INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of “sectioning” the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.
Dunn, Thurman Stanley

METHODOLOGY FOR THE OPTIMIZATION OF RESOURCES IN THE DETECTION OF COMPUTER FRAUD

The University of Arizona

University Microfilms International

Copyright 1982

by

Dunn, Thurman Stanley

All Rights Reserved
METHODOLOGY FOR THE OPTIMIZATION OF RESOURCES IN THE
DETECTION OF COMPUTER FRAUD

By
Thurman Stanley Dunn

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF BUSINESS ADMINISTRATION
In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
IN THE GRADUATE COLLEGE
THE UNIVERSITY OF ARIZONA

1 9 8 2

Copyright 1982 Thurman Stanley Dunn
As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Thurman Stanley Dunn entitled Methodology for the Optimization of Resources in the Detection of Computer Fraud and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copy of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

[Signature]  
Dissertation Director  
[Date]
STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotations from this dissertation are allowed without special permission, provided that accurate acknowledgement of source is made. This manuscript in whole or in part may be granted by the copyright holder.

SIGNED: [Signature]
This dissertation deals with computer fraud detection as a resource allocation problem. Based on the premise of limited resources, the methodologies presented do not allocate any resources to detection if other computer fraud abatement techniques are adequate. Thus, if it can be shown that deterrents such as high morals or fear of imprisonment are adequate to eliminate the possibility of fraud, zero resources would be allocated to the detection of fraud. Likewise, if deterrents are not adequate but prevention controls are adequate to thwart all would be perpetrators, zero resources would be spent in computer fraud detection.

For those situations where deterrents and prevention techniques are not adequate to preclude computer fraud, a methodology is presented for allocating resources to the detection of fraud in a near optimum fashion. Embedded in this methodology is the realization that organizations exist to do much more than protect themselves from possible computer rip-offs. It is assumed that a very small percentage of organizational resources will be available for the detection of computer fraud. Further, the available resources will probably be a very small percentage of those required to track every activity and transaction through computer systems.

The research for this dissertation included several objectives. The first objective was to assess the magnitude of the computer fraud
problem. The second objective was to evaluate reported incidents of computer fraud, categorize them by fraud type and perpetrator type and develop measures of vulnerability based on frequency of occurrence and dollar impact. The research in these two areas, primarily, forms the basis of the first two chapters. Remaining objectives focused on review and evaluation of techniques for evaluating risks or threats, investigative procedures and computer fraud abatement methodologies.

The methodologies presented in this dissertation incorporate various levels of innovation and contribution. In several instances existing techniques have been used in the development of a methodology. Every effort has been made to give appropriate credit through literature citation in these cases.

The major contributions of this dissertation and the highest levels of innovation and originality may be found in Chapters Four through Seven. These four Chapters deal, in numeric order, with the development of a Detection Quotient, a value which measures the effectiveness of computer fraud detection resource allocation; the development of a Specific Threat Assessment methodology for evaluating threats and assigning threat values in specific systems; a General Solution to the Combinatorial Dilemma which occurs when the large number of possible alternatives precludes a comprehensive analysis and the development of a Resource Optimization Model for allocating scarce resources to the detection of computer fraud in a near optimum fashion.

Although the methodologies presented in this dissertation are original, some of the problems being treated are well known. For
example, the solution to the Combinatorial Dilemma in Chapter Six is, to the author's knowledge, totally original. However, the problems associated with the Combinatorial Dilemma are well known. Specific contributions of the dissertation are further highlighted in Chapter One.

The last two chapters included as Appendices A and B, deal with Investigative and Automated techniques and tools associated with the general area of audit and evaluation of propriety in computer systems. Although the dissertation assumes a capability within organizations to examine the various components of computer systems for fraud once specific threats have been identified and resources made available, some background in the associated techniques and tools which are available was considered essential. In addition, the concept of "Live Monitoring" is introduced in Appendix A for those situations in today's computer environment where audit trails are not adequate or can be altered.

The most significant limitation confronting the researcher in computer fraud is the limitation on information available in the area of computer fraud occurrences. As shown in Chapter One, it has been estimated that only one percent of all computer crime is detected and only about seven percent of those that are detected are reported. Thus, existing data on specific computer fraud cases probably represents a very small percentage of actual cases. An associated limitation is that threat assessment techniques, lacking a comprehensive historical data base of actual cases from which
probabilities of various types of computer fraud may be drawn, must be subjective. The Threat Assessment technique presented in Chapter Five attempts to overcome this limitation by combining the features of the Delphi approach and the Churchman - Ackoff technique with a Matrix approach developed for this dissertation.

Several areas addressed in this dissertation would lend themselves to additional research. The most obvious is expansion of case data. This is probably the most difficult because of the deficiencies in reporting and the hesitance of many organizations to share this information for fear of being considered vulnerable to computer fraud, therefore ineffective. It should be noted that attempts have been made without much success to expand research in this area. An example of such an attempt is provided in Chapter One.

Another candidate for research is the deterring effects of computer fraud detection capabilities, or perhaps more appropriately, the perception of these capabilities in the minds of would be perpetrators. It would be highly beneficial to know whether a small perceived detection capability would discourage only a small percentage of would be perpetrators or whether a much larger percentage would be discouraged. This topic is introduced in Appendix A, where it is suggested, on an intuitive basis, that the latter is probably a more accurate assessment. Quantification of this relationship would greatly expand the usefulness of the Computer Fraud Detection Model presented in Figure 8 of this dissertation.
A final area which should provide a good research potential is expansion of the concept of "Live Monitoring" introduced in Appendix A. In today's rapidly expanding use of distributed systems, mini and micro computers, communications networks and real-time processing, the concept of "Live Monitoring" presents challenges, both manual and automated, well beyond the cursory treatment in Appendix A.

**Acknowledgements**

I wish to thank very much those people who have supported me and contributed to my doctoral research. Dr. Jay F. Nunamaker, my Major Advisor, has been very supportive in guiding me through the basic process of doctoral research as well as technical issues in my chosen topic. He opened my eyes to the exact nature of several problems confronting me during the research, allowing me to focus more precisely in the development of solutions. Dr. Mary S. Loomis has provided considerable technical assistance both in the subject matter and the compilation of the text. Her editorial assistance has been invaluable. Dr. Benn Konsynski has provided numerous research sources and stimulated me to address more thoroughly the issues closely associated with computer fraud. Donn Parker of The Stanford Research Institute has provided considerable assistance. First, by making his extensive data base of computer crime incidents available to researchers in general and, second, by providing me with additional information from his files when I requested it. Finally, my sincerest gratitude is extended to my wife, Karen, for her typing support, editorial support
and moral support and my children, Michelle and Stanley who were so patient on those many occasions when my research pre-empted recreational activities.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>xii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>xv</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Summary of Contributions</td>
<td>12</td>
</tr>
<tr>
<td>Vulnerability Formula</td>
<td>13</td>
</tr>
<tr>
<td>Computer Fraud Detection Model</td>
<td>13</td>
</tr>
<tr>
<td>Threat Analysis</td>
<td>13</td>
</tr>
<tr>
<td>Computer Fraud Detection - A Resource Optimization Problem</td>
<td>14</td>
</tr>
<tr>
<td>Detection Quotient</td>
<td>14</td>
</tr>
<tr>
<td>Specific Risk Assessment Methodology</td>
<td>14</td>
</tr>
<tr>
<td>Controls Analysis</td>
<td>14</td>
</tr>
<tr>
<td>Combinatorial Dilemma</td>
<td>14</td>
</tr>
<tr>
<td>Resource Optimization Model</td>
<td>15</td>
</tr>
<tr>
<td>2. THE TYPOLOGY</td>
<td>16</td>
</tr>
<tr>
<td>3. THREAT ASSESSMENT</td>
<td>26</td>
</tr>
<tr>
<td>Constraints</td>
<td>27</td>
</tr>
<tr>
<td>Time Criticality</td>
<td>27</td>
</tr>
<tr>
<td>Human and Dollar Constraints</td>
<td>28</td>
</tr>
<tr>
<td>The Typology Revisited</td>
<td>30</td>
</tr>
<tr>
<td>Input Transaction Manipulation Schemes</td>
<td>33</td>
</tr>
<tr>
<td>Program Modification Schemes</td>
<td>34</td>
</tr>
<tr>
<td>Larcenous Strategies for Modifying Programs</td>
<td>34</td>
</tr>
<tr>
<td>File Alteration Schemes</td>
<td>36</td>
</tr>
<tr>
<td>The Risk Assessment Methodology</td>
<td>36</td>
</tr>
<tr>
<td>Threat Analysis</td>
<td>40</td>
</tr>
<tr>
<td>4. DETECTION QUOTIENT</td>
<td>49</td>
</tr>
<tr>
<td>Discovery (Exploratory) Sampling</td>
<td>51</td>
</tr>
<tr>
<td>Sample Size - Unlimited Resources</td>
<td>57</td>
</tr>
<tr>
<td>Sample Size - Limited Resources</td>
<td>59</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS--Continued

5. SPECIFIC THREAT ASSESSMENT ........................................ 66
   Risk Analysis Techniques ........................................... 67
       The Statistical Approach (Wong 1977) ....................... 67
       Risk Estimation Approach ..................................... 69
       Benefits Approach (Krauss and MacGahan, 1979) ........... 69
       The National Bureau of Standards (NBS) Approach - (NBS, Special Publication April, 1980) ......................... 71
       The Matrix Approach (Fitzgerald 1978) ...................... 72
   A Specific Risk Assessment Methodology ......................... 74
       Specific Threat Analysis ...................................... 74
       Churchman - Ackoff Method .................................... 81
       Derivation of Threat Values from Churchman - Ackoff .... 87
       Controls Analysis ............................................. 90

6. THE COMBINATORIAL DILEMMA - A GENERAL SOLUTION ............ 96
   A General Solution To The Combinatorial Dilemma ............ 100

7. RESOURCE OPTIMIZATION MODEL ..................................... 106
   Mathematical Statement of the Problem ......................... 108
       Objective Function ........................................... 108
       Constraints .................................................. 111
       Decision Variables .......................................... 113
   Applicability of the Combinatorial Dilemma .................... 115
   The Resource Optimization Model ................................ 118
       Summary of the Mathematical Statement of the Problem .... 118
       An English Statement of the Resource Optimization Problem 119
       The Resource Allocation Solution ............................ 120
       Sample Sizes ................................................ 121
       Iterations ................................................... 123
       Application .................................................. 125
TABLE OF CONTENTS--Continued

APPENDIX A: THE INVESTIGATION

<table>
<thead>
<tr>
<th>Investigation Vs Audit</th>
<th>135</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assumptions</td>
<td>136</td>
</tr>
<tr>
<td>Treating The Deterring Effect Of Computer Fraud Detection</td>
<td>137</td>
</tr>
<tr>
<td>Relationship Between Controls And The Investigation</td>
<td>138</td>
</tr>
<tr>
<td>Investigative Peculiarities</td>
<td>139</td>
</tr>
<tr>
<td>Evaluating Internal Controls</td>
<td>140</td>
</tr>
<tr>
<td>Elements of an ADP System</td>
<td>141</td>
</tr>
<tr>
<td>Diminished Audit Trail</td>
<td>142</td>
</tr>
<tr>
<td>Auditing Around the Computer</td>
<td>143</td>
</tr>
<tr>
<td>Auditing Through the Computer</td>
<td>144</td>
</tr>
<tr>
<td>Use of Test Data</td>
<td>145</td>
</tr>
<tr>
<td>Reprocessing</td>
<td>146</td>
</tr>
<tr>
<td>Auditing with the Computer</td>
<td>147</td>
</tr>
<tr>
<td>Investigation of Threats From the Typology</td>
<td>148</td>
</tr>
<tr>
<td>Transaction Manipulation Schemes</td>
<td>149</td>
</tr>
<tr>
<td>Unauthorized Program Modification Schemes</td>
<td>150</td>
</tr>
<tr>
<td>File Manipulation Schemes</td>
<td>151</td>
</tr>
<tr>
<td>Improper Operation</td>
<td>152</td>
</tr>
<tr>
<td>Summary</td>
<td>153</td>
</tr>
</tbody>
</table>

APPENDIX B: AUTOMATED ANALYSIS

<table>
<thead>
<tr>
<th>Threat Assessment</th>
<th>154</th>
</tr>
</thead>
<tbody>
<tr>
<td>Churchman - Ackoff Process</td>
<td>155</td>
</tr>
<tr>
<td>Threat Matrix</td>
<td>156</td>
</tr>
<tr>
<td>Resource Optimization Model</td>
<td>157</td>
</tr>
<tr>
<td>Internal Control</td>
<td>158</td>
</tr>
<tr>
<td>The Investigation</td>
<td>159</td>
</tr>
<tr>
<td>Audit Software Packages</td>
<td>160</td>
</tr>
<tr>
<td>Selection of an Audit Software Package</td>
<td>161</td>
</tr>
<tr>
<td>Sample Packages</td>
<td>162</td>
</tr>
<tr>
<td>Summary</td>
<td>163</td>
</tr>
</tbody>
</table>

Page

135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
LIST OF ILLUSTRATIONS

Figure                                             Page
1. Computer Fraud Cases by Type of Scheme          20
    and Victim Organization
2. Relative Frequency of Occurrence and Average    21
    Loss of Computer Fraud by Type of Scheme
    and Victim Organization
3. Vulnerability by Type of Computer System       22
    and Organization
4. Types of Computer Fraud in Descending Order    25
    of Vulnerability
5. Methods of Computer Manipulation By Type       31
    of Victim
6. Methods of Computer Manipulation By Type       31
    of System
7. Methods of Manipulation by Percentage          32
8. Computer Fraud Detection Model                 39
9. Threat Analysis - Blank Matrix                 41
10. Threat Matrix with Scheme and Perpetrator Types 42
11. Expanded Threat Matrix                         44
12. Threat Values by Schemes and Perpetrators      46
13. Threat Values - Descending Order              47
14. Sample Size Example                            63
15. Calculation of DQ Values                       64
16. Frequency Distribution of Consequential Losses 68
    Over a Period of Time
17. Payroll Risk Analysis Example                  70
# LIST OF ILLUSTRATIONS--Continued

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.</td>
<td>Specific Threat Analysis - Blank Matrix</td>
<td>75</td>
</tr>
<tr>
<td>19.</td>
<td>Batch System</td>
<td>76</td>
</tr>
<tr>
<td>20.</td>
<td>On-Line System</td>
<td>77</td>
</tr>
<tr>
<td>21.</td>
<td>Sample Threat Matrix with Schemes and Perpetrators Identified</td>
<td>80</td>
</tr>
<tr>
<td>22.</td>
<td>Churchman - Ackoff General Form</td>
<td>82</td>
</tr>
<tr>
<td>23.</td>
<td>Churchman - Ackoff Ranking Evaluation for Threat Schemes</td>
<td>84</td>
</tr>
<tr>
<td>24.</td>
<td>Churchman - Ackoff Ranking Evaluation for Perpetrators</td>
<td>86</td>
</tr>
<tr>
<td>25.</td>
<td>Normalized Relative Averages of ST and PT Values</td>
<td>87</td>
</tr>
<tr>
<td>26.</td>
<td>Derivation of Threat Values</td>
<td>88</td>
</tr>
<tr>
<td>27.</td>
<td>Threat Values from Figure 26 Multiplied by 100</td>
<td>89</td>
</tr>
<tr>
<td>28.</td>
<td>Control Point Values</td>
<td>92</td>
</tr>
<tr>
<td>29.</td>
<td>Combined Threat/Control Matrix</td>
<td>93</td>
</tr>
<tr>
<td>30.</td>
<td>Post-Control Threat Matrix</td>
<td>95</td>
</tr>
<tr>
<td>31.</td>
<td>Elastic Demand Curve</td>
<td>98</td>
</tr>
<tr>
<td>32.</td>
<td>Combinatorial Dilemma Solution Logic</td>
<td>102</td>
</tr>
<tr>
<td>33.</td>
<td>Threat Matrix with Threats Numbered</td>
<td>112</td>
</tr>
<tr>
<td>34.</td>
<td>Combinatorial Example</td>
<td>117</td>
</tr>
<tr>
<td>35.</td>
<td>Comparative Sample Sizes for 99 Percent Probability of Finding at Least One Occurrence of an Event When Rate of Occurrence is 0.2 Percent</td>
<td>122</td>
</tr>
<tr>
<td>36.</td>
<td>Sample Iterations of Resource Optimization Model</td>
<td>126</td>
</tr>
</tbody>
</table>
## List of Illustrations—Continued

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>37.</td>
<td>Alternate Sample Iteration of Resource Optimization Model</td>
<td>128</td>
</tr>
<tr>
<td>38.</td>
<td>Perceived Detection Capability</td>
<td>141</td>
</tr>
<tr>
<td>39.</td>
<td>Perceived Detection Capability - Nonlinear</td>
<td>142</td>
</tr>
</tbody>
</table>
ABSTRACT

A methodology is proposed for optimizing the allocation of resources in the detection of computer fraud. The methodology consists of four major segments. First, a threat assessment is performed. A general threat assessment is provided which relies upon reported incidents of computer fraud. Then, recognizing the limitations of computer fraud reporting, a specific threat assessment technique is provided which is based entirely on the characteristics of a given computer system. Both the general and specific threat assessment techniques use a matrix approach which evaluates and assigns threat values by type of computer fraud and perpetrator.

Second, a Detection Quotient is established which measures the effectiveness of computer fraud detection resource allocation for all of the possible combinations of computer fraud types and perpetrators. However, for many computer systems, the large number of possible resource allocation alternatives results in a Combinatorial Dilemma whereby the phenomenally large number of alternatives precludes comprehensive analysis.

This leads to the third major segment of the dissertation, a General Solution to the Combinatorial Dilemma which ensures an alternative very near the optimum while evaluating only an extremely small percentage of possible alternatives.
Fourth, a Resource Optimization Model is provided which, beginning with the results of the Threat Assessment, iteratively assigns varying levels of computer fraud detection resources to different fraud type and perpetrator combinations. Using the general solution to the Combinatorial Dilemma and the Detection Quotient as a measure of the effectiveness of each combination, the model produces a statistically defensible near optimum allocation of available resources to computer fraud detection.

Also provided are the results of the research into reported incidents of computer fraud in the form of a Typology. This Typology combines frequency of occurrence and dollar impact of reported cases of fraud into a measure of vulnerability for various types of fraud and perpetrator.

Finally, an overview of investigative techniques and automated tools for evaluating the propriety of computer systems is provided.
CHAPTER 1

INTRODUCTION

This dissertation proposes a methodology for optimizing resources in detecting computer fraud in vulnerable computer systems. Vulnerability is measured in terms of the frequency of reported cases and their significance, as measured in monetary losses.

Among the objectives of the research for this dissertation is the desire to achieve a proper perspective of the elements or parameters involved in computer fraud. The need to develop this perspective was emphasized early in the research when a computerized legal data base was searched for computer fraud cases. The search included all cases with the words "fraud" and "computer" found in the narrative description of the cases contained in the data base. The resulting extract provided numerous cases meeting these criteria. However, the terms "fraud" and "computer" were found to have widely varying meanings with little consistency of use from one case to another.

For example, in one case the defendant was accused of fraudulently obtaining money from persons looking for dates or marriages by inducing them to use the facilities of a "computer matching institute" without having the intent or capability of performance. In another, a computer manufacturer was charged with breach of computer warranty.
Clearly, the term "computer fraud" may cause a problem in communicating, given such diverse usage.

The word "fraud" is often used in conjunction with another word or descriptive term which attempts to define the fraud by its most readily identifiable and distinguishable characteristic. The following examples are typical: art fraud, bank loan fraud, bankruptcy fraud, check fraud, commodity fraud, consumer fraud, contract fraud, credit-card fraud, disbursement fraud, employee fraud, insurance fraud, inventory fraud, mail fraud, payroll fraud, pension fraud, securities fraud, tax fraud and wire fraud.

When used with one of the above, "fraud" takes on a fairly specific meaning. Unfortunately, this is not the case when "fraud" is used with the word "computer". For instance, "computer fraud" might indicate any of the above types of fraud with computer involvement. Further, the computer may be primarily involved in the perpetration of the fraud or may be only incidentally involved.

Generally, "fraud" refers to a deception or form of trickery perpetrated in order to secure unlawful gain where the perpetrator's gain is the victim's loss. For purposes of this dissertation, "computer fraud" will refer to any perpetration of fraud wherein the computer is actively and significantly involved. The computer will be considered actively and significantly involved when input data or data files are tampered with or when computer operations, programs or equipment are manipulated in order to perpetrate the fraud. The computer is not actively and significantly involved simply because
fraudulent data are processed through the computer. Following these guidelines, if a person fraudulently obtains a bank loan by overstating his or her income on a loan application, the fact that the computer is used in processing the loan is only incidental and the perpetration is not considered computer fraud. On the other hand, if the same person modified the bank's personal data files or manipulated input data in order to overstate his (her) net wealth and obtain the loan, it would be considered computer fraud.

The following are several cases fitting the above guidelines which have been reported in the literature (Parker 1976, Whiteside 1978, Leibhotz and Wilson 1974).

CASE 1. PHONY MICR DEPOSIT SLIPS, WASHINGTON, D.C.

A depositor exchanged blank deposit slips on the counter in the bank with his own magnetically coded deposit slips, giving his own account number. Normally processed by machine, the deposit slips were not verified by the bank as to name and address of the depositor. He accumulated $250,000 in four days from other people's deposits. He then withdrew $100,000 and disappeared.

CASE 2. ALTERING BANK RECORDS, MINNEAPOLIS

A programmer altered the bank's demand-deposit accounting program to ignore overdrafts of $1357 before he was caught by manual accounting, only when the computer failed. He made restitution and received a suspended sentence.

CASE 3. EMBEZZLEMENT, CALIFORNIA

A chief accountant embezzled $1 million from his employer over six years. He used the company's computer to financially model his company. The model gauged which changes in accounts receivable and payable would remain undetected in auditing, then performed the indicated alterations. He was convicted and given a ten-year prison term.
CASE 4. POPULATION REGISTRY DATA THEFT, SWEDEN

Two employees borrowed tapes of population registry and copied them using another computer. They sold the copies at reduced prices to their employer's customers. Both employees were convicted and given a six-month jail sentence.

CASE 5. CONSPIRED BANK EMBEZZLEMENT, NEW JERSEY

The computer systems vice-president, senior computer operator, and three non-employees of a bank were charged with transferring money from infrequently used savings accounts to newly opened accounts. They were detected when conversion to a new computer disrupted their work.

CASE 6. COMPUTER SERVICE THEFT, DETROIT

Two engineers accidentally discovered that a password one digit different than their own happened to belong to the president of the time-sharing firm. The president's password allowed access to privileged customer and accounting data. It allowed the engineers to use unlimited amounts of computer time and obtain customer information and proprietary program listings. Discovery was made by computer operators who noticed the password being used at unusual times. The engineers were fired and no other action taken.

CASE 7. PROGRAM THEFT, CALIFORNIA

A programmer stole a copy of a program from the computer file of his employer's competitor through telephone circuits, using a remote job entry terminal. Criminal charges of theft of a trade secret and a civil suit resulted in conviction and a judgment for $300,000 damages for the plaintiff.

CASE 8. CUSTOMER LIST THEFT, WHITE PLAINS, NEW YORK

An employee tried to sell listings of new customers to an outside buyer. He was caught when a potential buyer reported the offer to the police. The list was estimated to be worth $37,000 in the address list market. The case was dropped for lack of evidence.

CASE 9. INDUSTRIAL ESPIONAGE, WEST GERMANY

A secret agent for East Germany is alleged to have copied
confidential financial and production data of 3,000 West German firms onto tape and given the tape to the government of East Germany. The outcome of the case is not known.

CASE 10. SALARY ROUND-DOWN, FRANCE

An employee was authorized to round salaries down to two decimal places, but when he did so he accrued the remainder amounts to his own salary. The total salaries, of course balanced correctly. The disposition of the case is not known.

CASE 11. OVERSTATED ACCOUNTS PAYABLE

An account clerk at a catering service, in collusion with a grocery store owner submitted false account numbers and invoices for undelivered food to a computer system. Thefts amounted to $120,000 over eight years. Both conspirators were convicted.

CASE 12. COLLECTION AGENCY REBILLING, TEXAS

A computerized collection agency sent new bills to people who had paid their bills the previous year. They relied for profit on the reluctance of most people to fight computerized systems. Disposition of the case is not known.

CASE 13. PENSION FRAUD, CANADA

An employee in an insurance company changed the account numbers of several deceased insured persons to his own to collect their pensions. He was caught when a staple in a punch card forced manual handling, which revealed several cards with the same number.

CASE 14. PAYROLL FRAUD, WEST GERMANY

An EDP operator pressed the "repeat" button on the printer to print 200 extra copies of his own paycheck. He was caught when he cashed 37 checks all at the same bank.

CASE 15. SALES COMMISSION FRAUD, ENGLAND

A programmer in a mail-order company created a sales commission account in the name of Zwana to be the last in order. He adjusted the commission program to collect
commission round-downs in the last account. He was discovered after three years when Marketing happened to choose the first and last accounts for a public relations project.

CASE 16. THEFT OF COMPUTER TIME, LOS ANGELES

The manager of EDP and part of his staff were using the firm's computer to analyze racehorse handicaps, making several thousand dollars each week. Case disposition is not known.

CASE 17. INFORMATION THEFT, CALIFORNIA

A Student copied 500 passwords from the system file by using a text editor program to gain access to a presumably protected file. The password file is now kept in scrambled form; sanctions were privately imposed on the student.

CASE 18. THEFT OF PATENTABLE PROCESSES

Two employees scheduled for layoff took program listings describing secret processes to be patented. One employee was fired, the other died of a heart attack.

CASE 19. PHONY WELFARE GRANTS, CALIFORNIA

In Los Angeles County, welfare grants are paid from vouchers based on punch cards. Someone put extra cards in the computer to produce authorized grants. No suspects were identified.

CASE 20. THEFT OF COMPUTER SERVICES, TEXAS

A high school student found a privileged password of the computer service's analyst on a listing in a wastebasket. He also obtained detailed specifications of the system—presumably by merely asking for them. He used large amounts of computer time, played computer games and obtained other customer's data. He was discovered when a computer operator noticed scratch tapes being read before being written. Restitution was made.

CASE 21. THEFT OF COMPUTER SERVICES, CALIFORNIA

High school students were allowed to use free computer terminal services on one project. The computer system employee involved subsequently left. The students continued to use the
services, using new passwords they found. They used $3,000 worth of services before being caught. Now a "poaching bit" is set to alert operators to suspected account activity. Sanctions were imposed privately on the students.

**CASE 22. BANK EMBEZZLEMENT, NEW YORK**

A bank teller manipulated hundreds of accounts through his teller terminal into the computer system. The details of manipulation are omitted. He was caught when a raid on a bookie showed large bets placed by the suspect. He was prosecuted for embezzling $1.5 million by the Manhattan District Attorney's Office.

**CASE 23. DIVIDENDS FRAUD**

A clerk adjusted a computer program used to prepare dividends, thereby generating dividend checks to former shareholders, but addressed to an accomplice. The program then erased records of the check. The clerk was convicted for embezzling $33,000.

**CASE 24. STOLEN PROGRAMS**

A programmer in Texas stole $5 million worth of programs he was maintaining and tried to sell them to one of his employer's customers. He served five years in prison on charges of Grand Theft.

**CASE 25. EQUITY FUNDING SCANDAL**

Phony life insurance policies were created and integrated with records of actual policies, then the mixture was resold to firms who bought the policies with the idea of collecting further premiums. Equity purportedly created 56,000 insurance policies amounting to some $2 billion out of a total 91,000 policies reportedly worth $6.5 billion. To prevent discovery, computer personnel had to perpetuate the deceit by manipulating records so as to show changes in the phony policies, such as a reasonable number of lapses, cancellations, and deaths. The computer program also had to be designed to conceal the fictitious business from auditors and state insurance examiners.

In further efforts to outwit auditors and examiners, the firm installed electronic surveillance equipment in various
rooms so that conversations about verification plans could be overheard.

The conspiracy fell apart in March of 1973, after a former employee of Equity reported details of it to the New York State Insurance Department and to a Wall Street insurance analyst who, in turn notified some of his commercial clients as well as New York Stock Exchange Officials.

Federal indictments were brought against 22 people by a Federal Grand Jury on 105 criminal counts.

Information on computer fraud is difficult to gather because cases are often not classified as such. Further, there is hesitance in organizations, particularly those which thrive on the confidence of customers and stockholders, to disclose their vulnerability to computer fraud (Randall 1978, Wong 1977, p. 60, Parker 1976, Alderman 1977, Allen 1971).

Additional limitations on obtaining computer fraud information result from the lack of federal laws covering this type of crime (Parker 1979, Schultz 1979, Boockholdt and Horvitz 1978). Future legislation, in conjunction with continued probing by researchers, will hopefully improve the availability of information on computer fraud over the next few years.

Wagner (1979), finding that library research did not reveal any "rich" reference resource -- bibliography, reference book, textbook, documentary, or trade or professional publication --for the citation of computer fraud cases, designed a survey to develop a network of information resources for such material. He thought the following resources would offer the greatest potential for computer fraud cases information:

Selected accounting firms
Selected consultants, educators and researchers
Selected boards of public accountancy
State societies of Certified Public Accountants
Selected business and electronic data processing periodicals
Selected information centers and regulatory agencies
Selected business, commercial, industry, professional, and trade organizations/associations
Selected computer vendors
Selected insurance companies
State officials having supervision of insurance activities

Selection of survey recipients was made by Wagner on a judgmental basis since there was no way of determining the appropriate universe from which to draw a sample. Wagner received 132 replies of 371 survey recipients, for a 35 percent response rate. Even though the overall 35 percent response rate indicated considerable interest in the subject matter, the findings of the survey were, in general, disappointing.

According to Wagner (1979, p. 53):

It was particularly disheartening to learn that none of the respondent CPA state boards and state societies maintained files on computer fraud cases. Seventy percent of the respondents overall had no such files. Only seven of 132 respondents --perhaps this is really good news -- had files which included "first hand (raw) data". One of these was a CPA assigned to help unravel, after the fact, the Equity Funding "fraud" maze. Two were consultants who headed up computer software firms. Another was Donn B. Parker of Stanford Research Institute. An EDP security consultant, who had actually been a former fraud perpetrator, indicated he would release such information only for a fee. Also among the first-hand (raw) data held by respondents were two apparently new computer fraud cases.

Several respondents were reluctant to release information which they had available. Only thirteen respondents indicated they would grant free and open access to any computer fraud materials. Although three insurance companies offering surety, fidelity, and professional liability insurance responded, none offered free access to relevant records.

Even more disturbing are the following statistics reprinted from
Security World (Becker 1978) in a recent U.S. Department of Justice publication (1980): "One percent of all computer crimes is detected—approximately 7 percent of the crimes that are detected are reported to the police—of those brought for prosecution, only 1 out of 33 results in a jail sentence—conclusion; 1 out of every 22,000 computer criminals is going to jail."

As a result of the numerous limitations on information available in the area of computer fraud, the extent of this type of crime is not really known. It has been estimated that only 15 percent of known crimes are recorded (Parker 1976). This is deduced by surveys of samples of the general public, counting the number of people who have been victims of crime. These statistics are then compared with police reports of crimes that are compiled by the FBI and published annually as the FBI Uniform Crime Report. Most of the crimes reported in this manner are of the more violent type, such as robberies, auto theft, and rape; far less is known about white-collar crime and even less is known about computer fraud.

H. Jeffrey Bayless, Chief Deputy District Attorney for Denver, estimates that only about 5 percent of computer crimes committed in the U.S. are ever reported because banks, insurance companies and other institutions would rather cover their losses than risk embarrassing publicity.

Although there is no way to accurately estimate the extent of computer fraud, most experts consider it an increasingly serious problem. In one study, Parker (1975) noted that "in 42 computer-
related bank frauds and embezzlements in the period 1962 to 1975, the average loss per case is $430,000 (total $18 million, range $200 to $6.8 million)."

Leonard I. Krauss, who is a computer security consultant with Ernst & Ernst estimates that half a billion dollars a year is stolen because of computer fraud. Loss will probably exceed a billion dollars in the near future (Randall 1978).

The rapid expansion of minicomputers may give many more people an opportunity to perpetrate frauds. Timothy B. Braithwaite, a systems security manager at the Defense Computer Institute agreed in his recent statement that "distributed processing, micro and minis, remote terminals and integrated file structures all place new complex security demands on ADP organizations that can barely secure traditional batch operation" (Randall 1978).

A problem which, according to August Bequai (1978), may be far more serious than the monetary losses is an antiquated and overbureaucratized legal apparatus for dealing with computer crime. He sees our very form of government as being at stake if we fail to adapt our legal system to the ever-growing computer technology. Detection is, of course, a necessary part of this adaptation.

Based on the research for this dissertation, it is evident that there is considerable concern regarding computer fraud. Several researchers have attempted to determine the extent of computer fraud and categorize it for analysis, (for example: Parker 1975, Parker 1976, Krauss and MacGahan 1979, Allen 1977, Allen 1979, Comptroller
General of the United States 1976, and Auerbach 1978/79). The most extensive and most often referenced database on computer fraud is Donn Parker's (Wagner 1979).


Many of the literature citations above provide guidance to auditors, managers and computer specialists regarding internal controls, audit techniques, physical security and related procedures. Some of the authors refer to the magnitude of particular types of computer fraud while others base their guidance on the frequency of occurrence. Clearly, adherence to the advice offered by these authors would greatly reduce computer fraud. Unfortunately, most prevention and detection techniques are labor intensive and many require expertise and human resources not available to most institutions.

The methodology for detecting computer fraud proposed in this dissertation is based on the premise that expertise and human resources are scarce resources that will continue to be inadequate to fully implement labor intensive prevention and detection techniques.

**Summary of Contributions**

A considerable amount of literature was reviewed in preparation for and development of this dissertation. The objective of this
research was to identify and evaluate existing data and methodologies in computer fraud detection and, then, to add significantly to the literature on this subject.

The major contributions of this dissertation which are either unique or over and above existing literature on computer fraud detection are summarized below.

Vulnerability Formula

The literature reviewed primarily looked at past occurrences of computer fraud in terms of their dollar magnitude or frequency of occurrence. The Formula Vulnerability \( V = F \times I \) introduced in Chapter Two combines these two factors and computes a vulnerability value for major cases reported in the literature. These values are then used in identifying combinations of computer systems and perpetrator types most vulnerable to fraud. The literature search did not reveal a comparable treatment.

Computer Fraud Detection Model

A model is introduced in Chapter Three which addresses the relationship between Computer Fraud Detection and the closely related subjects of Computer Fraud Deterrence and Computer Fraud Prevention. Formal treatment of these relationships is an extension of several informal references found during the research.

Threat Analysis

The Threat Analysis in Chapter Three, utilizing the Threat Matrix and Vulnerability Formula cited above, is an expansion of
several sources from the literature. As presented, it represents a unique methodology of analyzing Computer Fraud Threats based on reported cases.

Computer Fraud Detection - A Resource Optimization Problem

The formal treatment of fraud as a resource optimization problem in this dissertation is unique in contrast to other methodologies found in the literature.

Detection Quotient

The Detection Quotient presented in Chapter Four is unique to this dissertation. This quotient was developed specifically to facilitate the treatment of Computer Fraud Detection as a Resource Optimization Problem.

Specific Risk Assessment Methodology

The methodology for conducting Risk Assessment for specific systems presented in Chapter Five, built on the "Delphi" and "Churchman-Ackoff" techniques, in conjunction with the Matrix approach presented in the dissertation forms a unique approach.

Controls Analysis

The formal treatment of computer Fraud Controls as an extension to the Threat and Risk Assessment methodologies discussed above forms a unique conjunctive treatment of Controls and Threats to quantify Post-Control Threat Values.
Combinatorial Dilemma

The solution to the Combinatorial Dilemma presented in Chapter Six forms a unique methodology for reaching a quantifiably definable near optimum alternative where the phenomenally large number of possible alternatives precludes total analysis.

Resource Optimization Model

The model in Chapter Seven developed for this dissertation is a unique methodology for optimizing the utilization of resources in the detection of computer fraud.
CHAPTER 2

THE TYPOLOGY

The purpose of this typology is to identify, through manipulation schemes which have been reported to date, computer systems or situations which are vulnerable to computer fraud based on both frequency of occurrence and dollar impact. For this methodology, the formula for vulnerability is:

\[ V = F \times I \]

where \( F \) = relative frequency of occurrence and \( I \) = dollar cost of occurrence

It should be noted that, due to the limitations on information available in computer fraud, it is not possible to develop an exhaustive typology. While it is probably reasonable to assume that a large number of undetected or unreported computer frauds simply follow the patterns found in those which have been detected and reported, this assumption cannot be substantiated. Thus, there may be undetected computer frauds which could affect the outcome of the typology.

The typology is based on "best available" data. However, it should be viewed from the standpoint of the above limitations. Hopefully, information on the subject of computer fraud will improve with future legislation and research. The typology is based upon cases which have been reported through some form of media or discovered through various investigative efforts. Nearly all researchers or authors now writing on the subject
of computer fraud cite the work of Donn B. Parker at the Stanford Research Institute (SRI) (Wagner 1979).

SRI involvement in computer abuse began in the mid 1960's. Information gathering since that time by Parker has resulted in an extensive database, including hundreds of cases of computer abuse, a subset of which is computer fraud. The research effort being conducted by Mr. Parker and his associates at SRI is probably the only one not restricted to specifically defined boundaries such as country, state, local governmental jurisdiction, for-profit organization, not-for-profit organization, industry, profession or discipline. Computer fraud detection, while not the main thrust of Parker's work, is certainly an integral part of the broader topic of computer security.

Although it is generally desirable for a researcher to go to the raw data rather than use secondary sources, it is just not possible to establish a database of computer fraud incidents to rival Parker's. Thus, his database along with a rather extensive collection of governmental computer fraud cases by the General Accounting Office will provide the primary sources of data for this typology.

There are numerous ways of categorizing or classifying computer fraud cases. A few of the more common are by type of organization victimized, by type of computer system, by dollar magnitude and by perpetrator type or position. Many discussions have appeared in the literature which categorize computer fraud in one way or another. (Parker 1976, Allen 1977, Krauss and MacGrahman 1979, and GAO Report FGMSD-76-27, 1976).
Allen (1977) analyzed most of the publicly documented fraud cases detected at the time, focusing on 150 major cases contained in Parker's data base at SRI. At approximately the same time, the General Accounting Office (GAO) was analyzing computer-related crimes in federal programs (GAO 1977).

A categorization of Allen's cases, consolidated with federal government cases from GAO's survey, is presented in Figure 1. Examination of Figure 1 indicates that the most costly fraud type is corporate accounting and inventory control frauds with average losses of $1.3 million; second is payments to other individuals at the state and local government level with average losses of $487 thousand; third are corporate payment to creditor frauds with average losses of $324 thousand and so on.

Further analysis of Figure 1 indicates that, in terms of total average losses, corporations are first with average losses of $621 thousand; state and local governments are second with average loss totals of $329 thousand; banks/savings and loan companies third with $193 thousand; and the federal government is fourth with average loss totals of $45 thousand.

Many such comparisons can be made from Figure 1 and various conclusions may be drawn. Recall, however, that the methodology for this dissertation analyzes computer fraud in terms of vulnerability with the formula for vulnerability expressed as: Vulnerability (V) = Frequency (F) times Impact (I).

In Figure 2, relative frequencies of occurrence (F) were derived by dividing the y values in Figure 1 (total cases in each cate-
### Average Losses in Computer Frauds ($000s)

<table>
<thead>
<tr>
<th>Average Losses</th>
<th>Corporation</th>
<th>Banks/ Savings and Loan</th>
<th>State and Local Government</th>
<th>Federal Government</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payments to employees</td>
<td>$139(4/8)</td>
<td>$3(1/1)</td>
<td>$14(3/4)</td>
<td></td>
</tr>
<tr>
<td>Payments to other individuals</td>
<td>133(2/4)</td>
<td>-</td>
<td>-</td>
<td>487(6/9)</td>
</tr>
<tr>
<td>Payments to creditors</td>
<td>324(5/5)</td>
<td>252(8/12)**</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Accounting/inventory control</td>
<td>1,300(10/10)</td>
<td>195(10/12)</td>
<td>(-/1)*</td>
<td>56(25/28)</td>
</tr>
<tr>
<td>Collections/deposits</td>
<td>43(2/6)</td>
<td>157(8/9)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Billings</td>
<td>6(2/6)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>(-/2)*</td>
<td>-</td>
<td>(-/2)*</td>
<td>-</td>
</tr>
<tr>
<td><strong>Average Loss</strong></td>
<td><strong>$621(25/41)</strong></td>
<td><strong>$193(27/34)</strong></td>
<td><strong>$329(9/16)</strong></td>
<td><strong>$45(47/55)</strong></td>
</tr>
</tbody>
</table>

* Amount of loss unknown

** One case of $6.8 million deleted from figures to avoid distortion

*** GAO cases were categorized simply as "Fraudulent Direct Payments"

**Note:** The average loss figure is based upon x cases out of y total cases in that category where (xy) is shown just to the right of the average. Losses in some cases were unavailable or eliminated for other reasons.

Figure 1. Computer Fraud Cases by Type of Scheme and Victim Organization.

...
<table>
<thead>
<tr>
<th>Type of Fraud</th>
<th>Corporation</th>
<th>State and Local Government</th>
<th>Federal Government</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payments to employees</td>
<td>$139(.0548)</td>
<td>$3(.0068)</td>
<td>$14(.0274)</td>
</tr>
<tr>
<td>Payments to other individuals</td>
<td>133(.0274)</td>
<td>487(.0616)</td>
<td></td>
</tr>
<tr>
<td>Payments to creditors</td>
<td>324(.0342)</td>
<td>252(.0822)**</td>
<td>- -</td>
</tr>
<tr>
<td>Accounting/inventory control</td>
<td>1,300(.0685)</td>
<td>195(.0822)</td>
<td>-/1(.0068)* 56(.1918)</td>
</tr>
<tr>
<td>Collections/deposits</td>
<td>43(.0411)</td>
<td>157(.0616)</td>
<td>- -</td>
</tr>
<tr>
<td>Billings</td>
<td>6(.0411)</td>
<td>- -</td>
<td>- -</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td><em>/2(.0137)</em></td>
<td><em>/2(.0137)</em></td>
<td>- -</td>
</tr>
<tr>
<td>Average loss</td>
<td>$621(.2808)</td>
<td>$192(.2329)</td>
<td>$329(.1096) $45(.3767)</td>
</tr>
</tbody>
</table>

* Amount of loss unknown

** One case of $6.8 million deleted from figures to avoid distortion

*** GAO cases were categorized simply as "Fraudulent direct payments"

Note: Figures in parentheses are relative frequencies of occurrence \( F \) which were derived by dividing the \( y \) values in Figure 1 (total cases in each category) by 146 (total cases in all categories).

Figure 2. Relative Frequency of Occurrence and Average Loss of Computer Fraud by Type of Scheme and Victim Organization

parentheses beside the average loss values. The average loss values represent the "I" values. The vulnerability for each category is
computed by using the formula $V = FI$. The resulting values for "$V" are shown in Figure 3.

<table>
<thead>
<tr>
<th>Type of Fraud</th>
<th>Corporation</th>
<th>Banks/Savings and Loans</th>
<th>State and Local Government</th>
<th>Federal Government</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payments to employees</td>
<td>$ 7.62</td>
<td>$ .02</td>
<td>$ .38</td>
<td></td>
</tr>
<tr>
<td>Payments to other individuals</td>
<td>3.64</td>
<td>--</td>
<td>29.99</td>
<td>$ 6.10</td>
</tr>
<tr>
<td>Payments to creditors</td>
<td>11.08</td>
<td>20.71</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Accounting/inventory control</td>
<td>89.05</td>
<td>16.03</td>
<td>--</td>
<td>10.74</td>
</tr>
<tr>
<td>Collections/deposits</td>
<td>1.77</td>
<td>9.67</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Billings</td>
<td>.25</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Totals</td>
<td>$ 174.38</td>
<td>$ 44.95</td>
<td>$ 36.06</td>
<td>$ 16.95</td>
</tr>
</tbody>
</table>

Figure 3. Vulnerability by Type of Computer System and Organization

It should be noted that the values in Figure 3 are not "expected values" but rather indicators of relative vulnerabilities of different types or computer systems. Recall that the formula for vulnerability is based on "frequencies" of reported fraud rather than "probabilities", which would be required to compute expected values.

Given the fact that computer fraud is a serious problem, the primary concern at this point is to identify systems which are most
vulnerable and emphasize those in detection procedures. From this point on, when referring to the "V" values from Figure 3, the "000's" and dollar signs will be dropped. Thus, instead of referring to a "V" value of $89,050 for "corporate accounting and inventory control fraud", it will be referred to as $V = 89.05.

It is readily apparent that in terms of vulnerability as defined here, "corporate accounting and inventory control fraud" is the most significant with $V = 89.05. Examining Figure 1 again, this same category is also most significant in terms of absolute dollars with average losses of $1.3 million. This relationship also holds true for our second highest "V" value, "state and local government -- payments to other individuals" with $V = 29.99. Figure 1 illustrates that this same category is second highest in terms of absolute dollars with average losses of $487,000.

However, the relationship does not hold for the third highest "V" value, "banks/savings and loan -- payments to creditors" with $V = 20.71. Figure 1 shows the third highest category in terms of absolute dollars is "corporate payments to creditors" with average losses of $324,000. Neither does the relationship hold for computer fraud by type of organization. Notice that the aggregate "V" value for banks/savings and loan companies of 44.95 is clearly second highest in Figure 3. In terms of absolute dollars, however, state and local governments are clearly second highest with average losses of $329,000.

When viewed in terms of their vulnerabilities as defined in this dissertation, the relative significance and the resulting
prioritization of computer fraud types varies from a ranking based strictly on average dollar losses.

A listing of the computer fraud types from Figure 3 is presented again in Figure 4, but in descending order of vulnerability. It is apparent from Figure 4 that in terms of computer systems categorized by both type of system and organization, corporate accounting/inventory control fraud leads in vulnerability by a comfortable margin with $V=89.05$, compared to its closest competitor -- payments to other individuals by state and local governments with a $V=29.99$. It is interesting to note that the "$V$" value for corporate accounting/inventory control systems is 4,452 times larger than the "$V$" value for payments to employees by banks/savings and loan.

Many such comparisons may be made and they can be quite valuable in prioritizing or ranking specific types of computer fraud by vulnerability and need for emphasis. Further summarization is also possible due to the many similarities between the types of fraud in Figure 4. For example, there should be many similarities between "payments to creditors - bank/savings and loan" and "payments to creditors - corporations". Likewise, there should be many similarities between "accounting/inventory control - corporation" and "accounting and inventory control - federal government".

By following this logic we see that there are four broad categories of computer fraud in Figure 4: accounting/inventory control; payments; collections/deposits; and billings frauds.
<table>
<thead>
<tr>
<th>Type of Fraud</th>
<th>&quot;V&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Accounting/inventory control - corporation</td>
<td>89.05</td>
</tr>
<tr>
<td>2. Payments to other individuals - state and local gov't</td>
<td>29.99</td>
</tr>
<tr>
<td>3. Payments to creditors - bank/savings</td>
<td>20.71</td>
</tr>
<tr>
<td>4. Accounting/inventory control - bank/savings and loan</td>
<td>16.03</td>
</tr>
<tr>
<td>5. Payments to creditors - corporations</td>
<td>11.08</td>
</tr>
<tr>
<td>6. Accounting/inventory control - federal government</td>
<td>10.74</td>
</tr>
<tr>
<td>7. Collections/deposits - bank/savings and loan</td>
<td>9.67</td>
</tr>
<tr>
<td>8. Payments to employees - corporations</td>
<td>7.62</td>
</tr>
<tr>
<td>9. Payments (all) - federal government</td>
<td>6.10</td>
</tr>
<tr>
<td>10. Payments to other individuals - corporations</td>
<td>3.64</td>
</tr>
<tr>
<td>11. Collections/deposits - corporations</td>
<td>1.77</td>
</tr>
<tr>
<td>12. Payments to employees - state and local government</td>
<td>.38</td>
</tr>
<tr>
<td>13. Billings - corporations</td>
<td>.25</td>
</tr>
<tr>
<td>14. Payments to employees - bank/savings and loan</td>
<td>.02</td>
</tr>
<tr>
<td>TOTAL</td>
<td>207.05</td>
</tr>
</tbody>
</table>

Figure 4. Types of Computer Fraud in Descending Order of Vulnerability

By adding the "V" values from Figure 4, we derive the following cumulative vulnerabilities (CV) for the above categories:

<table>
<thead>
<tr>
<th>Category</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting/inventory control</td>
<td>115.82</td>
</tr>
<tr>
<td>Payments</td>
<td>79.54</td>
</tr>
<tr>
<td>Collections/deposits</td>
<td>11.44</td>
</tr>
<tr>
<td>Billings</td>
<td>.25</td>
</tr>
<tr>
<td>TOTAL</td>
<td>207.05</td>
</tr>
</tbody>
</table>
The information contained in this typology is designed to provide insight into the basic vulnerabilities of different systems to computer fraud. By incorporating both the frequency of occurrence and dollar impact into one measure of vulnerability, the typology, within the limitations discussed earlier, should provide a guideline to follow in determining whether or not there is cause for concern for a particular type of computer system.

The cases in the typology will also be used in Chapter 3 to provide insight into the specific threats surrounding those systems which are vulnerable to computer fraud.
CHAPTER 3

THREAT ASSESSMENT

Computer fraud detection would be relatively simple if we could track every transaction or change through automated systems and observe the impact and propriety at each step. This level of surveillance should, in fact, eliminate most computer fraud since very few potential perpetrators would be likely to tackle the high odds of getting caught in such an environment.

However, it is just not feasible to implement this methodology in most systems. Many systems today process hundreds of thousands or millions of transactions per month. Given even a minimal effort to fully track a transaction or change through large, often integrated systems, it would take huge cadres of people and massive amounts of money to implement such a program. Add real-time or near real-time processing and the feasibility of the approach is doubtful because of time criticality even with unlimited human and dollar resources.

The purpose of this chapter is to present a methodology which, recognizing time, people and dollar constraints, identifies threats to automated systems and ranks them, allowing efficient utilization of limited resources in the detection of computer fraud.

The typology in Chapter 2 identified and ranked, by vulnerability, automated systems based on reported incidents of computer fraud, in order that emphasis could be directed to these
systems, thus establishing the first level of prioritization and ranking among systems. The risk assessment in this chapter goes the next step in identifying and ranking specific threats within these systems.

**Constraints**

Like many other management decisions, computer fraud detection becomes a matter of resource allocation in an environment of limited time, people and dollar resources. Typically, as indicated above, there is not enough time, people or dollars in many of today's systems to fully monitor every transaction or change. However, the resource allocation responsibility goes beyond this. Even if resources are plentiful, it is incumbent upon management to utilize them in a prudent manner. Thus, it is not good management to spend $10,000 to preclude a probable loss of $1,000 by computer fraud unless some other factor more than offsets the $9,000 difference. (It is conceivable that other factors such as embarrassment or loss of confidence would more than offset the $9,000 in this example.)

**Time Criticality**

Time criticality, or the time constraint, is typically set to a large extent by the system or conditions surrounding it. Contrast, for example, the classic payroll system and a near real-time banking system. In the payroll system, transactions are usually entered all during the pay period; commonly a weekly, bi-weekly or monthly period. To a large extent, by relaxing the constraints on people and dollars, time periods in such a system would be adequate to track all
transactions and changes between pay periods although the costs might be prohibitive. In the banking system, on the other hand, the time between a deposit and authorized withdrawal against that deposit may be very short, perhaps minutes or seconds. Given that thousands or tens of thousands of transactions may occur in the brief period of a few hours in a large banking system, it may be virtually impossible to track all transactions or changes even with unlimited people and dollar resources due to the difficulties of administering such a program.

Prevailing conditions may also determine or affect the time criticality of system. For example, in certain Latin American countries where inflation rates have soared well beyond 100 percent, employees have demanded their pay on a daily basis in order to spend it as soon as possible on commodities as a hedge against inflation (Fitzgerald 1980). Time criticality in this environment would be considerably different than in the classic payroll example cited above.

Human and Dollar Constraints

Human and dollar constraints will be considered together since, for the most part, management may elect to utilize more or less of either resource for various activities, one of which is computer fraud detection. There are obvious constraints in terms of total people or dollars available but, within limits, management is free to adjust levels of either resource.

There are various factors which might influence the levels of people or dollar resources devoted to the detection of computer fraud. For example, management may decide that a certain number of people or
dollars will be dedicated to computer fraud detection based on their intuitive judgment regarding the honesty of people. Thus, one manager may require that two percent of total systems costs be expended in computer fraud detection. Another, with infinite trust in people, may not apply any resources to detection. Still another may have an inherent distrust in people but feel elaborate controls built into systems under his or her control provide adequate protection to preclude fraud and, thus, may not apply further resources to fraud detection.

A more sophisticated approach would be to develop, through some means, an estimated value of loss through computer fraud and dedicate corresponding resources to its detection. For example, a manager might estimate annual losses of $50,000 through computer fraud if no attempt is made at detection. Then, assuming relative linearity between resources applied to detection and decreases in losses through computer fraud, be willing to spend up to $50,000 for detection. If this same manager worked for a bank or some other institution which depends heavily upon public trust and confidence, he or she might be willing to spend considerably more than the estimated loss value to avoid the embarrassment and loss of confidence associated with an inability to uncover computer fraud.

If management is totally unyielding in the level of human and dollar resources which may be applied to computer fraud detection, then these constraints essentially become fixed constraints. More likely, however, management will be willing to adjust the level of
these resources if it can be shown that such a move is cost effective or vital to the firm's existence.

The threat assessment provides insight into the amount of resources which should be used for computer fraud detection, given management's basic aversion to being defrauded. Further, the threat assessment provides more specific guidance in the distribution of these resources to the various threats surrounding a system.

The Typology Revisited

Recall that the typology in Chapter 2 analyzed reported incidents of computer fraud to establish relative vulnerabilities of various types of computer systems to computer fraud. The cases in the typology were primarily from Parker's database at Stanford Research Institute as reported by Allen (1977) and the files of the General Accounting Office (GAO 1977).

In Allen's 1977 study which included 150 computer fraud cases, the cases were categorized by method of computer manipulation as depicted in Figure 5.

Figure 5 presents Allen's Methods of computer manipulation further categorized by type of victim. In Figure 6 the same 150 cases are presented by method of computer manipulation but are further categorized by the type of system victimized in his 1977 study plus additional cases into the categories shown in Figure 7.

In a more recent study Allen (1979) categorized cases reported in his 1977 study plus additional cases into the categories shown in Figure 7.
<table>
<thead>
<tr>
<th>Method of Computer Manipulation</th>
<th>Type of Victim</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Corp</td>
</tr>
<tr>
<td>Trans. Added</td>
<td>16</td>
</tr>
<tr>
<td>Trans. Altered</td>
<td>8</td>
</tr>
<tr>
<td>Trans. Deleted</td>
<td>3</td>
</tr>
<tr>
<td>File Changes</td>
<td>5</td>
</tr>
<tr>
<td>Program Changes</td>
<td>6</td>
</tr>
<tr>
<td>Improper Operations</td>
<td>4</td>
</tr>
<tr>
<td>Misc., Unknown</td>
<td>4</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>46</td>
</tr>
</tbody>
</table>

Note: Case totals do not add up to 150 because some are classified in more than one category.

Figure 5. Methods of Computer Manipulation By Type of Victim

<table>
<thead>
<tr>
<th>Method of Computer Manipulation</th>
<th>Type of System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Payments to Accounting/</td>
</tr>
<tr>
<td></td>
<td>Employees &amp; Inventory/</td>
</tr>
<tr>
<td></td>
<td>Others</td>
</tr>
<tr>
<td>Transactions Added or Altered</td>
<td>40</td>
</tr>
<tr>
<td>Transaction Deleted</td>
<td>2</td>
</tr>
<tr>
<td>File Changes</td>
<td>6</td>
</tr>
<tr>
<td>Program Changes</td>
<td>2</td>
</tr>
<tr>
<td>Improper Operation</td>
<td>4</td>
</tr>
<tr>
<td>Misc., Unknown</td>
<td>2</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>56</td>
</tr>
</tbody>
</table>

Note: Totals do not add up to 150 because some cases are classified in more than one category.

Figure 6. Methods of Computer Manipulation By Type of System
Allen concluded that the most common method of computer manipulation is the manipulation of transactions. He gives examples of transaction manipulation as (1) changing the batch controls or creating a new batch control document or record; (2) adding to a batch by substituting a fraudulent transaction for a legitimate one; (3) altering an otherwise legitimate transaction or a dummy transaction that had previously been introduced in some other fashion; (4) introducing an entire batch of fraudulent and dummy transactions; (5) deleting transactions by changing batch controls; and (6) altering or deleting through file maintenance.

In another study Krauss and MacGahan (1979) suggest that there are surprisingly few common forms of computer fraud manipulation - in fact, just these three:

Input Transaction Manipulation Schemes
Unauthorized Program Modification Schemes

File Alteration and Substitution Schemes

Referring back to Figure 7, it is apparent that Allen would agree that most computer fraud cases do, in fact, fall into one of these three categories. Krauss and MacGahan present various techniques for invoking the schemes. Abstracts are shown below.

Input Transaction Manipulation Schemes

Extraneous Transactions. Making up extra transactions and getting them processed by the system is a rather straightforward form of input manipulation. A perpetrator may either enter extraneous monetary transactions to benefit himself, or he may enter file maintenance transactions that change the indicative data about a master file entity (customer, vendor, product, general ledger account, salesman, department, etc.) in some way that he will later exploit.

Failure To Enter Transactions. Perpetrators can obtain substantial benefits simply by failing to enter properly authorized transactions. One of the simplest examples involved action on the part of check-processing clerks who simply destroyed their own canceled checks before they were debited to their accounts. The same thing can happen in a customer billing system. File maintenance can also be excluded dishonestly with similar benefits.

Modification of Transactions. Fraudulent gains can be realized by altering the amount of a properly authorized monetary transaction. For example a perpetrator may reduce the amount of charges against a particular account or increase payments into a particular account. Another scheme involves changing indicative data on file maintenance transactions. Examples are name, address, monthly closing date, account type and status, privileges and so on. Since errors in indicative data are fairly common and since controls over such transactions tend to be weak in many companies, this method is particularly promising to the perpetrator. The most insidious of all transaction modification methods involves "exploitation of blanket file maintenance transactions". More specifically, a transaction that instructs the system to change the corresponding master file data element for any and all corresponding fields filled out on the input form. (Best advice is to avoid the use of such transactions).
Misuse of Adjustment Transactions. Misuse of adjustment transactions is a common ingredient in input manipulation schemes. Here the term "adjustment" refers to monetary corrections of past errors or inaccuracies that have come about in a system through physical loss or spoilage of materials. Often, perhaps out of concern to set things straight as quickly as possible, adjustment transactions are processed without adequate control. The result can be computer fraud of massive proportions.

Misuse of Error-Correction Procedures. Millions of dollars have been embezzled by perpetrators under the guise of error corrections. Although many of these abuses are special cases of previously mentioned methods of manipulating input, it is felt that error corrections are often a problem and deserve special attention. Ways that perpetrators abuse error-correction procedures include entering extra error corrections, failure to enter necessary corrections, and modifications of properly authorized corrections.

Program Modification Schemes

Program modification schemes are the most insidious and difficult to detect. Even though the reported instances of such cases is fairly low, leading auditors and security consultants share a chilling view of reported statistics: "reported incidence bears no relation to the actual enormity of the problem".

To explain this commonly held view, consider the following:

Some program modification schemes are untraceable.

All program modification schemes are difficult to detect.

Motivation for perpetrators is high because a single blitz can effect large benefits rapidly with little chance of detection or prosecution.

Larcenous Strategies for Modifying Programs

Breakage. Siphoning off small sums from numerous sources is commonly referred to as breakage. This method is particularly well suited to being implemented via program modification, because a few simple lines of code can bring about repeated theft of a large number of amounts. Breakage can be employed whenever a computation is called for:
Computation of applicable service charge
Computation of discounts
Payroll withholding computations
Computation of retirement benefits
Computation of interest on savings
Computation of welfare, medicare, social security, or unemployment benefits.

In any of these situations, all the perpetrator has to do is to instruct the computer to accumulate amounts resulting from rounding, and possibly small additional amounts, and to allocate the sum of all such amounts to a single account to which he or she has access. This activity will not be readily detected by systems controls because the total amount of money involved will agree with any predetermined control totals. The individuals involved are unlikely to notice a discrepancy in their accounts. Even if they do notice a discrepancy they are unlikely to comment if the amounts involved are small.

Undocumented Transaction Codes. By programming the computer to accept undocumented types of transactions, perpetrators can arrange to receive substantial profits in a very short time. Once having made provisions for processing of the extra transaction type, there are several means of getting the necessary transactions into the system. The transactions may be computer generated, input by the programmer where controls (or lack or controls) allow it, input via the addition of an extra input file - etc.

Balance Manipulation. Simple, undisguised balance manipulation is a method that involves assuming that processing results will not be properly reviewed. A dishonest programmer can modify appropriate programs so that all totals and balances appear to be correct for any given day. The "work factor" involved in modifying all programs involved is typically high so the programmer will more often attack just one or two programs.

Deliberate Misreporting With Lapping. A program that has been manipulated to cause misreporting either fails to apply a charge to a perpetrator's account (the charge gets applied to another account) or credits a perpetrator's account with a payment (the account that should have been credited is not posted). Either way, certain problems are bound to arise. In the first case, complaints can be expected from those whose accounts now carry unauthorized charges. In the second case, complaints can be expected from those whose accounts were not credited. To avoid this a process of deliberate misposting, correcting the deliberate misposting, and creating another deliberate misposting called lapping is used to continue the fraud. All lapping schemes of any merit call for masterful
time management and meticulous record keeping on the part of the perpetrator.

File Modification. Altering programs to effect secret changes in account status is a fairly common programming technique for computer fraud. Examples of account status changes include: opening an account for subsequent fraudulent manipulation in order to receive automatic payments (payroll, retirement, unemployment, welfare, etc.); destroying the record of a fraudulent account; inhibition of printing of an account's pastdue status; increases to a credit limit on a credit account so that a greater charge will be authorized.

Fudging Control Totals. This tactic is often combined with other programming schemes. The approach involves processing that occurs without being properly reflected in control totals.

File Alteration Schemes

Access To A Live Master File. One fairly common form of fraudulent file alteration is to obtain "access to a live master file" and (using either a program specially written for the purpose, a general retrieval program, or a utility) make surreptitious changes to the file. Changes may include modification of monetary amounts or changes to other data.

Substitution Of Dummied-Up Version For The Real File. This scheme depends upon one of two possible sequences of events. In either case, the scheme begins with the perpetrator obtaining access to the master file, possibly under the guise of making a copy for use as test data. Then the file is run against a program, either in-house or at a service bureau. The program creates a very similar file, containing only a few modifications. The newly created file is then substituted for the live file and returned to the data library.

Access And Modification Of Transaction Files Prior To Processing. Possible fraudulent actions that may be involved in this type of scheme include addition, modification, and deletion of input transactions.

The Risk Assessment Methodology

The methodology, by identifying and prioritizing the major threats surrounding automated systems, establishes the framework for optimizing the use of resources in the detection of computer fraud.
The basic model on which the risk assessment methodology is based is shown in Figure 8. Examples of computer fraud deterrents are: fear of imprisonment, high morals and non-lucrative system types. Deterrents do not require the expenditure of resources since they are inherent, if they exist. As the model in Figure 8 shows, if deterrents are adequate to limit the threat of computer fraud to an acceptable level, as determined by threat assessment, there is no need to allocate resources to prevention or detection. If, however, the threat is not acceptable, computer fraud prevention through controls is the next logical check against fraud. The adequacy of prevention is determined through controls analysis. A distinction between existing systems and systems under development is appropriate at this point. If the system is fully developed and operating, particularly if it is a large complex system, it may be very expensive to add system controls. On the other hand, if the system is under development, it may be quite feasible and desirable to add controls based on the controls analysis. In either case, if it is economically feasible to add controls, serious consideration should be given to adding them since prevention is preferable to detection just as deterrence is preferable to prevention. The reason for this is rather obvious. Both controls and detection are expensive and require the expenditure of resources which could be used elsewhere. Thus, if either can be eliminated by avoiding fraud in other ways there will be a gain. If prevention, or the composite of deterrents plus prevention are adequate, then detection is not necessary. If, however, deterrents plus prevention are not adequate, computer fraud detection is needed.
Figure 8. Computer Fraud Detection Model
The model is circular since computer fraud detection becomes a deterrent as defined for the model. Thus, a highly publicized detection capability could decrease or even eliminate computer fraud if it is perceived by would-be perpetrators as being effective.

The model is consistent with time, people, and dollar constraints discussed earlier since it limits the allocation of resources for detection, and for prevention only to those levels considered essential. A feature of the model which may not be quite as obvious at this point is its inherent ability to limit total effort to only that level required for a specific system. This should become clear as "risk assessment" is further defined. Finally, the model avoids the flaw inherent in most presentations on the broader subject of computer security of assuming the largest computer systems and making it difficult or impossible to scale down recommended safeguards for the smaller system. The model is equally applicable to the smallest batch oriented system and the largest integrated, real-time, distributed system.

It is rather common knowledge that many systems, large and small, are fully operational today with few, if any, preventive controls built in. Where controls do exist they are often of the "cookbook" variety which are only partially effective in thwarting the computer felon. In a recent special report (Parker 1979), a clear distinction is drawn between the three basic threats with which the broader subject of computer security must deal: (1) natural disasters, e.g., extreme weather conditions; (2) human errors and omissions, e.g., incorrect tape labeling; and (3) intentional human acts such as fraud
or sabotage. Parker states that the first two threats are empirically predictable and lend themselves to treatment through careful application of the "cookbook approach" of checklist do's and don'ts currently so plentiful in the literature. However, he points out, "cookbook" safeguards are much less effective against intentional acts such as fraud than they are with natural disasters or human errors and omissions.

Finally, in all but the simplest of systems, the overall effectiveness of deterrents and preventive controls may be difficult, if not impossible, to quantify. The research for this dissertation did not reveal any scheme, for example, which could provide some specific level of confidence that fraud would be prevented if certain controls were implemented and followed. There is a significant element of uncertainty involved even with stringent controls in most systems.

Threat Analysis

The purpose of threat analysis is to identify and evaluate the basic threats surrounding systems. The computer fraud manipulation schemes discussed earlier and presented in Figures 5, 6, and 7 provide insight into the types of activities which appear to be most susceptible to fraud. The threat analysis phase combines these manipulation schemes with corresponding perpetrators in the establishment and ranking of systems threats. The analysis uses a matrix approach starting with the blank matrix shown in Figure 9.

Manipulation schemes will be listed across the top of the matrix, to be represented by the vertical columns of the matrix.
Perpetrators of these schemes will be listed down the left side of the matrix and represented by the horizontal rows of the matrix. For the risk assessment in this chapter, the major cases that have been publicized and which were discussed earlier were used in identifying the schemes and perpetrators shown in the threat matrix in Figure 10. The categories in Figure 10 were reported in Auerbach (1978-79), based on the publicized cases of computer fraud. The threat assessment
<table>
<thead>
<tr>
<th>Scheme</th>
<th>Transactions Added</th>
<th>Transactions Altered</th>
<th>Transactions Deleted</th>
<th>File Changes</th>
<th>Program Changes</th>
<th>Improper Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Entry/ Terminal Operator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clerk/Teller</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Officer/Manager</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Operator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Staff</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outsider (Non-employee)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10. Threat Matrix with Scheme and Perpetrator Types

Methodology for specific items in Chapter 5 also starts with the empty matrix in Figure 9, but a small group approach is presented for developing the threat matrix from that point, tailored to specific systems.

The perpetrators shown down the far left column of the matrix should be self-explanatory with two exceptions. The first is the distinction between "Data entry/terminal operators" and "Clerk/teller", the first two entries. This distinction is essentially that the clerks
and tellers deal directly with customers, suppliers, and others, whereas data entry and terminal operators do not.

The other category which might be somewhat vague is the last entry -- "Outsider (non-employee)". The perpetrator is considered an "outsider" if he or she is unknown and could conduct the scheme without specialized access or knowledge.

The next step in the threat analysis is to attempt to produce a ranking of the schemes and perpetrators in order of the relative frequency of occurrence and potential impact similar to the one in the typology for various types of computer systems. For this phase of the analysis, a generalized ranking is developed which is based on publicized case data.

The threat matrix shown in Figure 11 is an expanded version of the one in Figure 10. The schemes and perpetrators remain unchanged, but several items have been added. The far right column entitled "Average Loss ($000's)" has been added to show the average dollar value of computer fraud cases by type of perpetrator. For example, the average loss in fraud cases perpetrated by data entry/terminal operators was $727,000; by clerk/tellers, $58,000; etc.. The whole numbers appearing in the individual cells of the matrix represent the number of occurrences of fraud involving the intersecting schemes and perpetrators. For example, the upper left cell of the matrix contains the number 9 indicating that there were 9 cases perpetrated by data entry/terminal operators using the "transactions added" manipulation scheme. The decimal numbers in parenthesis in each cell were derived by dividing the whole number in the cell by the total
<table>
<thead>
<tr>
<th></th>
<th>Transactions Added</th>
<th>Transactions Altered</th>
<th>Transactions Deleted</th>
<th>File Changes</th>
<th>Program Changes</th>
<th>Improper Operation</th>
<th>Average Loss ($000's)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Entry/</td>
<td>9 (.113)</td>
<td>4 (.05)</td>
<td>1 (.013)</td>
<td></td>
<td></td>
<td>727</td>
<td></td>
</tr>
<tr>
<td>Terminal Operator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clerk/Teller</td>
<td>9 (.113)</td>
<td>6 (.075)</td>
<td>1 (.013)</td>
<td></td>
<td></td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>Programmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14 (.175)</td>
<td>53</td>
</tr>
<tr>
<td>Officer/Manager</td>
<td>8 (.1)</td>
<td>4 (.05)</td>
<td>3 (.038)</td>
<td>1 (.013)</td>
<td>3 (.038)</td>
<td>1 (.013)</td>
<td>314</td>
</tr>
<tr>
<td>Computer Operator</td>
<td>1 (.013)</td>
<td>4 (.05)</td>
<td>1 (.013)</td>
<td></td>
<td>3 (.038)</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>Other Staff</td>
<td>1 (.013)</td>
<td></td>
<td></td>
<td></td>
<td>1 (.013)</td>
<td>1 (.013)</td>
<td>92</td>
</tr>
<tr>
<td>Outsider (Non-</td>
<td>3 (.038)</td>
<td>1 (.013)</td>
<td></td>
<td></td>
<td></td>
<td>696</td>
<td></td>
</tr>
<tr>
<td>Employee)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% of Cases</td>
<td>39%</td>
<td>23.8%</td>
<td>5.1%</td>
<td>6.5%</td>
<td>21.3%</td>
<td>5.1%</td>
<td></td>
</tr>
</tbody>
</table>

Figure 11. Expanded Threat Matrix

number of cases studied. Referring once again to the upper left cell, the decimal (.113) was derived by dividing 9 by 80 (total cases studied). Thus, 11.3 percent of total cases studied were perpetrated by data entry/terminal operators using the "transactions added" manipulation scheme. It should be noted that of the cases publicized,
contained enough descriptive data to categorize them by both perpetrator and manipulation scheme - others were excluded.

The information in the expanded threat matrix in Figure 11 may now be used to calculate threat values for each cell of the matrix (i.e., each scheme/perpetrator combination). This value which will be referred to as simply "threat" or "T" will be computed as follows:

\[ \text{Threat} = \text{Frequency times Impact or } T = FI \]

Where \( F \) = Relative frequency of occurrence and \( I \) = Average dollar impact.

Referring once again to the upper left matrix cell in Figure 11, "T" is thus calculated by multiplying the decimal .113 by $727,000, giving a value of $82,141. Since the dollar indicator and thousands positions are not needed for this analysis, the dollar sign will be dropped and the value rounded, giving a threat value of 82.2 for the computer frauds perpetrated by data entry/terminal operators using the "transactions added" manipulation scheme (upper left matrix cell). Threat values calculated in this manner are shown in Figure 12 for all cells or perpetrator scheme combinations. Blank cells represent perpetrator/scheme combinations which were not reported. The threat values are shown in descending order in Figure 13.

Referring back to the Computer Detection Model in Figure 8, a few observations should be made. According to the model, if threat analysis indicates that an acceptable level of threat exists as a result of computer fraud deterrents, there is no need for expending resources in the prevention or detection of fraud. If threat analysis indicates an unacceptable level of threat, then prevention is suggested.
![Table showing threat values by schemes and perpetrators]

Figure 12. Threat Values by Schemes and Perpetrators

by the model. The effectiveness of prevention is determined through the use of "controls analysis" as indicated in block 5 of the model. Here a clarification is necessary. In this chapter the generalized threat assessment is based on actual reported incidents of computer fraud which occurred in spite of deterrents and preventive techniques in existence. Thus, for the general threat assessment in this chapter,
<table>
<thead>
<tr>
<th>Scheme/Perpetrator</th>
<th>&quot;T&quot; Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions added - data entry/terminal operators</td>
<td>82.2</td>
</tr>
<tr>
<td>Transactions altered - clerk/teller</td>
<td>36.4</td>
</tr>
<tr>
<td>Transactions added - officer/manager</td>
<td>31.4</td>
</tr>
<tr>
<td>Transactions added - outsider (non-employee)</td>
<td>26.4</td>
</tr>
<tr>
<td>Transactions altered - officer/manager</td>
<td>15.7</td>
</tr>
<tr>
<td>Transactions deleted - officer/manager</td>
<td>11.9</td>
</tr>
<tr>
<td>Program changes - officer/manager</td>
<td>11.9</td>
</tr>
<tr>
<td>File Changes - data entry/terminal operator</td>
<td>9.5</td>
</tr>
<tr>
<td>Program changes - programmer</td>
<td>9.3</td>
</tr>
<tr>
<td>Transactions altered - outsider (non-employee)</td>
<td>9.0</td>
</tr>
<tr>
<td>Transactions added - clerk/teller</td>
<td>6.6</td>
</tr>
<tr>
<td>Transactions altered - clerk/teller</td>
<td>4.4</td>
</tr>
<tr>
<td>File changes - officer/manager</td>
<td>4.1</td>
</tr>
<tr>
<td>Improper operation - officer/manager</td>
<td>4.1</td>
</tr>
<tr>
<td>Transactions altered - computer operator</td>
<td>1.8</td>
</tr>
<tr>
<td>Improper operation - computer operator</td>
<td>1.4</td>
</tr>
<tr>
<td>Transactions added - other staff</td>
<td>1.2</td>
</tr>
<tr>
<td>Transactions deleted - other staff</td>
<td>1.2</td>
</tr>
<tr>
<td>File changes - other staff</td>
<td>1.2</td>
</tr>
<tr>
<td>File changes - clerk/teller</td>
<td>.8</td>
</tr>
<tr>
<td>Transactions added - computer operator</td>
<td>.5</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>271.0</strong></td>
</tr>
</tbody>
</table>

Figure 13. Threat Values - Descending Order
blocks 2 through 5 of the model are compressed with controls analysis, in effect, embedded in the assessment.

This is not the case in the specific threat assessment methodology presented in Chapter 5. This alternative methodology requires that the threat analysis and controls analysis be conducted separately in the manner shown in the model in Figure 8. The discussion of controls analysis is deferred until Chapter 5 since it relates directly to the methodology presented there.
CHAPTER 4

DETECTION QUOTIENT

In Chapter 3 a threat value was derived for the various combinations of computer fraud manipulation schemes and perpetrators of these schemes. In this chapter, a detection quotient will be developed for each of the threats associated with the threat values shown in Figure 12 of Chapter 3. In Chapter 7 the detection quotients for each threat will be used in describing a computer fraud detection resource optimization model which maximizes the detection capability for a given system within available resources. As indicated previously, in the typical system today it is not feasible to examine every transaction or change in an automated system. In fact, in most large systems, only a very small percentage of transactions or changes may feasibly be examined thoroughly enough to detect fraud if it exists. The optimization model described in Chapter 7, by using the detection quotients explained in this chapter, can ensure an optimum or near optimum allocation of resources to the various threats surrounding a given system. In effect, the model uses individual detection quotients to determine the allocation of resources which will maximize the detection quotient for an entire system.

The detection quotient may be described as a value which measures the effectiveness of computer fraud detection resource allocation. This value, for individual system threats, is the product
of three sets of factors. The first is the set of threat values developed in Chapter 3. The second is a set of values which represent, for each of the threat values in Chapter 3 (Figure 12), the probability of detecting at least one occurrence of fraud when it occurs at a given level or rate. The third is a set of values representing the converse of the rate of occurrence.

For example, the detection quotient for transactions added by data entry/terminal operators would be computed as follows, given the sample conditions. Referring to Figure 12, the threat value for transactions added by data entry/terminal operators is 82.2. Now assume that it is possible to ensure with a probability of 95 percent that, if fraud occurs at a rate of .1 percent (or .001) in the transactions, at least one occurrence will be detected. Or, conversely, that it is possible to ensure with a probability of 95 percent that, if no occurrences are detected in the sample that 99.9 percent (converse of occurrence rate) of the transactions are fraud free. The detection quotient for this particular example is computed as follows:

\[
\text{Detection Quotient (DQ)} = \text{Threat Value (T)} \times \text{Probability of Detecting at Least One Occurrence of Fraud (P)} \times \text{Converse of Occurrence Rate (C)}
\]

Or, \( DQ = TPC \)

Or, \( DQ = (82.2) (.95) (.999) = 78.0 \)

The technique which will be used to ensure probabilities of detection for given rates of occurrence is the Discovery Sampling or Exploratory Sampling technique.
**Discovery (Exploratory) Sampling**

The purpose of discovery sampling is to disclose evidence of some activity, usually an irregularity, within a system. The type of evidence required need be only one example of such a serious deviation or irregularity. If found in the test, this one occurrence is sufficient to precipitate vigorous action such as a broader test or even a detailed examination. Thus, if the sample disclosed one example of a fraudulent transaction being added by a terminal operator, all transactions entered by that operator for the past year or more may be reviewed to determine the extent of the fraud.

The first thing that must be recognized is that discovery sampling does not provide a means for guaranteeing, with some small sampling, that the "needle in the haystack" type of case will be found. For example, if only one instance of fraud exists in a field of one million records, no sample short of virtual complete examination can give any reasonable assurance that the case will be found. Arkin (1967) suggests that, due to the sheer mass of records to be examined, that even a 100 percent check might not disclose such a unique instance. It should be noted that if the "needle in the haystack" case represents a small dollar value it may not be worth pursuing. If the one million records cited above represent $100 million, a single case of fraud for $1,000 represents only .00001 of the total dollars transacted. Except in very unusual circumstances, this amount would not cause much concern among higher management, in fact would probably be regarded as insignificant. On the other hand, if the fraud where for $10 million there would be much cause for concern. A
solution to this problem which will be discussed further later is the method of stratification. Using this approach dollar thresholds may be established above which all or some higher percentage of transactions are examined.

The mathematical formula for calculating the probability of at least one occurrence of an event given a particular field size is shown below (Arkin 1967):

$$Pr = 1 - \frac{\binom{d}{n} \binom{N-d}{n}}{\binom{N}{n}}$$

Where

- $N = \text{Field Size}$
- $n = \text{Sample Size}$
- $d = \text{Number of Events in Field}$
- $pr = \text{Probability of at least one event in the sample}$

$$\binom{N}{n} = \frac{N!}{n!(N-n)!}$$

$$\binom{b}{a} = \frac{b!}{a!(b-a)!}$$

Generally

It is apparent that the mathematical manipulations for the above formula become enormous for even a moderate field size. Consider, for example, that 10! (or $10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$) = 3,628,800, then exercise mentally for a moment, the above calculations with an N value (field size) of a modest 400 and an n value (sample size) of 50. Fortunately, tables have been prepared which satisfy all but very rare situations. These tables indicate for various field sizes and sample sizes, the probability that at least one example of the event sought will be included in a sample when it occurs at a specified rate.
For example, assume that 10,000 transactions have been processed through a system during a given month and that management is concerned with the probability that certain of these transactions might have been fraudulent. The field or N value in this case is 10,000. Assume for illustrative purposes that it takes 15 minutes, on the average, to verify a transaction's propriety or, conversely, its impropriety. Examination of all 10,000 transactions would take 2500 hours or roughly 15 people working full time for a month.

Typically, management would not be willing or able to devote this level of resources. While they would certainly like to find violations, even if only one or two occur, the managers must satisfy themselves with obtaining some high degree of assurance if some greater number of instances or a pattern occurs. Assume, for instance, that the managers are willing to accept a high degree of assurance of discovering one such occurrence if the occurrence rate is .2 percent (or 20 out of 10,000). From Arkin's tables, for a field size 10,000 and an occurrence rate of .2 percent, the following sample sizes and corresponding probabilities of discovering at least one occurrence have been extracted:

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>Probability of Finding at Least One Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>76.6%</td>
</tr>
<tr>
<td>1000</td>
<td>87.9</td>
</tr>
<tr>
<td>1400</td>
<td>95.1</td>
</tr>
<tr>
<td>2000</td>
<td>98.9</td>
</tr>
</tbody>
</table>

Assuming again that examination of each transaction in the
sample would require 15 minutes, on the average, the following estimates of resource requirements would apply:

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>Probability of Finding at Least One Occurrence</th>
<th>Resource Requirements (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>76.6%</td>
<td>175</td>
</tr>
<tr>
<td>1000</td>
<td>87.9</td>
<td>250</td>
</tr>
<tr>
<td>1400</td>
<td>95.1</td>
<td>350</td>
</tr>
<tr>
<td>2000</td>
<td>98.9</td>
<td>500</td>
</tr>
</tbody>
</table>

For the above example the resource requirements range from 175 hours, or approximately one person full time for a month with a 76.6% probability of detecting at least one occurrence, to 500 hours or about three people full time for a month with a 98.9% probability of detecting at least one occurrence if the rate of occurrence is .002. These resource requirements may be compared to the 2,500 hours (about 15 people full time for a month) required to examine all 10,000 records.

The resource requirements would decrease or increase if the managers were willing to accept a higher or lower risk, respectively. Assume, for example, that the managers were willing to accept a high probability of detecting at least one occurrence if the rate of occurrence is .5% (50 out of 10,000). Referring again to the tables, the following sample sizes and probabilities are extracted with the associated resource requirement estimates using the 15 minute per record examination time:

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>Probability of Finding at Least One Occurrence</th>
<th>Resource Requirements (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>63.7%</td>
<td>50</td>
</tr>
<tr>
<td>500</td>
<td>92.4</td>
<td>125</td>
</tr>
<tr>
<td>600</td>
<td>95.5</td>
<td>150</td>
</tr>
<tr>
<td>800</td>
<td>98.5</td>
<td>200</td>
</tr>
</tbody>
</table>
The resource requirements in this example are significantly less than in the last example.

Moving in the opposite direction the following sample sizes, probabilities and resource requirements would apply if the managers wanted a high probability of discovering at least one occurrence, if as few as 10 occurrences existed in the field of 10,000.

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>Probability of Finding at Least One Occurrence</th>
<th>Resource Requirements (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1400</td>
<td>77.9%</td>
<td>350</td>
</tr>
<tr>
<td>1800</td>
<td>86.3</td>
<td>450</td>
</tr>
<tr>
<td>3000</td>
<td>97.2</td>
<td>750</td>
</tr>
</tbody>
</table>

This time there is a significant increase in resource requirements to insure the detection of at least one occurrence with a high probability given that only ten exist in the data. However, it should be noted that even the largest sample size of 3000 requiring approximately 750 hours is modest compared to the 2500 hours required for a total examination.

How large should the sample be for the above 10,000 records? Unfortunately there is no pat answer to this question. As indicated previously, the nature of the business might indicate a widely varying sample size for different organizations. A bank, built on the trust and confidence of its clients, might require a much more extensive sample than a supplier of electronic components even though the expected losses in the latter firm may be greater.

Another factor which might have a significant influence on the sample size is the nature of the transactions. For example, assume
that in one case the dollar value of transactions was distributed as follows based on the above field size of 10,000:

<table>
<thead>
<tr>
<th>Percentage of Transactions</th>
<th>Dollar Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1% (100 Transactions)</td>
<td>$100,000 or Greater</td>
</tr>
<tr>
<td>80% (8,000 Transactions)</td>
<td>$10,000 to $100,000</td>
</tr>
<tr>
<td>19% (1,900 Transactions)</td>
<td>$10,000 or less</td>
</tr>
</tbody>
</table>

In this example a perpetrator could, with only a few manipulations extract a considerable sum without attracting too much attention due to the size of the transactions. Now assume that the 10,000 transactions are distributed in the following dollar categories:

<table>
<thead>
<tr>
<th>Percentage of Transactions</th>
<th>Dollar Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5% (500 Transactions)</td>
<td>$500 - $1,000</td>
</tr>
<tr>
<td>95% (9,500 Transactions)</td>
<td>$500 - or less</td>
</tr>
</tbody>
</table>

In the latter distribution it would take numerous transaction manipulations within the normal dollar values shown to extract a sizeable sum. Thus, in the latter example a sample based on an occurrence rate of .5% (50 out of 10,000) might be adequate. In the first example, a sample might have to be based on an occurrence rate of .1% (10 out of 10,000) or less to provide adequate protection.

For the first example above the stratification technique mentioned earlier might be desirable. For example, a total examination of all 100 transactions exceeding $100,000 might be performed. Then, discovery sampling used for the remaining 9,900 transactions. Still another approach might be to examine all transactions over $100,000. Then, establish two different discovery sampling schemes for the next two categories. Thus, for transactions between $10,000 and $100,000 a
probability of at least 95% of detecting at least one occurrence may be required if the rate of occurrence is .2%. For the third category, or those transactions under $10,000, a probability of 90% of detecting at least one occurrence with a rate of occurrence of .5% might be adequate.

It should be pointed out that a sample drawn for discovery sampling can also be interpreted to give an indication as to the degree of fraudulent activity at a given probability level. To illustrate, if a discovery sample is completed without disclosing an occurrence of fraud, this may be interpreted to mean that there is a high probability that the rate of occurrence is less than that used for the sample size selection.

Consider, for instance, the example above which was based on a field size of 10,000 and a rate of occurrence of .2% (20 out of 10,000). For this example a sample size of 2,000 transactions gives a probability of 98.9% of disclosing at least one example if there are 20 in the field. If none is found, the result may be interpreted to mean that there is a 98.7% probability that the rate is less than .2% or 20 out of 10,000.

Sample Size - Unlimited Resources

As indicated above, the sample size required is a direct function of several factors, one of which is management's aversiveness to risk, as measured by the probability of fraud in their systems. Many managers faced with the question "how aversive are you to fraud?" would probably initially indicate a total aversiveness to fraud by
stating that they would tolerate no fraud in their systems. However, after a brief explanation of the effort required to examine every transaction or change within their systems, most managers would probably agree that it is not feasible to eliminate all risk within a system. If, after acquiring a good understanding of the efforts involved, management is still totally averse to fraud, the sample size is set at 100 percent and necessary resources applied. Given this situation, which is probably highly unusual, the analysis which follows in this section and in Chapter 7 is unnecessary since there is not a resource allocation problem. By assuming unlimited resources and setting the sample size at 100 percent, the detection quotient (DQ) would be maximized for both the individual threats identified in Chapter 3 (or Chapter 5) and for the total system.

Recall that the DQ value for each threat is determined by the formula $DQ = TPC$ where $T =$ the threat value from Figure 12; $P =$ the probability of detecting at least one occurrence of fraud and $C =$ the converse of the occurrence rate on which the sample was based.

By taking a 100 percent sample the probability of finding at least one occurrence, even if only one occurrence exists, should be close to 100 percent. Recalling the previous example from Figure 12 which was based on transactions added by data entry/terminal operators, the following DQ would be applicable where the rate of occurrence is .0001 (1 out of 10,000):

$$DQ = TPC$$

$$= (82.2) (1.00) (.9999)$$

$$= 82.2$$
If, as Arkin (1967) suggested, a total sample would not ensure a 100 percent probability of detecting a single occurrence when only one exists in a large field, the above formula would have to be adjusted by an inefficiency factor of some sort. If, for example, it could be shown that for the above situation with a 100 percent sample there is only a 98 percent probability of discovering a single occurrence because of inefficiencies in examining procedures, the DQ would become:

$$DQ = (82.2) (.98) (.9999) = 80.5$$

The above inefficiency factor would be difficult to quantify. Since a total sample is typically not feasible for today's systems, the point becomes somewhat academic. The total sample is used in this section to illustrate the detection quotient rather than suggest it as a viable alternative.

As shown above, inefficiencies in investigating have a decreasing affect on the DQ value. However, in either of the above cases, the DQ has been maximized since 100 percent of the transactions were examined. It should be clear by now that the maximum possible DQ values are the corresponding threat values shown in the threat matrix (Figure 12). Thus, for the threat matrix in Figure 12 the maximum possible DQs correspond to the threat values shown in the matrix cells. The maximum total DQ for the matrix is 271 (the total for all threat values in Figure 12).

Sample Size - Limited Resources

The case described above probably represents an almost
nonexistent situation in larger systems. The cost of conducting a 100 percent examination of all transactions and changes in a large system would be very high if such an examination is even possible. Given that it is possible, the cost effectiveness of such an approach would be doubtful in most systems. More typically, the resources available to dedicate to the examination of transactions or changes in a system are only a fraction of those required to conduct a 100 percent examination.

In the latter situation the primary objective is to maximize the ability to detect fraud, as measured by the system DQ, given the limitations on available resources. The computer fraud detection resource optimization model in Chapter 7 is designed to provide this maximum or near maximum DQ value.

The model in Chapter 7 is based on the premise that some limited number of hours is available to allocate to computer fraud detection. For example, assume that only 300 hours are available per month in a system that would require 5,000 hours per month for a 100 percent examination. The objective of the resource optimization model in this case would be to achieve the maximum or near maximum DQ for the system given the constraint of 300 hours.

As indicated previously, management aversiveness varies from one organization to another based on the type of business, distribution of transactions or changes in their systems, their own tolerance for risk, etc. Thus, it is possible that the DQ value and associated probabilities for given rates of occurrence of the different fraud schemes generated by the model in optimizing the use of the 300 available hours will not be as high as management would like. In this
case management has two options. One is to allocate additional resources, thus raising the 300 hour constraint to some higher amount and rerunning the model. This process can be repeated until an acceptable combination of protection and resource allocation is reached. Management's second option is to accept the values associated with the 300 hours as the best that can be achieved, albeit not the most desirable, if additional hours cannot be allocated.

An alternative to the model was considered for establishing sample sizes and resulting probabilities for detecting fraud for the different schemes. This alternative is to determine, through interviews with management, the sample sizes for the various fraud schemes and perpetrator types, based on their aversiveness to fraud. Thus, management would, through some series of questions and answers, reveal their required probability for detecting at least one occurrence of fraud in transactions added by data entry/terminal operators, for example, if certain rates of occurrence exist. These probabilities and associated rates of occurrence would determine the sample sizes, resulting DQ values and required resources.

This approach was rejected for four reasons: First, management would be placed in the uncomfortable position of having to implicitly condone a certain level of fraud by agreeing to something less than a 100 percent probability of detection; second, it is difficult to convert fraud aversiveness to terms of probabilities and rates of occurrence; third, unless resources are highly flexible it would take numerous iterations (and a great deal of management's time) to achieve compromised sets of probabilities and rates of occurrence that could be
achieved within available resources; and fourth, given the level of available resources, it would be highly unlikely that management's probabilities and rates of occurrence would result in a system DQ as high as the model's.

In order to demonstrate the relationships between probabilities of detection, rates of occurrence, sample sizes and system DQ values, a sample application is presented.

The sample system contains the following volumes (field sizes) for the various scheme types each month:

<table>
<thead>
<tr>
<th>Scheme Type</th>
<th>Field Size (in Monthly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions Added</td>
<td>30,000</td>
</tr>
<tr>
<td>Transactions Altered</td>
<td>10,000</td>
</tr>
<tr>
<td>Transactions Deleted</td>
<td>5,000</td>
</tr>
<tr>
<td>File Changes</td>
<td>2,000</td>
</tr>
<tr>
<td>Program Changes</td>
<td>1,000</td>
</tr>
<tr>
<td>Operation Cycles</td>
<td>2,000</td>
</tr>
</tbody>
</table>

Now assume that management has decided that, if fraud exists with a rate of occurrence of .5% or greater, a probability of 90 percent must be ensured of detecting at least one such case. The sample sizes from Arkin's tables for each of the scheme types are shown in Figure 14.

As indicated in column 3 of Figure 14, as the population size increases, with the rate of occurrence held constant, the sample size needed to provide the same probability of discovering at least one occurrence of fraud increases at a decreasing rate. For populations above 10,000 the sample size remains almost constant for given rates of occurrence. It may be noted that for a population of 200,000, a sample
size of 500 provides a 90 percent probability of detecting at least one occurrence of fraud when the occurrence rate is .5% - the same protection shown in Figure 14 for a population of 10,000.

<table>
<thead>
<tr>
<th>Scheme Type</th>
<th>Monthly Field Size</th>
<th>Sample Size For 90% Probability* With .5% Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions Added</td>
<td>30,000</td>
<td>500</td>
</tr>
<tr>
<td>Transactions Altered</td>
<td>10,000</td>
<td>500</td>
</tr>
<tr>
<td>Transactions Deleted</td>
<td>5,000</td>
<td>400</td>
</tr>
<tr>
<td>File Changes</td>
<td>2,000</td>
<td>400</td>
</tr>
<tr>
<td>Program Changes</td>
<td>1,000</td>
<td>350</td>
</tr>
<tr>
<td>Operation Cycles</td>
<td>2,000</td>
<td>400</td>
</tr>
</tbody>
</table>

* Closest probability to 90% was taken from the table.

Figure 14. Sample Size Example

By combining the probabilities of detection, the rates of occurrence and systems threat values, DQ values may be derived for each of the systems threats and for the system as a whole. This has been accomplished in Figure 15 for the threat values derived in Chapter 3. The threat values from Figure 13 in Chapter 3, a 90% probability of detection and an occurrence rate of .5% from above, are used to calculate detection quotients using the formula DQ = TPC.

For purposes of illustration, the probabilities of detecting at least one occurrence and the rates of occurrence for all scheme/perpetrator combinations were set at a constant 90% and .5% respectively. This constraint is lifted in describing the resource optimization model in Chapter 7 where the probabilities and rates of occurrence become variables for each scheme/perpetrator combination in seeking a maximum or near maximum DQ for a given level of resource.
<table>
<thead>
<tr>
<th>Scheme/Perpetrator</th>
<th>&quot;T&quot; Value</th>
<th>&quot;P&quot; Value</th>
<th>&quot;C&quot; Value</th>
<th>DQ Value (TPC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions Added - Data Entry/Terminal Operator</td>
<td>82.2</td>
<td>.9</td>
<td>.995</td>
<td>73.6</td>
</tr>
<tr>
<td>Transactions Altered - Data Entry/Terminal Operator</td>
<td>36.4</td>
<td>.9</td>
<td>.995</td>
<td>32.6</td>
</tr>
<tr>
<td>Transactions Added - Officer/Manager</td>
<td>31.4</td>
<td>.9</td>
<td>.995</td>
<td>28.1</td>
</tr>
<tr>
<td>Transactions Added - Outsider (non-employee)</td>
<td>26.4</td>
<td>.9</td>
<td>.995</td>
<td>23.6</td>
</tr>
<tr>
<td>Transactions Altered - Officer/Manager</td>
<td>15.7</td>
<td>.9</td>
<td>.995</td>
<td>14.1</td>
</tr>
<tr>
<td>Transaction Deleted - Officer/Manager</td>
<td>11.9</td>
<td>.9</td>
<td>.995</td>
<td>10.7</td>
</tr>
<tr>
<td>Program Changes - Programmer</td>
<td>11.9</td>
<td>.9</td>
<td>.995</td>
<td>10.7</td>
</tr>
<tr>
<td>File Changes - Data Entry/Terminal Operator</td>
<td>9.5</td>
<td>.9</td>
<td>.995</td>
<td>8.5</td>
</tr>
<tr>
<td>Program Changes - Programmer</td>
<td>9.3</td>
<td>.9</td>
<td>.995</td>
<td>8.3</td>
</tr>
<tr>
<td>Transactions Altered - Outsider (non-employee)</td>
<td>9.0</td>
<td>.9</td>
<td>.995</td>
<td>8.1</td>
</tr>
<tr>
<td>Transactions Added - Clerk Teller</td>
<td>6.6</td>
<td>.9</td>
<td>.995</td>
<td>5.9</td>
</tr>
<tr>
<td>Transactions Altered - Clerk/Teller</td>
<td>4.4</td>
<td>.9</td>
<td>.995</td>
<td>3.9</td>
</tr>
<tr>
<td>File Changes - Officer/Manager</td>
<td>4.1</td>
<td>.9</td>
<td>.995</td>
<td>3.7</td>
</tr>
<tr>
<td>Operations - Officer/Manager</td>
<td>4.1</td>
<td>.9</td>
<td>.995</td>
<td>3.7</td>
</tr>
<tr>
<td>Transaction Altered - Computer Operator</td>
<td>1.8</td>
<td>.9</td>
<td>.995</td>
<td>1.6</td>
</tr>
<tr>
<td>Operations - Computer Operator</td>
<td>1.4</td>
<td>.9</td>
<td>.995</td>
<td>1.3</td>
</tr>
<tr>
<td>Transactions Added - Other Staff</td>
<td>1.2</td>
<td>.9</td>
<td>.995</td>
<td>1.1</td>
</tr>
<tr>
<td>Transactions Deleted - Other Staff</td>
<td>1.2</td>
<td>.9</td>
<td>.995</td>
<td>1.1</td>
</tr>
<tr>
<td>File Changes - Other Staff</td>
<td>1.2</td>
<td>.9</td>
<td>.995</td>
<td>1.1</td>
</tr>
<tr>
<td>File Changes - Clerk/Teller</td>
<td>.8</td>
<td>.9</td>
<td>.995</td>
<td>.7</td>
</tr>
<tr>
<td>Transactions Added - Computer Operator</td>
<td>.5</td>
<td>.9</td>
<td>.995</td>
<td>.4</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td><strong>271.0</strong></td>
<td></td>
<td></td>
<td><strong>242.8</strong></td>
</tr>
</tbody>
</table>

*Figure 15. Calculation of DQ Values*
The far right column of Figure 15 shows the DQ values for each scheme/perpetrator combination. The total for this column shows the system DQ of 242.8. The underlying resource requirements are dictated by the sample sizes needed to accomplish the 90% probability of detection if a .5% occurrence rate exists and the time required to examine each element in the sample.

The objective of the model in Chapter 7 is to optimize the utilization of available resources as measured by the system detection quotient values. The methodology for accomplishing this is to iteratively experiment with various probabilities and rates of occurrence for the various scheme/perpetrator combinations, within available resources until a maximum or near maximum DQ is achieved. This process requires the development of a technique for feasibly and quantitatively evaluating enormous combinations. As will be shown in Chapter 6 entitled "The Combinatorial Dilemma - A General Solution", evaluation of all possible probability/occurrence rate combinations for the various scheme/perpetrator combinations is not practical due to the phenomenally large number of combinations. The methodology in Chapter 6 is offered as a general solution since the approach should lend itself in a general sense to the "combinatorial dilemma" wherein the sheer volume of combinations which must be evaluated precludes the evaluation of all combinations.
CHAPTER 5

SPECIFIC THREAT ASSESSMENT

In contrast to the "General Threat Assessment" in Chapter 3, which transcends all organization and systems types, the assessment in this chapter is tailored to fit individual systems within given organizations.

Specific threat assessment will be defined as "the application of scientific procedures to the study and evaluation of the computer fraud risks surrounding individual automated systems" (Auerbach 1980). For purposes of this methodology, the term "specific threat assessment" will be used interchangeably with the terms "risk analysis", "threat analysis" and "vulnerability analysis".

Because of the limited resources available to the typical organization to combat computer fraud, threat assessment is vital in allocating these resources effectively. In a recent article (Auerbach 1980), Lobel speculated on the use of risk analysis in the 1980's. Although risk analysis is still in its infancy, Lobel predicts "major breakthroughs before the end of this decade -- both in improved techniques and greater utilization of risk analysis procedures by computer users". However, even by today's standards, he contends that risk analysis has real merit. The problem is "that many organizations have to be convinced of its potential".

Several techniques were found in the literature for evaluating
the threats or risks surrounding systems. Practically all of these techniques were oriented to the broader subject of computer security rather than computer fraud. Many of the risks or threats covered in these techniques are of the "unintentional" variety. Examples of this variety of risk or threat are natural disasters, e.g., extreme weather conditions and errors and omissions, e.g., incorrect tape labeling.

The unintentional variety of risks or threats are empirically predictable which facilitates threat assessment. Intentional acts such as computer fraud or sabotage are generally not empirically predictable and, thus, do not always lend themselves to the same threat assessment techniques as unintentional acts like natural disasters, errors and omissions.

In order to establish a framework for reference between various existing techniques in "risk analysis" and the methodology in this chapter, several of those found in the literature are synopsized below.

Risk Analysis Techniques

The Statistical Approach (Wong 1977)

Published literature in risk management theory and practice advocates a statistical approach utilizing an organization's own recorded historical data to plot a graph showing the frequency distribution of past occurrences of various hazards and risks against the loss or damage incurred.

According to Wong, if enough data is collected and collated, the graph will usually take the form of a positively skewed Poisson curve as in Figure 16 in the shapes of A, B, or C.
The peak of the curve is the modal value or mode. It represents the past average loss for most risks or threats. The spread of the curve or the modal region indicates the range of loss for most risks.

The advantage of this approach is that the Poisson curve gives a good indication of the average loss and a fair estimate of the maximum probable loss.

However, as Wong points out, where the method fails is that a lot of good and accurate data is required to establish the shape of the frequency distribution curve. Unfortunately, the converse is true with computer fraud. For individual organizations, good organizational
data on computer fraud is very difficult to find.

Although a statistical approach would be desirable because the results are defensible, such an approach is typically not feasible for computer fraud due to the lack of good and accurate data. This approach might work quite well for unintentional acts such as natural disasters, errors and commissions since considerably more data exists for these acts.

Risk Estimation Approach

This approach simply requires that systems vulnerabilities be identified and potential losses associated with these vulnerabilities estimated without the benefit of any appreciable historical data. The following techniques for performing risk estimation represent the views of several different authors as to how this might be accomplished.

Benefits Approach (Krauss and MacGahan, 1979)

In their book entitled "Computer Fraud and Counter-measures", Krauss and MacGahan attempt to quantify the benefits of computer fraud countermeasures. These benefits include risk reduction and other benefits. They suggest the four step approach, as shown in Figure 17.

The first step is to list the specific threats or risks that might materialize. In the sample payroll application shown in Figure 17, the first such risk is having a fraud perpetrated by adding an extra employee to the payroll system.

The second step is to estimate the number of occurrences for each risk on an annual basis. If it is estimated that an incident could happen once every four years, the frequency would be 0.25.
The third step is to estimate the average dollar loss per incident. An alternative approach is to use a high/average/low dollar estimate with corresponding frequencies of occurrence for each of the three dollar values in Step 2.

<table>
<thead>
<tr>
<th>Step 1</th>
<th>STEP 2</th>
<th>STEP 3</th>
<th>STEP 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumerate the Risks</td>
<td>Estimate Frequencies of Occurrence A</td>
<td>Estimate Dollar Loss per Occurrence B</td>
<td>Compute Annual Expected Losses AXB</td>
</tr>
<tr>
<td>Extra Employee on Payroll</td>
<td>3.00</td>
<td>$15,000</td>
<td>$45,000</td>
</tr>
<tr>
<td>Breakage in Computing Taxes</td>
<td>0.25</td>
<td>5,000</td>
<td>1,250</td>
</tr>
<tr>
<td>Unauthorized Pay Rate Change</td>
<td>2.00</td>
<td>8,000</td>
<td>16,000</td>
</tr>
<tr>
<td>Disclosure of Employee Data</td>
<td>0.10</td>
<td>15,000</td>
<td>1,500</td>
</tr>
<tr>
<td>TOTAL ANNUAL EXPECTED LOSS</td>
<td></td>
<td></td>
<td>$63,750</td>
</tr>
</tbody>
</table>

Figure 17. Payroll Risk Analysis Example

The final step is to compute the expected annual losses by multiplying the frequency times the dollar value of an occurrence. The total, then, represents the expected dollar losses from all risks considered. This total of $63,750 in Figure 17 would indicate a maximum to spend on countermeasures according to the authors.

The above process should be accomplished by a group of people, possibly including representatives from the user department, data processing, internal audit, independent audit, security, personnel, and
consultants on computer security and controls.

As pointed out by Krauss and MacGahan, the above results are difficult to defend since the factors are developed from guesses, inasmuch as a statistical basis does not generally exist for a given organization. In effect, there is no scientific defense to support the use of the total expected losses calculated in the above fashion in determining the amount to spend on countermeasures.


NBS has developed a general strategy for risk analysis which classifies undesirable computer events in terms of their general effects on computerized data rather than in terms of their ultimate effect such as denial of benefits, or loss of money or resources.

The NBS classification of undesirable events (vulnerabilities that are activated) relates them directly to three general security control objectives for all application systems. These vulnerabilities and their countering control objectives are:

- Modification or destruction of data - Data Integrity
- Disclosure of data - Data Confidentiality
- Unavailability of data or system service - ADP Availability

Computer fraud would fall into the first category. The last two categories pertain to other subsets in the broader topic of computer security.

The NBS approach is oriented primarily to the development of controls to offset vulnerabilities. The approach does not couple
vulnerabilities with specific controls. However, they have compiled a long list of vulnerabilities which occur in an application environment. Six basic control categories are provided by NBS along with the general problems that each will address. Included are:

- Data Validation
  - Consistency and reasonableness checks
- Data entry Validation
- Validation during processing
- Data element dictionary/directory
- User Identity Verification
- Authorization
- Journalling
- Variance Detection
- Encryption

The NBS approach also emphasizes the placement and use of appropriate controls at each stage of the system life cycle: initiation; development; and operation. The NBS approach provides valuable insight and direction into problems, vulnerabilities and controls. However, it does not, nor was it intended to, provide a specific methodology for implementing the approach.

The Matrix Approach (Fitzgerald 1978)

The most rigorous of the techniques reviewed, the Matrix Approach, subdivides the data processing function into nine components. Controls relating to each component are then enumerated and correlated with specific vulnerabilities. Fitzgerald identifies more than 650
practical controls. The nine data processing categories against which the controls are analyzed are:

- General Organization
- Input
- Data Communication
- Program/Computer Processing
- Output
- On-Line Terminal/Distributed Systems
- Physical Security
- Data Base
- System Software

For each of these components a matrix is developed which identifies specific controls that address a firm's concerns/exposures for protection of the resources/assets peculiar to that component of the overall system.

The Matrix Approach does not comprise a methodology but rather a tool for conducting an internal control review. Fitzgerald makes the following statement with regard to the approach:

"The nine control matrices do not comprise a methodology on the conduct of an internal control review. Instead, the overall methodology on how to conduct an internal control review is assumed to be already established within the organization. The matrix approach of this book is a tool that works with the diverse methodologies for conducting internal control reviews that are in use by a variety of organizations."

The Matrix Approach provides an excellent medium for identifying and reviewing relevant controls for the protection of resources/assets from general threats. The approach is, however, quite
cumbersome and complicated with nine matrices and over 650 controls. Further, in the final analysis, the approach provides a "yes/no" assessment. Thus, a determination is made that controls either exist or do not exist to protect a given resource or asset from a specific threat. The question still remains as to how effective the controls are.

**A Specific Risk Assessment Methodology**

The Specific Risk Assessment Methodology for this dissertation differs from other efforts in that it: concentrates solely on computer fraud; considers risk assessment and abatement a resource optimization problem; and provides a means of quantifying its effectiveness.

Specific Threat Analysis

As with the General Threat Analysis methodology in Chapter 3, the Specific Threat Analysis uses a matrix approach starting with the blank matrix shown in Figure 18.

Manipulation schemes are listed across the top of the matrix and potential perpetrators are listed down the left side of the matrix. The matrix size is determined by the number of threats and potential perpetrators identified for a given system. The matrix cells will be discussed in the next section on controls analysis. Consider for a moment what the threat matrix might look like for the simplified system schematic in Figure 19.

Now consider what the matrix might look like for the schematic in Figure 20. It should be apparent that the matrices for the two systems will vary significantly.
There are several techniques for identifying significant scheme and perpetrator threats for a system. The system might be reviewed by an internal or external audit group who, in turn, fills in the matrix. The systems design group or computer section could be called on to do the job, or the functional manager might do it. Another technique might be to look at similar systems for which schemes have been identified and assume similar schemes and perpetrators for the system at hand. Or, a simple review of recent literature on internal controls
for computerized systems might reveal "common" schemes and perpetrators.

Any of these techniques will probably identify most of the threats surrounding a given system with a modest effort, particularly for a relatively simple system such as that portrayed in Figure 19.

However, for the larger, more complicated systems which cross several functional lines, (e.g., accounting, budget, inventory, etc.) and several technical disciplines, (e.g., computer hardware, computer software, communications, engineering, etc.) a more sophisticated approach might be required.

A better approach is to form a group made up of an expert or
Figure 20. On-Line System
experts from each of the primary areas involved with the express objective of identifying the schemes and potential perpetrators for a given system. For example, assume that the system in Figure 20 is an integrated accounting inventory and disbursing system. In this case the group should include experts from each of the following areas: accounting; inventory control; paying and collecting; computer hardware; communications; systems design; systems analysis; computer programming and internal review or auditing. One individual might be an expert in more than one area (e.g., systems analysis, systems design and computer programming).

Using a good structured approach, such a group should be able to identify the schemes and potential perpetrators for the system with a high degree of accuracy within just a few hours.

A recommended approach for the threat analysis is the Delphi method. This method is characterized as a technique for structuring a group communication process so that the group of individuals, as a whole, can effectively deal with a complex problem.

Although there are various approaches to Delphi, it usually involves four phases (Linstone and Turoff 1975). The first phase involves exploration of the subject under discussions. Each individual contributes additional information he or she feels is pertinent to the issue. In the second phase an understanding is reached of how the group views the issue (i.e., where the members agree or disagree and what they mean by relative terms such as importance, desirability, or feasibility). If significant disagreement exists, then the disagreement is explored in the third phase to bring out the underlying
reasons for the difference and possibly to evaluate them. The last phase is a final evaluation, when all previously gathered information has been initially analyzed and the evaluations have been fed back for consideration and modification, if necessary.

The Delphi approach lends itself well to problems for which there are no precise analytical techniques but which can benefit from subjective judgements on a collective basis. Identification of schemes and potential perpetrators in automated systems of varying size and complexity generally fits this description.

The group monitor should be familiar with the Delphi method and, preferably have some experience in administering it. If such a person is not available it would probably be better not to attempt the Delphi approach, but rather to opt for a less structured "brainstorming" approach. Although the latter may not be as effective as a well run Delphi, it would probably be superior to a poorly run Delphi.

For purposes of illustration, assume that the schemes and perpetrators in Figure 21 have been identified for a given system.

In the general area of risk analysis there are several techniques in the literature for evaluating the significance of threats once they are identified. However, the basic process tends to follow three steps (Auerbach 1980).

- Estimate the frequency of threat events
- Estimate the cost per occurrence
- Calculate the expected annual loss for each event

If, for example, the estimated frequency for a given threat is once every 50 years and the estimated dollar loss per occurrence is
$100,000 the expected annual loss for the threat would be calculated as follows:

\[
\text{Average Annual Loss} = \frac{\text{Estimated Loss Per Occurrence}}{\text{Estimated Frequency}}
\]

or,

\[
\text{Average Annual Loss} = \frac{\$100,000}{50} = \$2,000
\]

This approach lends itself quite well to the unintentional variety of computer threats such as floods, fires, errors and mechanical failures since these events are empirically predictable with
some degree of accuracy. Unfortunately, the approach is not as effective for unintentional acts such as computer fraud or sabotage which are not empirically predictable.

Churchman - Ackoff Method

An approach which does not use expected values or probabilities is the Churchman - Ackoff method (R.I. Ackoff and M.W. Sasieni, 1968). This procedure is based on the premise that, given any two outcomes, a decision-maker can estimate how serious they are in terms of their relative significance. It is applicable only to the analysis of outcomes that can be expressed in terms of "yes" or "no". It is based on the following assumptions:

1. For the initial ranking, it is assumed that the decision-maker can make a rough measure of threats. For all possible pairs of threat choices (e.g., T(1) and T(2)), the decision-maker must know whether T(1) is more or less of a threat than T(2)).

2. The decision-maker is also assumed to be able to assign some value to every threat to identify how serious it is. The difference between these values should allow us to make statements such as "one choice is twice as risky as another". Every choice, therefore, must be assigned a number that represents a measure of its relative threat.

3. If T(1) is more of a threat than T(2), it will rank higher. If the values assigned to the two threats are equal, the importance of these threats is equal in the mind of the decision-maker.

4. If two threats (T(1) and T(2)) have different threat levels, the combined outcomes of these two threats is the sum of the two. This assumption will fail if the two threats are mutually exclusive. It will also fail if the occurrence of one threat implies the occurrence of the other threat.

5. This procedure works best with six to eight threat choices.

The relative significance of threats may be evaluated using the
Churchman - Ackoff method as follows:

1. Have the decision-maker (or group) rank the threats in the order of their importance to the organization. Referring back to Chapter 2 this ranking should reflect vulnerabilities considering both the likelihood and impact should fraud occur. Let T(1) represent the greatest threat and T(N) the least threat, where N equals the number of different threats being considered.

2. Assign a value of 1 to T(N) (the least threat) and have the decision-maker assign numbers to the remaining threats in order to identify their relative importance to the organization. In this way, the items that are of the greatest threat will have the highest values.

3. Present the decision-maker with the choices shown in the general form in Figure 22 below:

```
Start T(1):T(2) + T(3) + ... + T(N)  ...  T(N-2):T(N-1) + T(N)
T(1):T(2) + T(3) + ... + T(N-1)  ...  T(N-2)  T(N-1)
T(1):T(2) + T(3) + ... + T(N-2)  ...  STOP

\vdots
T(1):T(2) + T(3)
T(1)  T(2)
```

Figure 22. Churchman - Ackoff General Form

4. Offer the choices indicated, starting from the top left-hand column. If the left-hand side of a choice is greater than or equal to the right-hand side, proceed to the top of the next column to the right; otherwise, continue down the column.

5. Check the numbers originally assigned by the decision maker to determine if they are consistent with the inequalities obtained by working through the table in Figure 22. If not, modify the numbers as little as possible to make them consistent with the inequalities.

The inequalities mean:
- is less threat than
- is greater threat than
- is an equal or greater threat than

To illustrate how this process works, assume the fraud schemes from Figure 21 have been ranked in the following order when ST(1) represents the greatest threat and ST(7) represents the least threat.
ST(1) = Transactions Added
ST(2) = File Changes
ST(3) = Program Changes
ST(4) = Transactions Altered
ST(5) = Transactions Deleted
ST(6) = Communications
ST(7) = Improper Operations

Now assume that the series of comparisons as depicted in Figure 22 happens to work out as follows:

1. ST(1) ≤ ST(2) + ST(3) + ST(4) + ST(5) + ST(6) + ST(7)
2. ST(1) ≤ ST(2) + ST(3) + ST(4) + ST(5) + ST(6)
3. ST(1) ≤ ST(2) + ST(3) + ST(4) + ST(5)
4. ST(1) ≤ ST(2) + ST(3) + ST(4)
5. ST(1) ≈ ST(2) + ST(3)
6. ST(2) < ST(3) + ST(4) + ST(5) + ST(6) + ST(7)
7. ST(2) < ST(3) + ST(4) + ST(5) + ST(6)
8. ST(2) < ST(3) + ST(4) + ST(5)
9. ST(2) < ST(3) + ST(4)
10. ST(2) ≈ ST(3)
11. ST(3) < ST(4) + ST(5) + ST(6) + ST(7)
12. ST(3) < ST(4) + ST(5) + ST(6)
13. ST(3) < ST(4) + ST(5)
14. ST(3) ≈ ST(4)
15. ST(4) < ST(5) + ST(6) + ST(7)
16. ST(4) < ST(5) + ST(6)
17. ST(4) ≈ ST(5)
18. ST(5) < ST(6) + ST(7)
19. ST(5) ≈ ST(6)
20. ST(6) ≈ ST(7)

Next, the decision maker must assign specific weights to each threat. At this point the weights are preliminary and will be refined based on the evaluation of the series of choices. For example, assume that the following weights have been assigned:

ST(1) = 18
ST(2) = 9
ST(3) = 6
ST(4) = 5
ST(5) = 4
ST(6) = 2
ST(7) = 1

In relative terms, these weights would mean:
ST(6) is two times the threat of ST(7)
ST(5) is four times the threat of ST(7)

...ST(1) is eighteen times the threat of ST(7)
ST(5) is two times the threat of ST(6), Etc.

The assigned weight values should then be plugged into the series of comparisons and necessary adjustments made. Starting at the bottom of the list and working toward the top this is accomplished in Figure 23.

<table>
<thead>
<tr>
<th>Equation</th>
<th>Evaluation-Based on Assigned weights</th>
<th>Results</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 ST(6) ≦ ST(7)</td>
<td>2 ≦ 1</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>19 ST(5) ≦ ST(6)</td>
<td>4 ≦ 2</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>18 ST(5) &lt; ST(6)+ST(7)</td>
<td>4 &lt; (2+1)</td>
<td>ERROR</td>
<td>Let ST(5) = 2.5</td>
</tr>
<tr>
<td>17 ST(4) ≦ ST(5)</td>
<td>5 ≦ 2.5</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>16 ST(4) &lt; ST(5)+ST(6)</td>
<td>5 &lt; (2.5+2)</td>
<td>ERROR</td>
<td>Let ST(4) = 4</td>
</tr>
<tr>
<td>15 ST(4) &lt; ST(5)+ST(6)+ST(7)</td>
<td>4 &lt; (2.5+2+1)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>14 ST(3) ≦ ST(4)</td>
<td>6 ≦ 4</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>13 ST(3) &lt; ST(4)+ST(5)</td>
<td>6 &lt; (4+2.5)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>12 ST(3) &lt; ST(4)+ST(5)+ST(6)</td>
<td>6 &lt; (4+2.5+2)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>11 ST(3) &lt; ST(4)+ST(5)+ST(6)+ST(7)</td>
<td>6 &lt; (4+2.5+2+1)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>10 ST(2) ≦ ST(3)</td>
<td>9 ≦ 6</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>9 ST(2) &lt; ST(3)+ST(4)</td>
<td>9 &lt; (6+4)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>8 ST(2) &lt; ST(3)+ST(4)+ST(5)</td>
<td>9 &lt; (6+4+2.5)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>7 ST(2) &lt; ST(3)+ST(4)+ST(5)+ST(6)</td>
<td>9 &lt; (6+4+2.5+2)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>6 ST(2) &lt; ST(3)+ST(4)+ST(5)+ST(6)+ST(7)</td>
<td>9 &lt; (6+4+2.5+2+1)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>5 ST(1) ≦ ST(2)+ST(3)</td>
<td>18 ≦ (9+6)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>4 ST(1) &lt; ST(2)+ST(3)+ST(4)</td>
<td>18 &lt; (9+6+4)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>3 ST(1) &lt; ST(2)+ST(3)+ST(4)+ST(5)</td>
<td>18 &lt; (9+6+4+2.5)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>2 ST(1) &lt; ST(2)+ST(3)+ST(4)+ST(5)+ST(6)</td>
<td>18 &lt; (9+6+4+2.5+2)</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>1 ST(1) &lt; ST(2)+ST(3)+ST(4)+ST(5)+ST(6)+ST(7)</td>
<td>18 &lt; (9+6+4+2.5+2+1)</td>
<td>OK</td>
<td></td>
</tr>
</tbody>
</table>

Figure 23. Churchman - Ackoff Ranking Evaluation for Threat Schemes

Based on the analysis in Figure 23, the revised rankings are as
follows: ST(1) = 18; ST(2) = 9; ST(3) = 6; ST(4) = 4; ST(5) = 2.5; ST(6) = 2 and ST(7) = 1.

These values form the basis for calculating threat values for the computer fraud schemes in Figure 21. However, prior to developing these values, the perpetrators identified in Figure 21 will be ranked in the same manner as the schemes were in the analysis above. Assume that the perpetrator threats (PT) from Figure 21 have been ranked in the following order where PT(1) represents the greatest threat and PT(7) represents the least threat.

PT(1) = Data Entry/Terminal Operator
PT(2) = Officer/Manager
PT(3) = Programmer
PT(4) = Clerk/Teller
PT(5) = Other Staff
PT(6) = Outsider (non-employee)
PT(7) = Computer Operator

Now assume that the series of comparisons depicted in Figure 22 work out as follows when applied to the perpetrator threats:

1. PT(1) ≤ PT(2)+PT(3)+PT(4)+PT(5)+PT(6)+PT(7)
2. PT(1) ≤ PT(2)+PT(3)+PT(4)+PT(5)+PT(6)
3. PT(1) ≤ PT(2)+PT(3)+PT(4)+PT(5)
4. PT(1) ≤ PT(2)+PT(3)+PT(4)
5. PT(1) ≤ PT(2)+PT(3)
6. PT(2) ≤ PT(3)+PT(4)+PT(5)+PT(6)+PT(7)
7. PT(2) ≤ PT(3)+PT(4)+PT(5)+PT(6)
8. PT(2) ≤ PT(3)+PT(4)+PT(5)
9. PT(2) ≤ PT(3)+PT(4)
10. PT(2) ≤ PT(3)
11. PT(3) ≤ PT(4)+PT(5)+PT(6)+PT(7)
12. PT(3) ≤ PT(4)+PT(5)+PT(6)
13. PT(3) ≤ PT(4)+PT(5)
14. PT(4) ≤ PT(5)+PT(6)+PT(7)
15. PT(4) ≤ PT(5)+PT(6)
16. PT(4) ≤ PT(5)
17. PT(5) ≤ PT(6)+PT(7)
18. PT(5) ≤ PT(6)
19. PT(6) ≤ PT(7)

The next step is to assign specific preliminary weights to each
perpetrator threat. Assume that the following weights have been assigned:

\[
\begin{align*}
&PT(1) = 20 \\
&PT(2) = 15 \\
&PT(3) = 9 \\
&PT(4) = 6 \\
&PT(5) = 4 \\
&PT(6) = 3 \\
&PT(7) = 1 \\
\end{align*}
\]

These preliminary weights are plugged into the series of comparisons and necessary revisions made in Figure 24.

<table>
<thead>
<tr>
<th>Equation</th>
<th>Evaluation-Based on Assigned weights</th>
<th>Results</th>
<th>Revision</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20 PT(6) = PT(7)</td>
<td>3 = 1</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19 PT(5) = PT(6)</td>
<td>4 = 3</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18 PT(5) &lt; PT(6)+PT(7)</td>
<td>4 &lt; (3+1)</td>
<td>ERROR</td>
<td>Let PT(5) = 3.5</td>
<td></td>
</tr>
<tr>
<td>17 PT(4) = PT(5)</td>
<td>6 = 3.5</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 PT(4) &lt; PT(5)+PT(6)</td>
<td>6 &lt; (3.5+3)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 PT(4) &lt; PT(5)+PT(6)+PT(7)</td>
<td>6 &lt; (3.5+3+1)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 PT(3) = PT(4)</td>
<td>9 = 6</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 PT(3) &lt; PT(4)+PT(5)</td>
<td>9 &lt; (6+3.5)</td>
<td>ERROR</td>
<td>Let PT(3) = 9.5</td>
<td></td>
</tr>
<tr>
<td>12 PT(3) &lt; PT(4)+PT(5)+PT(6)</td>
<td>9.5 &lt; (6+3.5+3)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 PT(3) &lt; PT(4)+PT(5)+PT(6)</td>
<td>9.5 &lt; (6+3.5+3+1)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 PT(2) = PT(3)</td>
<td>15 &gt;= 9.5</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9  PT(2) &lt; PT(3)+PT(4)</td>
<td>15 &lt; (9.5+6)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8  PT(2) &lt; PT(3)+PT(4)+PT(5)</td>
<td>15 &lt; (9.5+6+3.5)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7  PT(2) &lt; PT(3)+PT(4)+PT(5)+PT(6)</td>
<td>15 &lt; (9.5+6+3.5+3)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6  PT(2) &lt; PT(3)+PT(4)+PT(5)+PT(6)+PT(7)</td>
<td>15 &lt; (9.5+6+3.5+3+1)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5  PT(1) = PT(2)+PT(3)</td>
<td>20 &gt;= (15+9.5)</td>
<td>ERROR</td>
<td>Let PT(1) = 24.5</td>
<td></td>
</tr>
<tr>
<td>4  PT(1) &lt; PT(2)+PT(3)+PT(4)</td>
<td>24.5 &lt; (15+9.5+6)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3  PT(1) &lt; PT(2)+PT(3)+PT(4)+PT(5)</td>
<td>24.5 &lt; (15+9.5+6+3.5)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2  PT(1) &lt; PT(2)+PT(3)+PT(4)+PT(5)+PT(6)</td>
<td>24.5 &lt; (15+9.5+6+3.5+3)</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1  PT(1) &lt; PT(2)+PT(3)+PT(4)+PT(5)+PT(6)+PT(7)</td>
<td>24.5 &lt; (15+9.5+6+3.5+3+1)OK</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 24. Churchman-Ackoff Ranking Evaluation for Perpetrators
Based on the analysis in Figure 24, the revised rankings are as follows: PT(1)=24.5; PT(2)=15; PT(3)=9.5; PT(4)=6; PT(5)=3.5; PT(6)=3 and PT(7)=1.

The PT values from above and the ST values from Figure 23 are used to derive threat values for the threat matrix shown in Figure 21. It should be noted that the analysis from this point on departs from the Churchman-Ackoff method to form a unique method for developing threat values beginning with the PT and ST values which were derived using Churchman-Ackoff.

Derivation of Threat Values from Churchman-Ackoff

Threat values are derived from the above Churchman-Ackoff derived PT and ST values as follows:

1. Calculate normalized relative averages for ST and PT values.

2. Place relative averages in their respective position in the threat matrix.

3. Calculate products of ST and PT relative averages and place in corresponding matrix cells to represent scheme/perpetrator threats.

<table>
<thead>
<tr>
<th>ST Values</th>
<th>Relative Averages</th>
<th>PT Values</th>
<th>Relative Averages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ST(i)/Total ST, Where i = 1,7</td>
<td>PT(i)/Total PT, Where i = 1,7</td>
<td></td>
</tr>
<tr>
<td>ST(1) = 18</td>
<td>18/42.5 = .424</td>
<td>PT(1)=24.5</td>
<td>24.5/62.5 = .392</td>
</tr>
<tr>
<td>ST(2) = 9</td>
<td>9/42.5 = .212</td>
<td>PT(2)=15</td>
<td>15/62.5 = .240</td>
</tr>
<tr>
<td>ST(3) = 6</td>
<td>6/42.5 = .141</td>
<td>PT(3)= 9.5</td>
<td>9.5/62.5 = .152</td>
</tr>
<tr>
<td>ST(4) = 4</td>
<td>4/42.5 = .094</td>
<td>PT(4)= 6</td>
<td>6/62.5 = .096</td>
</tr>
<tr>
<td>ST(5) = 2.5</td>
<td>2.5/42.5 =.059</td>
<td>PT(5)= 3.5</td>
<td>3.5/62.5 = .056</td>
</tr>
<tr>
<td>ST(6) = 2</td>
<td>2/42.5 = .047</td>
<td>PT(6)= 3</td>
<td>3/62.5 = .048</td>
</tr>
<tr>
<td>ST(7) = 1</td>
<td>1/42.5 = .024</td>
<td>PT(7)= 1</td>
<td>1/62.5 = .016</td>
</tr>
<tr>
<td>TOTAL</td>
<td>42.5</td>
<td>1.000</td>
<td>TOTAL 62.5</td>
</tr>
</tbody>
</table>

Figure 25. Normalized Relative Averages of ST and PT Values
To illustrate, the above three steps are completed in Figures 25 and 26. Step one is necessary in order to normalize the ST and PT values in order that they may be multiplied in step 3 to provide threat values for the scheme/perpetrator combinations represented by the cells of the matrix.

<table>
<thead>
<tr>
<th>SCHEMES</th>
<th>ST(1)</th>
<th>ST(4)</th>
<th>ST(5)</th>
<th>ST(2)</th>
<th>ST(3)</th>
<th>ST(7)</th>
<th>ST(6)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trans.</td>
<td>Trans.</td>
<td>Trans.</td>
<td>File</td>
<td>Program</td>
<td>Improper</td>
<td>Communication</td>
</tr>
<tr>
<td></td>
<td>Added</td>
<td>Altered</td>
<td>Deleted</td>
<td>Changes</td>
<td>Changes</td>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>Entry</td>
<td>.166</td>
<td>.083</td>
<td>.055</td>
<td>.037</td>
<td>.023</td>
<td>.018</td>
</tr>
<tr>
<td>E</td>
<td>Terminal Operator</td>
<td>.102</td>
<td>.051</td>
<td>.034</td>
<td>.022</td>
<td>.014</td>
<td>.011</td>
</tr>
<tr>
<td>R</td>
<td>Clerk/ Teller</td>
<td>.064</td>
<td>.032</td>
<td>.021</td>
<td>.014</td>
<td>.009</td>
<td>.007</td>
</tr>
<tr>
<td>P</td>
<td>Programmer</td>
<td>.04</td>
<td>.020</td>
<td>.014</td>
<td>.009</td>
<td>.006</td>
<td>.005</td>
</tr>
<tr>
<td>E</td>
<td>Officer/ Manager</td>
<td>.024</td>
<td>.012</td>
<td>.008</td>
<td>.005</td>
<td>.003</td>
<td>.003</td>
</tr>
<tr>
<td>R</td>
<td>Computer Operator</td>
<td>.02</td>
<td>.010</td>
<td>.007</td>
<td>.005</td>
<td>.003</td>
<td>.002</td>
</tr>
<tr>
<td>T</td>
<td>Other Staff</td>
<td>.007</td>
<td>.003</td>
<td>.002</td>
<td>.002</td>
<td>.001</td>
<td>.001</td>
</tr>
<tr>
<td>R</td>
<td>Outsider (Non-employee)</td>
<td>.424</td>
<td>.212</td>
<td>.141</td>
<td>.094</td>
<td>.059</td>
<td>.047</td>
</tr>
</tbody>
</table>

Figure 26. Derivation of Threat Values

It should be noted that the threat values for the
scheme/perpetrator combinations (matrix cells) are calculated under the assumption that no controls exist. This is consistent with the model in Figure 8 which indicates that threat analysis is performed prior to controls analysis.

<table>
<thead>
<tr>
<th>PT(1)</th>
<th>PT(4)</th>
<th>PT(3)</th>
<th>PT(2)</th>
<th>PT(7)</th>
<th>PT(5)</th>
<th>PT(6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Entry Operator</td>
<td>Clerk/Teller</td>
<td>Programmer</td>
<td>Officer/Manager</td>
<td>Computer Operator</td>
<td>Other Staff</td>
<td>Outsider (Non-employee)</td>
</tr>
<tr>
<td>Trans. Added</td>
<td>16.6</td>
<td>10.2</td>
<td>6.4</td>
<td>4.0</td>
<td>2.4</td>
<td>.7</td>
</tr>
<tr>
<td>Trans. Altered</td>
<td>8.3</td>
<td>5.1</td>
<td>3.2</td>
<td>2.0</td>
<td>1.2</td>
<td>.3</td>
</tr>
<tr>
<td>Trans. Deleted</td>
<td>5.5</td>
<td>3.4</td>
<td>2.1</td>
<td>1.4</td>
<td>.8</td>
<td>.2</td>
</tr>
<tr>
<td>File Changes</td>
<td>3.7</td>
<td>2.2</td>
<td>1.4</td>
<td>.9</td>
<td>.5</td>
<td>.2</td>
</tr>
<tr>
<td>Program Changes</td>
<td>2.3</td>
<td>1.4</td>
<td>.9</td>
<td>.6</td>
<td>.3</td>
<td>.1</td>
</tr>
<tr>
<td>Improper Operation</td>
<td>1.8</td>
<td>1.1</td>
<td>.7</td>
<td>.5</td>
<td>.2</td>
<td>.1</td>
</tr>
<tr>
<td>Communication</td>
<td>.9</td>
<td>.6</td>
<td>.4</td>
<td>.2</td>
<td>.1</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 27. Threat Values from Figure 26 Multiplied by 100

The threat values in Figure 26 are multiplied by 100 in Figure 27. This is accomplished without changing the meaning of the threat matrix since it is the relative magnitude of the threat values rather
than their absolute values which is important. The reasons for multiplying the values in Figure 26 by 100 are: to minimize the number of fractions that must be used in further analysis; and to present the threat values in a form which more closely resembles those in the General Threat Assessment in Chapter 3 (Figure 12).

At this point it should be possible to determine whether the threat level is acceptable for a given system. Referring back to Figure 8, recall that, if the threat level is acceptable, no further analysis is needed. If it is not, then it is necessary to perform Controls Analysis for each specific threat for which an unacceptable threat exists.

Controls Analysis

The purpose of Controls Analysis is to identify and evaluate the organizational and systems controls which are being used.

The suggested approach for performing the Controls Analysis is similar to the small group or brainstorming approaches described above for Threat Analysis. The first step is to identify all controls which are in effect for a given system. In accomplishing the first step of Controls Analysis the group might refer to a published list of controls (they are plentiful in the literature) and develop a subset of controls in effect within the system being evaluated. However, such an approach assumes the comprehensiveness of the published list, which may not be an accurate assumption. A preferred approach is for the group to develop a list of controls on their own. When the group, after several iterations, can think of no additional controls, a review of published
lists might be beneficial as a vehicle for identifying additional controls. By approaching it in this manner, the analysis is not limited to the contents of published lists which may not be comprehensive for a given system but may be beneficial in augmenting the group effort.

Once the group is satisfied that all significant controls have been identified, the controls should be correlated with the cells in the threat matrix. One method for accomplishing this is to number the controls from 1 to N where N = the total number of controls, then record the numbers of the controls which protect against specific threats in the corresponding matrix cells, similar to Fitzgerald's approach (Fitzgerald 1978). For example, assume that the first control limited access to computer programs to computer programmers only. This control would be given the number 1 and this number would be correlated with each cell in the matrix for which it provides protection. Referring to Figure 21 it is evident that the number 1 would appear in all cells under the column heading "Program Changes" with the exception of the one intersecting with row entitled "Programmer". This process would be continued until all controls have been exhausted.

The next phase of Controls Analysis is the evaluation of the controls which have been correlated with the threats in the matrix cells. Ideally, by reviewing the controls associated with each cell, a numeric confidence level of fraud prevention could be assigned. Thus, by reviewing controls associated with matrix cell 1,1 (Transactions Added, Data Entry/Terminal Operator") for example, it would be possible to state with some specific level of confidence that fraud will be prevented (e.g., "If these controls are used a 95% probability of
preventing fraud is assured”). Unfortunately, intentional acts such as computer fraud, are not amenable to such quantification, generally.

However, a small group of experts using an approach such as "Delphi" should be capable of subjectively categorizing the controls within each matrix cell according to their aggregate strength. The categorizations in Figure 28 or a similar set should be adequate for purposes of the Controls Analysis.

<table>
<thead>
<tr>
<th>Strength of Controls</th>
<th>Point Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>0</td>
</tr>
<tr>
<td>Very Strong</td>
<td>.1</td>
</tr>
<tr>
<td>Strong</td>
<td>.3</td>
</tr>
<tr>
<td>Moderate</td>
<td>.4</td>
</tr>
<tr>
<td>Weak</td>
<td>.6</td>
</tr>
<tr>
<td>Very Weak</td>
<td>.8</td>
</tr>
<tr>
<td>Nonexistent</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Figure 28. Control Point Values

It is not necessary that the group adopt the above categorizations or point values. Rather, the group should use whatever categorization scheme its members are most comfortable with. The scheme should, however, lend itself to further correlation and analysis with specific threats in the threat matrix cells.

The categorization scheme in Figure 28 lends itself quite well to further correlation and analysis with the threats in the matrix cells since "Total" protection with a point value of zero, when combined in a multiplicative fashion with the threats in the matrix cells, totally eliminates the threat value. "Nonexistent" protection with a point value of one, on the other hand, leaves the threat value
at full strength when combined multiplicatively. The various levels of protection between "Total" and "Nonexistent", when combined multiplicatively with the threats in the matrix cells, result in increasing "post-control" threat values.

<table>
<thead>
<tr>
<th>SCHEMES</th>
<th>Transactions Added</th>
<th>Transactions Altered</th>
<th>Transactions Deleted</th>
<th>File Changes</th>
<th>Program Changes</th>
<th>Improper Operation</th>
<th>Communications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Entry/ Terminal Operator</td>
<td>16.6 (.4)</td>
<td>8.3 (.3)</td>
<td>5.5 (.5)</td>
<td>3.7 (.5)</td>
<td>2.3 (0)</td>
<td>1.8 (0)</td>
<td>.9 (1.0)</td>
</tr>
<tr>
<td>Clerk/Teller</td>
<td>10.2 (.3)</td>
<td>5.1 (.3)</td>
<td>3.4 (.6)</td>
<td>2.2 (0)</td>
<td>1.4 (0)</td>
<td>1.1 (0)</td>
<td>1.6 (0)</td>
</tr>
<tr>
<td>Programmer</td>
<td>6.4 (0)</td>
<td>3.2 (0)</td>
<td>2.1 (0)</td>
<td>1.4 (0)</td>
<td>.9 (0)</td>
<td>.7 (0)</td>
<td>.4 (0)</td>
</tr>
<tr>
<td>Officer/Manager</td>
<td>4.0 (.8)</td>
<td>2.0 (.8)</td>
<td>1.4 (.8)</td>
<td>.9 (.5)</td>
<td>.6 (0)</td>
<td>.5 (0)</td>
<td>.2 (0)</td>
</tr>
<tr>
<td>Computer Operator</td>
<td>2.4 (0)</td>
<td>1.2 (0)</td>
<td>.8 (.8)</td>
<td>.5 (.8)</td>
<td>.3 (0)</td>
<td>.3 (.9)</td>
<td>.1 (0)</td>
</tr>
<tr>
<td>Other Staff</td>
<td>2.0 (.5)</td>
<td>1.0 (.5)</td>
<td>.7 (.5)</td>
<td>.5 (0)</td>
<td>.3 (0)</td>
<td>.2 (0)</td>
<td>.1 (0)</td>
</tr>
<tr>
<td>Outsider (Non-employee)</td>
<td>.7 (0)</td>
<td>.3 (0)</td>
<td>.2 (0)</td>
<td>.2 (0)</td>
<td>.1 (0)</td>
<td>.1 (0)</td>
<td>.1 (0)</td>
</tr>
</tbody>
</table>

Figure 29. Combined Threat/Control Matrix

To illustrate, for each cell in the threat matrix in Figure 27, assume that the controls associated with the threat identified in that cell have been evaluated and placed into one of the above categories, resulting in the combined threat/control matrix in Figure 29. The
first number in each cell in the matrix is the threat value from Figure 27. The second number in each cell (in parenthesis) represents the point value from Figure 28 which best depicts the aggregate strength of the controls associated with that cell. Thus, for cell 1,1 - Transactions Added by Data Entry/Terminal Operators, the controls are considered "Moderate" with a point value of .4 shown parenthetically immediately below the threat value of 16.6 extracted from the threat matrix in Figure 27.

The final step in Controls Analysis is to multiply the threat values and control point values to determine "post-control" threat values. Note that for the above example the threat value of 16.6 multiplied by the control point value of .4 results in a "post-control" value of 6.64.

"Post-control" threat values are computed for each of the matrix cells in Figure 29 by multiplying the threat values in each cell by the control point values in the same cell. The results of this operation are shown in Figure 30.

The Post Control Threat Matrix in Figure 30 is now ready to be used in developing a Detection Quotient in the same manner demonstrated in Chapter 4 for the General Threat Matrix in Chapter 3 (Figure 12).

Since the Detection Quotient methodology is discussed throughly in Chapter 4 and applies equally to the General Threat Matrix and Detailed Threat Matrix it will not be repeated here. However, it should be noted that in the Post-Control Threat Matrix in Figure 30 many of the threats fall out (cells with zeros).
<table>
<thead>
<tr>
<th>SCHEMES</th>
<th>Transactions Added</th>
<th>Transactions Altered</th>
<th>Transactions Deleted</th>
<th>File Changes</th>
<th>Program Changes</th>
<th>Improper Operation</th>
<th>Communications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Entry/ Terminal Operator</td>
<td>6.64</td>
<td>2.49</td>
<td>2.75</td>
<td>1.85</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Clerk/ Teller</td>
<td>3.06</td>
<td>1.53</td>
<td>2.04</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Programmer</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.72</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Officer/ Manager</td>
<td>3.2</td>
<td>1.6</td>
<td>1.12</td>
<td>.45</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Computer Operator</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.4</td>
<td>0</td>
<td>.27</td>
<td>0</td>
</tr>
<tr>
<td>Other Staff</td>
<td>1.0</td>
<td>.5</td>
<td>.35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Outsider (Non-employee)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 30. Post-Control Threat Matrix
CHAPTER 6

THE COMBINATORIAL DILEMMA - A GENERAL SOLUTION

Research or analysis often leads to the evaluation of various alternatives. Typically, the objective of this evaluation is to determine an optimum or near optimum alternative, based on some measurable criteria.

The combinatorial dilemma occurs when, due to the phenomenally large number of possible combinations, it is not practical to evaluate them all. The combinatorial dilemma is well known, however, is explained below to develop a perspective prior to presenting a solution. Although it may be intuitively apparent that certain combinatorial situations will result in large numbers of combinations, the magnitude may not be readily apparent. Consider for example, the number of possible deals in a game of bridge. Then, given the hands N, E, S, and W determine a partition of the 52 cards having four "cells", each with 13 elements. It is rather apparent that there are a large number of possible deals or combinations in this example. Actually, there are \[ \frac{52!}{13!13!13!13!} \] possible deals or combinations. Recalling that \( N! = ((N)(N-1)...(1)) \), this equates to \( 5.3645.10^{28} \), or approximately 54 billion billion billion billion deals (Kemeny, Snell, Thompson 1966).

Now assume that an algorithm has been developed to determine, based on some measurable criteria, the optimum possible deal by evaluating every possible combination. It should be apparent that a
manual evaluation would be prohibitive. Even a modest one minute
evaluation for each hand would take over one hundred thousand billion
years. However, with the speed of today's computers, it is highly
unlikely that a manual solution would be seriously considered.

Rather than performing the evaluation manually, assume that a
computerized evaluation requiring one nanosecond (one billionth of a
second) per hand has been developed. The computer could perform the
evaluations in an impressive .0167 billionth of the time required to
perform them manually or approximately 1.7 trillion years. Both the
manual and automated solutions are clearly infeasible in this example,
thus the combinatorial dilemma applies.

Another example which is more relevant to the world of business
is based on a marketing and pricing model for an electronic components
manufacturing firm (Dunn 1971). For purposes of illustration, assume
that ABC Incorporated manufactures electronic components such as
transistors, resistors, etc. For illustrative purposes, assume that
the plant (building, personnel, etc.) is fixed, at least in the short
term.

Now assume that ABC can manufacture any of 12 different
products up to the following volumes (e.g., production of the volumes
shown in the second column for any one product would totally utilize
ABC's production facilities):

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>MAX WHICH MAY BE PRODUCED IN EXISTING PLANT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,000</td>
</tr>
<tr>
<td>2</td>
<td>5,000</td>
</tr>
<tr>
<td>3</td>
<td>1,500</td>
</tr>
<tr>
<td>4</td>
<td>10,000</td>
</tr>
<tr>
<td>5</td>
<td>8,000</td>
</tr>
<tr>
<td>6</td>
<td>6,000</td>
</tr>
</tbody>
</table>
The decision which must be made in this example is how many units of each product should be produced to optimize the use of ABC's plant. The problem is further complicated by the fact that the demand for each product is affected by the price that ABC charges for it. Thus, each product has an elastic demand curve similar to that shown in Figure 31.

Further, each different product at each possible price results in a different contribution to ABC's profits. The primary decision may be restated as "what price should be charged for each product in order to drive demand for each to a level which will optimize the use of ABC's plant as measured by total contribution to profits".

Given that the reasonable price ranges for the various products are measured in dollars, there could be hundreds of different prices for each product since the alternative prices could vary by only a few
cents. More than likely, however, the number of prices to be considered for each product could be limited to a more reasonable number, for example 10 to 15 discrete prices. For purposes of illustration, assume that ABC's marketing department has developed a demand curve for each product based on 12 different prices per product.

The objective is to determine that combination of prices and the resulting demand levels for each product which optimizes the use of ABC's plant as measured by the profit potential. To fully evaluate all possible combinations of products and prices would require as many as $12^{12}$, or nine trillion combinations.

Assuming it takes only a few nanoseconds of central processor (CPU) time on a large, high speed computer to calculate the potential profit for each combination, a total evaluation would require several hours of CPU time. For example, if the evaluation of each alternative took four nanoseconds, approximately ten hours of CPU time would be required. Although ten hours of CPU is certainly more reasonable than the 1.7 trillion years in the bridge example, it still represents an expensive use of computer time.

Consider, however, that an alternative methodology were available which, by evaluating only three thousand combinations, would ensure with a 99+ percent confidence, a solution which, based on the measurable criteria, is equal or superior to 99.8 percent of the possible combinations. Using the .05 second example from above, this methodology would require only 150 seconds of CPU time. Unless there are some strange patterns in the data, a combination in the top .2 percent would probably very closely approximate the savings of a total
evaluation.

The methodology in this chapter is designed to provide such a capability. The only real limitation is in the case of unusual patterns or outliers in the data. For instance, if in the nine trillion possible combinations facing ABC there were one or two that were considerably more profitable than all of the others, the solution reached by the methodology, although within the top few percentile of possible solutions, might still be a considerable distance from the best solution. This situation parallels the "needle in the haystack" situation discussed in Chapter 4 on "Discovery Sampling".

If it is suspected that this condition exists, a decision would have to be made as to whether the potential superiority of the outlier solutions would justify the cost of a total evaluation.

In some cases the total evaluation may not be a feasible alternative regardless of the pattern of the data. For example, in the bridge example above, it just isn't feasible to perform a total evaluation requiring 1.7 trillion years.

**A General Solution To The Combinatorial Dilemma**

A general solution to the combinatorial dilemma will be presented in this chapter and cited in Chapter 7 in The Computer Fraud Detection Resource Optimization Model. The solution is based upon an iterative discovery sampling technique developed by Dunn (1971). The approach begins with Arkin's (1967) Discovery (Exploratory) Sampling technique. Recall from Chapter 4 that the purpose of discovery sampling is to disclose evidence of some activity, usually some...
irregularity within a system. Further, the disclosure can be assured with statistically defensible confidence levels if relatively small samples from the population are examined.

For the solution to the combinatorial dilemma the activity or irregularity will be defined to be a combination which, based on some measurable criteria, is superior to all others observed to that point. The process is to iteratively take additional samples until no improvements are discovered in one entire sample. At that point it is possible to state with a confidence level corresponding with the sample size that the combination which could not be beaten in a total sample is within some specific percentile of possible combinations. For example, by taking the appropriate number of samples of a given size, it is possible to assure with a 99.9 percent confidence level that a given combination, unbeatable for one total sample, is within the top .2 percent of all possible solution, etc.

The technique follows the general logic in Figure 32. In the first block the parameters are established. Recall, from Chapter 4 that for varying field sizes, different sample sizes will produce at least one occurrence of an event with some specific degree of confidence. Referring back to the example in Chapter 4, a sample of 2,000 from a field of 10,000, produced a 98.9 percent confidence of finding at least one occurrence of a given irregularity or event. The decision which must be made in block one is how close to the one-hundredth percentile of possible solutions to strive for and with what confidence level. For example, if there are 200,000 possible combinations and a combination in the top .002 of all possible
1. Set Parameters: Confidence Level, Percentile and Sample Size

2. Select A "Combination to Beat"

3. Take a Random Sample of Possible Combinations

4. Compare A Combination In Sample to "Combination to Beat"

5. Is Combination From Sample Superior?
   - YES
   - NO

6. Establish Combination From Sample As "Combination to Beat"

7. Have All Sample Combinations Been Compared?
   - NO
   - YES

8. Compare Next Combination From Sample to "Combination to Beat"

9. Solution

Figure 32. Combinatorial Dilemma Solution Logic
combinations is desired with a confidence of 99.8 percent, a sample size of 3,000 would be required. The parameters referred to in block 1 would thus be set at:

1. Confidence Level = 99.9%
2. Percentile = 99.8%
3. Sample Size = 3,000

Once the desired confidence level, percentile and corresponding sample size have been determined, the remainder of the logic in Figure 32 is oriented to seeking a near optimum combination as defined by the parameters from block one of the flowchart.

As shown in the second block in Figure 32, a "Combination to Beat" is selected. Although any combination could be selected for this purpose, a suggested approach is to run through one complete random sample taking the best combination from that sample as the starting "Combination to Beat". Thus, the starting "Combination to Beat" should be a relatively tough combination to beat, possibly cutting down on the total number of samples that must be taken before a combination is found which is unbeatable when compared to each combination in one complete random sample. The reason for this is that each time a combination from the current random sample is found to be superior to the existing "Combination to Beat", this combination is established as the new "Combination to Beat" and a new random sample is selected. Beginning with an easy "Combination to Beat" will probably result in several random samples being chosen in order to converge to a combination equal or superior to that chosen by evaluating one complete random sample and choosing the best combination from that sample.
Once a starting "Combination to Beat" has been selected, a random sample of combinations is taken as shown in block three. Each combination from this sample is then compared to the "Combination to Beat" until either a superior combination from the sample is found or all combinations from the sample have been compared without finding one superior to the "Combination to Beat". If, as shown in blocks five and six of Figure 32, a combination from the sample is superior to the "Combination to Beat", it is established as the new "Combination to Beat" and a new random sample is selected against which to compare it.

As shown in blocks seven and eight, additional combinations from the sample are compared to the existing "Combination to Beat" until either a combination from the sample beats the existing "Combination to Beat" or a complete random sample has been compared without finding one which is superior.

Once a complete random sample has been compared to the existing "Combination to Beat" without finding any combinations which beat it, a solution has been found as shown in block nine. Since a complete sample did not reveal any combinations superior to the existing "Combination to Beat" it may be stated with the confidence level from block one that the unbeatable combination is within the corresponding percentile of possible solutions. Another way of interpreting the results is that the nondiscovery of a superior combination in one complete random sample ensures that the rate of occurrence of superior combinations is no greater than some given percentage with a given confidence level. Thus, for example, it might be said as in the example cited earlier that a confidence level of 99.8 percent can be
assured that a solution is within the top 99.8 percent of possible solutions or, conversely, with the same confidence level that the rate of occurrence of superior combinations is no greater than .002.

The process described above forms the basis for the Resource Optimization Model in Chapter 7, which may be used to produce a near optimum utilization of resources in the detection of computer fraud for computer systems.
CHAPTER 7

RESOURCE OPTIMIZATION MODEL

The objective of the Resource Optimization Model in this chapter is to optimize the utilization of available resources in the detection of computer fraud as measured by the Detection Quotient described in Chapter 4. Recall that the Detection Quotient for a given system is the sum of the Detection Quotients for each threat in the system. As illustrated in Chapter 4, Figure 15, the Detection Quotient for each threat is a function of three factors; The numerical threat value of each threat (T); the probability of detecting at least one occurrence of fraud (P); and the converse of the rate of occurrence (C).

In Figure 15 the Detection Quotients (DQ's) for the 21 major threats identified during the research for this thesis, covered in Chapter 3, are calculated using the formula DQ = TPC. For illustrative purposes in Figure 15, a "P" value of .9 and a "C" value of .995 are assumed for each of the 21 threats. The underlying assumption is that investigative resources are available to ensure, for each of the 21 threats, a 90 percent probability of detecting at least one occurrence of fraud if the rate of occurrence is at least .5 percent.

In actuality, the "P" and "C" values may vary from one threat to another. Further, given the same expenditure level of investigative resources, the detection level, as measured by the DQ value, may vary
considerably from one "P" and "C" combination to another. The configuration in Figure 15, whereby all "P" and "C" values are .9 and .995 respectively, is only one of a very large number of possible combinations. This combination, chosen arbitrarily for illustrative purposes, may represent a very poor utilization of resources since any number of the other possible "P" and "C" combinations may be superior with resulting DQ's greater than the DQ value of 242.8 shown in Figure 15.

The magnitude of the possible number of combinations of "P" and "C" values precludes examination of them all, as will be demonstrated later in this chapter. The Combinatorial Dilemma referred to in Chapter 6 applies, since it is not feasible to examine all possible "P" and "C" combinations in order to ensure the optimum use of detection resources.

The purpose of the Resource Optimization Model is to seek a near optimum combination of "P" and "C" combinations while only examining a fraction of the possible combinations for given levels of investigative resources.

In order that the analysis may be more easily relatable to specific systems, the hypothetical system in Chapter 5, "Specific Threat Assessment", will be used to demonstrate the Resource Optimization Model. While the threats in Figure 15 could be used to demonstrate the model, it should be more meaningful to use the hypothetical "specific" system in Chapter 5 since risk assessment and computer fraud detection must ultimately be applied to specific systems.
Mathematical Statement of the Problem

The problem of optimizing the utilization of available resources in the detection of computer fraud is similar to the general problem of mathematical programming. The general problem in mathematical programming is to find the values of some variables which will optimize (i.e., maximize or minimize) the value of the objective function subject to a set of constraints. The mathematical programming problem can be formulated in the following general form (Kwak 1973):

Maximize (or Minimize)

\[ F = \sum_{j=1}^{n} C_j x_j \]

Subject to

\[ \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \text{(for } j=1,2,\ldots,m) \]

\[ x_j \geq 0 \quad \text{(for } j=1,2,\ldots,n) \]

and Where

\[ F = \text{Value of the objective function which measures the effectiveness of the decision.} \]

\[ x_j = \text{Variables that are subject to the control of the decision maker.} \]

\[ c_j = \text{Unit profit contribution of a product or unit cost of an input which is known.} \]

\[ a_{ij} = \text{Production (or technical) coefficients that are known.} \]

\[ b_i = \text{Available productive resources in limited supply} \]

Objective Function

The objective function is a mathematical equation describing a functional relationship between various decision variables and the outcome of the decisions. For the Resource Optimization Model in this
chapter the objective function may be stated, mathematically, as:

\[
\text{Maximize } D = \sum_{j=1}^{n} T_j P_j C_j
\]

Where

- \( D \) = Total Detection Quotient for a System
- \( t_j \) = Threat Value
- \( P_j \) = Probability of Finding at Least One Occurrence of Computer Fraud.
- \( C_j \) = Converse of Rate of Occurrence of Computer Fraud.
- \( n \) = Total Number of Threats Identified for a System.

Referring back to Figure 15 in Chapter 4, recall that the detection quotient for a given threat within a system is determined by the equation:

\[
\text{DQ} = \text{TPC}
\]

Where

- \( T \) = The "Threat" value for a given threat.
- \( P \) = The probability of detecting at least one occurrence of fraud for the given threat.
- \( C \) = The converse of the rate of occurrence for the given threat.

To illustrate, take the upper left threat shown in matrix position 1,1 of Figure 30, "Transactions Added - Data Entry/Terminal Operator". The threat value, "T" for this threat is 6.64. The detection quotient for this threat may be determined by setting "P" and "C" values between 0 and 1 and applying the formula: \( \text{DQ} = \text{TPC} \).

Assume for the moment that management requires, for this given threat a 99.99 percent probability of detecting at least one occurrence of fraud if it occurs with a rate of .02 percent. The "P" value is,
thus set at .9999 and the "C" value is set at .9998 (converse of the rate of occurrence is 1-.0002 = .9998). The DQ in this example may be calculated as:

$$DQ = TPC$$

$$= (6.64)(.9999)(.9998)$$

$$= 6.6377$$

As shown in this example, as the "P" and "C" values approach one, the DQ value approaches the threat value (T) for a given threat.

Given unlimited resources, the solution to the resource allocation problem is trivial. The "P" and "C" values would be set very near one; the resulting DQ values would approximate their corresponding threat values and the total DQ value for a system would be maximized. Resource allocation would be a matter of simply assigning enough resources to achieve the "P" and "C" values. It should be reiterated that, even with a one hundred percent investigation, it may not be possible to find the "needle in the haystack" (e.g. one instance of fraud in 10 million transactions), because of inefficiencies in investigative techniques. Thus, "P" can typically only approach, rather than equal one. For "C" to equal one, the rate of occurrence of fraud would have to be zero. This becomes nonsensical since, for a given "P" value, a "C" value of one suggests that, with the probability of "P", at least one occurrence of fraud will be detected if the rate of occurrence is zero. This is, of course, impossible for all "P" values except zero since there is no probability of detecting an occurrence if none exists.

As indicated previously, the one hundred percent investigation
is typically not feasible. In fact, the typical organization will probably have, or be willing to expend, only a small percentage of the resources required for this type of investigation. In this environment of limited resources, the problem of setting "P" and "C" values for the various systems threats and the associated levels of investigative resources becomes a complex resource allocation problem. As will be shown, the "Combinatorial Dilemma" applies since a phenomenally large number of possible combinations of "P" and "C" values must typically be considered.

First, the objective function for the threat matrix in Figure 30 will be developed to demonstrate the process. The first step is to number the threats in the threat matrix. Cells with threat values of zero are skipped since they represent zero threat combinations. This has been accomplished in Figure 33.

The objective function is to maximize the total detection quotient for the system represented by the threat matrix in Figure 30, or mathematically:

\[
\text{Maximize } D = \sum_{j=1}^{n} T_j P_j C_j
\]

Referring to Figure 36, the expanded equation is:

\[
\text{Maximize } D = (6.64 P_1 C_1 + 2.49 P_2 C_2 + \cdots + 35 P_{17} C_{17})
\]

Constraints

As shown above, the constraints in a mathematical programming problem are depicted by the general equation:

\[
\sum_{j=1}^{n} a_{ij} x_j = b_i \text{ (for } i=1, 2, \ldots, m) \quad x_j \geq 0 \text{ (for } j=1, 2, \ldots, n)\]
### Threat Matrix with Threats Numbered

<table>
<thead>
<tr>
<th>SCHEMES</th>
<th>Trans-actions Added</th>
<th>Trans-actions Altered</th>
<th>Trans-actions Deleted</th>
<th>File Changes</th>
<th>Program Changes</th>
<th>Improper Operation</th>
<th>Comm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Entry/</td>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Terminal Operator</td>
<td>6.64</td>
<td>2.49</td>
<td>2.75</td>
<td>1.85</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Clerk/Teller</td>
<td>T5</td>
<td>T6</td>
<td>T7</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Teller</td>
<td>3.06</td>
<td>1.53</td>
<td>2.04</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Programmer</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>0.72</td>
<td>0</td>
</tr>
<tr>
<td>Officer/Manager</td>
<td>T9</td>
<td>T10</td>
<td>T11</td>
<td>T12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Manager</td>
<td>3.2</td>
<td>1.6</td>
<td>1.12</td>
<td>0.45</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Computer Operator</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0</td>
<td>0.27</td>
<td>0</td>
</tr>
<tr>
<td>Other Staff</td>
<td>T15</td>
<td>T16</td>
<td>T17</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Outsider</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(Non-employee)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 33. Threat Matrix with Threats Numbered

Where

\[ x_j = \text{Variables that are subject to the control of the decision maker.} \]

\[ a_{ij} = \text{Production (or technical) coefficients that are known.} \]

\[ b_i = \text{Available productive resources in limited supply.} \]

For the resource optimization model the constraint equation may be stated as follows:

\[ \sum_{j=1}^{p} a_{ij} P_j C_j b_i \]
\[ P_j \quad 0 \quad (\text{For j=1,2,---,n}) \]
\[ C_j \quad 0 \quad (\text{For j=1,2,---,n}) \]

Where

\[ P_j \quad \text{and} \quad C_j = \text{Variables that are subject to the control of the decision maker, specifically, the probability of detecting at least one occurrence of fraud (P) and the converse of the rate of occurrence (C).} \]

(\text{It should be noted that the decision maker is not selecting the converse of the rate of occurrence of fraud but rather the converse of the rate of occurrence for which he or she would require the selected "P" value. For example, if the decision maker requires a 99 percent probability of detecting at least one occurrence of fraud if the rate of occurrence of fraud is .002, the "P" value selected is .99 and the "C" value selected is 1-.002 or .998.})

\[ b_i = \text{Available productive resources in limited supply, specifically, the amount of available investigative resources (hours).} \]

\[ a_{ij} = \text{Production (or technical) coefficients for the resource optimization model equates to the hours which are required to accomplish the "P" and "C" levels for a given threat. For example, referring to T1 in the Threat Matrix in Figure 36 (Transaction Added- Data Entry/Terminal Operator) assume that it has been determined that an average of 15 minutes are required to validate each transaction. In this example } a_{ij} \text{ equals .25 (hours).} \]

\[ n = \text{Total number of systems threats that have been identified for a system with a T value in excess of zero. (For the example in Figure 33 } n = 17). \]

Decision Variables

The decision variables in a mathematical programming problem are those variables whose values are to be chosen. In the resource optimization model these variables are "P" and "C" for each threat in the system.

The "P" value or probability of detecting at least one case of computer fraud as a decision variable is rather straightforward. Given that there is a positive relationship between "P" and the level of
investigative resources required, a decision must be made as to what value of "P" is acceptable. Thus, a "P" value of 99 percent for a given rate of occurrence of fraud will require more investigative resources than a "P" value of 90 percent given the same rate of occurrence.

The "C" value, or converse of the rate of occurrence, is not quite as straightforward since it is selected indirectly through the selection of the rate of occurrence which must not be exceeded for a given "P" value. As with the "P" value, there is a positive relationship between the "C" value and investigative resources required. For example, expanding on the example in the previous paragraph, more resources will be required to ensure a given "P" value for a "C" value of .995 than for a "C" value .990.

The selection of specific values for the decision variables "P" and "C" establishes the level of investigative resources that will be required. As will be demonstrated in the following sections, a comprehensive evaluation of all possible "P" and "C" combinations is generally not feasible due to the phenomenally large number of possible combinations. However, the Resource Optimization Model will provide a near optimum solution while only requiring evaluation of a fraction of the possible combinations. Given a certain level of available investigative resources, the model will assign "P" and "C" values to each of the systems threats in a manner which results in a near maximum total detection value for the overall system.

Referring to Figure 33, the "P" and "C" values for each of the 17 threats may be varied using the Iterative Discovery Sampling
technique from Chapter 6 and resulting Detection Quotients computed until a near maximum solution is reached.

Applicability Of The Combinatorial Dilemma

Prior to describing the Resource Optimization Model, the applicability of the Combinatorial Dilemma to the problem of allocating resources to computer fraud detection will be demonstrated.

It should be apparent that a very large number of combinations of "P" and "C" values exists for the 17 threats in Figure 33. The number of combinations may be computed as follows:

Let N = Number of possible combinations

\[ X = \text{Number of threats} \]
\[ Y = \text{Number of possible "P" values} \]
\[ Z = \text{Number of possible "C" values} \]

Then \[ N = (YZ)^X \]

To illustrate, assume there are only three threats \( (T_1, T_2, T_3) \) in a system and for each threat only two possible "P" values \( (P_1 \text{ and } P_2) \) and two possible "C" values \( (C_1 \text{ and } C_2) \). The number of possible combinations for this example is calculated below and shown graphically in Figure 37.

\[ N = (YZ)^X = (2 \cdot 2)^3 = 4^3 = 64 \]

Where \( X = 3 \) (Number of threats)

\[ Y = 2 \] (Number of possible "P" values)

\[ Z = 2 \] (Number of possible "C" values)

The formula \( N = (XZ)^X \) is an expansion of the following rule of permutations:
"If one thing can be done in exactly r different ways, for each of these a second thing can be done in exactly s different ways, for each of the first two, a third can be done in exactly t ways, etc., then the sequence of things can be done is the product of the numbers of ways in which the individual things can be done, i.e., r·s·t... ways." (Kemeny, Snell and Thompson, 1966).

For the first threat in the above example, there are two possible "P" values and two possible "C" values, or $2 \times 2 = 4$ possible "P" and "C" combinations. For each of these combinations for threat one, there are $2 \times 2 = 4$ possible "P" and "C" combinations for the second threat, resulting in $4^2 = 16$ "P" and "C" combinations for the first two threats. As shown in Figure 34, each additional threat acts in a similar manner, adding one to the power "X" in the formula $(YZ)^X$.

Now consider the possible combinations for the threat matrix in Figure 33. For purposes of illustration, "P" values will be limited to three decimals. Thus, "P" values may be set at any of the possible values between 0.000 and 1.0 containing three decimals, or .001, .002, .003, ..., .997, .998, .999. This results in 1,000 possible "P" values which might be considered for each threat. The probability of detecting at least one case of fraud for various occurrence rates within each threat identified in Figure 33 may, therefore, range from .1 percent to 99.9 percent.

"C" will be allowed to take on values to four decimals between 0 and 1, or every value between 0.0000 and 1.0 containing four decimals (.0001, .0002, .0003, ..., .9997, .9998, .9999). It is necessary to provide four decimals for "C" values in order to consider very small rates of occurrence since for certain business this may be required. Provision for four decimals for "C" values results in ten thousand
Figure 34. Combinatorial Example
possible "C" values.

The number of combinations (N) of "P" and "C" values for the threats in Figure 33 may now be calculated as follows:

\[ N = (YZ)^X = (1,000 \cdot 10,000)^{17} = 10,000,000^{17} \]

There is little to be gained from carrying out the mathematics of 10 million to the seventeenth power. It goes without saying that the Combinatorial Dilemma applies. It should be noted that even if only two threats had been identified there would still be ten million squared, or one hundred trillion combinations.

The Resource Optimization Model

Summary of the Mathematical Statement of the Problem

A recap of the mathematical statement of the problem to be solved by the Resource Optimization Model follows:

Maximize \[ D = \sum_{j=1}^{n} T_j P_j C_j \]

Where

- \( D \) = Total Detection Quotient for a system.
- \( T_j \) = Threat Value for threat \( j \).
- \( P_j \) = Probability of finding at least one occurrence of computer fraud.
- \( C_j \) = Converse of the rate of occurrence of computer fraud.
- \( n \) = Total number of threats identified for a system.

Subject to

\[ \sum_{j=1}^{n} a_j P_j C_j = b \]

\( P_j \geq 0 \) (For \( j = 1, 2, \ldots, n \))

\( C_j \geq 0 \) (For \( j = 1, 2, \ldots, n \))

Where
\( P_j \) and \( C_j \) = Variables that are subject to the control of the decision maker. Specifically, the probability of detecting at least one occurrence of fraud \( (P) \) and the converse of the rate of occurrence \( (C) \).

\( b = \) Available productive resources in limited supply, specifically the amount of available investigative resources (hours).

\( a_j = \) Production (or technical) coefficients. For the Resource Optimization Model this equates to the hours required to accomplish the "P" and "C" levels for a given threat.

\( n = \) Total number of systems threats which have been identified.

This mathematical statement of the problem assumes that it is possible to determine the hours required to accomplish the selected "P" and "C" values.

An English Statement of the Resource Optimization Problem

The objective of the Resource Optimization Model is to allocate resources to the various identified threats in such a way that the detection capability for the system is maximized as measured by the Detection Quotient for the system. The underlying assumption is that investigative resources are limited and systems threats must compete for them.

The resources allocated to the various threats is driven by the "P" and "C" values for each threat. As demonstrated in the section on the Applicability of the Combinatorial Dilemma, the number of possible combinations of "P" and "C" values, typically precludes a total evaluation, certainly for the hypothetical system represented in Figure 33.
The Resource Allocation Solution

The solution to the resource allocation problem is based on the Iterative Discovery Sampling process described in Chapter 6. By applying this technique it is possible to ensure with a high level of confidence, a near optimum solution while only examining a fraction of the possible combinations, or alternative solutions. Recall that the basic procedure of this technique is to select a base "combination to beat", then iteratively take random samples of additional combinations attempting to beat the base combination. When a superior combination is found it becomes the new "combination to beat" and the process is repeated. This process continues until an entire sample is compared to the residing "combination to beat" without discovering a superior combination. Based on the principles of discovery sampling, it can then be stated with a given statistical confidence, that the reigning combination is within some specific top percentage of possible combinations. For example, if there are 200,000 possible combinations, a sample of 3,000 which does not produce a combination superior to a reigning "combination to beat" may be interpreted to mean that the reigning combination is within the top .2 percent of possible combinations with a confidence level of 99.8 percent.

Two issues should be addressed at this point. First, is the size of samples which will be required for phenomenally large numbers of combinations. In the above example a sample size of 3,000 will provide the given statistics for a population of 200,000. But, how large a sample will be required for a much larger population, for example, 250 trillion possible combinations, or the 10 million
possible combinations for the threat matrix in Figure 33. The second issue concerns the number of iterations which may be required before a full sample is evaluated without discovering a combination superior to the base combination, or "combination to beat".

Sample Sizes

The larger the population, or field size is, the more effective the sampling technique becomes. For larger populations the sample size remains almost constant for given rates of occurrence. This may be noted in Figure 35 where comparative sample sizes are plotted for a 99 percent probability of finding at least one occurrence of an event when the rate of occurrence is .2 percent. Notice that the sample size only rises slightly for a population above ten thousand. In order to determine exact sample sizes for various field sizes, probabilities and rates of occurrence, the equation from Chapter 4 may be used, although for very large populations the equation is cumbersome. (Recall that the equation is \( Pr = 1 - \binom{N-d}{d} \binom{N}{n} \) where \( N = \) Field Size; \( n = \) Sample Size; \( d = \) Number of events within the field: \( \binom{N}{n} = \frac{N!}{n!(N-n)!} \) and \( \binom{b}{a} = \frac{b!}{a!(b-a)!} \) generally.

Use of the equation from Chapter 4 would, of course, require automation on a powerful, high speed computer since the manipulation of factorials for huge numbers is not feasible manually or even on smaller computers. A shortcut which may be used is to refer to Arkin's tables (Arkin 1967) to determine at what population size the sample size levels off, then use that sample size or a somewhat inflated sample size. Another alternative is to calculate sample sizes using the
The disadvantage of the shortcut methods is that they are not
scientifically defensible, thus they may be hard to justify. However, from an intuitive standpoint it should be relatively easy to accept them. With random selection of samples a sample of 3,000 should provide approximately the same probability of discovering one occurrence whether the population is 30,000 or 30 billion, given the same rate of occurrence. For example, with a rate of occurrence of .01 percent there would be three such occurrences in a population of 30 thousand and three million occurrences in a population of ten billion. It should be fairly easy to accept, on an intuitive level, that a sample of 3,000 from either group should produce approximately the same probability of finding at least one occurrence, assuming the samples are strictly random.

Iterations

The second issue which should be addressed is the number of iterations of samples which might have to be evaluated before a solution is found. In Chapter 6 an Iterative Discovery Sampling technique was described which provides a solution to the "Combinatorial Dilemma". This technique converges to a near optimum by iteratively taking samples of possible alternatives. The proximity of the solution to the optimum can be stated with statistical precision. Thus, it may be stated with a given level of confidence that the solution is within some specific top percentage of possible solutions.

The first step in the process described in Chapter Six is to take a full sample and select the best combination. This combination then becomes the "combination to beat". For example, given a
population of 200,000 and a desire to ensure, with a 99.8 percent confidence, a solution within the top .2 percent, a sample of 3,000 would be required. To establish the initial "combination to beat" a full sample of 3,000 would be taken and the best combination from this sample selected.

Following the establishment of this "combination to beat" another sample of 3,000 would be taken. If a combination is discovered in this new sample which is superior to the established "combination to beat" this combination becomes the new "combination to beat" and a new sample is drawn. Thus, for example, if the 352nd combination in the second sample of 3,000 is superior it becomes the new "combination to beat", the remainder of the second sample is discarded and the third sample of 3,000 is drawn. This process is continued until a full sample of 3,000 produces no combination which is superior to the existing "combination to beat". This reigning combination may then be interpreted as being within the top .2 percent of possible solutions with a 99.8 percent confidence.

It is suggested that only a few iterations will be required to converge to a solution using the technique described above. It may even be argued that the first full sample provides the solution since out of 3,000 combinations none are superior to the base "combination to beat" simply because the best combination from the initial full sample is chosen.

The contra to the above argument is that it violates the premise of discovery sampling. This premise is that a complete, independent, random sample be analyzed in search of an event; in the
above example, a superior combination. Although it may be interesting to debate this issue mathematically, this thesis will adopt the more conservative approach of not violating the premise of discovery sampling. Intuitively, however, it should be apparent that the first full sample will produce a combination approaching the desired solution.

In a slightly modified version of the "Iterative Discovery Sampling" technique, a Pricing Model (Dunn 1971) was developed which produced a near optimum combination of prices for a line of electronic products. This pricing model was run numerous times and always converged to a solution in three or less iterations.

Application

Application of the Resource Optimization Model begins with the threat matrix for a given system. For purposes of illustration, the threat matrix in Figure 33 will be used. These threats have been rearranged in Figure 36 in order to demonstrate the methodology.

The example in Figure 36 represents one of the possible ten million\textsuperscript{17} possible combinations of "P" and "C" for the 17 threats shown in Figure 33. The first column gives the scheme/perpetrator type associated with each of the threats. The second column shows the total number of activities for a given period. For instance, the number 50,000 in the first row indicates that 50,000 transactions are added by data entry/terminal operators during a given period. This number may also be referred to as the "population". The fourth row from the bottom under the "population" column indicates that 5,000 jobs are set-
<table>
<thead>
<tr>
<th>&quot;T&quot;</th>
<th>No. of Activities (Pop.)</th>
<th>Sample Size</th>
<th>Investigative Time Required per Sample (hours)</th>
<th>Total Sample Time Required (hours)</th>
<th>&quot;T&quot; Value</th>
<th>&quot;P&quot; Value</th>
<th>&quot;C&quot; Value</th>
<th>DQ Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>50,000</td>
<td>1,000</td>
<td>.25</td>
<td>250</td>
<td>.64</td>
<td>.983</td>
<td>.996</td>
<td>6.50</td>
</tr>
<tr>
<td>T2</td>
<td>15,000</td>
<td>700</td>
<td>.3</td>
<td>210</td>
<td>.972</td>
<td>.972</td>
<td></td>
<td>2.41</td>
</tr>
<tr>
<td>T3</td>
<td>10,000</td>
<td>500</td>
<td>.3</td>
<td>150</td>
<td>.924</td>
<td>.995</td>
<td></td>
<td>2.53</td>
</tr>
<tr>
<td>T4</td>
<td>2,000</td>
<td>200</td>
<td>.3</td>
<td>60</td>
<td>.880</td>
<td>.990</td>
<td></td>
<td>1.61</td>
</tr>
<tr>
<td>T5</td>
<td>10,000</td>
<td>1,200</td>
<td>.25</td>
<td>300</td>
<td>.994</td>
<td>.996</td>
<td></td>
<td>3.03</td>
</tr>
<tr>
<td>T6</td>
<td>4,000</td>
<td>200</td>
<td>.3</td>
<td>60</td>
<td>.995</td>
<td>.985</td>
<td></td>
<td>1.44</td>
</tr>
<tr>
<td>T7</td>
<td>2,000</td>
<td>300</td>
<td>.3</td>
<td>90</td>
<td>.962</td>
<td>.990</td>
<td></td>
<td>1.94</td>
</tr>
<tr>
<td>T8</td>
<td>1,000</td>
<td>200</td>
<td>.7</td>
<td>140</td>
<td>.894</td>
<td>.990</td>
<td></td>
<td>.64</td>
</tr>
<tr>
<td>T9</td>
<td>500</td>
<td>200</td>
<td>.25</td>
<td>50</td>
<td>.871</td>
<td>.992</td>
<td></td>
<td>2.76</td>
</tr>
<tr>
<td>T10</td>
<td>200</td>
<td>100</td>
<td>.3</td>
<td>30</td>
<td>.50</td>
<td>.995</td>
<td></td>
<td>.80</td>
</tr>
<tr>
<td>T11</td>
<td>400</td>
<td>200</td>
<td>.3</td>
<td>60</td>
<td>.938</td>
<td>.990</td>
<td></td>
<td>1.04</td>
</tr>
<tr>
<td>T12</td>
<td>600</td>
<td>100</td>
<td>.5</td>
<td>50</td>
<td>.422</td>
<td>.995</td>
<td></td>
<td>.19</td>
</tr>
<tr>
<td>T13</td>
<td>200</td>
<td>100</td>
<td>.4</td>
<td>40</td>
<td>.986</td>
<td>.970</td>
<td></td>
<td>.38</td>
</tr>
<tr>
<td>T14</td>
<td>5,000</td>
<td>200</td>
<td>.6</td>
<td>120</td>
<td>.871</td>
<td>.990</td>
<td></td>
<td>.23</td>
</tr>
<tr>
<td>T15</td>
<td>1,000</td>
<td>100</td>
<td>.25</td>
<td>25</td>
<td>.653</td>
<td>.990</td>
<td></td>
<td>.65</td>
</tr>
<tr>
<td>T16</td>
<td>200</td>
<td>100</td>
<td>.3</td>
<td>30</td>
<td>.751</td>
<td>.990</td>
<td></td>
<td>.37</td>
</tr>
<tr>
<td>T17</td>
<td>500</td>
<td>100</td>
<td>.3</td>
<td>30</td>
<td>.967</td>
<td>.970</td>
<td></td>
<td>.33</td>
</tr>
<tr>
<td>TOTALS</td>
<td></td>
<td></td>
<td></td>
<td>1,695</td>
<td></td>
<td></td>
<td></td>
<td>26.85</td>
</tr>
</tbody>
</table>

Figure 36. Sample Iterations of Resource Optimization Model

up and run by computer operators for a given period.

The third column is one random selection, or combination of sample sizes for the various schemes. For the first item, T1, the
sample size is 1,000 for transactions added by data entry/terminal operators. The fourth column represents the estimated time in hours to examine one transaction, or activity and the fifth column gives the total estimated hours required to examine the sample given in the fourth column.

The last four columns determine the detection quotient for each system threat (DQ = TPC) and for the system as a whole (System DQ = Sum of individual DQ's).

The system DQ value measures the effectiveness of each combination of "P" and "C" values. For the combination in Figure 36 the system DQ value is 26.85. Using the Resource Optimization Model, the DQ value would be compared for various randomly selected "P" and "C" combinations until a combination withstands comparison to a complete sample of combinations without being beat.

As indicated in the mathematical statement of the problem, the constraint in the Resource Optimization Model is the total number of hours available to spend in investigation. Notice that the hours required to perform the investigation described in Figure 36 is 1695. For purposes of illustration, this will be assumed to be the number of hours available. In actuality, the "P" and "C" values for a given combination will probably have to be adjusted up or down on a random basis to conform to the available hours. For example, given 1695 available hours, assume that the "P" and "C" combination at Figure 37 required 1950 hours. It would be necessary to adjust "P" and "C" values downward on a strictly random basis until the total adjusted hours required falls within (is less than or equal to) 1695.
<table>
<thead>
<tr>
<th>&quot;T&quot;</th>
<th>No. of Activities (Pop.) Size</th>
<th>Investigative Time Required per Sample (hours)</th>
<th>Investigative Total Sample (hours)</th>
<th>&quot;T&quot; Value</th>
<th>&quot;P&quot; Value</th>
<th>&quot;C&quot; Value</th>
<th>DQ Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>50,000 2,000</td>
<td>.25</td>
<td>500</td>
<td>6.64</td>
<td>.983</td>
<td>.998</td>
<td>6.51</td>
</tr>
<tr>
<td>T2</td>
<td>15,000 500</td>
<td>.3</td>
<td>150</td>
<td>2.49</td>
<td>.869</td>
<td>.996</td>
<td>2.16</td>
</tr>
<tr>
<td>T3</td>
<td>10,000 1,000</td>
<td>.3</td>
<td>300</td>
<td>2.75</td>
<td>.995</td>
<td>.995</td>
<td>2.72</td>
</tr>
<tr>
<td>T4</td>
<td>2,000 100</td>
<td>.3</td>
<td>30</td>
<td>1.85</td>
<td>.643</td>
<td>.990</td>
<td>1.18</td>
</tr>
<tr>
<td>T5</td>
<td>10,000 1,200</td>
<td>.25</td>
<td>300</td>
<td>3.06</td>
<td>.994</td>
<td>.996</td>
<td>3.03</td>
</tr>
<tr>
<td>T6</td>
<td>4,000 200</td>
<td>.3</td>
<td>60</td>
<td>1.53</td>
<td>.995</td>
<td>.985</td>
<td>1.44</td>
</tr>
<tr>
<td>T7</td>
<td>2,000 700</td>
<td>.3</td>
<td>210</td>
<td>2.04</td>
<td>.968</td>
<td>.996</td>
<td>1.97</td>
</tr>
<tr>
<td>T8</td>
<td>1,000 0</td>
<td>.7</td>
<td>0</td>
<td>.72</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T9</td>
<td>500 100</td>
<td>.25</td>
<td>25</td>
<td>3.20</td>
<td>.967</td>
<td>.970</td>
<td>.33</td>
</tr>
<tr>
<td>T10</td>
<td>200 100</td>
<td>.3</td>
<td>30</td>
<td>1.60</td>
<td>.751</td>
<td>.990</td>
<td>1.19</td>
</tr>
<tr>
<td>T11</td>
<td>400 0</td>
<td>.3</td>
<td>0</td>
<td>1.12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T12</td>
<td>600 0</td>
<td>.5</td>
<td>0</td>
<td>1.45</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T13</td>
<td>200 0</td>
<td>.4</td>
<td>0</td>
<td>.4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T14</td>
<td>5,000 100</td>
<td>.6</td>
<td>60</td>
<td>.27</td>
<td>.952</td>
<td>.970</td>
<td>.25</td>
</tr>
<tr>
<td>T15</td>
<td>1,000 0</td>
<td>.25</td>
<td>25</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T16</td>
<td>200 0</td>
<td>.3</td>
<td>0</td>
<td>.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T17</td>
<td>500 100</td>
<td>.3</td>
<td>30</td>
<td>.35</td>
<td>.967</td>
<td>.970</td>
<td>.33</td>
</tr>
</tbody>
</table>

**TOTALS**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>1,695</strong></td>
<td><strong>21.11</strong></td>
</tr>
</tbody>
</table>

Figure 37. Alternate Sample Iteration of Resource Optimization Model

For comparative purposes, consider the combination of "P" and "C" values in Figure 37 which also conforms to the overall limitation of 1695 available hours. Notice that the total DQ value for this
combination of "P" and "C" values is 21.11 compared to 26.85 for the combination in Figure 36.

Following the methodology described above, assume the combinations in Figures 36 and 37 are the first two combinations in the first full sample. Using the example above with a sample size of 3,000 the combination in Figure 36 would be retained as the "combination to beat" and 2,998 more combination comparisons made. The best of these combinations would be the base "combination to beat" and the next sample drawn. The process would be continued until a combination withstands comparison to a full sample of 3,000 without being beaten. At that point the reigning combination could be considered in the top .2 percent of possible combinations (in this case, allocations of available resources) with a confidence level of 99.8 percent.

The prospect of repeating the process contained in Figures 36 and 37 therein several thousand times should illustrate the benefits of automating the Resource Optimization Model. Although the process could be performed manually, it is very labor intensive and lends itself extremely well to automation.
CHAPTER 8

SUMMARY AND CONCLUSIONS

The research for this dissertation indicates that much is yet to be done in detecting and reporting computer fraud. As illustrated in Chapter One, reported cases of computer fraud probably represent a very small fraction of actual cases. While it may be logical to assume that reported cases of computer fraud are representative of the population of actual cases, this assumption cannot be defended on any scientific basis.

The typology in Chapter Two is provided as "best available" evidence of vulnerabilities of various types of computer systems. A measure of comparative vulnerabilities for various computer system types, developed for this dissertation, is provided which combines relative frequency of occurrence and dollar impact values for different computer fraud types using the formula vulnerability \( V = F \times I \).

The primary distinction between the treatment of reported cases in this dissertation and other sources reviewed during the research was this formal combining of frequency and impact into one indicator. Other sources deal with them separately, or deal with only one, to the exclusion of the other.

A methodology for threat assessment is developed in Chapter Three. This methodology is based on a matrix approach which combines
reported computer fraud types with perpetrator types, to produce a "threat value" for each fraud type/perpetrator type combination. While others have used a matrix approach to illustrate computer fraud and perpetrator types, the process of deriving a "threat value" for each cell in the matrix, representing a specific computer fraud type/perpetrator type, was developed for this dissertation and distinguishes it from other literature sources.

The typology in Chapter Two and the threat assessment methodology in Chapter Three provide valuable insight into the comparative vulnerabilities and threat values for different computer fraud type/perpetrator type combinations based on reported incidents of fraud. However, the research to this point indicates such a limited association between the number of reported cases and actual cases that the results cannot be extended to specific systems on any scientific basis. Although it would be nice to simply concentrate on highly vulnerable systems and perpetrator types as reported in Chapters Two and Three in assigning computer fraud detection resources to specific systems, the research does not support this approach.

Based on this conclusion, a methodology was developed for evaluating threats and allocating resources to computer fraud detection for specific systems. This methodology consists of four components in addition to the threat assessment matrix approach developed in Chapter Three.

First, is the detection quotient in Chapter Four, a measure of the effectiveness of different allocations of available resources to computer fraud/perpetrator combinations for given systems. The
detection quotient, developed specifically for this methodology, is based on the threat value discussed earlier in conjunction with attributes of the discovery sampling approach.

Second, is a threat assessment methodology for specific systems, presented in Chapter Five. This methodology is based on the general threat assessment methodology from Chapter Three in conjunction with the Delphi Approach, the Churchman-Ackoff technique and a Controls Analysis process developed for the methodology. This combination of techniques forms a unique approach to threat analysis aimed at maximizing the effectiveness of subjective threat assessment.

The third component was necessitated by the phenomenally large number of possible allocations of resources associated with large systems. This situation, entitled "The Combinatorial Dilemma" in Chapter Six, occurs when the large number of possible alternatives precludes comprehensive analysis. A general solution to the combinatorial dilemma is provided in Chapter Six which provides a statistically defensible near optimum alternative while only examining a small fraction of possible alternatives. Although the combinatorial problem is well known, the solution in Chapter Six is unique to this dissertation.

The fourth major component of the methodology for allocating resources to computer fraud detection in specific systems is the resource optimization model in Chapter Seven. This model converges, through an iterative process, to a near optimum allocation of computer fraud detection resources utilizing the concepts developed in Chapters Four through Six.
Finally, in Appendices A and B, investigative techniques and automated techniques associated with computer fraud detection are presented.

The research did not reveal any "Cookbook" approaches for detecting computer fraud. On the contrary, the lack of comprehensive historical data in conjunction with the unpredictable nature of intentional acts such as computer fraud discourages such an approach. Rather, the methodology, tailored to the specific characteristics of given computer systems, is suggested.

This latter approach presents two major challenges. First, is the need to identify comprehensively and effectively, the specific threats surrounding a given system and evaluate their relative significance. Second, available detection resources must be assigned to these specific threats. This creates a resource allocation problem since only a fraction of the resources required to completely examine every threat will typically be available.

The methodology presented in this dissertation answers these two challenges in a way which should produce a near optimum allocation of resources to computer fraud detection for the threats which are relevant to a given system.

Stronger legislation on reporting computer fraud and continued research will hopefully increase available data on computer fraud and provide more insight into threats surrounding various types of computer systems. However, it is suggested that "Cookbook" approaches which might be effective in detecting unintentional acts such as errors and omissions in data, will probably continue to have only limited success.
against intentional acts such as fraud.

In conclusion, computer fraud threat assessment and resource allocation must be tailored to specific systems. The methodology in this dissertation provides such an approach. Hopefully, this dissertation will stimulate further research in the areas indicated in the preface, specifically: expansion of case data; the deterring effects of perceived detection capabilities on would be perpetrators and the "live monitoring" technique discussed in Appendix A.
APPENDIX A

THE INVESTIGATION

There is a considerable amount of literature available on the subjects of Computer Auditing, EDP Auditing, Automated Auditing, etc. Additionally, an audit or investigative ability probably either exists internally or is available externally to most organizations large enough to be seriously concerned with computer fraud.

For these reasons this thesis does not address audit or investigative techniques from a comprehensive viewpoint. The primary thrust of the thesis is to provide methodologies for identifying systems threats and, then, to direct resources to the detection and abatement of these threats in an effective manner. Once this has been accomplished, the typical organization should be capable of auditing or investigating the threats in a system. For example, most audit or internal review groups should be capable of examining "Transactions Added by Data Entry/Terminal Operators" to determine their validity once they are identified as a threat and appropriate resource allocation is established.

There are several issues, however, which should be addressed to bring the methodologies in this thesis in line with the technologies and methodologies for auditing or investigating computer systems.

135
**Investigation VS Audit**

The first issue that will be addressed revolves around the selection of the term "The Investigation" as a title for this chapter rather than "The Audit". The reason for choosing "The Investigation" is its connotation of examining either past or current activities. "The Audit", on the other hand, connotes an examination of past activities. This does not mean that audits never address current activities, but rather that the word "Audit" typically connotes review of past activities. These connotations are supported by the dictionary (The American Heritage Dictionary of the English Language 1976) where the following definitions appear:

Audit - 1. An examination of records or accounts to check their accuracy. 2. An adjustment or correction of accounts. 3. An examined and verified account.

Investigate - To observe or inquire into detail; examine systematically. Search into: to trace or track.

Notice that the general connotation of "Audit" from the above definition is past tense while the definition of "Investigate" suggests examination of either current activity or past activity. This latter type of examination is necessary for evaluating the broad spectrum of computer fraud threats. A simple review of the threats in Figure 37 should confirm this.

For example, assuming that comprehensive transaction registers are maintained for all transactions, an audit or internal review group should be able to evaluate the first threat "Transactions Added by Data Entry/Terminal Operators". Further, this should be achievable after the fact by examining historical records, specifically, the transaction
registers. However, this technique may not be adequate for program changes, file changes or improper computer operations where there may be no record of extraneous activities. It might also be inadequate for the above "transactions added" example if it is possible to add transactions without entry on the transaction register. It certainly would be inadequate for transactions if no register existed.

Where "after the fact" examination is not adequate it is necessary to incorporate current activities as well as historical into the examination of computer system threats, thus, the term "Investigation" is appropriate.

Assumptions

Several assumptions are necessary in implementing the Resource Optimization Model in Chapter 7 and the investigative techniques which will be covered in this chapter. First, is the assumption of limited resources. This assumption is that there are inadequate resources available to examine every activity identified as a threat. The second assumption is that the investigative resources are capable of a partial examination. Thus, while resources are not adequate to examine every transaction processed through a system, for example, those which are will be accomplished in a competent manner. Finally, it is assumed that reasonable time estimates may be made of the amount of time required to examine each threat activity. The first of these assumptions should be relatively easy to accept. In today's environment it is not unusual for large systems to process hundreds of thousands of transactions per month requiring thousands of file
updates, program changes and computer operation cycles. In many cases there is simply not enough time between processes to examine every activity. Further, even if it were possible to apply enough resources to perform a total evaluation, the costs would probably be prohibitive. The infeasibility of total evaluations is pretty well established, certainly for large systems.

The second assumption is not too difficult to accept for activities easily evaluated through the use of traditional auditing techniques such as transactions added where good transaction registers exist. The assumption is more difficult to accept in those instances where traditional auditing techniques are not as easily applied such as improper computer operations. This latter category of activities will be the focus of much of the remainder of this chapter. Hopefully, the methodologies presented will diminish doubts regarding this assumption.

The third assumption, like the second assumption is quite believable for activities which are amenable to traditional audit techniques. Using the above example, it should be fairly easy to estimate the time required to examine each transaction in a system where good transaction registers exist. If evaluations have been made in the past, time estimates can probably be made from past experience. If not, it should be possible to estimate reasonably by simply evaluating several transactions and recording time requirements. It may be necessary to adjust the initial estimates after a few iterations to account for increased proficiency, unusual transactions, etc., however, it should not be too difficult to establish reliable estimates. For activities which are not amenable to traditional audit
techniques the time estimates will be more difficult but should still be achievable. It is suggested that, once the methodologies for dealing with this type of activities are established, time estimates to accomplish them become feasible. Therefore, as with the second assumption, the methodologies discussed in the remainder of this chapter should alleviate many of the doubts regarding the third assumption.

If, however, in the final analysis, it is determined that the capability simply does not exist to investigate a particular threat and it is not feasible to develop this capability, or if it is impossible to estimate the time required to investigate an activity, adjustments may have to be made. If, for example, the organization has no capability to examine computer operations by computer operators, zero resources could be applied to this threat. Although these resources could be applied to other threats, there is the potential that the resulting systems detection capability, as measured by the system DQ will be lower than if the capability had existed.

The above case of zero capability is not likely. Although it may be small, almost any investigative effort should produce some possibility of detection. Even a small detection capability might have a significant abatement potential. This leads to the final observation which revolves around the deterring effect of computer fraud detection.

**Treating The Deterring Effect Of Computer Fraud Detection**

Recall from the Computer Fraud Detection Model in Figure 8 that if deterrents to computer fraud are strong enough prevention and
detection are unnecessary. Thus, in an ideal situation, high morals or fear of imprisonment, for example, might totally eliminate computer fraud threat. Such an ideal situation is, unfortunately, the exception rather than the rule. As the research showed, computer fraud is all too often not detected and even when detected it is often not prosecuted. Assuming that prevention and detection are required there is, however, a relationship between these measures and deterrence. To clarify, assume that there is zero capability to detect fraudulent computer operations by computer operators. Actually, it is only necessary that the computer operators perceive a zero capability. Basically, given this total lack or perceived total lack of detection capability, it is fair to assume that those with a propensity and ability to commit fraud will do so since would be perpetrators will view it as a "nothing-to-lose, everything-to-gain" situation.

Now assume that, instead of a zero capability, the computer operators perceive a ten percent capability on the part of management to detect fraud. Does it follow that ten percent of would be perpetrators will be discouraged by this perceived ten percent detection capability? How about a perceived twenty percent; forty percent; eighty percent or ninety-nine percent perceived capability? If so, the relationship between detection capability perceived and deterrence would be as shown in Figure 38.

Observe in Figure 38 that there is a one-to-one positive relationship between the perceived detection capability and the percentage of would be perpetrators deterred from committing fraud. Does this relationship seem logical? Although a review of the
literature did not reveal the answer to this question, it does not seem logical from an intuitive viewpoint. It seems more likely that a relatively small perceived detection capability might produce a significantly larger deterring effect. As an analogy, for example, consider the expenditure of funds for fire insurance. Given a perceived probability of zero that a fire will occur in a person's home, it is highly unlikely that any money would be spent on fire insurance. However, given a very small perceived probability of fire, just slightly greater than zero, a very large percentage of people methodically purchase fire insurance.

Although unproven, it seems likely from an intuitive viewpoint that the relationship between the perceived detection capability and
deterrence might more closely resemble the curve in Figure 39 than the one in Figure 38.

The curve in Figure 39 indicates that a large percentage of would-be perpetrators will be discouraged from perpetrating fraud by a small perceived detection capability. The curve then continues upward but at a decreasing rate until it reaches 100 percent on each axis indicating that 100 percent of would-be perpetrators will be discouraged by a 100 percent perceived probability of getting caught as in Figure 38. However, there is a significant difference in the relationship between the perceived detection capability and deterrence in the two curves. The relationship is, of course, quite complex, and it is not intended that either Figure 38 nor Figure 39 be interpreted
as an accurate illustration, but rather to point out the diverse possibilities. The exact nature of this relationship would require a significant research effort to solve. In fact it would probably make a good research topic for a dissertation in the behavioral sciences.

As indicated above and illustrated in the Computer Fraud Model in Chapter 3, a perceived detection capability becomes a deterrent. The above unresolved questions indicate that the exact nature of this deterring effect is unknown. It is, therefore, impossible to totally quantify its effects. However, it is included as a topic in this chapter for two reasons: First, to suggest a lucrative area for research and second, to discourage the superficial elimination of threats from resource allocation simply because detection capabilities are low.

There might be a strong temptation to avoid allocating resources to threat activities for which a low detection capability exists. This action might well be supported by the curve in Figure 38. However, it would probably not be supported by the curve in Figure 39 where even a small perceived detection capability discourages a large percentage of would be perpetrators. While the Resource Optimization Model in Chapter 7 does not deal directly with the deterring effects of perceived capabilities, it does not eliminate threats from resource allocation simply because of a low detection capability. If incremental assignment of resources to a threat with a low detection capability results in a larger positive gain to the system "DO" value than incremental assignment to a threat with a higher detection capability the resources are assigned to the threat with the lower
detection capability. An analogy may be drawn from the example toward the end of Chapter 7 where the incremental assignment of resources to a threat with a small threat value resulted in a larger net gain to the system "DQ" value than the assignment of these same resources to a threat with a larger threat value.

Although the Resource Optimization Model does not deal directly with the quantitative effects of perceived detection capabilities, there is an indirect treatment. To illustrate, it is necessary to think beyond the first iteration of the Computer Fraud Detection Model in Figure 8 of Chapter 3. Assume, for example, that computer fraud threats were identified for the first time a year ago, for a given organization and that the Resource Optimization Model was run and investigations accomplished during the past year.

Assuming that the activities of the past year were at least marginally effective, would be perpetrators' perceptions of the organization's ability to detect computer fraud should be greater than they were a year ago. Now, assume that the Computer Fraud Detection Program for the next year is in the initial stages of implementation. When the systems threats for the second year are identified and evaluated, the deterring effect of computer fraud detection capabilities emanating from the previous year's program should be considered. Thus, the threat profile might change from one year to the next, thereby influencing the final allocation of resources. Over time, changing detection capabilities will effect this allocation, thus the deterring effects of these capabilities does have an indirect impact on resource allocation.
Relationship Between Controls And The Investigation

It was demonstrated in Chapter 5 that Controls Analysis had a tempering effect on the threat matrix. Threat values of the threats initially identified in an unconstrained fashion were adjusted downward or eliminated after Controls Analysis. Controls, thus, have a direct impact on the investigation since they determine, to some extent, which threats will be considered for examination and to what degree. Existing controls will also partially determine the nature of the investigation since certain controls will influence the need for specific techniques of examination. For example, less emphasis on terminal operator identification verification may be required in a system containing two levels of security protection (e.g., system identification and password) than a system with only one level (e.g., password only). Finally, certain controls are essential to the investigation. Without these controls it becomes difficult, if not impossible, to perform an evaluation. For the most part, these types of controls are also essential to the basic operation of a system. For example, in a system with a high volume of transaction input it would be difficult to operate the system without a transaction register on which transactions added, altered or deleted are recorded.

Investigative Peculiarities

Although there are many similarities between the computer fraud investigation and traditional audits or investigations, there are several factors which deserve special attention.
Evaluating Internal Controls

As indicated above, controls which are in effect influence the Threat Assessment as described in Chapter 5. In addition, these controls have a direct impact on the investigation of identified threats. The key to Controls Analysis, whether it is for Threat Assessment or Investigative Planning, is a basic knowledge of electronic data processing equipment and systems. Cook and Winkle (1980) cite this knowledge as necessary for the examiner to conform to the first general auditing standard which provides that an audit examination should be performed only by "... persons having adequate technical training and proficiency...". The following specific guidance is cited:

If a client uses EDP in its accounting system, whether the application is simple or complex, the auditor needs to understand the entire system sufficiently to enable him to identify and evaluate its essential accounting control features. Situations involving more complex EDP applications ordinarily will require that the auditor apply specialized expertise in EDP in the performance of the necessary audit procedures.

The evaluation of controls, thus, requires a reasonable degree of technical knowledge in computers. If the auditors or examiners do not possess this expertise, advisors or consultants having the expertise should be called on to work with them in evaluating controls. The following are some basic knowledges which the evaluators should possess.

Elements of an ADP System

Basic to an understanding of EDP systems is a familiarity with the elements involved. EDP systems generally include, in some form,
the following components or elements:

Hardware
Software
Documentation
People
Data (Files)
Procedures

Additionally, more and more systems involve local or long haul communications, or both.

**Hardware.** Hardware includes the physical equipment in a system. The physical equipment in a system may range from a small central processing unit (CPU) with a few peripheral input and output devices (e.g., card readers, tape drives, disk drives, card punches, and printers) to banks of huge CPUs with thousands of peripheral units including remote terminals, front-end minicomputers, etc., in addition to the above types of devices.

**Software.** Software includes the computer programs and routines that direct and facilitate the operation of the hardware. Software typically falls into two categories, systems software and applications software. The auditor or examiner should be familiar with both types. Systems programs generally fall into two categories, either operating systems which are used to process one or more applications programs simultaneously or data management systems which handle standardized data functions for applications programs. Systems programs are usually developed by equipment manufacturers or by software companies.
Applications programs are made up of instructions which cause the computer system to perform specific data processing tasks. These programs have typically been written by the user of the system or by a software developer for a specific application. However, with the advent of the minicomputer and microcomputer markets, thousands of applications software packages are being developed and sold "off-the-shelf".

**Documentation.** Documentation describes the activities surrounding a computer system. This documentation covers operations of the system; application and systems software and procedures. Unfortunately, documentation is all too often inadequate, particularly for applications software. In fact, in many cases, the documentation is totally inadequate if it exists at all. In these cases it might be necessary to read the code in the computer programs to determine what will happen when they are processed, or process comprehensive test decks and observe what happens.

Documentation generally falls into the following categories:

**Functional Description.** The functional description describes the application in terms of the functions it performs. This document should address the operations performed. For example, for an accounting system the description should describe incoming documents or data, required reports and information and the processes required to produce them.

**Systems Documentation.** This documentation generally describes the processes and flow of data through the computer system. As opposed to the functional description which concentrates on the functions performed, the systems documentation is oriented to the processes required to perform the functions including descriptions of input, output and file data.

**Program Documentation.** The program documentation shows the specific steps and logic of the computer program through
narrative listings and program flowcharts, data formats including input, output, file and record layouts.

Operations Manuals. These manuals contain instructions and information needed by the computer operator to run the system.

User Documentation. These documents instruct the user in the use of the system. They typically describe inputs, output listings, reports, error codes, correction procedures and processing cycles necessary to successfully operate the system.

People. The people involved in a computer system vary from organization to organization. However, the functions performed are similar even though they may be performed by people with different titles.

Functional Manager. The Functional Manager, from the users group, exercises control over and typically has responsibility for the function being performed. Thus, for example, the Accounting Manager, Vice President of Accounting, Chief Accountant, Director of Accounting, etc., might control the function of accounting and be responsible for accounting information and reports.

EDP Manager. Typically, the EDP Manager is responsible for overall systems planning, development and operation of systems, equipment and processes.

Systems Analyst. The Systems Analyst usually evaluates existing systems, designs new systems and prepares specifications for programmers. The Systems Analyst may be part of the user's group although, typically he or she belongs to the EDP group.

Programmer. The Programmer, generally, prepares detailed flowcharts, decision charts, etc., of system logic; programs; develops; debugs and documents computer programs.

Computer Operator. This individual operates the computer equipment and runs computer programs according to operating instructions.

Data Entry Operator. The Data Entry Operator prepares data for entry into the computer through such devices as keypunch, key-to-tape, key-to-disk machines and on-line data entry devices such as remote terminals.
The Librarian, typically, maintains systems documentation, programs and files.

Data (Files). Data include transactions and related information that are entered into a system, stored, processed or produced by a system. A collection of data with certain common characteristics is commonly referred to as a file. The files are processed through the computer equipment in accordance with the software instructions to produce information, reports, additional files, etc.

There are usually two types of files involved in a process. First, is the master file which contains information for given classes of data. Second, is the transaction file which is processed against the master file to update the master files.

Procedures. The procedures surrounding a system dictate when and how data is processed through the system. In addition to normal processing, these procedures cover emergency or unusual operations such as loss of data files, loss of power to the computer, etc.

Diminished Audit Trail

In their book on "Management Fraud - Detection and Deterrence", Elliott and Willingham (1980) discuss the prospect of diminished source documentation. They observe that;

Computers are designed to reduce paper work and as systems become more sophisticated they eliminate more and more of the backup paper documentation which can facilitate controls and audit testing. Source documentation may increasingly be eliminated by the expanded use of real-time systems for accounting. Unlike batch-operated systems, which process information when it is input from batches of source documents (e.g., sales invoices), real-time systems allow the operator to input data directly into the processing unit by a keyboard and
have it processed without delay. In many situations there need be no source document.

The observations made by Elliot and Willingham represent a major challenge to today's auditor. Although it may be possible to design systems, even real-time systems, with audit trails, the auditor is faced with the fact that these trails may not exist. This is particularly true in a computer fraud investigation since part of the fraud scheme may be to cover up audit trails which do exist. Ideally, the auditor should be integrally involved in the design of new EDP systems, ensuring that they are auditable and properly controlled. However, the computer fraud investigator should be able to deal with inadequate audit trails and be suspicious of even those which appear adequate.

Numerous articles in the current literature on EDP auditing suggest that the conventional audit trail may be eliminated altogether in future systems and auditors are advised to develop procedures that are not dependent on these trails (Cook and Winkle 1980). However, alternative audit procedures have not been easy to devise since procedures designed to trace transactions back through systems to source documents using an audit trail represent basic operating technique for auditors. Further, it is not a foregone conclusion that audit trails will completely disappear from EDP applications.

While the technology seems to be pushing development in this direction due to increased operating efficiencies such as speed of information delivery and reduction of paper usage, there are valid reasons for maintaining audit trails. Some of the reasons are the use
by management of transactions detail, internal and external audit requirements, safeguards against system error or breakdown, and requirements of the Internal Revenue Service. In this regard, Cook and Winkle cite the position of the IRS as stated in Revenue Procedure 64-12 under the heading "A.D.P. Record Guidelines":

Supporting Documents and Audit Trails - The audit trail should be designed so that the details underlying the summary accounting data, such as invoices and vouchers, may be identified and made available to the Internal Revenue Service upon request.

Recorded or Reconstructible Data. The records must provide the opportunity to trace any transaction back to the original source or forward to a final total. If printouts are not made of transactions at the time they are processed, then the system must have the ability to reconstruct these transactions.

Basically, three procedures for auditing in an EDP environment have evolved over the past few years; Auditing around the computer; Auditing through the computer and Auditing with the computer. The following discussion is patterned largely after Cook and Winkle, however, these procedures are quite common in the literature.

Auditing Around the Computer

Where a detailed audit trail exists auditors may be able to perform examinations without utilizing the computer. The basic procedure resembles the non-EDP audit in that print-outs of the details of account balances and the associated transaction print-outs are first obtained. Then, specific transactions are selected and traced back through the accounting records to appropriate source documents.

In certain applications this procedure may provide adequate evidence upon which to form an opinion with respect to the data. As
long as the auditor is fully informed regarding the limitations and potential pitfalls of such an approach and works within these constraints, there is no reason not to audit around the computer.

A couple of observations should be made, however, regarding the use of this approach in investigating for computer fraud. First, it should be noted that the approach is oriented mostly to the evaluation of transactions. Although transaction manipulation is certainly a significant computer fraud scheme as evidenced by its prominence in reported fraud cases, it is only one of several possible scheme types. Further, as indicated earlier part of the fraud scheme might be to purposely cover or distort the audit trail. Thus, while the technique of auditing around the computer might be useful in the computer fraud investigation in certain circumstances it will most likely have to be augmented with other techniques.

Auditing Through the Computer

If little hard copy exists and a discernible audit trail does not exist, the technique of auditing around the computer is likely to be inappropriate in a general audit and certainly will be of little use in a computer fraud investigation. An approach which might be useful in this situation is that of auditing through the computer. This technique uses the computer hardware and software to test data. Two common methods of auditing through the computer are the use of test data and reprocessing.

Use of Test Data

Using this approach, the auditor develops data similar to that
normally processed by the system. Then, these specially developed data are processed through the system in a controlled environment. Finally, the resulting processed information is compared with predetermined results to ascertain whether the test data were processed properly.

The test data should include both valid and invalid conditions. This approach provides evidence regarding the operation of the computer system at a given point in time. It provides little evidence regarding the operation of the system over time, for example during the entire period under examination. "In order to satisfy auditing standards, use of the Test Data method must be coupled with an examination of source documents and other source evidence supporting the records that are being produced." (Davis 1968).

Reprocessing

The Reprocessing approach requires that samples of actual data from the period under examination be reprocessed under controlled conditions. To perform this operation the investigator either uses a program in the EDP department that previously has been tested or uses a duplicate program over which the auditor maintains control. Upon completion of the reprocessing, the results are compared with the original data, as recorded in the system's records. One advantage to this approach over the Test Data approach is that the auditor may draw inferences regarding both the operation of the computer program and the validity of the data being processed.

Auditing through the computer has numerous advantages for the computer fraud investigator over auditing around the computer. However
there are computer fraud scheme/perpetrator combinations which might not be detected through this approach. For example certain file manipulation or improper operation schemes may not be detected through either the Test Data or Reprocessing approaches.

Auditing with the Computer

Computer programs can be devised so that various audit tasks can be efficiently performed using either internal or external EDP equipment. Generalized computer audit programs are available which accomplish a wide range of audit tasks and are adaptable to a number of different EDP systems.

Use of generalized computer audit programs may provide increased access to the data stored in the system. They can also be used to present the data in a more meaningful format than might otherwise be easily achieved.

The following audit tasks are commonly performed by generalized computer audit programs:

1. Statistical and judgemental selection of audit samples.
2. Mathematical verification of footings, extensions, and other calculations.
3. Multiple regression calculations or other procedures to select items for analytical review.
4. Data manipulation chores such as computation of subtotals, summarization, listing selectively, etc.
5. Examining records for completeness and correctness.
6. Comparing data that appear on separate files.

Other applications of computer audit programs include preparation of financial statements, preparation of accounts receivable
aging analysis, printing and addressing confirmation requests, comparison of budgetal and actual amounts, computation of ratios and other statistics, listing slow moving inventory items, matching credit limits to receivable balances, comparing physical inventory counts with master files, etc.

For computer fraud investigations, as with general audits, generalized computer audit packages can provide valuable assistance. However, they should be viewed as tools which can facilitate the investigative process rather than a means for actually performing the investigation.

Further discussion of computer audit programs is provided in Appendix B.

Investigation Of Threats From The Typology

In the remainder of this chapter an attempt will be made to provide some insight into the investigation of the computer fraud threats identified in the typology. Recall that twenty-one scheme/perpetrator threats were identified in the typology. These threats and their associated threat values are shown in Chapter 3 - Figure 13. The threats in Figure 13 may be grouped into the following broad categories:

1. Transaction Manipulation Schemes
2. Unauthorized Program Modification Schemes
3. File Manipulation Schemes
4. Improper Operation
The approach for investigating these schemes from the typology varies as does the applicability of the above audit approaches. The relationship between these schemes and the above audit approaches and other suggested investigative schemes is provided below:

**Transaction Manipulation Schemes**

Recall that transaction manipulation schemes may be further categorized as:

1. Transactions Added
2. Transactions Altered
3. Transactions Deleted

A primary factor in determining the investigative approach for transaction manipulation schemes is the existence or nonexistence of transaction registers. If the system automatically records critical information anytime a transaction is processed whether it is added, altered or deleted, it may be feasible to "Audit Around the Computer". However, the mere existence of a transaction register providing an apparent audit trail may not be adequate. If the transaction register can be bypassed or manipulated its use would be questionable.

If there is not a transaction register, or if the register may be bypassed or manipulated it will probably be necessary to use more stringent investigative procedures. The use of "Auditing Through the Computer" with Test Data or Reprocessing schemes might be adequate. However, in certain cases it might be possible for the computer fraud perpetrator to distort the results of this approach. For example, program changes or "patches" might have been invoked when certain data
was originally processed allowing transactions to process fraudulently. These changes or "patches" may have since been removed causing Test Data to process accurately.

It is suggested that a more aggressive investigative approach might be required. This approach which will be entitled "Live Monitoring" would select certain live transactions as they are being processed for investigation. Since these transactions would be randomly selected and unannounced it is possible that, in the above instance, a transaction being processed while the fraudulent program changes or "patches" were in place would be investigated, leading to the disclosure of fraudulent activity.

In summary, Transaction Manipulation - the most common type of computer fraud reported, may require any or all of several investigative approaches. Although the less aggressive approaches might be adequate in certain situations, their use should always be evaluated from the viewpoint of the fraud perpetrator who quite possibly is clever enough to cover up his or her tracks.

Unauthorized Program Modification Schemes

Program Modification Schemes are, perhaps, the most insidious of schemes and the most difficult to investigate. Program patches may be included in a "special" run, then removed so that the programs behave appropriately for normal operations. Further, part of the patch may be a special process to wipe out any evidence of the special run. Obviously, the typical perpetrator, clever enough to invoke such a scheme, is not going to leave a documented record of his or her
activities for an auditor to review after the fact. Thus, it is suggested that a "Live Monitoring" approach similar to that described above for difficult Transaction Manipulation Schemes will be required to effectively investigate Unauthorized Program Modification Schemes.

File Manipulation Schemes

File Manipulation Schemes are typically variants of Transaction Manipulation and Program Modification Schemes. Thus, the investigative approaches described in those sections generally apply. Once again, it may be necessary to invoke a "Live Monitoring" approach since "dummied-up files" can be run, then replaced with authentic files with no trace of the activity. The "Live Monitoring" approach enables the investigator to catch the perpetrator "red-handed" during the fraudulent process.

Improper Operation

The "Live Monitoring" approach is probably the only reasonable approach to investigating the Improper Operations Scheme. It is doubtful that any scheme involving the improper operation of the computer system will be documented for the benefit of the auditor or investigator. Thus, it is suggested that random, surprise visits whereby the current operations are investigated be used.

Summary

At the beginning of this chapter a distinction was drawn between an "Investigation" and an "Audit" in order to explain entitling the chapter "The Investigation" rather than "The Audit". It should be
evident at this point that in dealing with computer fraud the examination more clearly resembles an investigation than a traditional audit. This is not to say that the examination should not be conducted by auditors. To the contrary, it most likely will be. However, in conducting the examination it is suggested that, in addition to possessing or having access to significant expertise in EDP, the auditor must think like an investigator. No longer is it adequate to come in after the fact and methodically pour through reams of source documents constituting an audit trail to piece together activities surrounding an organization and comment on their appropriateness.

"Live Monitoring" is suggested as an essential next step in the evolution of the audit process as it pertains to automated systems.
Appendix B

Automated Analysis

The methodologies for computer fraud detection presented in this thesis may be facilitated to varying degrees by automated analysis. The specific threat assessment in Chapter Five contains several steps such as the completion of the threat matrix and, certainly the manipulations in the Churchman-Ackoff Ranking Process, which would benefit from automation. The Resource Optimization Model in Chapter Seven which utilizes the solution to the Combinatorial Dilemma presented in Chapter Six literally demands the use of automation.

The purpose of this chapter is to suggest specific techniques of automated analysis to support the above methodologies and provide information on available audit packages.

Threat Assessment

The process of identifying and evaluating threats described in Chapter Five - "Specific Threat Assessment" lends itself quite well to automated support. The processes which would benefit the most from automation follow.

Churchman-Ackoff Process

Referring back to Figures 24, 25 and 26 it is apparent that the Churchman-Ackoff process of first ranking the threats in the order of

161
their importance; then, iteratively comparing threat values of specific threats to combinations or other threat values based on these comparisons is both labor intensive, if accomplished manually, and quite amenable to automation.

The suggested approach is an interactive computer program whereby the decision-maker or group enters their identified threats and initial values into a terminal. Then, have the computer offer the choices between the identified threats and the various combinations of other threats described in Chapter Five. Finally, the computer program should check for inconsistencies and automatically adjust threat values to correct for them, replacing the manual process demonstrated in Figures 25 and 26.

 Threat Matrix

Following the identification of system scheme and perpetrator threats and their ranking using the Churchman-Ackoff Method, a threat matrix such as that in Figure 29 must be developed. This process combines scheme/perpetrator combinations and their associated threat values in matrix cells. In addition it is the beginning point for the Controls Analysis also discussed in Chapter Five.

Automation of the threat assessment routines including the Churchman-Ackoff, Threat Matrix and Controls Analysis processes should be relatively straight-forward, following the descriptions in Chapter Five.

  Resource Optimization Model

Automation of the Resource Optimization Model is considered a
virtual necessity. Although the model might conceivably be performed manually in certain situations, its labor intensity would probably preclude manual application in all but the simplest of systems. Following the solution to the Combinatorial Dilemma presented in the flow chart in Figure 32 (Chapter Six) and the description of the Resource Optimization Model in Chapter Seven, automation should not be too difficult.

The Iterative Discovery Sampling Technique is fairly straightforward. The process will be simplified significantly through the use of a "Random Number Generator" software package since the process requires random selection of potentially thousands of combinations.

**Internal Control**

Lieberman (1977) developed a "Methodology for the Automation of the Internal Control." Lieberman proposed a methodology for aiding by computer an analysis of a plan of internal control. The methodology consists of three major steps. First, a formal documentation of the company being audited is prepared. This document describes those functions of the client that will be evaluated by computer. This documentation serves as a model, representing processes, people, data and their associated interrelationships. The model is constructed in a formal language called "PSL/A" and stored in a computer database.

The second major step in Lieberman's methodology is a set of rules defined by the auditor. These rules take the form of search operations representing the evaluation criteria that the auditor uses during the audit process. The rules which are stated in a formal
language called "Rules" describe allowable and required entities, conditions and relations in the model of the client. The rules, in effect, describe an ideal plan of internal control and subsequently will be compared to the documented plan of internal control.

The final step in Lieberman's approach is an evaluation process which reads through each rule and searches through the data base under control of that rule. Any conditions in the data base that are in violation of the rule are reported as possible weakness in the plan of internal control evaluation as performed in an interactive mode, allowing the auditor to discover possible weaknesses and then exploring them further with additional rules.

The Investigation

Numerous automated systems exist which can facilitate the investigative process described in Appendix A. Many of these systems are available "off-the-shelf" for general use.

Recall that the investigation encompasses such techniques as; Evaluation of Internal Control; Auditing Around the Computer; Auditing Through the Computer; Auditing With the Computer; processing of Test Data; Reprocessing and Live Monitoring. All of these techniques may be facilitated by existing software packages.

Audit software provides the auditor or investigator with numerous capabilities for performing the actual investigation once systems threats have been identified, categorized and ranked. Perry and Kuory (1980) break these capabilities into fourteen categories:

Analyzing Records
Performing Computations
Comparing Two Files
Comparing Two Fields
Stratifying Files
Selecting a Random Sample
Resequencing Data
Summarizing Data
Preparing Data for Printing
Building Files
Restructuring Information
Updating Files
Statistical Analysis
Simulating Portions of a Whole System (by Parallel Simulation).

These capabilities relate, to varying degrees to the different techniques of auditing or investigating a system for computer fraud. A brief discussion of the capabilities from Perry and Kuory follows:

Audit Software Analyzes Records. Audit software provides the ability to perform an analysis on the information in records and data bases. The purpose of this analysis is normally to identify certain information for audit follow-up purposes. For example, audit software could analyze accounts receivable records to determine if the balances were positive or negative. The result of this analysis would be a listing of all of the accounts receivable records that carry a negative balance. The analyst looks at one field at a time and makes decisions bases on that examination. The types of analysis that can be performed on a specific field include:

- Whether it is positive or negative
- Whether it has a specific value (e.g., the state of Florida)
- Whether it is greater, equal, or less than a specified value (e.g., greater than $1,000)
- Whether it is numeric or alphabetic
- Whether it is zero
Information in data bases frequently cannot be accessed directly by audit software. However, the information in data bases can usually be converted to a "flat file" (i.e., a sequential record file) and then analyzed through the use of audit software.

Audit Software Can Perform Computations. Audit software is capable of performing any of the common accounting computations. These normally include addition, subtraction, multiplication, and division. Some audit software packages have the capabilities for calculating percentages. However, when special mathematical computations are needed, such as sum of the years' digit depreciation, the auditor will normally have to write the algorithm for performing that calculation using the previously mentioned computation capabilities. This computational capability enables auditors to perform such calculations as verifying gross wages by multiplying hours worked times hourly wage.

Audit Software Provides The Capabilities To Compare Two Files. This capability enables the auditor to examine two files simultaneously. This comparison capability enables the auditor to determine if records on the two files are equal. Other comparison capabilities, such as greater than, and less than might be available, but the most commonly used comparison is for equal or unequal values or records. This type of capability enables an auditor to examine weekly payroll records from two consecutive weeks. One reason for doing this would be to search for new employees on either this or last week's payroll file that are not on the other weeks' file. By comparing the two files, all equal records would be ignored, and only unequals would be listed.

Audit Software Provides The Capability Of Comparing Two Fields. This capability enables the auditor to determine between two fields, which is greater, which is the lesser, and if the two fields are equal or unequal. This is an analysis that is performed using two fields instead of one. This comparison can be performed with one record, or between records. For example, if an order was comprised of many line item records, the auditor might want to compare invoice number from one record to the next to determine if the line item record being examined belongs to the current invoice being analyzed, or to a new invoice.

Audit Software Can Stratify Files. This is one of the more powerful capabilities of audit software that is not available as a command in most computer languages. Stratification will determine how many records in a file fall into a given category, or strata, and the value and number of the records in that strata. This is extremely valuable when sampling is being
used. For example, an accounts receivable file could be stratified by accounts receivable balance. The auditor first decides the number of stratas desired. For example, the auditor might pick three stratas, one from zero to $1,000, the second from $1,000.01 to $3,000, and a third for those balances over $3,000. Then by example to contents of each strata the auditor can decide upon the sampling method.

Audit Software Can Be Used To Select A Random Sample. This capability permits the auditor to quickly generate a sample for audit purposes. One of the major drawbacks to using sampling has been the complex procedures an auditor had to follow in selecting a sample. Manually performed, selecting a sample can be an extremely time-consuming task. Computer audit software has automated this task and made random sampling practical. Different packages use different sampling algorithms. Some of the sampling methods available include:

- Every Nth item starting with item X
- A true random sample that will give X% of the file
- Dollar value sampling
- A sample designed to provide a predetermined confidence limit.

Audit Software Has The Capability To Resequence Data. Frequently computer files are not in the sequence that auditors need for analytical purposes. Some audit software packages have the capability to resequence the order in which data appears for processing. For example, if a payroll file is in employee number sequence, an auditor can have that file resequenced using audit software so that it would appear in department number sequence. Some audit software packages have the capability to resequence a few records at a time, but this is unusual. Most resequencing is done with entire files at one time. Some audit software packages use a sort utility program to perform this resequencing.

Audit Software Can Summarize Data. Many audit software packages have the capabilities to accumulate one or several levels of totals for a specific field. For example, if an auditor wanted to get payroll totals by department, by division, by plant, and by company, there would be four levels of totals. This summarization can be performed in two different ways. One way, all the detail records are listed with a total appearing each time there is a break in sequence. For example, when one department has ended, the department total is listed before the next department number starts. Then when division numbers change, the division total is printed before the next division records commence. The other way of handling summarization is only to list the totals.
example, payroll could be summarized and only list one payroll summary report line for each department in the company.

Audit Software Can Prepare Data For Printing. Data as represented in internal computer format may not be legible when printed in that format. For example, some methods of storing numerical data internally (i.e., packed decimal) in a computer cannot be printed in a legible format without some preparation work performed prior to printing. Even alphabetic data may not be readily readable without preparation work. For example, people's names may be stored without blanks in the computer. Without knowing the key to decode the data for printing purposes, the auditor would have to guess where the last name stops and the first name starts. In addition, numerical data does not contain decimal points, commas, or negative or positive value signs that are readily understandable when printed using internal computer coding. All this information must be included with the numerical values before they are printed. Audit software has the capability of doing much of this automatically.

Audit Software Can Build New Files. Many times the auditor needs to build a file for use in later processing. For example, the auditor may want to build a file of inventory test counts that are to be made. Once this file has been created, it can be used to compare actual counts to book values, using one of the other audit software capabilities, in order to determine if the test counts equal the book, physical inventory. Files can be built from scratch, or they can be built using information contained in other files.

Audit Software Can Restructure Information. Frequently, information in a computerized record is not in a format that the auditor needs. For example, states may be carried as a code from one through fifty, rather than as an English word. For example, the state of Florida may be code thirteen. If the code was not restructured (i.e., translated into something legible), it would mean that the auditor would either have to memorize the codes, or look each one up in a catalog every time it is encountered. Other types of data that require restructuring for presentation purposes include:

- Changing minus signs to "CR" or the word credit
- Converting numeric codes to English words (such as employee number to employee name)
- Encrypting and decrypting data

Audit Software Can Update Files. Audit software has the capability to create and recalculate files containing information which can be updated. Updating can mean adding values, changing values, or deleting values. For example, if
auditors are making numerous test counts, and inventory is scattered throughout many locations, the updating capability enables the auditor to combine the value of the various counts of the same product on a work file.

Audit Software Provides The Capability Of Performing Statistical Analysis. Some of the newer audit methods use statistical analysis. For example, certain data relationships can be shown through regression analysis. Some of the simpler statistical analyses performed include generation of bar graphs, generation of histograms, and trend charts. Some of the more complex statistical analysis available include linear programming, trend line analysis, correlation analysis, and multiple regression analysis.

Audit Software Provides The Capability Of Simulating Portions Of A Whole System (By Parallel Simulation). Audit software can duplicate production processing for audit purposes. For example, an auditor could develop a routine using audit software commands to calculate the interest on saving passbook accounts. The amount calculated by the auditor is then compared to that the bank calculated for their depositors using the production systems. Any discrepancies would be uncovered by the parallel simulation run.

Audit Software Packages

Selection Of An Audit Software Package

There are numerous audit software packages available. The selection of a generalized audit software package is a process of determining which of the available packages meets an organization's needs. Referring back to Chapter 8 recall that factors such as the existence or nonexistence of audit trails; real time processing environment; batch processing environment; etc., all affect the investigative techniques used to detect computer fraud. Further, the type of threats in a system might influence the selection of a specific package. As with most generalized software packages, audit software packages have different capabilities, strengths and weaknesses
which must be matched to the requirements. In addition, operational and economic considerations must be evaluated. Thus, factors such as the cost of the software package; the equipment required to process it and its cost; efficiency of use; ease of learning; etc., must be evaluated.

The literature provides rather extensive coverage of the generalized audit software package selection process as well as the use of audit software. A few sample packages are provided below as reported in a special publication of the U.S. Department of Commerce, National Bureau of Standards.

Sample Packages

The U.S. Department of Commerce, National Bureau of Standards (NBS) published a document giving the features of seven audit software packages (NBS Special Publication 500-13 1977). Although other packages are available, these seven packages basically typify the capabilities and features of generalized software packages.

The NBS document includes the following packages:

Auditape
DYL 260
Easytrieve
EDP-Auditor
Hewcas
Mark IV/Auditor
Score

The seven packages are compared on the basis of:
Availability And Cost Of Software. Different software distribution plans exist. Software to be installed at a customer's site may be available for purchase or lease. Software may also be available from time-sharing service centers. The rental and purchase costs for each of these systems were obtained from the General Service Administration (GSA) schedule, where available, or from the software vendors. In certain cases a discount is allowed if more than one copy of the system is acquired within the same government agency.

History Of Software Package. Historical information on the implementation of the software package indicates the extent of development and field usage. The specific information concerns the date of first installation and the name of significant installations.

General Systems Characteristics. The basic design of an auditing software package is important in analyzing the potential performance, flexibility, and transferability of the package. Other characteristics are the availability of separate functional components of the package, and the method of combining these components to generate data and information useful to the auditor. The language in which the package is written is of importance if changes have to be made to the program. Also of importance are the ability to link to other programs, and the ability to accommodate user generated routines, macros or programs.

Modes Of Use. The operator interaction with the software package is described here. Various specifications must be supplied to an auditing package describing client's file characteristics, functions to be performed, and desired presentation of results. The object computer configuration, i.e. the configuration of the computer on which the audit package is run to produce audit results, must also be specified. These specifications can be communicated to the audit program in an interactive dialog from a terminal, or they may be entered by punched cards, requiring preparation of multiple choice forms, questionnaires, coding forms, or other written instructions. Number of forms, detail specified, ability to accommodate changes, affect ease of use of the package.

Computer Environment. The auditing software package may produce a program capable of running on a computer different from the one on which the initial parameters were specified. The computer on which the initial parameters were specified is called the source computer. The computer on which the auditing software executes the audit functions and derives audit data is called the object computer. The source computer could be a computer at the auditor's location, while the target computer
may be a machine at the client's site. The source computer and the object computer could also be the same computer.

**Data Types.** A variety of data types that exist in various client's files must be accommodated by general purpose audit packages. These data types are machine representations, but are dealt with by the auditing software in a logical way. Examples of data types which the vendors are supporting are numeric, numeric signed, alphanumeric, decimal, binary, floating point etc.

**Input File Characteristics.** A variety of files and file types are typically accessible. These files may be sequential, index sequential, or their organization may depend on a particular data base system. Record types may be of fixed length or of variable length. Records may be grouped in blocks or they may be unblocked. Files may have ANSI standard labels, industry standard labels, non-standard labels or no labels. A system may have to recognize various file marks such as "end of reel", "end of file", or "end of volume". In some applications, the auditor must specify detail in the client's files which is to be processed by the audit system. The files may be either processed directly, in which case complete specifications for all format detail must be stored in the object computer, or an intermediate file may be prepared, which is in a standardized format.

**Media Characteristics.** There exist a variety of media on which information can be recorded. This section summarizes media which can be handled by the various systems. Most systems provide flexibility for accepting files on cards, tapes, or disks. Variations of media codes, such as tape codes of various manufacturers usually are handled by different versions of software packages.

**Basic Functions And Utilities.** Includes processing functions, which permit the auditor to analyze client files, and to display client data in a form suitable for analysis by the auditor.

**Copying.** Although there usually are utility programs available at the clients site, some generalized audit packages provide for file copying capabilities. Detail on systems analyzed are given below. In some cases the copying function includes a selection capability.

**Sorting.** Sorting refers to the arranging of a set of records in a specified order, according to sort keys. The order may be an alphabetic or numeric sequence. Ascending or descending sequences may be required. Several sort keys may be specified in some systems for one pass operation. In some system sorting
is combined with other input, processing or output functions to speed up or to simplify the auditor's task.

Multiple File Input. Ability to handle multiple files simultaneously on input improves flexibility of the package. This is useful for the comparison of files, for merging and matching operations. This capability requires the appropriate number of input equipments needed for simultaneous operation.

Merging. This refers to the combining of records of two or more files, that are each in the same sort order, into one file, in that order. Merging is useful for making files more understandable to the auditor e.g., a vendor name file may be merged with a sales file to provide vendor names rather than vendor codes to the auditor.

File Validation. File validation refers to the checking of data for correctness, or compliance with applicable standards, rules and conventions. Copies of files can be validated through bit-by-bit comparison with a master file. Discrepancies can be indicated and system reaction can be programmed.

File Matching. Matching of files permits the determination of identity of records or files. Duplicate records can be identified in one file, or by comparison of two files.

Numerical And Logical Operations. Auditing software provides assistance to the auditor in the evidence gathering process. The auditor collects data from client's files, and puts these data in a form which permits analysis and auditing. Both during the collection process and during the analysis of the data numeric and logical processes are used to combine fields, to compare fields, and to summarize and select data. Arithmetic operations such as addition, subtraction, multiplication and division are available. Other capabilities are the counting of records, simple computation of percentages, and computation of standard deviations. Relational operations permit comparison of magnitudes of numeric fields on the basis of operators such as equal to, not equal to, greater than or less than and combinations of these. Logical capabilities include AND, OR, and NOT operations, and a conditional operation similar to IF....THEN.

Arithmetic Operations. Arithmetic operations such as addition, subtraction, multiplication and division are available. Other needed capabilities are the counting of records, simple computation of percentages, and computation of standard deviations.
Relational Operations. Relational operations permit comparison of magnitudes of numeric fields on the basis of operators such as equal to, not equal to, greater than or less than and combinations of these.

Logical Operations. Logical capabilities include and, or, and not operations, and a conditional operation similar to IF...THEN.

Classification. Under this heading are grouped several functions which permit assignment of records into a set of predetermined classes. The two most common ways are stratification and aging. Classification permits the auditor to separate records or information into sets which then can be examined according to different criteria. It may be required to examine all disbursements above a certain dollar amount, but to only sample those below that amount.

Stratification. Stratification is defined here as the separation of records into sets of classes. Stratification may be based on dollar amounts, or dollar ranges, in which case these ranges must be specified by the auditor. Often the system provides a set of ranges as a default capability. Stratification may also be based on account classifications, or other criteria.

Aging. Aging refers to the placing of records into ranges on time based criteria. As an example all records may be classified into those from 0 to 3 months old, 3 to 6 months old, and older than 6 months.

Selection. The auditor selects those records from client files, which are of special interest to him. Selection may be based on record attributes, such as account number, vendor number, plant location etc., or it may be based on dollar range or account ranges, as defined during the classification process. Selection may also occur by means of sampling. Sampling may be done on a periodic basis, i.e. every tenth record may be selected, or it may be done on a random basis. Selection criteria may be combined by means of logical or relational operators. Selection may also occur on a temporary variable (a field produced as a result of an arithmetic calculation).

Summarization. Summarization refers to the summing of numeric totals, or subtotals, for items with common attributes, such as account number, vendor number etc. For purposes of internal control it is necessary to count and report total number of records. Summarization is used by the auditor to consolidate information of interest, which is fragmented in different files and records.
Diagnostic And Control Capability. In this section the basic diagnostic capability of the software package is discussed. Certain controls need to be exercised as part of an audit to assure integrity, and to maintain accuracy of the data. This pertains to the physical tape, the processing and the records resulting from the processing.

Report Preparation Capability. For many routine applications standard report formats are practical, and require only a minimum of specification. For ad-hoc reports, or one-time applications, report formats must however be flexible and must be easy to adjust. Points to be considered are number and widths of columns in reports, number of lines, heading and footing formats. Automatic editing capability with regard to dollar signs, commas and decimal points, zero suppression, etc., if required. Functions which are necessary for internal control and maintenance of an audit trail are automatic line numbering, page numbering, record numbering, the counting of records, and the provision of control totals as required by the auditor. Another feature often provided is the ability to generate confirmation notices. Often provision for printing of mailing labels is also provided.

The NBS document compares the above capabilities and features for seven generalized audit packages. A synopsis is provided below which provides a brief description of each of the packages. For the features of each package, the NBS document is referenced.

AUDITAPE. The system was originally developed in 1965 for internal use by a CPA firm in its audit practice. Based on its own experience and comments from clients the system was adapted for external use and is offered to clients and other organizations. A variety of tapes are available geared to various makes of computers and to specific operating systems. There are approximately 600 users, including city, state and federal agencies.

The system has been prepared for several makes and types of machines, including sets of programs specially prepared and programs furnished by the manufacturers. Auditapes are in machine language. Three different types of programs convert client file records into a standard Auditape format. IBM 1400 and Honeywell 200 machines use the basic edit program, DOS uses the manufacturer's utility, and OS and DOS/VS use the expanded edit program. A set of specialized programs manipulate these standardized files and produce the desired output.
DYL 260. System offered since 1973, over 600 installed. Listed in Datapro directory of commercial packages, won 1975 Datapro Award of Merit based upon satisfactory responses from user survey. Used by General Accounting Office and Veterans Administration.

A report writer, data manipulator, and file utility program, written in assembly language, and meant to be cataloged and used on the host computer. Required processing is controlled by simple parameters input on cards.


The system is written in assembly language and produces an audit report. It is a load-and-go system, and does not require intermediate program production and compilation. Options are available for interfacing with data base systems such as IMS and TOTAL. A CALL command permits use of auditor-written higher level language subroutines. A macro processor command processor permits use of auditor-defined macro routines.


EDP-AUDITOR is written in assembly language and produces an object code program which is executed to produce the audit listing. This type of system is a load-and-go type system. Up to 256 input files and 100 output files or reports can be specified. A library of routines is available supporting six auditing areas: file footing-control; exception analyses; summary analyses; special processing routines; confirmation of accounts; and selection and sampling. EDP-AUDITOR consists of the CULPRIT system plus the library of audit routines. Interfaces are available for data base management systems such as IDMS, IMS, DLl, TOTAL, DATACOM/DB etc., as well as for the TSO, complete cataloging and macro facility and exits are provided to user written programs.


HEWCAS is written in the BASIC programming language, and produces COBOL programs which can be compiled and executed on various machines, the user is prompted for the input specifications. Two simultaneous files can be handled on input, with any combination of media. Up to 91 simultaneous outputs are possible (one printer and 90 disk or tape files). No user written routines can be used, but produced COBOL programs can be saved. External program linkages are only possible by modification of the COBOL program.

MARK IV/AUDITOR. First delivery of Mark IV was in 1968. More than 1000 systems have been installed in the US and in other countries. AUDITOR is a special feature and was first introduced in 1976. Mark IV/AUDITOR has been installed at ERDA in Germantown, MD.

MARK IV/AUDITOR is written in assembly language and produces an assembly language program which is executed to produce results. The system can be considered a "load-and-go" system. It consists of the Mark IV system and the library of audit routines. Mark IV handles various data processing considerations in a manner transparent to the auditor. The auditor specifies the file characteristics and functions to be performed by filling out worksheets, which are translated into parameter cards. Standard file types and data bases are automatically supported, multiple files coordinated and up to 255 reports produced in one pass of the file. The auditor can sample, select, compute and report by specifying selection criteria and custom reports or by utilizing the library of audit routines. The library includes routines for aging, confirmation notices, random sampling, monetary sampling, stratification and grouping among others.


The system is written in COBOL and generates COBOL programs as output. Program generation is governed by about thirty different keyword parameter cards, of which generally only a few are necessary for any given run. The vendor states that "no superfluous PROCEDURE DIVISION CODE is generated, so that programs are only as long as they need to be". The user can insert his own COBOL code at almost any point.
Summary

Generalized audit packages such as those discussed above are typically oriented to assisting the internal or external auditor in forming conclusions regarding the accuracy and integrity of systems and their data. The audit traditionally has taken place after the fact, usually including processing for a given period or as of a certain date. This approach assumes, to a large extent, that certain internal controls and audit trails exist. Given that the necessary audit trails do exist, enabling the auditor to track transactions or activities all the way through the system, a fairly precise analysis can be performed. However, as indicated in Appendix A, many systems today with real-time, interactive capabilities are being built without audit trails. Further, in intentional acts such as computer fraud, the audit trail may be destroyed or modified to cover up the act. It is suggested that an investigative monitoring approach such as that addressed to in Appendix A will be necessary to evaluate systems with inadequate or suspicious audit trails.
REFERENCES


Auerbach. Lessons to be Learned From Computer Fraud. 1978-80.


