning Research Corporation
37.	G. B. McCarter	IBM Corporation
38.	Dan McCracken	Author, Ossining, New York
39.	Richard McLaughlin	Datamation Magazine
40.	Harlan D. Mills	IBM Corporation
41.	Dr. Norm Neilson	Stanford Research Institute
42.	Donn Parker	Stanford Research Institute
43.	Bill Payne	Lockheed
44.	Herb Safford	GTE Data Services

#	Name	Affiliation
45.	Dr. A. L. Scherr	IBM Corporation
46.	Dr. Norm F. Schneidewind	Naval Post Graduate School
47.	Keith Uncapher	USC Information Sciences Institute
48.	Jack C. VanPaddenburg	Southern California Edison
49.	Dick Van Vranken	Aerospace Corporation
50.	Frank Wagner	Informatics
51.	Dr. Ron Wigington	Chemical Abstracts Service

APPENDIX D

ALPHABETIC SUBJECT INDEX

☆ U. S. GOVERNMENT PRINTING OFFICE : 1978—261-238/437

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-24	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

Performance Assurance and Data Integrity Practices

5. Publication Date: January 1978

6. Performing Organization Code

7. AUTHOR(S) Robert L. Patrick
Editor(s) Robert P. Blanc

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Robert L. Patrick
Computer Specialist
Northridge, California 91324

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address (*Street, City, State, ZIP*)

National Bureau of Standards
Department of Commerce
Washington, D.C. 20234

13. Type of Report & Period Covered: Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 77-608309

16. ABSTRACT (*A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.*)

This report identifies the approaches and techniques now practiced for detecting, and when possible, correcting malperformance as it occurs in computer information systems.

This report is addressed to two audiences: to the systems designer using stock commercial hardware and software who is creating a system which will tax the available hardware, software, or staff skills; and to the manager who wishes to chronicle the deficiences in an existing system prior to improvement. It enumerates 67 items of current practice which prevent computer malperformance.

17. KEY WORDS (*six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons*)

Best practice; data integrity; data processing; error correcting; high integrity; performance assurance.

18. AVAILABILITY

[X] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[X] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13.10:500-24

[ ] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT): UNCLASSIFIED

21. NO. OF PAGES: 53

20. SECURITY CLASS (THIS PAGE): UNCLASSIFIED

22. Price $2.20

USCOMM-DC 29042-P74

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology, and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent NBS publications in NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $17.00; foreign $21.25. Single copy, $3.00 domestic; $3.75 foreign.

Note: The Journal was formerly published in two sections: Section A "Physics and Chemistry" and Section B "Mathematical Sciences."

DIMENSIONS/NBS

This monthly magazine is published to inform scientists, engineers, businessmen, industry, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on the work at NBS. The magazine highlights and reviews such issues as energy research, fire protection, building technology, metric conversion, pollution abatement, health and safety, and consumer product performance. In addition, it reports the results of Bureau programs in measurement standards and techniques, properties of matter and materials, engineering standards and services, instrumentation, and automatic data processing.

Annual subscription: Domestic, $12.50; Foreign $15.65.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a world-wide program coordinated by NBS. Program under authority of National Standard Data Act (Public Law 90-396).

NOTE: At present the principal publication outlet for these data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St. N.W., Wash., D.C. 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The purpose of the standards is to establish nationally recognized requirements for products, and to provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, D.C. 20402.*

Order **following** *NBS publications—NBSIR's and FIPS from the National Technical Information Services, Springfield, Va. 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services (Springfield, Va. 22161) in paper copy or microfiche form.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

The following current-awareness and literature-survey bibliographies are issued periodically by the Bureau:

Cryogenic Data Center Current Awareness Service. A literature survey issued biweekly. Annual subscription: Domestic, $25.00; Foreign, $30.00.

Liquified Natural Gas. A literature survey issued quarterly. Annual subscription: $20.00.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $30.00. Send subscription orders and remittances for the preceding bibliographic services to National Bureau of Standards, Cryogenic Data Center (275.02) Boulder, Colorado 80302.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

U.S.MAIL

SPECIAL FOURTH-CLASS RATE
BOOK