DATA INTEGRITY MANAGEMENT IN DATA BASE SYSTEMS

by

Michael Leslie Cunningham

August 1975 TR-51

This report constituted the author's Master's thesis in Computer Sciences at The University of Texas at Austin and was supported in part by Air Force contract #F33600-74-C-0314.

DEPARTMENT OF COMPUTER SCIENCES
THE UNIVERSITY OF TEXAS AT AUSTIN

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION AND OVERVIEW

1.1 Introduction

Data base management is one of the most rapidly expanding fields in computer systems. Data base management systems provide a means of mapping a logical structure onto a collection of related data. This logical structure allows rapid, convenient access and modification of the data. The integrity of this data is a matter of growing concern to the users of data base management systems.

To date, the emphasis in data base management systems has been focused on protection and privacy requirements. Protection and privacy are comprehensively presented by Lefkovitz [1] and Rubinoff [2]. No correspondingly thorough investigations exist into means of improving system reliability and thus data integrity. Chandy, Browne, et al [3] noted this lack.

> . . . the reliability of these [data base management] systems and their ability to constantly yield accurate results is a significant problem area. Much effort has gone into the study of privacy of computer-based information and protection of information systems from unauthorized access. Much less attention has yet been paid to increasing their reliability.

The goal of improving system reliability can be termed data integrity management. Data base errors are the obvious causes of

loss of system integrity. The primary purpose of data integrity management is the detection and correction of data base errors.

There have been two major developments in data integrity management. The first, redundant encoding, was derived from techniques developed in communication theory [4] as a result of work by many, including Hamming [5]. Such integrity techniques required redundant parity strings on each row and column of a block of data. Variations of these techniques were presented by Dalton [6] as software routines for error detection and correction. The second development in data integrity management concerns the use of Rollback and Recovery (R/R) strategies [3, 7]. R/R incorporates the making of periodic copies [checkpoints] of data bases and the recording of all transactions that modify the data bases [audit trail]. Upon error detection, the most recent archival copy of the data base replaces the existing copy of the data base and the transactions accumulated on the audit trail are then reapplied. A limiting assumption of R/R processing is that the data base was error-free at the most recent checkpoint.

Data base errors can be classified as being of two general types, hard-flaw errors and soft-flaw errors. Hard-flaw errors are attributable to failure of storage media or to transmission errors caused by noise over Input/Output channels. Soft-flaw errors are data base errors introduced by errors (bugs) in the data base software system.

Parity techniques are capable of detecting and correcting the majority of hard-flaw errors. Those hard-flaw errors which parity techniques detect but cannot correct can invoke a R/R instance. The detection and correction of soft-flaw errors is the unanswered problem. R/R depends on error detection from another source. Parity techniques are not able to detect bad source data; consequently, they are generally incapable of detecting soft-flaw errors.

This thesis analyzes alternative techniques for soft-flaw detection and correction. Furthermore, it presents these techniques in conjunction with parity and R/R techniques as the basis for a complete Data Integrity Management (DIM) System.

1.2 Overview of Topics To Be Presented

This thesis specifies methods by which combinatorial or differential applications of error detection/correction techniques may be analyzed for resource cost and for various probabilities associated with error detection and correction. The set of error detection/correction procedures comprise a Data Integrity Management (DIM) System.

Chapter II defines and discusses terms and concepts that are basic to a presentation concerning data base systems and error detection/correction procedures.

Chapter III develops the mathematical techniques and the model used for analysis of a DIM system. The model, called an Error Flow

Graph, is developed for a hypothetical system with a minimal, yet representative, set of error detection/correction procedures.

Chapter IV presents the error detection/correction techniques developed for the Advanced Logistics System (ALS) of the Air Force Logistics Command. Sections included will present: i) a simplified presentation of ZODIAC, the data base management system used in ALS, ii) a description of structure validation techniques with resource cost analysis for selected techniques, and iii) a description of content validation techniques with their resource cost analysis.

Chapter V develops the model of DIM processing resulting from the use of a specified subset of the ALS error detection/correction techniques given in Chapter IV. The problems encountered in modelling an actual system are presented along with a discussion of the utility of the model.

Chapter VI summarizes the major points of the thesis, including limiting assumptions of the DIM model, problems of quantification, and utility of the model as a general tool.

CHAPTER II

TERMINOLOGY AND CONCEPTS

2.1 Data Bases and Integrity Management

A data base can be considered as the concatenation of a structure and a content (or value) set. Integrity management in a data base must therefore concern itself with both factors, content and structure. The two are not disjoint; it is frequently the case that the structural definition of a data base is stored as part of the data base itself or as a separate data base whose structure is defined globally. Any data base system contains redundant information which is implicitly or explicitly built into the structure and content sets; integrity management consists of the utilization of this redundancy which exists between and within the structure and content of the data base.

2.2 Structure vs. Content Validation

It has already been noted that the structure and content sets of a data base are non-disjoint; this property implies an implicit degree of redundancy. It can be easily seen that a given technique/procedure for the validation of content of a data base has a direct impact on the validity of structure of the data base through checks of the content of structure definitions kept in the data base.

Likewise, although not to the same extent, techniques for structure validation (since they typically utilize content values) also effect a degree of content validation.

Structure validation is imperative for the integrity of a low reliability software system. Structure validation procedures are traps for soft-flaw errors. An invalid or incorrect address used to access other information allows the possibility of system failure or error propagation within the data base. Addresses or indices are particularly prone to start a "domino effect" which could lead to a halt in processing. However, it is again noted that addresses or indices belong to both sets, structure and content. Further, incorrect content allows the possibility of many errors such as incorrect reports, mishandled inventory, or error propagation (if indices or if other values in the data base are dependent on--computed from--the erroneous value).

2.3 General Technique for Redundancy Utilization

It is generally the case that the redundancy built into data bases is either implicit and must be abstracted (i.e., table entries) or coded and must be analyzed (i.e., checksums). The requirement for integrity is usually that agreement exist between redundantly recorded information. This implies that the implicitly coded or dually recorded redundant information must be directly compared. Thus, the basic process of integrity validation, whether it be for structure or content, can be generalized as a three step procedure:

1. To identify the redundancy to be used for comparison,
2. To determine the functions f and g which will map the two representations into comparable forms,
3. To compare the two representations yielded by step (2) for equality.

The most trivial of examples for this functional mapping technique is to use the identity operator for both function f and g. In this case, the redundancy in the data base is in the form of duplication of values, although it is seldom the case that this form of redundancy is either convenient or economic. The redundancy is in this case provided explicitly and for integrity. A further problem resulting from this form of redundancy testing occurs once an unequal comparison takes place--which value is in error or are they both invalid?

An example for the validation of structure is the use of an Index Verifier. This procedure verifies that each index in a system with an index directory points to a valid record entry. This is accomplished by comparing the value for the indexed data element kept in the index directory with the value for the data element stored in the data record. The mapping functions, f and g, in each case consist of extraction of the comparison element from each of its two environments. Note that this is a use of content to validate structure, and that in this case the redundancy is present because of the need for efficient access rather than for the needs of integrity management.

The use of a functional description of validation procedures is utilized in Chapter IV for the techniques presented for the Advanced Logistics System.

2.4 Continuous, Periodic, and Sporadic Validation Techniques

Any structure or content validation procedure can be further identified as being either a continuous technique, a periodic technique, or a sporadic technique.

Parity checking is an example of a continuous error detection procedure since parity checking can be carried out all the time a system is available. Any technique which performs redundancy comparison at fixed intervals is termed a periodic technique; an example of a periodic technique would be the comparison of dual copies of an index directory once every 24 hours. Sporadic validation techniques are similar to periodic techniques with the exception that there is no fixed interval between application. An example would be an operator activated validation procedure.

The concept of continuous and periodic validation procedures is of major importance in the development of the general DIM model presented in Chapter 3. Sporadic techniques cause special complications to the DIM model and are postponed until the presentation of the Advanced Logistics System DIM model of Chapter V.

2.5 Error Detection vs. Error Correction vs. Error Recovery

The problems of error detection, though considerable, are minimal in comparison to the correction problem which arises upon

detection of an error. Detection is typically characterized by a low data redundancy overhead but a high processing overhead. Correction, on the other hand, has a higher data redundancy cost, but typically has a small processing cost since it should occur infrequently.

The occurrence of correction processing is fixed. It must occur whenever an error is detected (assuming all errors detected are valued as being equally critical to continued processing). However, error detection can be introduced into a data base system in several forms. It can be incorporated into the normal processing by having the data handling routines verify as they process. It can be made a continuous process in the sense of running as one of the background tasks in the total system (perhaps on a selective basis). It can be periodic.

In the context of content errors, it has been established that combined error detection and correction is possible with certain combinations of multiple parities and checksums. Utilizing a block of 60 words (60 x 60 bit matrix), Dalton [6] identified techniques that were capable of detecting up to three erroneous bits and correcting at the moment of detection all 1 and 2 bit and the majority of 3 bit errors. Error correction here becomes a matter of content reconstruction. In the event of non-correctable errors, then recourse would typically be made to the traditional technique for recovery--periodic replication (checkpoint) with logged updating (audit trail) as part of a general purpose R/R system.

2.6 R/R and DIM

R/R is a data base reconstruction process as opposed to an error correction process. Previous discussion has defined a DIM system as being a set of error detection/correction procedures. However, any collection of techniques incorporated as part of a Data Integrity Management System must still necessarily include a R/R function. The techniques covered in this report are principally error detectors though some do have limited capability for error correction. At the level of current knowledge and technology it is impossible to conceive of an operational DIM system that does not incorporate a function at least analogous to R/R for reconstruction of a seriously flawed data base.

CHAPTER III

ERROR FLOW GRAPHS

3.1 Introduction

The primary goal of this paper is the formulation of a general model which will adequately simulate the processing patterns of a given DIM system. The remainder of this chapter is dedicated to the identification of an appropriate method(s) of mathematical analysis which are sufficient for the determination of varied forms of error probability (detection, correction) and the system resource costs associated with any subset of identified validation techniques. The methods identified produce as metrics the probabilities that a data base is in any one of three primary states (all of which are defined in detail in section 3.2): 1) error free, 2) contains an error condition which will be detected by a periodic, continuous, or sporadic (sampling) ED/C procedure, or 3) is in an error correction mode. These metrics are utilized in the generation of other error attributes of the data base and in the determination of resource costs of the ED/C procedures tested.

One means of characterizing a data base, other than listing typical data base characteristics such as size, structure, etc., is to enumerate the validation procedures applied against it, or equivalently, to identify the range of error types that can occur, be detected,

and/or be corrected. In other words, one means of identifying a data base is to specify its "error environment" and the set of validation procedures selected to operate within that error environment.

An error environment can be specified and defined by the possible error states of the data base system. The minimal set of states necessary are: 1) an error-free state, 2) an invalid state representing an error which will be detected by one or more detection mechanisms, 3) an error correction state representing the application of an automated or manual error correction procedure (not including R/R), and 4) an uncorrectable error state representing errors which are beyond the range of automated and manual correction procedures, thus implying interaction with a R/R system. The probabilities of these states are clearly interdependent and also dependent on the ED/C techniques employed. These relations suggest the use of a directed graph for the representation of the error state space (environment); this graph will be termed an ERROR FLOW GRAPH, or as it will be referred to in the remainder of this report, an EFG.

The following sub-sections will be devoted to an EFG's notation, adaptation from data base to data base, paramaterization, and analysis. Further sections will describe the techniques used to incorporate resource analysis in EFG's and the resultant increase in paramaterization, and the interrelation of EFG analysis with R/R analysis.

3.2 EFG Notation

An EFG will be represented by a finite number of nodes interconnected by unidirectional edges. An individual node of an EFG will be represented as a circle containing the node (state) number plus an abbreviated error-state description as shown in Figure 1.

Figure 1 - An EFG State

Edges between nodes will be labeled with appropriate branching probabilities p_{ij} where i is the source state and j is the target state as shown in Figure 2.

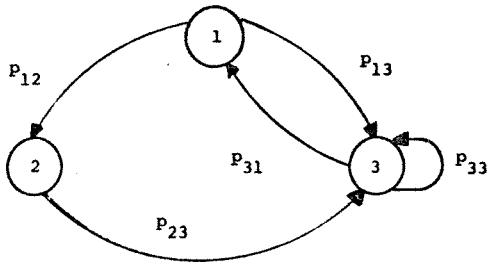
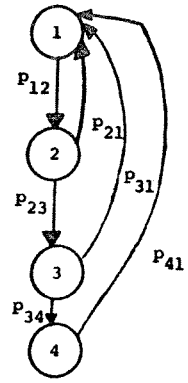


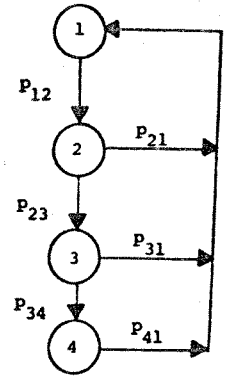
Figure 2 - Directed Edges and Probabilities

In certain situations where the cause for transition from state to state is felt to be unclear from the state descriptions an additional textual comment will be added to the edge along with the branching probability.

To avoid edge cluttering around any state that has a large number of transition paths into it, a convention is adopted that compacts a number of edges into a single unlabeled edge with multiple labeled entry edges. This graphic convention is best illustrated through the use of an example which is given in Figure 3. As can be seen in 3b the compacted edge is assigned no transition probability; instead the individual entry edges are labeled with their respective branching probabilities.



a) Edge-cluttering on State 1



b) Edge-compaction on State 1

Figure 3 - Edge Compaction

3.3 Typical States of an EFG

There exists a set of error states which are basic to even the most simplistic representation of an error environment. Definition of these states is presented and analyzed below.

The ERROR FREE State - represents a data base free of detected errors, or in special cases (see sections 3.6 and 3.13) a data base containing errors which have been detected, isolated, and tagged, but which have had corrective action postponed. This state is always numbered one (1).

This definition of the ERROR FREE state requires further explanation before proceeding.

1. It may contain errors not detectable by the set of ED/C techniques employed.
2. It may contain a set of detected, uncorrected but tagged errors which are awaiting the invocation of a rollback/restart/recovery instance, or which are regarded as insignificant. The stipulation that these errors be tagged is necessary in order to eliminate the possibility of a loop in error detection processing.
3. 1 and 2 represent identifiable system states, but the analysis which follows computes state probabilities by inclusion of these system states

as part of the ERROR FREE state. This compositing of states is utilized as a convenience to analytic development and should not be construed to imply that compositing precludes the computation of their probabilities.

The DETECT type State - represents the detection of a given type of error, e.g., Index error, BRT error, content error, etc. m is a unique element of the set {2,3,...,n} where n is the total number of states represented in the EFG.

The AUTO CORRECT State - represents the error correction activity of a given validation procedure; it is not typed since every "AUTO CORRECT" state is uniquely associated with a "DETECT type" state. m is defined the same as for the "DETECT type" state.

The MANUAL CORRECT State - represents the invocation of manual correction procedures (i.e., STRING UPDATE). m is defined the same as for the "DETECT type" state.

The NO CORRECT State - represents errors which could not be corrected by either automated or manual procedures;

implies the eventual (allowing the possibility of error collection) necessity of R/R application at or before the next checkpoint. m is defined the same as for the "DETECT type" state.

The few extensions to these basic state definitions will be identified and analyzed as they are introduced in the following sections.

3.4 Role of ED/C Techniques in EFG Construction

The opening discussion of this chapter introduced the concept of characterizing a data base by its error environment and the ED/C techniques applied within that environment. Including the knowledge that a validation technique (especially one incorporating corrective capability) has a marked effect on the behavior of the error environment, it becomes intuitively reasonable that the techniques represented should have a direct relation to the final structure of an EFG. This proves to be the case.

There are two basic types of transitions that can be made from the ERROR FREE state in a discrete time period: 1) no errors are detected and transition is made back to the ERROR FREE state, and 2) a specific type of error is detected forcing transition to a "DETECT type" state.

Focusing on the latter form of transition, it should be clear that branching to a "DETECT type" state is the result of application of

some form of validation technique. Reasoning thus, it is obvious that a direct relationship between the active validation procedures and the branching paths available from the "ERROR FREE" state exists. Therefore we can establish the initial portion of any EFG as being similar in form to that given in Figure 4.

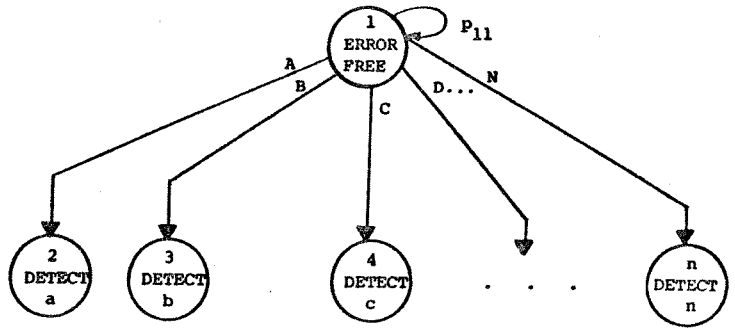


Figure 4

Branching From State-1- As a Result of Validation Procedures

In Figure 4, contrary to notational definition given in section 3.2 branching probabilities were not assigned to the edges between state 1 and the set of "DETECT" states. This omission is intentional and introduces the topic of the following section.

3.5 Conversion of Continuous and Periodic Paths to Discrete Paths

Any validation procedures that have paths represented out of the "ERROR FREE" state can be classified in one of three general

descriptive categories: 1) Continuous Application, 2) Sporadic Application (Sampling Techniques), or 3) Periodic Application. A technique that is continuously applied can for the purposes of its path be characterized by detected errors/unit time, while a technique that is periodically applied can be typified by its period of application. Sampling techniques with sporadic application maintain a middle ground between continuous and periodic techniques and thus present a special problem to analysis. A completed discussion of their quantification as discrete paths and the attendant assumptions and problems will be postponed to section 5.3.2 when explicit sampling techniques of the ALS-EFG are presented. The remainder of this section will deal exclusively with conversion of continuous and periodic techniques.

It should be clear that the branching probability of any validation procedure out of the ERROR FREE state is directly proportional to the probability of that type error <u>given</u> an error has been detected. The first step necessary to obtain these probabilities is to obtain a common metric for both the continuous and periodic validation techniques. The natural common metric to strive for is errors/unit time. Thus, a means must be found to convert a period of application of a periodic technique to a metric of errors/unit time. This conversion is accomplished by the following simple convention.

> If a is the period of application of a periodic technique A and c is the mean number of errors detected on a random application of A, then technique A can be characterized by an error rate of c/a.

The sole remaining task is selection of a standard measure of unit time for all error rates. Since the measure of unit time selected for this will also be used as the basic time quantum between state transitions, the selection of this time unit warrants special consideration in section 3.6.

Given that all the techniques are quantified by the metric, detected errors/unit time, the sum of their values yields net detected errors/unit time. Thus, the probability of any particular type of error given an error is detected can be expressed as

$$\frac{\text{detected errors/unit time}}{\text{net detected errors/unit time}} = \frac{\text{detected errors}}{\text{net detected errors.}}$$

If the validation techniques represented <u>all</u> paths out of the "ERROR FREE" state, then no further conversion would be necessary. However, there exists the path from state 1 back to itself. If p_{11} is the value of this path (the probability of 0 errors/unit time), then all the probabilities figured above are normalized by a factor of $(1-p_{11})$.

The above conversion procedure is demonstrated in an item by item fashion in the example given below.

<u>Example 1</u>

i) One periodic event A of period a sec.

ii) One continuous event B with detected error rate of b errors/sec.

iii) Identify C as the mean number of errors detected by a random application of technique A.

iv) Periodic technique A can then be characterized by the error rate C errors/a sec.

v) The net error rate of A and B is $\frac{ab + c}{a}$ detected errors per sec.

vi) Probability of an "A" error given an error is detected is $\frac{c}{ab + c}$.

vii) Probability of a "B" error given an error occurs is $\frac{ab}{ab + c}$.

viii) If the probability of no errors occurring over a given unit of time is p_{11}, then:

Branching Probability for A = $(1 - p_{11})(\frac{c}{ab + c})$

Branching Probability for B = $(1 - p_{11})(\frac{ab}{ab + c})$

3.6 Discrete Transition Quantum

A discrete transition quantum, Q, is defined as that period of time after entering a state that a "decision" must be made as to which path will be taken out of the state. A number of points are implicit in this definition--a) The transition quantum is inviolate, i.e., a transition must be made, b) a state may transition to itself, and c) the likelihood of transitioning to any given successor state is represented by the probability assigned to the connecting edge.

On a conceptual basis the time quantum represents the resolution of measurement of the state of the system. Actual system

transitions may occur at a rate more rapid than the time quantum; this will, in fact, typically be the case with respect to the application of continuous procedures. The EFG is then a sampling model, but no error is introduced so long as the system has long-term, equilibrium behavior. Actual transitions may also occur (particularly for periodic techniques) at a rate much slower than the quantum interval. This introduces negligible error so long as Q << Tp.

In order to establish a representative time quantum for an EFG, we note two bounding factors.

1. The smaller the chosen quantum is, the greater the probability becomes of transitioning from state 1 back to itself. This relation is demonstrated by Example 2.

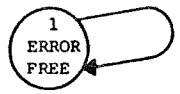
Example 2

Mean detected error rate = 1 error/day

a) Q = 1 hour

p_{11} = 23/24 = .958

b) Q = 1 min.



p_{11} = 1439/1440 = .999

Thus, as the transition quantum, Q, is reduced, p_{11} will tend toward one (1). As p_{11} tends toward 1 the remaining paths (error paths) out of state 1 must necessarily tend toward zero. Working with very small error branching probabilities

seriously impacts both the ease and accuracy of EFG analysis which utilizes solution of a system of linear equations with the error branching probabilities as the equations' coefficients (see section 3.10).

2. The mean number of detected errors per transition quantum cannot exceed 0.3. Since an EFG can be in only one state at a time it should be clear that an EFG can handle only one error at a time. An immediate consequence of this is that the mean number of detected errors possible in one transition quantum must be less than one. However, as the structure of an EFG is completed in future sections it will be found that on the average three transitions are made for any detected error. Therefore it is necessary to allow for three transition quantums between detected errors; as a result, the initially stated constraint of no more than 0.3 errors per transition quantum is imposed.

The net effect of these factors is to impel the selection of the largest available time unit (factor 1) such that factor 2 is not violated. Thus, the selection of the transition quantum can be reduced to the following problem where the set $\{x_i\}$ is the set of possible time units.

Q = transition quantum = max $\{x_i |$ mean number of detected errors per $x_i \leq 0.3\}$

Assume that through data collection on a data base of a given system it was found that errors were detected at a mean rate of one per minute. Since a minute cannot be used as the quantum, attention is moved to seconds which yield a rate of .017 detected errors per second. A second thus becomes a reasonable selection as the transition quantum. Even if a data base of the system demonstrated a detected error rate of 18 errors/min (highly improbable for any system), a one second transition quantum would still meet the constraints. As a result a one second transition quantum will be assumed as standard throughout the remainder of this report for convenience in notation, and in any case, should not be less than this.

As a sidelight to the above, if the net detected error rate is given with seconds as the time basis, then p_{11} can be determined. If r is the net detected error rate, then $p_{11} = 1 - r$. As an example, if the net detected error rate is determined to be .15 errors/sec., than p_{11} would be valued at .85. The branching probabilities derived for the error paths out of state 1 would be normalized by a factor of $1 - p_{11} = r = .15$.

3.7 A Complete Sample EFG

Figure 5 represents the simplest complete EFG for a data base which has one continuous validation technique A, and one periodic validation technique B, incorporated. The branching probabilities shown out of state 1 are shown as developed in previous sections. A complete EFG for a system containing multiple continuous and/or

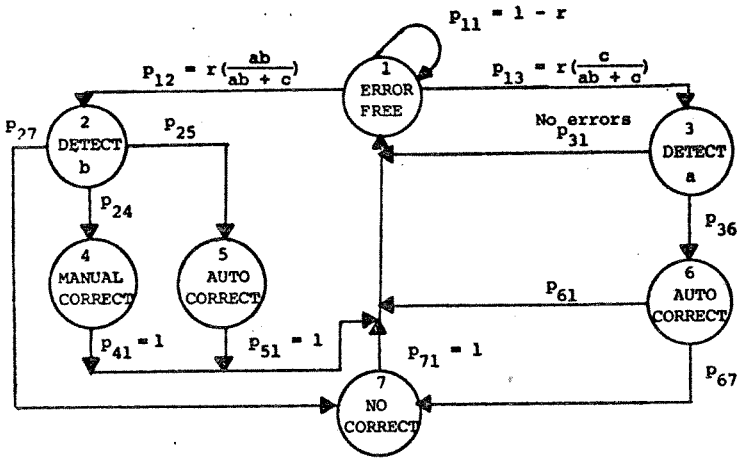


Figure 5 - A Complete EFG

periodic techniques differs only in that extra paths out of state 1 would exist. The remaining points to be discussed concern the varied paths out of either of the "DETECT type" states and their implications.

Having reached state 3, "DETECT a", there are two choices upon transition. 1) A return can be made to state 1 reflecting zero errors. This probability will be zero since transitions are made to DETECT a and AUTO CORRECT only for detected errors since periodic event A was converted to a quasi-continuous technique typified by a fixed error rate. Thus, the zero error path will be eliminated from all future EFGs with periodic ED/C procedures. 2) A transition is made to A's corresponding "AUTO CORRECT" state. If all the errors detected are correctable by A, then transition is made back to state 1; however, if

detected errors exist that cannot be corrected, transition is made to the "NO CORRECT" state. It is again noted that the EFG portrays the detection of errors over mutually exclusive time intervals; this feature guarantees that any transition into the "NO CORRECT" state requires tagging of only a single error. Thus, periodic techniques, which typically might at the conclusion of their processing have isolated more than one uncorrectable error, are represented one error at a time in their respective error states of the EFG. This approximation will cause no error if the system has a long-term equilibrium behavior. A further benefit of this EFG trait is that the determination of the branching probability becomes a straightforward problem of analyzing the probability an error is uncorrectable given that it has been detected by a given validation procedure.

If state 2--"DETECT b"--is entered, there are three possible successor states which are typical for continuous procedures: a) AUTO CORRECT--p_{25} reflects the percentage of "b" errors that procedure B can correct; transition out of state 5 is forced (i.e., $p_{51} = 1$) to state 1, b) MANUAL CORRECT--analagous in discussion to -a- with the exception that any manual procedures used are not explicitly related to B, and c) NO CORRECT--p_{27} reflects the percentage of errors of type "b" that were neither corrected by automated nor manual procedures.

If the NO CORRECT state is entered, it has a singular transition path back to the ERROR FREE state. Transitioning from an uncorrected error state to an "error free" state would appear to be

inconsistent, but the possibility of having uncorrected, detected errors present in state 1 is accounted for in the definition of an "ERROR FREE" state. The rationale behind this definition will be postponed to section 3.13, when the interaction of the EFG and R/R is presented.

3.8 The EFG as a Discrete Markov Process

A Markov process can be described as a set of mutually exclusive, collectively exhaustive states $I = \{s_1, s_2, \ldots, s_m\}$ which satisfy the Markovian criteria:

1. The present state of the system specifies all historical information relevant to the future behavior of the process, i.e., the successor state is dependent only on the current state.
2. The system cannot be in two states at once.
3. The set I includes states representing all possible paths of the system.

A discrete Markov process has the further property that all transition paths out of a state are weighted by branching probabilities, p_{ij}'s, that satisfy the relation $\sum_i p_{ij} = 1$.

Consideration of the discussion provided in the previous six sections for the development of the EFG, yields the conclusion that any properly constructed EFG will be a discrete Markov process. The relatively straightforward mathematical techniques for Markovian

analysis thus become the mathematical techniques presented in section 3.10 for the analysis of an EFG.

3.9 The Single Parameterized EFG

If the universal set of applicable validation techniques for a given system is: 1) Periodic technique A, and 2) Continuous technique B, then the EFG of any system data base will have structure identical to that shown in Figure 6. The fact that some data bases will use only one or neither of the available techniques can be adjusted to this EFG by setting either or both the period of A and the detected error rate of B to zero. Generalizing from this it can be stated that an EFG which is structured to represent the universal set of system validation procedures is capable of being equally representative of any of the system's data bases.

Referring to Figure 6, if this EFG is capable of accurately representing any data base of the sample system, then the branching probabilities of the EFG must reflect the changes in data base attributes, since all else is static. This is a reasonable development since the activity of either of the validation techniques is dependent on a number of system and data base attributes which affect the error environment. Examples of attributes which would impact the error environment are transaction rate, data volume, and system reliability.

The above discussion implies the parameterization of the single EFG with primary system and data base attributes in order to

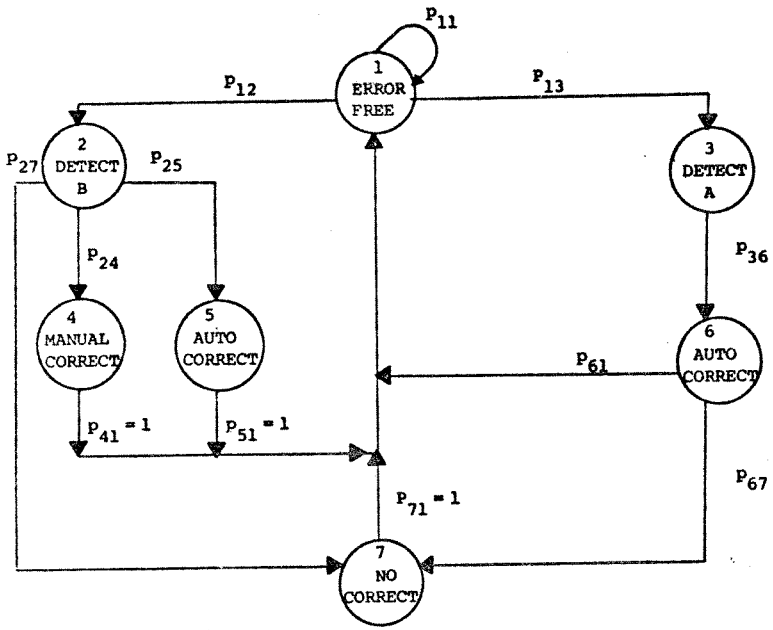


Figure 6

A General EFG for A System With One Periodic
And One Continuous ED/C Technique

generate "data base specific" branching probabilities. The parameterized EFG can be considered as a function which maps the primary attributes of the system and a data base onto the set (matrix) of branching probabilities.

The necessary parameters for the EFG will include:

1. Period of application for all periodic validation procedures of the universal set (set to zero if the respective procedure is unused for a given data base).
2. Percentage application of validation procedures over the data base (where relevant)--e.g., for a continuous content validation procedure only a portion of the data base may contain the necessary parities or checksums used for validation.
3. Transaction rate upon the data base. This could be broken down into two separate parameters:
   a) Update rate--rate of transactions which modify the contents of the data base.
   b) Accessing rate (read or write)
4. Data base size.
5. Flaw rate for individual memory hierarchies.
6. Incidence of transmission errors.
7. Further parameters identified as having an effect on particular validation procedures.

A sample parameterization of the EFG of Figure 6 could be given as:

$$EFG\ (P_A,\ \%_B,\ TR,\ S,\ FR_D,\ TE)$$

where P_A = period of technique A

$\%_B$ = % application of technqiue B

TR = transaction rate S = data base size

FR_D = disk flaw rate TE = transmission error rate

An example of the quantification of branching probabilities from EFG parameterization will be presented for the Advanced Logistics System EFG in section 5.6.

3.10 Basic Analysis of the EFG

It was established in section 3.8 that an EFG is a discrete Markov process. Drake [8] provides a comprehensive mathematical development of the analysis techniques used for Markov processes. The following material will present a simplified development of Markovian analysis using the branching probabilities of the EFG in Figure 6. It is assumed that the values of the branching probabilities have been determined.

Define the one step transition matrix of the EFG as:

$$\mathbb{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{17} \\ p_{21} & p_{22} & & \\ p_{31} & & & \\ \vdots & & & \vdots \\ p_{71} & \cdots & & p_{77} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{24} & p_{25} & 0 & p_{27} \\ 0 & 0 & 0 & 0 & 0 & p_{36} & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_{61} & 0 & 0 & 0 & 0 & 0 & p_{67} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let $p_{ij}(n)$ be the probability that the EFG is in state j after n transitions, given that the EFG is currently in state i. Define the n-step transition matrix of the EFG as:

$$\mathbb{P}(n) = \begin{bmatrix} p_{11}(n) & p_{12}(n) & \cdots & p_{17}(n) \\ \vdots & & & \vdots \\ p_{71}(n) & \cdots & & p_{77}(n) \end{bmatrix}$$

Theorem 1: $\mathbb{P}(n) = \mathbb{P} \cdot \mathbb{P}(n-1) = \mathbb{P}^n = \mathbb{P}(n-1) \cdot \mathbb{P}$

Let $p_i(n)$ be the probability that the EFG is in state i after n transitions. Define the state vector as:

$$\underline{P}(n) = [p_1(n) p_2(n) \cdots p_7(n)]$$

Theorem 2: $\underline{P}(n) = \underline{P}(0) \cdot \mathbb{P}(n) = \underline{P}(1) \cdot \mathbb{P}(n-1) = \cdots$

$$= \underline{P}(j)\mathbb{P}(n-j)$$

Using Theorem 2 and letting $n \to \infty$, $\underline{P}(\infty)$, the steady state (or equilibrium) probabilities, are determined as:

$\underline{P}(\infty) = \underline{P}(\infty) \cdot \mathbb{P}$ or

$\underline{0} = \underline{P}(\infty) \cdot (\mathbb{P} - I)$, where I = identity matrix

Note that:

$$\mathbf{P} - I = \begin{bmatrix} p_{11}-1 & p_{12} & p_{13} & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & p_{24} & p_{25} & 0 & p_{27} \\ 0 & 0 & -1 & 0 & 0 & p_{36} & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ p_{61} & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

where the diagonal elements are equal to negative the probability of leaving state i

Converting from matrix notation to a system of linear equations yields the balance equations which express the equilibrium concept that the probability of entering a state is equal to the probability of leaving the state. Thus, letting $P_i = P_i(\infty)$, we have

	Balance Equation for
$0 = -(1 - p_{11})P_1 + P_4 + P_5 + p_{61}P_6 + P_7$	state 1
$0 = p_{12}P_1 - P_2$	state 2
$0 = p_{13}P_1 - P_3$	state 3
$0 = p_{24}P_2 - P_4$	state 4
$0 = p_{25}P_2 - P_5$	state 5
$0 = p_{36}P_3 - P_6$	state 6
$0 = p_{27}P_2 - P_7$	state 7

However, $\mathbf{P} - I$ is singular since one of the equations is redundant. Consequently one of the balance equations is replaced by the normalizing equation $\sum_i P_i = 1$, i.e., $1 = P_1 + P_2 + P_3 \ldots + P_7$.

The resulting system of 7 equations will yield a unique solution for the steady state probabilities, P_i. Since the P_i express the probability of being in a given state at a random time, they constitute the primary result of EFG error analysis and are also the basis for other modes of analysis.

In order to simplify the presentation given, it should be noted that the balance equations can be easily generated upon a visual inspection of the EFG. As an example take state 1 of Figure 6. Remembering that a balance equation is an expression of the equilibrium relation that the probability of entering a state is equal to the probability of leaving a state, the following is obtained for state 1.

Probability of entering state 1 $\overbrace{(p_{12} + p_{13})P_1}$ = Probability of leaving state 1 $\overbrace{p_{41}P_4 + p_{51}P_5 + p_{61}P_6 + p_{71}P_7}$

or

$(1 - p_{11})P_1 = P_4 + P_5 + p_{61}P_6 + P_7$

$0 = -(1 - p_{11})P_1 + P_4 + P_5 + p_{61}P_6 + P_7$

3.11 Obtainable Statistics From An EFG

Any function of the system state probabilities can be obtained. The following list describes statistics that are obtainable either

from the derived p_{ij}'s or the steady state probabilities P_i obtained by solution of balance equations. The list is ordered in approximate order of statistical value as an analytic aid in performance evaluation or decision making.

1. Detected, uncorrected error rate - Considering the EFG over an infinite number of transitions (seconds), and noting that state 7 is only entered with single errors, the detected, uncorrected error rate can be expressed by

$$\lim_{n \to \infty} \frac{n \cdot P_7 \text{ errors}}{n \text{ seconds}} = P_7 \text{ errors/second.}$$

2. Probability of error detection - There are no direct statistics yielded by the EFG for a simple formulation of error detection probability; it appears the best that can be done is to establish a reasonable lower bound on the probability. The establishment of this lower bound is achieved by segmenting the data base into the logical components on which each individual validation procedure acts. Within each component the probability of error detection by the relevant validation procedure is determined. Given these probabilities along with the fractional portion of the total data base volume that the particular component represents, a formulation for a lower bound can be given as

$$\text{LB of Probability of Error Detection} = \sum_{\text{components}} \left(\frac{\text{component volume}}{\text{data base volume}}\right) \times \text{relevant validation procedures' detection probability}$$

Although this lower bound formulation fails to consider numerous complicating factors, it serves as a crude but reasonable approximation.

3. Probability that a detected error of any type will be corrected short of R/R - This probability can be expressed as the sum of the probabilities of moving from a "DETECT type" state to a "MANUAL or AUTO CORRECT" state weighted by the conditional probability that the detected error is of the given type. This weighted sum must also reflect any failures of "AUTO or MANUAL CORRECT" (i.e., transition to "NO CORRECT" state). Applying this to the example of Figure 6 yields:

$$\begin{aligned}\text{Probability} = {} & P(\text{A error} \mid \text{error}) * p_{36} \\ & + P(\text{B error} \mid \text{error}) * (p_{24} + p_{25}) \\ & - p_{67} * P(\text{A error} \mid \text{error}) * p_{36}\end{aligned}$$

4. Probability of error correction - expressed as the sum of the branching probabilities out of a "DETECT" state to its associated "CORRECT" states. For state 2 of Figure 6 this probability can be expressed as PROBABILITY = $p_{24} + p_{25}$.

5. Probability error is of a specific type given an error has been detected - for a detailed explanation of generation see section 3.5.

6. Mean detected error rate for all validation procedures, and mean detected error rate for specific validation procedures - for a detailed explanation of generation see section 3.5.

The preceding represent a set of measures which are deemed to have significant impact for analysis and decision making with respect to data base integrity management. It should be noted that no quantification of the probability of error occurrence of various types has been presented. Since validation procedures may be only partially applied across a data base and have less than 100% error detection ability, there exists no inherent means for quantification or reasonable bounding of these probabilities. It appears that the ability to derive these probabilities would presuppose in most cases a knowledge of their causes. However, identification of the causal factors would imply implementation of system corrections or error avoidance procedures while ruling against validation procedures as developed.

3.12 Resource Usage Analysis With the EFG

The previous sections have provided the capability for analyzing the error environment of data bases subjected to differential or combinatorial application of validation techniques. It only remains to extend the concept of an EFG to allow analysis of resource requirements. The ability to include resource analysis as

a part of the EFG implies a minimal adjustment of basic concepts along with minor additional mathematical manipulation.

3.12.1 State Assignment of Resource Costs

The expansion of the EFG to include resource usage will be accomplished by an extension to the state definitions. Every state which utilizes system resources beyond that level associated with normal (non-error) processing has an addition made to its definition which quantifies, or minimally, formalizes, the resource usage.

Returning to the sample EFG of Figure 6, repeated here, it will be assumed that the resource costs of validation techniques A and B for detection are D_A and D_B respectively. C_A and C_B are the respective automated correction resource costs, and M_B represents the resource cost of manual correction resulting from technique B.

The following paragraphs will present on a state-by-state basis the assignment of the resource costs to the states of the EFG of Figure 6 along with discussions of the reasoning employed. All figures are based on analysis of the EFG over a standard time frame of t_Q seconds. The basic technique employed will follow the general form:

t_Q = time frame (unit of seconds)

Q = transition quantum (unit of sec/EFG transition)

$t_r = \frac{t_Q}{Q}$ = number of EFG transitions made in time frame t_Q (unit of EFG transitions)

P_i = steady state probability of entering state i (unitless)

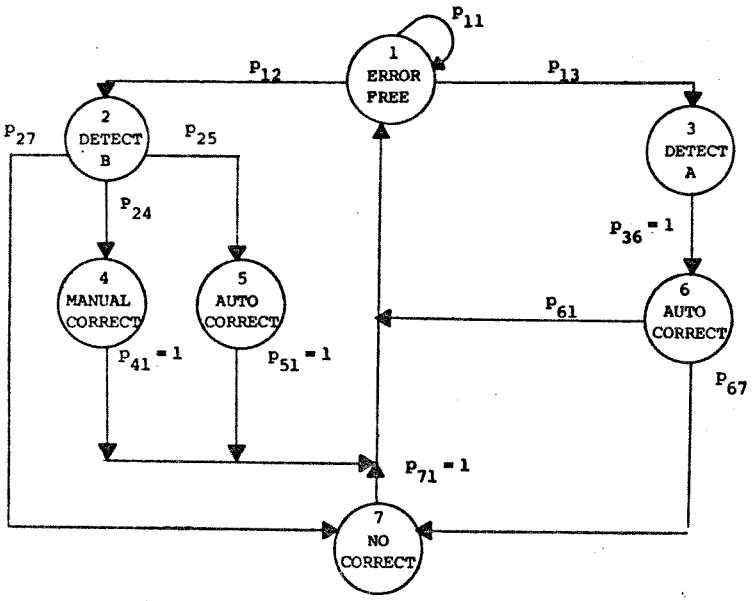


Figure 6

A General EFG for A System With One Periodic And One Continuous ED/C Technique

$P_i t_r$ = number of EFG transitions made into state i (unit of EFG-transitions

R_i = resource cost associated with an EFG transition into state i (units of cost/EFG transition)

$R_i P_i t_r$ = total resource cost associated with state i (unit of cost)

Analysis of the individual R_i's for each state follow.

1. ERROR FREE State - Zero resource cost. It would seem that all detection costs should be absorbed in the ERROR FREE state since transitions occur from state 1 only upon error detection. However, detection costs are assigned to the DETECT states in order to distinguish the resource costs contributed by the individual validation procedures.

2. DETECT B State - $\frac{D_B}{t_Q r_B}$ resource cost. Given that D_B is the total cost of application of continuous technique B over time frame t_Q (typically, t_Q = checkpoint interval), and that r_B is the equilibrium detected error rate, then

   $t_Q r_B$ = total number of B type errors detected in time frame t_Q

   $\frac{D_B}{t_Q r_B}$ = average resource cost of a single detected error which includes processing expended before detection

Since the ERROR FREE state was assigned no resource costs, that portion of validation processing which produces no error detection must be accounted for in the cost of the DETECT state; consequently, all resource usage of technique B is costed to the errors detected by B.

3. DETECT A State - $\frac{D_A}{C}$ resource cost. Given that D_A is the total cost of a single application of periodic technique A, which has period p_a, and that c is the mean number of errors detected by A on random application, the following results are obtained for time frame t_Q.

   $\frac{t_Q}{P_a}$ = mean number of applications of technique A

   $\frac{t_Q D_A}{P_a}$ = total resource costs of application

   $\frac{t_Q c}{P_a}$ = total number of A type errors detected

   $\frac{D_A}{C}$ = average resource cost of a single detected error

   As in (2) the total resource cost of technique A is costed solely to the errors detected by A.

4. AUTO CORRECT (B) - C_B resource cost. Since C_B is valued at the average resource cost for single error correction, the cost of error correction is simply C_B.

5. MANUAL CORRECT (B) - M_B resource cost. The same reasoning used in (4) is applied for Manual Correct. It should be noted that in actual cases only estimates of computer resources will be used. Resource costs of manual correction procedures are typically not amenable to accurate quantification due to inherent time delay and human cost factors.

6. AUTO CORRECT (A)- C_A resource cost. By reasoning comparable to that presented in (4), the cost will be C_A, the average resource cost for such corrections.

7. NO CORRECT - zero resource cost. Since the no-correct state's only processing function is to tag uncorrected, detected errors--a process which has no significant cost--it is assigned zero resource cost.

Having assigned the resource costs expended upon entering any state of the EFG, R_i, the total resource costs expended in validation processing are expressed as follows:

$P_i t_r$ = number of EFG transitions made into state 1

hence:

$t_r \sum_i P_i R_i$ = total resource costs expended in validation processing

Thus, for the sample EFG of Figure 6:

$$\text{total resource costs} = t_r(P_1 * 0 + P_2 * \frac{D_B}{t_Q r_B} + P_3 * \frac{D_A}{C} + P_4 * C_B + P_5 * M_B + P_6 * C_A + P_7 * 0)$$

simplifying,

$$\text{total resource costs} = t_r(P_2 * \frac{D_B}{t_Q r_B} + P_3 * \frac{D_A}{C} + P_4 * C_B + P_5 * M_B + P_6 * C_A)$$

The development presented for procedures A and B are applicable to any other periodic or continuous techniques. However,

a number of simplifying assumptions have been made in this presentation which would not necessarily be found in an actual systems' EFG. These assumptions are summarized as follows:

a. All resource costs are expressed by a single common metric. As will be found in the ALS-EFG presented in chapter 5, a single metric is not always available. However, this does not invalidate the resource usage analysis presented. Total resource costs can be collected separately for each metric and then consolidated into a single metric.

b. Continuous procedure B was presented using 100% application over the given data base. If the percentage application varies it will impact r_B, the equilibrium detected error rate of technique B. By computing a new value of r_B the effect of % of application on total resource costs can be allowed for.

c. Manual Correction procedures are assumed to have a fixed resource cost. As noted in the discussion for the MANUAL CORRECT State no accurate means for quantification or metric determination has been developed for manual correction due to the associated human cost factors.

In summary, this section presents a general technique for determination of resource usage of validation techniques which is expressed as the sum of the resource costs of entering each state

weighted by the number of transitions made into each state respectively.

3.12.2 Additional Parameterization of the EFG

Formalizations of resource costs of certain validation techniques can involve parameters not provided by the sample parameterization given in section 3.9. The parameterization of the EFG in such cases is extended to meet the needs of the resource cost formalizations. In other cases, as in the example given in the previous section, no extension of the input parameters is necessary.

3.12.3 Optimization Capability

If there is more than one combination of available validation procedures that can maintain a specified level of data base integrity then the technique presented for resource usage analysis of validation procedures serves as a tool for the determination of which combination yields the minimal resource cost.

3.13 The EFG and R/R

Since R/R is a data integrity measure, as are validation procedures, it is not surprising that there exists a relationship between the EFG and whatever mode of R/R is present in a given system. This relationship was mentioned in section 3.7 during the discussion of the single exit path from the "NO CORRECT" state to the "ERROR FREE" state. The following sections will develop the logic behind this relationship.

3.13.1 Interface of R/R and the EFG

The usual result of an uncorrected, detected error (given a high valuation on validity) is the invocation of some form of R/R or alternative to R/R. However, a R/R state has no place in the EFG as developed since it is not a validation technique; it is a reconstruction technique. Thus, while there does exist a concrete relation between the "NO CORRECT" state and R/R, it is not explicitly represented as a separate state in the EFG. The relation (interface) between the "NO CORRECT" state and R/R will be represented as depicted in Figure 7 with the "NO CORRECT" state surrounded by a dotted box representing the interface.

3.13.2 The Use of the R/R Interface For Decision Making (Memory)

A further, stronger reason exists for the separation of R/R and the EFG. It is desirable when analyzing a complete data integrity system--error detection/correction plus R/R--to allow for the possibility of accumulating more than one error before invoking R/R. This partially offsets the generally high resource usage of R/R. Without the capability to accumulate a number of errors, any analysis is forced to the worst case level of R/R application upon every uncorrected, detected error.

If the accumulation of errors is allowed, then some mode of memory must be instituted in order to account for the errors and as a basis for an eventual decision to invoke R/R. From the definition of a Markov process, this memory feature cannot be incorporated as part

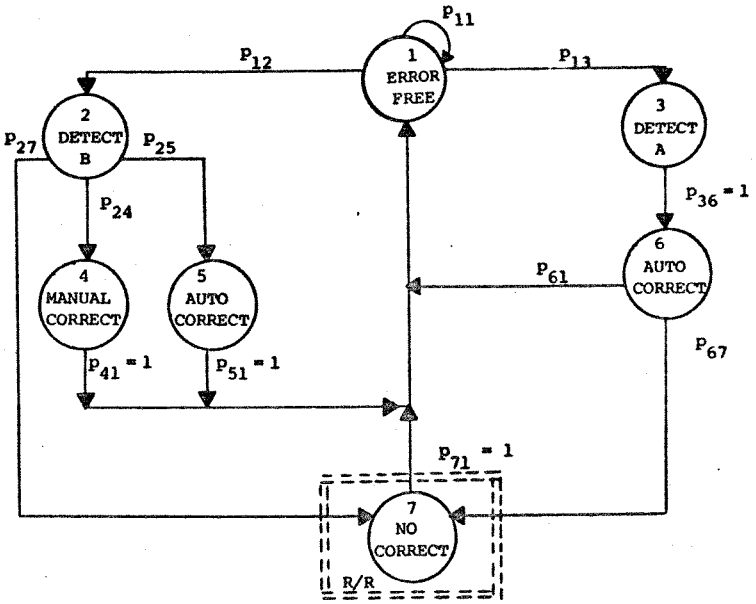


Figure 7 - Graphical Representation of the R/R, EFG Interface

of the EFG (thus the transition directly from state 7 to state 1); instead it must be incorporated as a function of some external mechanism. The logical choice to perform this composite memory/decision-making process is the R/R interface.

In conclusion, the lack of memory in an EFG serves as both a necessary and sufficient model of the real environment. Necessary because Markov models can have no memory; sufficient because a R/R utility would be responsible for such memory in an actual system.

CHAPTER IV

THE VALIDATION PROCEDURES APPLICABLE TO ALS

4.1 Introduction

Chapters IV and V develop an EFG which models a Data Integrity Management system for the Advanced Logistics System (ALS) of the U.S. Air Force Logistics Command. After describing ZODIAC--the data base management system of ALS--this chapter discusses validation techniques that can be incorporated into a data integrity management system for ALS.

The techniques are divided into two general categories, structure validation techniques and content validation techniques. Structure validation techniques check the integrity of data base tables which define the logical structure of the data records. Content validation techniques check the integrity of data base entries.

Analysis of error detection and error correction resource costs is included for three of the validation techniques. These techniques are Index Access Verifier, Block Reservation Table Verifier, and Algorithm HVD (a parity technique). Resource costs of these three are analyzed because

1. They are the techniques which will be incorporated into the EFG for the Advanced Logistics System, i.e., they are the

techniques judged to have the greatest applicability and utility for the error-environment of ALS.

2. All three techniques demonstrate a high error correction capability upon error detection.
3. The mathematical procedures used in modelling these techniques are representative of the methods necessary for other techniques.

Only a functional description will be given for the rest of the validation procedures. A complete discussion of the resource requirements for all of the procedures is presented by Browne and Cunningham [9].

4.2 ZDMS--The Data Base Management System for ALS

The ZODIAC Data Management System (ZDMS) of the U.S. Air Force's Advanced Logistics System (ALS) is a data base management system implemented on a multi-mainframe Control Data Cyber 70 installation. The system supports several million words of Extended Core Storage which serves principally as an intermediate storage medium between main memory and the disk units.

This section presents a summary of the basic interrelationships, format, and terminology of the ZDMS tables, indexes, and data records. Only those tables and table detail, necessary for an understanding of succeeding development, are presented. Figure 8 presents a graphical presentation of a simplified ZDMS table

relationship. The following paragraphs briefly describe the format and use of each table.

The Primary Directory is permanently maintained in Extended Core Storage. The first portion of the Primary Directory is used for bookkeeping and system control functions. The second portion of the table (Data Base Table) contains an identification for each data base in the system including pointers (Extended Core Storage addresses) to the Secondary Directory for each data base.

A Secondary Directory for each active data base is maintained in Extended Core Storage. The directory is divided into two parts. The first part consists of pointers (disk addresses) to tables of primary importance in the data base. The second part is the Index Pointer Table; the Index Pointer Table consists of one entry for each data base index which includes an index identification and the disk address of the highest level block for that index.

The Data Element Table Directory is a disk resident table that lists the blocks of the Data Element Table. It is an ordered list of the first entry in each block of the Data Element Table. A user therefore reads only those blocks that contain the elements that he needs.

The Data Element Table is a disk resident table which contains information about each element of the data base; this information includes the elements length, number of possible occurrences, and whether or not the element is indexed. The table includes a search list of elements ID's and a corresponding element description list. Given an element ID, a binary search technique is used on the search

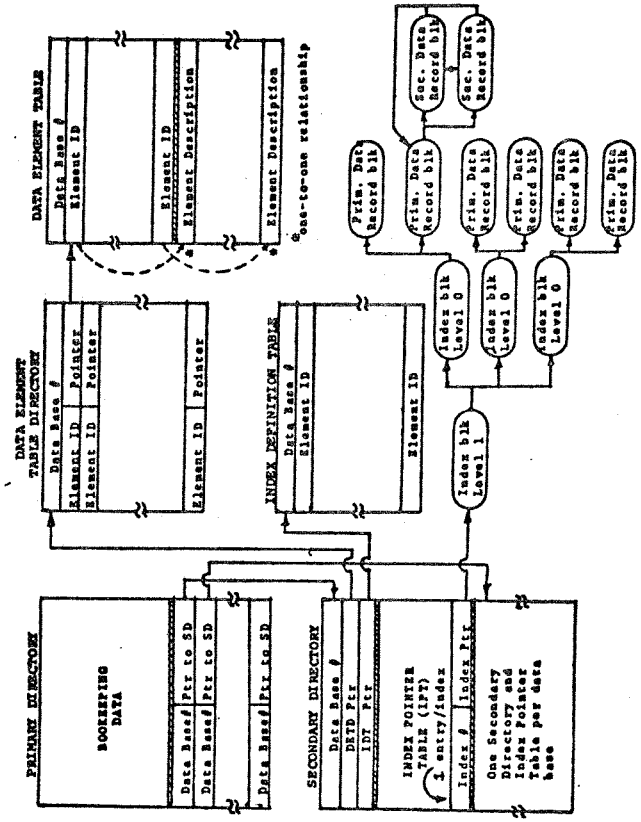


Figure 8 - Simplified ZDMS Tables' Relationship

list to locate the element. When a match is found, the corresponding element information entry is obtained.

The Index Definition Table is a disk resident table which lists the element ID's and field length of indexed elements. The first entry of the table is index 1, the second entry is index 2, etc.

Index Blocks are built in a hierarchy. As an index is built, entries are placed in its level 0 block; an entry includes the value of that key and a pointer (disk address) to its respective Data Record Block. When the first level 0 block is filled, a new one is started. A 1-level block is created to point to the two level-0 blocks present (and to any others that may be created). If enough indexes to fill the level 1 block are created, then another level (2) is used and so on, up to level 7. The disk address of the highest level block in the hierarchy is maintained in the Index Pointer Table of the Secondary Directory. This index structure provides a ready means for matching with index queries. All modifications to the index blocks (including expansion and contraction of the blocks) is handled by a system routine, INDEX UPDATER.

Data Record Blocks represent the logical records of a data base and are maintained on disk. A data record can occupy more than one physical block of disk storage (512 words). The first block of the record is the primary block; the remaining blocks (if any) are the secondary blocks. The primary block contains pointers to the secondary blocks, and each secondary block contains the disk address of the primary block. Each block contains information specifying the element

ID and occurrence number of the first data element in the block; the same information is presented for the last data element contained in the block.

Figure 8 does not present one type of ZDMS table which will be of interest in future discussion. For each disk unit, a Block Reservation Table exists which maps the allocated and unallocated sectors (sector = 64 words) of the given disk unit to a directory entry in the table. Each sector is represented by a bit. The Block Reservation Table occupies a fixed position on its respective disk unit. As Index or Data Record Blocks are allocated (deallocated) on a given disk unit, the appropriate number of adjacent bits in the BRT are set (reset).

This section has presented an extremely simplified overview of ZDMS tables. Much of the actual table detail has been omitted and some ZDMS tables were not presented. Those tables discussed are sufficient to allow the development of topics in future sections. A complete description of all ZDMS tables is given in [10].

4.3 Structure Validation Techniques

4.3.1 Index Verifier

Assuming the existence of an index orientation for a data base, two desirable tests are possible: 1) validation of index values versus data values, which tacitly implies a validation of index addresses used for data value retrieval, and 2) validation of the

format of the index tables. Given the index block structure of ALS, both of these tests can be incorporated. For test 1, the functional description is:

$f_1(IB_i) \rightarrow S$

where:
IB_i: index block for element i
S: a set of triples {a,b,c} where:
a is disk address of the data record
b is the value of the index
c is the occurrence number of the index

The set S is then validated against the data record blocks of the data base. For test 2, owing to the sorted order of the index blocks in ZDMS, the functional description is:

$f_2(IB_i) \rightarrow \{T,F\}$

where:
IB_i: index blocks for element i
{T,F}: a true or false value selected according to satisfaction of correct index sort sequence

4.3.2 Inverse Index Verifier

As a corollary to the Index Verifier an Inverse Index Verifier makes the determination that every indexed data element is pointed to by an index entry. The functional description is presented as a compound function:

$g(IB_R) \rightarrow DRB$

$f(DRB) \rightarrow \{S_1, S_2, S_3, \ldots, S_n\}$

where: IB_R are the index blocks of the root element
DRB are the data record blocks of the data base
S_i are a set of triples {a,b,c} as defined in section 4.3.1 above, for each single index i
n is the number of single indexes

The set S is then evaluated against the index blocks of the data base (correct index blocks are referenced through the use of the Data Element Table and the Index Pointer Table).

Validation with this technique may not always be possible. All data base strings are accessed through the index list of the root element. A missing or incorrect entry in the root element index list would cause the corresponding data string to be inaccessible or "lost." Other indexed elements may point to the "lost" block(s), but this is not guaranteed.

Although an Inverse Index Verifier is a logical and desirable extension of an index verify procedure, it necessitates an unfortunately large amount of disk accessing in comparison to an Index Verifier. This is due to the fact that index blocks are read and traversed repeatedly as each new string is processed for triples. This potentially large duplication of index block accessing would result in inordinately large resource usage.

4.3.3 Index Access Verifier

4.3.3.1 Functional Description

An Index Access Verifier (IAV) was derived initially only as a means to eliminate the replication of disk accessing inherent in the Inverse Index Verifier. It was then observed that this modification yielded a technique which effectively accomplishes both Index Verify and Inverse Index Verify processing plus the ability to identify "lost" blocks, (see discussion in section 4.3.2), missing or duplicated index block entries, incorrect index block entries, index entries for non-existent data string element values, or, in general, any discrepancy between index block information and actual values in the strings.

The processing of an Index Access Verifier is described as follows:

1. Begin retrieval of data record blocks by a scan of the zero-level index blocks of the root element (same as Inverse Index Verifier).
2. Perform Inverse Index Verify type operation on individual data record blocks, but--instead of accessing the index blocks--record to tape 4-tuples of the form

   (index ID, value, occurrence, disk address of data record block)

   for every valued element index as defined in the Index Definition Table.

3. Dump the existing zero-level index blocks to tape.
4. Sort tape on a triple key using as the primary key the index ID, and the value/occurrence and disk address as the secondary keys. Compare to dumped index elements.

In addition to the positive features of this method discussed above, a further benefit is gained by reducing the foreground nature of the processing since both steps (3) and (4) can be relegated as background tasks from tape. The technique does suffer in two respects: a) The object data base needs to be locked (write prevention) until the completion of step 3 so that queued updates are not processed; update entires would be detected as "errors" since the 4-tuples on tape would not reflect the change. b) Errors and/or corrections to be posted must take into account transactions completed since the data base was unlocked.

It has been found that the effect of both of these factors can be diminished by using the existing system utility LOGQUE which produces an audit trail. Once IAV begins processing the time of IAV initiation is recorded, as is IAV's termination time (where the termination time is the time at which the dumping of index entries has been completed.) By recording all index transactions from LOGQUE, the impact of the two previous detracting aspects can be reduced as follows:

1. Only individual index blocks need to be locked during dumping. The dumped index file can be modified later by the relevant LOGQUE entries.
2. Utilizing the LOGQUE entries, the discrepancies between the dumped indexes and the IAV generated indexes can be validated against LOGQUE index entries. This determines whether the discrepancy found is indeed an error or merely an index entry generated prior to an applicable update.

The following modified processing pattern is produced by incorporating the above modification into IAV:

a. Initiate LOGQUE. Begin retrieval of data record blocks by a scan of the zero-level index blocks of the root element.
b. For each logical string retrieved, record on tape a sequence of four-tuples of the form (index ID, value, occurrence, IAID). The sequence will have one four-tuple for every valued index element as defined in the Index Definition Table.
c. Dump the existing zero-level index blocks to tape, locking block-by-block.
d. Dump LOGQUE index entries for source data base from the time of process initiation.
e. Sort generated four-tuples and LOGQUE entries separately on a triple key using index ID as the primary key, and the value/occurrence and IAID as the secondary keys.

f. Compare dumped zero-level index blocks with sorted four-tuples. Validate discrepancies as errors with the use of the sorted LOGQUE entries.

In summary the IAV modifications described provide a means of efficiently validating a data base's indexes that avoids data base lockout and that allows for the index transactions made during its processing cycle. A further benefit is that this change in IAV processing involves no attendant development costs since LOGQUE is an existing system function.

4.3.3.2 IAV Error Detection Resource Costs

Resource usage will be presented as a macro-level flow-chart with resource usages appended to each major sub-process. The flow-chart presents the resource usage for a whole data base where:

α_i = # of index blocks (all levels) for index$_i$

β_i = average # of entries in an index block for index$_i$

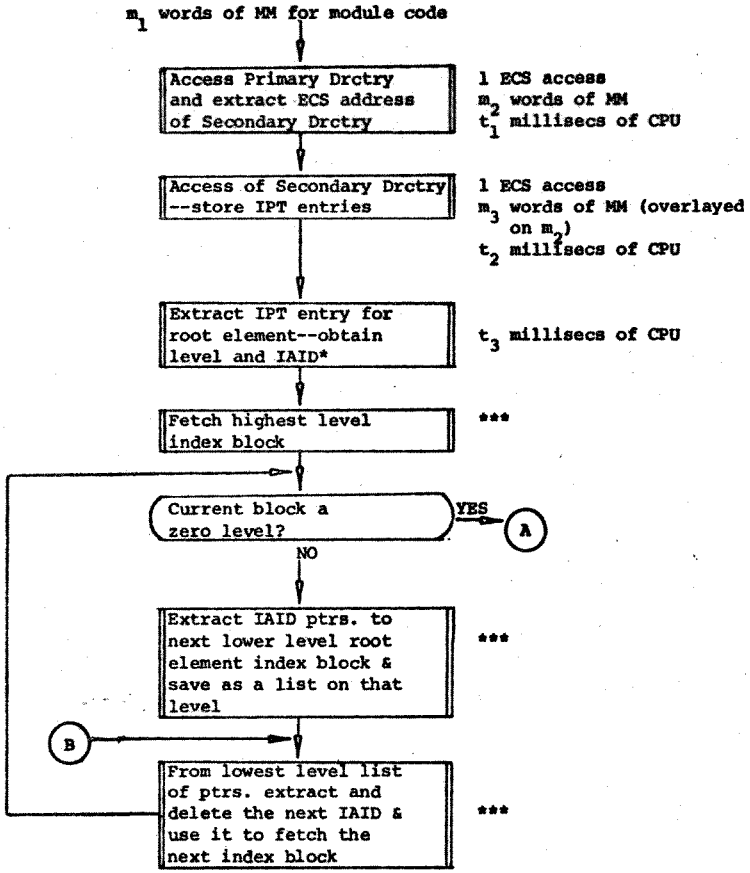
δ_i = # of levels in index block hierarchy for index$_i$

τ_i = # of zero-level index blocks for index$_i$

n = # of indexes

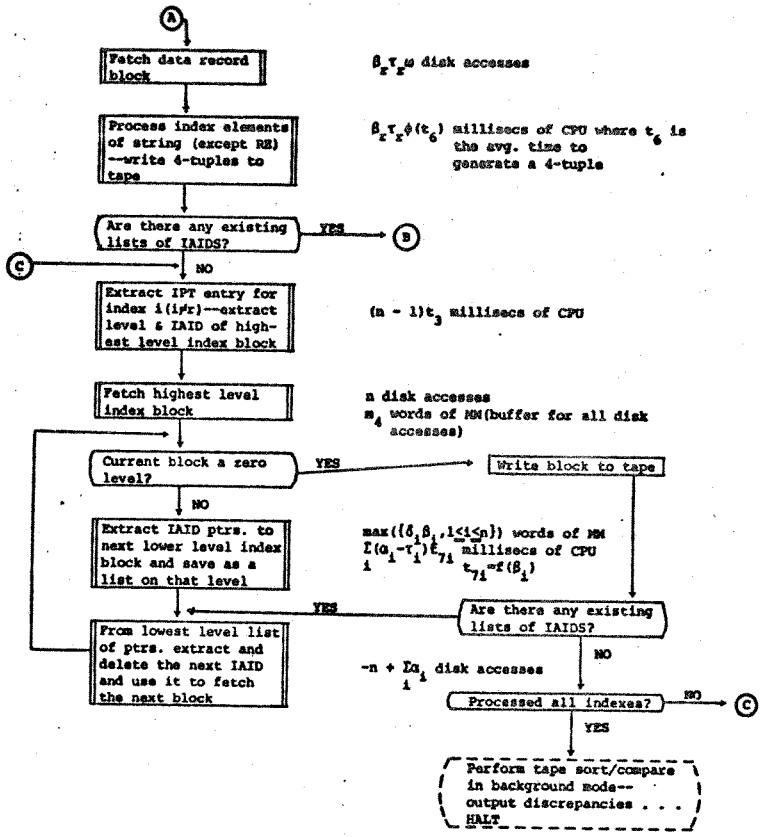
w = average # of blocks in a data string

ϕ = average # of valued index elements per string



*: IAID ≡ pointer to a disk block

***: Resource usage combined and reflected in corresponding process during index block dump (next page)



The following formulas yield the resource usaged for the Index Access Verify procedure in its' foreground processing:

ECS accesses = 2

Disk accesses $\tilde{=} \beta_R \tau_R W + \sum_{i \neq r} \alpha_i$

words of MM = $M_1 + \max(m_2, M_3) + m_4 + \max(\{\delta_i \beta_i, 1<i<n\})$

time = $t_1 + t_2 + nt_3 + 2t_E + \beta_R \tau_R (Wt_D + \phi t_6) + \sum_i (\alpha_i (t_{7i} + t_D) - \tau_i t_{7i})$

where t_D = average disk access time

t_E = average ECS access time

If a comparison were made with Index Verify it would be found that fewer disk accesses are made over an entire data base ($\tilde{=} \sum_{i \neq r} \tau_i \beta_i \gamma_i$ disk accesses). This savings is accounted for by noting that the Index Verifier accesses data record blocks as many times as there are index entries for a given data record.

The sort/compare processing done as a background task is not included in the analysis of the foreground processing since the only resource affected other than tape drive handling is the use of computation cycles which fill in otherwise possibly inactive periods of CPU activity. The actual time necessary for completion of the background task would be a function of the degree of multiprogramming, CPU utilization, and the volume of index information.

An idea of the resources consumed by IAV can be given through the use of an example. Example 3 computes the disk accesses made by IAV for an average size ALS data base.

Example 3:

β_R = average number of entries in a root element index block = 5

τ_R = number of zero-level root element index blocks = 200

W = average number of blocks in a data string = 3

$\sum_{i \neq r} \alpha_i$ = total number of non-root element index blocks = 500

Thus,

Disk Accesses = $\beta_R \tau_R W + \sum_{i \neq r} \alpha_i$

= 5·200·3 + 500

= 3,500 disk accesses

It should be clear that as the index volume of a data base increases, the disk accesses made by IAV increase dramatically.

4.3.3.3 IAV Error Correction Resource Costs

IAV can accomplish automated correction of errors by 1) posting corrective actions for Index Updater to process, or by 2) building a new pyramid index structure from the generated zero-level index blocks and then performing a total replacement of the old index block hierarchy. For a data base with large index volume or few index errors, alternative 1) would appear to be appropriate, while

for data bases of small to moderate size with large numbers of index errors alternative 2) becomes more attractive.

For the purposes of this report alternative 1)--Correction through use of Index Updater--will be used for quantification of correction resource cost. This selection presupposes correct processing by Index Updater.

Three general categories of errors are detectable and correctable by IAV. They are characterized by the following Index Updater activity.

1. Bad value or pointer--post deletion to Index Updater
   post addition to Index Updater
2. Duplicate entry--post deletion to Index Updater
3. Missing entry--post addition to Index Updater

Assuming that a great majority of errors encountered by IAV will be of class (1), (requiring 2 Index Updater calls) the average correction cost of one error can be approximated as 1.9 calls to IU.

4.3.4 BRT Validator

4.3.4.1 Functional Description

The majority of large-scale data base systems are now disk oriented or, minimally, have some degree of disk interface for backing storage. It thus becomes of interest to attempt validation of the disk storage structure by physical sector or logical block or both. Given that a table exists which maps allocated and unallocated

sectors of each disk unit onto a bit directory, then an error could cause the (re)allocation of a currently active sector, or in a less critical case cause the deallocation of a sector that is unallocated. Utilizing the Block Reservation Table of ZDMS, the functional description of a BRT Validator can be given as:

$f(BRT_i) \rightarrow S_i$

where:
BRT_i is the Block Reservation Table for disk unit i
S_i is the bit map of the set of sectors which have been allocated on disk unit i

$v(S_i, S_i') \rightarrow \{T,F\}$

where:
S_i' is the set of sectors on disk unit i which have allocated DB blocks associated with them (all ZDMS blocks have a distinct header). S_i' is obtained by a scan of disk unit i.
{T,F} is an assigned true or false value dependent on the validation of S_i against S_i'

The generation of S_i' is the key to the BRT Validator. However, beginning sectors of allocated blocks in ZDMS cannot be determined. When a block is deallocated in ZDMS only the BRT is effected; the block is retained on disk with a valid header. Therefore, when

performing a disk scan to generate S_i' it is impossible to distinguish between the header of an allocated block or deallocated block. For a BRT Validator to be plausible under ZDMS implies the necessity of a special write to disk "zeroing" (any pattern not recognizable as a block header) deallocated sectors as blocks are deallocated, moved, or migrated.

4.3.4.2 BRTV Error Detection Resource Costs

There is a system cost incurred with the implementation of a BRTV. BRTV requires a rewrite operation to disk with every block deallocation so that the deallocated block will be found invalid. If r_a is the average rate of sector deallocation, the additional system cost expended in disk utilization over a time frame t_Q can be given as:

$kr_a t_D t_Q$ where

$k \equiv$ compression factor, $.6<k<1$, which reflects the degree to which contiguous segments are deallocated

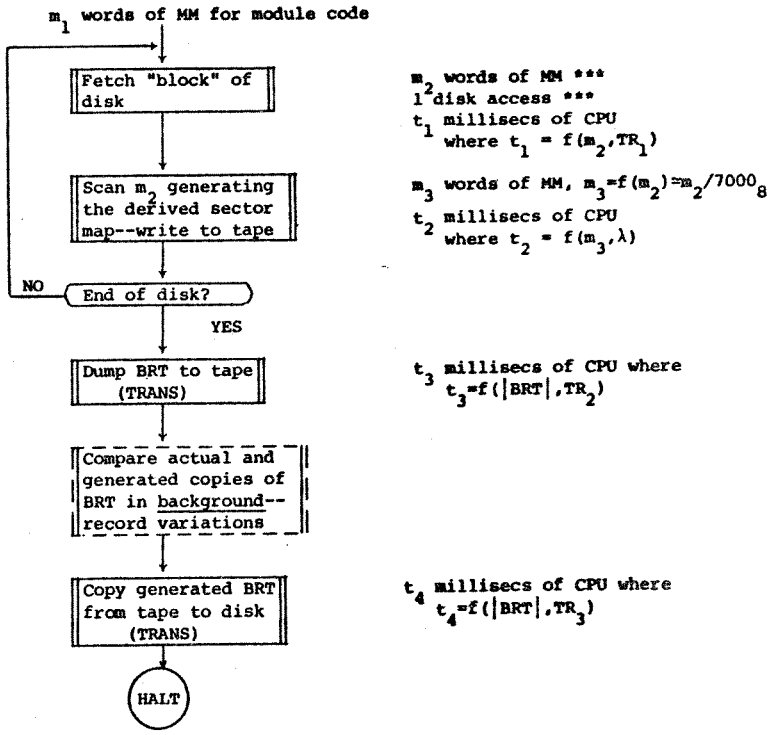
$t_D \equiv$ average real time for disk access plus rewrite

Assuming the rewrite operation was implemented, the resource usage of BRTV processing can be given as follows where:

TR_1 = transfer rate from disk to MM

TR_2 = effective transfer rate from disk to tape

TR_3 = effective transfer rate from tape to disk



***: a tradeoff must be made between the size of m_2 and the number of disk accesses necessary to dump the disk

V = allocated sector % for a given disk unit

S = total words storage on disk

Thus, resource usage can be summarized as:

$$\text{disk accesses} = \frac{S}{m_2}$$

$$\text{words of MM} = m_1 + m_2 + m_3 \tilde{=} m_1 + m_2$$

$$\text{time} = \frac{S}{m_2}(t_1 + t_2) + t_3 + t_4 \tilde{=} \frac{S}{m_2}(t_1 + t_2)$$

Those resources consumed in background processing are not included as part of the analysis.

BRTV is a type of disk dumping routine. There exist two other disk dumping utilities in ZDMS. Pack-to-Pack Copy (PPC) which dumps from disk pack to disk pack, and TRANS which dumps from disk to tape. Using figures for the processing time of PPC and TRANS, bounds can be set on the run-time performance of a BRTV.

PPC $\tilde{=}$ 5 min < BRTV run-time < 15 min $\tilde{=}$ TRANS

A BRTV will run in a time span closer to that of PPC. This allows a lowering of the upper bound to approximately 10 minutes.

4.3.4.3 BRTV Error Correction Resource Costs

As such, a BRTV does not perform a correction function but insures conformability of the BRT to active segments; thus, the BRTV is assigned no correction resource costs. Even if one wishes to argue that BRTV does perform a correction function it can be stated that no

additional resource cost is encumbered beyond the costs already assigned to "detection."

4.3.5 Element/Occurrence Partitioning

Demarcation of the ends of all valued occurrences for each element of a data record provides a means of data record partitioning by element. Use of this type feature provides a means of data record format validation along with an explicit check of schema information (element picture length). Assuming a partition symbol of # (the selection of a single character that can be universally used for # is one of the principal disadvantages of this technique), the following basic data record format would be noted:

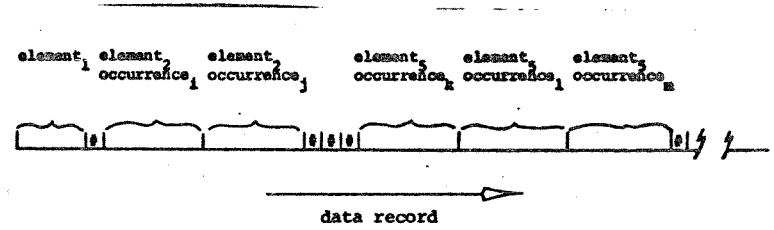


Figure 9 - Element Partitioning

Two or more partition symbols in sequence implies an element(s) that is void of valued occurrences. Counting the number of characters (bits) between respective #'s yields a check on redundantly stored information of element (field) picture length and the number of valued

occurrences. Or, vice versa, the picture length and number of valued occurrences provides a check against the number of characters between consecutive #'s.

Since this structure implies not storing unvalued element occurrences, a problem of occurrence identification occurs for multiply defined elements with non-valued occurrences. Either an extraneous recording of valued occurrences must be kept, or a second partitioning symbol, say $, is introduced to demarcate individual occurrences within an element. Use of a second partitioning symbol would yield a structure:

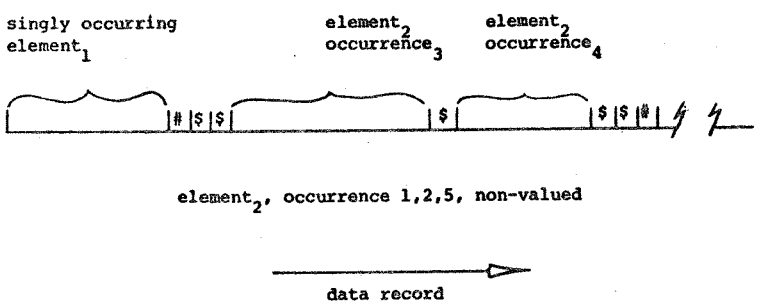


Figure 10 - Element and Occurrence Partitioning

A variation of element/occurrence partitioning already exists as part of the data record format in ZDMS.

4.3.6 Access Authorization

A typical reliability and security measure present in most data management systems is authorization codes. Most authorization codes control access to data bases at a number of levels of resolution: at the data base level, group level, and single element level. These authorization codes further may be utilized to limit access authority for any of the given levels to read-only, read-write, deletion and/or creation.

Access authorization can be used for structure validation in the following ways: 1) Implement access authorization codes for data bases containing system critical data. 2) Implement similar codes for user access of any structure defining tables--i.e., Primary Directory, Secondary Directory,..., Index Blocks, Block Headers, etc. Any access authorization procedure can be described as the basic function.

$V(AC,E) \rightarrow \{T,F\}$

where
AC is an authorization code
E is the system table to which access is being attempted
{T,F} is a true or false value assigned according to the validity of AC

While access authorization codes are not actually capable of structure validation, they are presented due to their positive effect on the probability of structure validity.

4.3.7 Consistency Checks

Consistency checks are procedures which validate element content against established upper or lower bounds or some other form of base field description which describes a range of legal values. It can easily be argued that consistency checks comprise a form of content validation and not structure validation. However, particularly if the consistency check utilizes field picture descriptions, a validation is not being made of the field content but of the structure of the field content. Indeed, the use of field picture description is in many cases useless for content validation due to the many variations in content capable of going undetected (i.e., any field with picture X). A functional description is given as:

$f(DET) \rightarrow P_i$

$V(P_i, D) \rightarrow \{T,F\}$

where:
DET is the Data Element Table
P_i is the picture for element i
D is the data element value
{T,F} is a true or false value assigned according to the comparison made by the function V

4.4 Content Validation Techniques

4.4.1 Checksum/Parity--Algorithm HVD

Instead of considering the wide range of available checksum/parity schemes, a single algorithm has been selected due to its

relatively low systmem cost in comparison to its ability for content error detection and correction.

Algorithm HVD, as presented by Dalton [6], utilizes three parities, horizontal, vertical, and diagonal with wraparound. For a mxm matrix a diagonal with wraparound is composed of diametricly opposed diagonals that satisfy the condition that if p and q are the lengths of the opposed diagonals then they comprise a diagonal with wraparound if and only if p + q = m. All one and two bit errors are detectable and correctable, while all three bit errors are detectable but only 85% of the three bit errors are readily correctable.

4.4.1.2 HVD Error Detection Resource Costs

The relevant resource analysis presented by Dalton for HVD with respect to ALS is summarized as follows:

Storage Requirements - if M represents the total number of words of data to which HVD is to be applied, then $\sim\frac{M}{20}$ words of storage will be necessary to support the generated parities.

Processing Requirements - assuming HVD application across the whole system the following percentage increases in response time occur:

lightly loaded system - 5% increase

heavily loaded system - 3% increase

Assuming a linear decline of effect on response time as system loading increases, we can note that if x=transaction rate then

(1) % increase in response time = $\begin{cases} 5 & \text{if } x \leq p_1 \\ 5-2\left(\frac{x-p_1}{p_2-p_1}\right) & \text{if } x>p_1 \end{cases}$

where $p_1 \equiv$ transaction rate representative of light loading

$p_2 \equiv$ maximum supportable system transaction rate

p_1 can be assigned the mean system transaction rate while a value for p_2 can be obtained through the use of a system limiting capability analysis as presented by Browne, et al [11].

If HVD is not applied across the whole system or a subject data base, further adjustments to the above formulation need to be made. Assuming p_3 is the percentage of transactions that reference HVD blocks then equation (1) becomes:

% increase in response time = $\begin{cases} 5_{p3} & \text{if } x \leq p_1 \\ (5-2\left(\frac{x-p_1}{p_2-p_1}\right) p_3) & \text{if } x>p_1 \end{cases}$

4.4.1.3 HVD Error Correction Resource Costs

Because the ratio of the mean number of HVD blocks processed with errors to those without error is very small, the additional processing costs of automated correction will be small and have no appreciable effect on the stated processing costs for detection. Thus, HVD is assigned zero resource cost for automated error correction.

4.4.2 Dual Recording by Indexing

If a data base contains a small number of elements (in general no more than two) which have been identified as critical due to either

usage or content, a means of effectively eliminating resource costs associated with validation of these elements is to key them. The following rules can be used to establish use of this quasi-technique:

a. No more than two elements of the data base are identified as being "critical".

b. The ratio of (index volume/data volume) is at or below a median level established for the whole system.

c. The indexing of the element(s) does not significantly alter the ratio of step b.

CHAPTER V

THE ALS-EFG AND ITS ANALYSIS

5.1 Introduction

This chapter concludes the Advanced Logistics System example of modelling a data integrity management system. A subset of the validation procedures presented in chapter IV will be incorporated as the data integrity management system for ALS. With the identification of these techniques the ALS-EFG is constructed and then analyzed according to the techniques developed in Chapter III. Concluding sections of the chapter will present data gathering requirements for the analysis along with a discussion of the model's utility.

5.2 Applicable Validation Techniques

As was mentioned in section 4.1, three of the validation techniques from Chapter IV will be incorporated into the data integrity management system for ALS. They are Index Access Verifier, Block Reservation Table Verifier, and Algorithm HVD.

5.3 Other Detection Procedures

ALS has inherent error detection capabilities along with some existing validation utilities. These procedures are reflected in the following two sections.

5.3.1 Bent Pointer Detection

Any time an access is made through ZDMS indexes, the access can be blocked by the system's detection of an invalid pointer. This inherent system capability for detection of "bent pointers" can be characterized as a continuous error detection procedure and will be treated as such in the ALS-EFG. Since bent pointer (BP) detection is an offshoot of normal system processing, no resource costs are assigned for detection; further, since this detection capability has no matching correction ability there is no assigned correction resource costs.

Even though a BP error path is provided in the ALS-EFG, this path has non-zero branching probability only if Index Access Verifier is not being used. This relation is present in order to avoid double error accounting in the model since Index Access Verifier is capable of detecting 100% of the bent pointers.

5.3.2 Sporadic Techniques

As was discussed in section 2.4, sporadic or sampling techniques represent the third general category of validation techniques along with periodic and continuous techniques. The sporadic techniques present in ALS are typified by such routines as ZXA001 (random consistency checking) and ZXA006 (random string printing for manual verification). Errors detected by interactive or batch users as a result of system output or generated reports are also considered in this category.

Since none of these techniques is truly continuous or periodic they present special problems for inclusion in the ALS-EFG. The problems encountered are twofold: a) identification of a means to make these sources of error detection compatible with the continuous-periodic technique development of the EFG, and b) establishment of a means to formalize or quantify resource costs.

For the ALS-EFG problem (a) has been resolved by i) lumping all error detection from sporadic techniques into a single DETECT state and ii) assumming the resultant DETECT state represents a single quasi-continuous procedure. Most of the individual techniques would require approximations of their behavior and costs; it is therefore preferable to collapse all such techniques into a single quasi-technique. This technique would require only a single knowledgeable approximation.

Problem (b) is amenable to a partial solution by considering only the system related expenses that are quantifiable--i.e., the machine costs of ZXA001. For EFG analysis those portions of resource costs attributable to human factors are disregarded.

Thus, the set of sporadic techniques for the ALS-EFG will be represented as a single quasi-continuous validation technique. The technique is assigned a resource cost that reflects only the system-related costs of the composited techniques.

5.4 The ALS-EFG

Given the set of error paths out of the ERROR FREE STATE discussed in the previous sections, the structure of the ALS-EFG becomes apparent. The ALS-EFG is presented in Figure 11. The following sections will deal with parameterization, branching probability computation, and error/resource analysis of this EFG.

Before proceeding with discussion it is advantageous to reiterate the assumptions inherent in any EFG and the specific assumptions necessary for the ALS-EFG.

1. An EFG presupposes all system related functions--hardware, system software, data bases, ED/C procedures, etc.--demonstrate long term equilibrium (steady state) behavior.
2. Detected errors are separated by a minimal average time span of three transitions (seconds)--see section 3.6.
3. Due to the lack of memory in an EFG, detected-uncorrected errors are merely tagged in the NO CORRECT state before transition to the ERROR FREE state.
4. Different data bases of the same system are reflected by changes in the branching probabilities of the system's EFG.
5. Manual correction procedures are characterized only by the computer resources consumed since an adequate representation of human cost factors has not been identified.
6. The Set of system sporadic techniques are composited into a single quasi-continuous procedure.

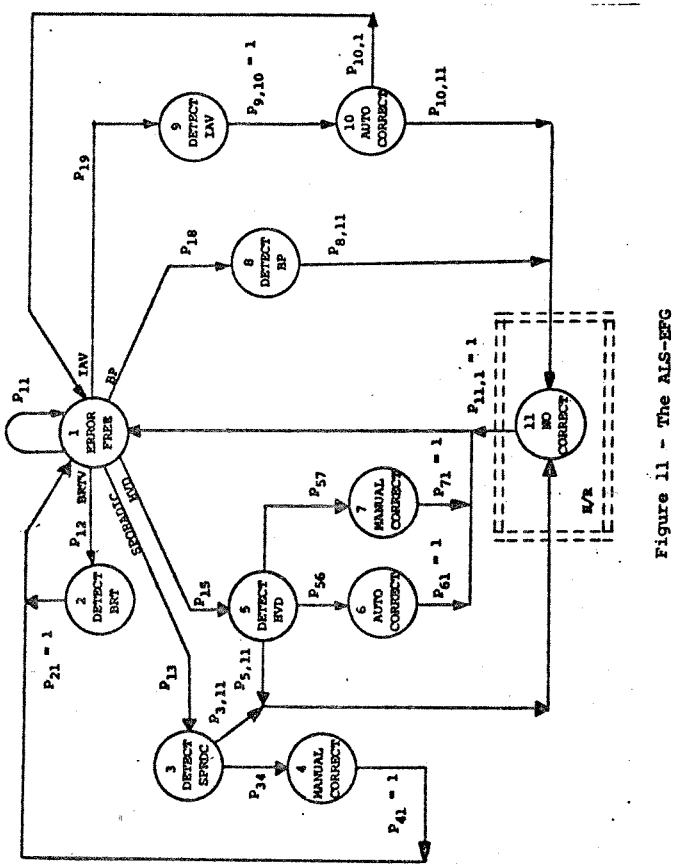


Figure 11 - The ALS-EFG

5.5 Parameterization of the ALS-EFG

Drawing from the discussions of parameterization presented in sections 3.9 and 3.12.2 the ALS-EFG can be seen to require the following:

1. Periods of all periodic techniques
2. % application of each continuous technique
3. Approximation of detected error rate for sporadic techniques
4. Hardware error rate due to memory flaws and transmission errors
5. Transaction rate against given data base
6. System transaction rate (system load level)
7. Data base size
8. Additional variables to satisfy the parametric formulations for resource usage developed for the various validation procedures (Chapter IV).

Quantification of any of the above parameters should be straightforward and reasonably exact with the exception of 3) which is an approximation. Parameters 1) and 2) do not have unique "correct" values since one of the purposes of the model is to allow analysis of mixes of validation techniques in order to provide a capability for optimization of cost and effect.

5.6 Computation of Branching Probabilities

A cursory examination of Figure 8 reveals that some of the states have single exit paths. These paths' branching probabilities are static at one. Other branching probabilities are also static but are not obvious. The remainder of the branching probabilities are dynamic, i.e., they are dependent on the parameterization. Some of these dynamic branching probabilities are simple functions of the parameterization while others--e.g., the discrete branching probabilities for the validation procedures out of the ERROR FREE state--are complex functions of the parameterization.

The following discussions will present analyses of the branching probabilities with respect to the parameterization on a state-by-state basis. Each analysis will deal only with the exit paths out of the given state (except when those exit paths are static at one).

ERROR FREE STATE - Using the analytic development of section 3.5 the problem becomes one of generating the mean detected error rate for each validation procedure. For continuous procedure HVD these error rates can be expressed as a function f.

$$\text{HVD error rate} = f(\%, S, TR_D, TR_S, E)$$

where % ≡ % technique application over data base

S ≡ data base size mean

TR_D ≡ transaction rate against data base mean

TR_S ≡ transaction rate for system

E ≡ flaw rate of memory hierarchies and transmission error rate

This error rate is directly proportional to each of the parameters but data is not available to establish the exact functional relation. Data requirements and analysis are discussed in section 5.9.

The bent pointer net error rate is a datum readily established by trace data for index accessing. However, this error rate is used if and only if the period of Index Access Verifier is zero (see discussion of section 5.3.1).

The error rates for BRT Validation and Index Access Verifier are obtained by determining the expected number of errors detected upon a random application of the technique and then dividing by the respective period. The number of errors for BRTV will be a function of the disk utilization, flaw rate, and transmission error rate. The number of errors for IAV will be a function of index volume and index transaction rate. Approximations of these rates can be made using a linear relation of the parameters pending collection and analysis of relevant data.

The error rate for the sporadic path is supplied in the parameterization, leaving only p_{11} to be determined. Summing the error rates of the ED/C procedures yields the net number of errors detected per time quantum. If r is this net error rate, then $p_{11} = 1 - r$.

DETECT SPORADIC - Through subjective analysis of the errors detected by the composited sporadic techniques, an approximation of

the percentage of these errors which would undergo manual correction can be made. If this percentage is given as x, then $p_{34} = x$ and $p_{3,11} = 1 - x$. These two branching probabilities are static values.

DETECT HVD - In a manner similar to that discussed for DETECT SPORADIC, subjective analysis is used to provide the percentage of errors detected by HVD which would be corrected automatically and which would be corrected manually. Considering HVD's ability to automatically correct all one and two bit errors and 85% of the three bit errors, it is reasonable to let $p_{56} = .99$. Assuming that the majority of the remaining errors would be beyond the scope of most manual correction procedures, p_{57} is set at zero while $p_{5,11} = .01$. These values are static.

AUTO CORRECT (IAV) - The only type of error that cannot be corrected by Index Access Verifier is a bent root index pointer for a data record which contains no other valued index elements. Thus $p_{10,11}$--the probability of this type of error--will be a function of the number of indexes in the data base and the percentage of possible index elements (other than the root element) which are valued. A statement of this function can be given as

$p_{10,11} = k * \frac{V}{n}$ where k ≡ a constant equal to the probability of an uncorrectable error for a data base with no indexes besides the root element

n ≡ total number of indexes

V $\prod_{i=2}^{n} (1 - f_i)$, f_i = % of possible elements for index i that are valued. The pi-product starts at i = 2 to indicate that the root element index is not used in computing V.

The value of $p_{10,1}$ is logically set to $1 - p_{10,11}$.

In summary all of the branching probabilities of the ALS-EFG will be static with the exception of the paths out of the ERROR FREE state and the paths out of the IAV AUTO CORRECT state. Of these dynamic paths only those exiting the ERROR FREE state pose problems for formalization; this is due to the complex interrelation of data base and system characteristics in determination of the mean detected error rates of the individual validation procedures. In lieu of adequate data gathering and analysis to establish these functional relations, the detected error rates of the validation procedures must initially be manually approximated.

5.7 Steady-State Analysis of the ALS-EFG

It was established in the previous section that the following branching probabilities of the ALS-EFG have the static values:

$p_{21} = 1$ $\quad$ $p_{9,10} = 1$

$p_{41} = 1$ $\quad$ $p_{8,11} = 1$

$p_{61} = 1$ $\quad$ $p_{56} = .99$

$p_{71} = 1$ $\qquad$ $p_{5,11} = .01$

$p_{11,1} = 1$ $\qquad$ $p_{57} = 0$

Using these values and the symbolic p_{ij}'s for all other branching probabilities, the steady state analysis of the ALS-EFG begins by listing the balance equations. The balance equation for state 1 is omitted and replaced by the normalizing equation (1).

1. $P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8 + P_9 + P_{10} + P_{11} = 1$
2. $p_{12}P_1 = P_2p_{21} \Rightarrow P_2 = p_{12}P_1$
3. $p_{13}P_1 = P_3(p_{34} + p_{3,11}) \Rightarrow P_3 = p_{13}P_1$
4. $p_{34}P_3 = P_4p_{41} \Rightarrow P_4 = p_{34}P_3 \Rightarrow P_4 = p_{34}p_{13}P_1$
5. $p_{15}P_1 = P_5(p_{5,11} + p_{56} + p_{57}) \Rightarrow P_5 = p_{15}P_1$
6. $p_{56}P_5 = P_6p_{61} \Rightarrow P_6 = .99P_5 \Rightarrow P_6 = .99p_{15}P_1$
7. $p_{57}P_5 = P_7p_{71} \Rightarrow P_7 = 0$
8. $p_{18}P_1 = P_8p_{8,11} \Rightarrow P_8 = p_{18}P_1$
9. $p_{19}P_1 = P_9p_{9,10} \Rightarrow P_9 = p_{19}P_1$
10. $p_{9,10}P_9 = P_{10}(p_{10,11} + p_{10,1}) \Rightarrow P_{10} = P_9 \Rightarrow P_{10} = p_{19}P_1$
11. $p_{3,11}P_3 + p_{5,11}P_5 + p_{8,11}P_8 + p_{10,11}P_{10} = P_{11}P_{11,1}$

    $\Rightarrow P_{11} = (p_{3,11}p_{13} + .01p_{15} + p_{18} + p_{10,11}p_{19})P_1$

Substituting in equation (1) the values of P_2 through P_{11} generated above and then factoring and regrouping, P_1 is found to be:

12. $$P_1 = \frac{1}{1 + p_{12} + 2p_{13} + 2p_{15} + 2p_{18} + 2p_{19} + p_{19}p_{10,11}}$$

With an expression for P_1, all the remaining steady state probabilities P_2 through P_{11} can be obtained by the substitution of the value of P_1 into the respective equations (2) through (11).

Having solved for the P_i's, the error statistics discussed in section 3.11 become a matter of simple value substitution and as such will not be repeated here.

5.8 Resource Usage Analysis of the ALS-EFG

The total resource costs of each procedure can be computed using the mean detected error rates of each validation procedure available. Instead of manipulating the rather lengthy formalizations of resource cost developed in sections 4.3 and 4.4 of this report, it will be assumed the numeric values of the single error correction and detection resource costs have already been computed and yield the results portrayed in the table given in Figure 12.

As can be seen from Figure 12 the resource usage of the various techniques are expressed in four different metrics. This requires a separate analysis of the EFG for each resource metric. However, this causes no real complication since the collection of resource costs per state is a simple, fast procedure. By matching the resource costs of Figure 12 with their corresponding states in the ALS-EFG the following

resource costs are determined for time frame t_Q (and $t_r = \frac{t_Q}{Q}$ = the number of state transitions made in the EFG):

1. Disk Accesses = $(P_2BD_d + P_3SD_d + P_4SC_d + P_9ID_d)t_r$
2. CPU time = $(P_2BD_t + P_3SD_t + P_4SC_t + P_9ID_t)t_r$
3. Response degradation = $P_5HD_r t_r$
4. Calls to Index Updater = $1.9P_{10}t_r$

TECHNIQUE	COST ELEMENT	RESOURCES CONSUMED			
		DISK ACCESSES	CPU TIME	RESPONSE DEGRADATION	CALLS TO IU
BRT	DETECTION	BD_d	BD_t	0	0
	AUTO CORRECTION	0	0	0	0
SPORADIC	DETECTION*	SD_d	SD_t	0	0
	MANUAL CORRECTION*	SC_d	SC_t	0	0
HVD	DETECTION	0	0	HD_t	0
	AUTO CORRECTION**	0	0	0	0
	MANUAL CORRECTION**	0	0	0	0
IAV	DETECTION	ID_d	ID_t	0	0
	AUTO CORRECTION	0	0	0	1.9

*reflects only system related resource costs
**zero resource usage since P_7 equals zero

Figure 12

Single Error Detection & Correction Resource Usage

If it was wished to express these costs in a single metric, response degradation would be an adequate metric; this common metric could be generated by a model of the ALS system [6, 11] parameterized by (1)-(4). Even if a single resource metric is obtained, it is still advantageous to have the individual metrics since they provide a better insight into the mix of resource costs.

The technique of resource usage analysis developed yields a viable tool for determination of the effect of varying mixes of validation procedures. And, even though most analyses will yield multiple metrics for the cost, this neither precludes obtaining cost in a single metric nor does it necessarily obscure the total cost.

5.9 Data Gathering

Accurate analysis of the ALS-EFG is dependent on the availability of data reflecting the effect of various data base and system parameters on the set of validation procedures incorporated. Analysis of the data is performed on two levels: 1) to establish if there exists a relation between a data base or system parameter and a validation technique, and 2) if a relation exists, then what is the degree of its effect?

Data relating to the processing of a given technique requires the implementation of that technique over a set of data bases of divergent characteristics with varying system load levels. The implication of this statement is that accurate EFG analysis of mixes of validation procedures must follow implementation of the validation

procedures. However, the EFG can be used as a pre-implementation tool with a reasonable confidence interval if knowledgeable estimations of relations can be made.

As an illustration of data gathering and analysis that would be performed after implementation of the set of validation techniques the following example is presented.

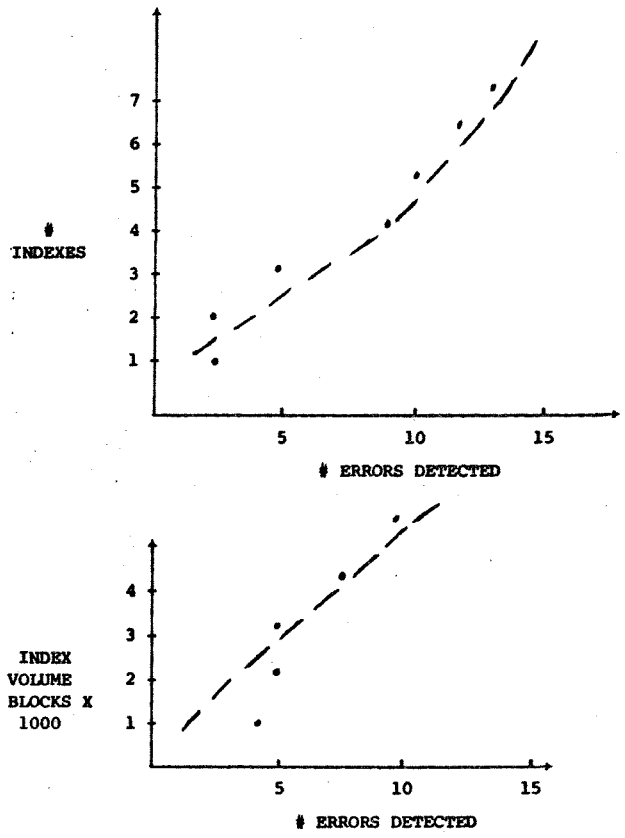
Example 4

Index Access Verifier has been implemented across a number of data bases with widely varying index structures and index volume. IAV has been run with a constant period of 24 hours. The subject data bases over that 24 hour interval were accessed at widely varying rates and under varying system load levels.

By recording the number of errors detected by IAV along with the number of indexes and the volume of the indexes for each data base, the following graphs were obtained.

Similar graphs could be generated for other parameters such as index access rates, index update rates, system load level, etc. Data gathered in this way over a long number of periods should yield graphs reflecting the equilibrium tendencies of the technique for various parameters.

This form of data gathering and analysis will yield concrete results for the accurate analysis of a validation techniques' behavior. It should be emphasized that the graphs discussed will yield the equilibrium relationship between data base and system parameters and



the validation techniques. This is ideal given that the first underlying assumption of an EFG is that all data base and system functions demonstrate equilibrium (steady state) behavior.

5.10 Utility of the ALS-EFG

The utility of the ALS-EFG can be viewed from two standpoints--either as a post-implementation tool or as a pre-implementation tool.

As a post-implementation tool with the equilibrium behavior of the various validation procedures established, the principal function of an EFG would be in accurately establishing the minimal system costs necessary for a given integrity level (assuming there exists more than one mix of validation procedures capable of producing a desired integrity level). The EFG's utility is not however limited to existing data bases. The EFG can yield accurate predictions for an unimplemented data base's needs for validation if the data base's characteristics can be accurately expressed.

As a pre-implementation tool the EFG serves as a means of approximating the resource costs of varying mixes of the set of validation procedures. If approximations of validation techniques' behavior are within $\pm$ 25% of actual behavior, the resource analysis will yield figures within a confidence interval of $\pm$ 25% also. The EFG will minimally demonstrate the cost tendencies even if the confidence interval of the approximations is large.

Either as an accurate post-implementation tool or as an approximate pre-implementation tool, the ALS-EFG has high utility as both a predictive and optimization tool for data integrity management.

CHAPTER VI

CONCLUSION

This thesis has presented three main developments: i) identification of structure validation techniques used for detecting and correcting soft-flaw errors, ii) identification of a set of validation techniques plus R/R which comprise a data integrity management system, and iii) introduction of a Markov model which accurately simulates the processing behavior of a data integrity management system.

The structure validation techniques and data integrity management system derived were for the Advanced Logistics System (ALS) of the U.S. Air Force Logistics Command. Analysis of the ALS data integrity management system resulted in formalizations of resource costs and error probabilities. This analysis also identified relevant data base and system attributes for which it would be desirable to gather data.

The general tool used for all analysis was the Error Flow Graph (EFG) which is a dynamic, equilibrium model of data integrity management systems. An EFG model is not limited either by the type of data base system chosen or by the validation procedures used in the system. The primary restriction placed upon modelling with an EFG

is that all functions of the modelled system must demonstrate--or can be approximated by--long term equilibrium behavior.

An EFG can be used in two ways. As an accurate means of analysis after processing patterns of validation techniques have been established, or as means of deducing cost and error tendencies for unimplemented validation procedures. In either mode of operation, an EFG serves as a firm basis for predictive and decision-making functions within the context of data integrity management for data base systems.

BIBLIOGRAPHY

1. David Lefkovitz, Data Management for On-Line Systems, Hayden Book Company, Inc., N.J., 1974.

2. Morris Rubinoff, ed., Advances In Computers, Vol. 12, Academic Press, N.Y. and London, 1972.

3. K. M. Chandy, J. C. Browne, C. W. Dissly, and W. R. Uhrig, "Analytic Models for Rollback and Recovery Strategies In Data Base Systems," Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712, 1975.

4. P. Elias, "Error-Free Coding," IRE Transactions on Information Theory, PGIT-4, Sept., 1954.

5. R. W. Hamming, "Error Detecting and Correcting Codes," Bell System Technical Journal, Vol. 29, April, 1950.

6. S. F. Dalton, "Error Detection and Correction to Improve Data Base Reliability," Thesis, Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712, 1974.

7. K. M. Chandy, C. V. Ramamoorthy, "Rollback and Recovery Strategies for Computer Programs" IEEE Transactions on Computers, Vol. C-21, Number 6, June, 1972.

8. Alvin W. Drake, Fundamentals of Applied Probability Theory, McGraw-Hill Book Company, N.Y., 1967.

9. J. C. Browne, M. L. Cunningham, "Data Integrity Management--Methods/ Techniques of Data Base Validation," Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712, Jan., 1975.

10. ZODIAC Data Management System Internal Maintenance Specification, Publication #17235800, Revision 5, Control Data Corporation, Oct., 1974.

11. J. C. Browne, K. M. Chandy, R. B. Brown, D. F. Towsley, T. W. Keeler, C. W. Dissly, Limiting Capability Analysis of the Cyber 70 ALS, Information Research Associates, 2317 Longview, Austin, TX 78705, Sept., 1973.