UC San Diego
Research Theses and Dissertations

Title
A Decision Support System for Infrastructure Management

Permalink
https://escholarship.org/uc/item/32f96185

Author
Staneff, Stephen T.

Publication Date
1997

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A Decision Support System for Infrastructure Management

by

Stephen Thomas Staneff

B.S. (University of Colorado, Boulder) 1986
M.S. (University of California, Berkeley) 1991

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy
in

Engineering-Civil Engineering

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor C. William Ibbs, Chair
Professor Robert G. Bea
Professor Robert M. Oliver

Spring 1997

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UMI Number: 9803366

Copyright 1997 by
Staneff, Stephen Thomas

All rights reserved.

UMI Microform 9803366
Copyright 1997, by UMI Company. All rights reserved.

This microform edition is protected against unauthorized
copying under Title 17, United States Code.

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A Decision Support System for Infrastructure Management

Copyright 1997

By

Stephen Thomas Staneff

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The dissertation of Stephen Thomas Staneff is approved:

Chair — Date: May '97

Date: May 6, 1997

Date: 5/4/97

University of California, Berkeley

Spring 1997

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To Mom

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

TABLE OF CONTENTS

1	**INTRODUCTION**	**1**
1.1	RESEARCH NEEDS AND OBJECTIVES	1
1.2	RESEARCH SUMMARY	2
1.3	DISSERTATION OVERVIEW	4
2	**PRIOR RESEARCH**	**6**
2.1	BACKGROUND	6
2.2	DESIRABLE CHARACTERISTICS OF AN INFORMATION SYSTEM FOR INFRASTRUCTURE MANAGEMENT	14
2.3	SUMMARY	16
3	**DEVELOPMENT OF A DATA STRUCTURE FOR AN INFRASTRUCTURE MANAGEMENT INFORMATION SYSTEM**	**17**
3.1	DESIRABLE CHARACTERISTICS OF A DATA MODEL FOR AN INFRASTRUCTURE INFORMATION MANAGEMENT SYSTEM	17
3.2	APPLYING A TRADITIONAL MODEL TO THE DAESSIM REQUIREMENTS: MANAGING SETS OF FACTORS	19
3.2.1	Flat Files	19
3.2.2	Hierarchical Flat Files	22
3.2.3	Traditional Evaluation Sub-Class Structure	25
3.2.4	Traditional Hierarchical Evaluation Sub-Classes	30
3.3	A NON-TRADITIONAL APPROACH TO DATABASE DESIGN: MANAGING FACTORS RATHER THAN SETS OF FACTORS	34
3.3.1	Preliminary Factor Name / Factor Value Design	34
3.3.2	Hierarchical Factor Name / Factor Value Design	39
3.3.3	Modified Hierarchical Factor Name / Factor Value Design	43
3.4	OTHER FEATURES OF THE AS-BUILT DAESSIM DATA STRUCTURE	49
3.4.1	Evaluation Objects	50
3.4.2	Handling of Outputs	52
3.4.3	Modeling of Physical Structures	53
3.4.4	Handling of Fleets	54
3.4.5	Handling of Scenarios	55
3.5	CONCLUSIONS	55
4	**DEMONSTRATIONS OF CAPABILITY**	**61**
4.1	INTRODUCTION	61
4.1.1	Chapter Overview	61
4.1.2	PIMS and Its Relationship to DAESSIM	61
4.1.3	Disclaimer	62
4.2	SAMPLE EVALUATION CLASSES INCLUDED IN PIMS	62
4.2.1	Information Flow for Simplified Risk Analysis of a Platform or Fleet	63

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.1.1	L1I1	66
4.2.1.2	L1RSR1	68
4.2.1.3	L1C1	70
4.2.1.4	L1Risk1	74
4.2.1.5	L1Risk1_Fleet1	77
4.2.2	Information Flow for Structural Analysis of a Platform	80
4.2.2.1	PlatformData	81
4.2.2.2	L2I1	84
4.2.2.3	L2VMax1	89
4.2.2.4	L2Eng1	91
4.2.2.5	L2RSRx - L2RSR1 and L2RSR2	92
4.3	DAESSIM VIABILITY DEMONSTRATIONS	98
4.3.1	Demonstration of Non-Proprietary Data Format, Ad Hoc Query Capability, and Referential Integrity	100
4.3.2	Demonstration of Database Searches	111
4.3.3	Demonstration of Iterative Capabilities	114
4.3.3.1	L1C1 Iteration Example	114
4.3.3.2	L2RSR1 Iteration Example	116
4.3.4	Demonstration of Technology Transfer	118
4.3.5	Demonstration of Altered Decision Criteria	119
4.3.6	Demonstration of Data Integrity	123
4.3.7	Tracking Changes Over Time	123
4.3.8	Other DAESSIM Capabilities	125
4.4	SUMMARY	126
5	**SCREENING CYCLE DECISION MODELING FOR INFRASTRUCTURE FLEETS**	**129**
5.1	BACKGROUND	129
5.2	PRELIMINARY ECONOMIC DECISION MODEL	138
5.3	CALCULATION OF CONDITIONAL PROBABILITIES	146
5.3.1	Matrix Calculations	146
5.3.2	Calculation of Decision Probabilities	148
5.3.2	Calculation of Branch Probabilities	151
5.4	REVISED ECONOMIC DECISION MODEL	154
5.4.1	Default Variable Values	154
5.4.2	Model Simplification	156
5.4.3	Model Structure	156
5.4.4	Sensitivity Analysis	159
5.5	THE IMPACT OF TIME ON OPTIMAL POLICY	168
5.5.1	Expected Cost of Failure Over Time	168
5.5.2	Sensitivity Analysis Incorporating Time	171
5.6	CONSERVATISM IN SCREENING SYSTEM DESIGN	181
5.7	ECONOMIC DECISION MODEL INTEGRATION INTO SCREENING SYSTEM MODEL	183
5.8	INTEGRATION INTO DAESSIM	187

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.8.1	Coding	187
5.8.2	Inputs	188
5.8.3	Outputs	189
5.8.4	Interface	189
5.9	OTHER APPLICATIONS	190
5.1	SUMMARY	191
6	**PLATFORM INFORMATION MANAGEMENT SYSTEM (PIMS) OPERATIONS**	**193**
6.1	INTRODUCTION	193
6.1.1	Warnings	194
6.2	CUSTOMIZING PIMS	196
6.2.1	Main Menu Operations	196
6.2.2	Adding a New Evaluation Type to PIMS	197
6.2.3	Adding a New Evaluation to PIMS:	210
6.2.4	Adding a New Structure Type to PIMS	215
6.2.5	Adding a New Structure to PIMS	217
6.2.6	Adding a New Fleet to PIMS	218
6.2.7	Adding a New Scenario Type to PIMS:	220
6.2.8	Adding a New Scenario to PIMS	222
6.2.9	Adding a Multiple Hierarchy Evaluation Type to PIMS	227
6.3	BUILT-IN EVALUATION TYPES	236
6.3.1	Using the Sample Evaluation Types	236
6.3.2	Creating a "PlatformData" Evaluation	239
6.3.3	Performing an "L1I1" Inspection	242
6.3.4	Performing an "L1RSR1" Analysis	242
6.3.5	Performing an "L1C1" Evaluation	244
6.3.6	Creating an "L1Risk1" Evaluation	245
6.3.7	Creating an "L1Risk1_Fleet1" Scenario	249
6.3.8	Creating an "L2I1" Inspection	254
6.3.9	Creating an "L2Eng1" Evaluation	258
6.3.10	Creating an "L2VMax1" Evaluation	259
6.3.11	Creating an "L2RSRx" Evaluation	261
6.3.12	Creating an "L2C1" Evaluation	263
6.3.13	Creating an "L2Eng2" Evaluation	264
6.3.14	Creating an "L2Pf3" Evaluation	265
6.3.15	Creating an "L2Risk1" Evaluation	267
6.3.16	Creating an "L1HMax1" Evaluation	270
6.3.17	Creating an "L1VMax1" Evaluation	271
6.4	SUMMARY	273
7	**CONCLUSIONS**	**274**
7.1	SUMMARY	274
7.2	FUTURE WORK	277
7.2.1	DAESSIM Implementation Issues	277

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

7.2.2	DAESSIM Applications	279
7.2.3	Comparison with STEP	281
	REFERENCES	**283**
	GLOSSARY	**288**
A	**REVIEW OF DATABASE TECHNOLOGY**	**296**
A.1	FUNDAMENTALS OF RELATIONAL DATABASE MANAGEMENT TECHNOLOGY	296
A.1.1	Basic Definitions	296
A.1.1	Flat File Databases	297
A.1.2	Relational Databases	298
A.1.3	Data Integrity and Normalization	305
A.1.4	Advantages and Disadvantages of Relational Systems	311
A.2	FUNDAMENTALS OF OBJECT-ORIENTED DATABASE MANAGEMENT TECHNOLOGY	314
A.2.1	Object-Oriented Terminology	315
A.2.1	Advantages and Disadvantages of Object-Oriented Systems	322
A.3	OTHER DATABASE TECHNOLOGIES	325
B	**DAESSIM ENTITY-RELATIONSHIP DIAGRAM**	**327**
C	**DAESSIM TABLE DEFINITION**	**331**
D	**SAMPLE VALUES OF P(U|"S") AND CORRELATION**	**393**
E	**FOUR-LEVEL DECISION TREE ("MODEL 5T") INCORPORATING LEVEL-SKIPPING AND TIME**	**397**
F	**SAMPLE CODE FOR MULTIPLE HIERARCHY EVALUATIONS**	**414**

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

LIST OF FIGURES

3-1	FLAT-FILE ENTITY RELATIONSHIP MODEL	20
3-2	FLAT-FILE OBJECT ANALOG	20
3-3	TABLE IMPLEMENTATION OF E-R MODEL	20
3-4	HIERARCHICAL FLAT-FILE ENTITY-RELATIONSHIP MODEL	23
3-5	HIERARCHICAL FLAT-FILE OBJECT ANALOG	24
3-6	HIERARCHICAL FLAT FILE TABLE IMPLEMENTATION OF E-R MODEL	24
3-7	TRADITIONAL SUB-CLASS ENTITY-RELATIONSHIP MODEL	27
3-8	TRADITIONAL SUB-CLASS OBJECT MODEL	28
3-9	TRADITIONAL SUB-CLASS RELATIONAL IMPLEMENTATION	28
3-10	TRADITIONAL HIERARCHICAL ENTITY-RELATIONSHIP MODEL	31
3-11	TRADITIONAL HIERARCHICAL OBJECT MODEL	32
3-12	TRADITIONAL HIERARCHICAL RELATIONAL IMPLEMENTATION OF E-R MODEL	32
3-13	PRELIMINARY FACTOR NAME / FACTOR VALUE ENTITY-RELATIONSHIP MODEL	36
3-14	PRELIMINARY FACTOR NAME / FACTOR VALUE RELATIONAL IMPLEMENTATION	37
3-15	HIERARCHICAL FACTOR NAME / FACTOR VALUE ENTITY-RELATIONSHIP MODEL	41
3-16	HIERARCHICAL FACTOR NAME / FACTOR VALUE RELATIONAL IMPLEMENTATION	42
3-17	MODIFIED HIERARCHICAL FACTOR NAME / FACTOR VALUE ENTITY-RELATIONSHIP MODEL	45
3-18	MODIFIED HIERARCHICAL FACTOR NAME / FACTOR VALUE RELATIONAL IMPLEMENTATION	46
3-19	ALTERNATIVE MODIFIED HIERARCHICAL FACTOR NAME / FACTOR VALUE RELATIONAL IMPLEMENTATION	48
3-20	EVALUATION OBJECT HIERARCHY	51
4-1	FORWARD CHAINING OF L1 RISK INFORMATION	65
4-2	CONSEQUENCE EVALUATION [AGGARWAL 1991], 1 OF 2	71
4-3	CONSEQUENCE EVALUATION [AGGARWAL 1991], 2 OF 2	72
4-4	L1RISK1 RISK EVALUATION CHART	76
4-5	L1RISK1_FLEET1 RESULTS CHART	78
4-6	FORWARD CHAINING OF L2 RSR INFORMATION	81
4-7	L2VMAX1 DATA INPUT CHART	90
4-8	L2RSRX END-ON LOADING CHART	94
4-9	L2RSRX BROADSIDE LOADING CHART	94
4-10	L2RSRX WAVE KINEMATICS CHART	95
4-11	L2RSRX AXIAL PILE CAPACITY CHART	95
4-12	L2RSRX END-ON LOADING RELIABILITY CHART	96

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4-13	L2RSRX BROADSIDE LOADING RELIABILITY CHART	96
4-14	BROADSIDE LOADING CHART, EVALUATION #5021, PRELIM DESIGN = "YES"	117
4-15	BROADSIDE LOADING CHARTS, EVALUATION #5022, PRELIM DESIGN = "NO"	118
4-16	L1RISK1_FLEET1 RESULT, EVALUATION #269	120
4-17	L1RISK1_FLEET2 RESULT, EVALUATION #4998	122
5-1	4-LEVEL SCREENING CYCLE [AGGARWAL 1991]	132
5-2	4-LEVEL SCREENING CYCLE [BEA AND CRAIG 1993]	134
5-3	DETAIL OF 1ST LEVEL OF SCREENING	135
5-4	DETAIL OF 1ST LEVEL OF SCREENING, WITH ECONOMICS	137
5-5	OVERALL VIEW OF TREE (ROOT NODE)	139
5-6	FIRST BRANCH, 4-LEVEL SCREENING CYCLE DECISION TREE	140
5-7	2ND BRANCH, 4-LEVEL SCREENING CYCLE DECISION TREE	141
5-8	3RD BRANCH, 4-LEVEL SCREENING CYCLE DECISION TREE	142
5-9	TWO-WAY CONTINGENCY TABLE OF EVENT OUTCOMES C	146
5-10	FORECAST PROBABILITY MATRIX F	147
5-11	DECISION PROBABILITY MATRIX, P	147
5-12	DECISION TREE WITH LEVEL-SKIPPING CAPABILITY ("MODEL 5"), UPPER HALF	158
5-13	TORNADO SENSITIVY ANALYSIS, ROOT NODE, MODEL 4	160
5-14	TORNADO SENSITIVITY ANALYSIS, NODE D1	161
5-15	TWO-WAY SENSITIVITY ANALYSIS OF F AND L4(1,2) AT NODE D1	162
5-16	TWO-WAY SENSITIVITY ANALYSIS OF F AND C4 AT NODE D1	163
5-17	TORNADO SENSITIVITY ANALYSIS AT NODE C4	164
5-18	TORNADO SENSITIVITY ANALYSIS AT NODE D2	165
5-19	TWO-WAY SENSITIVITY ANALYSIS OF F AND C3 AT NODE D2	165
5-20	TORNADO SENSITIVITY ANALYSIS, NODE D3	166
5-21	TWO-WAY SENSITIVITY OF F AND C2 AT NODE D3	167
5-22	TORNADO SENSITIVITY ANALYSIS, NODE C42	168
5-23	DECISION TREE FOR CALCULATING EXPECTED FAILURE COSTS OVER TIME	170
5-24	TORNADO SENSITIVITY ANALYSIS, NODE CO (ASSESSED AT DEFAULT Y=1)	171
5-25	TORNADO SENSITIVITY ANALYSIS, NODE C0 (ASSESSED AT DEFAULT Y=5)	172
5-26	TWO-WAY SENSITIVITY ANALYSIS OF F AND Y AT D1	174
5-27	TWO-WAY SENSITIVITY ANALYSIS OF F AND Y AT D202	174
5-28	TWO-WAY SENSITIVITY OF F AND Y AT D202, PFIR=0	175
5-29	TWO-WAY SENSITIVITY ANALYSIS ON F AND L4(1,2) AT D1	177
5-30	TWO-WAY SENSITIVITY ANALYSIS ON F AND L1(1,2) AT D1	177
5-31	TWO-WAY SENSITIVITY ANALYSIS ON F AND C4 AT NODE D1	178
5-32	TWO-WAY SENSITIVITY ANALYSIS ON F AND PFIR AT D1	179

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5-33	TWO-WAY SENSITIVITY ANALYSIS OF F AND IRR AT D1	179
5-34	TWO-WAY SENSITIVITY ANALYSIS OF F AND L4(1,1) AT D1	180
5-35	TORNADO SENSITIVITY ANALYSIS, NODE D1	181
5-36	SCREENING FLOWCHART INCORPORATING ECONOMIC DECISION MODEL, PART A	185
5-37	SCREENING FLOWCHART INCORPORATING ECONOMIC DECISION MODEL, PART B	186
6-1	MAIN MENU	196
6-2	ADDING AN EVALUATION TYPE	198
6-3	ADDING FACTORS TO THE NEW INSPECTION EVALUATION TYPE	199
6-4	ENTERING BASIC FACTORS	201
6-5	CREATING A NEW EVALUATION	210
6-6	DATA INPUT FOR INSPECTION	211
6-7	SELECTING AN INSPECTION FOR INPUT	213
6-8	REVIEWING DATA IN INPUT INSPECTION	214
6-9	VIEWING EVALUATION RESULTS	215
6-10	ADDING A NEW STRUCTURE TYPE	216
6-11	ADDING A NEW STRUCTURE	217
6-12	ADDING A NEW FLEET	218
6-13	PICKING A STRUCTURE TO ASSIGN TO A FLEET	219
6-14	ASSIGNING A STRUCTURE TO A FLEET (1)	219
6-15	ASSIGNING A STRUCTURE TO A FLEET (2)	220
6-16	CREATING A NEW SCENARIO TYPE	221
6-17	ADDING A NEW SCENARIO	222
6-18	PICKING EVALUATIONS TO USE IN A SCENARIO	223
6-19	REVIEWING DATA IN A SCENARIO'S COMPONENT EVALUATIONS	224
6-20	VIEWING SCENARIO RESULTS IN PIMS	225
6-21	VIEWING SCENARIO RESULTS IN EXTERNAL EDITOR	226
6-22	SUMMARY FORM FOR ASSIGNING HIERARCHIES	231
6-23	FORM FOR ASSIGNING SUB-OBJECTS	232
6-24	SUB-OBJECT ASSIGNMENT	233
6-25	SELECTING A SUB-OBJECT	233
6-26	COMPLETED HIERARCHY ASSIGNMENT	234
6-27	ADDING A PLATFORMDATA EVALUATION	240
6-28	ENTERING DATA FOR A PLATFORMDATA EVALUATION	241
6-29	SELECTING AN L1I1 INSPECTION FOR USE WITH AN L1RSR1 EVALUATION	243
6-30	L1RSR1 RISK EVALUATION CHART	248
6-31	ADDING AN L1RISK1_FLEET1 SCENARIO	250
6-32	PICKING L1C1 AND L1RSR1 EVALUATIONS FOR USE WITH L1RISK1_FLEET1 SCENARIO	251
6-33	L1RISK1_FLEET1 RESULTS CHART	252

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6-34	SELECTING A PLATFORM FOR FURTHER ANALYSIS	253
6-35	SELECTING AN L2I1 INPUT METHOD	255
6-36	MANUAL L2I1 INSPECTION DATA ENTRY OPTIONS	257
6-37	L2VMAX1 DATA INPUT CHART	260
6-38	SELECTING AN INSPECTION, AN ENGINEERING ASSUMPTIONS EVALUATION, AND A PLATFORM DATA EVALUATION FOR USE WITH AN L2RSR1 EVALUATION	262
6-39	L2RISK1 RISK EVALUATION CHART	269
6-40	L1VMAX1 DATA INPUT CHART	272
A-1	FLAT-FILE DATABASE	298
A-2	RELATIONAL DATABASE	298
A-3	TABLE DERIVED FROM FIGURE A-2	299
A-4	MANY-TO-MANY RELATIONSHIP	301
A-5	ENTITY-RELATIONSHIP DIAGRAM OF EXAMPLE DATABASE	303
A-6	E-R FOREIGN KEY DECLARATION (MODIFIED DETAIL OF FIGURE A-5)	305
A-7	EXAMPLE OF FIRST NORMAL FORM	308
A-8	EXAMPLE OF SECOND NORMAL FORM	309
A-9	EXAMPLE OF THIRD NORMAL FORM	309
A-10	OBJECT REPRESENTATION	318
A-11	INHERITANCE	319
A-12	MESSAGING	321
B-1	ENTITY-RELATIONSHIP DIAGRAM (PART 1 OF 3)	328
B-2	ENTITY-RELATIONSHIP DIAGRAM (PART 2 OF 3)	329
B-3	ENTITY-RELATIONSHIP DIAGRAM (PART 3 OF 3)	330

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

LIST OF TABLES

2-1	SUMMARY OF PRIOR SCREENING SYSTEMS	8
4-1	L1I1 OBJECT ATTRIBUTES (FACTORS)	67
4-2	L1RSR1 OBJECT ATTRIBUTES (FACTORS)	69
4-3	L1C1 OBJECT ATTRIBUTES (FACTORS)	73
4-4	L1RISK1 OBJECT ATTRIBUTES (FACTORS)	76
4-5	L1RISK1_FLEET1 OBJECT ATTRIBUTES (FACTORS)	79
4-6	PLATFORMDATA OBJECT ATTRIBUTES (FACTORS)	83
4-7	L2I1 OBJECT ATTRIBUTES, PLATFORM HIERARCHY	86
4-8	L2I1 OBJECT ATTRIBUTES, PLATFORM / DECK HIERARCHY	87
4-9	L2I1 OBJECT ATTRIBUTES, PLATFORM / BAY HIERARCHY	87
4-10	L2I1 OBJECT ATTRIBUTES, PLATFORM / ORIENTATION / BAY HIERARCHY	87
4-11	L2I1 OBJECT ATTRIBUTES, PLATFORM / BAY / HORIZONTAL BRACE HIERARCHY	87
4-12	L2I1 OBJECT ATTRIBUTES, PLATFORM / JOINT HIERARCHY	88
4-13	L2I1 OBJECT ATTRIBUTES, PLATFORM / ORIENTATION / BAY / DIAGONAL BRACE HIERARCHY	88
4-14	L2VMAX1 OBJECT ATTRIBUTES	91
4-15	L2ENG1 OBJECT ATTRIBUTES	92
4-16	L2RSRX OBJECT ATTRIBUTES	93
4-17	DEMONSTRATIONS OF DESIRED DAESSIM CHARACTERISTICS	99
4-18	ORIGINAL AND CONVERTED ATTRIBUTES	101
4-19	TOP 20 QUESTIONABLE PLATFORMS, STRU3394 VS. PIMS	112
4-20	L1I1 INPUT VALUES, EXAMPLE 4.6.1	115
4-21	ULSLEA 2.1 (PRELIMINARY VERSION) VS. L2RSR2 RESULTS	123
4-22	CHANGES IN PLATFORMDATA OVER TIME	125
5-1	VALUES OF P(U|"S") AND ρ_T	150
5-2	VARIABLE RANGES AND DEFAULTS	155
5-3	VARIABLE DEFAULT VALUES, MODEL 5T	172
5-4	VARIABLE DEFAULT VALUES, MODEL 5C	182
5-5	EXPECTED COSTS, INCREASING VS. DECREASING CONSERVATISM, $ THOUSANDS	183

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ACKNOWLEDGEMENTS

Many people deserve thanks for their direct and indirect contributions to this research. The work would not even have begun if not for the principle investigators, Bill Ibbs and Bob Bea. Bob Oliver of the Industrial Engineering and Operations Research department worked far above and beyond the call of duty as the "third committee member," and without him the research could never have been completed. The help of Mari Cook, Sally Olson, Karen Carkhuff, Suzanne Kempton, Lydia Briedis, and other University staff members is greatly appreciated, as is the support of all the members of the Construction Group faculty.

As he has done for so many other construction students, Glenn Ballard provided much of the funding for this research through his consulting business. My thanks go also to Charles Smith and Martin Eskijian of the U. S. Minerals Management Service and of California State Lands, respectively, for their roles in funding this research.

Of course, my friends and family have my eternal thanks for their patience, support, and sympathy over the past few years.

The decision tree models in Chapter Five were constructed using DATA version 3.03, produced and donated by TreeAge Software of Williamstown, Massachusetts.

Direct funding for this research was received from the U. S. Minerals Management Service (contract USDI-MMS 14-35-0001-30634) and the California State Lands Commission. Indirect funding for this research, or direct funding for related research, was received from the National Sea Grant College Program, the University of California (three fellowships), the Project Management Institute's Steven White Fellowship, Ballard Management Services, and Staneff Consulting.

The views expressed herein are those of the authors and do not necessarily reflect the values of any of the funding organizations or of any of their sub-organizations. The U. S. Government and the State of California are authorized to reproduce and distribute this report for governmental purposes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1 INTRODUCTION

1.1 RESEARCH NEEDS AND OBJECTIVES

In 1972, a call for help was raised [Wright et al 1973]:

> In addition to insuring safety of new construction, it is important to ascertain the safety of the large inventory of existing buildings . . .
>
> There presently are no nationally recognized effective systematic and economical procedures for assessing the hazards of existing buildings for future extreme loads . . . Appropriate evaluation methods should be developed for systematic predisaster surveys of safety for long term use . . . Criteria for acceptance or abatement of hazards should be capable of reflecting the responsible authorities assessment of the social and economic consequences of action . . .

Demand for effective ways to assess the safety of existing structures has not lessened since Wright's call. Regulators and fleet operators - and any group charged with the safe operation of large numbers of similar, existing structures - are increasingly faced with employing scarce resources to assess safety issues within their "domains." The problems vary with the type of structure involved, the characteristic(s) of interest, and the authority having jurisdiction. Following are a few examples:

- For the Bureau of Indian Affairs to continue with its plans to assess the safety of its dams, it first had to find out how many dams it had [Slade 1994].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Minerals Management Service regulators are charged with insuring the safety of over 3,800 offshore oil platforms in the Gulf of Mexico, for which little historical information has been maintained.

- There are approximately 577,000 bridges listed in the Federal Highway Administration's National Bridge Inventory, over 238,000 of which were rated as deficient in 1988 [Arockiasamy et al 1993].

These organizations and others like them require efficient methods to either perform or audit detailed safety assessments on the structures in their jurisdiction.

The goal of this research was to create a computerized system for the management of information relating to infrastructure risk and safety assessments. Among the priorities for the system were that it allow access to information for decision making and analysis, that it function in a non-proprietary format so as to facilitate information exchange, and that it be extensible to a wide and unanticipated variety of types of structures, fleets of structures, and analyses of those structures and fleets.

1.2 RESEARCH SUMMARY

A robust information management system requires a solid foundation. To fulfill that role, this research developed a hybrid database schema, "Relational Object" technology. Relational Object technology splits the database model into a relational substructure and an

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

object-oriented superstructure. The substructure manages individual factors rather than sets of factors, while the superstructure manages entities such as structures, fleets, and analyses. Implemented in DAESSIM (the Decision, Analysis, and Engineering Support System for Infrastructure Management), Relational Object technology provides an effective tool for the management and integration of information used in engineering analyses. Designed to be applied to the analysis of risk for infrastructure systems (fleets as well as individual structures), DAESSIM serves as a shell that may be customized to meet users' needs.

That ability was demonstrated with PIMS (the Platform Information Management System), an instance (version) of DAESSIM. PIMS was created to show that DAESSIM incorporates the desired characteristics of an infrastructure information management system, as developed in this research. The methodologies, attributes, storage protocols, and interrelationships of the sample evaluation objects included in PIMS were examined in the context of two major example types of "information flow" – 1) the information flow leading to a subjective assessment of the structural risk of an oil platform due to storm waves, and 2) that leading to an objective assessment of the reliability of an oil platform due to storm waves. These information flows provide insight into the structure of DAESSIM.

Perhaps the most advanced use to which the information stored in an infrastructure management system can be applied is a risk screening system. This research developed a sample 4-level screening system for later integration with DAESSIM. Test levels in the sys-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

tem progress from the "quick and dirty" to the expensive and complex. The screening system allows the user to employ common sense and economics to move to another structure within a fleet, to iterate a present type of analysis on the present structure using different assumptions, or to decide whether to accept a "safe" result for a structure. The latter, a recognition of the significant potential effects of even a small probability that an unsafe structure has been declared "safe," was analyzed in detail with a decision tree model. It was found to be both possible and necessary to calculate, rather than estimate, conditional probabilities for a multi-level screening system (starting with a small number of initial estimates of test quality). The ability to skip from an i-level test to an i+2 or i+3 level test was found to be critical in the decision model, as was the need to account for the time lag between screens. The assumption that screening system tests should decrease in conservatism with increasing complexity and cost was challenged, and found to be incorrect in one case. Given the example data used for demonstrating the model, the most influential factors upon the decision whether or not to continue testing after receiving a Level One "safe" result were found to be failure cost, the quality of a Level Four test, and the number of years until the next screening.

1.3 DISSERTATION OVERVIEW

Chapter 2 of this dissertation reviews prior efforts at infrastructure management and screening systems. Characteristics thought to be desirable in such systems are identified. Chapter 3 of this dissertation identifies the necessary features of a data structure whose

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

purpose is to support the type of infrastructure information management system described in Chapter 2, and presents DAESSIM as a prototype of such a system. Chapter 4 introduces the sample analysis methodologies that were added to an instance (version) of DAESSIM to produce PIMS, and uses PIMS to demonstrate that the desired characteristics, as identified in Chapters 2 and 3, have been provided. Chapter 5 then proposes an example 4-level screening methodology that can be incorporated in a version of DAESSIM to aid decision-making during the screening process. Chapter 6 describes the use of PIMS in detail, and Chapter 7 summarizes the research and discusses possible future work.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2 PRIOR RESEARCH

2.1 BACKGROUND

Past solutions to managing safety assessment processes centered around screening systems. These systems concentrated on controlling the level of analysis effort put into the safety assessment of one pre-selected structure at a time. If a structure passed an initial, cursory level of analysis, it was considered "safe"; if not, more effort was devoted to more rigorous levels of analysis, until either the structure passed or it was reasonably certain that the structure was "unsafe." The initial level of analysis was generally referred to as a "level one" analysis. Subsequent, progressively more detailed analyses used corresponding labels (thus, the most detailed analysis in a four-level scheme was referred to as a "level four" analysis). This review will cover screening systems intended for buildings, dams, highway bridges, and offshore platforms.

For buildings, Okada and Bresler proposed a screening system for seismic safety in 1976; Thurston, Dong, Boissonnade, Neghabat, Gere, and Shah followed with one of their own [Thurston et al 1986]. Predecessor seismic building studies included the 1969 to 1975 audit of pre-1933 San Francisco schools, and the screening of buildings in Long Beach in 1971 [Bresler 1977]. Bridges have been the subject of numerous efforts, including Weissmann, Hudson, Burns, and Harrison's [Weissmann et al 1989] Texas bridge management system module, the Organisation (sic) for Economic Co-operation and Develop-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ment's [OECD 1992] report on member states' bridge management systems, and Miyamoto, Morikawa, Kushida, and Tokuyama's [Miyamoto et al 1993] fuzzy-logic based expert system for bridge structural safety assessment. For dams, McCann, Franzini, Kavazanjian, and Shah [McCann et al 1983] put forth a screening system for failures stemming from a number of causes. Aggarwal proposed a system for Gulf of Mexico steel offshore platforms [Aggarwal 1991], and Bea and Craig did likewise, for Gulf and for West Coast platforms [Bea and Craig 1993]. Staneff, Ibbs, and Bea [Staneff et al 1994] followed the offshore screening system proposals with a working prototype of a system for West Coast platforms. The American Petroleum Institute (API) has developed its own screening system for US offshore platforms, and a draft of their proposal [API 1994] is reviewed here. An overview of the major features of the reviewed screening systems is given in Table 2-1.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

		Buildings				Bridges			Dams	Offshore Platforms		
		Long Beach 1971. Goal: identify bldgs subject to substantial seismic risk	Okada and Bresler 1976. Goal: evaluate seismic safety of low-rise concrete schools [1]	San Francisco 1977. Goal: assess seismic risks of pre-1933 schools [2]	Thurston, et al 1986. Goal: identify bldgs subject to seismic risk in region	Weissmann, et al 1989. Goal: Further rank bridge hazards after FHWA "first cut."	OECD 1992. Goal: review [3] Primarily data & insp. mgt prior to screening. Fleet vs. project decisions	Miyamoto, et al 1993. Goal: 1st-level screening of preselected bridges w/ encoded senior engrs' knowledge	McCann, et al 1985. Goal: rank dams within jurisdiction by risks they represent	Aggarwal 1991. Goal: simplified structural assessmen of GOM platforms	Bea and Craig 1993. Goal: Simplified structural assessment of existing US platforms [4]	API 1994. Goal: structural assessment of existing US platforms
Analysis-level features	Multiple levels of failure (damage)?	yes	yes	n/a	yes	n/a	n/a	yes	no	no	no	no
	Multiple causes of failure?	1, seismic	1, seismic	1, seismic	1, seismic	n/a	n/a	age, not event oriented	3 seismic, hydrological, upstream failure	1, storms	storms or quakes	storms or quakes
	Simple first pass	scoring system	no	no	yes	database search	yes [5]	ranking	no	no	scoring factors	triggers
	Cost/benefit analysis	n/a	no	n/a	no	yes	yes	no	yes	no	no	no
	Retrofit/abatement	n/a	no	n/a	no	no [6]	yes	no	yes	no	no	no
Both	Uncertainty handling	no	no	no	no	no	yes, service life	yes	yes	yes	yes	no
	Biases/error handling	no	no	no	no	no	n/a	no	no	yes	yes	no
Management features	Selects structures from initial pool?	no	no	no	yes	yes	yes [5]	no	yes	no	no	yes
	Number of screening levels	1	2; 1 described	1	essentially 2	1	1, expandable	1	1	4; 2 described	4; 1 described	essentially 5
	Able to compare structures at each stage?	n/a	no	n/a	limited	n/a	n/a	n/a	n/a	no	no	no
	Safety standards	implicit	explicit	n/a	implicit	implicit	n/a	implicit	explicit	explicit	explicit	implicit
	Decision/safety criteria modifiable?	no	with difficulty	n/a	no	no	n/a	no	with difficulty	no	no	no
	Decreasing conservatism with increasing detail?	n/a	no	n/a	no	n/a	n/a	n/a	n/a	no	yes	yes
	Statistical verification over time?	n/a	no	n/a	no	n/a	n/a	no	no	no	no	no
	Database management?	no	no	no	no	yes	yes	yes	no	no	no	no

Notes
1 Effectively a Level 2 analysis methodology
2 Effectively at least a Level 3 analysis methodology
3 A general description of several systems
4 Some of the information shown is based on the general work of Bea
5 Listed factors for later development of analyses
6 Considers costs of retrofit/abatement, unclear whether system generates these costs or uses district engineers' assessments

Table 2-1: Summary of Prior Screening Systems

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The systems reviewed were, to varying degrees, hybrids of analysis techniques and information management systems. The two have been inevitably intertwined: the type of analysis incorporated determined the required capabilities of the information system (e.g., a probabilistic structural analysis demands a management system capable of storing, displaying, and interpreting the results); and the purpose of the information system determined the type of analysis required (e.g., to identify the risks posed by seismically sensitive bridges, an assessment of bridge traffic loads must be included).

Most of the analyses were structural engineering procedures, and most considered only a single cause of failure. The first to consider multiple causes of structural failure was McCann et al.'s [1985] system, which considered problems due to upstream failures of other dams as well as hydrologically- and seismically-caused failures. Arguably, building and bridge analyses had no need to consider any failure modes other than quake-induced structural damage. Bea and Craig's [1993] and the API's [1994] offshore platform screening methods accounted for the most likely cause of failure depending on platform location: storms for the Gulf of Mexico and quakes for offshore California. All of the offshore systems pointed out the need to consider other causes of failure (such as operational error and mechanical failure).

The Long Beach building study [Bresler 1977] and the Thurston [1986] system introduced the concept of a "first pass" analysis that was simple enough to permit a rapid evaluation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of large numbers of structures. The purpose of such a simplified starting point is to select structures for analysis from the initial pool by identifying those that are most at risk: once identified, resources can be efficiently allocated to them. For types of structures that lack historical databases, such an extremely simplified first pass analysis appears to be the most effective element of the screening system. The disadvantage is that all structures must be reviewed before the most unsafe can be identified. Dams and offshore platforms are two types of structures that lack comprehensive historical databases. Bridge management systems have a wealth of information available to them, gathered from inspections and stored in a national database (the National Bridge Inventory). Thus a bridge system's "first pass" takes the form of a database search and a processing of risk factors (as in Weissmann [1989] and OECD [1992]).

For structures lacking extensive historical information, a screening system can theoretically be designed to select structures for further analysis from an initial pool or fleet, it can be designed to direct analysis efforts on one structure, or it can be designed to do both: select the structure, and then manage engineering efforts upon it. The difference is in the management of the analyses, not in the analyses themselves. Bea and Craig's [1993] system, while featuring a simple first pass methodology, focused on one platform at a time; thus a "not fit for purpose" first pass result initiated a more detailed "level two" analysis rather than a search within the fleet for other, more risky platforms. API [1994] used triggers for its first pass, effectively handling the initial selection of structures for analysis. Again, however, once a platform was selected, the management process did not allow

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

comparisons with other platforms. More efficient management should be possible if the results of each analysis for each structure were stored, allowing the user to compare that structure with similar structures in order to decide whether to continue with it, or to move on to another, less "safe" structure. This would be fleet-level rather than platform-level management.

The handling of bias and uncertainty is an important part of the design of an information system, as well. The dam, platform (excepting API [1994]), and bridge methodologies (excepting Weissmann et al. [1989]) employed probabilistic analyses. Miyamoto et al.'s [1993] bridge ranking procedure was notable for using fuzzy logic to simplify uncertainty handling in an initial, cursory screening analysis. Aggarwal [1991] and Bea and Craig [1993] introduced the ability to handle biases in basic parameters.

Aggarwal [1991] and Bea and Craig [1993] also included provisions for graphically displaying uncertainties, many of which were implemented in Staneff, Ibbs, and Bea [1994]. Miyamoto [1993], however, textually presented quantitative, non-intuitive judgment factors and fuzziness factors. The contrast between the Miyamoto approach and the others demonstrates that any system that incorporates uncertainties at the analysis level must be able to handle them at the storage, display, and interpretation levels.

All screening systems use some sort of decision criteria or standard to determine a structure's adequacy. The bridge systems used decision criteria such as "adequate traffic

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

flow." Okada and Bresler [1976] used criteria for "repairable damage" and "strong damage" to buildings, but did not discuss how these criteria were determined. McCann [1985] used a similarly predetermined, though probabilistic, set of criteria for dams. API's [1994] platform safety criteria were obscure, presumably based on previous API attempts at standard-setting. Aggarwal [1991] and Bea and Craig [1993] included a sliding standard of allowable probability of failure vs. consequences, based on both economic performance and industry standards-of-practice. A key to using a safety standard is the ability to understand it and its implications. Only the Aggarwal [1991] and Bea and Craig [1993] methodologies approach that ideal. Staneff and Ibbs [1994] took an in-depth look at the Aggarwal [1991] / Bea and Craig [1993] type of Whitman-derived safety standard, and introduced the concept of fleet risk (the impact of a safety standard not upon a single structure, but upon the entire fleet of affected structures).

Starting from the ability to understand a safety standard/decision criteria, the user may wish to modify it and to view the consequent effects upon previous rankings or decisions. For example, would the Bea standard produce different results than the Iwan standard [Staneff and Ibbs, 1994]? The ability to adjust and analyze decision criteria is a feature that has yet to be introduced to the screening system field.

Several of the reviewed systems attempted to directly compare results between simplified analyses and more involved, traditional analyses, but the Bea and Craig [1993] system (as described in Bea [1993]) implied that simplified analysis levels within a screening system

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

should be more conservative (less likely to give "false positive" results) than their more detailed counterparts. API [1994] continues this feature. A related topic is the calibration of analysis results with real-world failure rates; industrial engineering's field of forecasting can contribute to this area. Chow [1987], Chow, Oliver and Vignaux [1990] and Oliver and Yang [1990] investigated nuclear plant failure rates, while Datta and Oliver [1992] looked at mid-air collisions.

All of the above reflects a general point of view that risk analysis is desirable. There is a large body of literature to support this view. However, it is important to remember the limitations of risk analysis when designing an information management system. Among the limitations described by Henderson [1982], Reid [1989], Rhoads [1985], Wenk [1986], and Winner [1986] were the deceptively quantitative presentation of essentially subjective data and results, and the temptation to optimize subsystems at the expense of their enfolding systems. An interesting semantic point was brought up by Goldstein [1982]: While technicians view safety as "the reciprocal of risk, . . . the probability that harm will not occur under specified conditions," the general public views safety as the condition of being "freed from harm or risk . . . secure from threat of danger, harm or loss . . . harmless." Goldstein suggested referring to a "margin of protection" rather than a "margin of safety."

This review began with the statement that information management systems and analysis techniques have been "inevitably intertwined." Yet a separation of management system

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

design from analysis design, coupled with a reliance upon databases (as in the bridge management systems) and an emphasis upon calibrating the analyses with each other and with the real world, should yield great benefits. If a particular analysis technique should prove inaccurate or unreliable, a modular information management system would allow the replacement of that particular portion of the system with an improved version. Previously stored information need not be scrapped, and structures could be reassessed by using the new analysis methodology to reinterpret the data. Staneff, Ibbs, and Bea [1994] demonstrated the ease with which this can be accomplished in a personal computing environment. Weissmann et al. [1989] was an early example of a module designed to incorporate a particular analysis technique into an existing bridge management system, which itself was built to interact with the National Bridge Inventory system.

2.2 DESIRABLE CHARACTERISTICS OF AN INFORMATION SYSTEM FOR INFRASTRUCTURE MANAGEMENT

The review of the literature of screening systems revealed many factors for consideration in the design of an improved information management system for infrastructure application:

- Storage and display of, and user interaction with, data and results.

- Ability to perform either a quick database search (if possible), or a simplified review of all structures, as dictated by available historical records and analysis method.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Ability to compare results from different structures to determine relative degree of need.

- Ability to retrieve data for subsequent reassessment upon alteration of decision criteria or updating of analysis methodology.

- Relaxation of conservatism with increasing analysis sophistication.

- Calibration of analysis results with historical record, if available.

- Modifiable decision criteria/safety standards.

- Modularity, for updating/replacing analysis procedures.

- Ability to handle multiple aspects of safety (i.e., analyses for operational error, structural integrity, inspection management, etc.), as dictated by available analyses.

- Decision support functionality without usurping the decision maker's role.

- Ability to handle biases and/or uncertainties, as dictated by available analysis methods.

- Multiple levels of failure - minor, major, collapse, etc.

- Cost/benefit economics in consequence assessment, repair alternatives.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.3 SUMMARY

In Section 2.1 of this chapter, prior infrastructure management and screening systems were reviewed. Desirable characteristics of such systems were identified and listed in Section 2.2. Chapter 3 of this dissertation develops the data structure necessary to provide those characteristics in an information management system, and describes such a system (called DAESSIM – the Decision, Analysis, and Engineering Support System for Infrastructure Management). Chapter 4 introduces sample analysis methodologies that were added to a version of DAESSIM to produce PIMS (the Platform Information Management System), and uses PIMS to demonstrate that the desired characteristics of Section 2.2 have indeed been provided. Chapter 5 then proposes an example screening methodology that can be incorporated in a version of DAESSIM to aid decision-making during the screening process. Chapter 6 describes the use of PIMS in detail, and Chapter 7 discusses possible future work.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3 DEVELOPMENT OF A DATA STRUCTURE FOR AN INFRASTRUCTURE MANAGEMENT INFORMATION SYSTEM

3.1 DESIRABLE CHARACTERISTICS OF A DATA MODEL FOR AN INFRASTRUCTURE INFORMATION MANAGEMENT SYSTEM

This research was intended to produce a system for the management of data for infrastructure systems. That system was meant to possess the characteristics developed in Section 2.2, to be adaptable to nearly any sort of engineering analysis while maintaining data integrity, and to be capable of examining not only individual structures, but large collections (fleets) of structures. Considering these requirements in view of existing database technology (Appendix A), the following desirable characteristics of a data model for the system were identified:

- Extensibility for handling unknown and unanticipated (at design time) analysis types and structure types.
- Flexibility of handling non-standard hierarchical schemes (see Section 3.2.2).
- Non-proprietary format.
- Data integrity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- *Ad hoc* query capability.

- Referential integrity.

- Interoperability with existing or external applications.

- High performance was not a consideration.

The prototype system resulting from the data model developed with the above considerations is called the Decision, Analysis, and Engineering Support System for Infrastructure Management (DAESSIM). The basic entities managed by DAESSIM are structures, evaluations, fleets, and scenarios. Structures are individual physical entities; fleets are collections of structures. Evaluations are analyses performed on structures; scenarios are analyses performed on fleets. Many types of structures, evaluations, fleets, and scenarios may be managed by the system.

Sections 3.2 and 3.3 describe alternative designs that were considered in the search for a data model for DAESSIM. For clarity, only the modeling of evaluations is examined in these sections. Section 3.4 then discusses the complete data structure of DAESSIM.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.2 APPLYING A TRADITIONAL MODEL TO THE DAESSIM REQUIREMENTS: MANAGING SETS OF FACTORS

Database design is more of an art than a science, but a traditional relational or object-oriented design philosophy might lead to one of the following example table structures. For relational designs, this discussion concentrates on inputs to evaluations; output storage would be modeled similarly.

In the figures in this chapter, the ' - ' in a cell indicates a null value: the complete absence of information. An underlined attribute in an E-R diagram or a table indicates a primary key. If an entity or table must be stated in order to properly reference an attribute or field, the reference will be of the form *ENTITYNAME.AttributeName*. Terminology, entity-relationship diagrams (e.g., Figure 3-1), object representations (e.g., Figure 3-2), and table representations (e.g., Figure 3-3) are discussed in Appendix A ("Review of Database Technology").

3.2.1 Flat Files

In Figure 3-1, an EVALUATION entity tracks the evaluations' IDs as well as their types and uses Eval_ID as its primary key. All evaluation data are stored in the EVALUATION entity. Whenever an additional factor for a new evaluation type is required, a new attribute is added to EVALUATION (see field F4).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

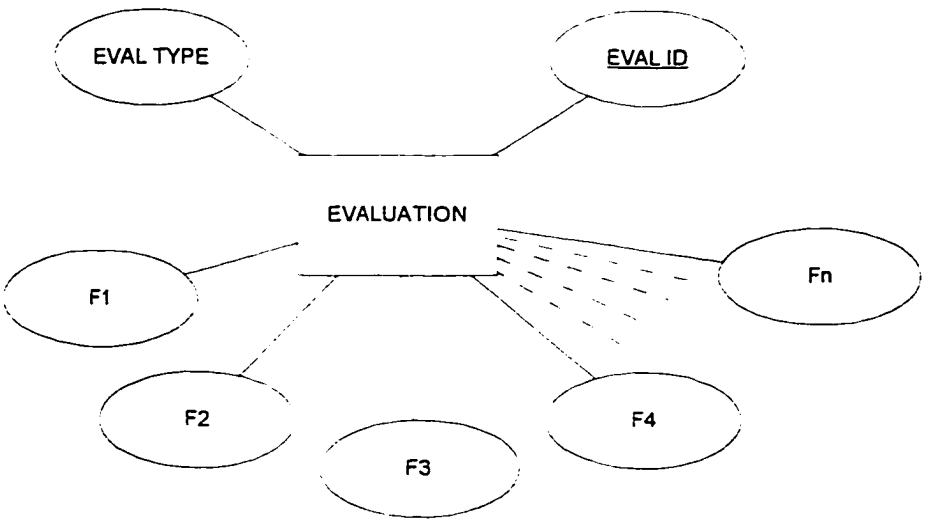


Figure 3-1: Flat-File Entity Relationship Model

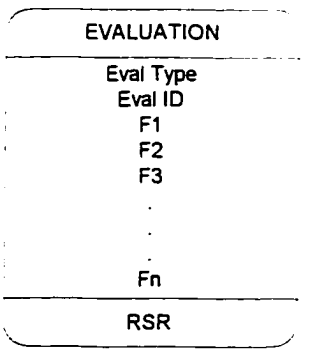


Figure 3-2: Flat-File Object Analog

EVALUATION

EvalType	*Eval ID*	*F1*	*F2*	*F3*	*F4*	*F5*	*F6*
L1RSR50	1	1.1	1.2	1.3	-	1.4	1.5
L1RSR50	2	1.2	1.5	1.8	-	1.1	1.3
L1RSR60	3	1.1	1.2	1.3	1.9	1.4	1.5

Figure 3-3: Table Implementation of E-R Model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Advantages:

- This format allows the quick creation of a simple system for elementary data storage – for example, spreadsheet modeling.

- No special user input or programming is required to determine which table or object to use.

- Behavior of the system is "hard-wired."

Disadvantages:

- Rows for evaluations for which a particular factor is irrelevant (eval type L1RSR50 in Figure 3-3, for example, doesn't require factor F4) will contain null values in the corresponding fields. To the writer of an ad hoc query, looking at the EVALUATION table out of context, these nulls may mean 'missing information' rather than the correct 'not applicable' - a controversial source of error.

- The designer of a new evaluation must add attributes to EVALUATION to meet its needs - and risks damaging existing data.

- EVALUATION can grow to thousands of fields wide (and many, many nulls). For example, if 300 runs of an L1RSR50 evaluation were stored at 5 inputs each, and 1 run of an L2RSR40 evaluation happened to involve 1500 completely different input

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

factors, then EVALUATION would be 1506 fields wide, 301 records deep, and contain 450,005 nulls. However, while most RDBMS's place no upper limit on the number of records, they do restrict the number of fields (Microsoft Access, for example, restricts tables to 256 fields).

3.2.2 Hierarchical Flat Files

Engineering data often are sorted into an hierarchical format. For example, a steel-jacketed oil platform has bays, each bay has braces, and each brace has a length. Figures 3-4 through 3-6 illustrate the approach of just adding a new field for each new factor.

For example, the platform Level One Reserve Strength Ratio evaluation of Bea and Craig [1993], herein known as evaluation type L1RSR1, is an evaluation with a single level of hierarchy: it requires nine input factors that pertain to the entire offshore platform and produces one output factor that also pertains to the entire platform. If, however, it were modified such that it required the nine factors to be input for each leg of the structure, it could be either a one-level or a two-level evaluation. DAESSIM could then require 36 individual factors to be defined, or it could require ten factors to be defined at a base level and another factor (called a 'factor modifier' to describe its relationship to the base factors) to be defined at a higher level. The ten base-level factors would consist of the original nine plus one to specify how many legs are in the platform; the factor modifier's values would identify each leg (from one to the number of legs). To allow either schema to be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

chosen by the DAESSIM user (when a new evaluation is added), two additional attributes are required for each additional level of hierarchy in the structure of EVALUATION.

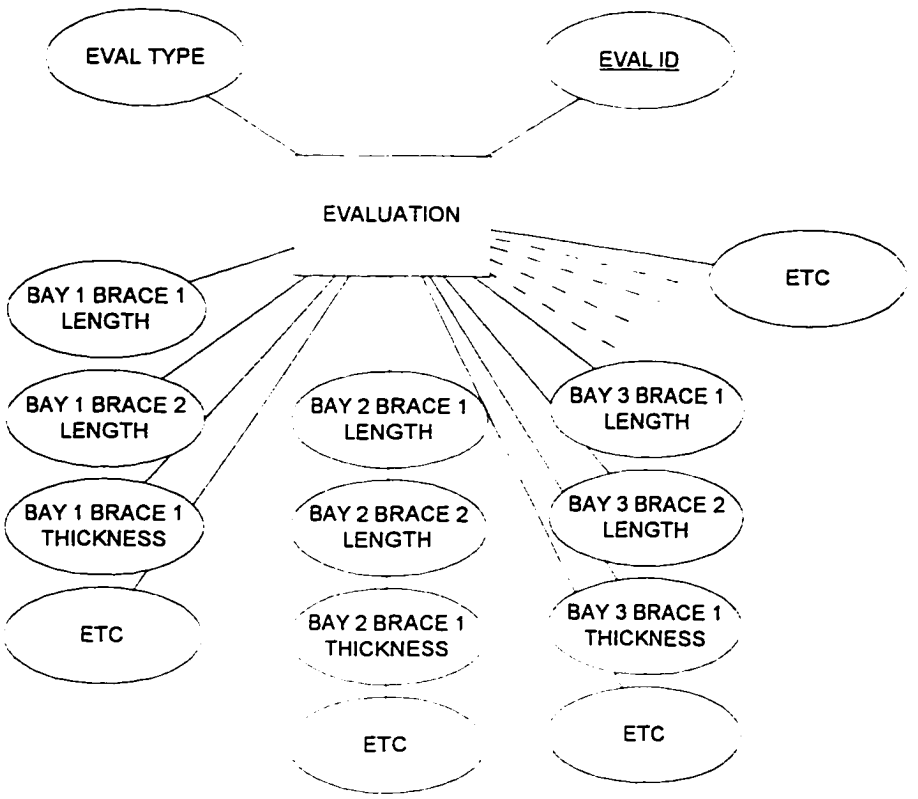


Figure 3-4: Hierarchical Flat-File Entity-Relationship Model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

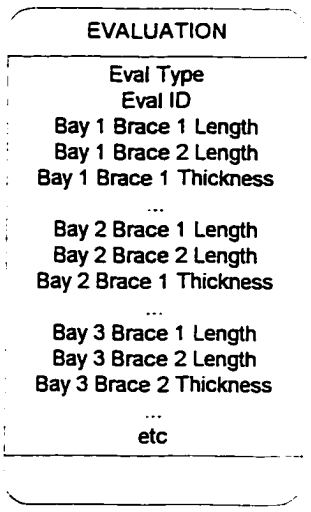


Figure 3-5: Hierarchical Flat-File Object Analog

EVALUATION

EvalType	*Eval ID*	*Bay 1Brace 1Length*	*Bay 1Brace 2Length*	*Bay 1Brace 1Thickness*	*Bay 1Brace 2Thickness*
L1RSR50	1	12	-	1.5	-
L1RSR50	2	12	-	1.5	-
L1RSR60	3	15	20	1.825	1.825

Figure 3-6: Hierarchical Flat File Table Implementation of E-R Model

Advantages:

- Allows the quick creation of a simple system for somewhat complex data storage.
- No special user input or programming is required to determine which table or object to use.
- Behavior of the system is "hard-wired."

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Simple queries.

Disadvantages:

- EVALUATION can become very wide indeed - especially if it is a consolidated data table as in Figure 3-3, and two complex evaluation types use two different schemes of identifying data. For example, type L2RSR40 may use a platform:bay:brace:property scheme, while type L2RSR41 may use a platform:member_#:property scheme. Fields would be required for all factors.

- The designer must know in advance the maximum number of items at each level of the hierarchy. Such foreknowledge is contrary to the assumptions of this research. For example, a table with Bay_i_Brace_j_Length fields, for i from 1 to 10 and j from 1 to 20, cannot accommodate a platform with eleven bays or a bay with twenty-one braces.

3.2.3 Traditional Evaluation Sub-Class Structure

In Figures 3-7 through 3-9, EVALUATION again tracks evaluation IDs and types. The two evaluation-specific entities, L1RSR50 and L1RSR60, track the values of input factors for each of two types of evaluations. Both L1RSR50 and L1RSR60 use Eval ID as a primary key, as do any other evaluation-specific entities. These keys are linked to EVALUATION.Eval_ID in a one-to-one relationship, and the choice of entity to use for a particular evaluation is determined by the entry in EVALUATION.EvalType.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 3-7 introduces the IS-A relationship that exists between a parent class and its child classes (subclasses). The arc in Figure 3-7 between EVALUATION and its subclasses indicates an exclusive relationship (an evaluation may be either an L1RSR50 or an L1RSR60 evaluation type or another type, but it may not be more than one type). The shaded box in Figure 3-7 indicates extensibility - other, possibly unpredictable, attributes are part of this relationship. In Figure 3-9, a new relationship notation for tables is introduced - "1" for one and "∞" for many.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

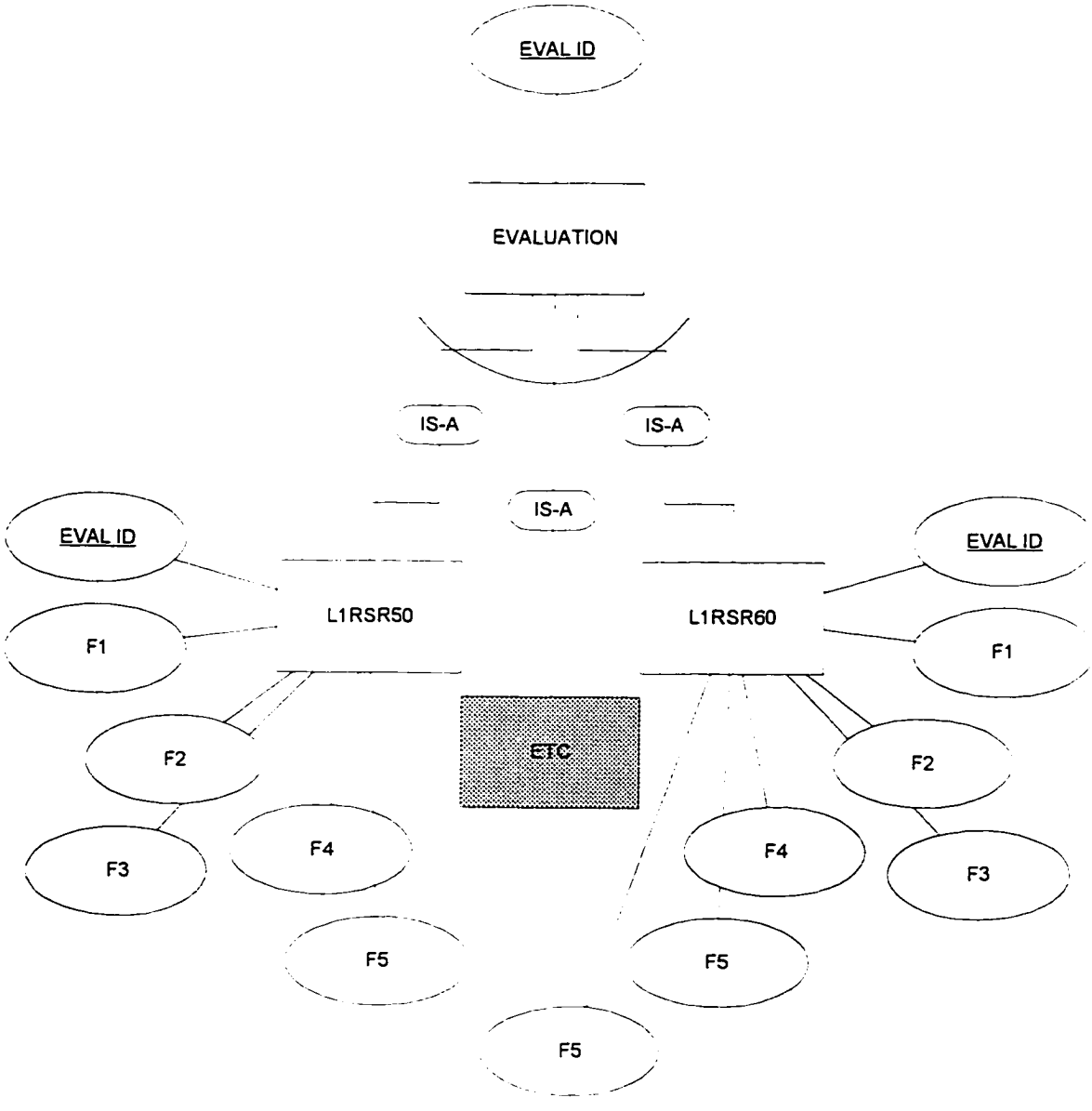


Figure 3-7: Traditional Sub-Class Entity-Relationship Model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

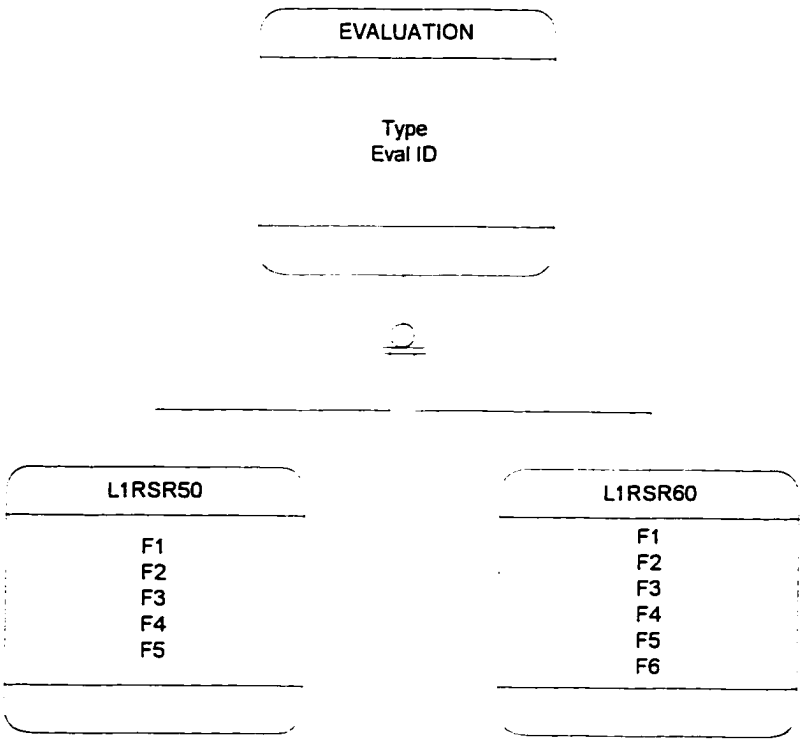


Figure 3-8: Traditional Sub-Class Object Model

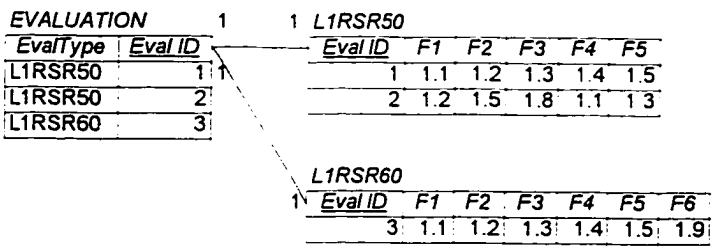


Figure 3-9: Traditional Sub-Class Relational Implementation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Advantages:

- This is an efficient storage and retrieval scheme, if all evaluation types are known and included at design time. However, that is contrary to the assumptions of this research.

- In a relational implementation, this format includes all the mathematical rigor of the relational model. This is a very desirable feature.

Disadvantages:

- The choice of which data table to use, based on field EVALUATION.EvalType, must be determined by the user (the most common method) or by code. This is a minor disadvantage in this research.

- If an L1RSR60 evaluation (#3) is performed on the same platform as an L1RSR50 evaluation (#1), then the data for the latter must be re-entered into the table for the former (assuming one-to-one correspondences between the attributes F1-F5 in the two evaluation types). This is a serious violation of data integrity principles.

- If another type of evaluation is added to the system (say, L2RSR40), another data table (tblL2RSR40) must be added and relationships established. This requires the designer of the new evaluation to be fluent in database design as well as analysis design.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Adding an output - for example, an intermediate result - to a table of outputs requires changing the table's structure.

The latter three are significant shortcomings in the context of this research.

3.2.4 Traditional Hierarchical Evaluation Sub-Classes

Figures 3-10 through 3-12 illustrate a structure that mirrors the hierarchical structure of the data. Here BAY uses Eval_ID (in a many-to-one relationship with EVALUATION.Eval_ID) and Bay_ID for its primary key, while BRACE uses Eval_ID, Bay_ID (together, in a many-to-one relationship with BAY.Eval_ID and BAY.Bay_ID), and Brace_ID as its primary key.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

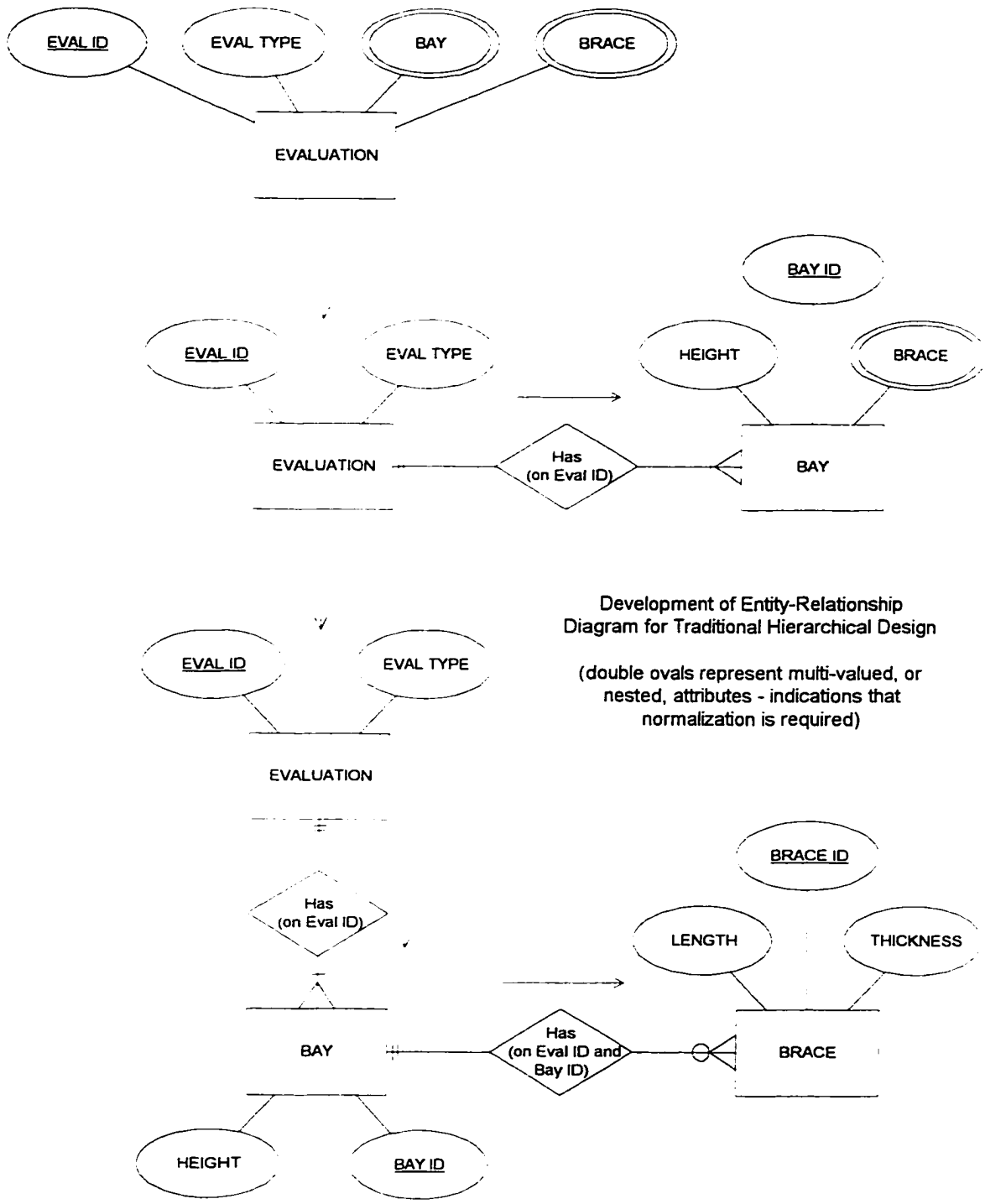


Figure 3-10: Traditional Hierarchical Entity-Relationship Model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

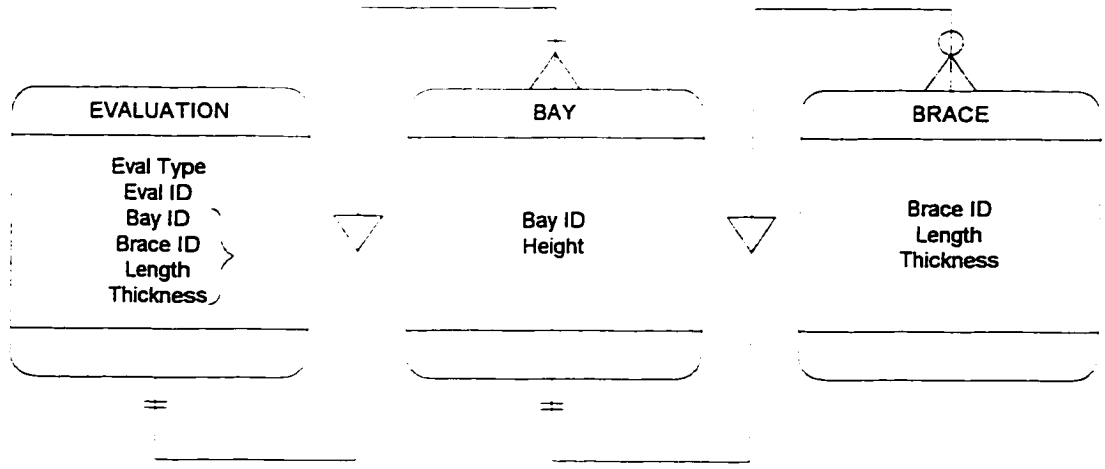


Figure 3-11 Traditional Hierarchical Object Model

EVALUATION

EvalType	*Eval_ID*
L1RSR50	1
L1RSR50	2
L1RSR60	3

BAY

Eval_ID	*Bay_ID*	*Height*
1	1	12
1	2	12
2	1	20

BRACE

Eval_ID	*Bay_ID*	*Brace_ID*	*Length*	*Thickness*
1	1	1	1	1
1	1	2	2	2
1	1	3	3	3
1	2	1	1	1
1	2	2	2	2
2	1	1	1	1
2	1	2	2	2
2	1	3	3	3

Figure 3-12: Traditional Hierarchical Relational Implementation of E-R Model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Advantages:

- There is no limit to the maximum number of items at each hierarchy level – a significant advantage.

- This format is clearly understood and used by many information management professionals. This is of marginal importance to the present research.

- If all of the types of entities, evaluation types, and attributes are known at system design time, this format is very efficient. This is of marginal importance to the present research.

- Again, in a relational implementation, this format includes all of the mathematical rigor of the relational model – a desirable feature.

Disadvantages:

- With each new level of hierarchy, the queries that must be written to store or retrieve data become more complex. This is of marginal importance to this research.

- Inflexibility. Each time a new type of evaluation is added, the appropriate table structure must be added as well. If there are many levels of hierarchy - such as plat-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

form:orientation:bay:brace:orientation:location:thickness - this would be a formidable task. This is a significant disadvantage for this research.

- 'Data mining' and *ad hoc* querying in general become difficult – a serious issue. For example, if it is desired to determine how many platforms use steel of 3" thickness, where is that information stored for each type of evaluation?

3.3 A NON-TRADITIONAL APPROACH TO DATABASE DESIGN: MANAGING FACTORS RATHER THAN SETS OF FACTORS

Flexibility and extensibility have been the stumbling points of the designs presented so far. In Section 3.3 a non-traditional approach will be presented that captures the flexibility and extensibility of an object-oriented system while retaining the data management capabilities of a relational system. This approach will be called *relational object* technology.

3.3.1 Preliminary Factor Name / Factor Value Design

In Figures 3-13 and 3-14, EVALUATION is retained and a look-up entity (FACTOR_ALLOWED) is added to control the types of factors allowed in the database. Entity FACTOR_DATA (represented in Figure 3-13 as many-to-many relationship #1) serves as the storehouse for evaluation data. Attributes Eval_ID and Factor_ID are its combination primary key, and are in many-to-one relationships with attributes EVALUATION.Eval_ID and FACTOR_ALLOWED.Factor_ID. This structure allows only a single level of hierar-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

chy. (Figure 3-13 introduces a notation for tracking which fields are involved in a many-to-many relationship. For example, relationship #2 involves Eval_Type and Factor_ID.)

In Figure 3-13, relationship #1 joins EVALUATION.Eval_ID and Factor_ID from many-to-many relationship #2. Since Factor_ID in relationship #2 is by definition not unique, and since all foreign keys must be unique, relationship #1 would require a view of relationship #2 that only contains unique occurrences of attribute Factor_ID. This isn't possible at the table level, so application code is required to execute a query at a higher level.

As implemented (Figure 3-14), when new evaluation types are defined in EVAL_TYPE, the required new factors for these types are added or verified as row entries to FACTOR_ALLOWED. They are also assigned to particular EVAL_TYPE instances in table EVAL_TYPE_FACTOR (the implementation of relationship #2). A new instance of the new type is recorded in EVALUATION and the value of each factor for each new evaluation instance is stored in FACTOR_DATA (the implementation of relationship #1). Application code, implemented as object class EVAL_TYPE_CODE, controls the sequence of operations, restricts the allowable inputs (as defined by the entries in EVAL_TYPE_FACTOR), and stores them properly. Sub-classes (not shown) of object EVAL_TYPE_CODE correspond to each instance of entity EVAL_TYPE. The combinations of Eval_ID's, Factor_ID's, and Factor_Values constitute the attributes of the EVAL_ TYPE_CODE objects.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The relational portions of the data structures in this section are referred to as the *sub-structures*. The object-oriented application code aspects are the *superstructures*.

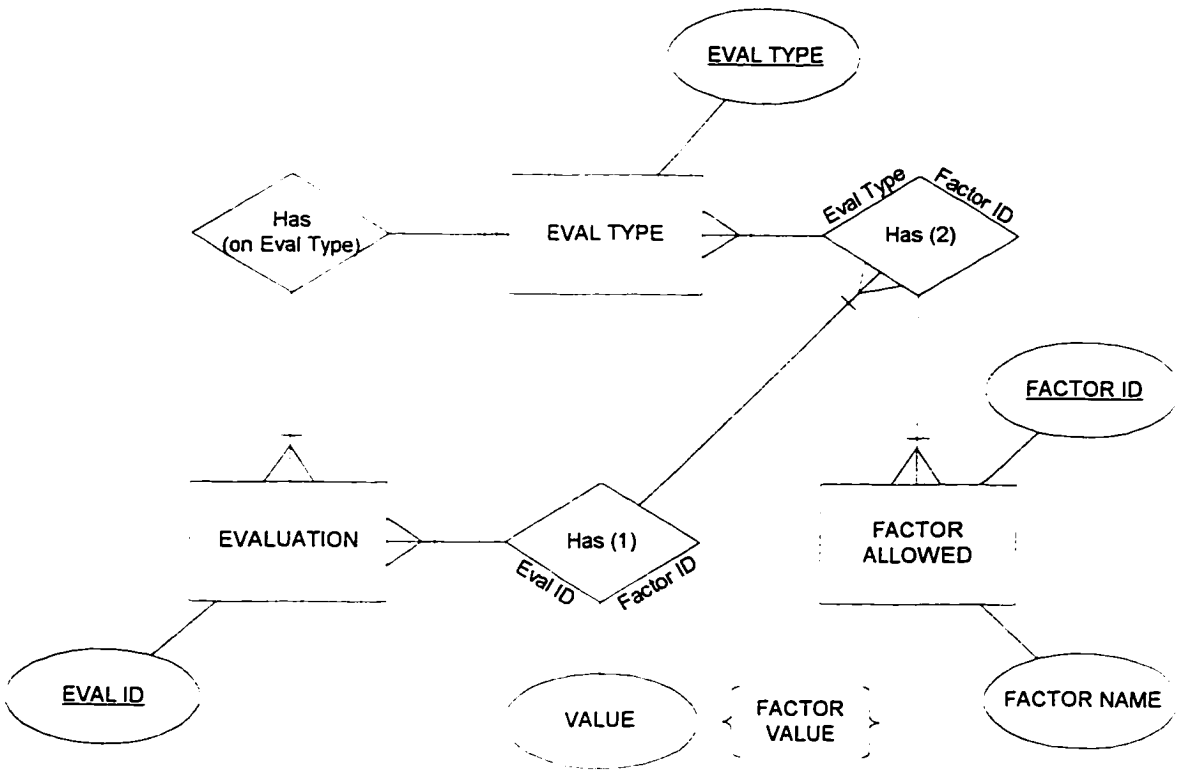


Figure 3-13: Preliminary Factor Name / Factor Value Entity-Relationship Model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

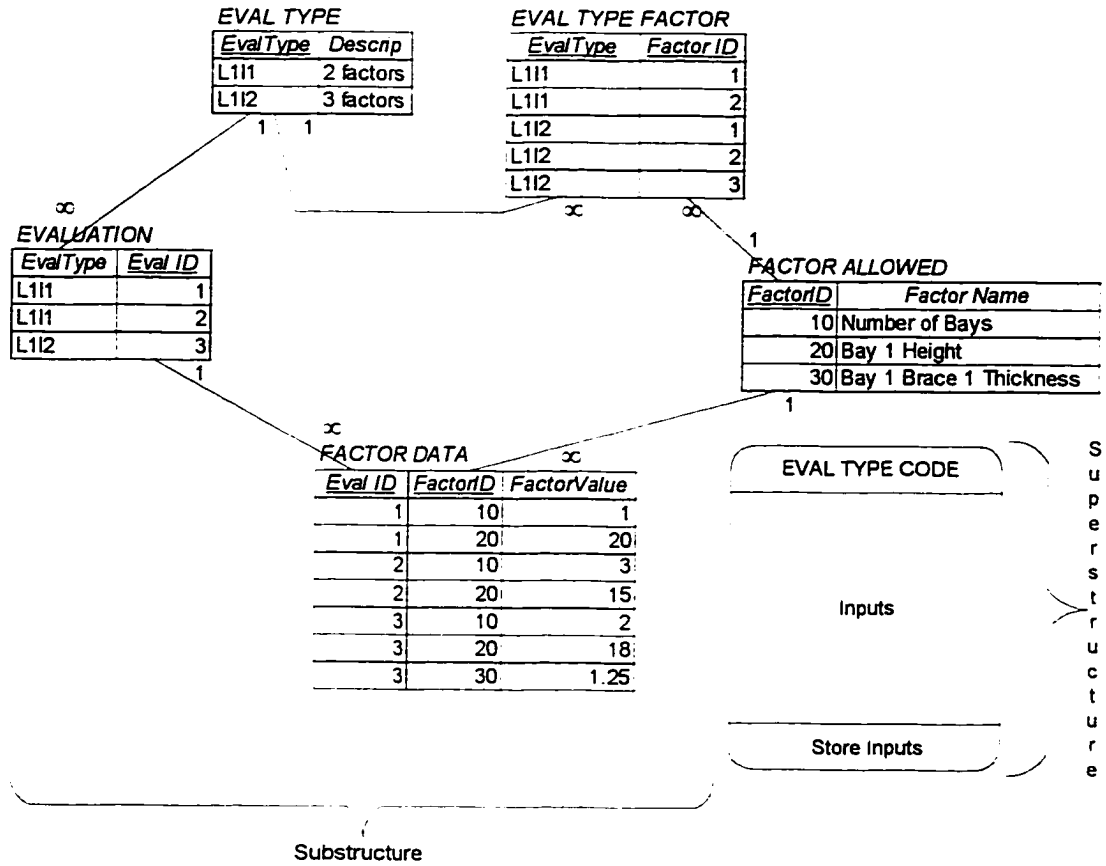


Figure 3-14: Preliminary Factor Name / Factor Value Relational Implementation

Advantages:

- Flexibility - new types of evaluations can be added without prior knowledge of their requirements. This is an important advantage for the present research.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Unencapsulated data - unlike a purely object-oriented structure, the data are available to the knowledgeable user, making *ad hoc* queries possible. An example of an *ad hoc* query would be to find the average height of the first bay of all platforms. With exposed data, such a query would not need to be a built-in method of the EVAL_TYPE_CODE class or its sub-classes. This is a significant factor.

- Open architecture (another consequence of unencapsulated data) - should a new evaluation be added that can use inputs from other evaluations, the information can be readily retrieved. Such data sharing is important in the management of infrastructure.

Disadvantages:

- Because evaluation data are stored and retrieved one factor at a time, rather than in sets of factors, many more queries must be performed. This requires complex coding, and high numbers of disk accesses that slow performance. This is a marginal issue for this research.

- There are no provisions for hierarchical data – a significant disadvantage.

- In the absence of a customizable data type for FACTOR_DATA.FactorValue, all factor values must be of the same data type – another significant disadvantage. For example, if FactorValue were a double-precision numerical field, logical ("yes/no" or "0/1") and integer data (counts of bays, for example) would be difficult to manage.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Separate copies of the database fall out of sync quickly as new factors are added to FACTOR_ALLOWED (a unique ID is assigned for each new factor; if two identical factors are added to two identical, separate copies of the database, but in reverse order, the two factors will have different Factor_ID assignments). Therefore, databases of this structure must be run from a central server. Though seemingly a major disadvantage, this is a common situation among large information systems.

3.3.2 Hierarchical Factor Name / Factor Value Design

To gain hierarchical capabilities, the attribute FM_ID is added to relationship #1 (implemented as table FACTOR_DATA). This attribute references a new entity, FACTOR_MODIFIER (Figure 3-15). The FM_ID field combines with fields Eval_ID and Factor_ID to form FACTOR_DATA's primary key. All the fields in FACTOR_MODIFIER (except FM_ID) are combined in a unique index. Attribute FACTOR_MODIFIER.FM_ID is itself a unique, auto-incrementing field. FACTOR_MODIFIER's FM# fields each denote a level of hierarchy, and each is in a many-to-one relationship with FACTOR_ALLOWED.Factor_ID; the FM#Value fields each denote a value at that particular level.

For example, to store the value 1.25 for the thickness of brace #100 of bay #3, one would first verify that a FACTOR_MODIFIER entry existed where FM1=20, FM1Value=100, FM2=10, FM2Value=3, and all the other FMx's in that record were defaults. This is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

shown in Figure 3-16 for FACTOR_MODIFIER.FM_ID=5. Then, an entry for a thickness (Factor_ID=50) of 1.25 and FM_ID=5 is made in FACTOR_DATA (the seventh row of FACTOR_DATA in Figure 3-16). The first entry in FACTOR_DATA means that bay #3 is 15.8' high, according to evaluation #1; the second entry shows that bay #3 is 16.1' high, according to evaluation #2. The third instance of FACTOR_DATA indicates that bay #4 is 15.75' high according to evaluation #3 (which is of type L1I2, while the other two were of type L1I1); the fourth and fifth instances mean that brace #51 of bay #2 is forty feet long and 1.5" thick. Finally, the sixth and seventh instances indicate that brace #100 of bay #3 is 15 feet long and 1.25" thick.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

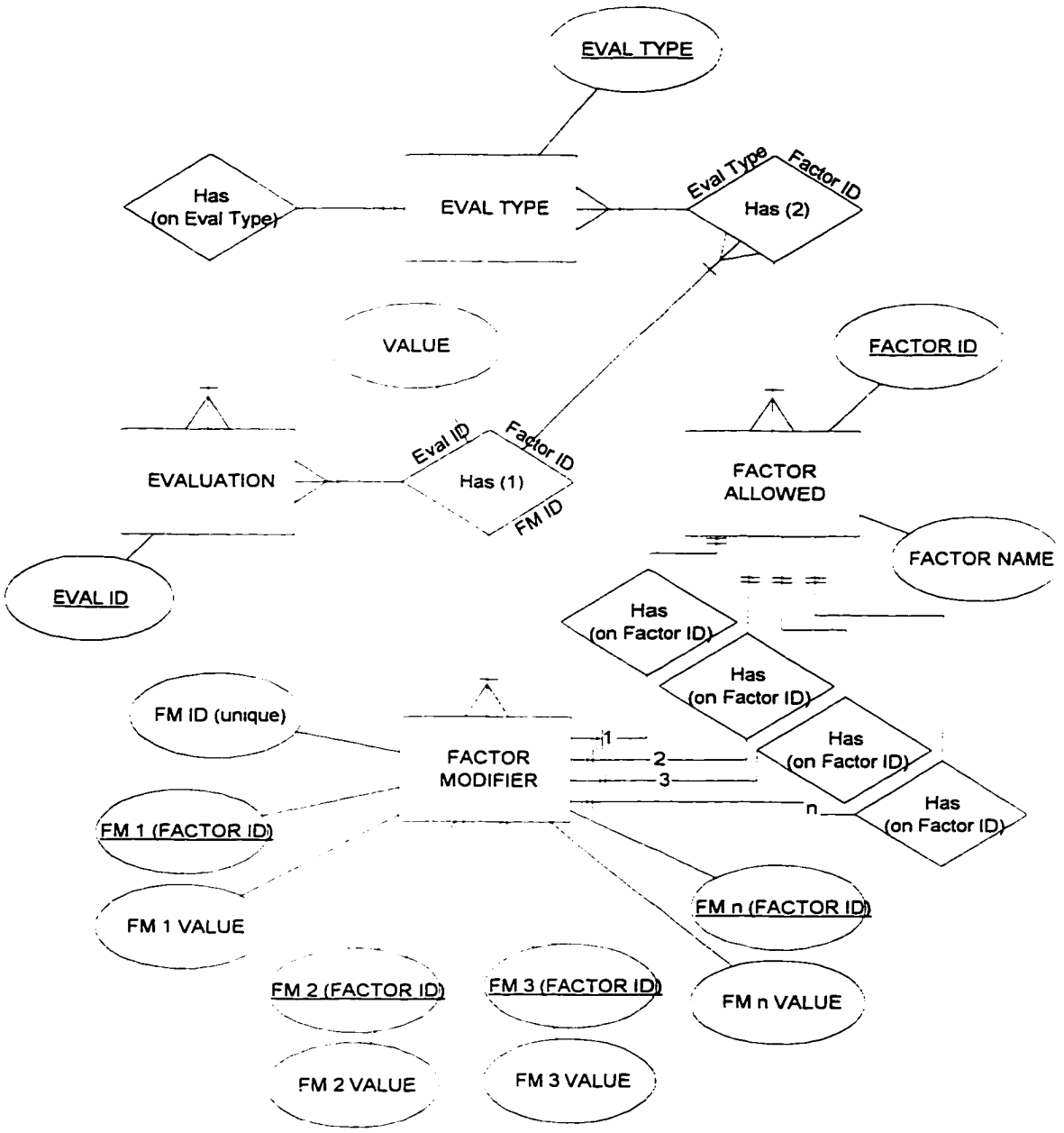


Figure 3-15: Hierarchical Factor Name / Factor Value Entity-Relationship Model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

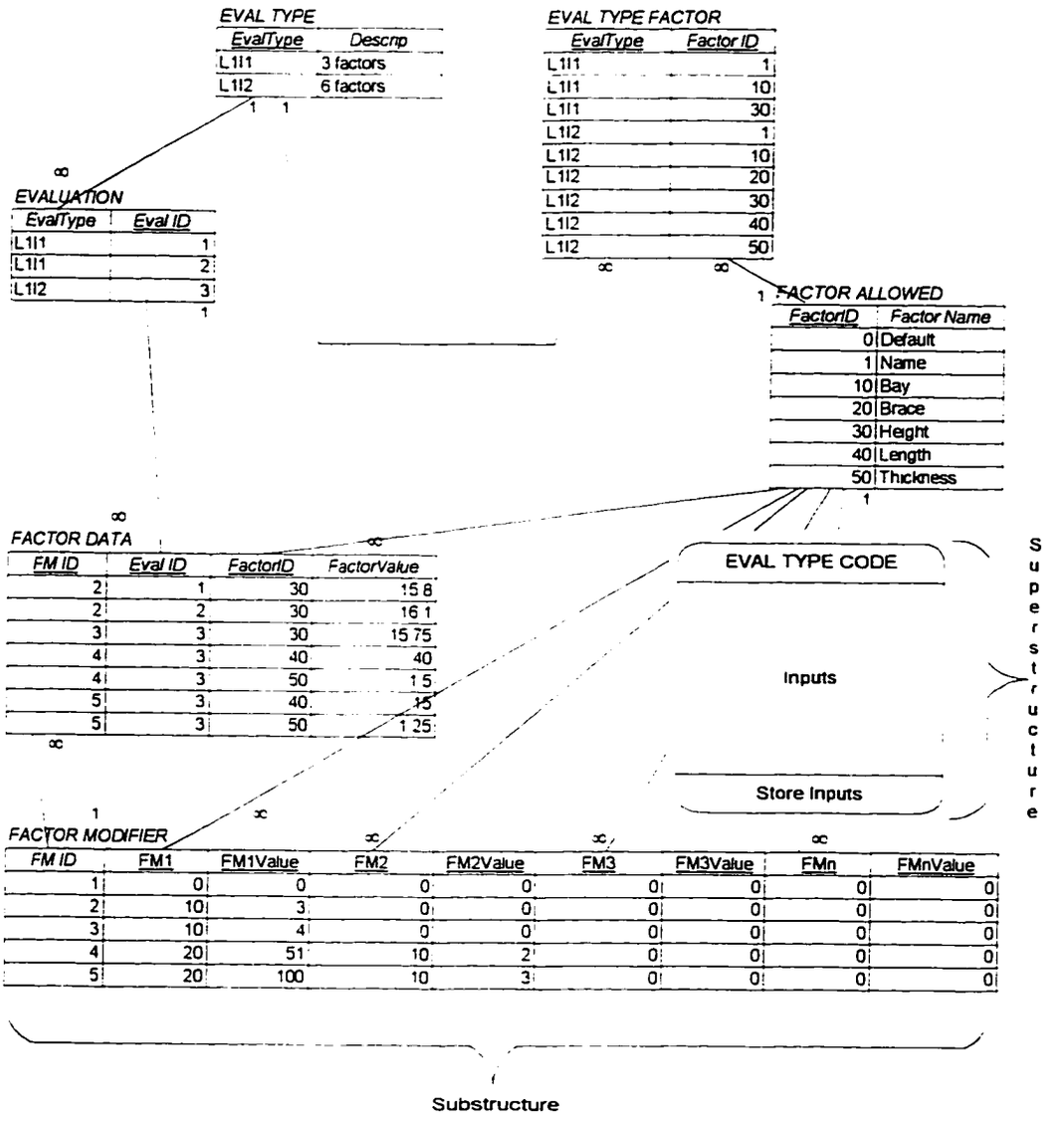


Figure 3-16: Hierarchical Factor Name / Factor Value Relational Implementation

Advantages (in addition to those listed in Section 3.3.1):

- The number of items within a level of hierarchy is unlimited, a significant feature.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- The number of factors that need to be defined is reduced. For example, instead of defining Bay 1 Height, Bay 2 Height, Bay 3 Height, etc., all that is required is to define the factors Bay and Height. This is a significant advantage.

- Hierarchy "chains" are flexible. For example, the "chain" [thickness → vertical brace → orientation → bay] is possible in the same table as the chain [thickness → diagonal brace → top/bottom → bay]. This is another significant advantage.

Disadvantages (in addition to those listed in Section 3.3.1):

- Given current database technology, all factor values must be of the same data type – a major disadvantage.

- The number of hierarchy levels (fields tblContainers.FM1 to tblContainers.FMn in Figure 3-16) is limited by the number of fields permissible within an index. This is a restriction of the individual database software used to implement the structure, and is a marginal disadvantage in the context of the present research.

3.3.3 Modified Hierarchical Factor Name / Factor Value Design

The modifications to FACTOR_DATA and FACTOR_ALLOWED shown in Figure 3-17 illustrate a solution to the problem of storing factors of differing data types. This is the solution adopted by DAESSIM. The former field FACTOR_DATA.FactorValue is re-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

placed with a series of fields accommodating every possible data type that the database management software allows. When values are read into or out of FACTOR_DATA, the value of FACTOR_ALLOWED.FieldName for that FactorID is used to determine which FACTOR_DATA field to use. In the example, the values for the height and thickness factors are stored in FACTOR_DATA.fldDouble, the values for length are stored in FACTOR_DATA.fldInteger, and names are stored in FACTOR_DATA.fldText. All FM# and FM#Value fields in FACTOR_MODIFIER are integer fields. (The oval enclosed in a rectangle in Figure 3-17 indicates that the Value attribute has been decomposed into sub-attributes.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

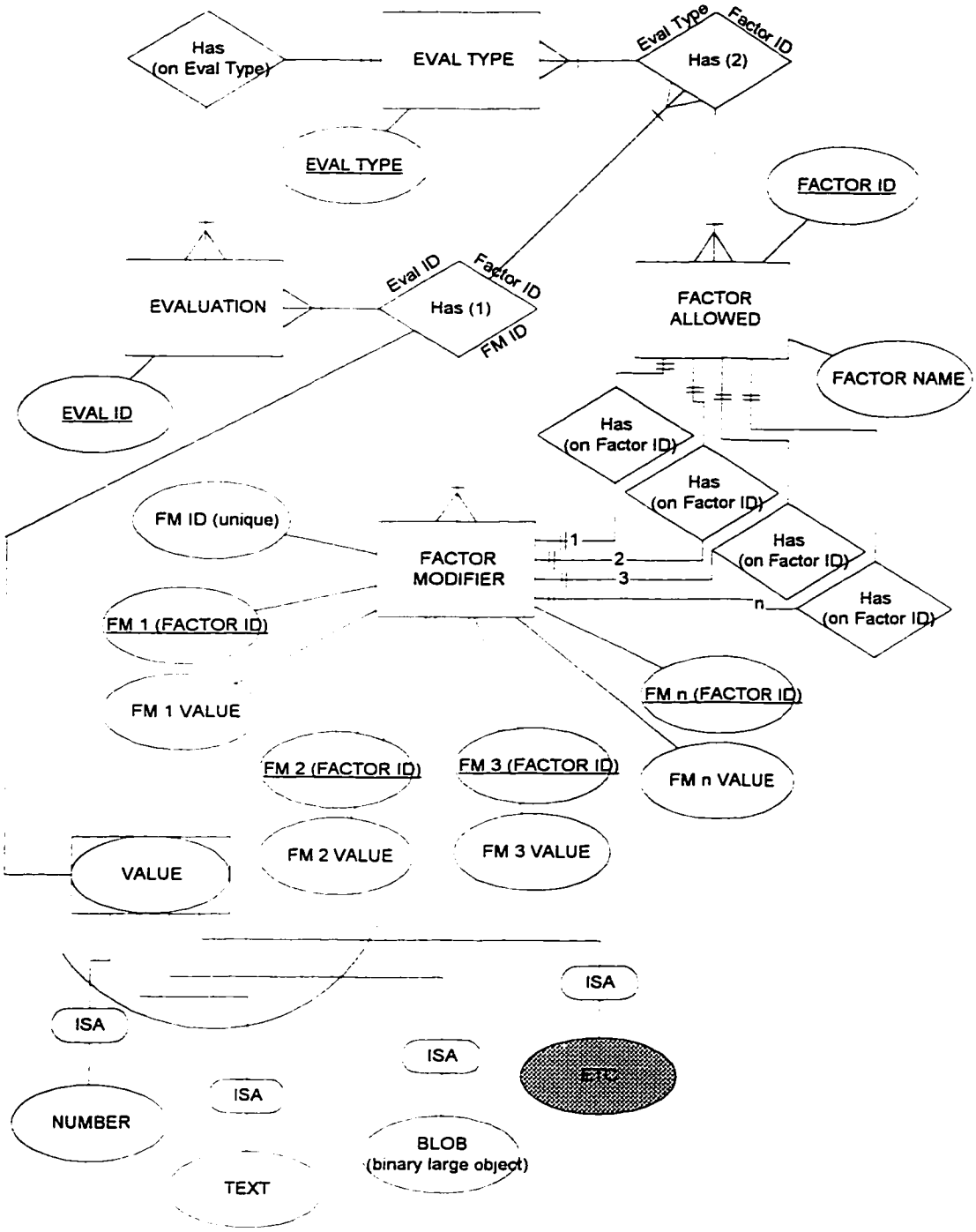


Figure 3-17: Modified Hierarchical Factor Name / Factor Value Entity-Relationship Model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

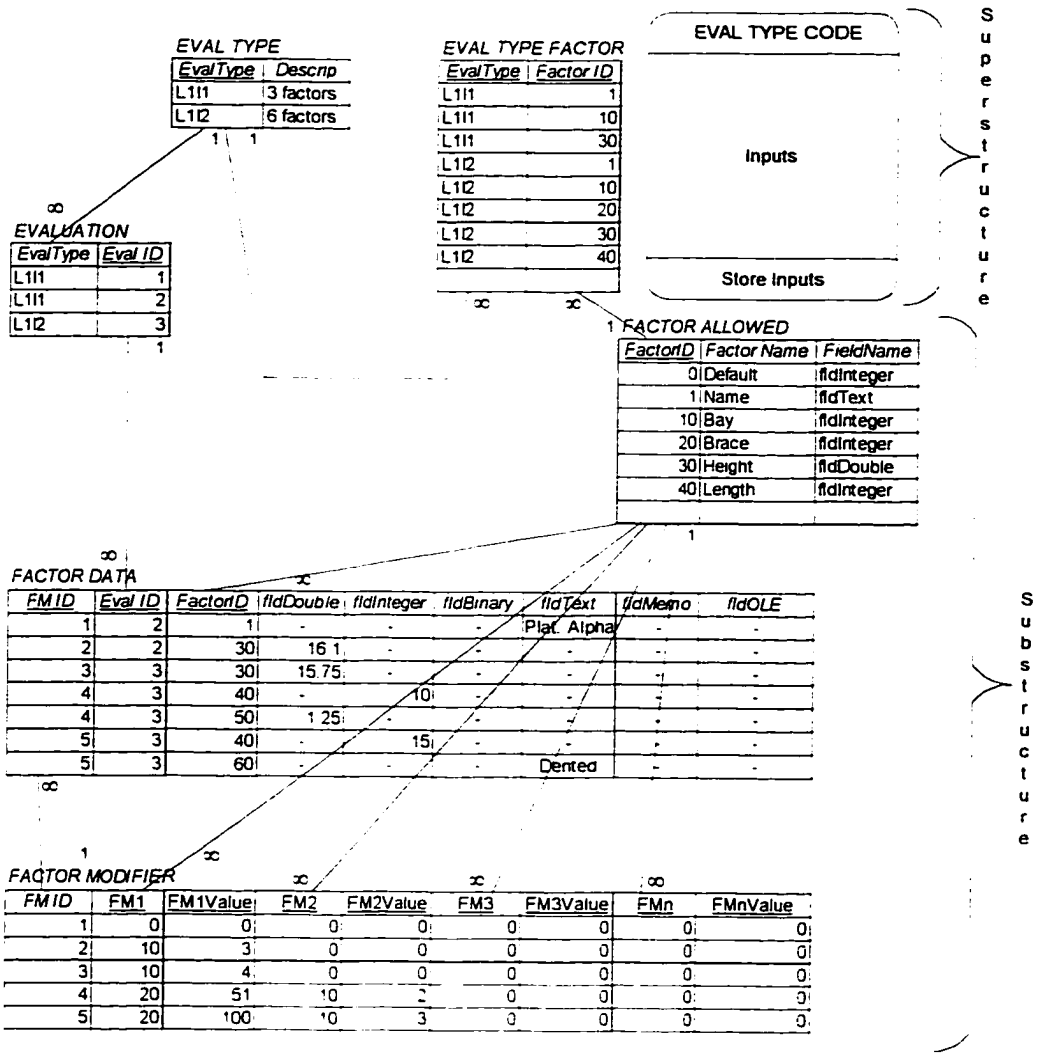


Figure 3-18: Modified Hierarchical Factor Name / Factor Value Relational Implementation

Advantages (compared with those of Section 3.3.2):

- Factors having values of any conventional data type may now be entered. In DAESSIM, eleven are allowable: Currency, Date/Time, Memo, Byte, Integer, Long Integer,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Single Precision, Double Precision, OLE (BLOB – Binary Large Object), Text, and Yes/No data types. For example, strings such as "Risk is Acceptable" are now accommodated. This is a significant advantage.

Disadvantages:

- FACTOR_MODIFIER includes many nulls and/or otherwise irrelevant values (data entered in incorrect fields will not exist from DAESSIM's point of view). This is a marginal disadvantage to the present research.

- The number of hierarchy levels is still limited – also a marginal disadvantage to the present research.

Figure 3-19 shows a different method of accommodating varying data types. Figure 3-18's FACTOR_DATA is split into many component tables - FACTOR_DOUBLE, FACTOR_INTEGER, FACTOR_MEMO, etc., each of which retains the basic structure and relationships of FACTOR_DATA but replaces field fldValue with a type-specific value field.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

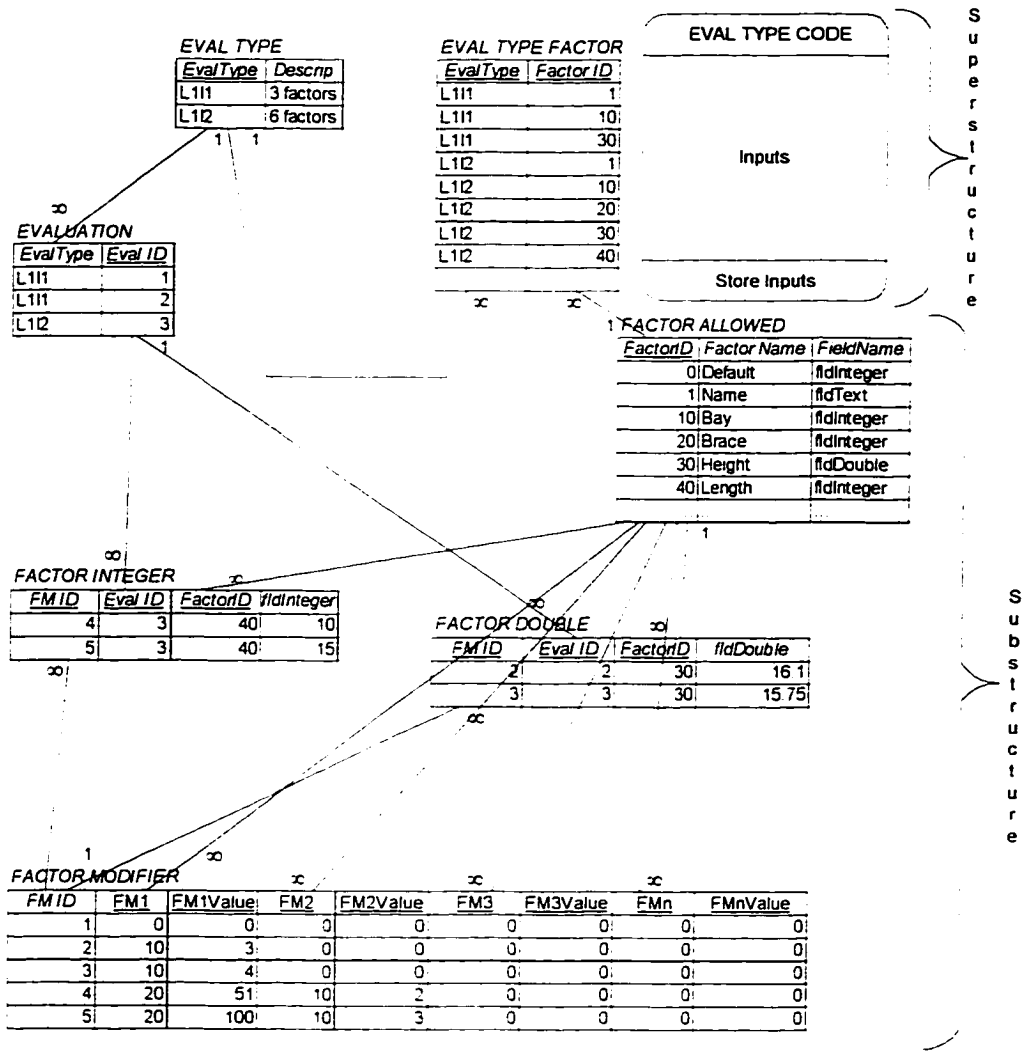


Figure 3-19: Alternative Modified Hierarchical Factor Name / Factor Value Relational Implementation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Advantages:

- The segregated data tables will contain far fewer (if any) null values, resulting in more efficient disk usage. Since optimal speed performance is not a goal of this research, disk usage is of minor importance.

- Backup procedures should be easier for the segregated tables than for the consolidated FACTOR_DATA – again, an advantage of minor importance.

Disadvantages (compared with Figure 3-18):

- FACTOR_DATA's primary key can no longer be used to prevent the duplication of data - a value for FM 1, Eval_ID 1, Factor_ID 10 can be entered in both FACTOR_DOUBLE and FACTOR_INTEGER, for example. However, only the value that's in the correct table will be known to DAESSIM. This is a moderately significant disadvantage.

3.4 OTHER FEATURES OF THE AS-BUILT DAESSIM DATA STRUCTURE

Throughout this section, please refer to Appendix B (DAESSIM Entity-Relationship Diagram) and Appendix C (DAESSIM Table Definitions).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.4.1 Evaluation Objects

As discussed in Section 3.3, the relational portion of the DAESSIM data structure stores individual factors while code objects manage sets of factors. These code objects are organized into *functions* and *evaluations*. Functions in DAESSIM handle routine operations such as mathematical calculations and standardized queries, and are unseen by the user.

There are two primary classes of evaluations (Figure 3-20): output objects, representing the results of analyses performed on structure objects (e.g., a structural analysis of a platform); and input objects, representing transient data that refer to structure objects and/or to output objects. There are also three classes of input objects: structure input objects, containing semi-permanent data pertaining to structures; inspection objects, containing data required to perform a specific output analysis; and engineering assumption objects, containing assumptions required to perform a specific output analysis.

The division between inspection and engineering assumption objects is indistinct and subject to the whims of the object types' designer. A good rule is that inspections contain hard data while engineering assumptions contain variables or options in method. For example, the original ULSLEA program [Mortazavi 1995] contained an option to automatically calculate certain structural information rather than to manually input it. In the sample PIMS application (an instance of DAESSIM, discussed in Chapter 4), this option is embodied in an engineering assumption object.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

EVALUATION OBJECT(S)
- OUTPUT OBJECT(S)
  - *L1RSR1 (Section 4.2.1.2)*
  - *L1C1 (Section 4.2.1.3)*
  - *L1Risk1 (Section 4.2.1.4)*
  - *L1Risk_Fleet1 (Section 4.2.1.5)*
  - *L2VMax1 (Section 4.2.2.3)*
  - *L2RSRx (Section 4.2.2.5)*
- INPUT OBJECTS(S)
  - STRUCTURE INPUT OBJECT(S)
    - *PlatformData (Section 4.2.2.1)*
  - INSPECTION OBJECT(S)
    - *L1I1 (Section 4.2.1.1)*
    - *L2I1 (Section 4.2.2.2)*
  - ENGINEERING ASSUMPTION OBJECTS(S)
    - *L2Eng1 (Section 4.2.2.4)*

Figure 3-20: Evaluation Object Hierarchy (Sample Evaluation Objects Included in PIMS are Italicized)

Sub-classes of each class of input and output evaluation object are called *evaluation types* in DAESSIM. Evaluation types may be added by the user, and serve as templates for evaluation instances. For example, in Figure 3-18, if a user wished to add an L2I1 evaluation to DAESSIM, she would select the L2I1 evaluation type. This act of selection would cause a new entry into EVALUATION, and initiate the gathering of data from the user (assuming that the L2I1 evaluation type is an inspection type). Thus a new instance of L2I1 would be created, with the unique ID assigned by EVALUATION.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Any of the input or output object instances may serve as inputs to any other object instances, provided they pertain to the same structure; e.g., a structural analysis on platform A cannot use inspection data from platform B. This is enforced by DAESSIM's relational substructure. Thus, evaluation objects possess the closure property.

3.4.2 Handling of Outputs

Section 3.3.3 discussed the way DAESSIM handles input data. Outputs are handled in nearly the same way. The differences from the system's point of view are that: a) inputs are either user-generated or obtained from stored values, while outputs are generated by evaluation objects; and b) factors are defined as either inputs or outputs in the Factor_I/O field of the EVALUATION_TYPE_FACTOR table. This definition allows objects to pull only inputs or only outputs from the system without knowing the specific ID's of each factor.

In a sense, storing outputs violates data integrity - a stored output may not necessarily match the value that could be recalculated by its associated object. Avoiding output storage would negate this problem, but would greatly increase system access times. For example, if a object that calculates overall fleet risk required as inputs the risk assessments of each structure in a fleet, then a system that does not store outputs would need to recalculate each structure's risk each time a user asked for the fleet's risk assessment. Because DAESSIM stores outputs, however, these time-consuming individual calculations are

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

performed only once. (An OODBMS uses messages to allow one object to ask for the results of another object's methods. Storing outputs is DAESSIM's analog to messaging.)

3.4.3 Modeling of Physical Structures

As an infrastructure information management system, DAESSIM models physical *structures*, like oil platforms or buildings. Each structure instance is recorded in table STRUCTURE, in the form of its attributes Structure_Name, Structure_ID (unique), Structure_Type (restricted by table STRUCTURE_TYPE), Structure_Description, and Structure_Comment. No other descriptive information - geometry, installation date, builder, etc - is stored in STRUCTURE. Also, there is no sub-class of STRUCTURE for storing such information. Instead, it is stored as a special type of evaluation, called 'structure input' object instances, in FACTOR_DATA. (Omitted for clarity from Section 3.3 was the Structure_ID attribute of the EVALUATION entity, used to register each evaluation with a structure.)

Thus, there is only one structure instance per structure object, but there can be many structure input object instances. Using structure input object instances to store most structure information, rather than the structure instances themselves, allows DAESSIM to track characteristics over time. For example, if a structure is moved and its latitude and longitude are contained in a structure instance, the past information would be lost as it is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

rewritten with each move. If, however, that structure's latitude and longitude are contained in a structure input object, then a comparison of the object instances over time will reveal the platform's many locations (see Section 4.3.7 for an example).

If an organization decides to change the type of data that's stored for a type of structure, no database redesign is needed for DAESSIM. Instead, another type of structure input entity would be added, and used instead of, or in addition to, the existing one. For example, a more traditional STRUCTURE entity might store geometric information in a manner that makes sense to the DBMS designer - perhaps oil modeling platforms from the bottom up. An engineering methodology might be acquired, though, that requires platform geometry to be modeled from the top down. The traditional system would need to be redesigned to accommodate the difference; DAESSIM would simply need the definition of a new evaluation type (see Section 5.2.9 for details of this operation).

Another benefit of DAESSIM's structure modeling methodology is that any structures may be added to the system, not just steel-jacketed offshore oil platforms. A user needs only to define the new structure type, define one or more structure input objects, and begin data entry.

3.4.4 Handling of Fleets

A *fleet* is defined in DAESSIM as a collection of one or more structures. Much as for structure instances, minimal fleet information is stored in the FLEET entity: Fleet_ID,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Fleet_Type, Fleet_Description, and Fleet_Comment. Through the many-to-many relationship implemented as the FLEET_STRUCTURE_REGISTER table, structures are (optionally) registered as belonging to fleets.

3.4.5 Handling of Scenarios

Also omitted for clarity from Section 3.3 was the Fleet_ID attribute of table EVALUATION. This allows evaluation instances to be registered as pertaining to fleets. Special evaluation types that are designed to apply to fleets are known in DAESSIM as *scenarios*. Table SCENARIO_EVALUATION_REGISTER is the implementation of a many-to-many relationship between EVALUATION and FLEET_STRUCTURE_REGISTER that tracks which scenarios pertain to which fleets, while allowing data to be stored in FACTOR_DATA.

3.5 SUMMARY

This chapter introduces Relational Object technology that splits the database model into a relational substructure and an object-oriented superstructure, and uses the substructure to manage factors rather than sets of factors. Implemented in DAESSIM (the Decision, Analysis, and Engineering Support System for Infrastructure Management), Relational Object technology provides an effective tool for the management and integration of information used in engineering analyses. Designed to be applied to infrastructure systems

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(fleets as well as individual structures). DAESSIM may be customized to meet users' needs.

Advantages of the DAESSIM data structure:

- Substructure is relational and based on mathematical theory - operations are consistent and predictable The substructure is normalized.

- Non-proprietary substructure - data may be accessed, imported, and exported by SQL-enabled applications.

- *Ad hoc* queries may be executed - because DAESSIM data are not encapsulated, *ad hoc* queries may be run: not only on input factors, but on outputs as well (due to the storage of outputs).

- Referential integrity - cascading deletes are enabled on evaluations, fleets, and structures. For example, if a structure type is deleted, then all structure instances of that type are deleted, as well as all evaluation instances (including factors) pertaining to any of the instances of that structure type.

- Intuitive interface - the user-accessible components of DAESSIM were designed to be familiar to engineers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Extensibility - new structure types, structure instances, fleet types, fleet instances, evaluation types, and evaluation instances may be freely added to DAESSIM by the user.

- Flexibility – DAESSIM can adapt to previously unknown hierarchy schemes, and can provide for more than one scheme per structure type.

- Interoperability - since evaluations are units of program code, they can easily interface with each other as well as external applications. For example, in the PIMS sample application, the LIVMax evaluation type utilizes an Excel spreadsheet.

- Object ID's - each evaluation instance is uniquely identified. Unlike object-oriented systems, DAESSIM's objects are identified only once: OODBMS objects are stored intact in the data management system, and identified both internally and by external pointers; DAESSIM objects have no internal identification because they don't "exist" as intact objects (their ID's allow code objects - 'evaluation types' - to retrieve a certain set of data and to store a certain set of data).

- Joins - because output factors are stored, evaluations may be joined for data exploration.

- Temporal information - different evaluation instances, pertaining to the same structure, can be sorted to reveal changing information over time.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Closure - the output of an evaluation can serve as the input of another.

Disadvantages of the DAESSIM structure:

- Lack of declarative support at the factor level impedes factor data integrity - since factors are defined as instances in FACTOR_ALLOWED rather than as attributes, table-level data integrity constraints cannot be applied. Factor data integrity must be implemented as on-entry and subsequent periodic-polling application code based on Low_Value and High_Value attributes of FACTOR_ALLOWED. Declarative support exists at the structure and evaluation levels.

- Proprietary superstructure - DAESSIM functions and evaluation objects are proprietary. It is possible, however, that a combination of server technology and a platform-independent programming language may be used to create a non-proprietary superstructure.

- Slow operations - because evaluations are treated as objects with non-encapsulated data, data retrieval operations are much slower than with traditional RDBMS's.

- Slow transactions - while an engineer is using the DAESSIM to perform a lengthy evaluation, another user may run the same type of evaluation using the same evaluation ID. The last information entered will overwrite earlier data for that evaluation ID.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Abstract data types - abstract data types are awkwardly handled in DAESSIM, but in an improved way than in traditional relational models: by breaking down each data type into its components, and storing those components as factor instances. A relational system would require separate attributes for each component. When such a breakdown cannot be performed, as in the storage of a picture, the data in the object are fully encapsulated and cannot be accessed except by accessing the entire object.

- Flexibility and extensibility - when a new evaluation type or scenario type is added to DAESSIM, it must be coded using DAESSIM's programming language. Templates are supplied with the PIMS sample application to aid the user. Additionally, some familiarity with analysis techniques and programming concepts is assumed to be possessed by anyone wishing to perform these operations.

- Inheritance - for expediency, inheritance was not incorporated into DAESSIM beyond the evaluation / evaluation type / evaluation instance level (evaluation types inherit the basic properties of the evaluation object, and evaluation instances inherit the basic properties of their types, but sub-types do not automatically inherit the properties of their parent types).

- Impedance mismatch - the DAESSIM superstructure (at least as prototyped in Microsoft Access) must be programmed with an upper-level language, while SQL is used for the substructure.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Hierarchy levels - the substructure is currently limited to thirty-two hierarchy levels, while the superstructure is limited to four. Extending the former would involve an extensive redesign of the system; extending the latter is a matter of programming the proper functions.

- Inefficient storage - table FACTOR_MODIFIER includes ten null attributes for each datum stored.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4 DEMONSTRATIONS OF CAPABILITY

4.1 INTRODUCTION

4.1.1 Chapter Overview

The purpose of this chapter is to demonstrate that DAESSIM possesses its desired capabilities, as developed in Sections 2.2 and 3.1 and repeated in Table 4-17. To this end, an instance of DAESSIM was created, called the Platform Information Management System (PIMS).

Section 4.2 of this chapter describes the evaluation classes included in PIMS and how they interrelate. Section 4.3 uses PIMS and DAESSIM to demonstrate DAESSIM's capabilities. Throughout, this chapter is intended to demonstrate DAESSIM's data management capabilities; step-by-step instructions for the use of PIMS are provided in Chapter 6. Details of the database's structure are available in Appendices B and C.

4.1.2 PIMS and Its Relationship to DAESSIM

DAESSIM is a software environment that provides a relational object structure for the management of data relating to infrastructure analysis and decision-support activities. Using the terminology of Chapter 3, DAESSIM may be considered as an *object class*. PIMS is a version, or *instance*, of the DAESSIM object which contains sample evaluation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

objects designed to demonstrate some of DAESSIM's capabilities. These evaluation objects are inspection objects, engineering assumption objects, structure input objects, and analysis objects relating to the safety assessment of steel jacketed offshore platforms located in the Gulf of Mexico.

4.1.3 Disclaimer

While some of the evaluation object classes included in PIMS are based on the published or unpublished works of other authors, all are intended only as examples and not for decision making.

4.2 SAMPLE EVALUATION CLASSES INCLUDED IN PIMS

Sections 4.2.1 and 4.2.2 are organized with an emphasis on the interrelationships between objects. Section 4.2.1 is concerned with the tasks involved in performing simple risk analyses of a structure and of a fleet. Section 4.2.2 is concerned with the tasks involved in performing a moderately complex structural analysis on a platform.

The subsections under Sections 4.2.1 and 4.2.2 list the individual sample evaluation object classes that are included in PIMS, and briefly discuss the methodologies behind them. The sections also list the evaluation classes' attributes, and discuss the methods by which the attributes are stored. Finally, the interrelationships between the evaluation objects are discussed.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

None of the interrelationships shown in this section were created during the design of DAESSIM; rather, they were created during the production of the demonstration PIMS application. In other words, all of them were produced by a user of DAESSIM during the normal course of using it. These interrelationships correspond to the relationships of a traditional entity-relationship diagram – except that these are dynamically created by the user upon the creation of an evaluation object class. The ease with which such linkages are created is demonstrated in Section 6.2.2.

4.2.1 Information Flow for Simplified Risk Analysis of a Platform or Fleet

In Figure 4-1, sample evaluation instances are represented by the object notation introduced in Chapter 3 and in Appendix A (see, especially, Figure A-10). Object names are sample evaluation type names; e.g., the name of the upper-left evaluation is "L1I – Structure 1," which is an L1I1 instance applicable to a structure named "Structure 1" (more than one evaluation instance may apply to a structure, a fact disregarded here for clarity). Object attributes are those of the evaluation type involved (e.g., "Factor 2" in the "L1I1 – Structure 1" instance), as are object methods (e.g., "Factor 27" in the "L1RSR1 – Structure 1" instance).

Direct links (established by the user upon creation of new evaluation classes) between evaluations are denoted by solid arrows in Figure 4-1. *Indirect links*, which are consequences of the direct links, are denoted by dashed arrows. As indicated in the figure, direct links (analogous to foreign keys in a relational database) are established when the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Evaluation_ID of one evaluation is specified as one of the input factors of another evaluation. Later, when the second evaluation opens the first (based on the direct link), all the of the first evaluation's attributes are made available to the second evaluation. This is the *indirect linkage* process.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

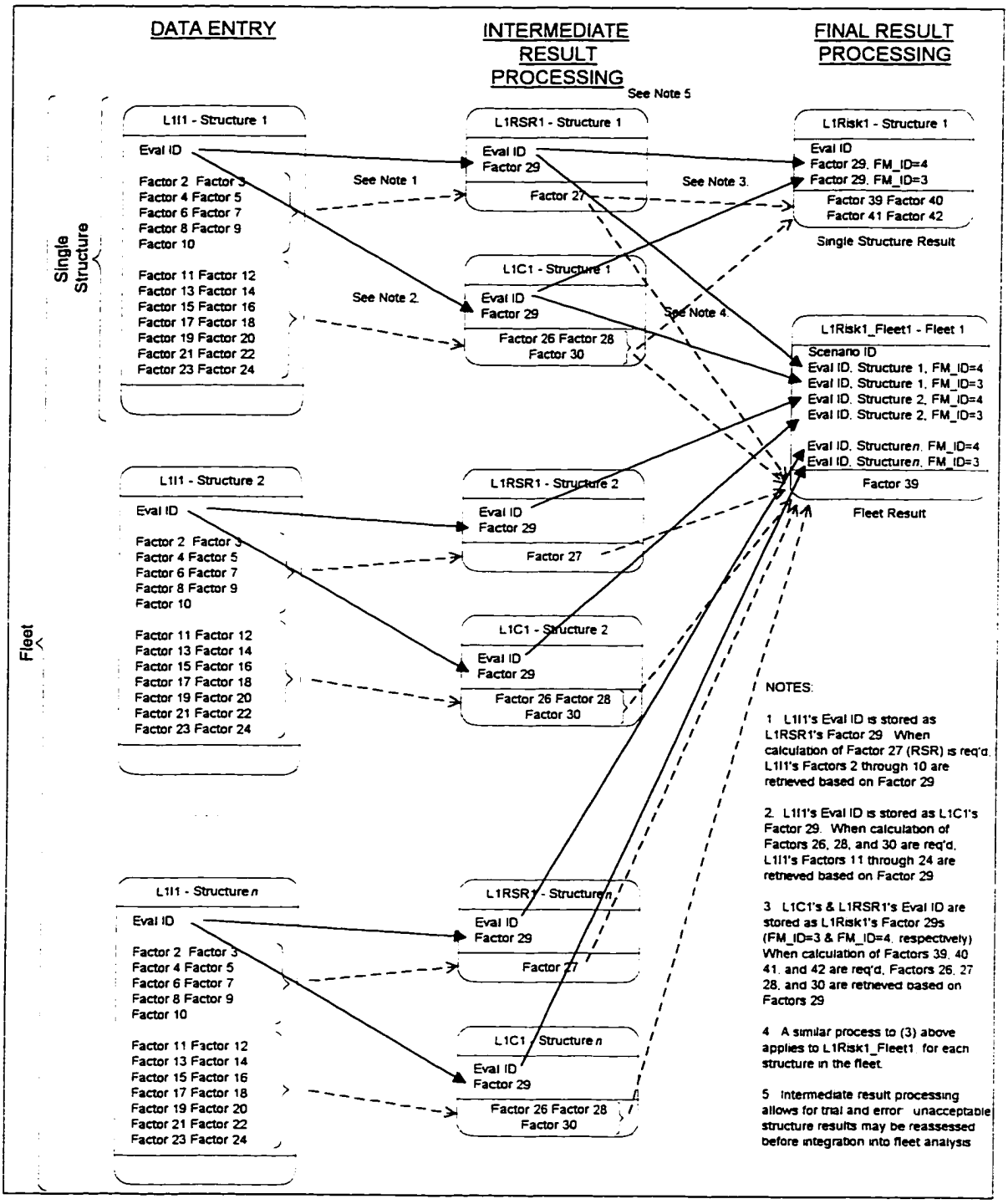


Figure 4-1: Forward Chaining of L1 Risk Information

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The following subsections discuss the component evaluation classes shown in Figure 4-1.

4.2.1.1 L1I1

Methodology: An L1I1 ("Level One Inspection, version 1") inspection instance is necessary to provide data to the L1C1, L1RSR1, L1Risk1, and L1Risk1_Fleet1 evaluations.

Attributes: In DAESSIM (not just within PIMS), there are basic attributes that are stored regardless of the type of evaluation being performed. All structure objects store Name, ID, Type, Description, and Comments attributes. All evaluation objects store Evaluation_ID, Structure_ID, Evaluation_Type, Begin_Date, End_Date, Evaluator, Description, and Comments attributes. Fleet objects store Fleet_ID, Type, Description, and Comments attributes. Scenario objects store Fleet_ID, Scenario_Type, Scenario_ID, Evaluator, Begin_Date, End_Date, Description, and Comments. Evaluation and scenario objects, however, store many more attributes. These are the factors that are listed in this and the following sections. Table 4-1 lists the factors (object attributes) solicited and stored by an L1I1 inspection (object instance). Factors are listed by their unique ID's, names, types, and descriptions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Factor ID	Factor Name	Factor Type	Factor Description
2	R1	Double	Structure & foundation design and construction criteria
3	R2	Double	Structure condition: corrosion, dented & bent members, dropped objects, fouling, scour
4	R3	Double	Structure and foundation modifications developed during installation, operations, or reassessments that result in increases or decreases in capacity
5	R4	Double	Structure and foundation configuration
6	R5	Double	Loading-capacity effects factor, Fv
7	S1	Double	Storm loadings design criteria (ref 1993 API RP 2A)
8	S2	Double	Lower equipment deck elevation (not in design wave loading)
9	S3	Double	Loading modifications: elements added or removed, marine growth management
10	S4	Double	Operating/gravity loading modifications
11	C1_L1	Yes/No	Is the platform permanently manned?
12	C1_L2	Yes/No	Is the platform temporarily manned?
13	C1_S1	Yes/No	Is crude stored on the deck?
14	C1_S2	Yes/No	Does the platform have oil wells?
15	C1_S3	Yes/No	Do the wells have functioning SSSVs?
16	C1_S4	Yes/No	Are any risers connected to the platform?
17	C1_S5	Yes/No	Do the risers have functioning ESD valves?
18	C1_E1	Yes/No	Is the production level significant?
19	C1_E2	Yes/No	Is the platform multi-functional?
20	C1_E3	Yes/No	Is it connected to other platforms?
21	C1_E4	Yes/No	Will the operation of other platforms be significantly affected?
22	C1_E5	Yes/No	Will contractual obligations be affected by loss of the platform?
23	C1_E6	Yes/No	Will the platform be costly to replace?
24	C1_L3	Yes/No	Is an evacuation system provided?

Table 4-1: L1I1 Object Attributes (Factors)

Storage: Since the L1I1 object has only a single hierarchy level, storage is straightforward. For each L1I1 instance, a unique Evaluation_ID is allocated to an existing Structure_ID, and one value per factor is assigned to the Evaluation_ID. Accordingly, the value of FM_ID (the hierarchy indicator attribute introduced in Section 3.3.2) will be the default, 1, for all L1I1 factor values stored in PIMS's table FACTOR_DATA (see Ap-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

pendix B, "DAESSIM Entity-Relationship Diagram," for an overview of the DAESSIM/PIMS structure). Since FM_ID and Factor ID form part of a composite primary key in FACTOR_DATA, multiple values would be allowable for any factor for which multiple FM_ID's are available. Restricting the FM_ID to a single value for a factor restricts the factor itself to a single value.

Relationships: L1I1 object instances serve as collections of input attributes for L1C1 and L1RSR1 object instances. Factors 2 through 10 are used for the L1RSR1 object, and the others are used for the L1C1 object. All factors are used indirectly (via the L1C1 and L1RSR1 objects) for L1Risk1 and L1Risk1_Fleet1 evaluations.

4.2.1.2 L1RSR1

Methodology: The logic of the L1RSR1 ("Level One Reserve Strength Ratio, version 1") output object included with PIMS was originally developed by Bea and Craig [1993], and is based on the equation:

$$RSR = \frac{R_1 \cdot R_2 \cdot R_3 \cdot R_4 \cdot R_5}{S_1 \cdot S_2 \cdot S_3 \cdot S_4}, \qquad (1)$$

where RSR is the Reserve Strength Ratio, and factors R_1 through S_4 are subjective factors obtained from a prior L1I1 evaluation.

Attributes: Table 4-2 lists the factors (object attributes) calculated and stored by an L1RSR1 evaluation (object instance).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Factor ID	Factor Name	Factor Type	Factor Description
27	RSR	Double	Reserve Strength Ratio
29	ID, Prior Evaluation	Long Integer	ID of source evaluation

Table 4-2: L1RSR1 Object Attributes (Factors)

Factor 29 stores the ID of an existing L1I1 inspection, thus designating it as an input. This enables the L1RSR1 evaluation, when executed, to retrieve the values of the attributes stored in the L1I1 instance. An L1RSR1 evaluation results in one output, Factor 27, per equation (1).

Storage: The L1RSR1 object is another object with only a single hierarchy level. For each L1RSR1 instance, a unique Evaluation_ID is allocated to an existing Structure_ID, and one value per factor is assigned to the Evaluation_ID. Thus the value of FM_ID will be 1 for all L1RSR1 factor values stored in table FACTOR_DATA, with the exception of Factor 29.

Factor 29 is managed slightly differently. Without additional restrictions, Factor 29 could store the ID of any existing evaluation – which may or may not be of the desired L1I1 class. When the L1RSR1 object was created, therefore, a specific FM_ID value (in PIMS table FACTOR_MODIFIER) was mandated to be used in conjunction with Factor 29. The FM_ID value was established by the sequence in which evaluations were added to DAESSIM, and in this case happens to be 5 (selecting the proper FM_ID when adding a new evaluation type is handled automatically by DAESSIM – see Section 6.2.2).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Relationships: An L1RSR1 instance cannot exist without a corresponding L1I1 instance. By the simple expedient of specifying an L1I1 evaluation to be used as an L1RSR1 input, the nine pertinent L1I1 attributes (Factors 2 through 10) are indirectly assigned to the L1RSR1 evaluation.

4.2.1.3 L1C1

Methodology: The L1C1 ("Level One Consequence, version 1") output evaluation class included with PIMS is based on procedures developed by Aggarwal for platforms in the Gulf of Mexico [Aggarwal 1991]. It is a subjective approach, as illustrated in Figure 4-2 and Figure 4-3:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

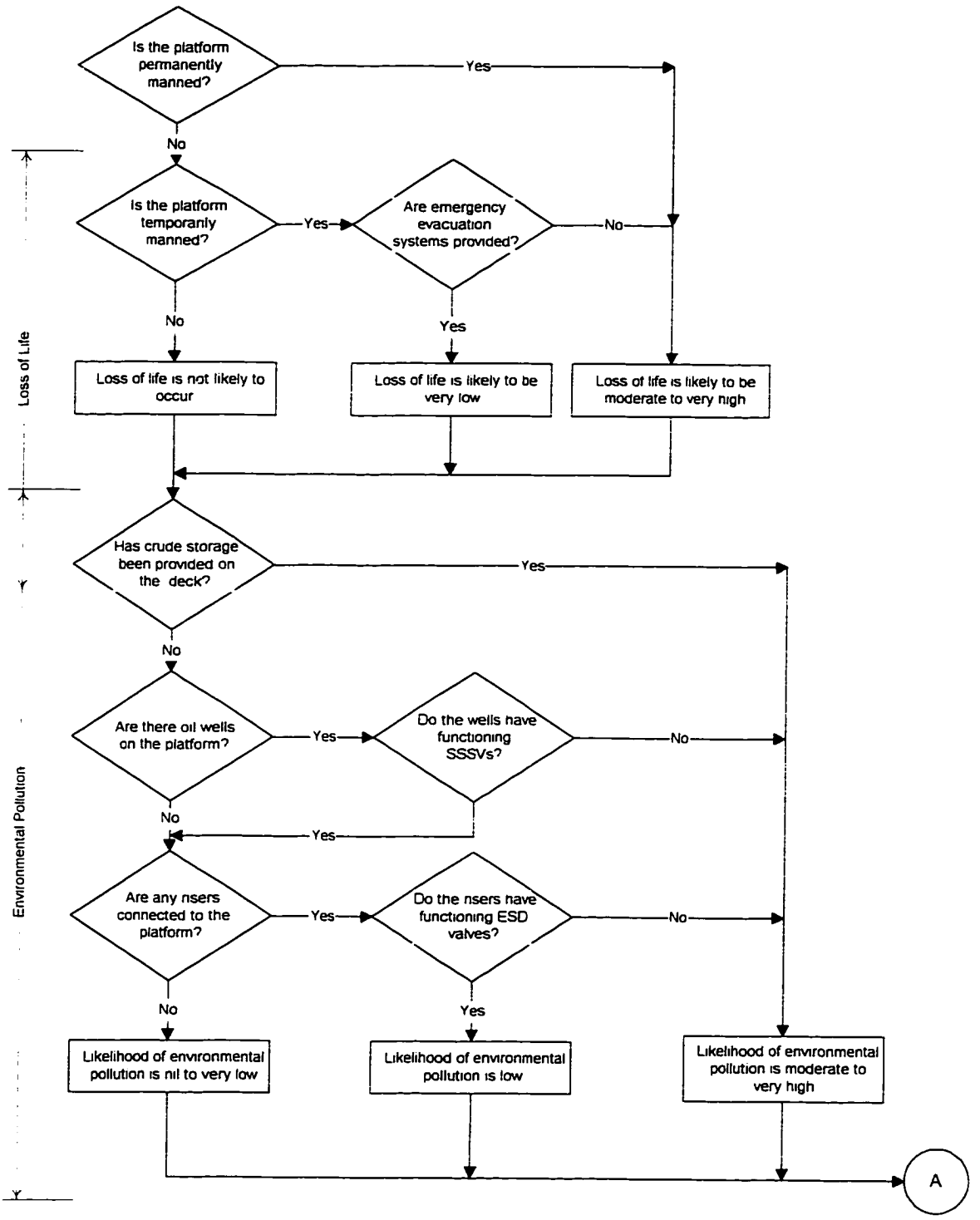


Figure 4-2: Consequence Evaluation [Aggarwal 1991], 1 of 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

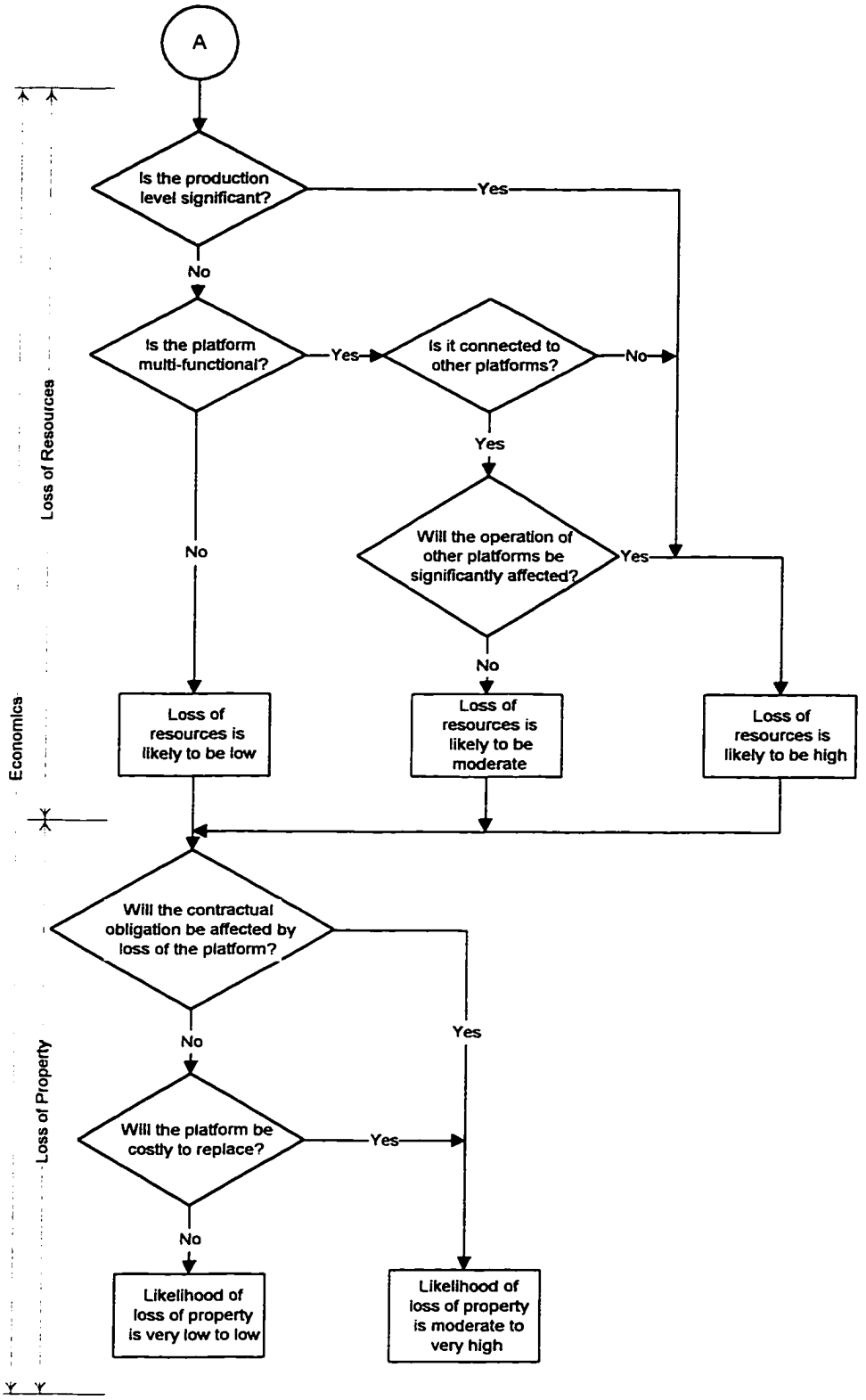


Figure 4-3: Consequence Evaluation [Aggarwal 1991], 2 of 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Attributes: Table 4-3 lists the factors calculated and stored by an L1C1 evaluation.

Factor ID	Factor Name	Factor Type	Factor Description
26	CL1	Text	Life Safety consequence assessment
28	CE1	Text	Economics consequence assessment
29	ID, Prior Evaluation	Long Integer	ID of source evaluation
30	CS1	Text	Spillage consequence assessment

Table 4-3: L1C1 Object Attributes (Factors)

Again, Factor 29 allows an existing L1I1 inspection to be designated as an input. This enables the L1C1 evaluation, when executed, to retrieve the values of the attributes stored in the L1I1 instance. An L1C1 evaluation results in three outputs, Factors 26, 28, and 30, which store the appropriate Life Safety, Economics, and Spillage consequence assessments.

Storage: The L1C1 object is yet another object with only a single hierarchy level. For each L1C1 instance, a unique Evaluation_ID is allocated to an existing Structure_ID, and one value per factor is assigned to the Evaluation_ID. The value of FM_ID is 1 for all L1RSR1 factor values stored in table FACTOR_DATA, except for Factor 29. For the reasons discussed in Section 4.2.1.2, the value of FM_ID is 5 for the Factor 29 value of each instance of L1C1.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Relationships: An L1C1 instance cannot exist without a corresponding L1I1 instance. By specifying an L1I1 evaluation to be used as an L1C1 input, fourteen L1I1 attributes are assigned to the L1C1 evaluation (Factors 11 through 24).

4.2.1.4 L1Risk1

Methodology: The L1Risk1 ("Level One Risk, version 1") output evaluation class included with PIMS is based on procedures developed by Aggarwal [1991], Bea and Craig [1993], and Staneff et al [1994, 1995]. It assesses the risk due to storm waves for oil platforms. Incorporating the subjective L1RSR1 and L1C1 methodologies makes L1Risk1 a subjective approach.

The L1Risk1 methodology plots consequences against structural integrity to achieve an assessment of risk (Figure 4-4). It does not combine consequences into one measure but plots the different categories – life, spillage, and economics – separately (an arbitrary decision for illustrative purposes). To do this, the methodology converts the textual results of an L1C1 analysis as follows:

1. Loss of Life:
- "Loss of life is not likely to occur" = 0.5
- "Loss of life is likely to be very low" = 2.5
- "Loss of life is likely to be moderate to very high" = 4.5
- Otherwise = 4.5

2. Spillage:
- "Likelihood of environmental pollution is nil to very low" = 0.5

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- "Likelihood of environmental pollution is low" = 2.5
- "Likelihood of environmental pollution is moderate to very high" = 4.5
- Otherwise = 4.5

3. Economic Loss:

- "Resource loss is likely to be low and property loss is likely to be very low to low" = 0.5
- "Resource loss is likely to be moderate and property loss is likely to be very low to low" = 2.5
- Otherwise = 4.5

These numeric consequence values are completely arbitrary, and are provided for illustration only. They are plotted separately along the vertical axis. The RSR result from an L1RSR1 analysis is plotted along the horizontal axis. Thus the three consequence results will form three points on the graph, which will be identified as "Acceptable," "Marginal," or "Not Acceptable" according to their location relative to the solid "Acceptable" and "Marginal" lines. (The L1Risk1 evaluation can serve as a template for a new L1Risk*x* evaluation class, using different equations for the "Acceptable" and "Marginal" lines – thus using a different safety standard. See Section 4.3.5.)

The user should not draw false conclusions from the Consequence Measure vs. RSR chart. This is possible because the chart apparently illustrates deterministic results. The user must bear in mind that uncertainties exist in the assessment of RSR, in the assessment of consequence measures, and in the placing of the "Acceptable" and "Marginal" lines themselves. Thus the data points in the output charts should be interpreted as

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

blurred circles or ovoids, possibly overlapping much thicker "Acceptable" or "Marginal" lines, and labeled as "probably acceptable" or "probably marginal."

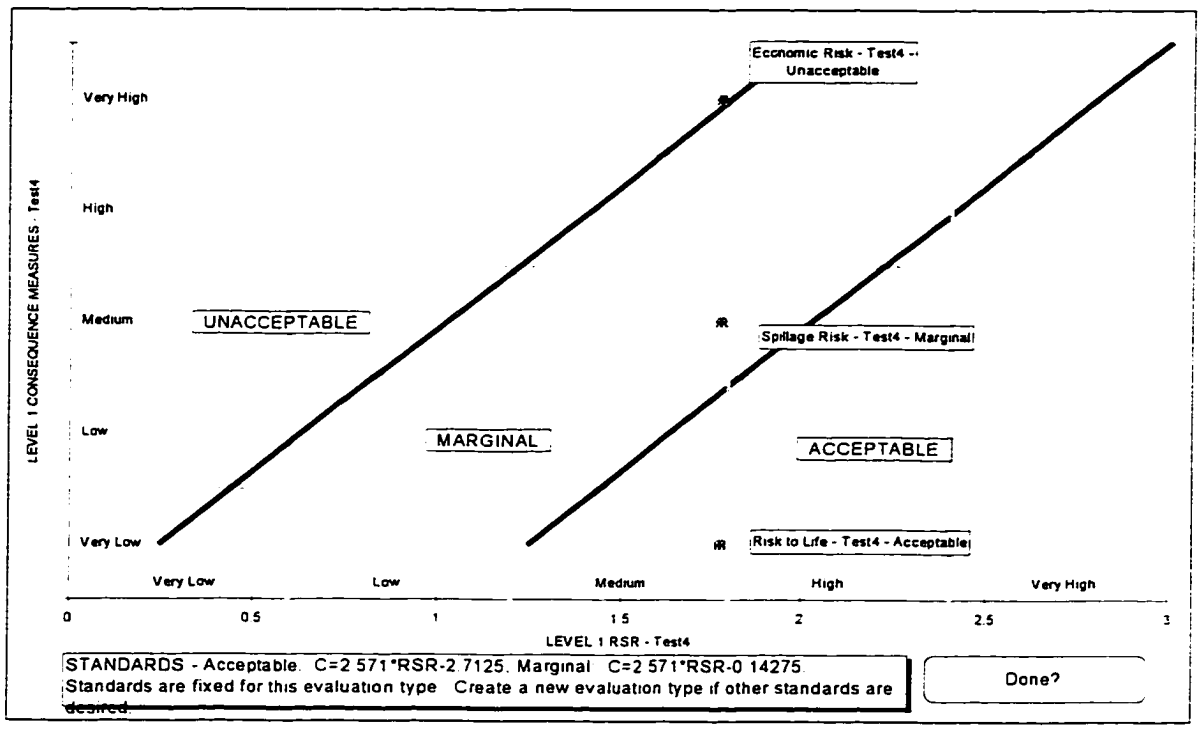


Figure 4-4: L1Risk1 Risk Evaluation Chart

Attributes: Table 4-4 lists the factors calculated and stored by an L1Risk1 evaluation.

ID	Factor Name	Factor Type	Factor Description
29	ID. Prior Evaluation	Long Integer	ID of source evaluation
39	L1 Risk Chart	OLE	Picture of chart generated in L1 risk evaluation
40	L1 Life Risk	Text	L1 Life Risk Assessment
41	L1 Spillage Risk	Text	L1 Spillage Risk Assessment
42	L1 Econ Risk	Text	L1 Economics Risk Assessment

Table 4-4: L1Risk1 Object Attributes (Factors)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Possible Factor 40, 41, and 42 values are "Unacceptable," "Marginal," and "Acceptable."

Storage: The L1Risk1 object has a single hierarchy level. For each L1Risk1 instance, a unique Evaluation_ID is allocated to an existing Structure_ID, and one value per factor is assigned to the Evaluation_ID. The value of FM_ID will be 1 for all L1Risk1 factor values stored in table FACTOR_DATA (with the exception of Factor 29).

With L1Risk1, Factor 29 gets more complicated. There are two input evaluations needed for an L1Risk1 instance, and the ID of each is stored as Factor 29. However, each is stored in table FACTOR_DATA with a different FM_ID. For the Factor 29 value intended to designate the L1C1 instance, the value of FM_ID is 3; for the Factor 29 value designating the L1RSR1 instance, FM_ID is 4.

Relationships: An L1Risk1 instance cannot exist without component L1C1 and L1RSR1 instances, and they in turn cannot exist without an L1I1 instance(s). Specifying the input L1C1 and L1RSR1 instances assigns the attributes of those evaluations. It also makes the results of the L1Risk1 evaluation directly related to the inputs into the L1I1 evaluation(s) assigned to the L1C1 and L1RSR1 instances involved.

4.2.1.5 L1Risk1_Fleet1

Methodology: Like the L1Risk1 evaluation class, the L1Risk1_Fleet1 ("Level One Fleet Risk, version 1") output object depends heavily upon the results of the L1RSR1 and L1C1 evaluation types (which in turn depend on the L1I1 evaluation type). Rather than

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

create an L1Risk1 risk assessment chart for a single platform, an L1Risk1_Fleet1 scenario creates a storm wave-induced risk assessment chart for an entire fleet of up to 200 structures. As with the L1Risk1 evaluation class, L1Risk1_Fleet1 considers consequences categorically (and uses the same arbitrary weighting factors as L1Risk1). Also like the L1Risk1 evaluation class, the user must not be misled because of the false impression of deterministic results. Unseen uncertainties may exist in the results.

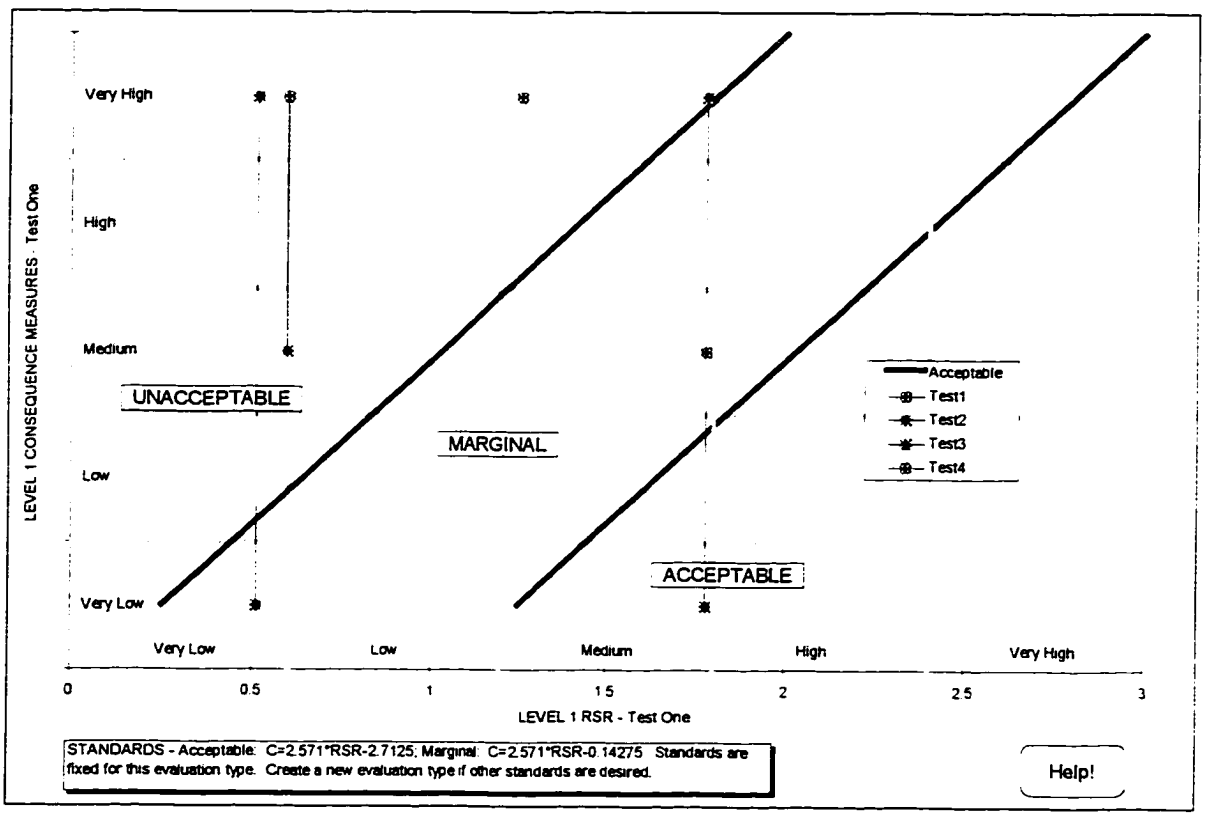


Figure 4-5: L1Risk1_Fleet1 Results Chart

Attributes: Table 4-5 lists the factors calculated and stored by an L1Risk1_Fleet1 evaluation. The only output of the evaluation class is Factor 39, the L1 Risk Chart.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ID	Factor Name	Factor Type	Factor Description
29	ID, Prior Evaluation	Long Integer	ID of source evaluation
39	L1 Risk Chart	OLE	Picture of chart generated in L1 risk evaluation

Table 4-5: L1Risk1_Fleet1 Object Attributes (Factors)

Storage: L1Risk1_Fleet1 is a single hierarchy scenario object (a special case of an evaluation object). For each L1Risk1_Fleet1 instance, a unique Evaluation_ID is assigned not to an existing Structure_ID, but to an existing Fleet ID. The single value of Factor 39 is then assigned to the Evaluation_ID in table FACTOR_DATA. The value of FM_ID is 1 for this entry.

Two input evaluations (an L1C1 instance and an L1RSR1 instance) must be tracked for each platform in the fleet to which the L1Risk1_Fleet1 scenario is assigned. PIMS table SCENARIO_EVALUATION_REGISTER is used for this. For each input evaluation, the Evaluation_ID of the scenario is entered (as "Scenario_ID"), along with the Fleet ID, the Structure_ID that the input evaluation applies to, and the ID of the selected input evaluation (see Section 6.3.7 for details of evaluation selection).

Factor 29 itself is not directly used in the storage of L1Risk1_Fleet1 attributes. Entered twice in the setup of the object class, once with FM_ID = 3 (for L1C1) and once with FM_ID = 4 (for L1RSR1), it serves as a "pointer" that tells the object which types of input evaluations to look for.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Relationships: An L1Risk1_Fleet1 instance cannot exist without component L1C1 and L1RSR1 instances for each platform in the fleet, nor can it exist without a collection of platforms that has been designated as a "fleet." The L1C1 and L1RSR1 instances, in turn, cannot exist without L1I1 instances. Specifying the input L1C1 and L1RSR1 instances assigns the attributes of those evaluations. It also makes the results of the L1Risk1_Fleet1 evaluation directly related to the inputs into the L1I1 evaluation(s) assigned to the L1C1 and L1RSR1 instances involved.

4.2.2 Information Flow for Structural Analysis of a Platform

Figure 4-6 shows the direct (solid arrows) and indirect links (dashed arrows) leading to a PIMS L2RSR1 or L2RSR2 structural analysis of a platform. The subsections following Figure 4-6 discuss its component evaluation classes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

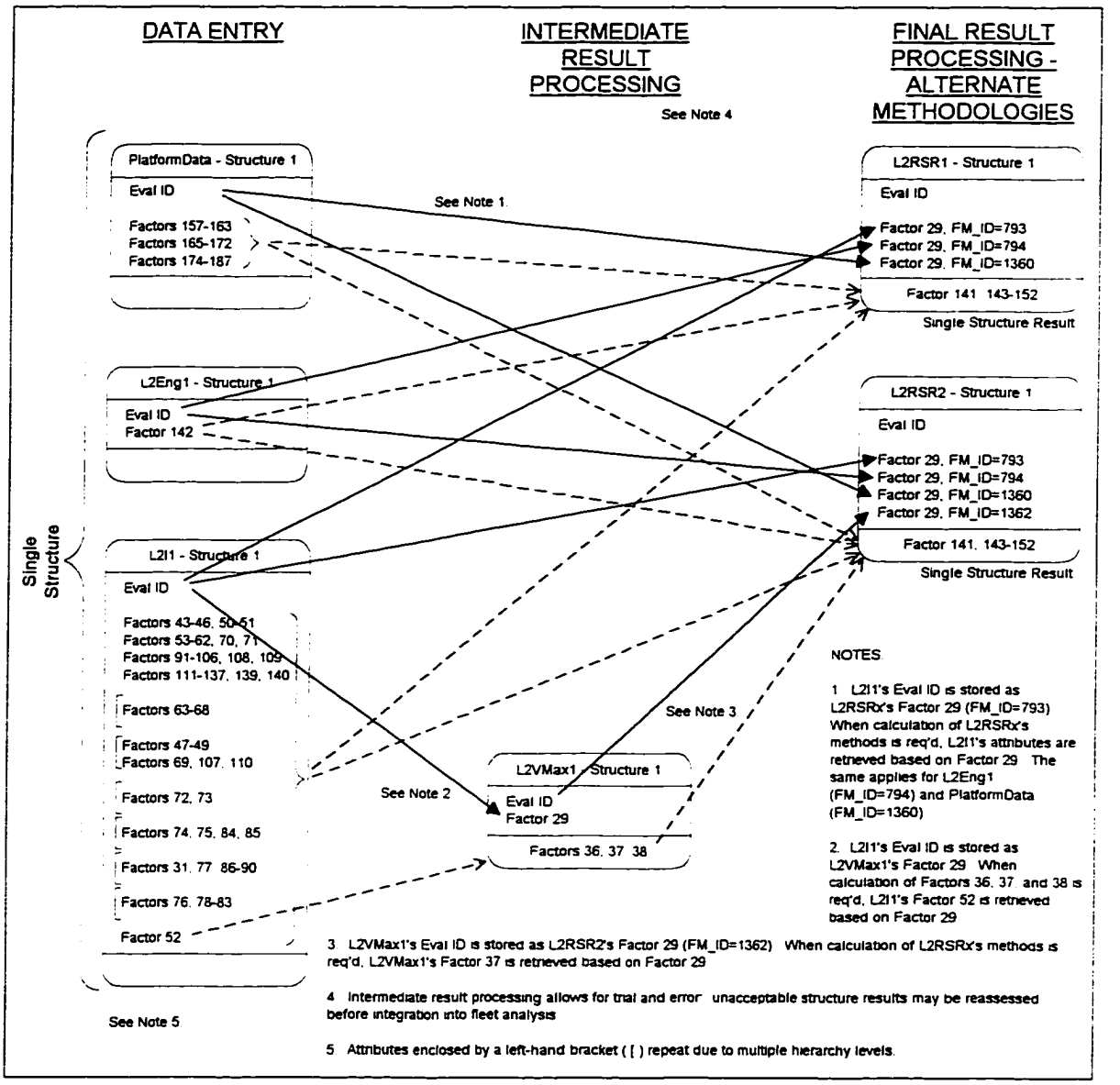


Figure 4-6: Forward Chaining of L2 RSR Information

4.2.2.1 PlatformData

Methodology: A PlatformData evaluation is a structure input object that stores basic data about platforms. It is generally used to store data that do not often change (e.g., water

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

depths). However, multiple PlatformData evaluations can be performed on a platform; conflicting information indicates a change in the platform's basic configuration (e.g., if a platform has been moved, the water depths will probably differ). Such changes are revealed to the user by running an *ad hoc* query that lists the data for all PlatformData evaluations on a particular structure, ordered by the date of the evaluation.

Attributes: Table 4-6 lists the factors solicited and stored by a PlatformData evaluation. Factors are listed by their unique ID's, their names, types, and allowable extreme values (if any).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ID	Factor Name	Factor Type	Factor Description
157	Latitude	Text	Latitude
158	Longitude	Text	Longitude
159	Lambert N	Double	Lambert N
160	Lambert E	Double	Lambert E
161	Lease #	Text	Lease Number
162	Lease Expiration	Date/Time	Lease Expiration
163	Slots	Double	Slots
165	Water Depth	Double	Water Depth
166	Miles to Land	Double	Miles to Land
167	Installed Date	Date/Time	Installed Date
168	First Production Date	Date/Time	First Production Date
169	Location	Text	Location
170	MBBL Daily	Double	Production, Thousands of Barrels, Daily
171	MMCF Daily	Double	Production, Millions of Cubic Feet, Daily
172	Status	Text	Status
174	Blockline E/W	Text	Blockline East/West
175	Blockline N/S	Text	Blockline North/South
176	Quarters Type	Text	Quarters Type
177	Designer	Text	Firm Name, Designer
178	Fabricator	Text	Firm Name, Fabricator
179	Installer	Text	Firm Name, Installer
180	Area Block	Text	Area Block
181	Removed Date	Text	Removed Date
182	Feet From Lease Line	Double	Feet From Lease Line
183	X Coordinate	Double	X Coordinate
184	Y Coordinate	Double	Y Coordinate
185	Source	Text	Source
186	MMS Structure ID	Text	Structure ID assigned by MMS
187	MMS Complex ID	Text	Complex ID assigned by MMS

Table 4-6: PlatformData Object Attributes (Factors)

Storage: Since the PlatformData object has only a single hierarchy level, storage is straightforward: for each PlatformData instance, a unique Evaluation_ID is allocated to an existing Structure_ID, and one value per factor is assigned to the Evaluation_ID. Thus the value of FM_ID will be the default, 1, for all PlatformData factor values stored in PIMS table FACTOR_DATA.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Relationships: As implemented in PIMS, the PlatformData object serves as a collection of inputs for the L2RSRx objects. Only one of the inputs is required by the L2RSRx objects, however: Factor 165, Water Depth.

4.2.2.2 L2I1

Methodology: An L2I1 ("Level Two Inspection, version 1") inspection evaluation is necessary to provide data to the L2VMax1, L2RSR1, and L2RSR2 evaluations. An L2I1 inspection is hierarchical: it stores data not just about the structure, but about physical sections and subsections of the structure.

Attributes: The L2I1 object has several multiple-level hierarchies, meant to describe aspects of the geometry of an oil platform per the ULSLEA methodology [Mortazavi 1995] (and corresponding to the FOR...NEXT loops in the ULSLEA program code). Tables 4-7 through 4-13 list the factors solicited and stored by each hierarchy of an L2I1 inspection object. The maximum number of levels per hierarchy is four:

- Platform (Table 4-7)
- Platform / Decks (Table 4-8)
- Platform / Bays (Table 4-9)
- Platform / Orientations / Bays (Table 4-10)
- Platform / Bays / Horizontal Braces (Table 4-11)
- Platform / Joints (Table 4-12)
- Platform / Orientations / Bays / Diagonal Braces (Table 4-13)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Factors 121 through 136 in Table 4-7 are bias and coefficient of variation factors, indicating DAESSIM's capability to handle uncertainties. Section 4.2.2.5 further discusses DAESSIM's support for ULSLEA's reliability features.

ID	Factor Name	Type	Factor Description
43	Platform Type	Integer	1=4 leg, 2=6 leg, 3=8 leg, 4=12 leg.
44	Number of Decks	Integer	How many decks are there?
45	Number of Jacket Bays	Integer	How many jacket bays are there?
46	Number of Joints	Integer	How many joints are in the platform?
50	Surge Depth	Double	Surge depth
51	Wind Velocity	Double	Wind velocity at 30' elevation.
52	Wave Height	Double	Wave height
53	Wave Period	Double	Wave p. iod.
54	Current Velocity at SWL	Double	Current velocity at SWL.
55	Current Velocity at Mudline	Double	Current velocity at mudline.
56	Current Profile	Integer	Current profile
57	Total Deck Load	Double	Total deck load
58	End-on Base Centerline Width	Double	End-on base centerline width
59	End-on Top Centerline Width	Double	End-on top centerline width.
60	Broadside Base Centerline Width	Double	Broadside base centerline width
61	Broadside Top Centerline Width	Double	Broadside top centerline width
62	Broadside Middle Section Width	Double	Broadside middle section width
70	Deck Leg Diameter	Double	Deck leg diameter.
71	Deck Leg Thickness	Double	Deck leg thickness.
91	Pile Length	Double	Pile length.
92	Pile Diameter	Double	Pile diameter.
93	Pile Thickness	Double	Pile thickness.
94	Piles Plugged	Integer	Piles plugged?
95	Skirt Piles	Integer	Skirt piles?
96	Skirt Pile Thickness	Double	Skirt pile thickness.
97	Skirt Pile Diameter	Double	Skirt pile diameter.
98	Skirt Pile Length	Double	Skirt pile length.
99	Skirt Piles Plugged	Integer	Skirt piles plugged?
100	Number of End-on Skirt Piles	Integer	Number of end-on skirt piles.
101	Number of Broadside Skirt Piles	Integer	Number of broadside skirt piles.
102	Drag Coefficient, Jacket	Double	Drag coefficient of jacket.
103	Current Blockage	Double	Current blockage.
104	Directional Spreading	Double	Directional spreading.
105	Global Load Factor	Double	Global load factor.
106	Marine Growth	Integer	Marine growth?
108	Boatlanding Broadside Equiv.	Double	Boatlanding broadside equivalent area.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ID	Factor Name	Type	Factor Description
	Area		
109	Boatlanding End-on Equiv. Area	Double	Boatlanding end-on equivalent area.
111	Yield Stress	Double	Yield stress.
112	Elastic Modulus	Double	Elastic modulus.
113	Brace Buckling Length Factor	Double	Brace buckling length factor.
114	Brace Residual Strength Factor	Double	Brace residual strength factor.
115	Soil Type	Integer	Soil type
116	Shear Strength @ Mudline	Double	Shear strength at mudline.
117	Shear Strenght @ Pile Tip	Double	Shear strength at tip of pile.
118	Angle of Friction	Double	Angle of friction.
119	Submerged Specific Weight	Double	Submerged specific weight.
120	Scour Depth	Double	Scour depth.
121	Wave on Deck Loading Bias	Double	Wave on deck loading bias.
122	Wave on Deck Loading COV	Double	Wave on deck loading coefficient of variation.
123	Wave on Jacket Loading Bias	Double	Wave on jacket loading bias.
124	Wave on Jacket Loading COV	Double	Wave on jacket loading coefficient of variation.
125	Tubular Brace Capacity Bias	Double	Tubular brace capacity bias.
126	Tubular Brace Capacity COV	Double	Tubular brace capacity coefficient of variation.
127	Tubular Joint Capacity Bias	Double	Tubular joint capacity bias.
128	Tubular Joint Capacity COV	Double	Tubular joint capacity coefficient of variation.
129	Clay Axial Foundation Capacity Bias	Double	Clay axial foundation capacity bias.
130	Clay Axial Foundation Capacity COV	Double	Clay axial foundation capacity coefficient of variation.
131	Clay Lateral Foundation Capacity Bias	Double	Clay lateral foundation capacity bias.
132	Clay Lateral Foundation Capacity COV	Double	Clay lateral foundation capacity coefficient of variation.
133	Sand Axial Foundation Capacity Bias	Double	Sand axial foundation capacity bias.
134	Sand Axial Foundation Capacity COV	Double	Sand axial foundation capacity coefficient of variation.
135	Sand Lateral Foundation Capacity Bias	Double	Sand lateral foundation capacity bias.
136	Sand Lateral Foundation Capacity COV	Double	Sand lateral foundation capacity coefficient of variation.
137	Prior Eval	Long Integer	Type of Prior Evaluation Required for This Evaluation
139	Iterative Factor	Long Integer	Iterator – get these first, then assign values
140	Orientation	Integer	Orientation

Table 4-7: L2I1 Object Attributes, Platform Hierarchy

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ID	Factor Name	Factor Type	Factor Description
63	Bottom Elevation	Double	Bottom elevation.
64	Top Elevation	Double	Top elevation.
65	End-on Width	Double	End-on width.
66	Broadside Width	Double	Broadside width.
67	CD	Double	CD
68	CS	Double	CS

Table 4-8: L2I1 Object Attributes, Platform / Deck Hierarchy

ID	Factor Name	Factor Type	Factor Description
47	Number of Diagonal Braces	Integer	How many diagonal braces, in all vertical planes?
48	Number of Diagonal Braces: Broadside	Integer	How many diagonal braces in all vertical planes, broadside?
49	Number of Horizontal Braces	Integer	How many horizontal braces are there?
69	Bay Height	Double	Bay height.
107	Marine Growth Thickness	Double	Marine growth thickness.
110	Equiv. Diameter of Appurtenances	Double	Equivalent diameter of appurtenances.

Table 4-9: L2I1 Object Attributes, Platform / Bay Hierarchy

ID	Factor Name	Factor Type	Factor Description
72	Jacket Leg Thickness	Double	Jacket leg thickness.
73	Jacket Leg Diameter	Double	Jacket leg diameter.

Table 4-10: L2I1 Object Attributes, Platform / Orientation / Bay Hierarchy

ID	Factor Name	Factor Type	Factor Description
74	Diameter	Double	Diameter
75	Thickness	Double	Thickness
84	Length	Double	Length
85	Angle	Double	Angle

Table 4-11: L2I1 Object Attributes, Platform / Bay / Horizontal Brace Hierarchy

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ID	Factor Name	Factor Type	Factor Description
31	Grout Filled?	Integer	Grout filled?
77	Type	Integer	Type
86	Chord Diameter	Double	Chord diameter.
87	Chord Thickness	Double	Chord thickness.
88	Branch Diameter	Double	Branch diameter.
89	Gap in K-Joint	Double	Gap in K-joint.
90	Angle (Branch/Chord)	Double	Angle, branch/chord.

Table 4-12: L2I1 Object Attributes, Platform / Joint Hierarchy

ID	Factor Name	Factor Type	Factor Description
76	Position	Integer	Position
78	Configuration	Integer	Configuration
79	Joint I	Integer	Joint I
80	Joint j	Integer	Joint j
81	Condition	Integer	Condition
82	Dent Depth	Double	Dent depth.
83	Out of Straightness	Double	Out of straightness.

Table 4-13: L2I1 Object Attributes, Platform / Orientation / Bay / Diagonal Brace Hierarchy

Storage: For each instance of the L2I1 object, a unique Evaluation_ID is allocated to an existing Structure_ID. For the factors listed in Table 4-7, one value per factor is assigned to the Evaluation_ID in PIMS table FACTOR_DATA, using the default FM_ID of 1. The factors in Tables 4-8 through 4-13, however, may have more than one value assigned to them. For example, deck #1 may have a bottom elevation of 300, while deck #2 may have a bottom elevation of 330. Both bottom elevations are denoted by Factor 63; the former uses an FM_ID value of 7 while the latter uses 8 (see Section 3.3.2 for a detailed

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

explanation of the use of FM_ID). For each factor stored in DAESSIM, the combination of Evaluation_ID, FM_ID, and Factor_ID will be unique. This allows the use of FM_ID to denote the hierarchical object involved (bay, brace, etc), while Factor_ID denotes the object's attribute.

Within each level of hierarchy, each object is unlimited in extent – there is no maximum number of bays per platform, for example, and no maximum number of diagonal braces per bay.

Relationships: An L2I1 instance is a collection of input attributes for the L2RSR1, L2RSR2, and L2VMax1 instances, and is a required prerequisite to their creation. Most of the L2I1 attributes are used by the L2RSRx objects; Factor 52 is used by L2VMax1.

4.2.2.3 L2VMax1

Methodology: The L2VMax1 ("Level Two Maximum Wind Velocity, version 1") output object is based on the Level One Maximum Wind Velocity sub-routine of the California Coastal Platform Information Management System [Staneff et al 1994]. It is included solely as an example of using Excel as an input form to PIMS. The wind velocity vs. wave height curve probably should not be applied to Gulf of Mexico platforms, though it may be altered appropriately and used within another object class (e.g., L2VMax2).

An L2VMax1 evaluation develops and stores the expected maximum wind velocity (VMax), given the maximum wave height. The expected maximum wind velocity is a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

probabilistic value; an L2VMax1 evaluation also develops and stores the coefficient of variation of the maximum wind velocity (the COV reflects the user's confidence in the VMax value). The VMax chart used to determine the other two values is also stored.

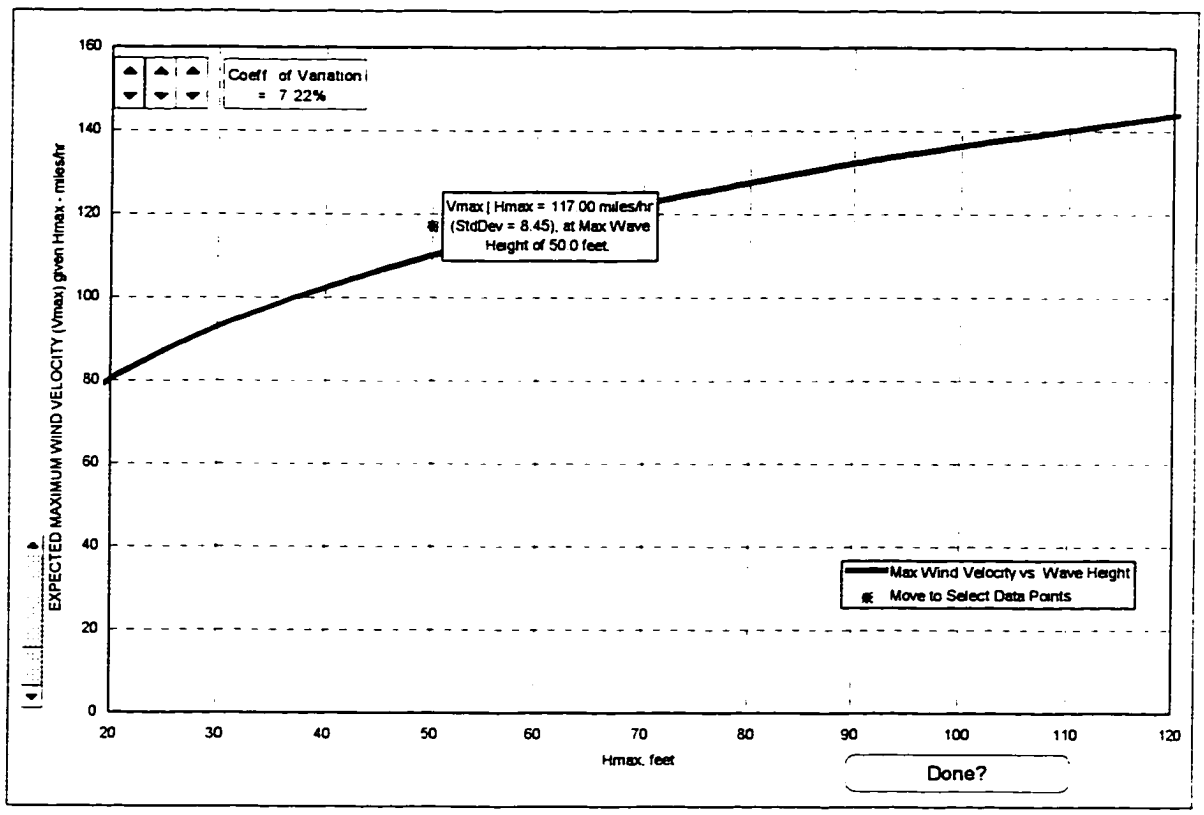


Figure 4-7: L2VMax1 Data Input Chart

Attributes: Table 4-14 lists the factors solicited and stored by an L2VMax1 evaluation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ID	Factor Name	Factor Type	Factor Description
36	VMax Chart	OLE	Picture of chart result of VMax evaluation
37	VMax	Double	Expected Maximum Wind Velocity given Expected Maximum Wave Height, L1
38	VMax COV	Double	Coeffiecient of Variation of VMax
29	ID, Prior Evaluation	Long Integer	ID of source evaluation

Table 4-14: L2VMax1 Object Attributes

Factor 29 stores the ID of an existing L2I1 inspection, which is used as input (for maximum wave height, L2I1's Factor 52). An L2VMax1 evaluation results in three outputs, Factors 36, 37, and 38. These store a picture of the L2Vmax1 chart, the value of VMax, and the coefficient of variation of VMax, respectively.

Storage: The L2VMax1 object has a single hierarchy level. The value of FM_ID for factors 36, 37, and 38 is 1; for Factor 29, it is 793.

Relationships: An L2VMax1 instance cannot exist without a corresponding L2I1 instance. By specifying an L2I1 instance to be used as an L2VMax1 input, one L2I1 attribute is assigned to the L2VMax1 evaluation.

4.2.2.4 L2Eng1

Methodology: The L2Eng1 ("Level Two Engineering Assumption, version 1") evaluation class is an example of an "engineering assumption" evaluation class, a mechanism

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

by which non-inspection, non-structural data can be accounted for in a subsequent evaluation.

ID	Factor Name	Factor Type	Factor Description
142	L2RSR1 Prelim setting	Yes/No	If "Yes," ULSLEA will size the diameter & thickness of vertical diagonal braces.

Table 4-15: L2Engl Object Attributes

Attributes: An L2Engl evaluation stores one value, the "L2RSR Prelim" setting. This setting tells an L2RSRx evaluation whether or not to calculate preliminary values according to the ULSLEA methodology, and is strictly an engineering choice.

Storage: A single-hierarchy object, L2Engl uses an FM_ID value of 1.

Relationships: An L2Engl object is a necessary prerequisite to performing an L2RSR2 evaluation (an L2RSR1 evaluation uses the value for VMax stored in L2I1 as Factor 51).

4.2.2.5 L2RSRx – L2RSR1 and L2RSR2

Methodology: There are two L2RSR*x* ("Level Two Reserve Strength Ratio, version *x*") output objects included with PIMS: L2RSR1 follows the ULSLEA 2.0 methodology [Mortazavi 1995], and L2RSR2 follows a preliminary version of the revised ULSLEA 2.1 methodology [Stear and Bea 1997]. Both are designed to provide "a simplified analytical method for (the) assessment of structural reliability of template-type offshore plat-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

forms" [Mortazavi 1995] due to storm waves. An L2RSRx evaluation requires inputs from an L2I1 inspection, a PlatformData inspection (for water depth), and an L2Eng1 inspection. In addition, an L2RSR2 evaluation requires a VMax input from an L2VMax1 evaluation.

Attributes: Table 4-16 lists the factors solicited and stored by an L2RSRx evaluation.

ID	Factor Name	Factor Type	Factor Description
141	Broadside Loading Chart	OLE	Chart, broadside loading
143	End-on Loading Chart	OLE	Chart, end-on loading
144	Wave Kinematics Chart	OLE	Chart, wave kinematics
145	Axial Pile Capacity Chart	OLE	Chart, axial pile capacity
146	Broadside Loading Reliability Chart	OLE	Chart, broadside loading reliability
147	End-on Loading Reliability Chart	OLE	Chart, end-on loading reliability
148	Pf	Double	Probability of failure
149	Pf, Foundation, Compressive (?)	Double	Probability of failure, foundation, compressive (?)
150	Pf, Foundation, Tensile (?)	Double	Probability of failure, foundation, tensive (?)
151	Pf, Upper Bound	Double	Probability of failure, upper bound (uncorrelated system)
152	Pf, Lower Bound	Double	Probability of failure, lower bound (correlated system)
29	ID, Prior Evaluation	Long Integer	ID of source evaluation
29	ID, Prior Evaluation	Long Integer	ID of source evaluation
29	ID, Prior Evaluation	Long Integer	ID of source evaluation
29	ID, Prior Evaluation	Long Integer	ID of source evaluation

Table 4-16: L2RSRx Object Attributes

Samples of the six charts (Factors 141, 143, 144, 145, 146, and 147) are shown in Figure 4-8 through Figure 4-13.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

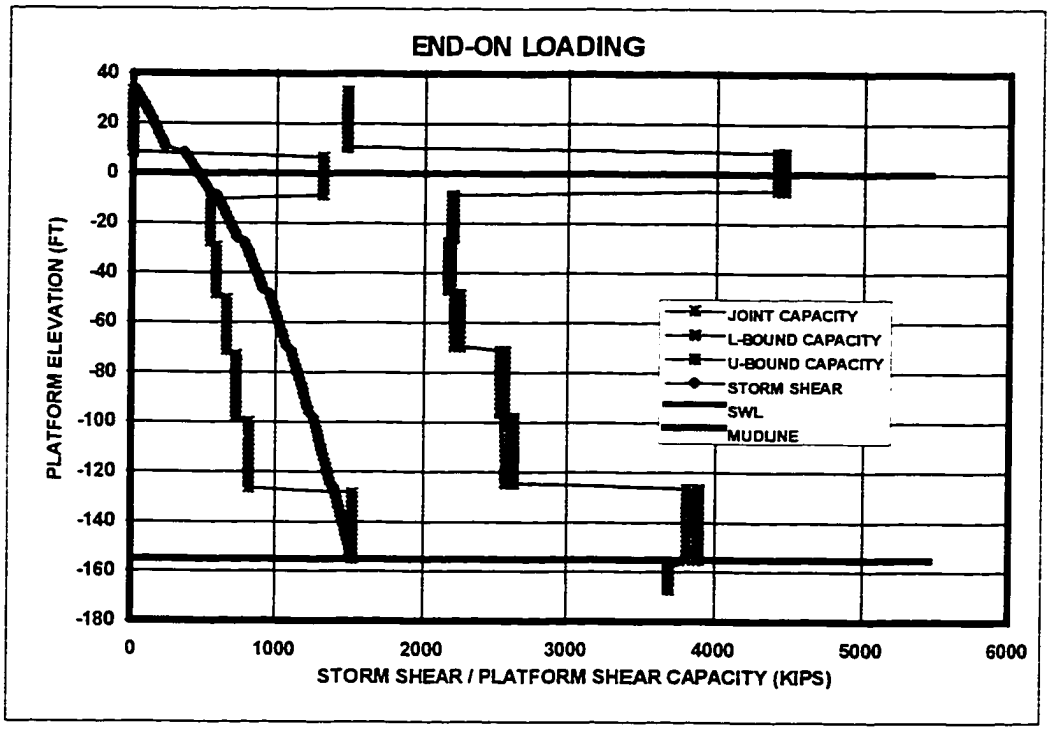


Figure 4-8: L2RSRx End-On Loading Chart

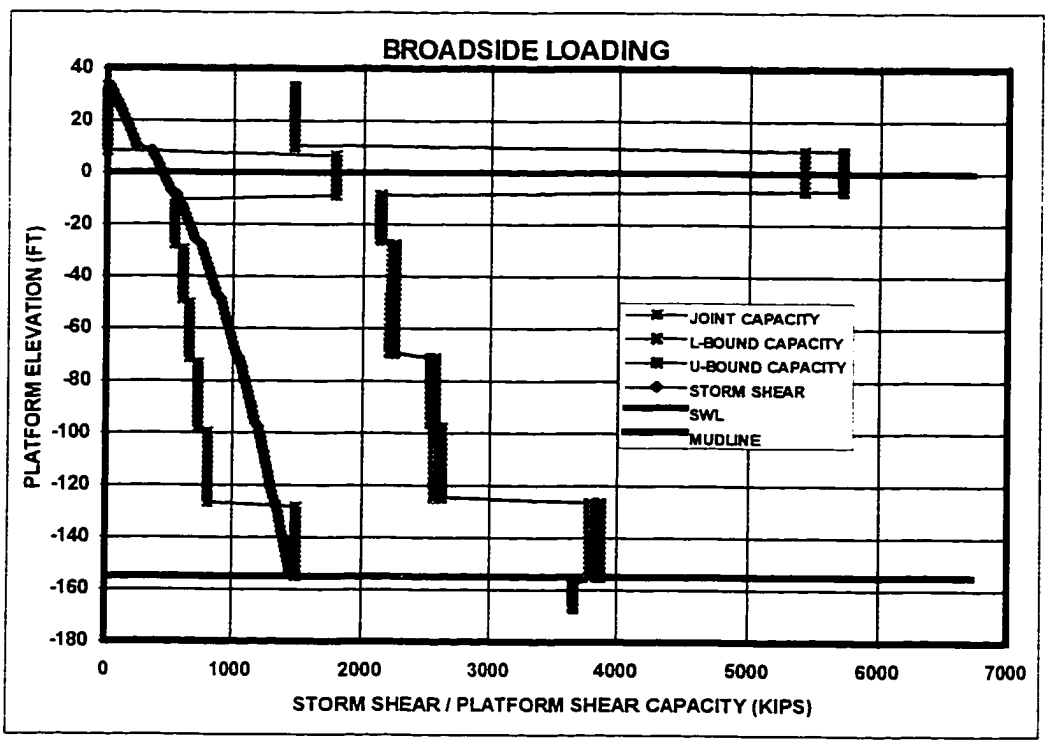


Figure 4-9: L2RSRx Broadside Loading Chart

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

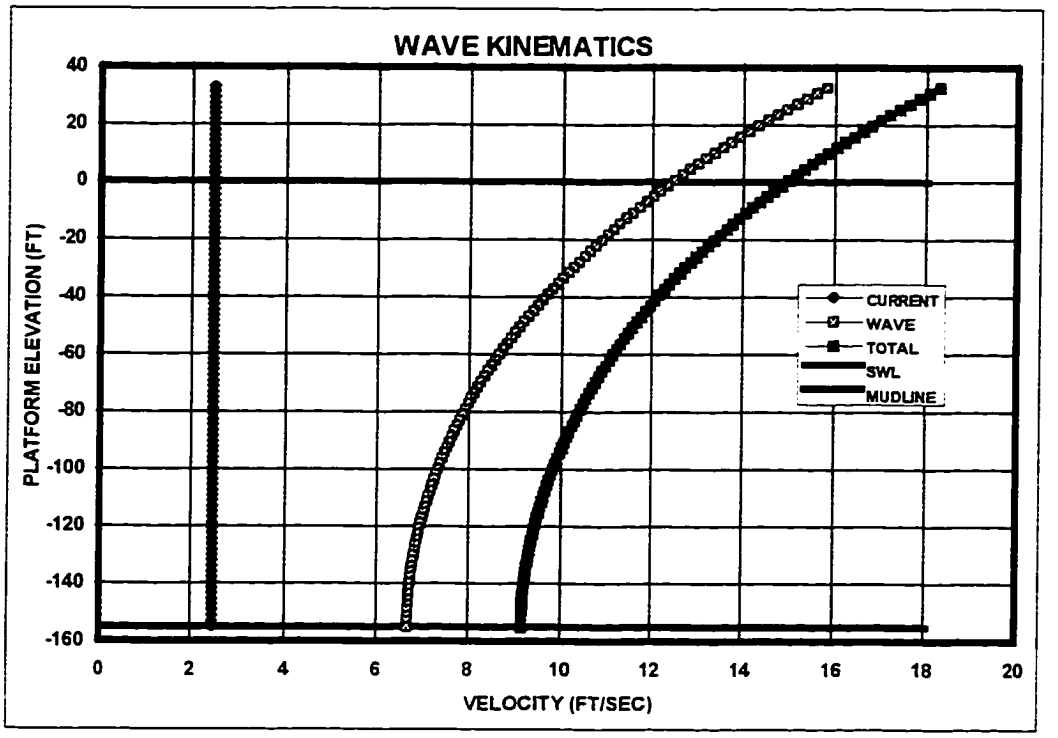


Figure 4-10: L2RSRx Wave Kinematics Chart

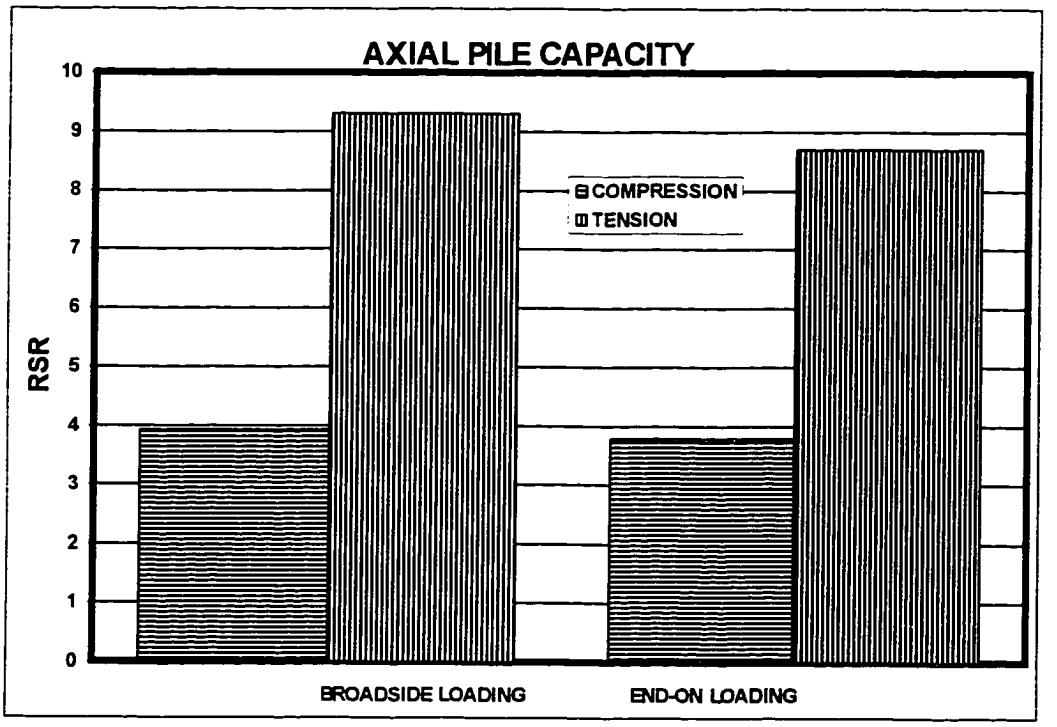


Figure 4-11: L2RSRx Axial Pile Capacity Chart

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

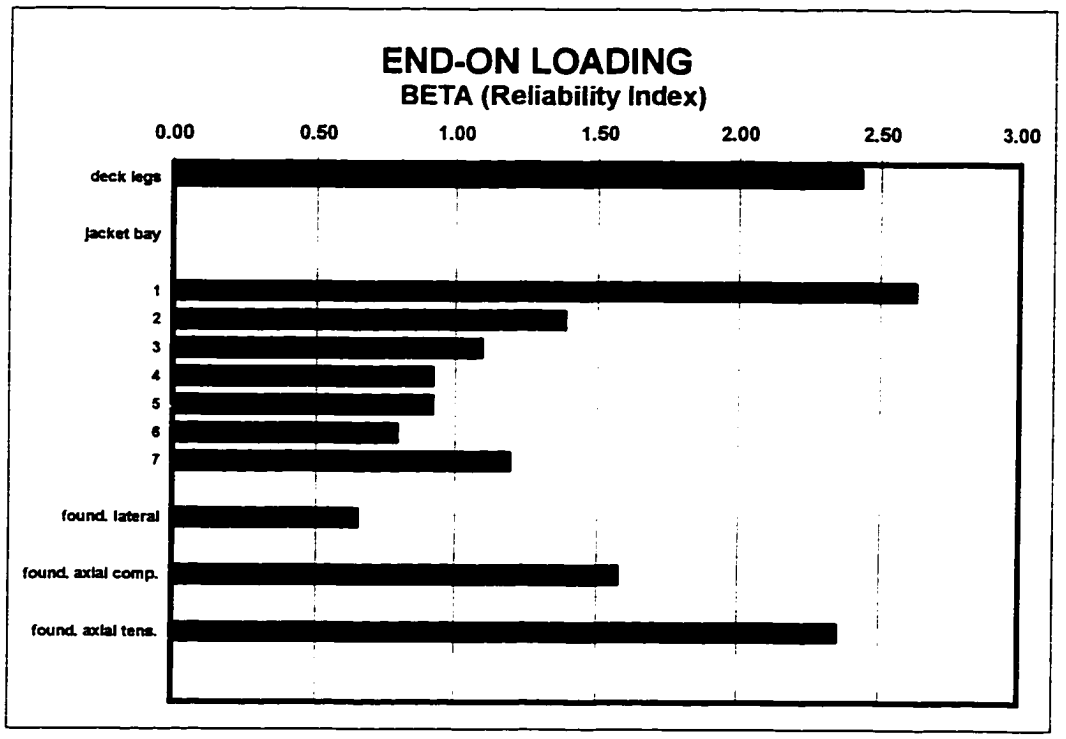


Figure 4-12: L2RSRx End-On Loading Reliability Chart

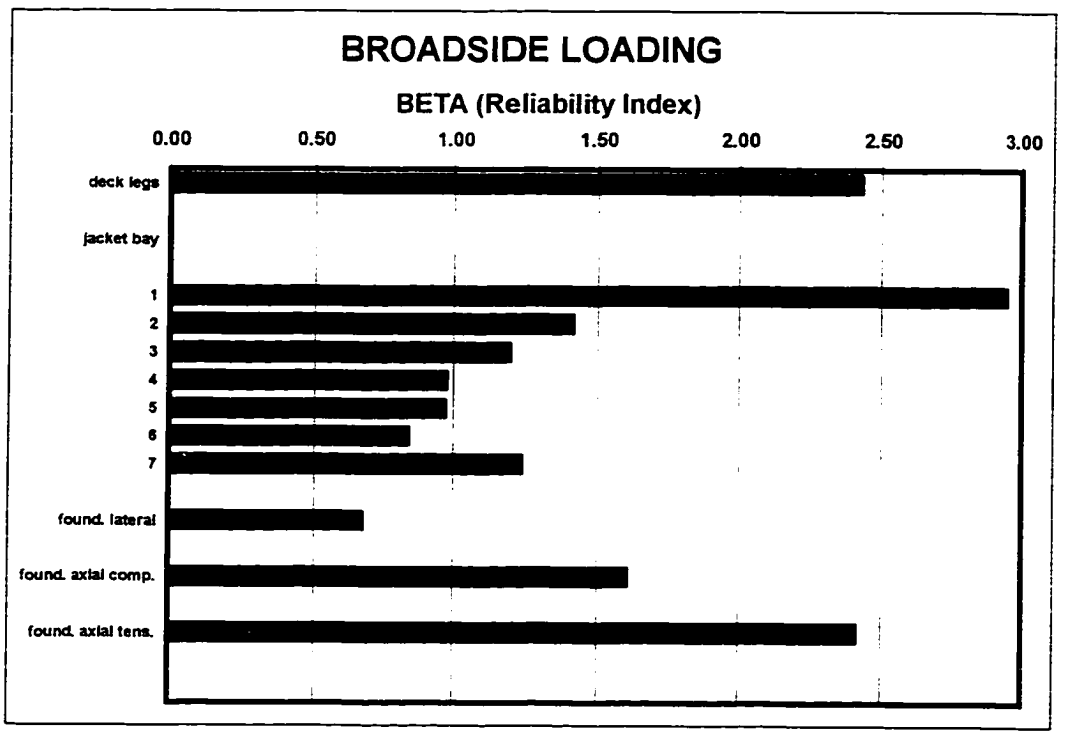


Figure 4-13: L2RSRx Broadside Loading Reliability Chart

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The probabilities of failure (Factors 148 – 152) are not part of the original ULSLEA reliability methodologies, but are derived from their respective ULSLEA beta values per the formulation:

$$P_f = 0.475e^{-(Beta^{1.6})}, Beta \geq 0 \quad [Bea, 1990]$$
$$P_f = 1 - \left(0.475e^{-(Beta^{1.6})}\right), Beta < 0 \qquad (2)$$

The lower estimate for system P_f (Factor 152) assumes that failure mechanisms are correlated, so that the system P_f is the maximum of all component probabilities of failure. The upper estimate for system P_f (Factor 151) assumes that failure mechanisms are uncorrelated, and is the sum of all component probabilities of failure [Mortazavi 1995].

Since ULSLEA calculations are conditional on wave heights, these calculated probabilities of failure are also conditional on wave height (properly, "P_f | wave height"). These probabilities are included for demonstration purposes only.

Storage: In terms of data storage, the L2RSR*x* objects have a single level of hierarchy (internally, they must process information from the L2I1 object and apply it to a multiple-hierarchy structure). FM_ID for all factors other than Factor 29 is 1. The values of Factor 29 are 793 (L2I1), 794 (L2Eng1), 1360 (PlatformData), and 1362 (L2VMax1).

Relationships: An L2RSR*x* instance cannot exist without component L2I1, L2Eng1, PlatformData, and (in the case of L2RSR2) L2VMax1 instances. Specifying the input Evaluation_ID's assigns the attributes of those evaluations. It also makes the results of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the L2RSRx evaluation directly related to the inputs into the L2I1, L2Eng1, PlatformData, and L2VMax1 instances involved.

4.3 DAESSIM VIABILITY DEMONSTRATIONS

As previously stated, the purpose behind the creation of PIMS was to demonstrate the capabilities of DAESSIM. Table 4-17 lists the desirable characteristics developed for an information management system in Chapters 2 and 3. Listed next to each characteristic is the section of this dissertation where its functionality is demonstrated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Desirable Characteristics per Chapter 2	Demonstrations of Capability
Storage and display of, and user interaction with, data and results.	Section 4.2; Chapter 6
Ability to perform either a quick database search (if possible), or a simplified review of all structures, as dictated by available historical records and analysis method.	Section 4.3.2
Ability to compare results from different structures to determine relative degree of need (fleet risk).	Sections 4.2.1.5 (graphical), 4.3.2
Ability to retrieve data for subsequent reassessment upon alteration of decision criteria or updating of analysis methodology.	Section 4.3.4, 4.3.5
Relaxation of conservatism with increasing analysis sophistication.	Section 4.3.8
Calibration of analysis results with historical record, if available.	Section 4.3.8
Modifiable decision criteria/safety standards.	Section 4.3.5
Modularity, for updating/replacing analysis procedures (technology transfer).	Section 4.3.4, 4.3.5
Ability to handle multiple aspects of safety (i.e., analyses for operational error, structural integrity, inspection management, etc.), as dictated by available analyses.	Section 4.3.8
Decision support functionality without usurping the decision maker's role.	Section 4.2.1.4, 4.2.1.5
Ability to handle biases and/or uncertainties, as dictated by available analysis methods.	Sections 4.2.2.2, 4.2.2.5
Multiple levels of failure - minor, major, collapse, etc.	Section 4.3.8
Cost/benefit economics in consequence assessment, repair alternatives.	Section 4.3.8
Desirable Characteristics per Chapter 3	**Demonstrations of Capability**
Extensibility for handling unknown and unanticipated (at design time) analysis types and structure types.	Sections 6.2, 4.2 (user-added data integrity via relational integrity)
Flexibility of handling non-standard hierarchical schemes.	Section 4.2.2.2
Non-proprietary format.	Chapter 3; Sections 4.3.1, 6.3.8 (importing data from Excel)
Data integrity.	Chapter 3; Sections 4.2 (user-added data integrity via relational integrity), 4.3.6
Ad hoc query capability.	Sections 4.3.1, 4.3.2
Referential integrity.	Chapter 3; Sections 4.2, 4.3.1
Interoperability with existing or external applications.	Sections 4.2.1.4, 4.2.1.5, 4.2.2.3, 4.2.2.5, 4.3.3.2, 4.3.5
Ability to track structure changes with time.	Section 4.3.7

Table 4-17: Demonstrations of Desired DAESSIM Characteristics

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.3.1 Demonstration of Non-Proprietary Data Format, *Ad Hoc* Query Capability, and Referential Integrity

The purpose of this exercise was to demonstrate the non-proprietary nature of DAESSIM data model, and to demonstrate DAESSIM's *ad hoc* query capability. PIMS was used for the demonstration. Data were converted from a Minerals Management Service dBASE-format file [MMS 1994] into MS Access format (through standard MS Access functions), and then converted to Structure and PlatformData instances in PIMS through *ad hoc* queries. At the start of the exercise, PIMS table STRUCTURE held 10 records; EVALUATION held 52; and FACTOR_DATA held 6,247 records. The MMS file (STRU3394.dbf) held 4710 records, each with 14 text attributes. These attributes were converted to Structures and PlatformData attributes of various data types (see Table 4-18).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

STRU3394 Attribute	PIMS Attribute	PIMS Object	PIMS Data Type
CPXID	1st part, Structure ID	Structures	Text
STRNO	2nd part, Structure ID	Structures	Text
STRNME	Structure Name	Structures	Text
SAREABLK	Area Block	PlatformData	Text
YRINST	Installed Date	PlatformData	Date
YRRMV	Removed Date	PlatformData	Date
1st part, LOCDPT	Lambert N	PlatformData	Double
2nd part, LOCDPT	Lambert E	PlatformData	Double
XSURLOC	X Coordinate	PlatformData	Double
YSURLOC	Y Coordinate	PlatformData	Double
LEASES	Lease #	PlatformData	Text
MILES	Miles To Land	PlatformData	Double
WTRD	Water Depth	PlatformData	Double
CABBRV	Owner	PlatformData	Text
MANNED	Manned	PlatformData	Yes/No

Table 4-18: Original and Converted Attributes

If the exercise was to be successful, the final record counts for the PIMS tables would be:

- STRUCTURE: 10 + 4710 = 4720
- EVALUATION: 52 + 4710 = 4762
- FACTOR_DATA: 6247 + 14*4710 = 72,187

In addition, the MMS structures would be visible through the PIMS "STRUCTURE OPERATIONS: Structure Edit/Input" form and all would have PlatformData instances available for review through the PIMS "STRUCTURE OPERATIONS: Examine Existing Evaluations" form.

The exercise proceeded as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. From the PIMS database window, dBASE file STRU3394.dbf was imported into table MMS_STRU3394 (this import capability is inherent in MS Access but varies with the particular Access installation; converters for dBASE must have been installed).

2. The suspicion that the combination of fields CPXID and STRNO would result in a unique ID was investigated: a query was created with the *ad hoc* SQL statement

   "SELECT DISTINCTROW [CPXID] & [STRNO] AS [Structure ID] FROM MMS_STRU3394 ORDER BY [CPXID] & [STRNO];"

   The result was copied into the first column of a new Excel spreadsheet. The formula "=if(a1=a2,"Not Unique","")" was entered in the second column. No values of "Not Unique" were found.

3. It was discovered that PlatformData had no "Owner" attribute. It was added through normal DAESSIM user interface features:

   a. From the Main Menu, clicking on Evaluations – Types – Review opened the "Create a New Type of Evaluation" form.

   b. Moving to the PlatformData evaluation type and clicking on "Assign Factors" opened the "Assign Factors to Evaluations" form.

   c. "PlatformData" was highlighted in the "Select an Evaluation Type" box, and then "Add Assign't" was selected.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

d. "PlatformData" was entered in the "Eval Type to Store" box, and then the "Create factors..." button was pushed.

e. "Owner" was entered in Factor Name, "Text" was selected as the factor type, "Corporate Owner" was entered as Factor Description 1, and "Possibly operator, too" was entered as Factor Description 4. "Done" was selected.

f. The new factor was selected in "Factor ID," the selection of "Input" was confirmed in "Input or Output?", and the form closed.

g. The "Create a New Type of Evaluation" form was closed.

4. Step 3 was repeated when it was discovered that no "Manned" attribute existed for PlatformData. This was a Yes/No data type with Factor Description 1 = "Manned" and Factor Description 4 = "Classification per MMS."

5. From the Main Menu, Fleets-Individuals-Add opened the "Fleet Entry" form. A new fleet was added, with Fleet_ID = 6, Fleet_Type = "Platforms", Fleet_Description = "All MMS Platforms", and Fleet_Comment = "Platforms imported from MMS STRU3394.dbf database."

6. A new structure instance was added for each new structure with the *ad hoc* SQL query,

"INSERT INTO tblStructures ([Structure ID], [Structure Name], [Structure Type], [Structure Comments]) SELECT DISTINCTROW [cpxid] & [strno] AS [Structure ID],

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Iif(IsNull([STRNME])," ",Iif([STRNME] Like
"*(removed)*",Left([STRNME],Len([STRNME])-9),Iif([STRNME] Like
"*(remove)*",Left([STRNME],Len([STRNME])-8),Iif([STRNME] Like
"*(rem)*",Left([STRNME],Len([STRNME])-5),Iif([STRNME] Like
"*(rm)*",Left([STRNME],Len([STRNME])-4),[STRNME])))))) AS [Structure Name], "Platform" AS [Structure Type], "Imported from STRU3394" AS [Structure Comments] FROM MMS_STRU3394 ORDER BY [cpxid] & [strno];"[1]

(This query also deleted the redundant "(removed)" text from Structure_Name. Removed platforms are indicated by entries in the Removed_Date attribute.)

7. A PlatformData instance was added for each structure instance:

"INSERT INTO tblEvaluations ([Evaluation Type], [Structure ID], [Fleet ID], [Evaluation End Date], Evaluator, [Evaluation Comments]) SELECT DISTINCTROW "PlatformData" AS [Evaluation Type], [MMS_STRU3394].[cpxid] & [MMS_STRU3394].[strno] AS [Structure ID], 6 AS [Fleet ID], Now() AS [Evaluation End Date], "MMS_STRU3394" AS Evaluator, "Imported from STRU3394" AS [Evaluation Comments] FROM tblEvaluations, MMS_STRU3394;"

8. An intermediate query was created and saved under the name "atmpQuery9":

"SELECT DISTINCTROW [cpxid] & [strno] AS [Structure ID], MMS_STRU3394.CPXID AS [MMS Complex ID], MMS_STRU3394.STRNO AS [MMS Structure ID], MMS_STRU3394.SAREABLK AS [Area Block], Iif(IsNull([YRINST]),Null,"19" & [YRINST]) AS [Installed Date], Iif(IsNull([YRRMV]),Null,"19" & [YRRMV]) AS [Removed Date], Iif([LOCDPT]="",0,Iif(Right(Left([LOCDPT],6),1)="S",-Left([LOCDPT],5),Iif((Right(Left([LOCDPT],6),1)="W" Or Right(Left([LOCDPT],6),1)="E"),Iif(Right([LOCDPT],1)="N",Mid([LOCDPT],7,5),-

---

[1] "tblStructures" in the SQL excerpts is the same as table "STRUCTURES." "tblEvaluations" is the same as table "EVALUATION," and "tblEvaluationFactorData" is the same as table "FACTOR_DATA." The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Mid([LOCDPT],7,5)),Left([LOCDPT],5)))) AS [Lambert N],
Iif([LOCDPT]="",0,Iif(Right([LOCDPT],1)="E",Mid([LOCDPT],7,5),Iif((Right([LOCDPT],
1)="N" Or
Right([LOCDPT],1)="S"),Iif(Right(Left([LOCDPT],6),1)="E",Left([LOCDPT],5),-
Left([LOCDPT],5)),-Mid([LOCDPT],7,5)))) AS [Lambert E], MMS_STRU3394.XSURLOC
AS [X Coordinate], MMS_STRU3394.YSURLOC AS [Y Coordinate],
MMS_STRU3394.LEASES AS [Lease #], MMS_STRU3394.MILES AS [Miles to Land],
MMS_STRU3394.WTRD AS [Water Depth], MMS_STRU3394.CABBRV AS Owner,
Iif(IsNull([MANNED]),"Y",[MANNED]) AS Manning FROM MMS_STRU3394;"

This operation converted null entries in the Manning attribute to "Yes." It also assumed that LOCDPT is a Lambert coordinate attribute, and that North and East are positive Lambert values. It did not attempt to correct apparently erroneous LOCDPT entries (e.g., two S's or one E and one W in the same field, etc.) but allowed a later query to search for them based on the criteria that if the two Lambert attributes had the same absolute value, an error probably exists.

9. PlatformData attributes were assigned through a series of *ad hoc* queries:

   a. **MMS Complex ID:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldText) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], atmpQuery9.[MMS Complex ID] AS fldText FROM (tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN (tblEvaluationFactorsAllowed INNER JOIN tblEvaluationTypeFactors ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID]) ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="MMS Complex ID"));"

"tbl" names are used in the actual DAESSIM software. They are in Leszynski/Reddick format [Jennings 1994] and were shortened and capitalized for clarity in the text.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

b. **MMS Structure ID:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldText) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], atmpQuery9.[MMS Structure ID] AS fldText FROM (tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN (tblEvaluationFactorsAllowed INNER JOIN tblEvaluationTypeFactors ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID]) ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="MMS Structure ID"));"

c. **Area Block:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldText) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], atmpQuery9.[Area Block] AS fldText FROM (tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN (tblEvaluationFactorsAllowed INNER JOIN tblEvaluationTypeFactors ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID]) ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Area Block"));"

d. **Installed Date:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldDate) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], Iif(IsNull([Installed Date]),DateSerial(1901,1,1),DateSerial([Installed Date],6,30)) AS fldDate FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Installed Date"));"

e. **Removed Date:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldDate) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], Iif(IsNull([Removed Date]),Null,DateSerial([Removed Date],6,30)) AS fldDate FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvalua-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

tionTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Removed Date"));"

f. **Lambert N:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldDouble) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], Iif(IsNull([Lambert N]),Null,CDbl([Lambert N])) AS fldDouble FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Lambert N"));"

g. **Lambert E:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldDouble) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], Iif(IsNull([Lambert E]),Null,CDbl([Lambert E])) AS fldDouble FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Lambert E"));"

h. **X Coordinate:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldDouble) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], CDbl([X Coordinate]) AS fldDouble FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="X Coordinate"));"

i. **Y Coordinate:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldDouble) SELECT DISTINCTROW tblEvalua-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

tions.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], CDbl([Y Coordinate]) AS fldDouble FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Y Coordinate"));"

j. **Lease #:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldText) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], atmpQuery9.[Lease #] AS fldText FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Lease #"));"

k. **Miles to Land:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldDouble) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], CDbl([Miles to Land]) AS fldDouble FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Miles to Land"));"

l. **Water Depth:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldDouble) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], CDbl([Water Depth]) AS fldDouble FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Water Depth"));"

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

m. **Owner:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldText) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], atmpQuery9.Owner AS fldText FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Owner"));"

n. **Manned:** "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor Modifier], [Factor ID], fldLogical) SELECT DISTINCTROW tblEvaluations.[Evaluation ID], 1 AS [Factor Modifier], tblEvaluationFactorsAllowed.[Factor ID], Iif([Manning]="N","No","Yes") AS fldLogical FROM tblEvaluationFactorsAllowed INNER JOIN ((tblEvaluations INNER JOIN atmpQuery9 ON tblEvaluations.[Structure ID] = atmpQuery9.[Structure ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluations.[Evaluation Type] = tblEvaluationTypeFactors.[Evaluation Type]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] WHERE ((tblEvaluationFactorsAllowed.[Factor Name]="Manned"));"

10. After eight queries similar to the above had been executed, an error was discovered in some of them (all of the attributes had been transferred to FACTOR_DATA.fldText, regardless of their proper data type). At this point, STRUCTURE had 4720 records, EVALUATION had 4762, and FACTOR_DATA had 43927. To test referential integrity, structure instances "1000101" and "1000201" were deleted from STRUCTURE. This had the desired effect of causing a cascade delete, resulting in a total of 4718 STRUCTURE records, 4760 EVALUATION, and 43911 FACTOR_DATA records. Deleting the rest of the MMS structures from STRUCTURE brought the record counts for all tables back to the original quantities, further demonstrating referential integrity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

11. Operations 6 through 9 were performed again (excepting step 8), using corrected queries in step 9 (as given above).

12. The query "atmpQuery9" was deleted.

13. Using another *ad hoc* query, the MMS structure instances were appended to tblFleetStructureRegister:

> "INSERT INTO tblFleetStructureRegister ([Fleet ID], [Structure ID]) SELECT DISTINCTROW 6 AS [Fleet ID], tblStructures.[Structure ID] FROM tblStructures WHERE ((tblStructures.[Structure Comments]="Imported from STRU3394"));"

14. The record count for STRUCTURE was now 4720; for EVALUATION, 4762; and for FACTOR_DATA, 72187. These counts matched expectations.

15. Evaluations-Individuals-Review was selected from the Main Menu to allow the review of PlatformData evaluations in Fleet #6. These were compared at random to a direct view of table MMS_STRU3394, and no errors were found.

16. Structures-Individuals-Review was selected from the Main Menu to allow the review of structure instances. These were compared at random to a direct view of table MMS_STRU3394, and no errors were found.

17. The temporary table MMS_STRU3394 was deleted from PIMS.

The final record counts equaled the expected record counts, and the imported Structure and PlatformData instances were visible through the normal DAESSIM interface. Thus

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the exercise successfully demonstrated *ad hoc* query capability, referential integrity (through cascade deletion), and non-proprietary data formats (through data import).

4.3.2 Demonstration of Database Searches

As a demonstration of the ability to perform a quick database search and of the ability to perform a simplified review of a fleet of structures, given appropriate available data, a likely 'first pass' screen on the MMS platforms (imported in Section 4.3.1) was performed. Platforms were ranked by their Manning, Installed_Date, and Removed_Date attributes: the *ad hoc* query selected the platforms having Manned values of "Yes" and having null Removed_Dates (i.e., the platforms were still in existence), and ranked them by ascending Installed_Date. This was performed on the original MMS data (from file STRU3394.dbf) as a control, and on the imported PIMS PlatformData instances. Table 4-19 compares the 'top' twenty platforms returned by the two.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

STRU3394			PIMS		
Installed Date	**Structure ID**	**Manned**	**Installed Date**	**Structure ID**	**Manned**
1942	2087401	Y	1942	2087401	Y
1948	2063901	Y	1948	2063901	Y
1948	2063905	Y	1948	2063905	Y
1949	2179801	Y	1949	2179801	Y
1949	2179802	Y	1949	2179802	Y
1949	2179803	Y	1949	2179803	Y
1949	2179804	Y	1949	2179804	Y
1952	2127301	Y	1952	2127301	Y
1953	2063902	Y	1953	2063902	Y
1953	2063903	Y	1953	2063903	Y
1954	2056701	Y	1954	2056701	Y
1954	2178001	Y	1954	2178001	Y
1954	2183101	Y	1954	2183101	Y
1954	2183102	Y	1954	2183102	Y
1955	2004601	Y	1955	2004601	Y
1955	2088501	Y	1955	2088501	Y
1955	2088502	Y	1955	2088502	Y
1955	2088504	Y	1955	2088504	Y
1955	2088506	Y	1955	2088506	Y
1955	2180601	Y	1955	2180601	Y

Table 4-19: Top 20 Questionable Platforms, STRU3394 vs. PIMS

The query on the original MMS data returned 1041 records while the query set on PIMS returned 1043. This was because structures 2389101 and 1055701 had null "Manning" attributes in the original data. These were interpreted as "Yes" values when imported. Otherwise, the queries returned identical information. Thus, the PIMS database search was successful, and returned a useful simplified screen of a fleet of structures.

The query used on STRU3394 was:

SELECT DISTINCTROW '19' & [YRINST] AS [Installed Date], [CPXID] & [STRNO] AS [Structure ID], MMS_STRU3394.MANNED FROM MMS_STRU3394 WHERE (((''19' &

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[YRINST])>1901) AND (MMS_STRU3394.MANNED="Y") AND (MMS_STRU3394.YRRMV Is Null)) ORDER BY '19' & [YRINST], [CPXID] & [STRNO];

The secession of queries used on PIMS was:

Query c4qry1: SELECT DISTINCTROW tblStructures.[Structure ID], tblEvaluations.[Fleet ID], tblEvaluationFactorData.[Evaluation ID], tblEvaluationTypes.[Evaluation Type], tblEvaluationFactorsAllowed.[Factor Name], tblEvaluationFactorData.fldDate FROM tblEvaluationTypes INNER JOIN (tblStructures INNER JOIN (tblEvaluationFactorsAllowed INNER JOIN (tblEvaluations INNER JOIN tblEvaluationFactorData ON tblEvaluations.[Evaluation ID] = tblEvaluationFactorData.[Evaluation ID]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationFactorData.[Factor ID]) ON tblStructures.[Structure ID] = tblEvaluations.[Structure ID]) ON tblEvaluationTypes.[Evaluation Type] = tblEvaluations.[Evaluation Type] WHERE ((tblEvaluations.[Fleet ID]="6") AND (tblEvaluationTypes.[Evaluation Type]="PlatformData") AND (tblEvaluationFactorsAllowed.[Factor Name]="Installed Date") AND (tblEvaluationFactorData.fldDate>#01/1/01#)) ORDER BY tblEvaluationFactorData.fldDate;

Query c4qry4: SELECT DISTINCTROW tblStructures.[Structure ID], tblEvaluations.[Fleet ID], tblEvaluations.[Evaluation ID], tblEvaluationTypes.[Evaluation Type], tblEvaluationFactorsAllowed.[Factor Name], tblEvaluationFactorData.fldDate FROM tblEvaluationTypes INNER JOIN (tblStructures INNER JOIN (tblEvaluationFactorsAllowed INNER JOIN (tblEvaluations INNER JOIN tblEvaluationFactorData ON tblEvaluations.[Evaluation ID] = tblEvaluationFactorData.[Evaluation ID]) ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationFactorData.[Factor ID]) ON tblStructures.[Structure ID] = tblEvaluations.[Structure ID]) ON tblEvaluationTypes.[Evaluation Type] = tblEvaluations.[Evaluation Type] WHERE ((tblEvaluations.[Fleet ID]="6") AND (tblEvaluationTypes.[Evaluation Type]="PlatformData") AND (tblEvaluationFactorsAllowed.[Factor Name]="Removed Date")) ORDER BY tblEvaluationFactorData.fldDate;

Query c4qry6: SELECT DISTINCTROW tblStructures.[Structure ID], tblEvaluations.[Fleet ID], tblEvaluationFactorData.[Evaluation ID], tblEvaluationTypes.[Evaluation Type], tblEvaluationFactorsAllowed.[Factor Name], tblEvaluationFactorData.fldLogical FROM tblEvaluationTypes INNER JOIN (tblStructures INNER JOIN (tblEvaluationFactorsAllowed INNER JOIN (tblEvaluations INNER JOIN tblEvaluationFactorData ON tblEvaluations.[Evaluation ID] = tblEvaluationFactorData.[Evaluation ID]) ON tblEvaluationFactor-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

sAllowed.[Factor ID] = tblEvaluationFactorData.[Factor ID]) ON tblStructures.[Structure ID] = tblEvaluations.[Structure ID]) ON tblEvaluationTypes.[Evaluation Type] = tblEvaluations.[Evaluation Type] WHERE ((tblEvaluations.[Fleet ID]="6") AND (tblEvaluationTypes.[Evaluation Type]="PlatformData") AND (tblEvaluationFactorsAllowed.[Factor Name]="Manned")) ORDER BY tblEvaluationFactorData.fldLogical;

And finally, query c4qry5: SELECT DISTINCTROW Year([c4qry1].[fldDate]) AS [Installed Date], c4qry1.[Structure ID], Iif([c4qry6].[fldLogical]=Yes,"Y","N") AS Manned FROM (c4qry1 INNER JOIN c4qry4 ON c4qry1.[Evaluation ID] = c4qry4.[Evaluation ID]) INNER JOIN c4qry6 ON c4qry1.[Evaluation ID] = c4qry6.[Evaluation ID] WHERE (((Iif([c4qry6].[fldLogical]=Yes,"Y","N"))="Y") AND (c4qry4.fldDate Is Null)) ORDER BY Year([c4qry1].[fldDate]), c4qry1.[Structure ID];

4.3.3 Demonstration of Iterative Capabilities

DAESSIM is capable of performing iterative analyses. This allows variables to be changed so that their effects can be examined. Two examples are given here: one using L1C1, in which an input variable is changed; and another using L2RSR1, in which an engineering assumption is changed.

4.3.3.1 L1C1 Iteration Example

In the PIMS sample application, platform Test 2 from Fleet 2 was selected and a new L1I1 inspection instance created for it (this was assigned Evaluation_ID 4981 by PIMS). The values entered for structural attributes R1 through S4 are not pertinent to this example; the values entered for consequence attributes C1_L1 through C1_E6 (factors 11 through 24) are shown in Table 4-20 below.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ID	Factor Name	Factor Description	Value
18	C1_E1	Is the production level significant?	No
19	C1_E2	Is the platform multi-functional?	No
20	C1_E3	Is it connected to other platforms?	No
21	C1_E4	Will the operation of other platforms be significantly affected?	No
22	C1_E5	Will contractual obligations be affected by loss of the platform?	No
23	C1_E6	Will the platform be costly to replace?	No
11	C1_L1	Is the platform permanently manned?	No
12	C1_L2	Is the platform temporarily manned?	Yes
24	C1_L3	Is an evacuation system provided?	Yes
13	C1_S1	Is crude stored on the deck?	No
14	C1_S2	Does the platform have oil wells?	Yes
15	C1_S3	Do the wells have functioning SSSVs?	Yes
16	C1_S4	Are any risers connected to the platform?	Yes
17	C1_S5	Do the risers have functioning ESD valves?	No

Table 4-20: L1I1 Input Values, Example 4.6.1

When an L1C1 consequence evaluation (Evaluation_ID 4982) was run using this input data, the categorical results were:

- Life: "Loss of life is likely to be very low."
- Economics: "Resource loss is likely to be low and property loss is likely to be very low to low."
- Spillage: "Likelihood of environmental pollution is moderate to very high."

Given relatively good results in the first two categories, it was desired to improve the third category. Another inspection was produced for the platform (Evaluation_ID 4983), using a value of "Yes" for factor C1_S5 – i.e., simulating the addition of emergency shut-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

down valves to the platform's risers. When an L1C1 evaluation (#4984) was run using the new inspection, the spillage results improved while the other two results were unchanged:

- Life: "Loss of life is likely to be very low."
- Economics: "Resource loss is likely to be low and property loss is likely to be very low to low."
- Spillage: "Likelihood of environmental pollution is low."

Performing iterations by changing inspection data aids an engineer in asking, "what can be done to improve the situation?"; i.e., in performing a "what if" analysis. However, it is important to distinguish real inspection (and dependent analysis) results from speculation. Therefore, the user must be certain to make appropriate comments in the Description and/or Comments fields of the evaluation input forms (such comments then become the value of the evaluations' Description and/or Comments attributes).

4.3.3.2 L2RSR1 Iteration Example

In the PIMS sample application, a new L2RSR1 evaluation instance was created (Evaluation_ID 5021) for platform Test 1 in Fleet 2. Inspection L2I1 #165, L2Eng1 #70, and PlatformData #208 were chosen as input evaluations. L2Eng1 #70 specifies an UL-SLEA Preliminary Design attribute of "yes" for platform Test 1.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Subsequently, another L2RSR1 evaluation instance (#5022) was created for platform Test 1, also using L2I1 #165 and PlatformData #208; however, L2Eng1 #76 was specified for a Preliminary Design setting of "no." Thus the same inspection data and platform data were used for each L2RSR1 evaluation, with the only difference being the user's setting for the evaluations' assumptions. The differences in the results may be seen in the Broadside Loading charts, Figures 4-14 and 4-15.

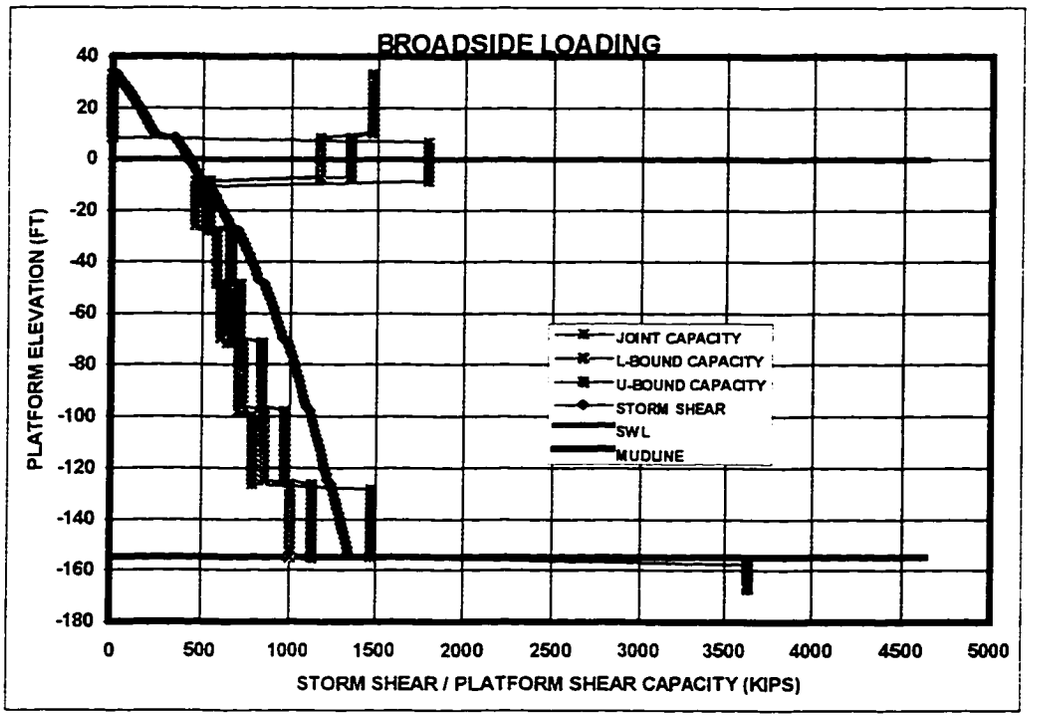


Figure 4-14: Broadside Loading Chart, Evaluation #5021, Prelim Design = "Yes"

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

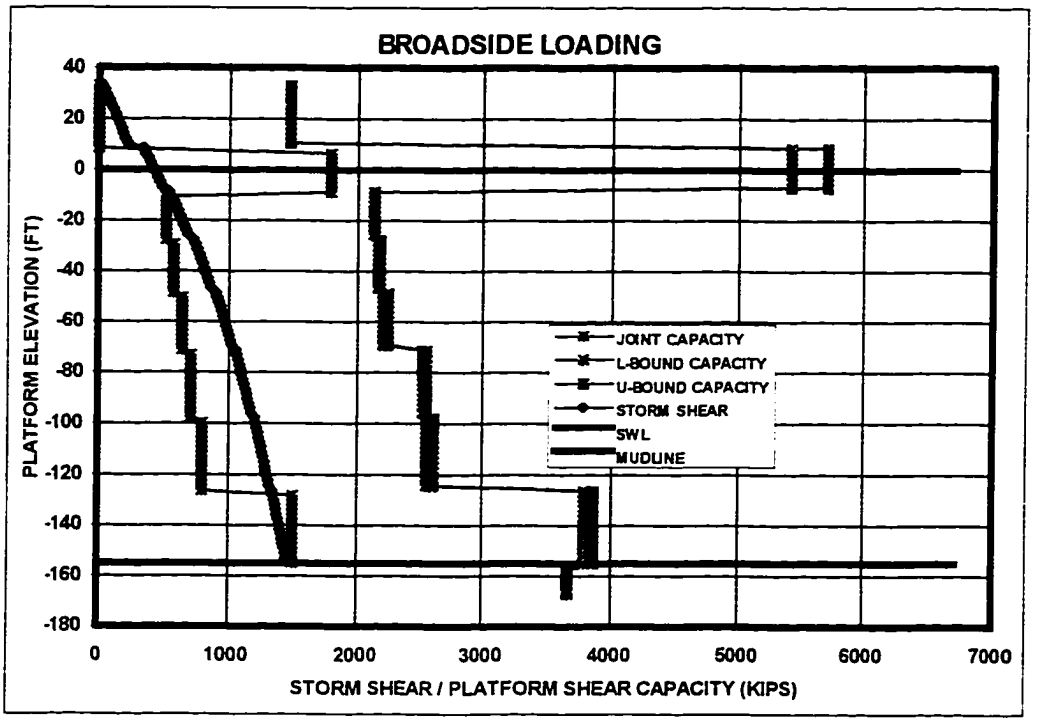


Figure 4-15: Broadside Loading Charts, Evaluation #5022, Prelim Design = "No"

4.3.4 Demonstration of Technology Transfer

Once a DAESSIM instance such as PIMS is implemented with working evaluation objects, updated methodologies for those evaluation objects may be produced or released (by, for example, the University of California). Updates may be incorporated into the DAESSIM instance in one of two ways: 1) by modifying an existing object (which may or may not then be deleted), or 2) by directly inserting a ready-for-DAESSIM object.

An example of the former method occurred during the development of PIMS. The existing ULSLEA methodology had been incorporated into PIMS (as the L2RSR1 evaluation object) on September 1, 1996. This was based on ULSLEA 2.0 [Mortazavi 1995].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Later, in October 1996, a pre-release version of an ULSLEA revision became available. By copying the L2RSR1 code, replacing all "L2RSR1" text with "L2RSR2", and altering the specific code that had been affected in the ULSLEA revision, the code for the L2RSR2 object was created. The object's creation was then finalized by registering it with PIMS in the manner described in Section 6.2.2.

An alternative to this is for the supplier of the updated methodology to supply a text file to the user that is ready to be copied into DAESSIM. This would then be copied into a new Access code module by the user and registered with DAESSIM (per Section 6.2.2), after which it would be ready for use.

4.3.5 Demonstration of Altered Decision Criteria

The ability to assess risk according to various safety criteria is a key tool of policy analysis. To DAESSIM, however, it is just a matter of providing alternate methodologies.

For example, an L1Risk1_Fleet1 evaluation was performed in PIMS on Fleet 2. The L1Risk1_Fleet1 evaluation uses the following risk guidelines:

- Acceptable: $C = 2.571 \cdot (RSR) - 2.7125$
- Marginal: $C = 2.571 \cdot (RSR) - 0.14275$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The graphical result of the evaluation (#269) is shown in Figure 4-16. This evaluation used as inputs the L1C1 evaluations 3, 90, 92, and 93, and the L1RSR1 evaluations 2, 94, 96, and 97. Clearly, three of the four platforms are decidedly in the "unacceptable" range. A decision-maker may decide to begin further work with any of the platforms shown, but might be inclined to begin with the second or third from the left (platform labels are unfortunately unavailable in the results graph).

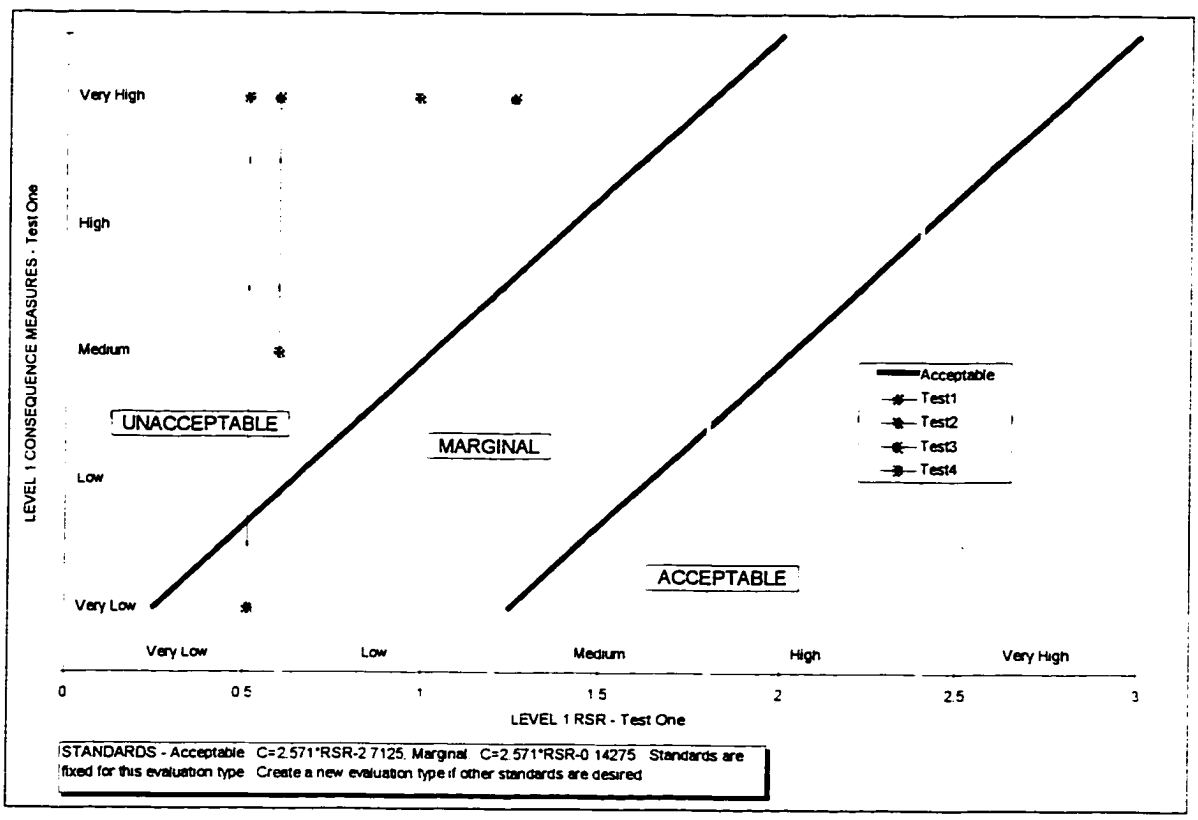


Figure 4-16: L1Risk1_Fleet1 Result, Evaluation #269

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To determine the impact of different safety standards on any decision that might be made, another evaluation object was added to PIMS using those standards. In this case, the desired standards were:

- Acceptable: $C = 1.1 \cdot (RSR) + 0.5$

- Marginal: $C = 0.6 \cdot (RSR) + 2.5$

The first step was to open (while holding the <shift> key to deactivate the opening macro) the Excel file (L1FR3.xls) used to produce the risk chart. The file was saved as L1FR5.xls and cells M2 through O5 on worksheet "Inputs and Outputs" were changed to produce the new acceptability standards. The worksheet was then saved and closed.

Next, in PIMS, the evalPlatformFleetL1Risk1 module was opened and copied to a new module, evalPlatformFleetL1Risk2. All "Risk1_Fleet1" text was replaced with "Risk1_Fleet2," and the module was compiled, saved, and closed.

From the Main Menu, Scenarios-Types-Review was selected, and all information for the existing L1Risk1_Fleet1 scenario was noted. The "Create a New Type of Scenario" form was then advanced to a new record, and information for the L1Risk1_Fleet2 scenario was entered (based on that for L1Risk1_Fleet1). The same factors were assigned to L1Risk1_Fleet2 as to L1Risk1_Fleet1.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

With setup complete, the L1Risk1_Fleet2 scenario was run for Fleet 2 using the same input evaluations as in evaluation #269: L1C1 evaluations 3, 90, 92, and 93, and L1RSR1 evaluations 2, 94, 96, and 97. This resulted in evaluation #4998, and the chart shown in Figure 4-17.

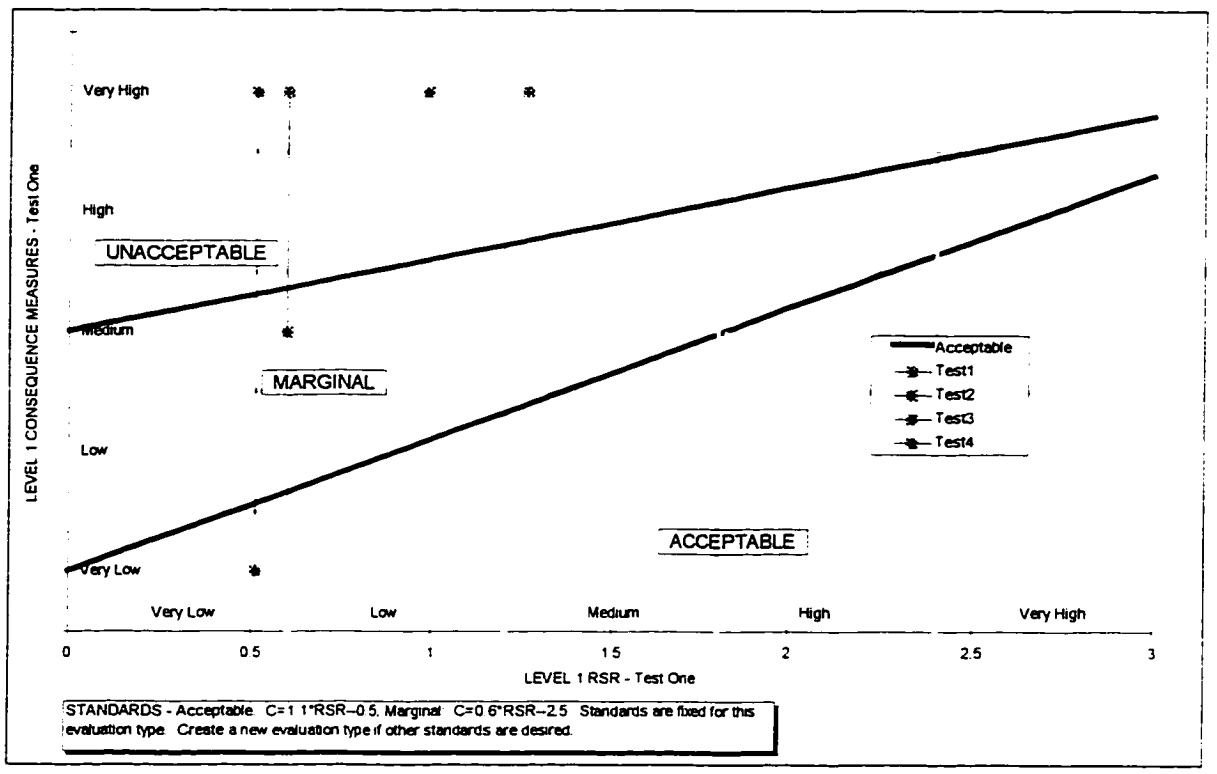


Figure 4-17: L1Risk1_Fleet2 Result, Evaluation #4998

Using the standards of the L1Risk1_Fleet2 evaluation, further work might begin with the third platform from the left instead of the second.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.3.6 Demonstration of Data Integrity

As a demonstration of data integrity, L2RSR2 (the most complicated evaluation type in PIMS) was run on three platforms (known herein as A, B, and C). The results of these evaluations were then compared with the results of evaluations performed directly in ULSLEA (using a preliminary version of ULSLEA 2.1). The comparisons were based not on the ULSLEA output charts, but on the data which constitute those charts. Those data were intercepted during the course of the analyses.

Platform	MAX ERROR (AGAINST ULSLEA)	Average Error	Variance in Error
A	0.011%	-0.00034%	0.000000000
B	0.905%	0.00116%	0.000000077
C	0.084%	0.00184%	0.000000016

Table 4-21: ULSLEA 2.1 (preliminary version) vs. L2RSR2 Results

The relatively high maximum error for platform B was due to rounding in the axial foundation calculations.

4.3.7 Tracking Changes Over Time

To demonstrate the ability of DAESSIM to track changes to structures over time, two PlatformData evaluations were created (#5007 and #5008) for platform Test 3 in Fleet 2 (an imaginary structure). The date for evaluation #5007 was given as 1/15/40, while the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

date for evaluation #5008 was given as 3/22/97. Factors 159, 160, 163, 165, 166, 180, and 189 were changed in the two evaluations. The following *ad hoc* query was run:

```
SELECT DISTINCTROW tblEvaluations.[Evaluation Begin Date], tblEvaluationFactor-
Data.[Factor ID], tblEvaluationFactorsAllowed.[Factor Name], tblEvaluationFactor-
Data.[Evaluation ID], tblEvaluationFactorData.fldDate, tblEvaluationFactorData.fldInteger,
tblEvaluationFactorData.fldLong, tblEvaluationFactorData.fldSingle, tblEvaluationFactor-
Data.fldDouble, tblEvaluationFactorData.fldText, tblEvaluationFactorData.fldLogical, Find-
FactorValueOnType([Factor
Type],[fldcurrency],[flddate],[fldmemo],[fldbyte],[fldinteger],[fldlong],[fldsingle],[flddouble]
,[fldole],[fldtext],[fldlogical],[Low Value],[High Value]) AS [Value], tblEvaluationFactor-
sAllowed.[Factor Type], tblEvaluationFactorsAllowed.[Low Value], tblEvaluationFactor-
sAllowed.[High Value] FROM tblEvaluations INNER JOIN (tblEvaluationFactorsAllowed
INNER JOIN tblEvaluationFactorData ON tblEvaluationFactorsAllowed.[Factor ID] = tblE-
valuationFactorData.[Factor ID]) ON tblEvaluations.[Evaluation ID] = tblEvaluationFactor-
Data.[Evaluation ID] WHERE ((Not tblEvaluationFactorData.[Factor ID]=157 And Not
tblEvaluationFactorData.[Factor ID]=158 And Not tblEvaluationFactorData.[Factor ID]=161
And Not tblEvaluationFactorData.[Factor ID]=162 And Not (tblEvaluationFactorData.[Factor
ID]>166 And tblEvaluationFactorData.[Factor ID]<188)) AND (tblEvaluationFactor-
Data.[Evaluation ID]=5007 Or tblEvaluationFactorData.[Evaluation ID]=5008)) ORDER BY
tblEvaluations.[Evaluation Begin Date], tblEvaluationFactorData.[Factor ID], tblEvaluation-
FactorData.[Evaluation ID];
```

This resulted in the following table:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Evaluation Begin Date	Evaluation ID	Factor ID	Factor Name	Value
1/15/40	5007	159	Lambert N	11111
1/15/40	5007	160	Lambert E	22222
1/15/40	5007	163	Slots	2
1/15/40	5007	165	Water Depth	15
1/15/40	5007	166	Miles to Land	.5
1/15/40	5007	188	Owner	ABC PRODUCTION COMPANY
1/15/40	5007	189	Manned	No
3/22/97	5008	159	Lambert N	33333
3/22/97	5008	160	Lambert E	44444
3/22/97	5008	163	Slots	15
3/22/97	5008	165	Water Depth	130
3/22/97	5008	166	Miles to Land	1.2
3/22/97	5008	188	Owner	XYZ PRODUCTION COMPANY
3/22/97	5008	189	Manned	Yes

Table 4-22: Changes in PlatformData Over Time

Clearly, the platform is in a new location, it is supporting more slots, it is in much deeper water, it is farther from land, it has new owners, and it is now manned.

4.3.8 Other DAESSIM Capabilities

Some of the desired characteristics discussed in Chapter 2 are actually characteristics of analyses or sets of analyses, rather than characteristics of DAESSIM itself. These are:

- Relaxation of conservatism with increasing analysis sophistication.

- Multiple levels of failure - minor, major, collapse, etc.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Cost/benefit economics in consequence assessment and/or repair alternatives.

Given that the right analyses are placed in DAESSIM (in the form of evaluation objects), all of these items can be characteristics of the specific DAESSIM instance. The actual creation of such DAESSIM instances, or simply the creation of the necessary analyses that can later be imported into a DAESSIM instance such as PIMS, is a matter of future work.

Similarly, demonstrating the calibration of analysis results with historical records (e.g., "how many L1Risk1 "unacceptable" platforms have failed?") is also a matter of future work. While the actual execution can easily be handled by an *ad hoc* query, insufficient numbers of analysis results exist for meaningful answers (at least for the sample evaluations included in PIMS).

4.4 SUMMARY

This chapter introduces an instance of the DAESSIM object called the Platform Information Management System (PIMS). PIMS was created to demonstrate the capabilities of DAESSIM. The methodologies, attributes, storage protocols, and interrelationships of the sample evaluation objects included in PIMS are discussed in the context of two major types of "information flow" – 1) the information flow leading to a subjective assessment of the structural risk of an oil platform due to storm waves, and 2) that leading to an objective assessment of the reliability of an oil platform due to storm waves. These information flows are used to provide insight into the structure of DAESSIM.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Individual DAESSIM characteristics, listed in Chapters 2 and 3 as desirable in an infrastructure information management system, are demonstrated with PIMS and summarized in Table 4-17. These characteristics are:

- Storage and display of, and user interaction with, data and results.
- Ability to perform either a quick database search (if possible), or a simplified review of all structures, as dictated by available historical records and analysis method.
- Ability to compare results from different structures to determine relative degree of need (fleet risk).
- Ability to retrieve data for subsequent reassessment upon alteration of decision criteria or updating of analysis methodology.
- Relaxation of conservatism with increasing analysis sophistication (possible if appropriate evaluations are available).
- Calibration of analysis results with historical record, if available.
- Modifiable decision criteria/safety standards.
- Modularity, for updating/replacing analysis procedures (technology transfer).
- Ability to handle multiple aspects of safety (i.e., analyses for operational error, structural integrity, inspection management, etc.), as dictated by available analyses.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Decision support functionality without usurping the decision maker's role.

- Ability to handle biases and/or uncertainties, as dictated by available analysis methods.

- Multiple levels of failure - minor, major, collapse, etc.

- Cost/benefit economics in consequence assessment, repair alternatives (possible if appropriate evaluations are available).

- Extensibility for handling unknown and unanticipated (at design time) analysis types and structure types.

- Flexibility of handling non-standard hierarchical schemes.

- Non-proprietary format.

- Data integrity.

- Ad hoc query capability.

- Referential integrity.

- Interoperability with existing or external applications.

- Ability to track structure changes with time.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5 SCREENING CYCLE DECISION MODELING FOR INFRASTRUCTURE FLEETS

5.1 BACKGROUND

An infrastructure screening system is a test (analysis) or sequence of tests designed to quickly and efficiently find high-risk structures. Screening system test sequences are generally set up in levels. The test at the first level is relatively easy and inexpensive to perform, i.e., "quick and dirty." Structures judged as "safe" or "acceptable" are "screened out," and only those judged "unsafe" remain in the screening system. (Throughout this chapter, a test result is referred to in quotes, while an actual condition is expressed without quotes.) Subsequent levels of tests are incrementally more expensive and more complex.

There are two general types of screening systems:

Structure-oriented screening systems – This is the more common approach. Starting with the selection of a structure to analyze, a structure-oriented system sticks with that structure until it is deemed "safe" or decisively "unsafe." Then the system moves on to the next structure in the pool (or "fleet"). Examples of this philosophy are Okada and Bresler [1976], Aggarwal [1991], Bea and Craig [1993], and API [1994]. Unfortunately, if screening funds (or time) are exhausted before the entire pool of structures can be examined, some unsafe structures may not be screened.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Fleet-oriented screening systems – In a fleet-oriented system, no (i+1)-level tests are performed until after the i-level test is completed for all structures in the fleet. Examples of this philosophy are McCann et al [1985] and Thurston [1986]. A disadvantage is that lower-level tests that are obviously required may be delayed pending the completion of upper-level tests. For example, the first level of Thurston [1986] depends on seismic mapping. It certain situations, it may be more efficient to proceed to detailed analyses on structures with obvious risk factors instead of waiting for seismic maps to be completed and analyzed for all the structures in the pool.

It is desirable to combine the features of structure-oriented and fleet-oriented screening systems. To do so, it is necessary to find criteria on which the decision to analyze a particular structure out of turn may be based.

Possible criteria for decisions in a screening system include:

- Common sense – common sense should be accommodated. The user may have foreknowledge that a given structure is in need of a detailed test, or of circumstances unforeseen by the system (such as a pending governmental review of a certain subgroup of structures).

- Economics – costs to be considered by a screening system are failure costs, testing costs, repair costs, and investment costs (in the case of multi-year lags between screens).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A screening cycle accounting for these criteria is described in this chapter. Loosely based on the Aggarwal [1991] system, it is meant to demonstrate the flexibility of DAESSIM in assimilating information for decision-making: information that is originally entered for a Level One inspection is used to generate Level One safety "acceptability" results, which may in turn be used within the screening cycle. An analogous process may be created for each level of the four-level screening cycle once appropriate tests are made available.

The principles of the screening cycle discussed herein are applicable to many types of structures, not just offshore oil platforms, and are independent of the tests that may actually be used in any given situation. The terms "Level One" through "Level Four" ("L1" through "L4") are meant to indicate levels of screening, and should not be confused with the American Petroleum Institute's Level I through Level IV inspections (API 1991).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

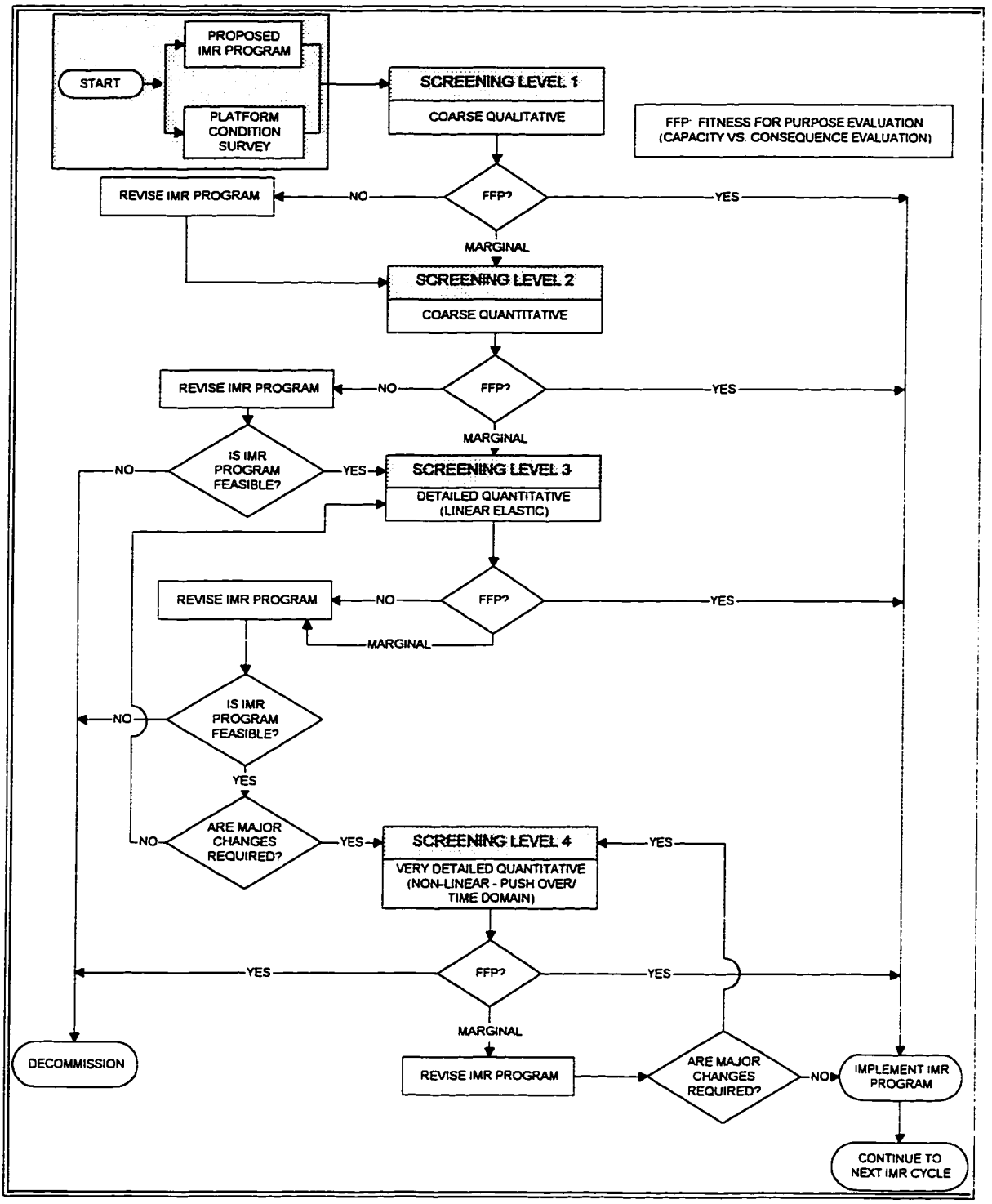


Figure 5-1: 4-Level Screening Cycle [Aggarwal 1991]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Aggarwal's four-level screening cycle for offshore oil platforms is shown in Figure 5-1. The term 'FFP' indicates a Fitness For Purpose evaluation, essentially an assessment of whether or not the platform is safe. The term 'IMR program' means an Inspection, Maintenance, and Repair program for an individual structure. The tests specified at 'Screening Level 3' and 'Screening Level 4' are specific structural analysis tests for platforms; those specified at 'Screening Level 1' and 'Screening Level 2' were planned and/or proposed by Aggarwal and have evolved into the evaluation types included in PIMS as 'L1Risk1' (Section 4.2.1.4) and 'L2Risk1' (Section 6.3.15).

Aggarwal's screening cycle can be simplified to that of Bea and Craig [1993], shown in Figure 5-2. Implicit in Figure 2 is the ability for the assessor to use any of the tests described by Aggarwal, and to iterate as required.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

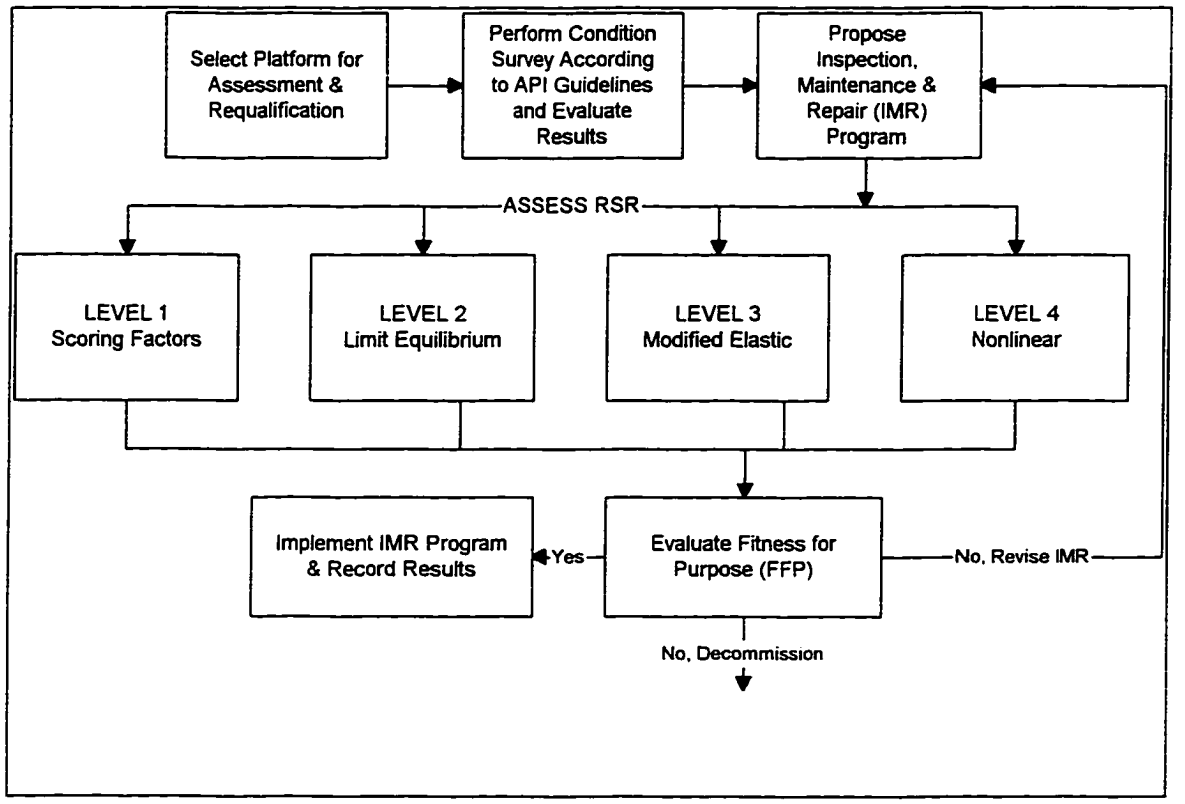


Figure 5-2: 4-Level Screening Cycle [Bea and Craig 1993]

Figure 5-3 illustrates some of the possible reasoning behind the choice to move to a next level of testing or to iterate at the present level. For the rest of this chapter, it is assumed that tests become increasingly expensive, complex, and accurate as they advance in level (i.e., a Level Three test is more expensive, more complex, and more accurate than a Level One test). Tests at Levels One, Two and Three are assumed to have three possible results ("FFP," "Marginal," and "not FFP," or "Safe," "Marginal," and "Unsafe"). The screening cycles discussed are not specific to specific types of tests, or to specific types of structures.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

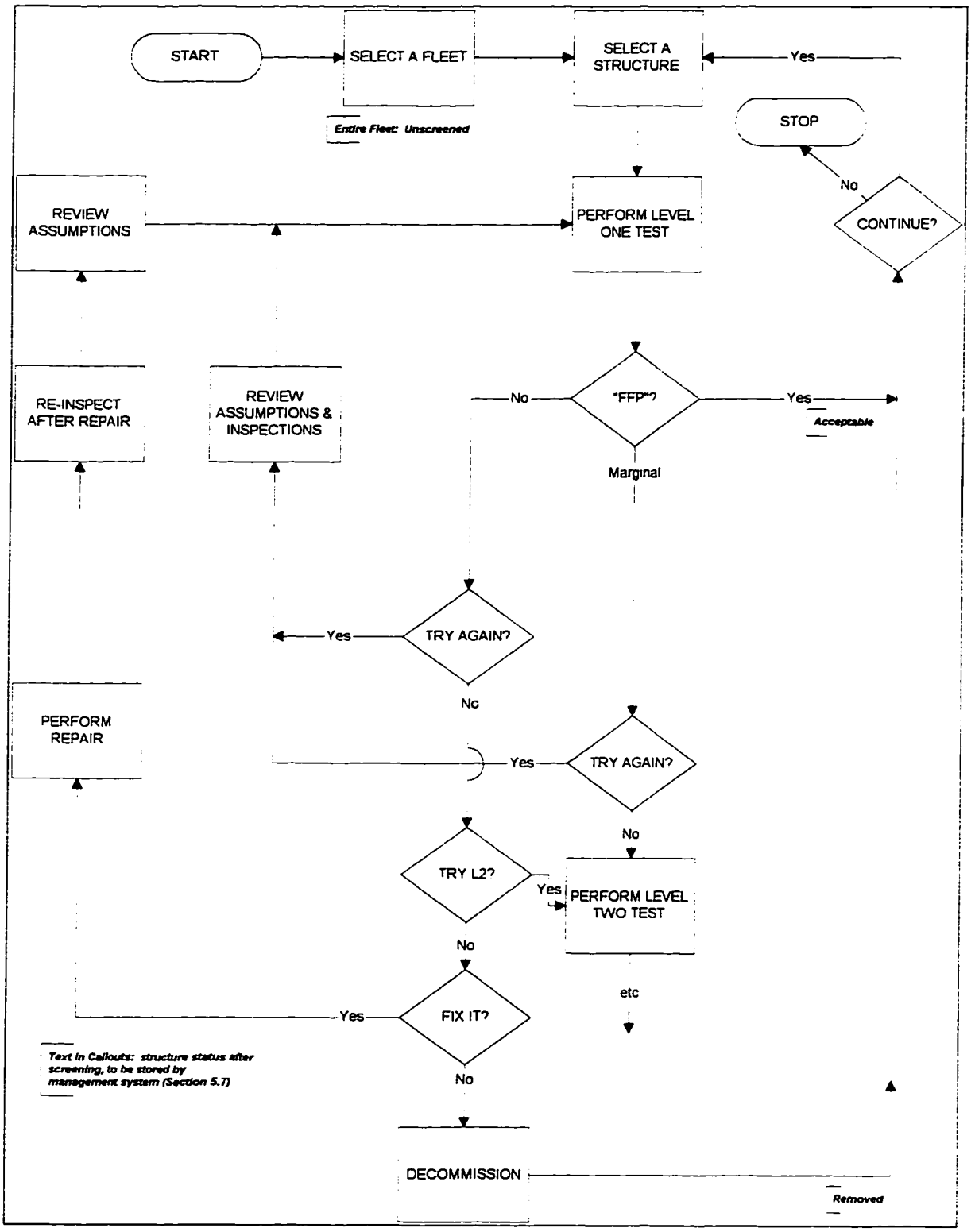


Figure 5-3: Detail of 1st Level of Screening

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

For example, for oil platforms, a "Level One" test might be the L1Risk1 test described in Section 4.2.1.4, or it might be the age- and manning-status search conducted in Section 4.3.2, or it might be some other appropriate test. The question of which test to use at which level of screening is a design issue whose largest factor is likely to be test availability.

Figure 5-3 shows the first level of screening; other levels are similar. Figure 5-3 stresses decisions and actions based purely on engineering judgment. Figure 5-4 adds decision points based on expediency ("Act Now?"). This allows a temporary suspension of work, which makes the storage of structure status in a system such as DAESSIM an important issue. Most importantly, Figure 5-4 introduces the possibility that a test result may be incorrect, and uses economics to help manage that possibility:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

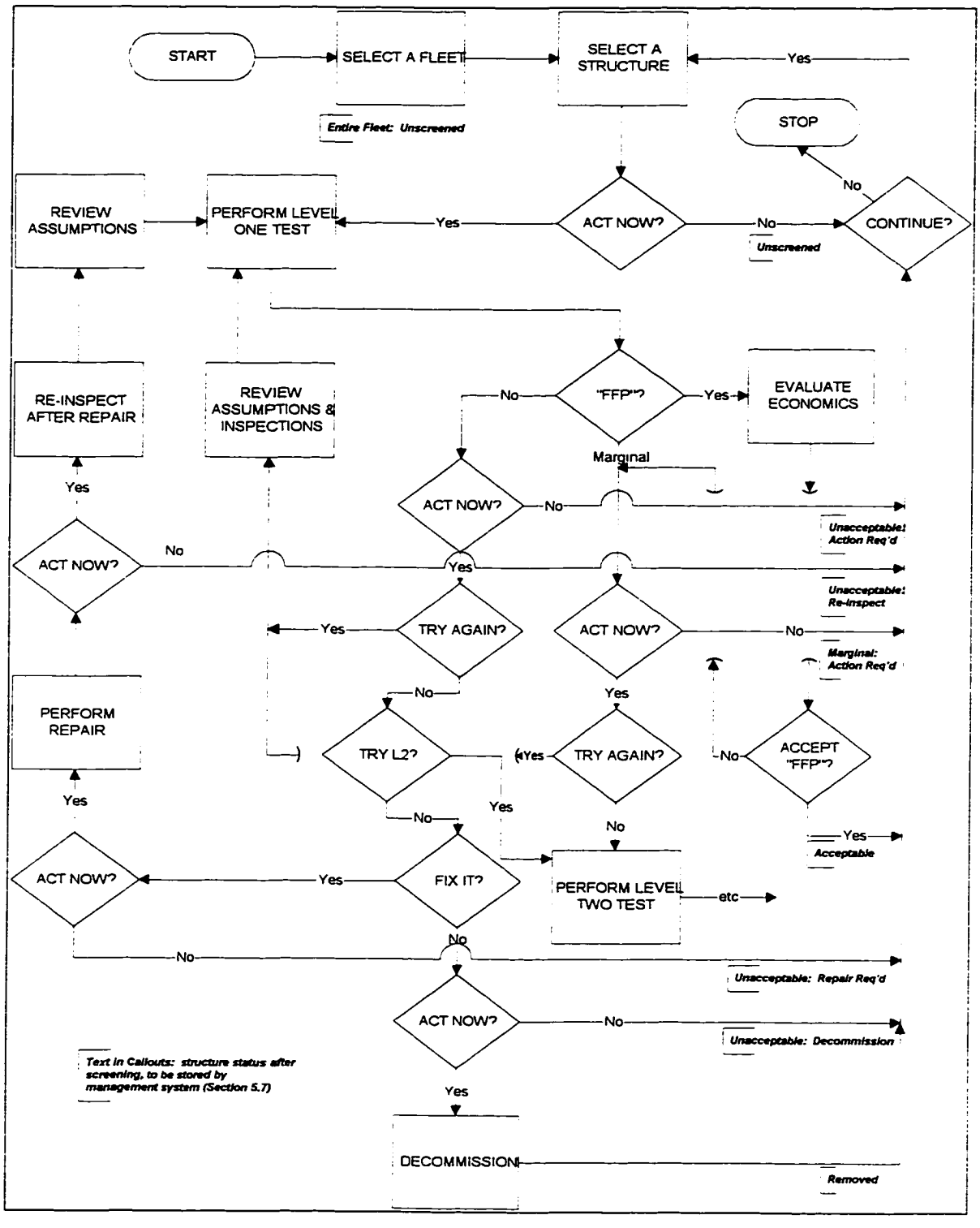


Figure 5-4: Detail of 1st Level of Screening, with Economics

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Only the possibility that the structure is actually other than FFP while the test indicates "FFP" is explored in Figure 5-4 (in the decision to accept or not accept the "FFP" result). Other possibilities, the potential that the structure is actually other than marginal when the test indicates "marginal," and the potential that the structure is actually other than unfit for purpose when the test indicates "not FFP," are not considered because they require no decision on the part of the user. In both cases, a reasonable user would continue treating the structure as if it were either marginal or not FFP.

5.2 PRELIMINARY ECONOMIC DECISION MODEL

A decision tree may be used to model the chance that a structure may not actually be FFP when a test indicates that it is "FFP." Figures 5-5, 5-6, 5-7, and 5-8 follow from Figure 5-4 and together show the decision tree that underlies the economic decisions in a four-level screening system. (Decision trees differ from fault trees in that they present the opportunity to choose an optimal path in an environment of chance events, rather than simply documenting those events. See [Hillier and Lieberman 1990] for more detail.)

Figure 5-5 shows the root node of the decision tree. This is the performance of an L1 test, and corresponds to the "Perform Level One Test" task of Figure 5-4. Figure 5-5 shows the possible outcomes of such a test. Figures 5-6, 5-7, and 5-8 each show the subtrees that result from the branches of Figure 5-5: each starts with the results of the first tests upon a structure. If the Level One test says "unsafe" or "marginal" (using the term "safe" instead of "FFP"), there is no decision to be made: more testing is required. If the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Level One test says "safe," the decision to perform or not perform a subsequent test is shown in Figure 5-6 as node D1, and in Figure 5-4 as the "Accept 'FFP'?" decision. Result nodes such as those following node C3 express one of two possibilities – either the structure is truly safe and will not incur a failure cost, or it is truly unsafe and will incur a failure cost.

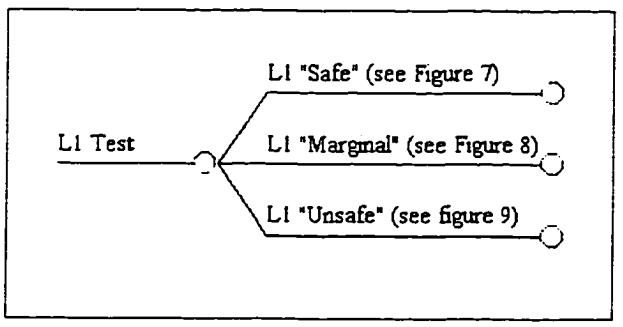


Figure 5-5: Overall View of Tree (Root Node)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

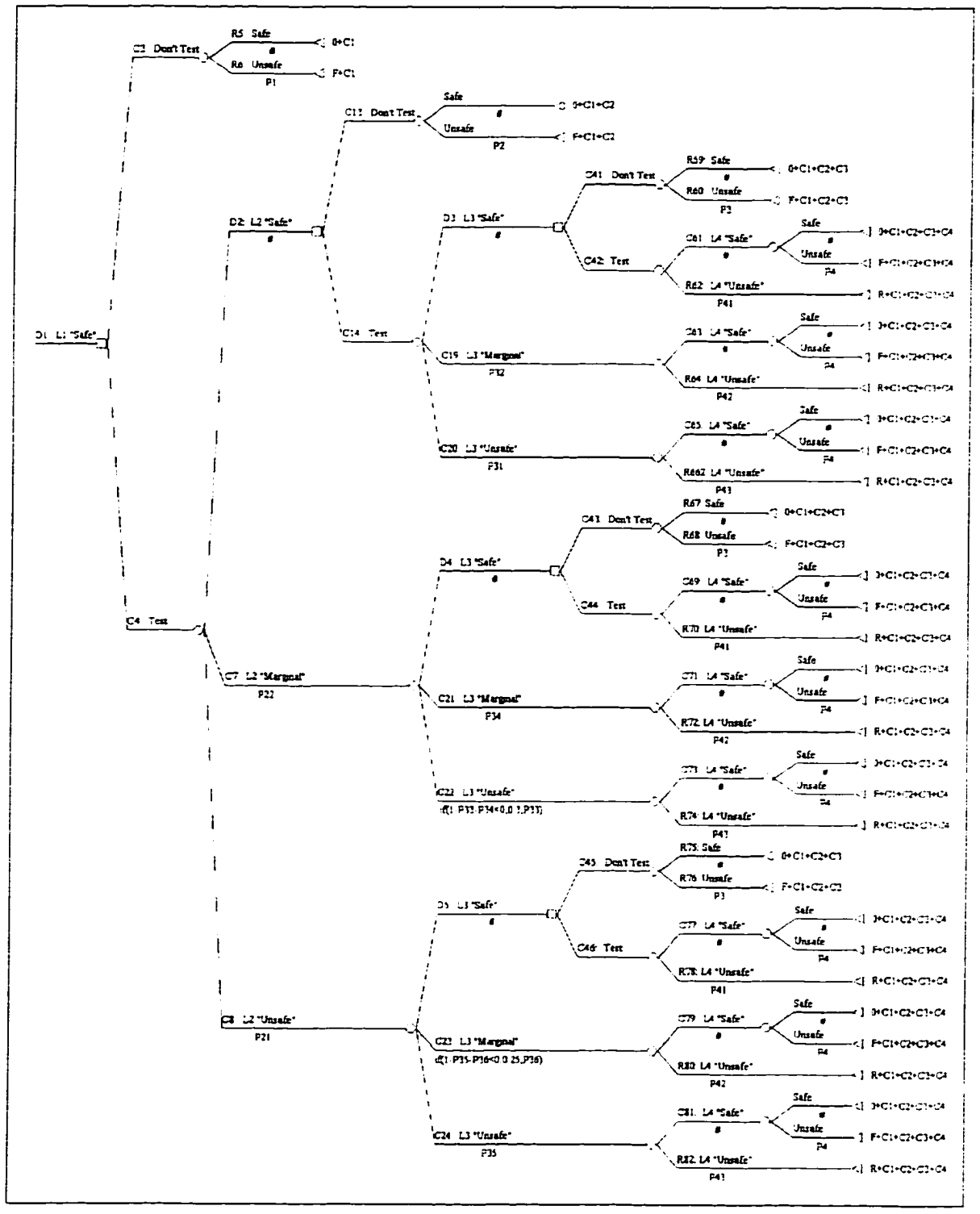


Figure 5-6: First Branch, 4-Level Screening Cycle Decision Tree

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

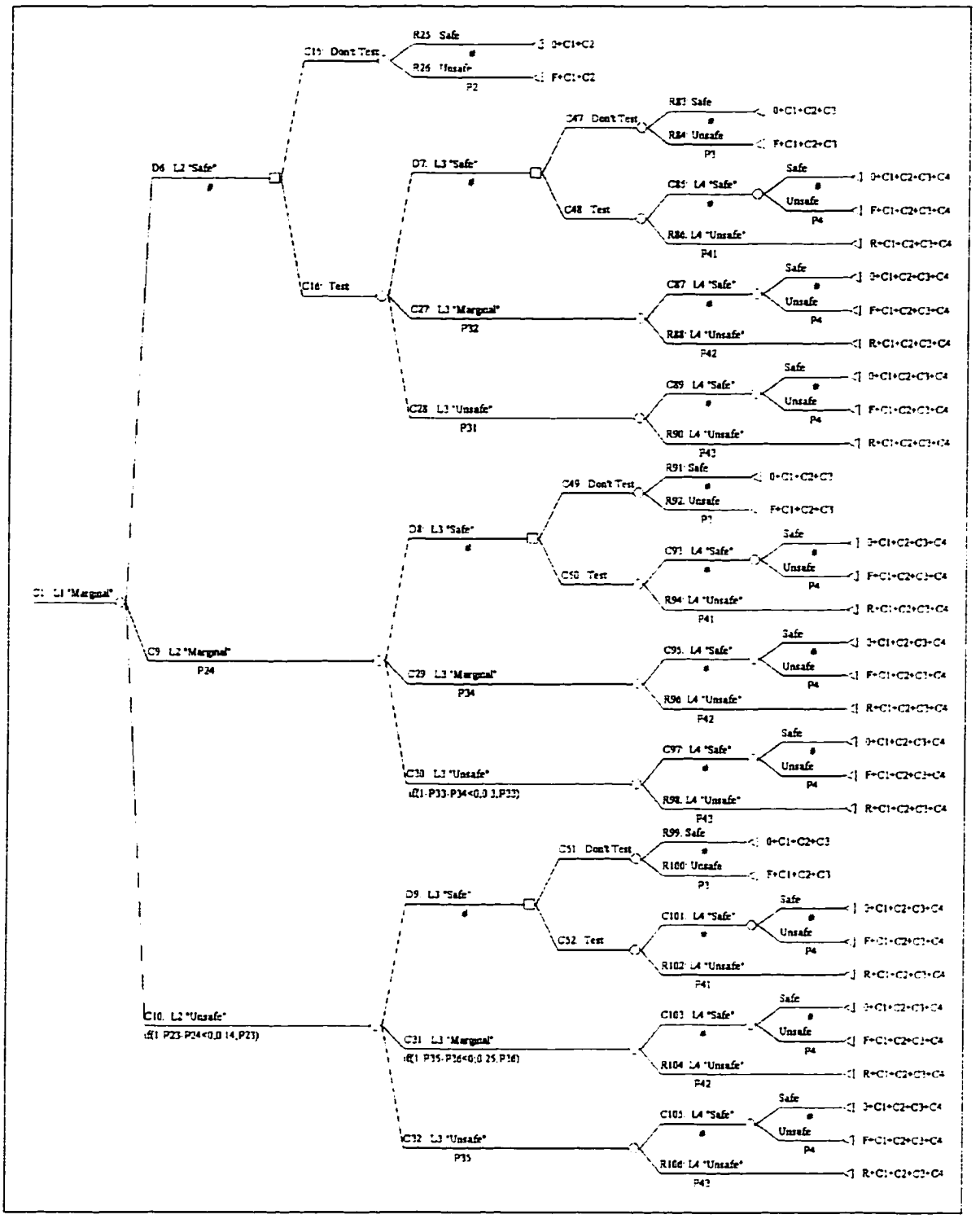


Figure 5-7: 2nd Branch, 4-Level Screening Cycle Decision Tree

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

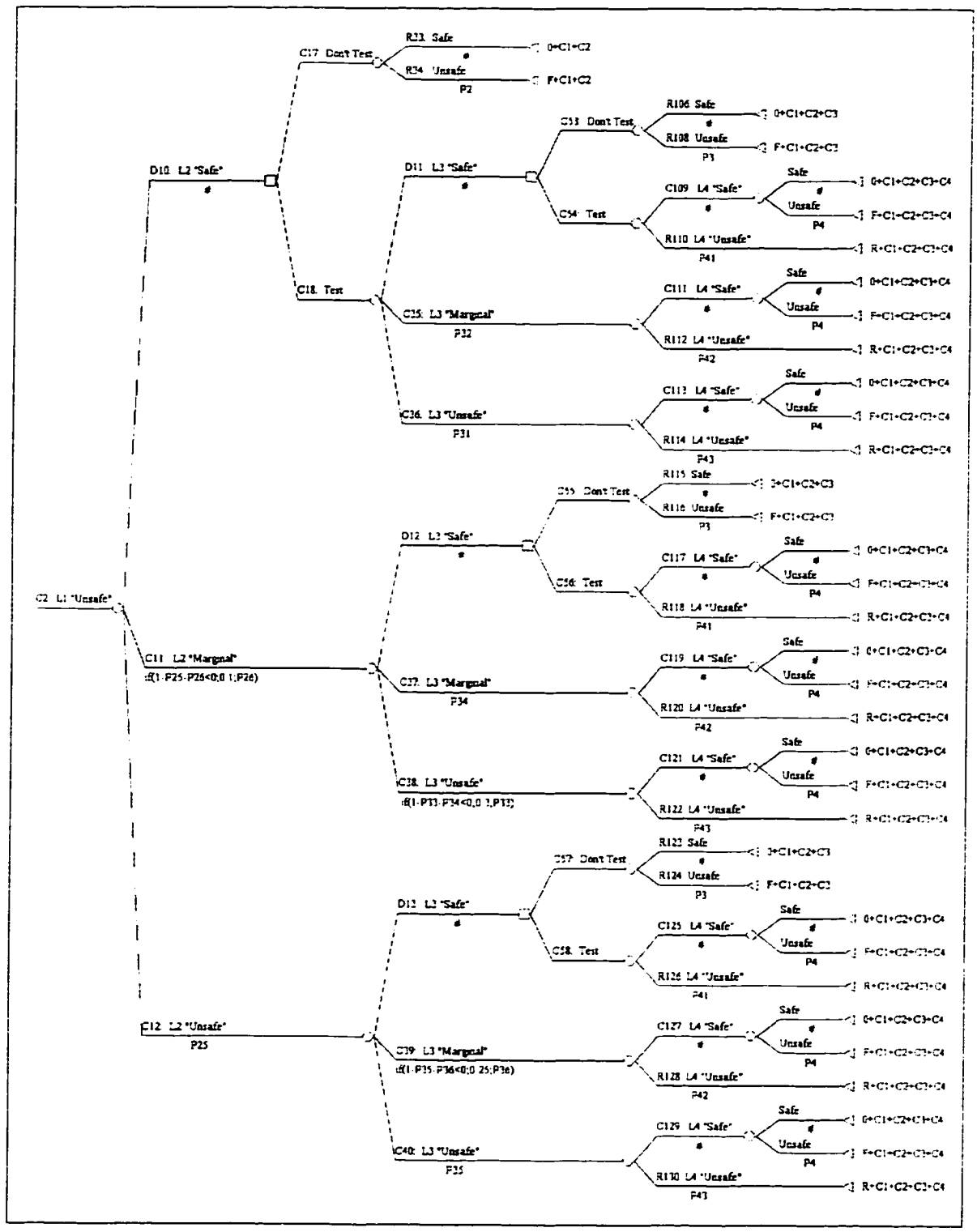


Figure 5-8: 3rd Branch, 4-Level Screening Cycle Decision Tree

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To evaluate economic decisions, values or ranges of values for each of the variables in the model must be established. Once set, a "rollback" operation can be performed on the tree to determine the expected cost resulting from the screening cycle. The path of minimum expected cost is the economically recommended ("optimal") course of action. In addition to the assumptions stated above, it is assumed that costs are defined by the user in a consistent manner. Furthermore, risk aversion and risk preference are ignored for simplicity, as are multi-attribute cost or benefit functions.

There are cost variables and probability variables in the 4-level screening cycle decision tree. The cost variables involved are:

- F Cost of Failure (includes all costs generated by a structure's failure)
- R Cost of Repair (includes all repair costs)
- C1 Cost of Level 1 Test (includes all test costs, including inspections)
- C2 Cost of Level 2 Test (includes all test costs, including inspections)
- C3 Cost of Level 3 Test (includes all test costs, including inspections)
- C4 Cost of Level 4 Test (includes all test costs, including inspections)

The initial listing of the probability variables is (from Figure 5-6 through 5-8):

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- P1 P(U|L1 "S") (probability that a structure is actually unsafe, given a Level 1 test result of "safe")

- P2 P(U|L2 "S") (probability that a structure is actually unsafe, given a Level 2 test result of "safe")

- P3 P(U|L3 "S") (probability that a structure is actually unsafe, given a Level 3 test result of "safe")

- P4 P(U|L4 "S") (probability that a structure is actually unsafe, given a Level 4 test result of "safe")

Probabilities P1 through P4, dealing with real outcomes, are called *decision probabilities*. The following probabilities are concerned only with the path taken towards an outcome, and are termed *branch probabilities*:

- P11 P(L1 "U") (probability of a Level 1 test result of "unsafe")

- P12 P(L1 "M") (probability of a Level 1 test result of "marginal")

- P21 P(L2 "U"|L1 "S") (probability of a Level 2 test result of "unsafe" given a Level 1 test result of "safe")

- P22 P(L2 "M"|L1 "S")

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- P23 P(L2 "U"|L1 "M")
- P24 P(L2 "M"|L1 "M")
- P25 P(L2 "U"|L1 "U")
- P26 P(L2 "M"|L1 "U")
- P31 P(L3 "U"|L2 "S")
- Probabilities P32 through P36 and P41 through P43 are similarly defined.

To make the problem workable, it is important to understand a) which variables are under the evaluator's control, and b) which of those variables have the greatest impact. Unfortunately, none of the variables are really under the evaluator's control: the cost variables depend in large part on which structure is selected for the given phase of screening and on which tests are used in the system (it is assumed that the same test is used at a given level of screening throughout the screening of a fleet). While it may seem a prodigious task to estimate all of the probability variables, this is neither necessary nor desirable. They may be calculated directly from a few initial estimates, as shown in the next section.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.3 CALCULATION OF CONDITIONAL PROBABILITIES

5.3.1 Matrix Calculations

For any binary "safe/unsafe" test run on any set of structures, results can be presented as follows [Marshall and Oliver 1995]:

		j=1 safe	j=2 unsafe	
i=1	"safe"	(1,1)	(1,2)	# of "safe" structures
i=2	"unsafe"	(2,1)	(2,2)	# of "unsafe" structures
		# of safe structures	# of unsafe structures	= total # of structures

Figure 5-9: Two-Way Contingency Table of Event Outcomes C

In the *two-way contingency table of event outcomes, C* ("contingency table"), in Figure 5-9, the quantity represented by the (i,j) pair (1,1) is the number of safe structures that are judged "safe" by the test in question; (2,1) is the number of safe structures judged "unsafe;" etc. The sum of (1,1) and (1,2) is the number of structures judged "safe" by the test; the sum of (2,1) and (2,2) is the number of structures judged "unsafe;" the sum of (1,1) and (2,1) is the number of truly safe structures; and the sum of (1,2) and (2,2) is the number of truly unsafe structures. Safe and "safe" should be consistent terms: if a "safe" result refers to structural seismic safety, then the sum of (1,1) and (2,1) should be the number of structures that are truly structurally seismically safe.

An infrastructure screening system usually deals with low probability, high-consequence failures. Thus, the first priority of a test is to catch as many unsafe structures as possible;

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

i.e., minimize table C quantity (1,2). The second priority is to waste as little money as possible in needless testing; i.e., minimize quantity (2,1), on the assumption that all unsafe structures are either retested, repaired, or removed – all at added cost.

The *likelihood* or *forecast probability matrix, F,* can be derived from the contingency table:

	safe	unsafe	=	safe	unsafe
"safe"	P("S"|S)	P("S"|U)		(1,1)/((1,1)+(2,1))	(1,2)/((1,2)+(2,2))
"unsafe"	P("U"|S)	P("U"|U)		(2,1)/((1,1)+(2,1))	(2,2)/((1,2)+(2,2))

Figure 5-10: Forecast Probability Matrix F

In Figure 5-10, P("S"|S) denotes the probability of a safe test result for a safe structure. Also derivable from the events matrix is the *decision probability matrix, P*:

	safe	unsafe	=	safe	unsafe
"safe"	P(S|"S")	P(U|"S")		(1,1)/((1,1)+(1,2))	(1,2)/((1,1)+(1,2))
"unsafe"	P(S|"U")	P(U|"U")		(2,1)/((2,1)+(2,2))	(2,2)/((2,1)+(2,2))

Figure 5-11: Decision Probability Matrix, P

In Figure 5-11, P(S|"S") denotes the probability that a structure that has been tested as "safe" truly is safe. The columns of Figure 5-10 sum to one, while the rows of Figure 5-11 sum to one. For a level *k* test, P(U|"S") = P*k* in the decision tree.

Measures of test quality can be derived from the F matrix. A *discriminating* test is one for which P("S"|S) > P("S"|U), and P("U"|U) > P("U"|S); i.e., a discriminating test will

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

be more likely to correctly identify a condition than not. A *perfectly discriminating* test is one for which P("S"|U) and P("U"|S) are each equal to zero, and indicates a test that is always correct. Finally, the *test correlation coefficient* ρ_T is:

$$\rho_T = [P(S|\text{"S"}) - P(S|\text{"U"})] \cdot \sqrt{\frac{P(\text{"S"}) \cdot P(\text{"U"})}{P(S) \cdot P(U)}}, \quad (1)$$

$$= [P(\text{"S"}|S) - P(\text{"S"}|U)] \cdot \sqrt{\frac{P(S) \cdot P(U)}{P(\text{"S"}) \cdot P(\text{"U"})}} \quad (2)$$

[Marshall and Oliver 1995]. The further ρ_T is from zero (either towards −1 or towards 1), the more useful the test (a test with ρ_T = -1 would reliably indicate the opposite of real-world conditions).

5.3.2 Calculation of Decision Probabilities

The relationships between the contingency table, the F matrix, and the P matrix mean that probabilities can be assessed from differing points of view, depending on the information available. For example, when tests are well-established, either the F matrix or the contingency table may be used to establish P. Of more interest is the case where tests are *not* well-established, but the total number of structures N is known, the annual number of failing structures D can be estimated, and estimates can be made for at least two of the members of the contingency table – for example, (1,1) and (1,2). In such a case,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$P(U|\text{“S”}) = (1,2)/((1,1)+(1,2)), \tag{3}$$

and

$$\rho_T = ((1,1)/((1,1)+(1,2))) - ((N - D - (1,1))/((N - D - (1,1)) + (D - (1,2)))) \cdot \sqrt{\frac{((1,1)+(1,2)) \cdot (N - ((1,1)+(1,2)))}{(N - D) \cdot (D)}} \tag{4}$$

Table 5-1 gives examples of $P(U|\text{“S”})$ and ρ_T for certain arbitrary numbers of structures and annual failures, and arbitrary estimates of counts (1,1) and (1,2). More are presented in Appendix D. For a given number of structures, $P(U|\text{“S”})$ decreases with decreasing numbers of failures, with decreasing quantities (1,2), and with increasing quantities (1,1). The correlation coefficient ρ_T increases with decreasing (1,2) and increasing (1,1). Since (2,1) and (1,1) sum to a constant (the number of safe structures), minimizing (1,2) and maximizing (1,1) meets the goals of minimizing (1,2) and (2,1).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Number of Structures	Number of Failures	(1,2): unsafe & tested "safe"	(1,1): safe & tested	P(U|"S")	ρ_T
1000	5	2	950	0.0021	0.1831
2000	10	8	400	0.0196	-0.1048
4000	5	4	2000	0.0020	-0.0212
4000	5	1	2000	0.0005	0.0212
4000	5	0	2000	0.0000	0.0354
4000	5	4	3600	0.0011	0.0120
4000	5	1	3600	0.0003	0.0827
4000	5	0	3600	0.0000	0.1061
4000	5	4	3920	0.0010	0.0469
4000	5	1	3920	0.0003	0.1984
4000	5	0	3920	0.0000	0.2476
4000	12	10	2000	0.0050	-0.0363
4000	12	5	2000	0.0025	0.0093
4000	12	3	2000	0.0015	0.0275
4000	12	0	2000	0.0000	0.0549
4000	12	10	3600	0.0028	0.0128
4000	12	5	3600	0.0014	0.0891
4000	12	3	3600	0.0008	0.1194
4000	12	0	3600	0.0000	0.1646
4000	12	10	3920	0.0025	0.0624
4000	12	5	3920	0.0013	0.2283
4000	12	3	3920	0.0008	0.2917
4000	12	0	3920	0.0000	0.3840
4000	20	16	2000	0.0079	-0.0420
4000	20	8	2000	0.0040	0.0145
4000	20	4	2000	0.0020	0.0427
4000	20	0	2000	0.0000	0.0709
4000	20	16	3600	0.0044	0.0250
4000	20	8	3600	0.0022	0.1197
4000	20	4	3600	0.0011	0.1664
4000	20	0	3600	0.0000	0.2127
4000	20	16	3920	0.0041	0.1040
4000	20	8	3920	0.0020	0.3103
4000	20	4	3920	0.0010	0.4055
4000	20	0	3920	0.0000	0.4962

Table 5-1: Values of P(U|"S") and ρ_T

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.3.2 Calculation of Branch Probabilities

The values of P(U|"S") in Table 5-1 are taken from the P matrix, as computed from the contingency table estimates (1,1) and (1,2). For a multi-level screening system with independent tests, these values must be estimated for *each* test level (quantities L1(1,1), L1(1,2), L2(1,1), L2(1,2), ..., L*n*(1,1), and L*n*(1,2), where L*k* denotes the test level). The number of structures N and the expected annual number of failures D for the fleet in question must also be estimated.

Once L1(1,1) through L*n*(1,2) estimates have been supplied, the F matrices for each test level can be derived. These can be used to calculate the conditional probabilities for each result node as well as for each branch node. In Figures 5-6 through 5-8, for example, the decision probability P2 was used whenever the "safe" result of an L2 test was to be accepted (P(U|"S")). However, this probability must be conditioned by the events leading up to it: the probability of an unsafe structure being labeled "safe" by an L2 test after an L1 test has labeled it "safe" is smaller than the probability of an unsafe structure being labeled "safe" by an L2 test after an L1 test has labeled it "unsafe" (0.0004745 vs. 0.0020142 at the example defaults, described later in this section). The probability that an unsafe structure has been labeled "safe" by an L4 test is 0.0000068 when that structure previously received "safe" verdicts from L1, L2, and L3 tests; it is 0.12044 when that structure previously received "unsafe" verdicts from those tests (under the example defaults).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This large discrepancy is a graphic illustration that conditional probabilities should be calculated – if the unconditioned L4 P(U|"S") were used (i.e., the P4 value of Section 5.2), its value (0.000251 at the example defaults) might produce errors. The various tests are still independent; however, the ratio of safe to unsafe structures available to a test has been changed by the prior tests. A sample real-world conditional P(U|"S") value may be derived from Puskar et al's [1994] post-high-level analysis (L3 or L4), post-Hurricane Andrew data for selected offshore oil platforms. For this group, P(U|"S") is 0.14.

In addition to the decision probabilities P(U|"S") (previously labeled P1, P2, P3, and P4), the branch probabilities previously labeled P21 through P26, P31 through P36, and P41 through P43 can be calculated, further increasing the coherence of the model. Since the number of structures that can be expected to pass to each node of the tree differs, the conditional probabilities can also be expected to differ. For example, at node C0 (the root node of Figure 5-5), there can be expected to be N-D safe structures and D unsafe ones. Once these are screened by the L1 test, there can be expected to be $(N\text{-}D)\cdot\frac{L1(1,1)}{N-D}$ safe structures and $D\cdot\frac{L1(1,2)}{D}$ unsafe structures passed to node D1 of Figure 5-6. The conditional probability of node D1 is then the quotient of the total expected number of structures at node D1 divided by the total expected number of structures at the prior node C0:

$$p(D1) = \frac{L1(1,1)+L1(1,2)}{N}. \qquad (5)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Further "rightward" in the tree, the expressions for conditional probabilities become fairly complicated. The probability of node D10 (the probability of an L2 "safe" result following an L1 "unsafe" result) is

$$p(D10) = \frac{\left(\frac{(N-D-L1(1,1))\cdot L2(1,1)}{N-D}\right)+\left(\frac{(D-L1(1,2))\cdot L2(1,2)}{D}\right)}{(N-D-L1(1,1))+(D-L1(1,2))}, \qquad (6)$$

while the probability of node C20 (the probability of an L3 "unsafe" result following an L2 "safe" and an L1 "safe" result) is

$$p(C20) = 1-\frac{\left(\frac{L1(1,1)\cdot L2(1,1)\cdot L3(1,1)}{(N-D)^2}\right)+\left(\frac{L1(1,2))\cdot L2(1,2)\cdot L3(1,2)}{D^2}\right)}{\frac{L1(1,1)\cdot L2(1,1)}{N-D}+\frac{L1(1,2)\cdot L2(1,2)}{D}}. \qquad (7)$$

Alternatively, p(C20) can be stated directly by deleting the "1-" term, replacing the L3(1,1) terms with (N-D-L3(1,1)), and the L3(1,2) terms with (D-L3(1,2)).

Returning to the decision probabilities (P(U|"S")), the probability of a failure event occurring at node C13 ("Accept L2 Safe," following L1 "safe") is

$$p(C13) = \frac{\left(\frac{L1(1,2)\cdot L2(1,2)}{D}\right)}{\left(\frac{L1(1,1)\cdot L2(1,1)}{N-D}\right)+\left(\frac{L1(1,2)\cdot L2(1,2)}{D}\right)}, \qquad (8)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

while the probability of a failure event occurring at node C45 ("Accept L3 Safe," following L1 "safe" and L2 "unsafe") is

$$p(C45) = \frac{\left(\frac{L1(1,2)\cdot(D - L2(1,2))\cdot L3(1,2)}{D^2}\right)}{\left(\frac{L1(1,1)\cdot(N - D - L2(1,1))\cdot L3(1,1)}{(N-D)^2}\right)+\left(\frac{L1(1,2)\cdot(D - L2(1,2))\cdot L3(1,2)}{D^2}\right)} \quad (9)$$

Compared with an estimation process, calculating conditional probabilities in this manner is more accurate, requires less expert knowledge of probability, and implicitly accounts for the probabilities P(S|"U") that some of the "unsafe" structures at each branch node are actually safe.

5.4 REVISED ECONOMIC DECISION MODEL

5.4.1 Default Variable Values

With these insights for conditional probability, it is appropriate to select the example default values for a revised model. The number of structures (4000), annual expected number of failures (12), failure costs, and repair costs in Table 5-2 are approximations for offshore oil platforms in the Gulf of Mexico. The default test values reflect an assumption of increasing correlation and decreasing P(U|"S") as tests become more expensive:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Level One through four values of (1,1) are 3000, 3500, 3900, and 3980, respectively, and the (1,2) values are 5, 3, 2, and 1 (see Section 5.6 for an investigation of the assumption of decreasing P(U|"S")). Correlation coefficients ρ_T for Levels One through Four are 0.0425, 0.1041, 0.2870, and 0.7275, respectively. Decision probabilities P1, P2, P3, and P4, and branch probabilities P21 through P26, P31 through P36, and P41 through P43 are not shown in Table 5-2 since they are derived on a per-node basis as described above. Arbitrary high and low limits for sensitivity analysis purposes are also shown in Table 5-2.

Name	Descrip	Low	High	Default	P(U|"S") (unconditioned)	ρ_T
F	Failure Cost	$1,000,000	$500,000,000	$150,000,000	n/a	n/a
R	Repair Cost	$500,000	$10,000,000	$1,000,000	n/a	n/a
C1	L1 Test Cost	$1	$20,000	$5,000	n/a	n/a
C2	L2 Test Cost	$1,000	$30,000	$10,000	n/a	n/a
C3	L3 Test Cost	$5,000	$75,000	$30,000	n/a	n/a
C4	L4 Test Cost	$10,000	$150,000	$75,000	n/a	n/a
L1(1,1)	Safe Structures L1 "Safe"	2500	3500	3000	0.00166	0.0422
L1(1,2)	Unsafe Structures L1 "Safe"	4	12	5		
L2(1,1)	Safe Structures L2 "Safe"	3000	3800	3500	0.00086	0.1041
L2(1,2)	Unsafe Structures L2 "Safe"	1	5	3		
L3(1,1)	Safe Structures L3 "Safe"	3500	3988	3900	0.00051	0.2870
L3(1,2)	Unsafe Structures L3 "Safe"	1	4	2		
L4(1,1)	Safe Structures L4 "Safe"	3900	3988	3980	0.00025	0.7275
L4(1,2)	Unsafe Structures L4 "Safe"	0	3	1		
N	# of Structures	4000	4000	4000	n/a	n/a
D	Yearly # of Failures	12	12	12	n/a	n/a

Table 5-2: Variable Ranges and Defaults

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.4.2 Model Simplification

A problem existed with the model at this point: while the tree (Figures 5-5 through 5-8) performed tests with three outcomes ("safe," "marginal," and "unsafe"), the probability model used 2x2 matrices. (In programming terms, this is an *impedance mismatch*.) Either the probability model had to be expanded to use 3x3 matrices, or the tree had to be simplified to a binary ("safe" and "unsafe") format. The latter was chosen, for the following reasons:

- It is very difficult to estimate 2x2 probabilities in any form, much less 3x3;
- The explicit calculation of conditional probabilities implicitly calculates P(S|"U");
- The actions following a "marginal" verdict are the same as those following an "unsafe" one.

5.4.3 Model Structure

The decision tree was modified in accordance with the above. Concurrently, a two-level model was also built. This model used Table 5-2's L1 and L4 example default values. Lower rollback values with the two-level model than with the four-level indicated that, in some situations, it is preferable to bypass some levels of testing.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This result motivated the creation of Model 5. Model 5 includes the option of jumping from a level i test to a level i+1, i+2, or i+3 test (within the four-level limit). It predicts an expected cost of $107,000 at the example default values. The upper half of Model 5 (starting with node D1) is shown in Figure 5-12; the bottom half is similar. Both the upper and lower halves branch off a common root node, similar to that of Figure 5-5 but having only two branches. Variable ranges and defaults are those listed in Table 5-2.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

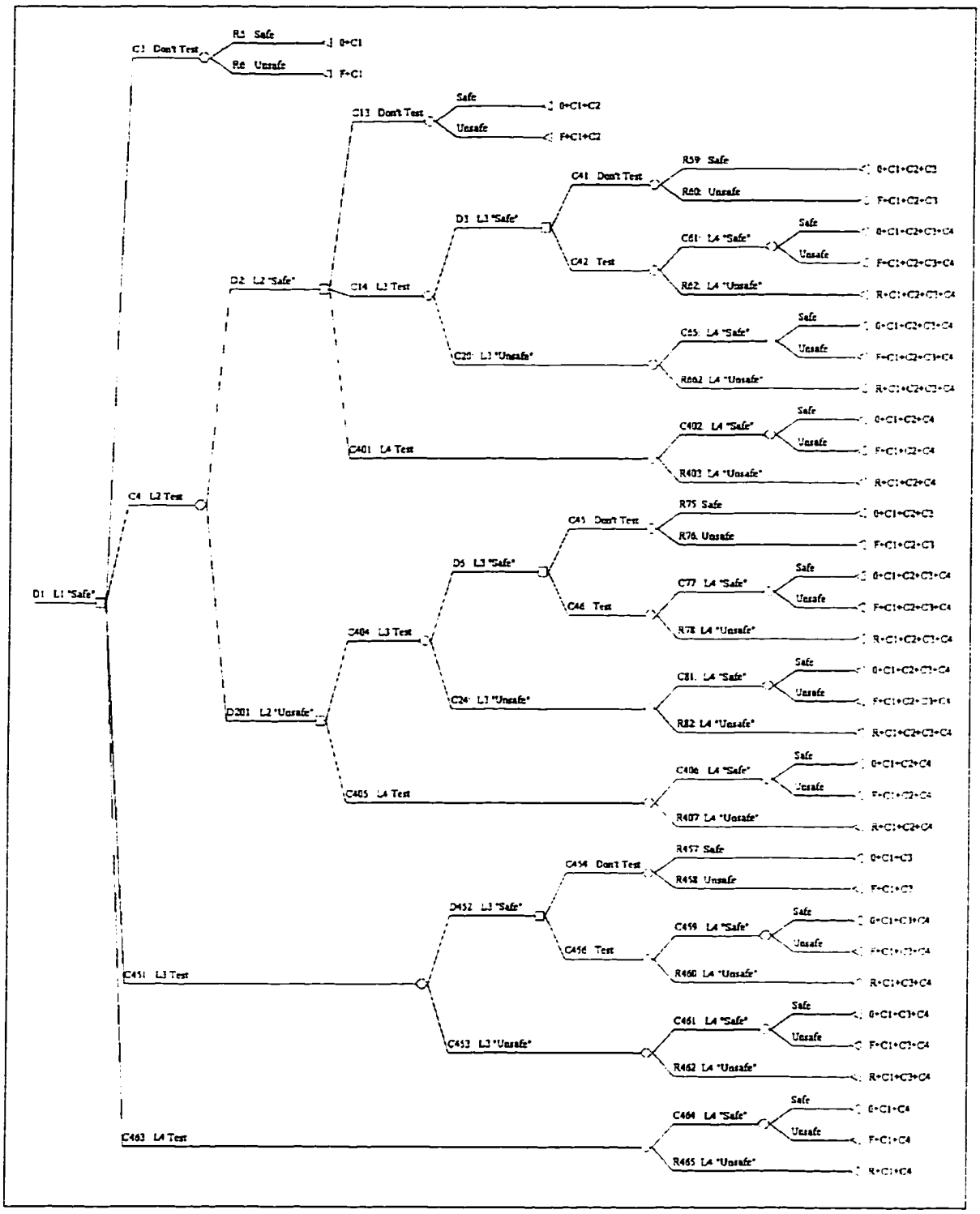


Figure 5-12: Decision Tree with Level-Skipping Capability ("Model 5"), Upper Half

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.4.4 Sensitivity Analysis

The impacts of the variables in the model were examined using sensitivity analysis. The first analysis was performed at the root node (C0) of the Model 5 decision tree, just to the left of the D1 node of Figure 5-12. This analysis produced a "tornado" sensitivity graph, Figure 5-13, designed to show the relative influence of the variables on the model's expected value. Variables are ranked in descending order of influence; each bar of the graph is a one-way sensitivity analysis of a variable. Unfortunately, only one variable at a time is varied in such an analysis, while the others are held at their defaults. Nevertheless, a tornado diagram provides a good starting point for further analysis.

Figure 5-13 shows that F is the most influential variable. All the variables of the tree have influence at C0 because, for the example default values, all tests are in an optimal path: no test is certain to be skipped. For example, one of the optimal (cheapest) paths with the default values is: C0 (L1 Test) → D1 (L1 "Safe") → C4 (L2 Test) → D2 (L2 "Safe") → C14 (L3 Test) → C20 (L3 "Unsafe") → C65 (L4 "Safe"). Other optimal paths are possible due to the influence of random events – for example, there is no guarantee that an L1 test will assess a structure as "safe." Performing sensitivity analyses at appropriate nodes further to the right on the tree allows the assessment of such events.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

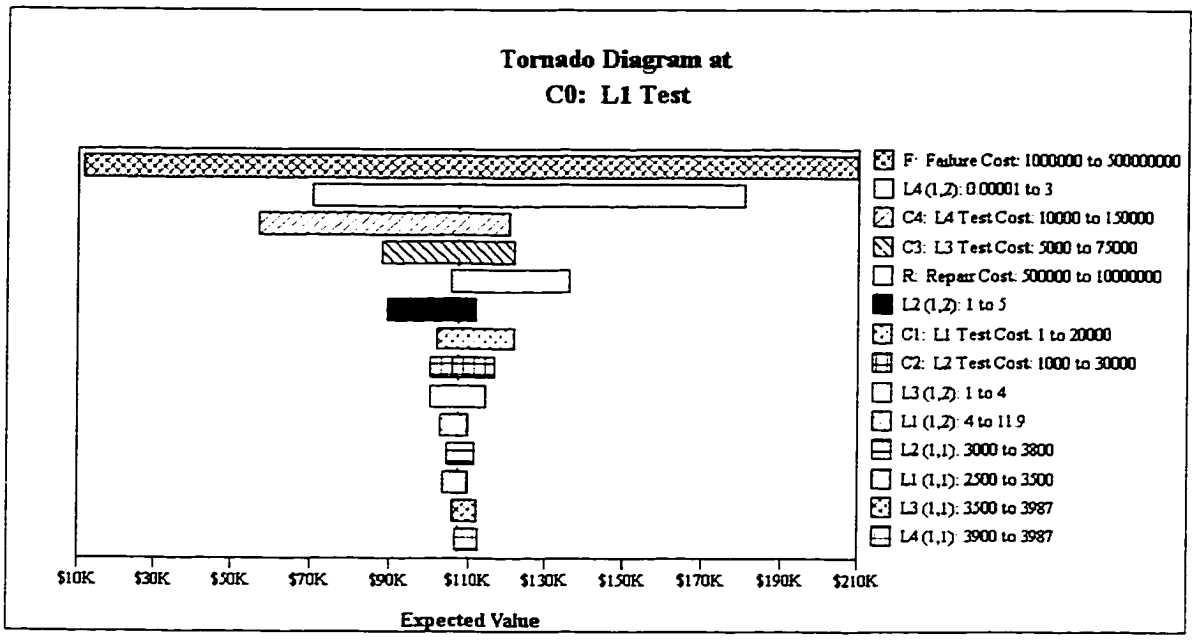


Figure 5-13: Tornado Sensitivy Analysis, Root Node, Model 4

At decision node D1, an L1 result of "safe" has been returned. The analyst must decide whether to accept that result and cease testing (node C3), to perform an L2 test (node C4), to skip to an L3 test (node C451), or to skip directly to an L4 test (node C463). A tornado diagram at a decision node should illustrate (with the heavy vertical lines on each variable's bar) how the changing values of each variable change not only the expected value of the tree, but the optimal decision that might be made. In Figure 5-14, two such lines can be seen for F. Lines in many other variables' bars are near the bars' end points, which may only be a function of the chosen default value for F. To see how the optimal policy changes with the top two most influential variables, a two-way sensitivity analysis was performed (Figure 5-15).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A two-way sensitivity analysis plots one variable against another and examines how the optimal decision at the node being assessed changes with the changing variables. In Figure 5-15, for example, L4(1,2) (the estimated number of unsafe structures assessed as "safe" by an L4 test) is plotted against F (the estimated cost of failure for the structure in question). The analysis is conducted at node D1, which is the point at which a "safe" result has been rendered by an L1 test. The choices facing the analyst are thus to accept the L1 result and halt testing, to proceed with an L2 test, to skip L2 and perform an L3 test, or to skip directly to an L4 test. The extreme left-hand side of Figure 5-15 (the diagonally checked region) indicates that accepting the L1 test is cheapest. As F increases, the cheapest path is to perform an L2 test (the checked region); and if F is very large and L4(1,2) is small, it is optimal to skip directly to an L4 test (the diagonally striped region).

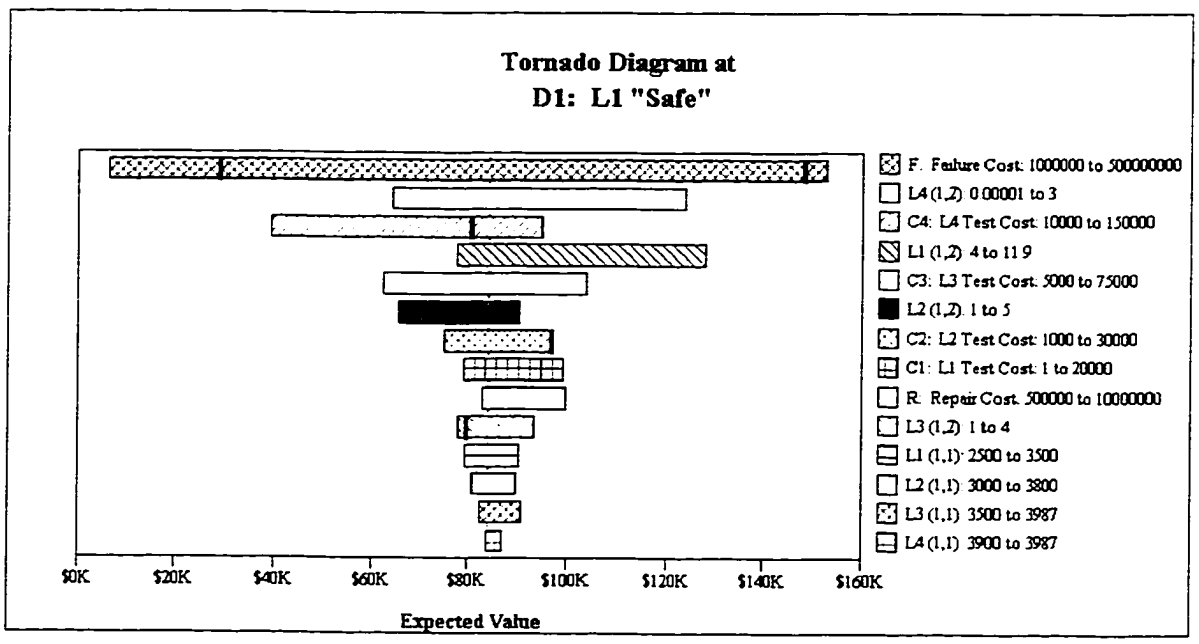


Figure 5-14: Tornado Sensitivity Analysis, Node D1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

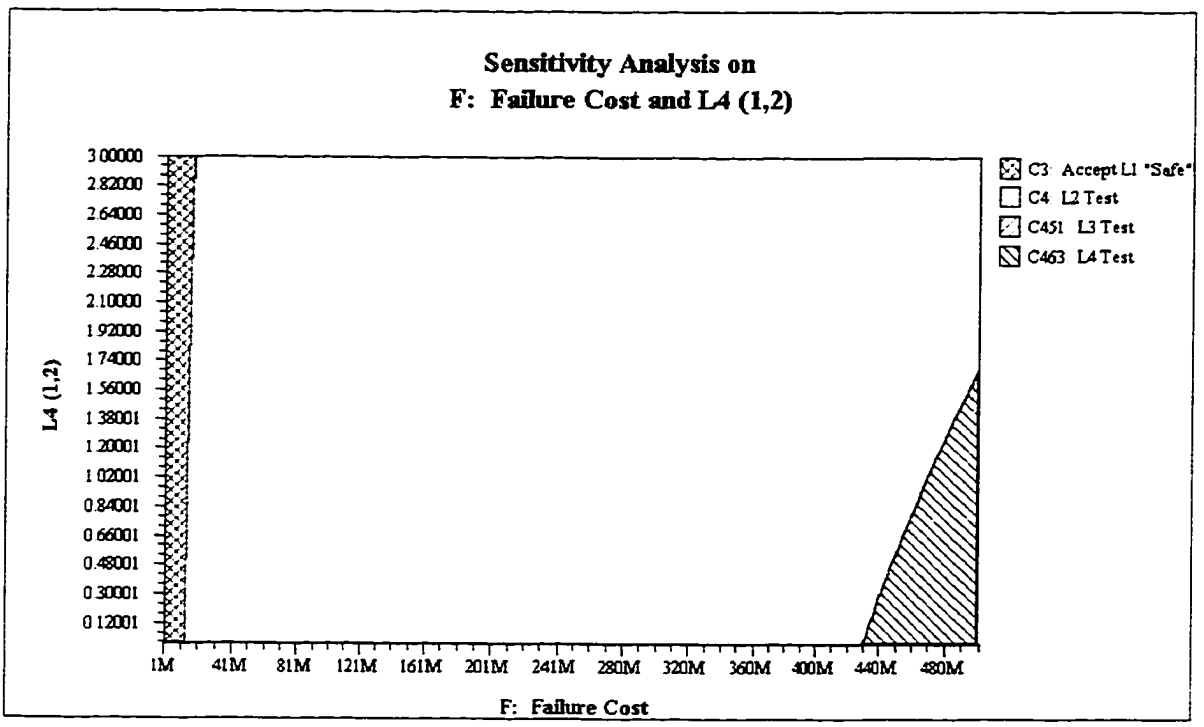


Figure 5-15: Two-Way Sensitivity Analysis of F and L4(1,2) at Node D1

Figure 5-16 is a similar sensitivity diagram at D1, this time comparing F with C4 (the cost of performing an L4 test, and the third-ranked variable in Figure 5-14). While C4 has little influence at low values of F, where it is always preferable to accept an L1 "safe" judgment, it does affect the optimal decision at higher values of F. Of particular interest is the region centering around F = ~$75 million: as the L4 test gets more expensive, the optimal decision changes from skipping directly to an L4 test to moving to an L2 test to, finally, skipping to an L3 test. The reason for the unusual shape of the L3 Test region is that, for this narrow range of F, an accurate test is preferred but an expensive L4 test is not justified. A screening system designer, faced with this result, may decide to use a set of tests with different balances of cost and accuracy than these L3 and L4 defaults..

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

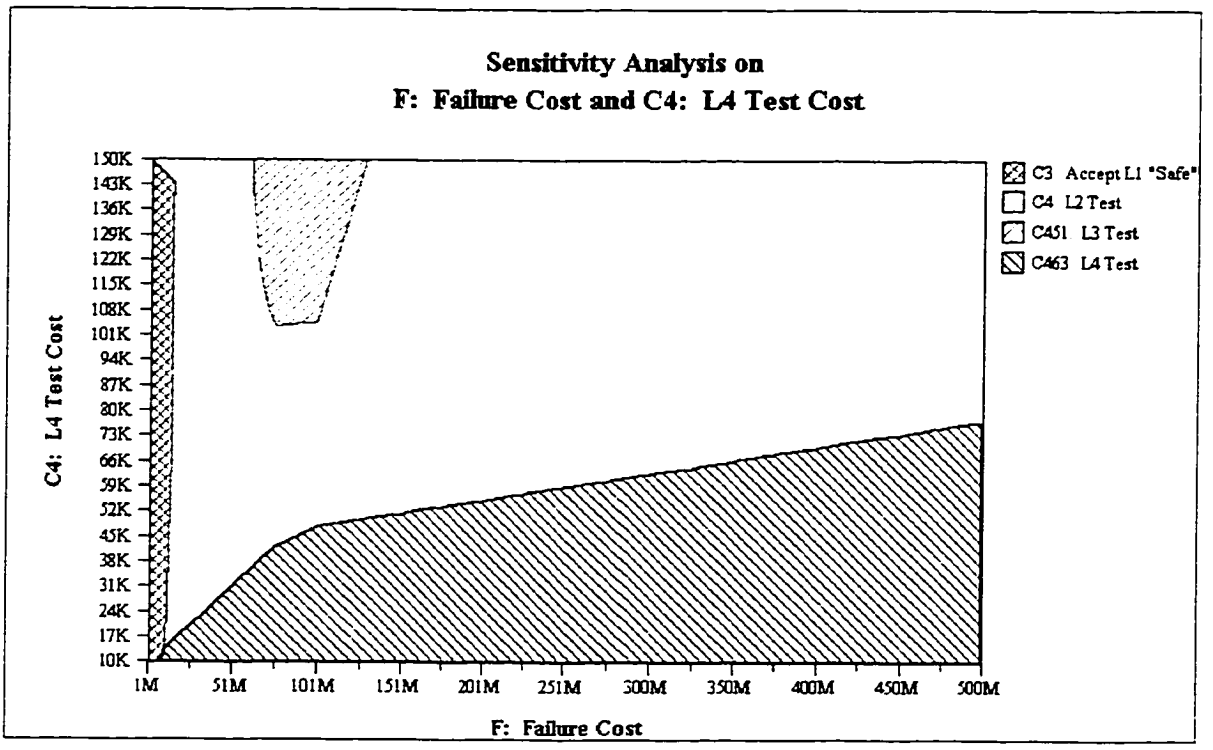


Figure 5-16: Two-Way Sensitivity Analysis of F and C4 at Node D1

If the "Accept L1 "Safe"" option is chosen, then testing is ended for the structure in question. Of more interest is what happens if another path is chosen: for example, the "L2 Test" path. Figure 5-17 shows the tornado diagram at that point, node C4 (no decisions are made at chance nodes such as C4, so vertical lines indicating changes in optimal policy do not appear in chance node tornado diagrams). Note that none of the nodes so far have been very sensitive to repair cost R.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

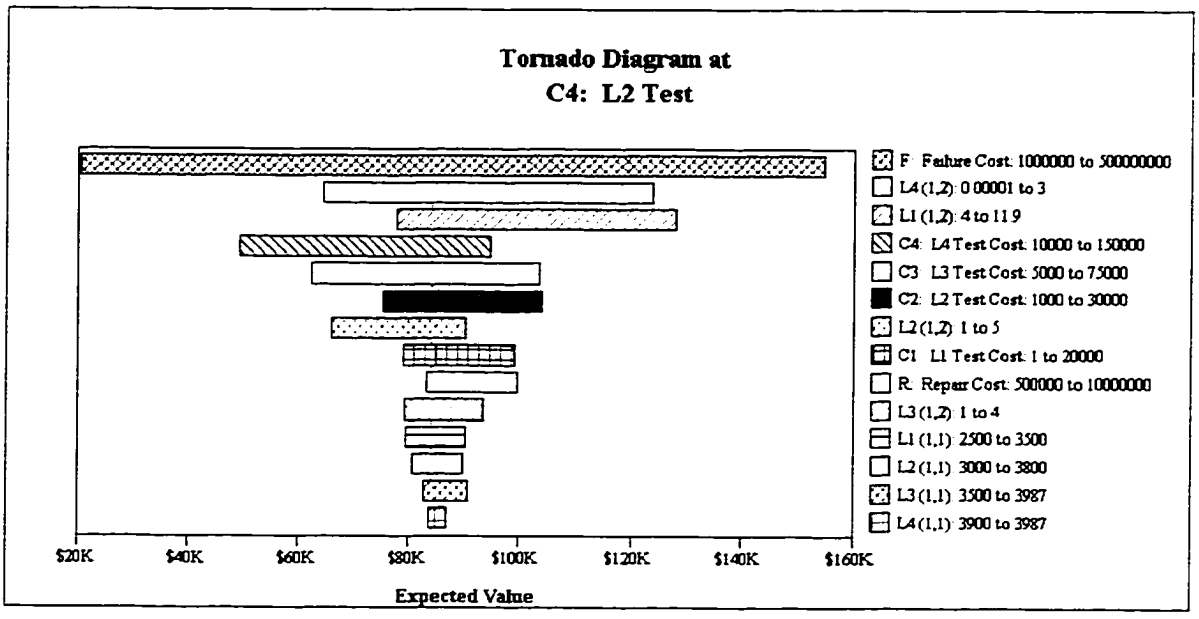


Figure 5-17: Tornado Sensitivity Analysis at Node C4

After an L2 test has returned a "safe" result, the analyst is faced with the question of whether or not to accept that result. This is node D2, which features a choice between "Accept L2 "Safe,"" "L3 Test," and "L4 Test." A tornado diagram is shown in Figure 5-18, and an F vs. C3 sensitivity diagram in Figure 5-19 explores Figure 5-18's indicated shifts in optimal policy.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

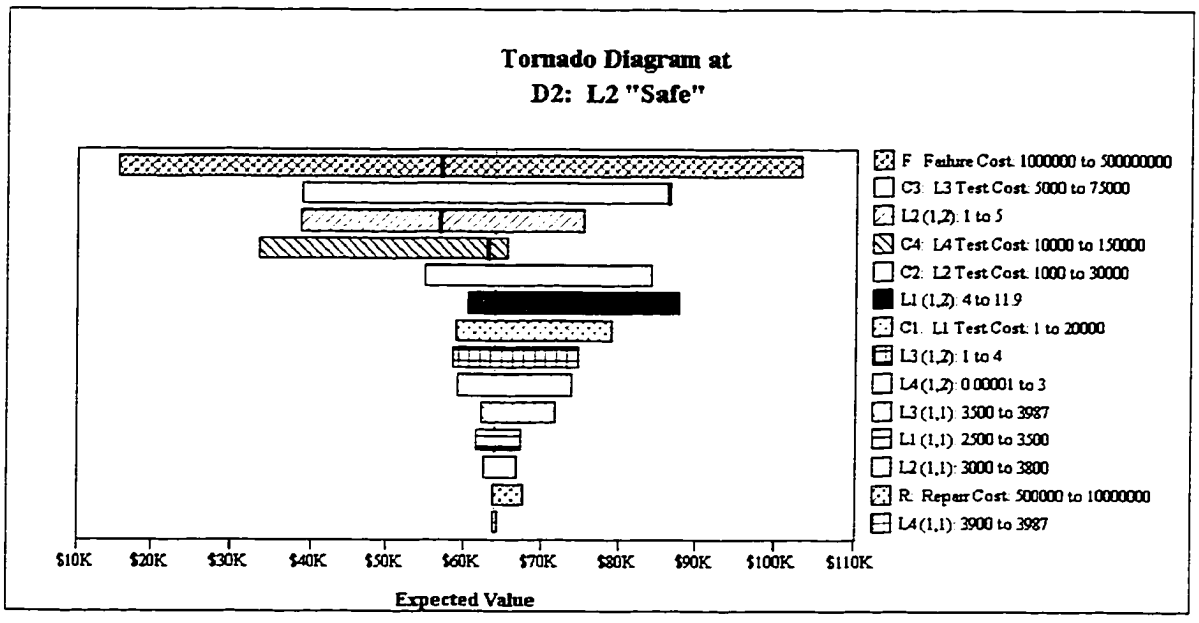


Figure 5-18: Tornado Sensitivity Analysis at Node D2

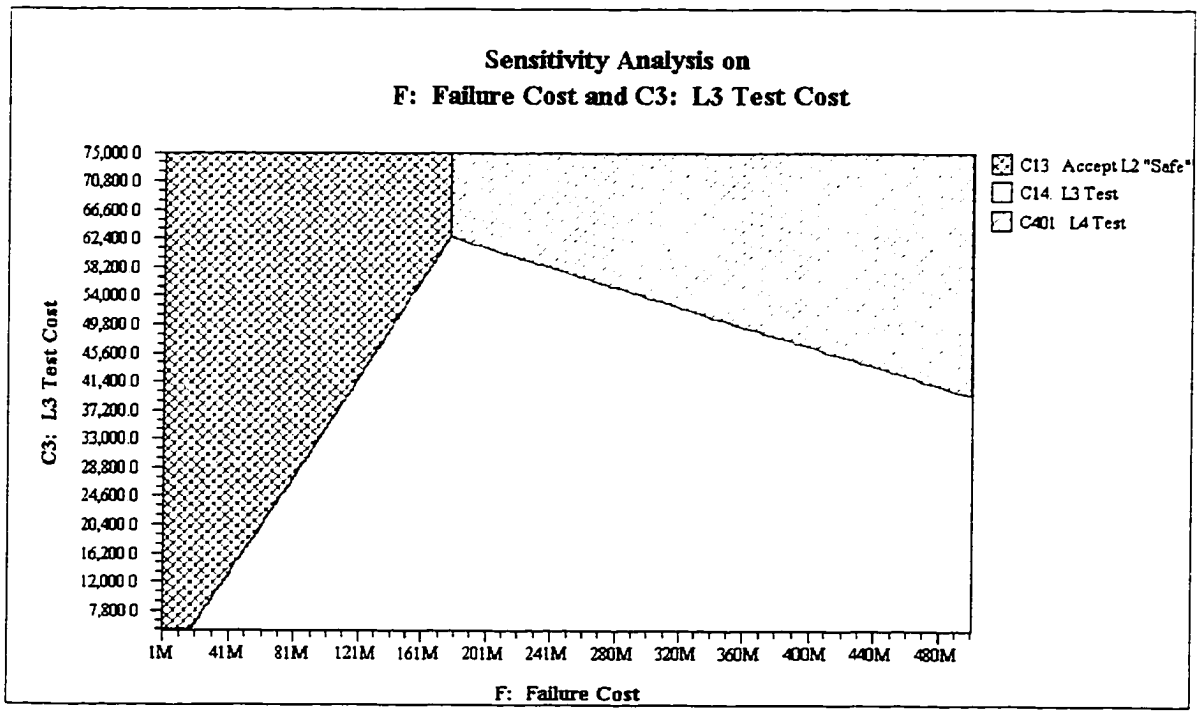


Figure 5-19: Two-Way Sensitivity Analysis of F and C3 at Node D2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 5-20 is a tornado sensitivity analysis at D3. At D3, tests L1, L2, and L3 have already occurred, so the factors listed in Figure 5-20 from C2 down don't have much effect on expected cost and have no effect on optimal policy. This is confirmed in Figure 5-21, in which C2 has no impact on the optimal decision.

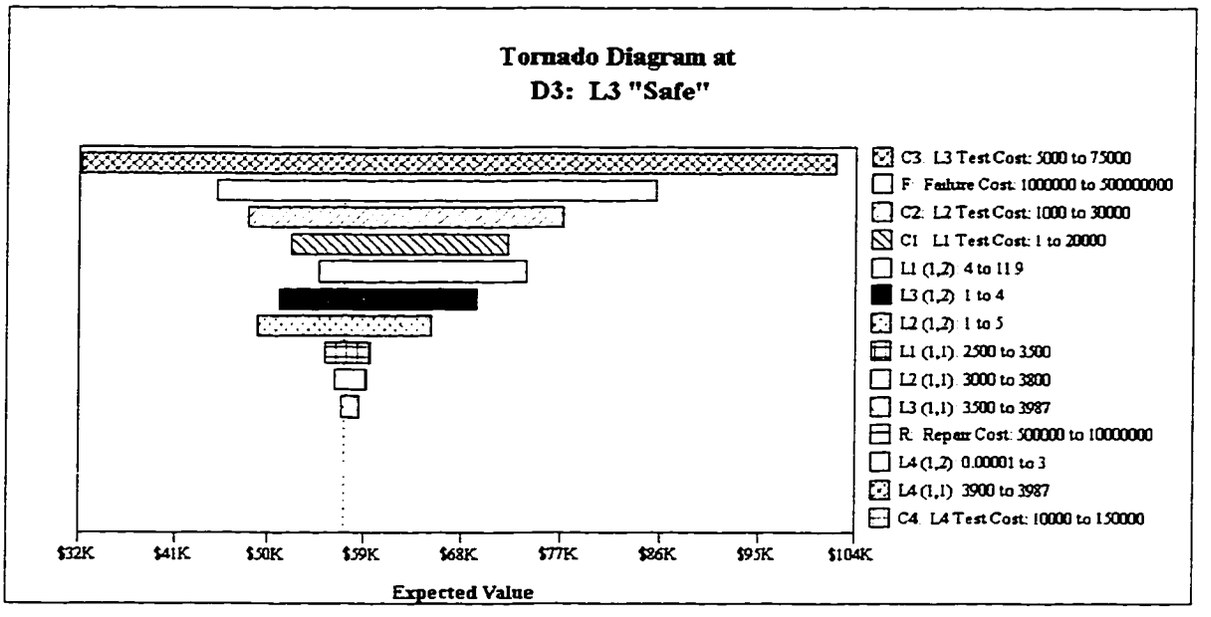


Figure 5-20: Tornado Sensitivity Analysis, Node D3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

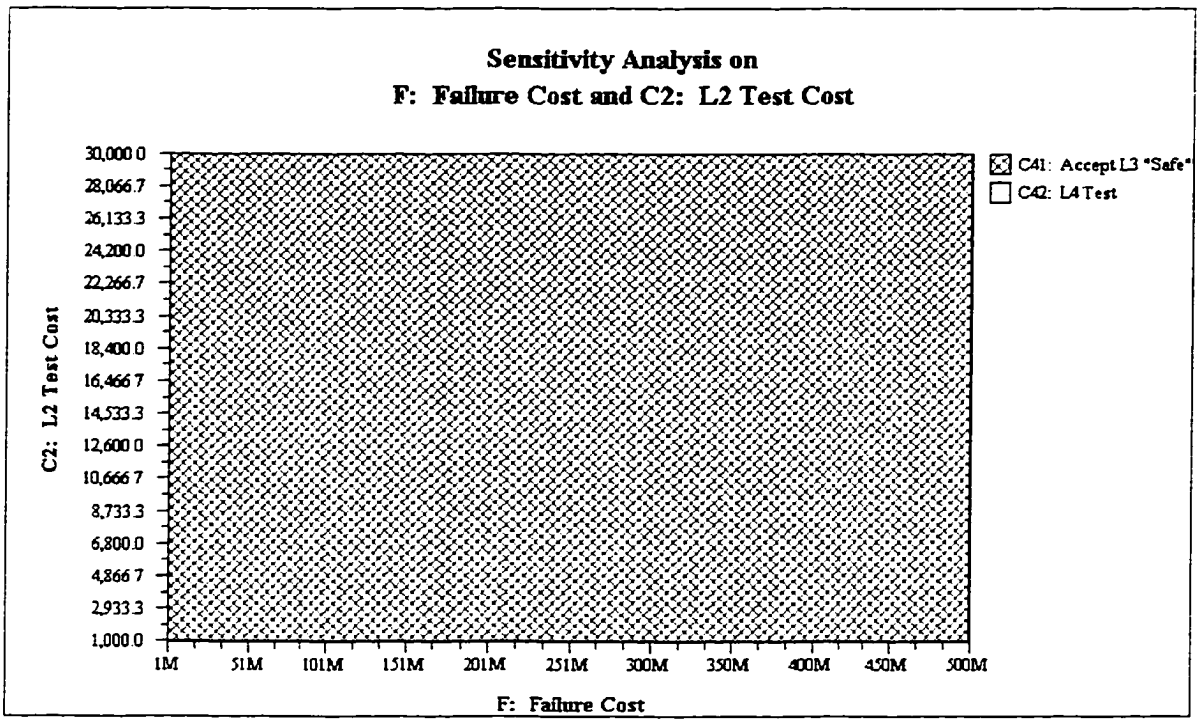


Figure 5-21: Two-Way Sensitivity of F and C2 at Node D3

Figure 5-22 is the tornado diagram at node C42. The cost of an L4 test, and the probability of an unsafe structure being labeled "safe," are beginning to have much more influence.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

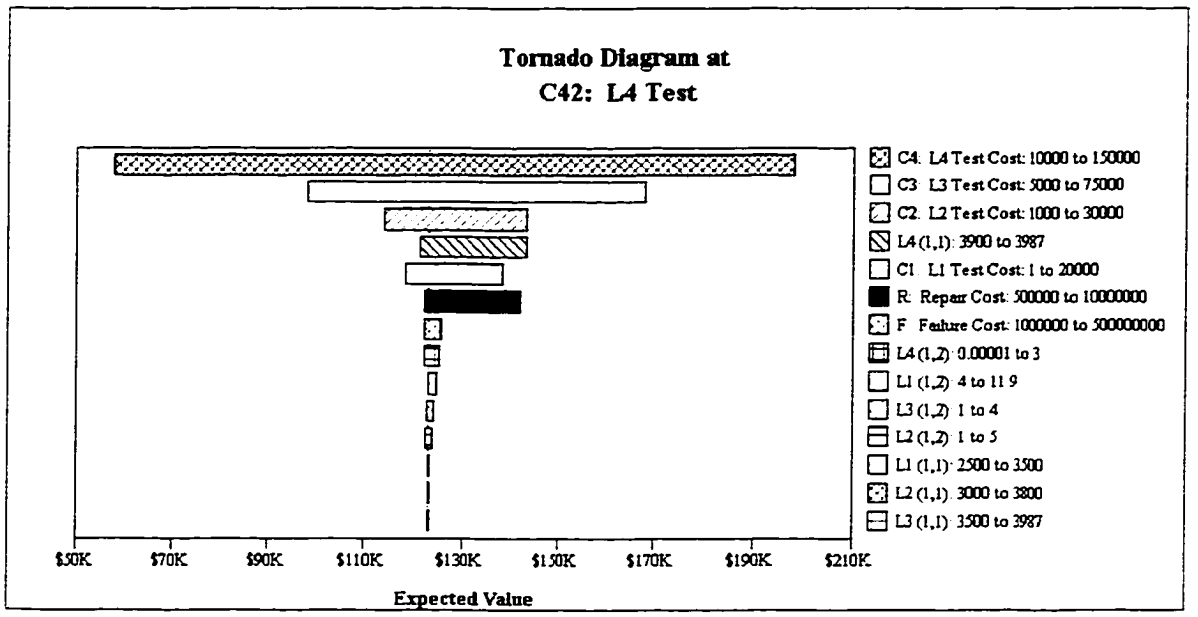


Figure 5-22: Tornado Sensitivity Analysis, Node C42

5.5 THE IMPACT OF TIME ON OPTIMAL POLICY

5.5.1 Expected Cost of Failure Over Time

Safety screening programs for large fleets of structures can't always be conducted on a yearly basis. Instead, a screening program must account for the anticipated period of time until the next cycle of evaluations. Thus, for a four-level screening program that occurs every *n* years, the test costs C1, C2, C3, and C4 occur only in the first year, while the failure cost F may be expected to occur at any time in the *n* years. (For the following discussion, it is assumed that repair costs will occur only in the first year.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

If a structure is expected to fail with cost F and annual probability P0, its expected cost of failure E(F) for an *n* year period is:

$$\begin{aligned} E(F) = {} & P0 \cdot F \\ & + P0 \cdot (1 + PFIR) \cdot F/(1 + IRR) \\ & + P0 \cdot (1 + PFIR)^2 \cdot F/(1 + IRR)^2 \\ & + P0 \cdot (1 + PFIR)^3 \cdot F/(1 + IRR)^3 \quad , \\ & + \ldots \\ & + P0 \cdot (1 + PFIR)^{n-1} \cdot F/(1 + IRR)^{n-1} \end{aligned} \tag{8}$$

or, summing up,

$$E(F) = P0 \cdot \left(\frac{(1 + PFIR)^n - 1}{PFIR} \right) \cdot F \cdot \left(\frac{1 - (1 + IRR)^{-(n-1)}}{IRR} \right), \tag{9}$$

where IRR is the evaluating organization's *annual internal rate of return*, and PFIR is the *probability of failure annual inflation rate* (an estimate of the increase in the overall fleet's probability of failure due to wear and tear, fatigue, and other aspects of normal aging). Unfortunately, the probabilities must be separated from the consequences in a decision tree, so Equation 9 cannot be used. The decision tree equivalent of Equation 8 is shown in Figure 5-23, which also includes the costs, ΣC_i, of all preceding tests:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

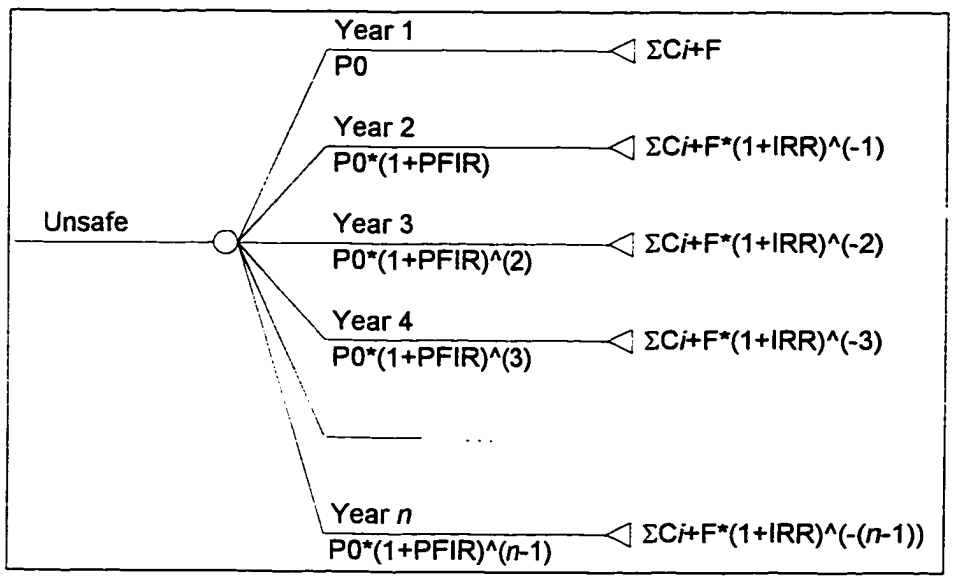


Figure 5-23: Decision Tree for Calculating Expected Failure Costs Over Time

Each of the branches of Figure 5-23 must also account for the possibility that, given a large value of P(U|"S"), a structure could fail before *n* years are reached. This would be indicated by a total probability for the sub-tree that exceeds 1.0, and is handled by probability statements of the following form:

```
If P0*(((1+PFIR)^n)-1)/PFIR>1 Then
    If P0*(((1+PFIR)^(n-1))-1)/PFIR>1 Then
        0
     Else
        1-P0*(((1+PFIR)^(n-1))-1)/PFIR
     End if
 Else
    P0*(1+PFIR)^(n-1))
 End if                                                    (10)
```

Incorporating Equations 8 and 10 into Model 5 requires replacing each result node having F consequences with a sub-tree of the form of Figure 5-23. The new tree, Model 5t, accommodates spans of up to 10 years (starting with year 1) and is shown in Appendix E.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.5.2 Sensitivity Analysis Incorporating Time

As seen in Figure 5-24, a tornado diagram assessed at the root node, time is now the single most influential factor in the model. The influence of defaults (in this case of variable Y, "years till screening") can be seen by comparing Figures 5-24 and 5-25. For example, test costs diminish in influence in Figure 5-25, while PFIR and IRR have no influence in Figure 5-24. Defaults for Model 5t are given in Table 5-3. Defaults for variables shared with Model 5 are retained. Model 5 is, indeed, Model 5t with Y set to 1. This can be seen by comparing the F through L4(1,1) bars of Figure 5-24 with Figure 5-13.

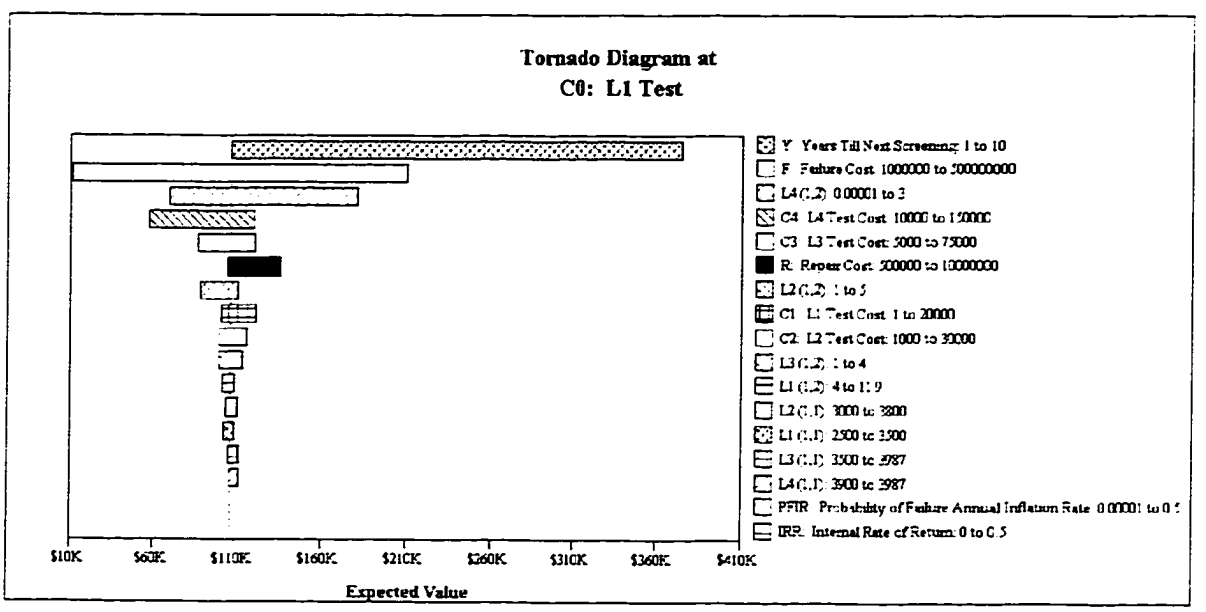


Figure 5-24: Tornado Sensitivity Analysis, Node CO (assessed at default Y=1)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

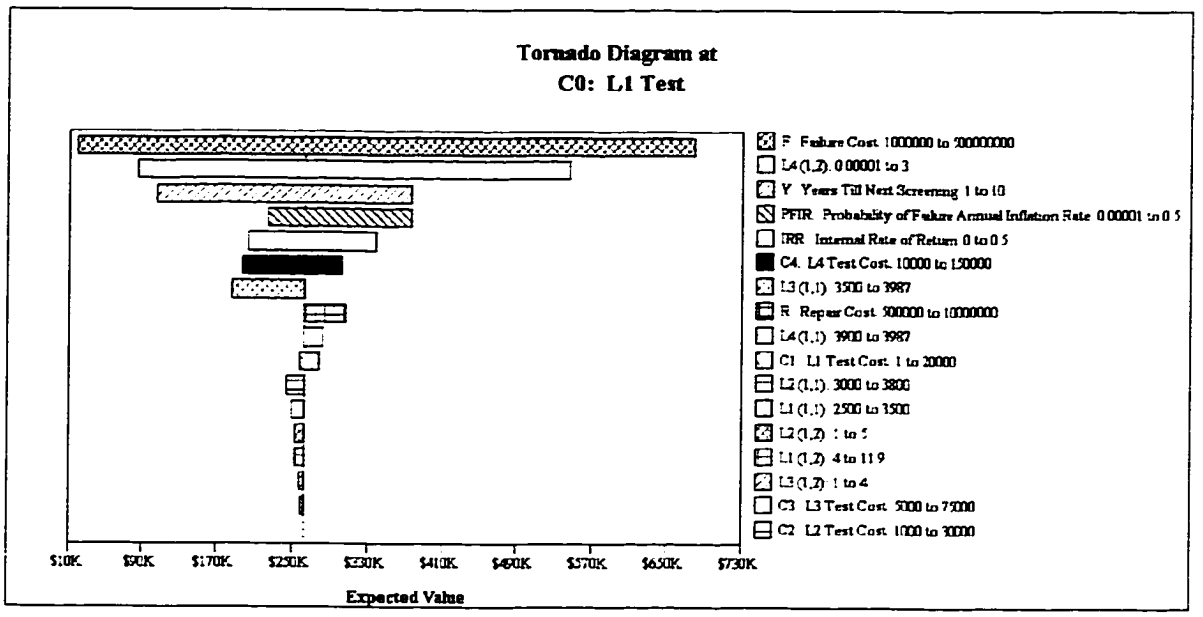


Figure 5-25: Tornado Sensitivity Analysis, Node C0 (assessed at default Y=5)

Name	Descrip	Low	High	Default
F	Failure Cost	$1,000,000	$500,000,000	$150,000,000
R	Repair Cost	$500,000	$10,000,000	$1,000,000
C1	L1 Test Cost	$1	$20,000	$5,000
C2	L2 Test Cost	$1,000	$30,000	$10,000
C3	L3 Test Cost	$5,000	$75,000	$30,000
C4	L4 Test Cost	$10,000	$150,000	$75,000
L1(1,1)	Safe Structures L1 "Safe"	2500	3500	3000
L1(1,2)	Unsafe Structures L1 "Safe"	4	12	5
L2(1,1)	Safe Structures L2 "Safe"	3000	3800	3500
L2(1,2)	Unsafe Structures L2 "Safe"	1	5	3
L3(1,1)	Safe Structures L3 "Safe"	3500	3988	3900
L3(1,2)	Unsafe Structures L3 "Safe"	1	4	2
L4(1,1)	Safe Structures L4 "Safe"	3900	3988	3980
L4(1,2)	Unsafe Structures L4 "Safe"	0	3	1
N	# of Structures	4000	4000	4000
D	Yearly # of Failures	12	12	12
IRR	Internal Rate of Return	0	0.5	0.18
PFIR	Probability of Failure Inflation Rate	0	0.5	0.15
Y	Years Till Next Screening	1	10	5

Table 5-3: Variable Default Values, Model 5t

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figures 5-26 and 5-27 show the influence of the time variable Y on decision making, juxtaposed with the failure cost variable F. Figure 5-26 applies to node D1, which occurs after an L1 test has judged a structure as "safe." However, the "accept L1 'Safe' result" alternative holds only for small values of F and small values of Y. For greater failure costs or greater spans of time between screens, an L1 "safe" result should never be accepted without further testing. A similar argument holds for L2 tests. The decision at D1 should be to proceed with the level of testing that is most economic – for moderate to high F, this decision will usually be to skip L2 and L3 tests and proceed directly with L4.

Figure 5-27 presents a similar situation at node D202, which occurs after an L1 test has judged a structure "unsafe." Thus, there is no "accept L1 test result" alternative – the model assumes that no structures will be repaired or decommissioned until after an L4 test has been performed.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

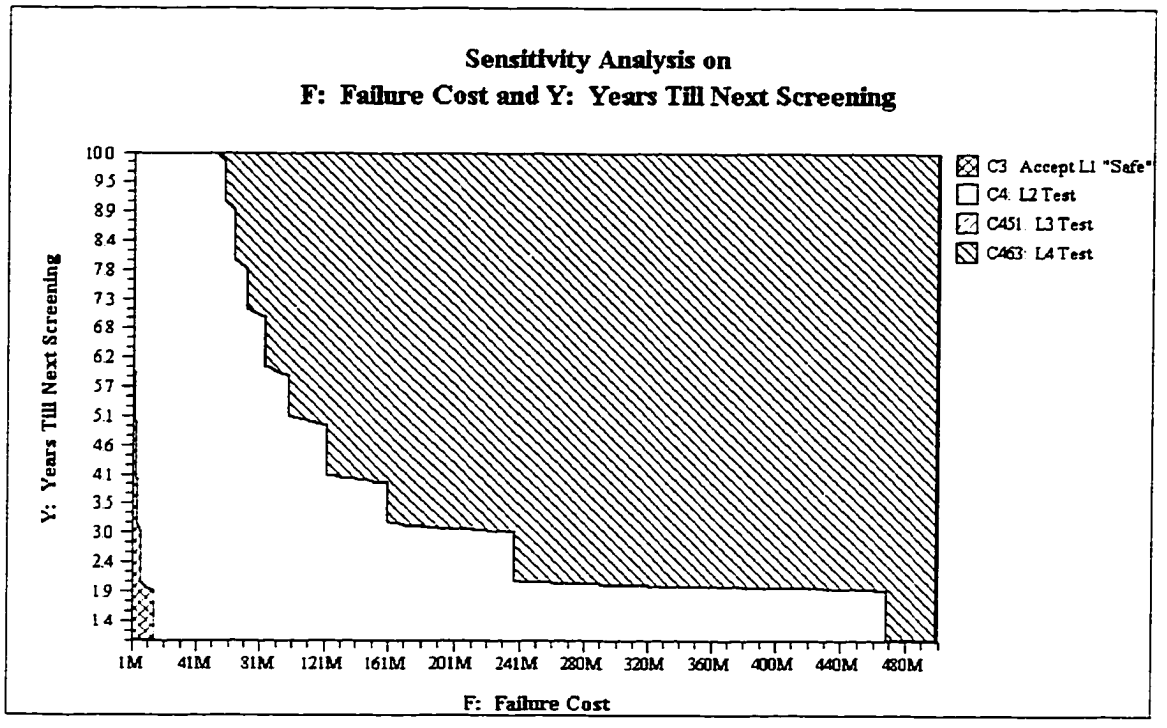


Figure 5-26: Two-Way Sensitivity Analysis of F and Y at D1

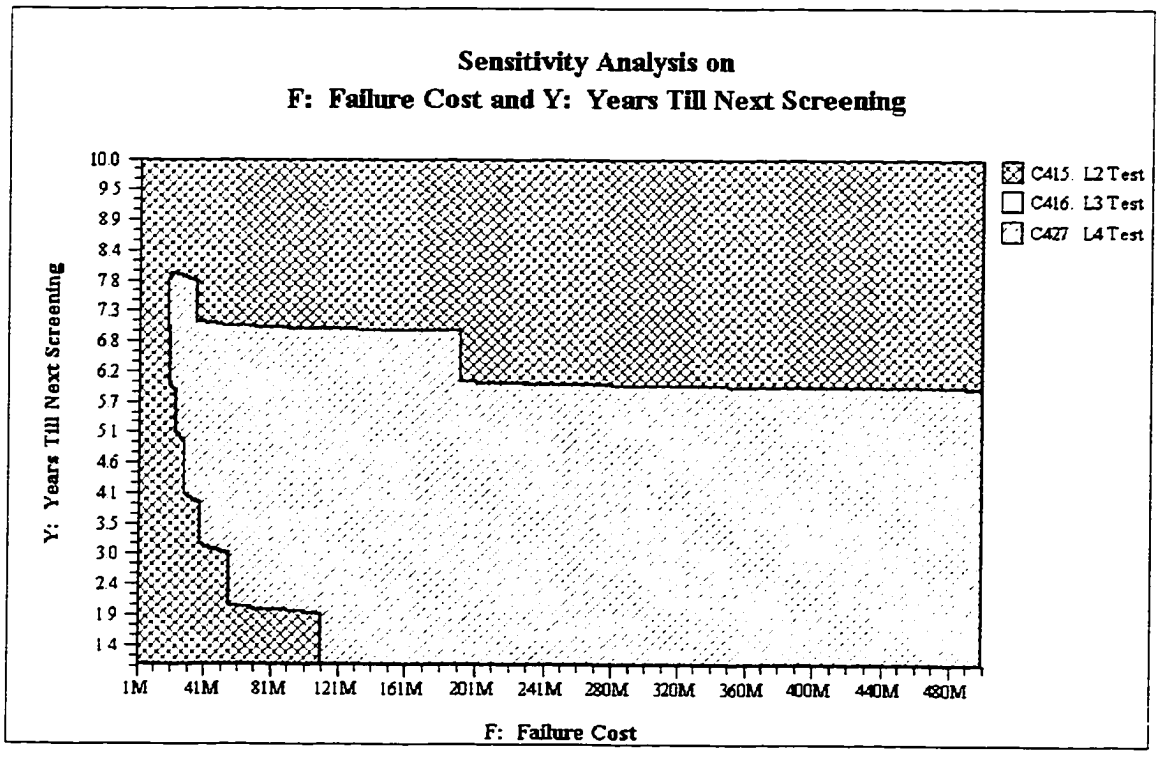


Figure 5-27: Two-Way Sensitivity Analysis of F and Y at D202

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The unusual pattern in Figure 5-27 is caused by the fact that the probabilities at node C129 indicate that the structure will fail by the seventh year (i.e., by the seventh year, the sum of the per-year probabilities of U|"S" will have equaled or exceeded one) – at the default value of PFIR. Thus, the optimal policy is pessimistic: while the structure, for most values of F, is clearly worth expending the cost of an L4 test on if Y is low, there is no point to it if an outlook of more than six or seven years is assumed. Changing the PFIR default from 0.15 to 0.00001 confirms this: reducing future failure probabilities moves the boundary upward to the nine-year neighborhood, as in Figure 5-28:

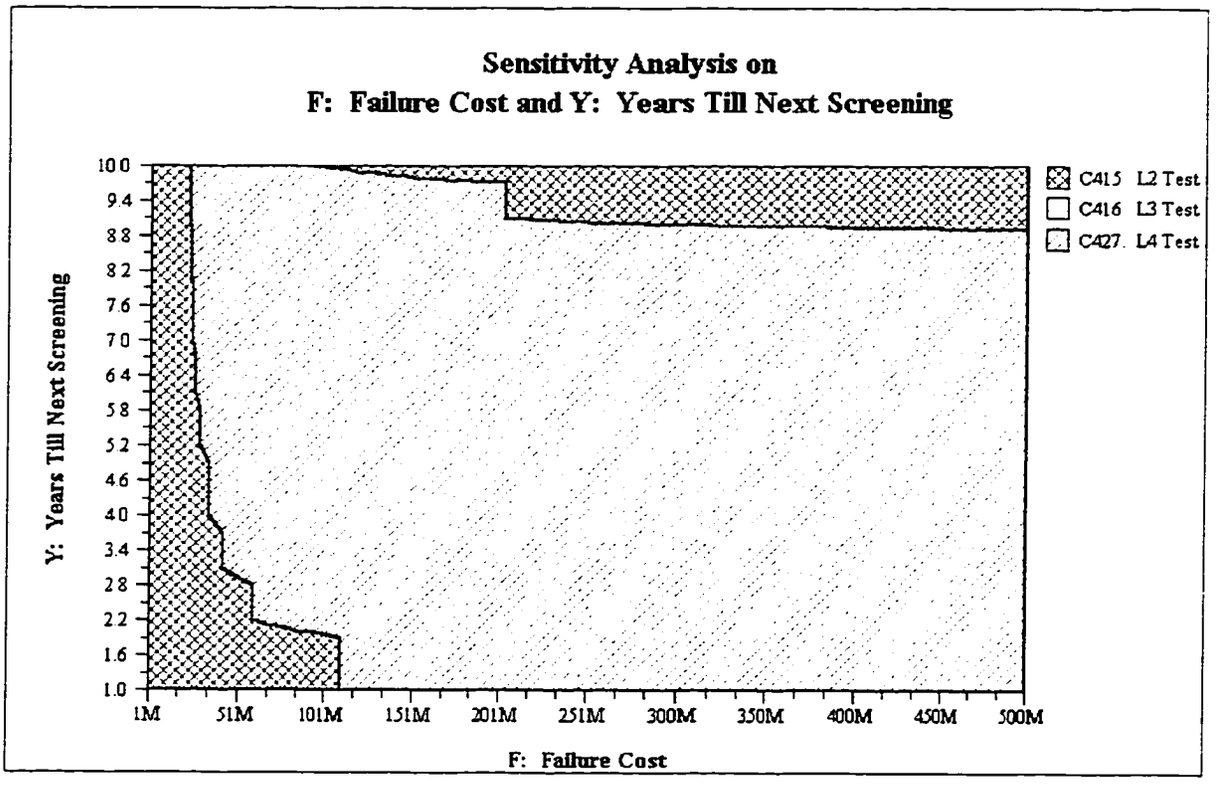


Figure 5-28: Two-Way Sensitivity of F and Y at D202, PFIR=0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 5-25 suggests that, holding Y at 5 years, the next variable having influence is L4(1,2). By the F matrix in Figure 5-10, the lower the value of L4(1,2), the lower the probability that an unsafe structure has been identified as "safe," and thus the better the quality of the test. Figure 5-29 illustrates the sensitivity of the decision at D1 to L4(1,2): given default values for all other variables, if F is between about $90 million and $110 million, and if the quality of the L4 test goes down (L4(1,2) increases), then the L2 test should be performed. But Figure 5-29 also shows that the decision's sensitivity to L4(1,2) is small: the value of L4(1,2) doesn't have much effect on the choice of optimal decision, except in the F = $91 million to $111 million range.

This principle applies throughout the multi-level screening cycle model: if the balance of quality (in the form of P(U|"S")) and cost are more attractive for a lower-level test than an upper-level, then there is no reason to perform that upper-level test. Figure 5-30, the two-way sensitivity of F and L1(1,2) at D1, is another illustration of the principle. Figure 5-31 shows the influence of test cost (in this case, C4) on decision-making.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

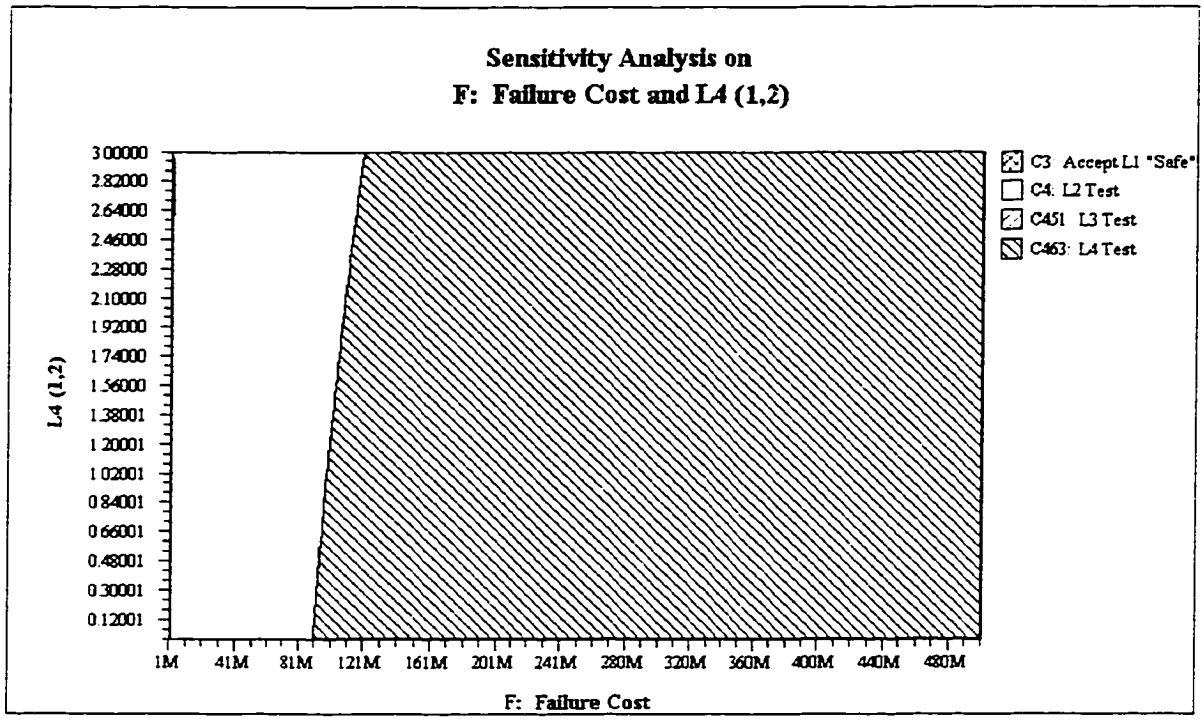


Figure 5-29: Two-Way Sensitivity Analysis on F and L4(1,2) at D1

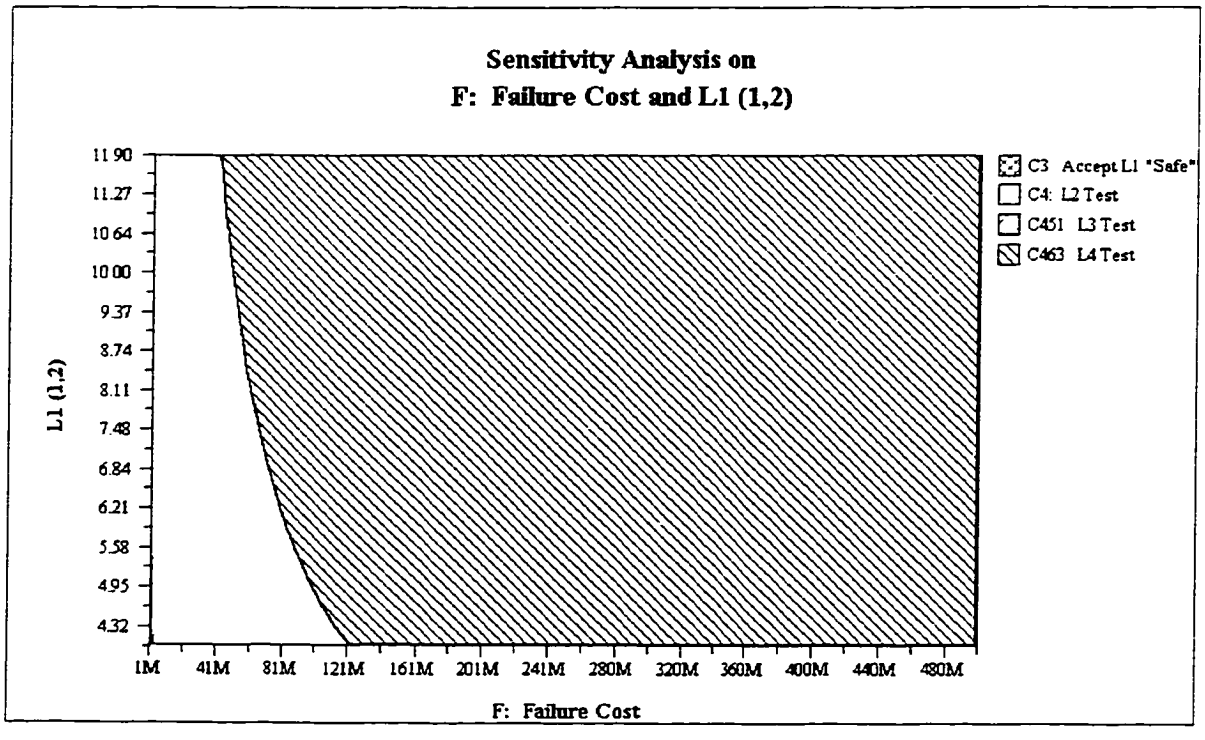


Figure 5-30: Two-Way Sensitivity Analysis on F and L1(1,2) at D1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

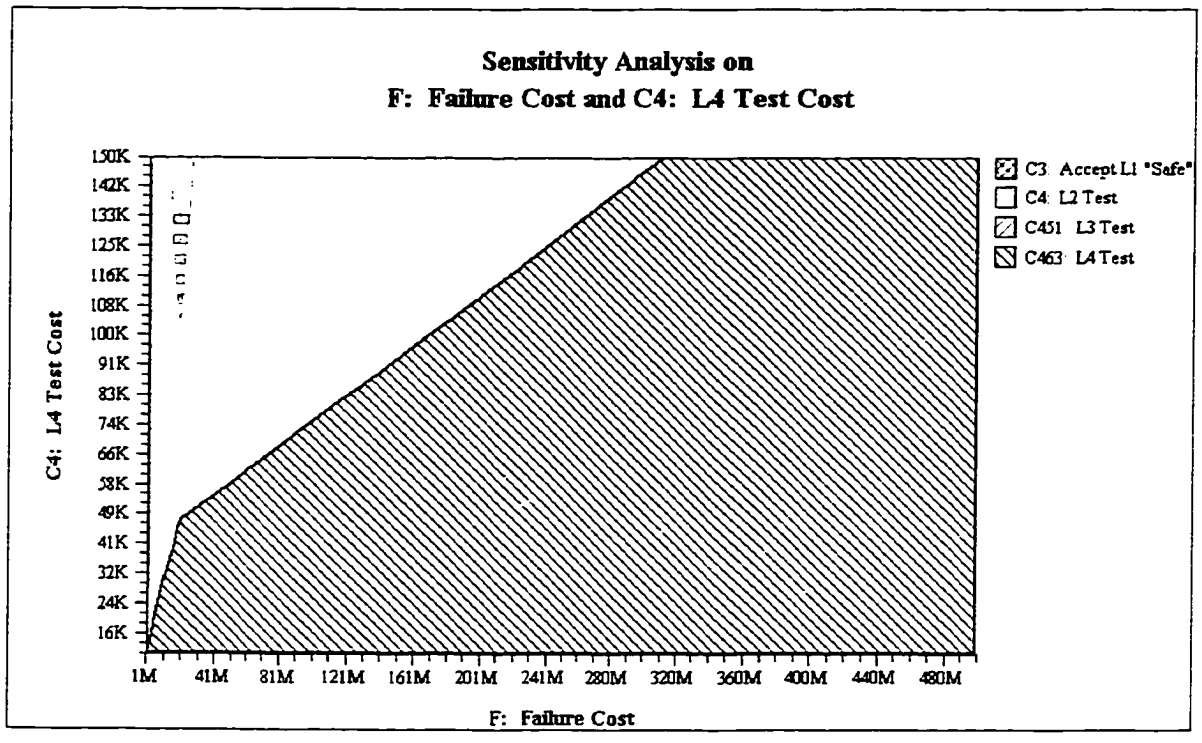


Figure 5-31: Two-Way Sensitivity Analysis on F and C4 at Node D1

Figure 5-32 shows a sensitivity analysis of F and PFIR, the probability of failure annual inflation rate, at D1. Since PFIR is probably very difficult to quantify with confidence, its relatively low influence on decision-making is fortunate.

Figure 5-33 is a similar analysis of IRR (an organization's required internal rate of return on investments). While PFIR is a characteristic of a structural type, IRR is a characteristic of perspective. Thus, two different organizations (perhaps a regulator and an owner) screening the same fleet of structures would use the same value of PFIR but perhaps different values of IRR.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

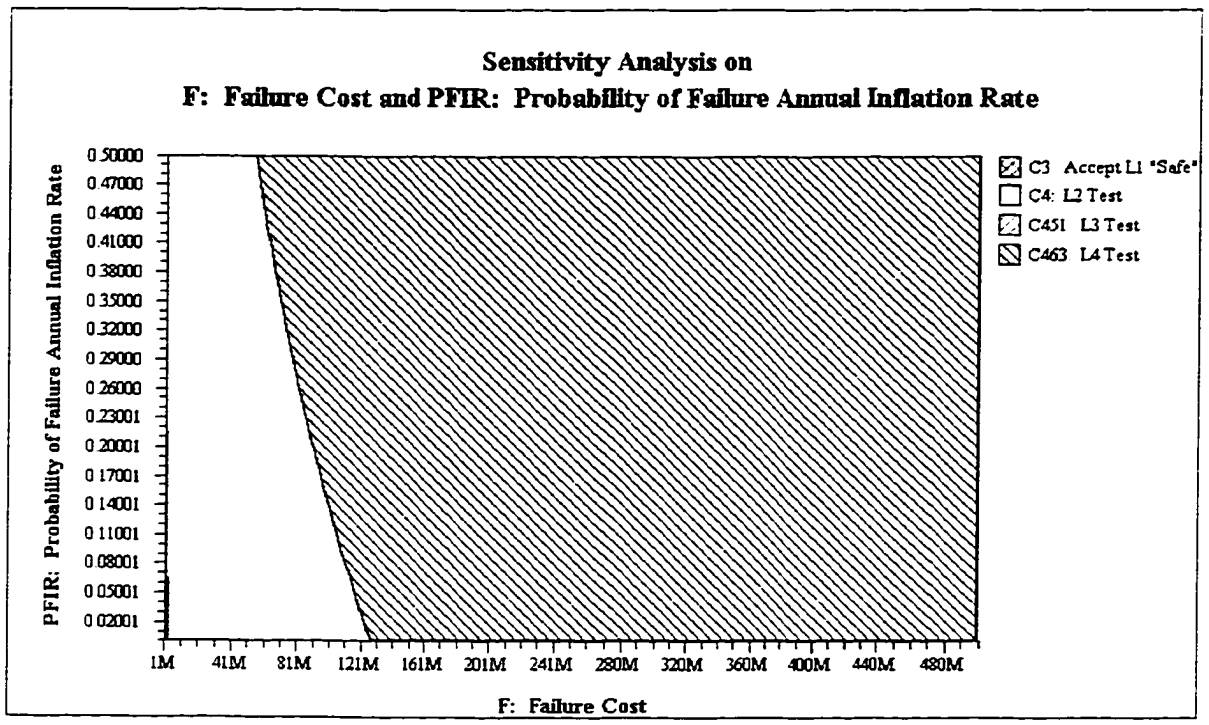


Figure 5-32: Two-Way Sensitivity Analysis on F and PFIR at D1

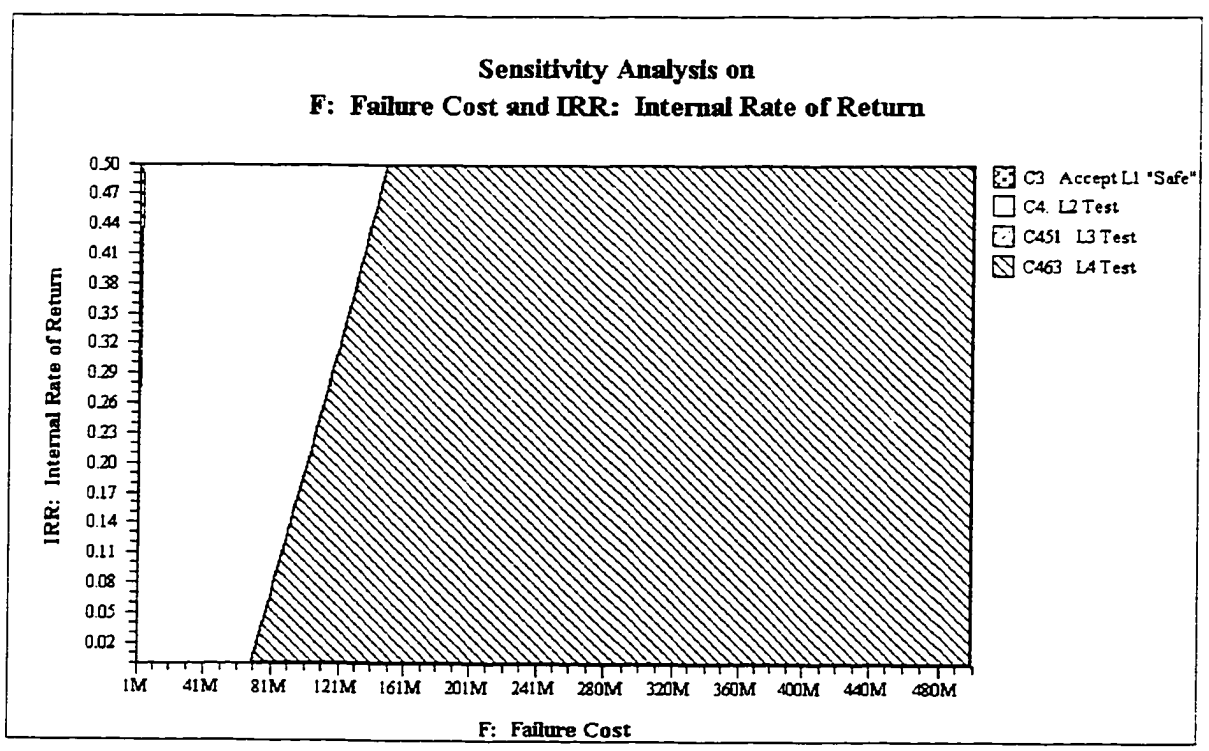


Figure 5-33: Two-Way Sensitivity Analysis of F and IRR at D1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Looking at other measures of test quality, Figure 5-34 shows the impact of L4(1,1) – the number of safe structures expected to be labeled "safe" by L4. The default value of L4(1,1) (3980) is so close to the maximum possible value (3988) that the L3 test is rarely optimal at D1. As L4(1,1) decreases (thus, the quality of the L4 test decreases), L3 is more often optimal. In the tornado diagram of Figure 5-35, L4(1,1) is seen to have minimal impact. Even the second most-influential variable, L4(1,2), is seen to have an impact only on the expected cost, and not on the optimal decision (there are no vertical lines in the L4(1,2) bar).

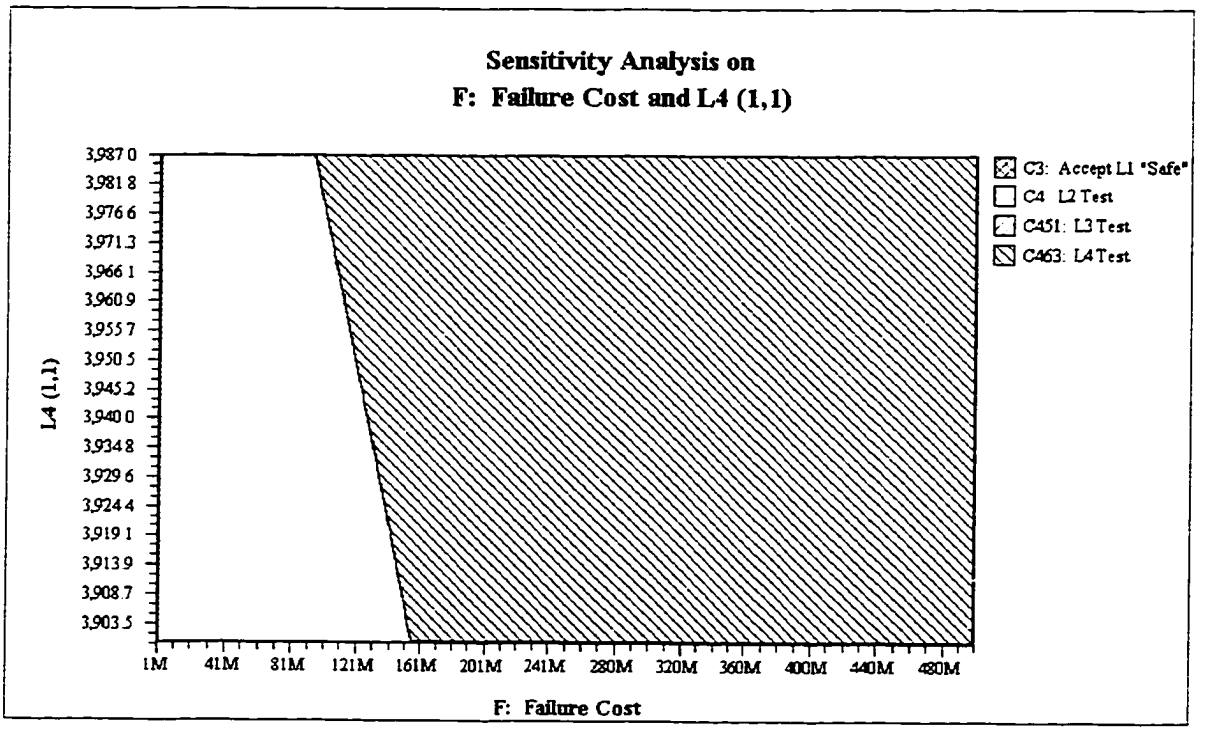


Figure 5-34: Two-Way Sensitivity Analysis of F and L4(1,1) at D1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

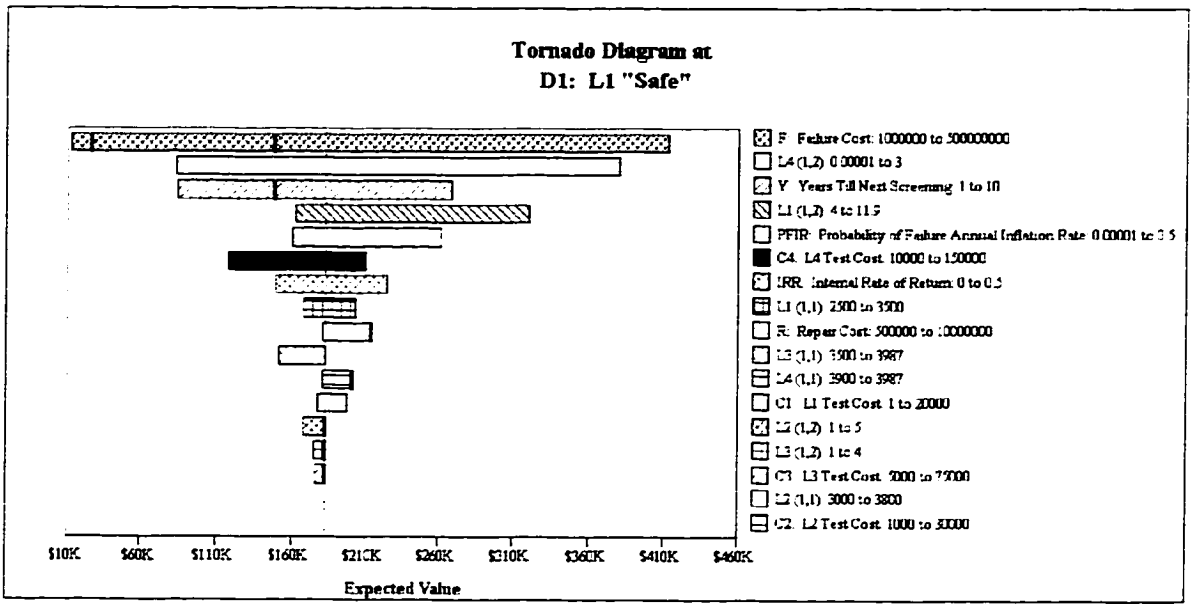


Figure 5-35: Tornado Sensitivity Analysis, Node D1

5.6 CONSERVATISM IN SCREENING SYSTEM DESIGN

The decision models of Sections 5.2 through 5.5 have all assumed that as test levels grow more complex and more expensive, they also grow more accurate and more conservative (i.e., the test quantity (1,1) goes up while quantity (1,2) goes down – indicating that the probability of a safe structure being labeled as "safe" goes up, while the probability of an unsafe structure being labeled "safe" goes down). While it is not desirable for a series of tests to become less accurate as they become more complex, it may be reasonable to design a screening system such that the tests become less conservative as they increase in complexity. Bea [1993] describes such a system. The basic idea is that the first levels of tests are so intentionally conservative that any structure that manages to achieve a "safe" rating is indeed safe. The increased conservatism is achieved by setting higher implicit

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

safety standards for the first test levels, decreasing gradually to the "real" standard by the time the final (Level 4) tests are reached.

To examine the effects of decreasing test conservatism, Model 5t was altered to create Model 5c. Compared to Model 5t, the ranges and default values for Model 5c's L1(1,2) through L4(1,2) estimates were switched, while all other values were retained. Table 5-4 shows the default values for Model 5c.

Name	Descrip	Low	High	Default
F	Failure Cost	$1,000,000	$500,000,000	$150,000,000
R	Repair Cost	$500,000	$10,000,000	$1,000,000
C1	L1 Test Cost	$1	$20,000	$5,000
C2	L2 Test Cost	$1,000	$30,000	$10,000
C3	L3 Test Cost	$5,000	$75,000	$30,000
C4	L4 Test Cost	$10,000	$150,000	$75,000
L1(1,1)	Safe Structures L1 "Safe"	2500	3500	3000
L1(1,2)	Unsafe Structures L1 "Safe"	0	3	1
L2(1,1)	Safe Structures L2 "Safe"	3000	3800	3500
L2(1,2)	Unsafe Structures L2 "Safe"	1	4	2
L3(1,1)	Safe Structures L3 "Safe"	3500	3988	3900
L3(1,2)	Unsafe Structures L3 "Safe"	1	5	3
L4(1,1)	Safe Structures L4 "Safe"	3900	3988	3980
L4(1,2)	Unsafe Structures L4 "Safe"	4	12	5
N	# of Structures	4000	4000	4000
D	Yearly # of Failures	12	12	12
IRR	Internal Rate of Return	0	0.5	0.18
PFIR	Probability of Failure Inflation Rate	0	0.5	0.15
Y	Years Till Next Screening	1	10	5

Table 5-4: Variable Default Values, Model 5c

Roll-backs were performed to find the expected per-structure cost of each model, at various years Y until the next screening. Costs for Model 5c averaged almost 2.5 times those of Model 5t, and are shown Table 5-5 in thousands of dollars:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Y=	1	2	3	4	5	6	7	8	9	10
Model 5t	107	151	193	229	263	296	317	337	356	375
Model 5c	232	401	481	558	634	708	779	849	916	980

Table 5-5: Expected Costs, Increasing vs. Decreasing Conservatism, $ Thousands

Decreasing test conservatism with increasing complexity means that, in Model 5c, the tests become less discriminatory as they become more complex. While some of the various probabilities P(U|"S") were reduced (e.g., at node C3, P(U|"S") fell from 0.00166 to 0.00033), others increased (node C129's P(U|"S") increased from 0.1204 to 0.5183). The quantities of unsafe structures that can be expected to be judged "safe" at the result nodes are larger for Model 5t at nodes D1, C461, D2, and D3, but the quantities at all other result nodes are larger for Model 5c. In all, 77% more unsafe structures can be expected to be labeled "safe" by Model 5c than by Model 5t, using the example default values.

Of course, the design of a screening system rests on the tests available for its use. If a choice is available, however, the choice of ascending or descending test conservatism with increasing cost and complexity should be thoroughly investigated.

5.7 ECONOMIC DECISION MODEL INTEGRATION INTO SCREENING SYSTEM MODEL

The flowcharts of Figures 36 and 37 build on Figure 5-4 (omitting structure status indicators and break points) and incorporate the decision model of Section 5.5. Economic evaluations are used to decide whether to accept a "safe" test result, and if not, to which

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

level of testing to proceed. Per Section 5.4.2, "marginal" results are eliminated from the flowcharts. Also illustrated is a mechanism by which an iteration can move from an *i*-level test to an *i-n* level test, where $n = 1$ or greater (within the 4-level limit). This is useful to check assumptions: for example, if an engineer working at the third level of testing wishes to make a major repair to a structure, but would like to do an inexpensive check before proceeding with an expensive Level Three reassessment, the engineer may go back to the first or second level.

Figures 36 and 37 are not meant to imply that the work on an individual structure must proceed until a final result is reached. Indeed, an ideal opportunity for work prioritization occurs after a "safe" test result is found to be economically undesirable: the structure may then be set aside in the reasonable expectation that it presents no immediate problems, while other, possibly more dangerous, structures are examined. The original structure must not be forgotten, however: eventually, its progress through the screening process should be resumed.

One issue not discussed earlier is that of selecting a structure for screening. The method for doing this will vary by the type of structure and the type of fleet. It will likely be based on engineering judgment, on API-style triggers [API 1994], or on simplified assessment criteria of the type discussed in Section 4.3.2.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

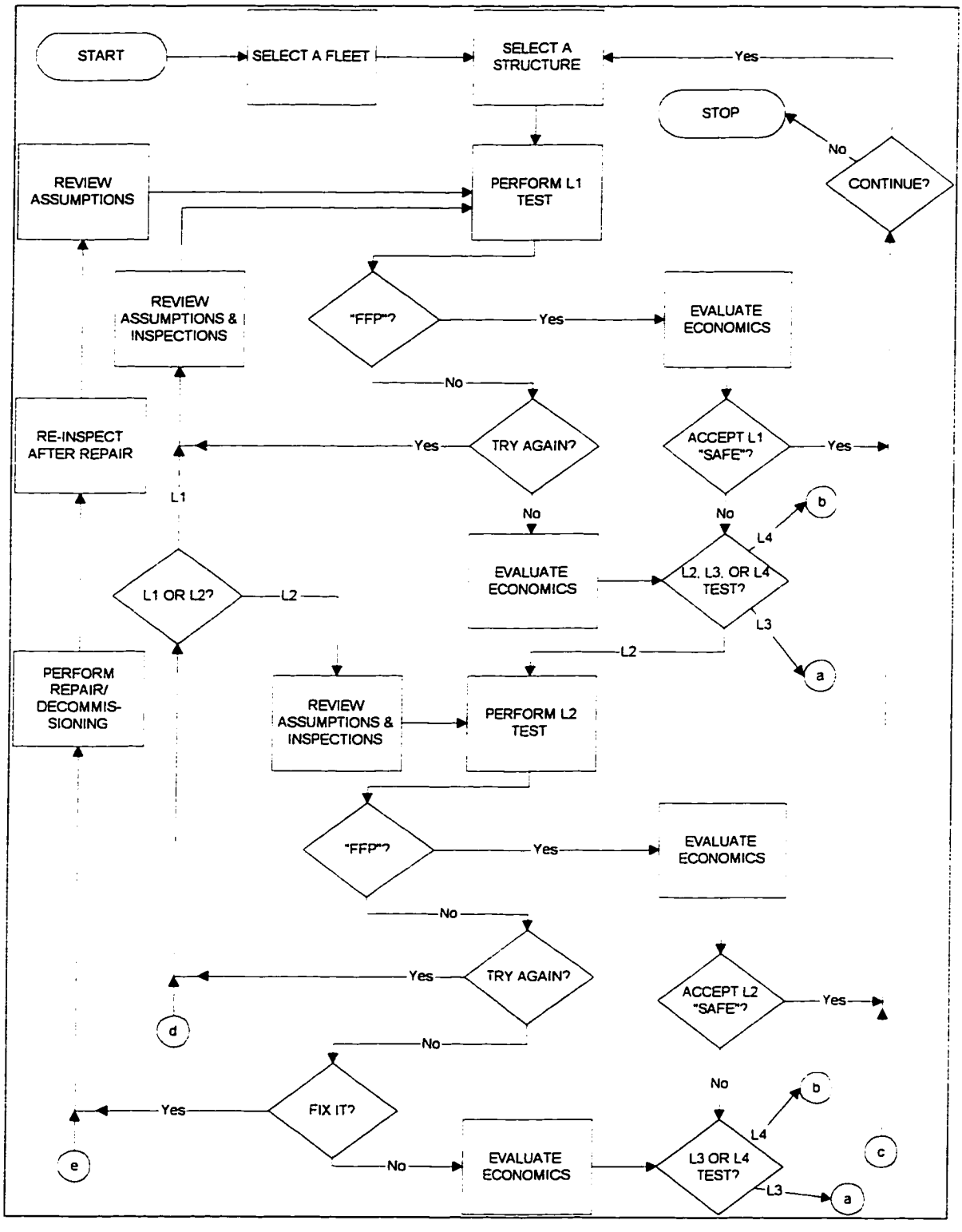


Figure 5-36: Screening Flowchart Incorporating Economic Decision Model, part A

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

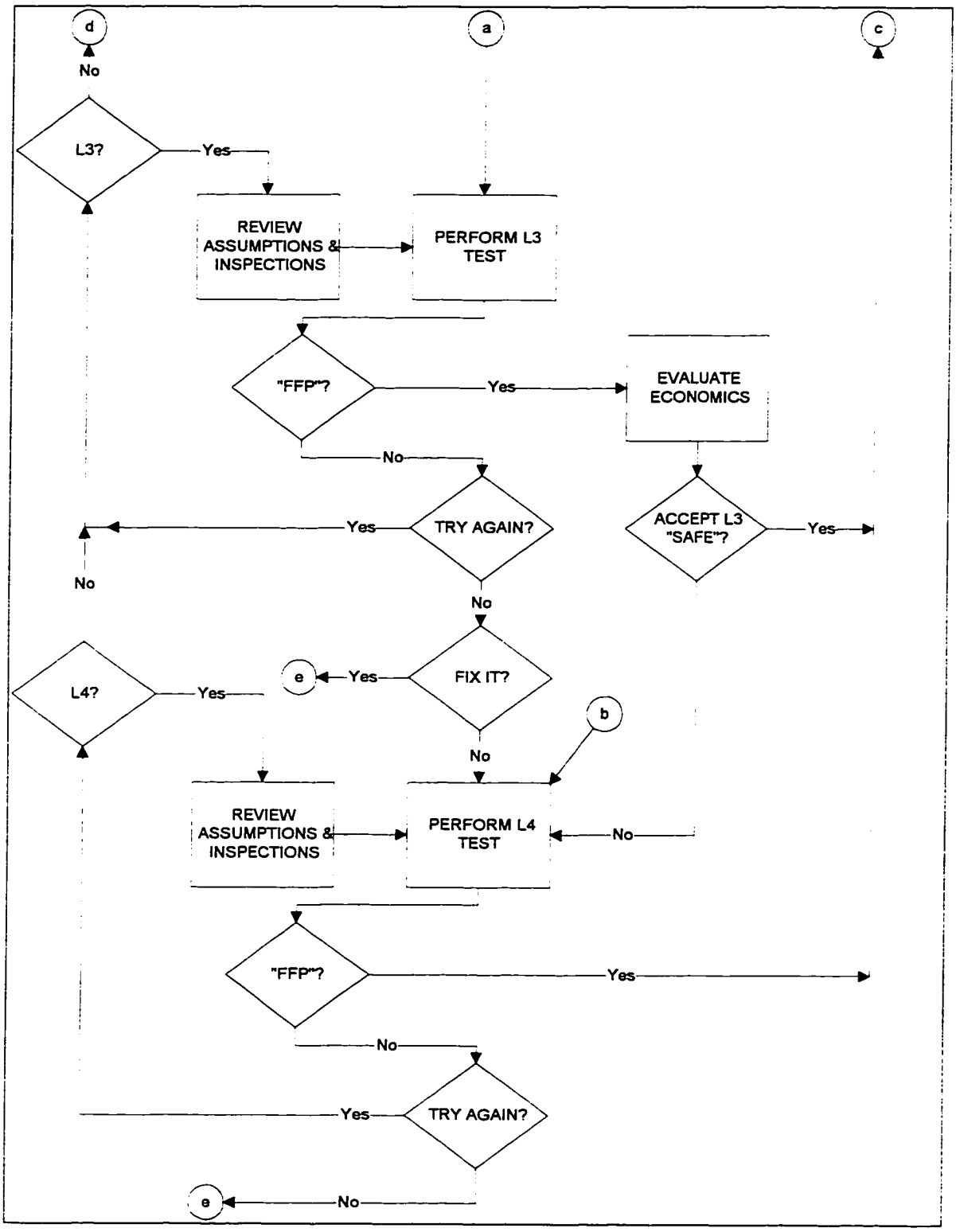


Figure 5-37: Screening Flowchart Incorporating Economic Decision Model, part B

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.8 INTEGRATION INTO DAESSIM

The screening cycle model described in this chapter can serve as an implementation of the DAESSIM model. Integration of the model into DAESSIM would proceed as either the establishment of DDE (Microsoft's Direct Data Exchange™ protocol) links with a decision analysis program such as TreeAge Software's DATA™, or as conversion to a fully-incorporated DAESSIM scenario type. The latter process is sketched below. Estimates of the time required to complete each task are given as examples of working with DAESSIM, and sum to about seventy-five hours.

5.8.1 Coding

For integration as an scenario type in DAESSIM, the decision tree should be coded in the Access Basic language. Since there is no need to modify the tree's structure (by adding, deleting, or changing nodes), the coding process would be relatively simple. An example based on matrix calculations would be:

```
'Node R465
CMatrix(4, 465, 0, 1, 0, 0, 0) = (CMatrix(1, 463, 0, 0, 0, 1, 1) * L4FMatrix(2, 1) + CMatrix(1,
    463, 0, 0, 0, 1, 2) * L4FMatrix(2, 2)) / (CMatrix(1, 463, 0, 0, 0, 1, 1) + CMatrix(1, 463, 0,
    0, 0, 1, 2))
CMatrix(4, 465, 1, 0, 0, 0, 0) = (R + C1 + C4)

'Node C464
CMatrix(1, 464, 0, 0, 0, 1, 1) = CMatrix(2, 1, 0, 0, 0, 1, 1) * L4FMatrix(1, 1)
CMatrix(1, 464, 0, 0, 0, 1, 2) = CMatrix(2, 1, 0, 0, 0, 1, 2) * L4FMatrix(1, 2)
CMatrix(1, 464, 0, 1, 0, 0, 0) = (CMatrix(1, 463, 0, 0, 0, 1, 1) * L4FMatrix(1, 1) + CMatrix(1,
    463, 0, 0, 0, 1, 2) * L4FMatrix(1, 2)) / (CMatrix(1, 463, 0, 0, 0, 1, 1) + CMatrix(1, 463, 0,
    0, 0, 1, 2))
Prob = CMatrix(1, 464, 0, 0, 0, 0, 1, 2) / (CMatrix(1, 464, 0, 0, 0, 0, 1, 1) + CMatrix(1, 464, 0,
    0, 0, 1, 2))
CMatrix(1, 464, 1, 0, 0, 0, 0) = TimeResultNode(C1 + C4, F, IRR, Prob, PFIR, Y)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In the above, the first element of CMatrix denotes the node type (1 for chance, 2 decision, 4 for result, etc), the second element denotes the node number, the third stores the expected value at the node, the fourth stores the probability of the node, the fifth stores the number of the following node that provides the optimal path, and the sixth and seventh store the expected number of safe and unsafe structures at the node. Starting with the root node and working "rightward" in the tree, the sixth and seventh elements of CMatrix would be calculated. Then, working "leftwards" from the result nodes, the third, fourth and fifth elements would be computed (a "rollback" procedure). Preliminary testing indicates that the coding described here would require about fifty hours of repetitive work.

Also required for integration into DAESSIM would be a method for interrupting the decision and chance processes. For example, if a structure has been evaluated by an L1 test as "safe," and this result is made known to the screening cycle procedure (see Section 5.8.2 below), then the probability of L1 "safe" at node C0 is 1.0, and the L1 "unsafe" branch cannot occur. Similarly, an economic recommendation at a decision node may not necessarily be followed, and the screening cycle must be made aware of this. Coding for these tasks will probably require about eight hours.

5.8.2 Inputs

Necessary inputs to the decision tree, to be declared as factors for the DAESSIM scenario type, would be N, D, L1(1,1), L1(1,2), L2(1,1), L2(1,2), L3(1,1), L3(1,2), L4(1,1), L4(1,2), PFIR, IRR, F, R, and Y. Other inputs would be the ID's of prior evaluations to

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

be used in the screening cycle. For example, if a structure is being screened using an L1 test, and the potential for performing that test exists in DAESSIM, then the screening cycle must ask the user to pick an L1 evaluation instance to be used. This is a routine function of scenarios in DAESSIM. Since the tasks described in this section are normal functions of DAESSIM, execution time will be minimal (perhaps an hour).

5.8.3 Outputs

The outputs from a screening cycle scenario type would be, primarily, the "structure status" information shown in Figure 5-4. However, the results of each test would be stored in the normal course of performing the individual evaluations, and that information could later be used to refine estimates of L1(1,1) through L4(1,2) (possibly through Bayesian procedures).

5.8.4 Interface

A graphical interface would work well for a screening cycle model. One possibility is to create an Access™ form based on Figures 36 and 37. Clicking the mouse on a line of text in each flowchart symbol would activate the screening cycle code – to change engineering assumptions, to proceed to a Level Four test, to accept "safe" results (based on DAESSIM recommendations), etc. Creating such a form would probably require about sixteen hours to complete.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.9 OTHER APPLICATIONS

Besides the actual work of infrastructure screening, the decision model described in Sections 5.2 through 5.5 has other uses. Some of these are:

- If a regulator decides to sponsor the development of a 'perfect' Level Five test (a "crystal ball" which always correctly predicts safety; i.e., is perfectly discriminatory),

  How much should be spent in test development?

  How much should each execution of the test cost?

- If an owner decides to improve the test procedure for a certain level,

  How much should be invested in test development?

  How much should each execution of the test cost?

- If many tests are available when a screening system is being designed, which ones should be chosen for each level of testing?

- What are the impacts of changing a test mid-way through the evaluation of a fleet?

High-consequence structures such as dams and oil platforms are not the only beneficiaries of the screening cycle model. Others might include:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Houses subjected to earthquake loading

- Public buildings subjected to earthquake loading

- Wharves and piers

- Railroad bridges

A screening model need not necessarily use monetary consequences. For example, an offshore oil industry regulator may wish to use as a failure criteria the number of lives that would be lost in failure events. Another option in such a situation is a multi-attribute model: the number of lives at risk as one criteria, and environmental consequences as another.

5.10 SUMMARY

This chapter presents a screening system for infrastructure fleets that allows the user to employ common sense and economics to move to another structure, to iterate a present analysis on the present structure using different assumptions, or to decide whether to accept a "safe" judgment for a structure. The latter is analyzed in detail, including aspects such as the necessity of calculating (rather than estimating) conditional probabilities for a multi-level screening system, analyzing the impact of the ability to skip from an i-level test to an i+2 or i+3 level test, and accounting for the time lag between screens. The model's variables and assumptions are examined for influence and validity. For example,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

failure cost, the quality of a Level Four test, and the number of years until the next screening are found to be the most influential factors upon the decision whether or not to continue testing after receiving a Level One "safe" result in a four-level system assessed at the default values. The assumption that screening system tests should decrease in conservatism with increasing complexity and cost is challenged, and found to be false in one case. The assumption thus requires investigation when screening systems are designed.

Integration of the screening cycle model with DAESSIM is explored, and a number of alternate uses for the model are suggested. These include deciding how much to spend on improving testing procedures, and applying the screening model to various types of infrastructure systems.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6 PLATFORM INFORMATION MANAGEMENT SYSTEM (PIMS) OPERATIONS

6.1 INTRODUCTION

The Platform Information Management System (PIMS), an instance of the Decision, Analysis, and Engineering Support System for Infrastructure Management (DAESSIM), is a system for the management of data for the requalification and reassessment of offshore platforms. PC-based, it is adaptable to nearly any sort of engineering analysis while maintaining data integrity. It is designed to be capable of examining not only individual structures, but large collections (fleets) of structures.

PIMS is based upon physical structures, evaluations (analyses) upon those structures, fleets of structures, and scenarios (analyses) upon those fleets. Additionally, PIMS distinguishes between types and individuals - e.g., a building is a type of structure, while building T-1 on the Berkeley campus is an individual building, and the collection of "temporary buildings" T-1 through T-4 is a fleet of buildings.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.1.1 Warnings

6.1.1.1 System Issues

This version of PIMS (version 7b) is written in Microsoft Access™ version 2 (not 7 or 8). It interacts with Microsoft Excel™ versions 5 or 7 (not 8). Both Access and Excel must be present for PIMS to run many of the sample evaluations (analyses).

PIMS has been tested with Microsoft Windows 95™ and as little as 24mb of RAM. It has not been tested on either Microsoft Windows 3.x™ or Microsoft Windows NT™. PIMS will not run in IBM OS/2 Warp's™ Win 3.x port. PIMS has not been tested on a network.

Occasionally, a PIMS evaluation involving Microsoft Excel will come to a halt, and an error message will appear to the effect that a "DDE channel" has closed. Should this happen, use <ALT> <TAB> to switch to Excel, and open Options on the Tools menu. In the General category, uncheck the "Ignore Other Applications" option. Hit "OK", then return to Access, clear the error message, highlight the code module containing the offending code (it will have opened automatically), and hit the <F5> key. The evaluation should continue unfazed. This problem occurs sporadically, especially after ULSLEA [Mortazavi 1995, Stear & Bea 1997] has been run in Excel.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.1.1.2 Application Issues

PIMS is designed to be a flexible system for the management of data resulting from any potential engineering application. To this end, PIMS contains many sample engineering applications, known as evaluations or analyses. These are provided to guide the user in developing appropriate evaluations for user-specific purposes, and should not be relied upon for actual analyses.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.2 CUSTOMIZING PIMS

6.2.1 Main Menu Operations

Upon opening PIMS, the user will be presented with the Main Menu:

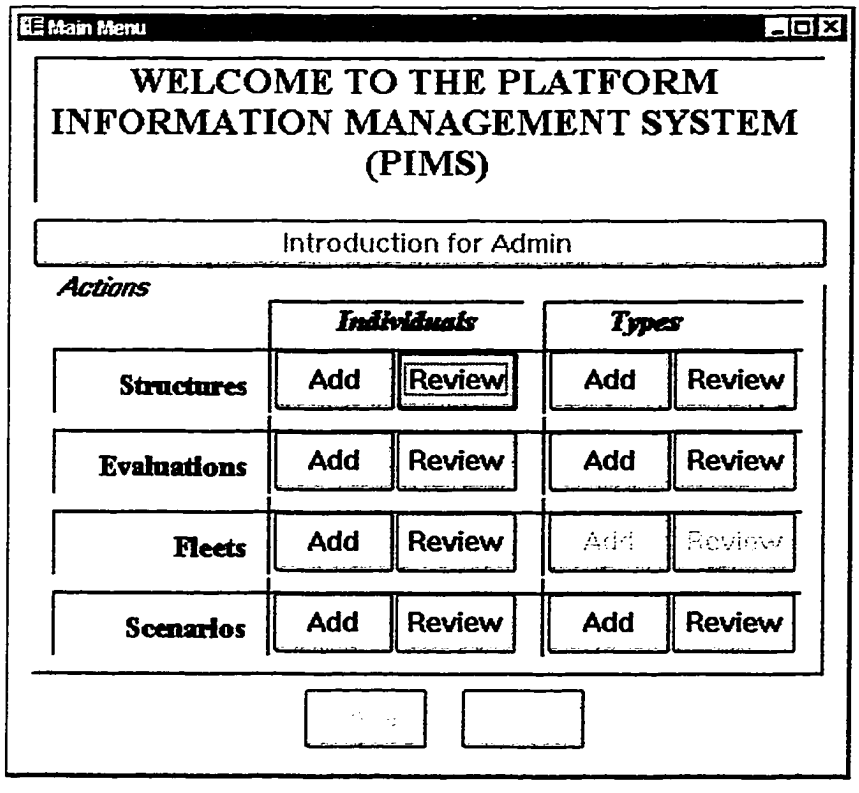


Figure 6-1: Main Menu

From the Main Menu form, the user decides which actions PIMS should perform. Existing types of structures, evaluations, and scenarios may be reviewed and new types may be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

added. Existing structures, evaluations, fleets, and scenarios may be examined; new structures, evaluations, fleets, and scenarios may also be added.

The complete title of the Main Menu's "Introduction for . . . " button indicates the log-in name of the current user of PIMS.

6.2.2 Adding a New Evaluation Type to PIMS

As an example of adding a new evaluation type to PIMS, add a function that performs higher mathematics: $z = \sqrt{x^2 + y^2}$. Needed are:

1. An "inspection" evaluation type that collects and stores the values of x and y (other types of input evaluations, "engineering assumptions" and "structure data," are not required for this simple example); and

2. An "analysis" evaluation type that retrieves the values of x and y from inspection evaluations and computes and stores the value of z.

Note that any evaluation type may serve as input for any other evaluation type, but that analysis evaluations require input from inspection evaluations, engineering assumption evaluations, or other analysis evaluations.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A name is needed for the evaluations that are about to be added: "PythI1" for the inspection routine, and "PythA1" for the analysis routine.

1. First, add the inspection routine. From the Main Menu, click on Add - Evaluations - Types. This opens the "Create a new type of evaluation" form:

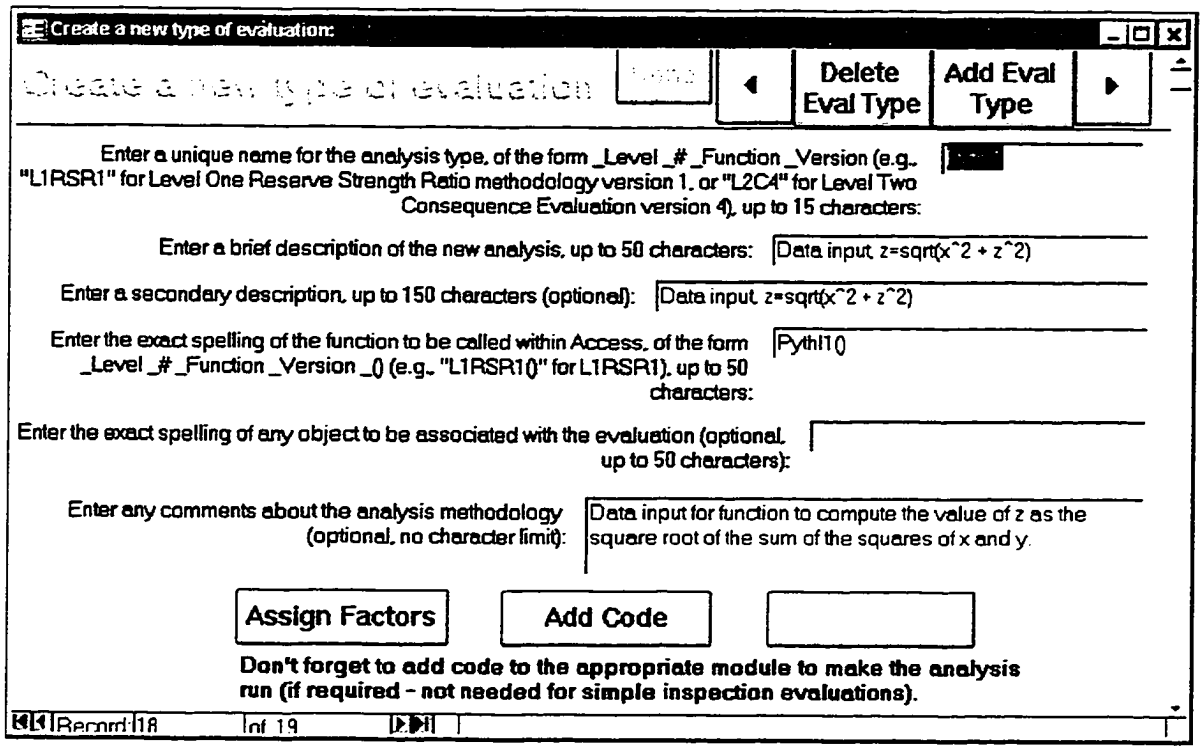


Figure 6-2: Adding an Evaluation Type

2. In the first box, enter "PythI1".[1]

3. In the second box, enter "Data input, z=sqrt(x^2 + y^2)".

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. Enter the same in the third box.

5. In the fourth box, enter "PythI1()".

6. Skip the fifth box.

7. Enter "Data input for function to compute the value of z as the square root of the sum of the squares of x and y." in the sixth box.

8. Now click the "Assign Factors" button.

9. This brings up the "Assign Factors to Evaluations" form:

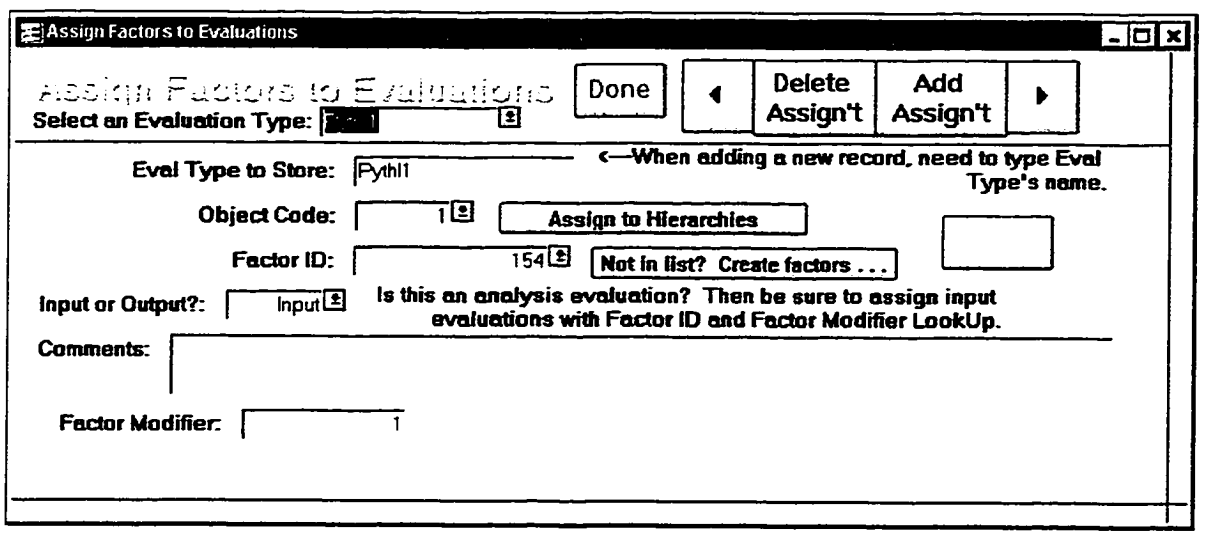


Figure 6-3: Adding Factors to the New Inspection Evaluation Type

---

[1] To avoid confusion, end-of-sentence punctuation marks are placed outside of quote marks representing data entry characters.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

10. Click the arrow next to the "Select an Evaluation Type" box and select PythI1.

11. Click in the "Eval Type to Store" box and type "PythI1".

12. Leave "Object Code" at the default value of "1" (the factors used for PythI1 have no sub- or super-objects).

13. Click on the arrow next to "Factor ID" and look for factor x. It won't be there, so click on the "Not in list? Create factors..." button. This opens the "Add Factors: Factor ID (basic factors)" form:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

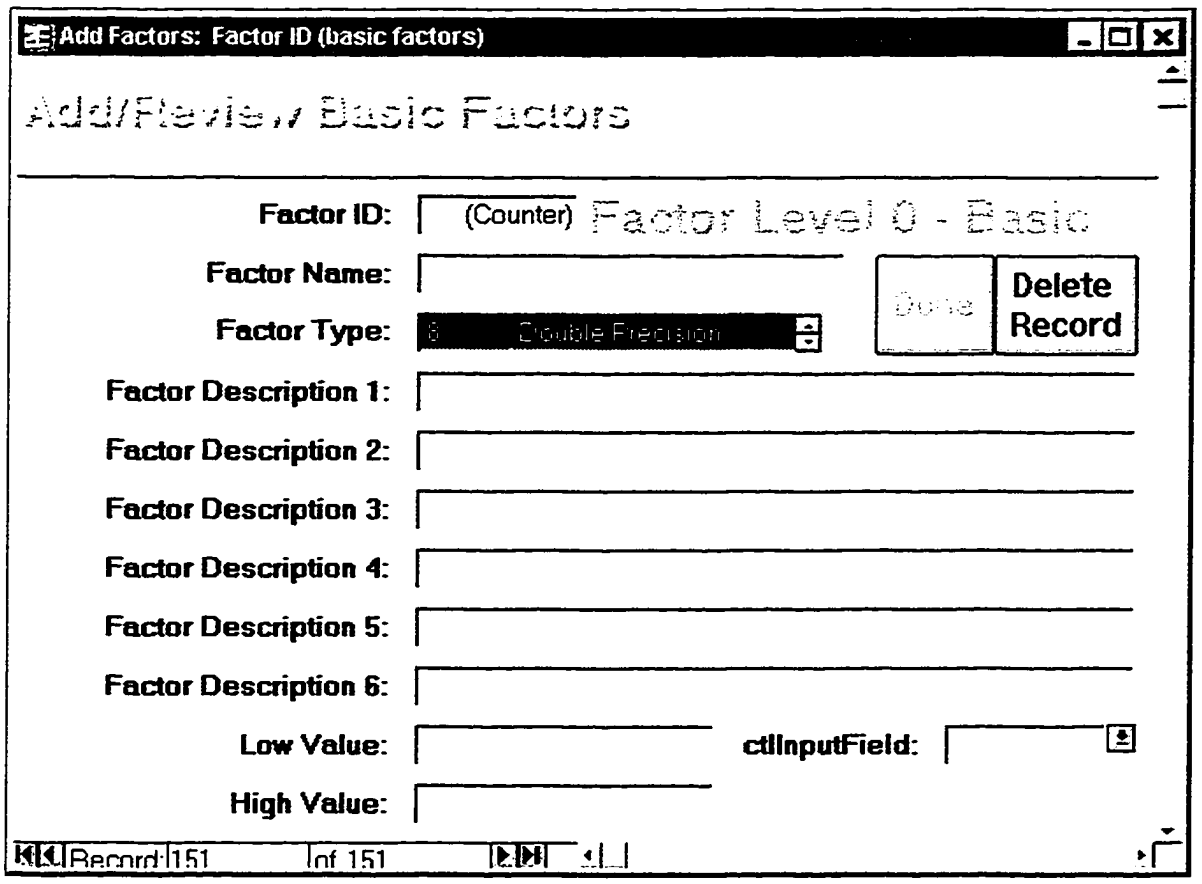


Figure 6-4: Entering Basic Factors

14. After placing the cursor in the "Factor Name" box and searching for "x" (with the "Match Whole Field" enabled, and finding nothing), enter "x" in the "Factor Name" box.

15. Use the arrows to select "Double Precision" in the "Factor Type" box. This defines the data type of the x variable.

16. Enter "x factor" for "Factor Description 1".

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

17. Enter "x factor" again for "Factor Description 4".

18. If you wish to restrict the low and high values of x, enter those limits in the "Low Value" and "High Value" fields.

19. Enter nothing in the "Input Field" box.

20. Click the "Done" button on the "Add Factors: Factor ID (basic factors)" form to return to the "Assign Factors to Evaluations" form. You should still be in the process of adding the first factor (x).

21. Now click on the drop-list arrow next to "Factor ID", move down within the list to factor x, and select it.

22. In field "Input or Output", select "Input".

23. Leave the Factor Modifier entry as 1.

24. Factor x will be registered as soon as the record is saved. Click on the large "forward" arrow in the upper right corner of the form to do so, and to prepare for the entry of factor y.

25. Run through steps 11 to 23 again for factor y.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

26. Factor z has no role in the inspection evaluation, and will not be entered at this time. Click on the "Done" button of the "Assign Factors to Evaluations" form.

27. For simple, single-level hierarchy inspection evaluations, no code is required. The inspection routine for "PythI1" is complete.

28. Now add the "analysis" evaluation routine. Back in the "Create a new type of evaluation" form (if it's already closed, click on the Main Menu's Review (or Add) - Evaluations - Types button), review the information you've just put in for PythI1. Now hit the "Add Eval Type" button.

29. In the first box, enter "PythA1". In the second, enter "Analysis, z=sqrt(x^2 + y^2)". In the third, enter the same text again. In the fourth, enter "PythA1()". Don't bother with the fifth box, and enter "Function to compute the value of z as the square root of the sum of the squares of x and y." in the sixth box.

30. Click the "Assign Factors" button to bring up the "Assign Factors to Evaluations" form again.

31. Click the arrow next to the "Select an Evaluation Type" box and select PythA1. Click in the "Eval Type to Store" box and type "PythA1".

32. Leave "Object Code" at the default value of "1" (the factors used for PythI1 have no sub- or super-objects). Click on the arrow next to "Factor ID" and look for factor z.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

It won't be there, so click on the "Not in list? Create factors..." button. This opens the "Add Factors: Factor ID (basic factors)" form.

33. In the "Add Factors: Factor ID (basic factors)" form, enter "z" in the "Factor Name" box.

34. Select "Double Precision" for "Factor Type." Enter "z factor" for "Factor Description 1" and "Factor Description 4".

35. It's best not to constrain the low and high limits of outputs with entries in the "Low Value" or "High Value" fields. Leave these blank.

36. Enter nothing in the "Input Field" box.

37. Click the "Done" button on the "Add Factors: Factor ID (basic factors)" form to return to the "Assign Factors to Evaluations" form. You should still be in the process of adding factor z. Select "z" in the Factor ID box.

38. In field "Input or Output", select "Output".

39. Factor z will be registered as you click the "forward" arrow to advance to the next factor: ID, Prior Evaluation. This factor is used to designate which input evaluation (inspection) is assigned to which output evaluation (analysis), and is required for all output evaluations. Enter "PythA1" in the "Eval Type to Store" field and move to the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

"Factor ID" field. Search the drop-down list for "ID, Prior Evaluation" (probably factor #29). Select that factor.

40. The Object Code field will automatically change to 2, and the Factor Modifier LookUp field will appear. Use the drop-down list to select "PythI1" as the input evaluation for PythA1.

41. A value will appear in the Factor Modifier field. Leave it alone; enter "Input" in the "Input or Output?" box.

42. The data will be registered as the form closes. Click on the "Done" button of the "Assign Factors to Evaluations" form.

43. Because this is an analysis evaluation, code is required. The "Create a new type of evaluation" form should again be in focus; click on the "Add Code" button. The new code will be created in Microsoft's Access Basic.™ Programming is performed in functions and subroutines, and these are stored in modules.

44. Four things should happen: 1) module evalPlatformsL1RSR1 should open to function L1RSR1, 2) a dialog box will pop up, which should be read and cleared, 3) a new module should be created, and within it a new function PythA1, and 4) you should approve PIMS's request that you proceed with an edit.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

45. Select the "evalPlatformsL1RSR1" window. Highlight the text from just below "Function L1RSR1()" to just above "End Function". Copy this text and paste it into the "Module1" window between the "Function PythA1" and "End Function" lines.

46. Starting with the second line of "Function PythA1()," there are thirteen lines of descriptive comments. Replace all of these with " ' This is a trial function to determine the square root of the sum " and "' of the squares of two numbers." (Two lines. Don't forget the leading apostrophes; these designate the lines as comments.)

47. If, when performing a "replace" operation, an error message pops up saying "One or more replacements too long and not made," clear the error message and perform the replace operation again, using the "verify" option rather than "replace all."

48. From the Edit menu of the toolbar, select Replace. Within the current procedure only, replace "L1RSR1" with "PythA1". Replace all "RSR1" with "z".

49. Since Module1 has been declared with "Option Explicit" (unlike module evalPlatformsL1RSR1), all variables must be explicitly declared. The fourth line of PythA1 is blank; insert the statement "Dim z as double". Hit <enter> and insert the statement "Dim t as variant".

50. Under " ' Save the value", replace "y = PythA1Save(z)" with "t = PythA1Save(z)".

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

51. Scroll to the bottom of the function, below "End Function," and enter "Function PythA1Main()". Hit <enter>. Move to the bottom again, and enter "Function PythA1Save(z)". Again, hit <enter>. Hit <ctrl><up arrow> twice to return to function PythA1. Now hit <ctrl><down arrow> to move to function PythA1Main.

52. Move to the "evalPlatformsL1RSR1" window. Hit <ctrl><down arrow> until you arrive at function L1RSR1Main. Highlight the text from just below "Function L1RSR1Main()" to just above "End Function". Copy this text and paste it into the "Module1" window between the lines "Function PythA1Main" and "End Function".

53. Still in Module1, replace line "Dim R1 as Double" with "Dim x as Double". Replace "Dim R2 as Double" with "Dim y as double". Delete the "Dim" (variable declaration) statements for R3 through S4. Replace "Dim RSR as Double" with "Dim z as double". Insert "dim Lo, Hi, badLimit as variant".

54. Replace all "ErrorHandlerLimitsL1RSR1" with "ErrorHandlerLimitsPythA1Main".

55. Replace all "L1RSR1Main" with "PythA1Main".

56. Replace the line "' Get values for R1-S4 from SQL; if no value, process error; " with "' Get values for x and y from SQL; if no value, process error; ". Don't forget the leading apostrophe.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

57. Replace all "R1" with "x". Replace all "R2" with "y" (remember, all of these replacements should be within the current procedure only).

58. Delete the section starting with " ' R3" and ending with "End if". Also do this for R4, R5, S1, S2, S3, and S4.

59. Replace the line "RSR = x * y * R3 * R4 * R5 / (S1 * S2 * S3 * S4)" with "z = sqr(x^2 + y^2)". Replace the line "PythA1Main = RSR" with "PythA1Main = z". Replace all "Calculate RSR" with "Calculate z", and all "Calculating RSR" with "Calculating z".

60. Replace all "x parameter 1" with "x".

61. That's all for PythA1Main. Now use <ctrl><down arrow> to go to PythA1Save.

62. Go back to the "evalPlatformsL1RSR1" window and move down to function L1RSR1Save. Copy everything between (and not including) "Function L1RSRSave (RSR)" and "End Function". Move back to the "Module1" window and paste the text between "Function PythA1Save (z)" and "End Function".

63. Close the "evalPlatformsL1RSR1" window (if asked, don't save it the module).

64. Back in Module1, replace all "L1RSR1Save" with "PythA1Save". Replace all "RSR" with "z".

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

65. Immediately before the statement "Dim FactorID1 as integer", add the statement "Dim t as variant".

66. Replace the line "x = DLookup("[fldLong]", "tblEvaluationFactorData", "[Evaluation ID] = " & EvalIDSelected & " AND [Factor ID] = " & FactorID1)" with "t = DLookup("[fldLong]", "tblEvaluationFactorData", "[Evaluation ID] = " & EvalIDSelected & " AND [Factor ID] = " & FactorID1)".

67. Replace the statement "If IsNull(x) Then GoTo Error_PythA1Save_1" with "If IsNull(t) Then GoTo Error_PythA1Save_1".

68. Replace the line "x = DLookup("[fldLong]", "tblEvaluationFactorData", "[Evaluation ID] = " & EvalIDSelected & " AND [Factor ID] = " & FactorID1)" with "t = DLookup("[fldLong]", "tblEvaluationFactorData", "[Evaluation ID] = " & EvalIDSelected & " AND [Factor ID] = " & FactorID1)".

69. From the Run menu, select Compile Loaded Modules.

70. Save the module as "evalDeleteMe1" and close it.

71. On the "Create a new type of evaluation" form, click "Done."

72. Now, test the new evaluation type:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.2.3 Adding a New Evaluation to PIMS:

1. Test the evaluation. From the Main Menu, click on Add - Evaluations - Individuals. This opens the "STRUCTURE OPERATIONS: Add New Evaluations" form:

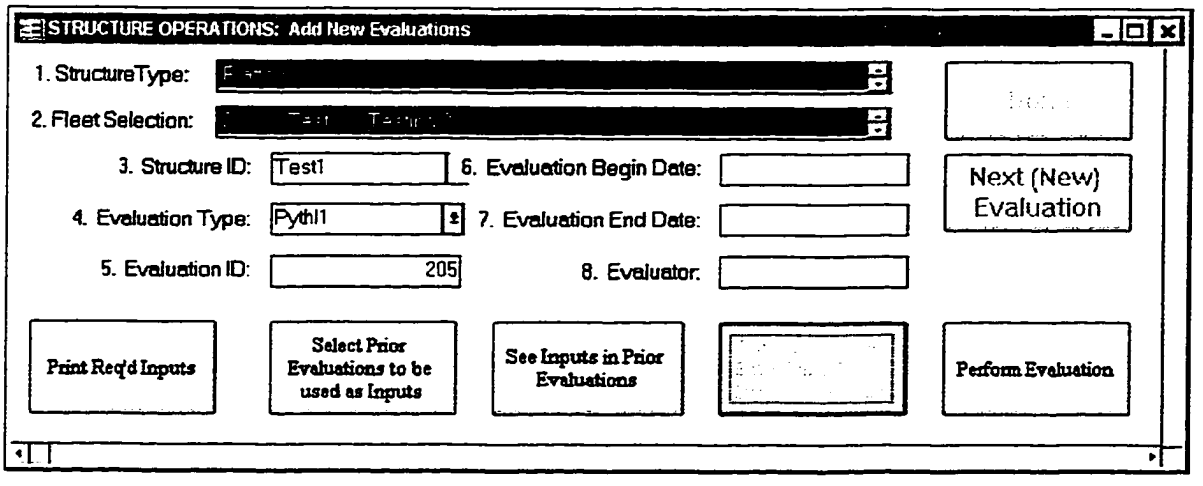


Figure 6-5: Creating a New Evaluation

2. Select "Platform" for Structure Type, and "2" for Fleet Selection (make sure that the selections have been highlighted before moving on).

3. In Structure ID, select platform Test 1. In Evaluation Type, select "PythI1" (inspections must be performed prior to analyses). Move the cursor to Evaluation ID, where the pre-assigned unique ID for this evaluation is visible.

4. Enter today's date in Evaluation Begin Date and Evaluation End Date. Enter your name in Evaluator.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. Click on the now-active "Select Prior Evaluations to be used as Inputs" button. Since this inspection requires no prior evaluations, a warning message to that effect will appear. Click OK to clear it.

6. Clicking on "See Inputs in Prior Evaluations" will produce a query with no entries. Close it with one of the top corner control boxes.

7. Click on the "Enter Other Data / See Results" button. This activates the "Data Entry / Results" form:

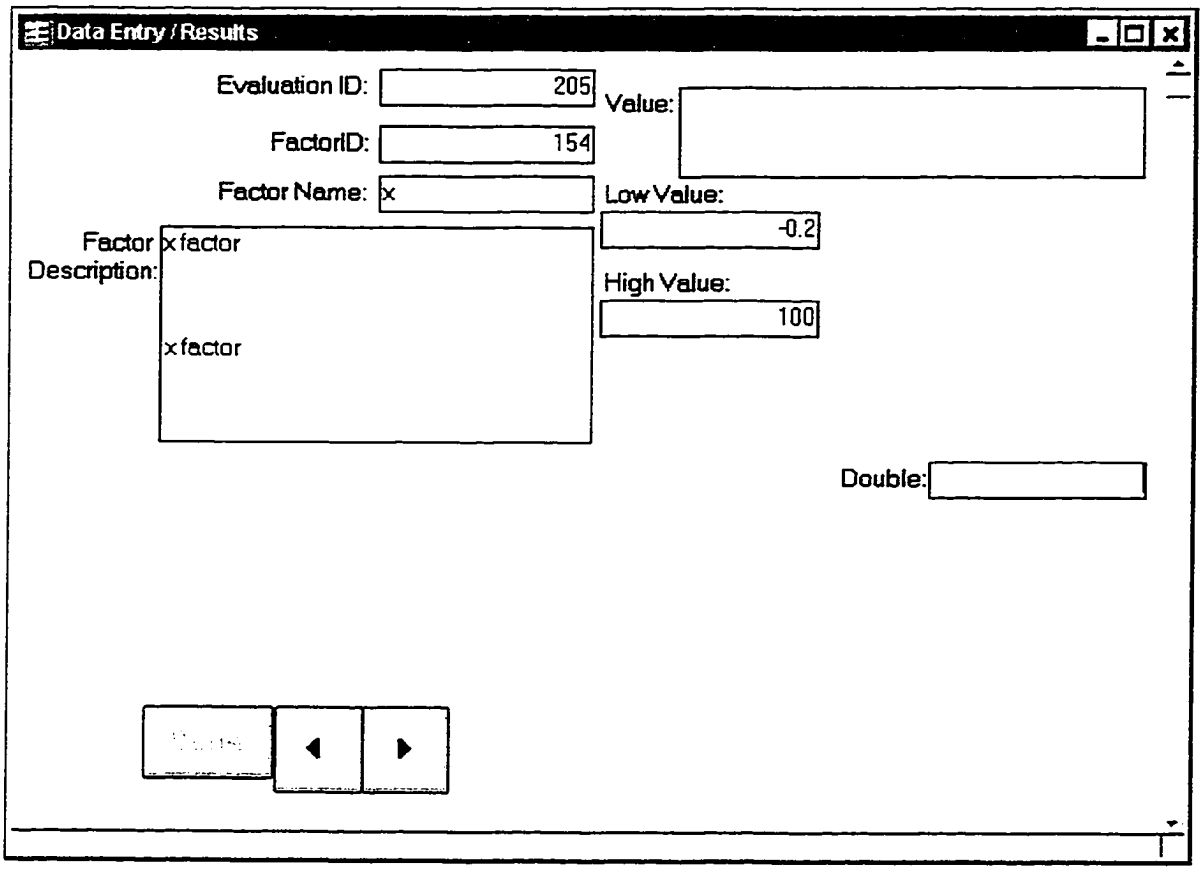


Figure 6-6: Data Input for Inspection

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

8. Data are entered directly into this form for each of the inputs. The first record opened is for factor x; since x was declared as type Double, enter the value 30 in the "Double" box. Note that data cannot be entered in the "Value" field.

9. Click on the forward arrow to enter 40 in the "Double" box for y. Note that you cannot move forward from here - there are only two factors available for each PythI1 evaluation. Click "Done".

10. You will be returned to the "STRUCTURE OPERATIONS: Add New Evaluations" form. Do not click the "Perform Evaluation" button - it does not apply here and will produce an error.

11. The inspection is complete. Click on "Next (New) Evaluation" to prepare for the PythA1 analysis.

12. Pick the same Structure Type, Fleet Selection, and Structure ID as before. For Evaluation Type, pick "PythA1". Enter today's date and your name in the appropriate fields.

13. Click "Select Prior Evaluations to be Used as Inputs". This brings up the "Prior Eval Entry" form:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

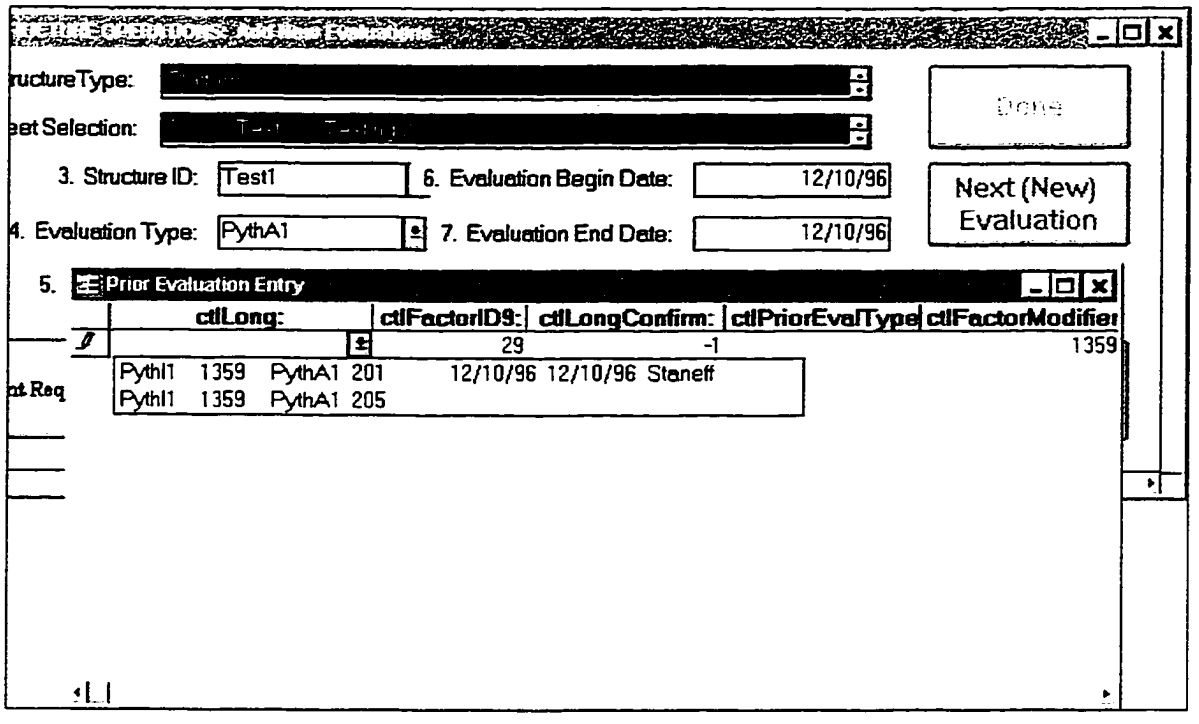


Figure 6-7: Selecting an Inspection for Input

14. Select the inspection just entered - in this case, #201.

15. Close the "Prior Eval Entry" form. Click on the "See Inputs in Prior Evaluations" button. This allows a review of inspection data:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

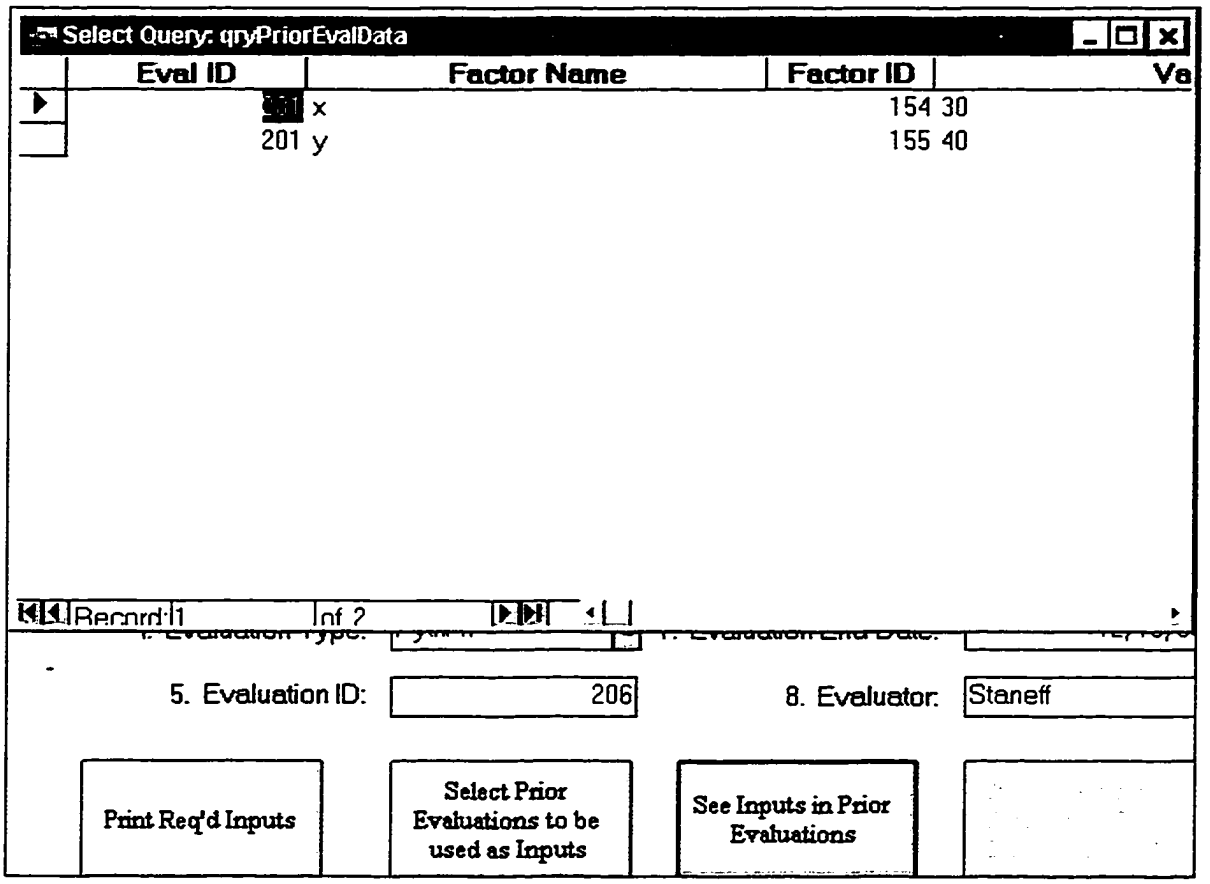


Figure 6-8: Reviewing Data in Input Inspection

16. Close the review query (using the upper-right control box) and click on the "Perform Evaluation" button. PIMS will quickly cycle through a number of information screens: "Processing...," "Retrieving data...," and "Saving z result..." and then will return to the "STRUCTURE OPERATIONS: Add New Evaluations" form.

17. Click on the "Enter Other Req'd Data / See Results" button to see the value of z:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

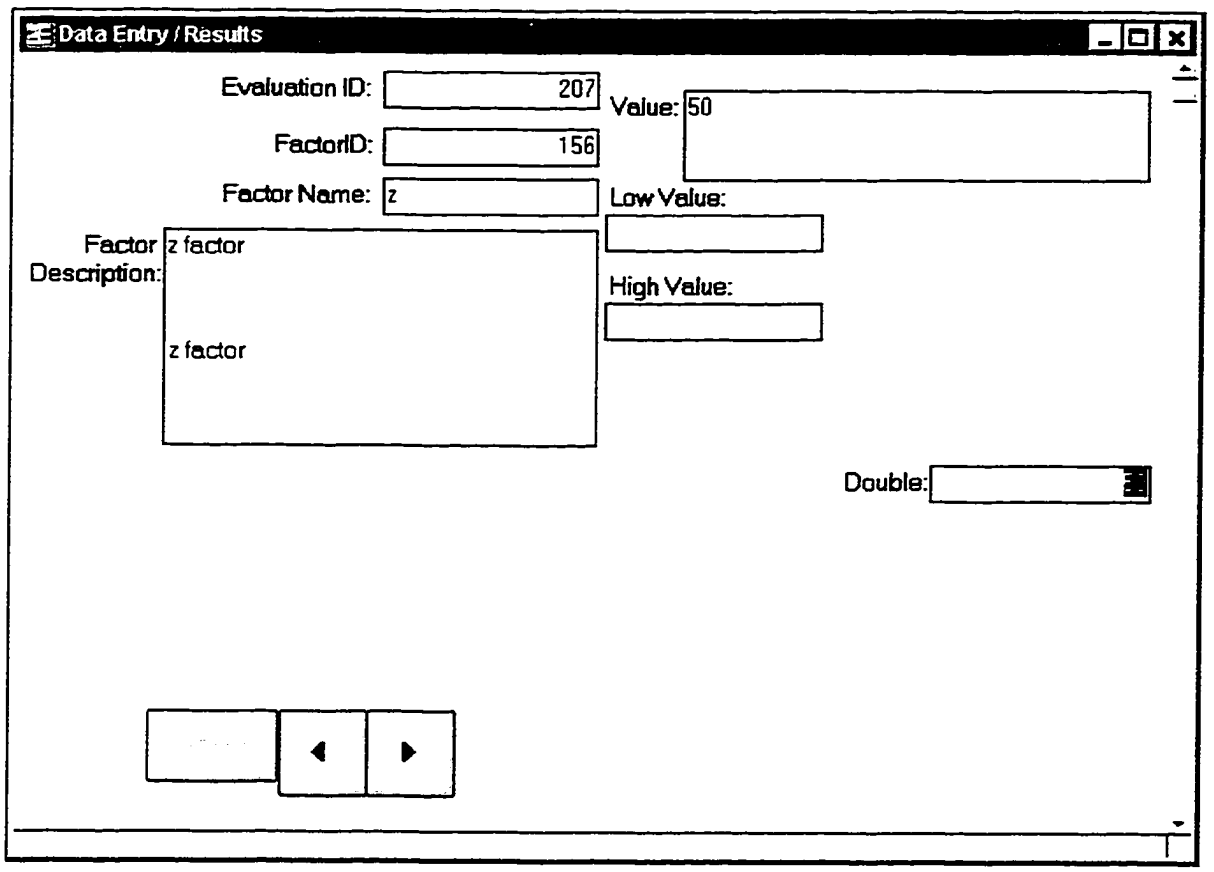


Figure 6-9: Viewing Evaluation Results

18. Close the "STRUCTURE OPERATIONS: Add New Evaluations" form with the "Done" button.

6.2.4 Adding a New Structure Type to PIMS

As an example of adding a new structure type to PIMS, add a wine cellar.

1. Proceed by selecting Add - Structures - Types from the Main Menu to bring up the "Structure Types" form:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

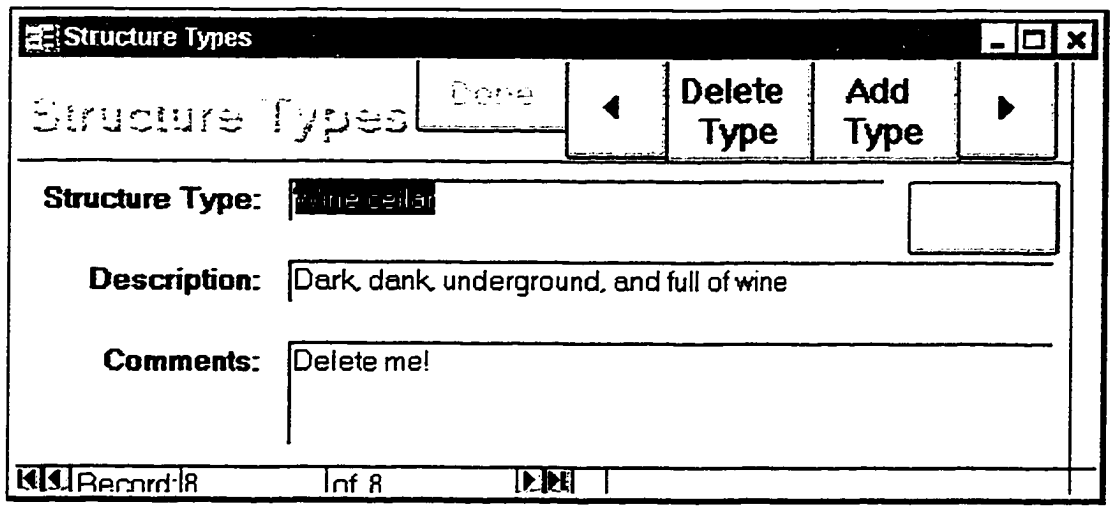


Figure 6-10: Adding a New Structure Type

2. A new record is automatically selected. In the "Structure Type" field, type "Wine Cellar".

3. For "Description," type "Dark, dank, underground, and full of wine." Add comments as appropriate.

4. Click the "Done" button to return to the Main Menu.

5. Now verify that the structure type has been registered by adding a new structure to PIMS.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.2.5 Adding a New Structure to PIMS

1. Select Add - Structures - Individuals from the Main Menu to open the "STRUCTURE OPERATIONS: Structure Edit/Input" Form:

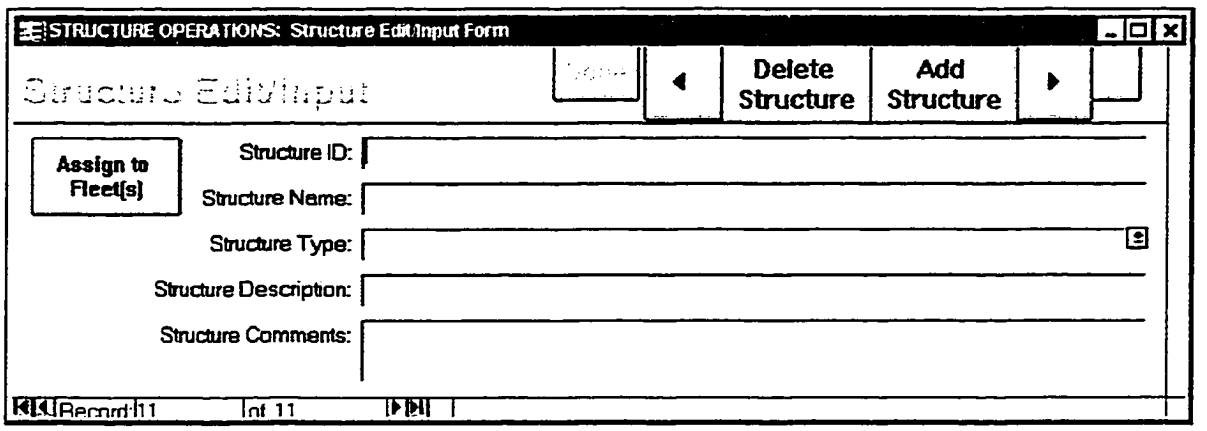


Figure 6-11: Adding a New Structure

2. Fill in "McLaughlin407" for Structure ID, "McLaughlin Cellars" for Structure Name, "The Dark Side of Davis Hall" for Structure Description, nothing for Structure Comments, and select "Wine Cellar" from the Structure Type drop-down list.

3. Click "Done" to complete the addition.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.2.6 Adding a New Fleet to PIMS

1. From the Main Menu, select Add - Fleets - Individuals to open the "Fleet Entry" form:

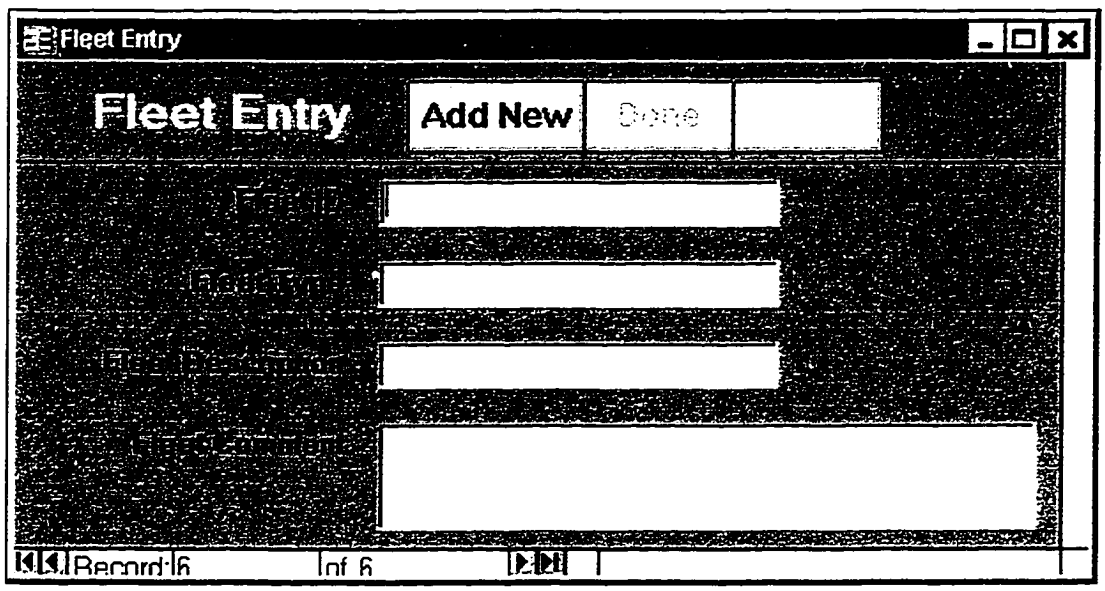


Figure 6-12: Adding a New Fleet

2. Type "F6" in Fleet ID, "Platform Fleet" in Fleet Type, and "Test Fleet" in Fleet Description.

3. Select "Done" to close the form.

4. Now select Review - Structures - Individuals from the Main Menu. This opens the "STRUCTURE OPERATIONS: Structure Edit/Input" form. Use the arrow keys to move forward to structure "Test1" (a sample platform):

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

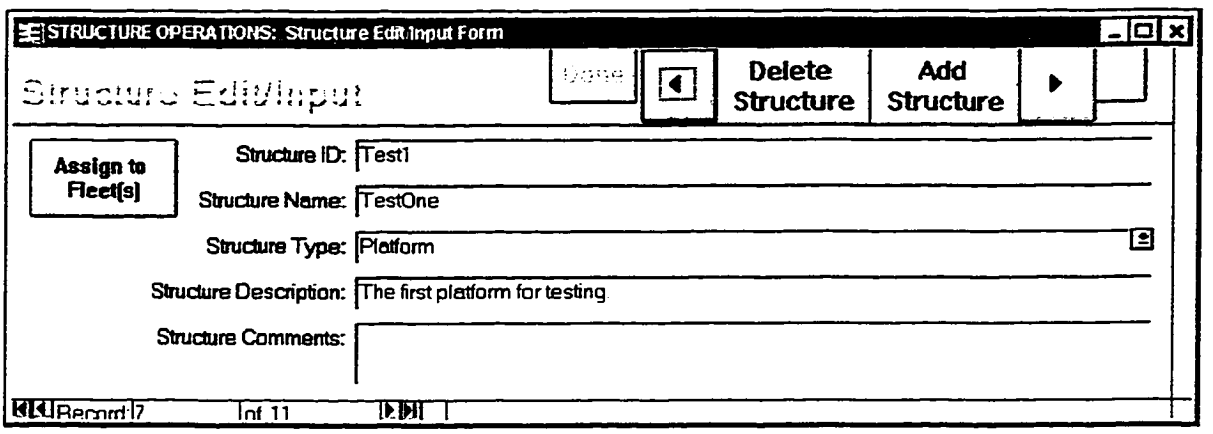


Figure 6-13: Picking a Structure to Assign to a Fleet

5. Click on "Assign to Fleet(s)" to bring up the "Assign Structures to Fleets" form:

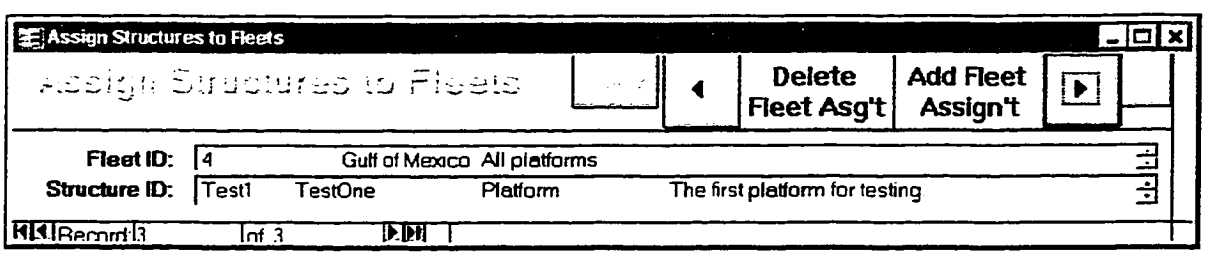


Figure 6-14: Assigning a Structure to a Fleet (1)

6. Click on "Add Fleet Assign't" to move to a new record.

7. In the Fleet ID box, use the right-hand arrows to select fleet F6.

8. In the Structure ID box, use the right-hand arrows to select structure Test1:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

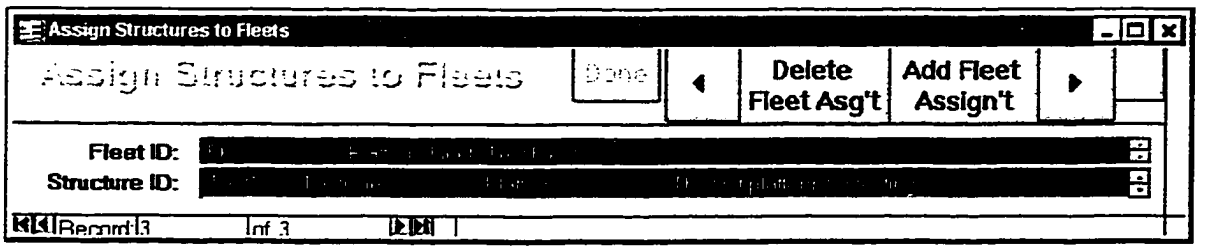


Figure 6-15: Assigning a Structure to a Fleet (2)

9. Click on "Add Fleet Assign't" again to add another platform to fleet F6: this time, select fleet F6 and platform Test3.

10. Click on Done to close the "Assign Structures to Fleets" form. In the "STRUCTURE OPERATIONS" form, click on Done again to close it.

6.2.7 Adding a New Scenario Type to PIMS:

1. From the Main Menu, select Add - Scenarios - Types to open the "Create a new type of scenario" form:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

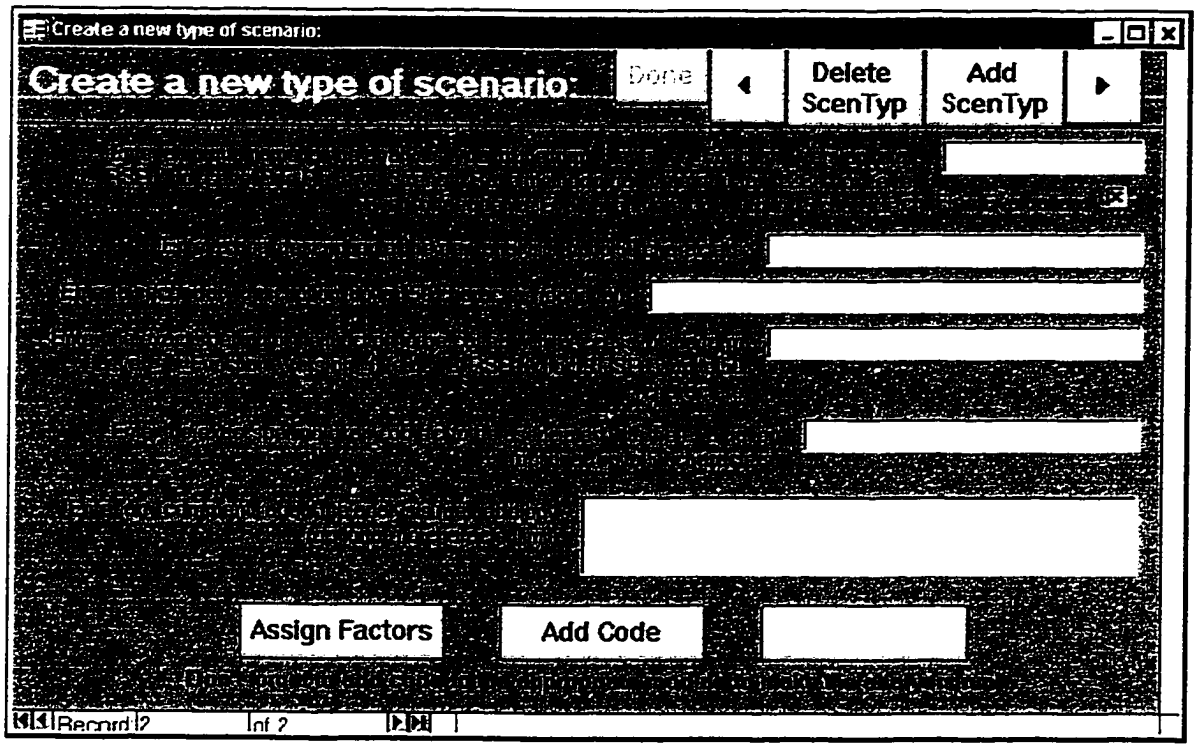


Figure 6-16: Creating a New Scenario Type

2. Adding a new scenario type is very similar to adding a new evaluation type. Enter the appropriate information in the appropriate boxes, assign factors as needed, and add code. Clicking the "Add Code" button opens a sample code module (for scenario type "L1Risk1_Fleet1").

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.2.8 Adding a New Scenario to PIMS

1. From the Main Menu, select Add - Scenarios - Individuals to open the "FLEET OPERATIONS: Add New Scenarios" form:

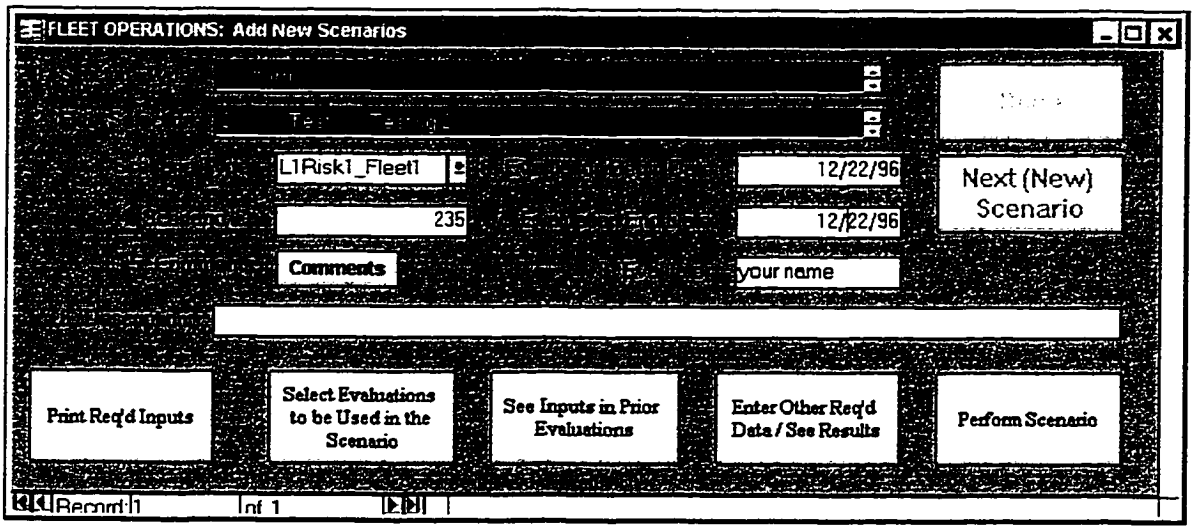


Figure 6-17: Adding a New Scenario

2. In the Structure Type box, select "Platform". In the Fleet Selection box, select fleet 2.

3. In the Scenario Type box, select "L1Risk1_Fleet1". A Scenario ID will be automatically assigned.

4. Click on the Comments button to add any comments.

5. An "Evaluation Begin Date" is automatically assigned. Enter today's date in "Evaluation End Date" as well.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6. Enter your name as Evaluator.

7. The L1Risk1_Fleet1 type of scenario requires that both an L1RSR1 and an L1C1 evaluation have been performed on every platform in the chosen fleet. To assign these evaluations, click the "Select Evaluations to be Used in the Scenario".

8. This opens form "FLEET OPERATIONS: Register Evaluations with Scenario". Each platform in the fleet is presented twice in the form - once for assignment of an L1RSR1 evaluation, and once for assignment of an L1C1 evaluation. Move down the form, selecting input evaluations with the drop boxes as follows:

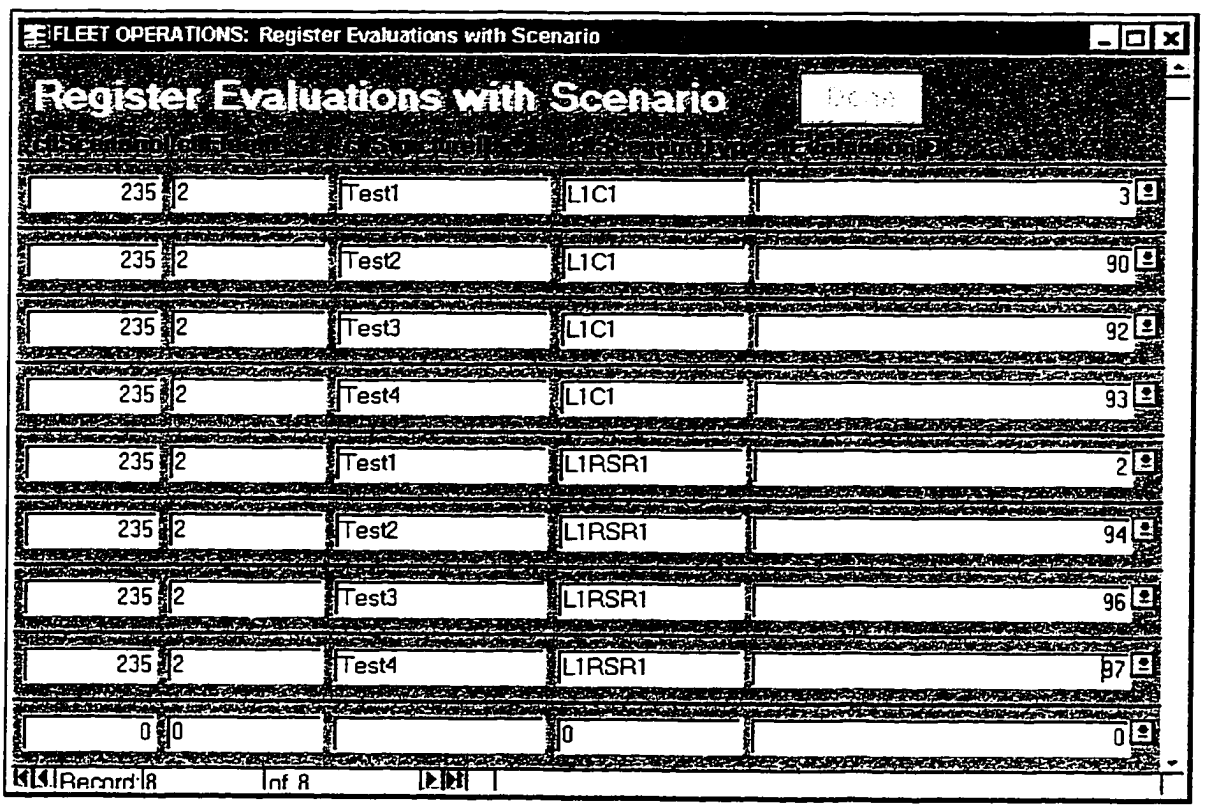


Figure 6-18: Picking Evaluations to Use in a Scenario

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

9. Click on the Done button to close the form.

10. Back in the "FLEET OPERATIONS: Add New Scenarios" form, click on "See Inputs in Prior Evaluations" to see the "Prior Scenario Data" form:

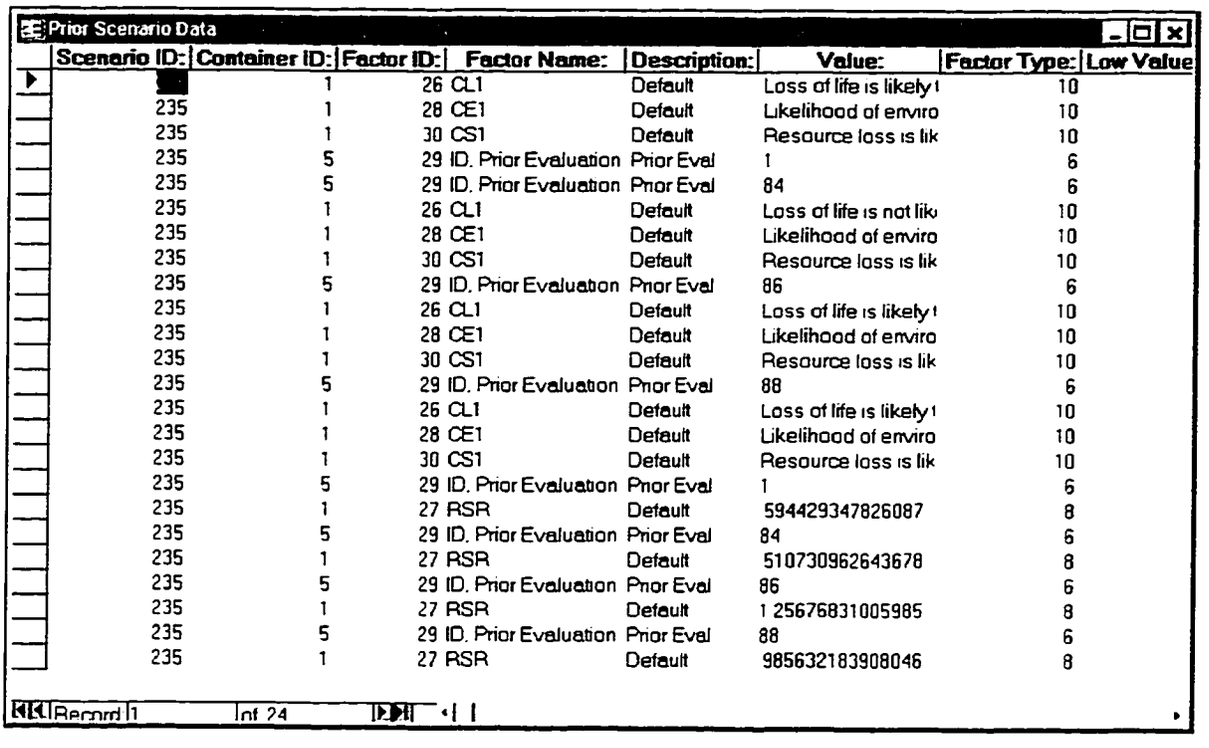


Figure 6-19: Reviewing Data in a Scenario's Component Evaluations

11. Close the Prior Scenario Data form using the upper-right control box.

12. Back in the "FLEET OPERATIONS: Add New Scenarios" form, click on "Perform Scenario" to perform the actual scenario.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

13. When finished, click on "Enter Other Req'd Data / See Results" to see the results of the scenario in form "Data Entry / Results":

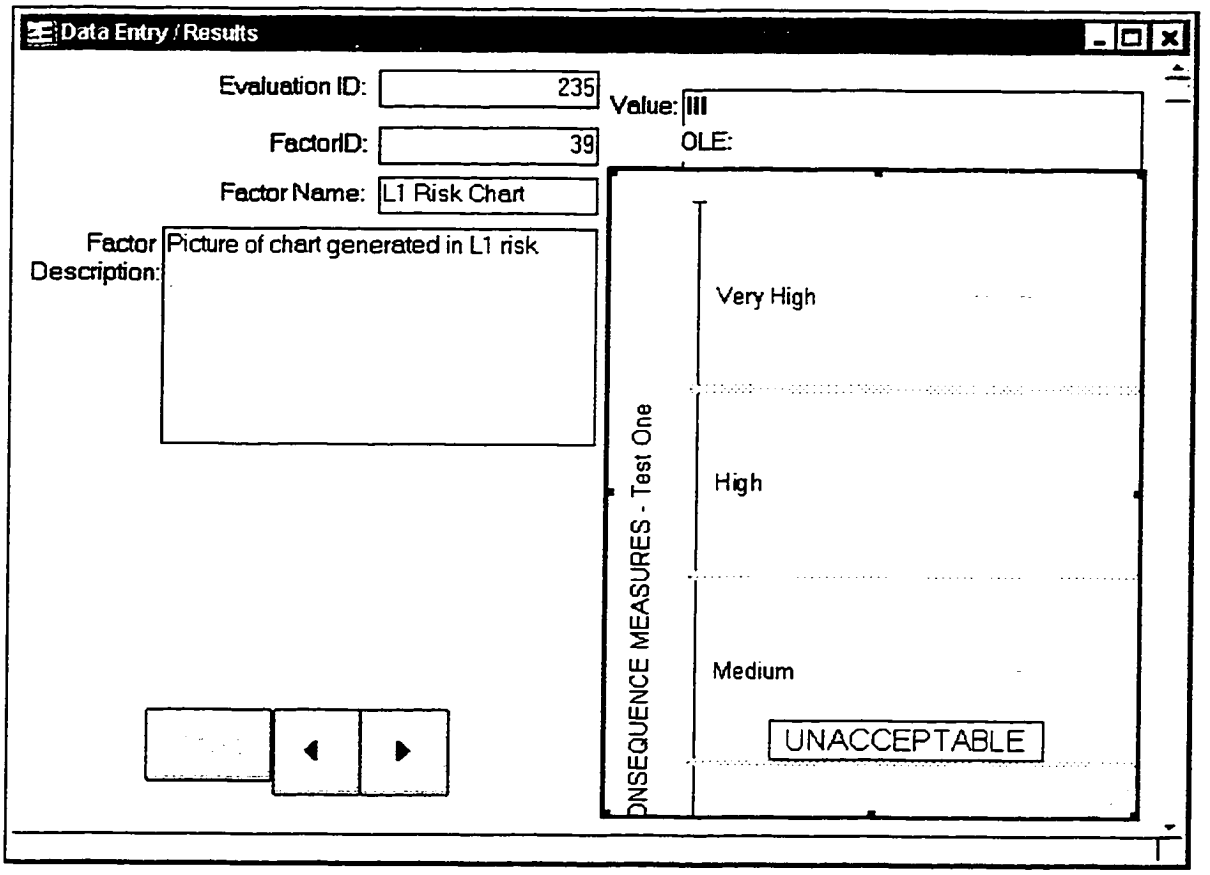


Figure 6-20: Viewing Scenario Results in PIMS

14. To see the complete picture of the results chart, highlight the displayed segment and then select "Copy" from the "Edit" menu. From Windows, open Paintbrush or another graphics program and paste the picture. This should yield something like:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

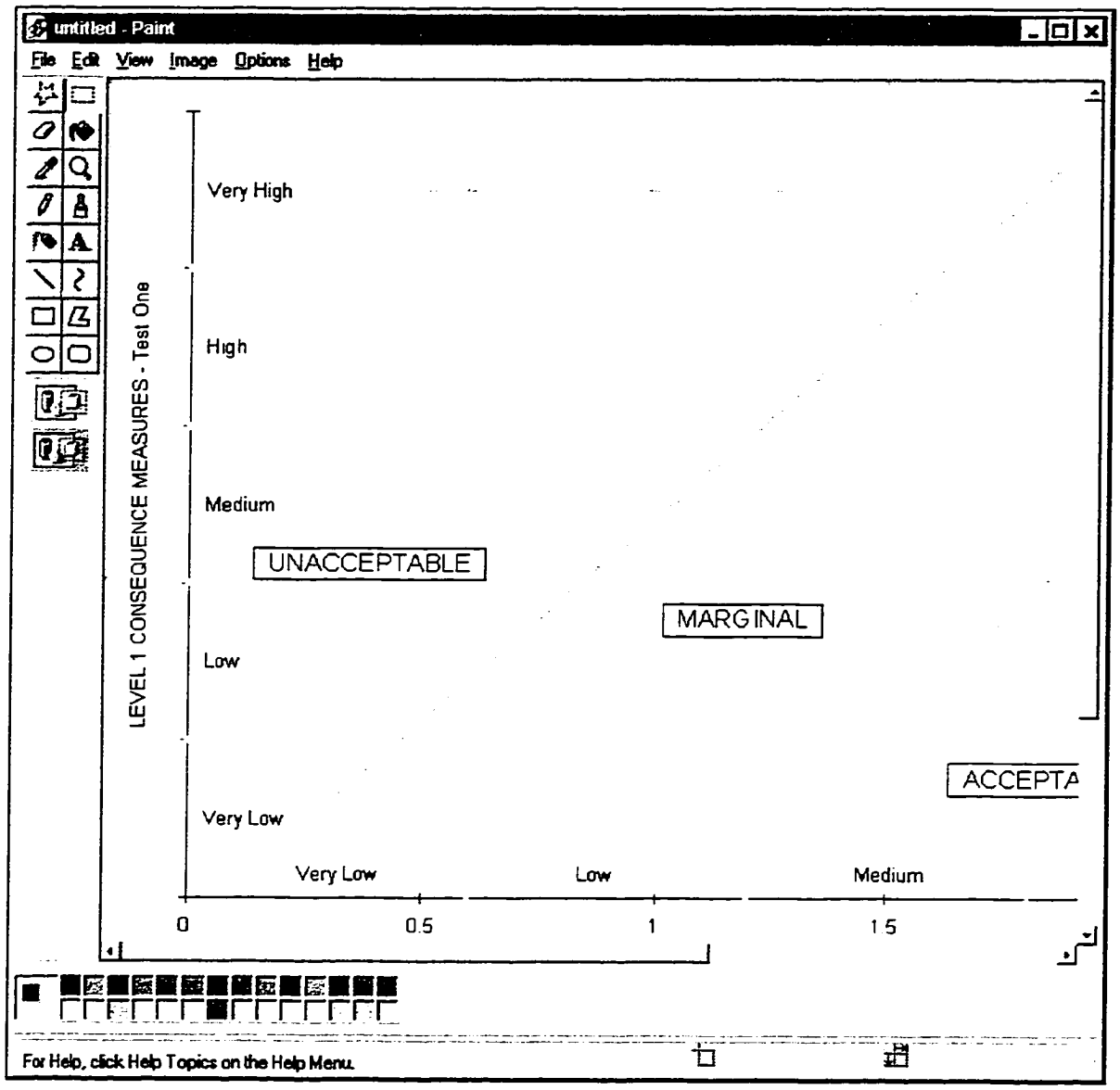


Figure 6-21: Viewing Scenario Results in External Editor

15. When finished, close the graphics editor, close the "Data Entry / Results" form, and close the "FLEET OPERATIONS: Add New Scenarios" form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.2.9 Adding a Multiple Hierarchy Evaluation Type to PIMS

As an example of a multiple-level hierarchy, consider a cost estimating system for the equipment in a simple central control and monitoring system (CCMS) for large buildings. This particular CCMS is priced by input and output points per panel, by numbers of panels, and by panel type. There are two panel types: remote terminal units (RTU) and smart remote terminal units (SRTU). There can be more than one panel and more than one type of panel per floor, but no particular panel can be used for more than one floor. The maximum number of inputs per panel is 64 and the maximum number of outputs is 8. Each RTU costs $1000 and each SRTU costs $5000; each input costs $50 and each output costs $150.

The cost estimating code might use a series of nested loops:

```
'i=number of particular floor (e.g., I=4 means "4th floor")
'j=type of panel (j=1 means "RTU;" j=2 means "SRTU")
'k=input or output (k=1 means input, k=2 means output)

For i = 0 To NFloors                'Floors, from the basement to the maximum
  For j = 1 To 2                    'Panel type
    N1 = Points(i, j, 1)            'Get number of inputs per panel type per floor
    N2 = Points(i, j, 2)            'Get number of outputs per panel type per floor
    If Points(i, j, 1) > 0 Then                    'Find number of panels req'd
      If Int(Points(i, j, 1) / 64) >= .5 Then
        NPanels1(j) = NPanels(j) + Int(Points(i, j, 1) / 64)
      Else
        NPanels1(j) = NPanels(j) + Int(Points(i, j, 1) / 64) + 1
      End If
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        Else
        End If
      If Points(i, j, 2) > 0 Then
        If Int(Points(i, j, 2) / 64) >= .5 Then
          NPanels2(j) = NPanels(j) + Int(Points(i, j, 2) / 8)
        Else
          NPanels2(j) = NPanels(j) + Int(Points(i, j, 2) / 8) + 1
        End If
      Else
      End If
    NPanels(j) = IMSMathMax(NPanels1(j), NPanels2(j))
    PointCost = PointCost + Points(i, j, 1) * 50 + Points(i, j, 2) * 150
  Next j
Next i
PanelCost = NPanels(1) * 1000 + NPanels(2) * 5000
SystemCost=PanelCost + PointCost
End Function
```

The task faced by the designer of a hierarchical evaluation is that of storing and retrieving arrays. In this case, the (i, j, k) array (representing floor number, panel type, and point type, and storing the number of points) must be stored in an inspection routine and retrieved in a cost estimation routine. The inspection routine is named "EstCostCCMSI1" and the cost estimation routine is "EstCostCCMS1."

The inspection routine and its accompanying code shows how arrays are stored in PIMS. Adding factors to the new inspection routine is complicated by the necessity of defining their hierarchy. However, a generic form ("frmL2RSR1Start") is used to simplify the cy-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

clical gathering of inputs, and an array handling function ("InputTier1A") manages the data hierarchy.

The following example is for advanced users only. Before attempting to implement the it, make a backup copy of PIMS.

1. Add the inspection routine. From the Main Menu, click on Add – Evaluations – Types. This opens the "Create a new type of evaluation" form (as in Figure 6-2).

2. In the first box, enter "EstCostCCMSI1".

3. In the second and third boxes, enter "Data input, CCMS cost estimating example".

4. In the fourth box, enter "EstCostCCMSI1()".

5. Enter "Data input for CCMS cost estimating example" in the sixth box.

6. Click the "Assign Factors" button to bring up the "Assign Factors to Evaluations" form (as in Figure 6-3).

7. Select "EstCostCCMSI1" in the "Select an Evaluation Type" and "Eval Type to Store" boxes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

8. Click on the "Not in list? Create factors..." box, and add a factor: Factor Name = "Floor ID", Factor Type = Integer, Factor Description 1 = "Floor ID", and Factor Description 4 = "High value is maximum number of floors". Click "Done".

9. Back in the "Assign Factors to Evaluations" form, enter the new factor's ID in the Factor ID box, and then assign to it an "Object Code" of 3 (this tells DAESSIM that the factor is an iterator). This is an input factor, so leave the "Input or Output?" box at 1.

10. Click the forward arrow to enter another factor.

11. Select "EstCostCCMSI1" again, and click on the "Not in list? Create factors..." box, to add a factor: Factor Name = "Panel Type", Factor Type = Integer, Factor Description 1 = "Panel Type (1=RTU, 2=SRTU)", and Factor Description 4 = "Panel type". Enter 1 for Low Value and 2 for High Value. Click "Done".

12. Back in the "Assign Factors to Evaluations" form, enter the new factor's ID in the Factor ID box. Leave "Object Code" at 1 for now, as well as "Input or Output". Click the forward arrow to enter another factor.

13. Select "EstCostCCMSI1" again, and click on the "Not in list? Create factors..." box, to add a factor: Factor Name = "Point Type", Factor Type = Integer, Factor Descrip-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

tion 1 = "Input or Output (1=Input, 2=Output)", and Factor Description 4 = "Input or Output". Enter 1 for Low Value and 2 for High Value. Click "Done".

14. Back in the "Assign Factors to Evaluations" form, enter the new factor's ID in the Factor ID box. Leave "Object Code" at 1 for now, as well as "Input or Output".

15. There are now three factors entered for evaluation type EstCostCCMSI1: Floor ID, Panel Type, and Point Type. Use the backward arrow to move to the factor for Panel Type.

16. Click on the "Assign to Hierarchies" button next to the Object ID field. This opens the "Assign to Hierarchies Summary" form, where arrays are assigned (Figure 6-22). Use the arrows at the bottom of the form to find a description that matches that of the array, in reverse order: for this array, look for "Point Type / Panel Type / Floor ID" It won't be there yet, so click on "Add Object" to move to a new record.

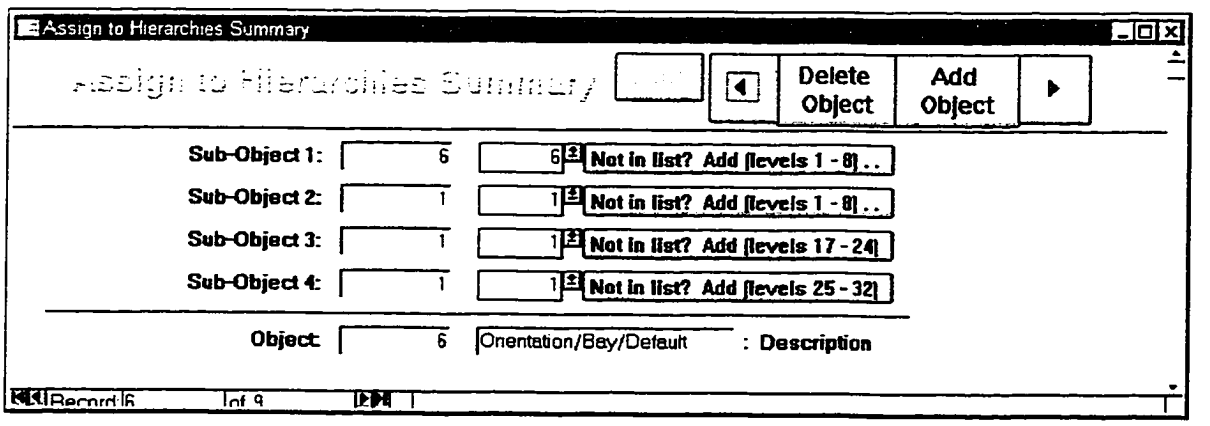


Figure 6-22: Summary Form for Assigning Hierarchies

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

17. Click on the "Not in list? Add (levels 1 – 8).." button. This will bring up the sub-object entry screen (Figure 6-23) (a few error messages may appear when this form opens; press <ESC> or <RETURN> to ignore them):

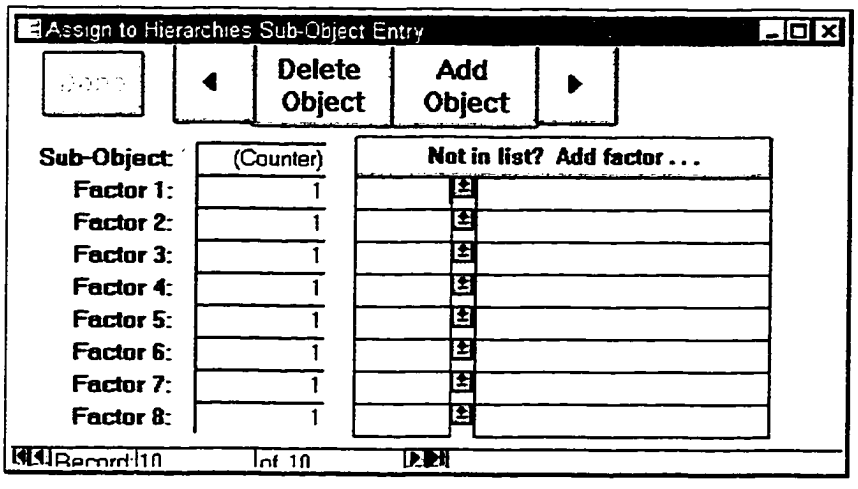


Figure 6-23: Form for Assigning Sub-Objects

18. Be sure that the "Sub-Object" box reads "(Counter)" as in Figure 6-23 - if not, click the forward arrow until it appears. In Factor 1, Factor 2, and Factor 3, enter the Factor ID's created above for the Floor ID, Panel Type, and Point Type factors. These ID's can be found either by clicking on the drop arrows in the "Assign to Hierarchies Sub-Object Entry" form, or by inspecting the "Assign Factors to Evaluations" form. The result should appear something like Figure 6-24 below. Click "Done" when finished to return to the "Assign to Hierarchies Summary" form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

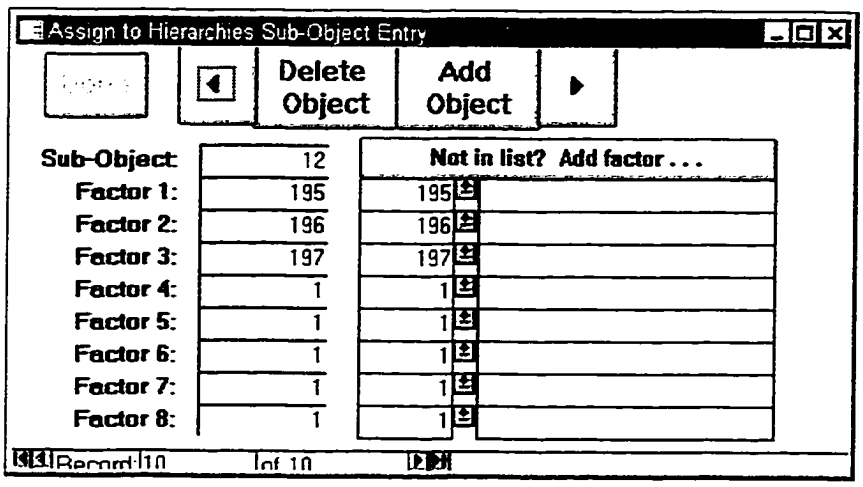


Figure 6-24: Sub-Object Assignment

19. Back in the "Assign to Hierarchies Summary" form, click "Add Object" so that "(Counter)" appears in the Object box. Using the drop arrow next to the Sub-Object 1 box, select the entry for the sub-object that was established in step 18. In Figure 6-25, this is ID #12.

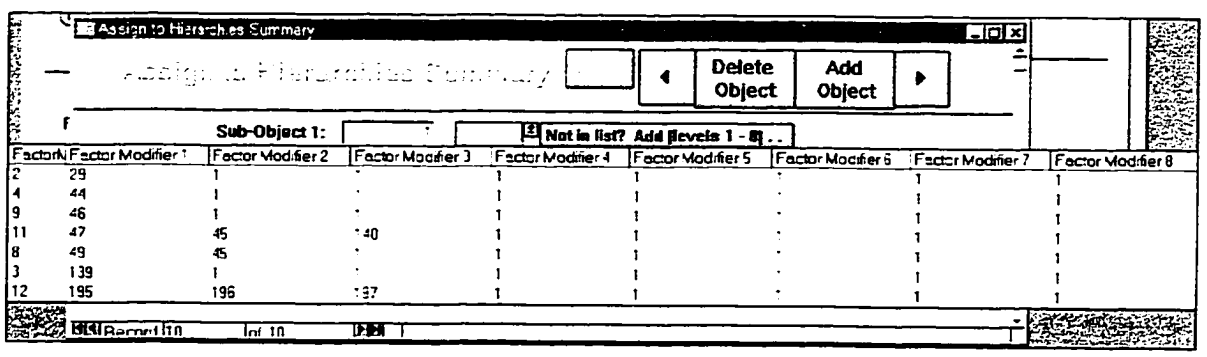


Figure 6-25: Selecting a Sub-Object

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

20. Leave the other sub-objects at their defaults of 1. In the Object Description box, enter "Floor ID / Panel Type / Point Type" (Figure 6-26). Click Done to return to the "Assign Factors to Evaluations" form.

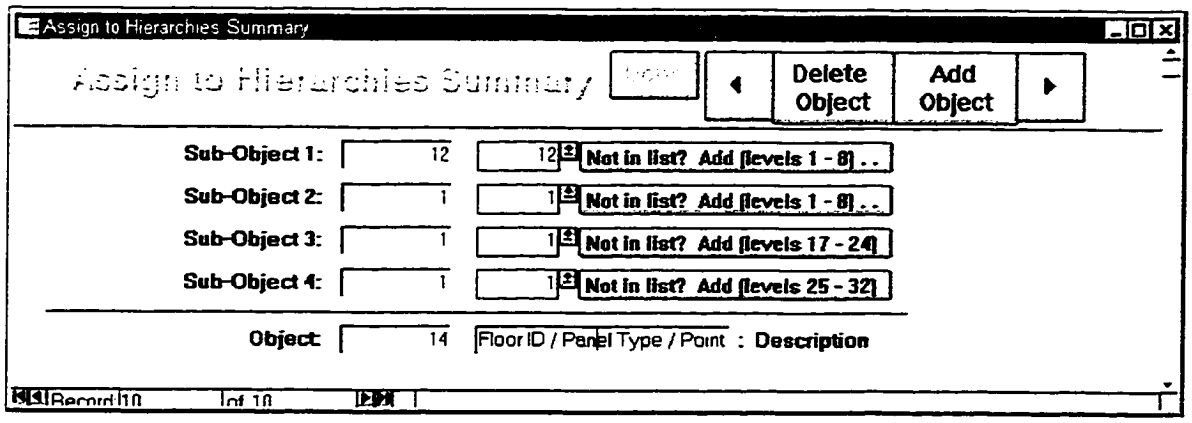


Figure 6-26: Completed Hierarchy Assignment

21. For the Panel Type and Point Type factors, enter the object code created in step 20: in this case, 14.

22. Click on the forward arrow to add another factor. Click on the Create Factors button to add a new factor: Factor Name = "Number of Points", Factor Type = Integer, Factor Description 1 and 4 = "Number of points". Click Done to return to the "Assign Factors to Evaluations" form, and select the new factor in the Factor ID box. Leave Object Code and Input / Output as 1. There should now be four factors defined for this evaluation: "Floor ID", with an object code of 3; "Panel Type", with the object code defined in step 20; "Point Type", with the object code defined in step 20; and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

"Number of Points", with an object code of 1. Click Done when finished to return to the "Create a New Type of Evaluation" form.

23. Click on the Add Code button. Two modules will open – "evalPlatformsL2RSR1" and "Module1." Close "evalPlatformsL2RSR1." Copy the "EstCostCCMSI1", "EstCostCCMSI1_2", and "EstCostCCMSI1_3" code from Appendix F to Module1. From the menu, choose <Run> <Compile Loaded Modules>, and close the module – renaming it "evalDeleteMeCCMSI1".

A separate evaluation type is needed to do the actual calculations. No special hierarchy management is necessary when defining factors for this evaluation (that's handled by the inspection routine). The code serves as an example of array retrieval in PIMS.

24. Create a new type of evaluation, with name "EstCostCCMS1", descriptions "Cost Estimation for CCMS", and function name "EstCostCCMS1()". Assign two factors: one for the ID of the prior inspection, and another for the System Cost result, as follows:

25. In the "Assign Factors to Evaluations" form, select EstCostCCMS1" (not "Est...I1") in the "Select an Evaluation Type" and "Eval Type to Store" fields. For Factor ID, select 29. Highlight Input in the "Input or Output?" field, and then select EstCostCCMSI1 in the "Factor Modifier Look Up" field. Leave the Object Code as 2, and hit the forward arrow.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

26. Again select EstCostCCMS1 in the "Eval Type to Store" field. Click on "Not in list? Create factors...", and create a Double Precision factor called "Cost". Enter Cost in the Factor Description 1 and Factor Description 4 fields, and click Done.

27. Back in the "Assign Factors to Evaluations" form, select the new cost factor for the Factor ID field. Select Output in "Input or Output?". Click Done to return to the "Create a New Type of Evaluation" form.

28. Click on Add Code, close the "evalPlatformsL2RSR1" module, and copy the "Est-CostCCMS1" code from Appendix F to Module1. From the menu, choose <Run> <Compile Loaded Modules>, and close the module – renaming it "evalDelete-MeCCMS1".

6.3 BUILT-IN EVALUATION TYPES

6.3.1 Using the Sample Evaluation Types

PIMS (version 7a) includes a number of sample evaluation types designed as prototypes for use with steel-jacketed oil platforms in the Gulf of Mexico. These evaluation types are:

PlatformData Basic data for platforms (example of automatic data entry)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

L1I1	L1 inspection for L1C1 and L1RSR1 (example of automatic data entry)
L1RSR1	L1 RSR, Bea & Craig (example of calculation within Access Basic)
L1C1	L1 consequence, Aggarwal (example of calculation within Access Basic)
L1Risk1	L1 risk, separate consequences (example of using external Excel spreadsheet via OLE (Microsoft's Object Linking and Embedding™ protocol))
L1Risk1_Fleet1	Risk analysis, Level 1, fleet (example of using external Excel spreadsheet via OLE)
L2I1	L2 inspection for L2RSR1 and L2RSR2 (example of multiple-hierarchy data management)
L2Eng1	Engineering inputs for L2RSR1 and L2RSR2 (example of automatic data entry)
L2RSR1	L2 RSR, Mortazavi & Bea (example of multiple-hierarchy data management and use of external Excel spreadsheet via DDE (Microsoft's Direct Data Exchange™ protocol))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

L2RSR2 L2 RSR, Mortazavi & Bea, revised methodology (example of multiple-hierarchy data management and use of external Excel spreadsheet via DDE)

L2C1 L2 consequence data entry for L2Risk1 (example of automatic data entry)

L2Eng2 Engineering inputs for L2Pf3 (example of automatic data entry)

L2Pf3 L2 probability of failure analysis, based on L2RSR2 (example of iterative capabilities)

L2Risk1 L2 risk, consolidated consequences (example of using external Excel spreadsheet via OLE)

The following sample evaluation types are included but are based on California Coastal platform methodologies [Staneff et al 1994]:

L1HMax1 Expected Max Wave Height, L1 (example of automatic data entry)

L1VMax1 L1 Max Wind Velocity based on Max Wave Height (example of using external Excel spreadsheet, via OLE, for data input)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

L2VMax1 L2 Max Wind Velocity based on Max Wave Height (example of analytical input for L2RSR2; example of using external Excel spreadsheet, via OLE, for data input)

In all cases (except fleet operations), to create a new evaluation, click on Add - Evaluations - Individuals from the Main Menu to open the "STRUCTURE OPERATIONS: Add New Evaluations" form. If creating more than one evaluation in a session (while keeping the "STRUCTURE OPERATIONS..." form open), be sure to click "Next (New) Evaluation" before beginning each new analysis.

6.3.2 Creating a "PlatformData" Evaluation

A PlatformData evaluation stores basic data about platforms. It is generally used to store data that do not often change (e.g., water depth). However, multiple PlatformData evaluations can be performed on a platform; conflicting information indicates a change in the platform's basic configuration (e.g., if the platform has been moved, the water depths will probably differ). Table 4-6 lists the input data.

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "PlatformData" as the evaluation type:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

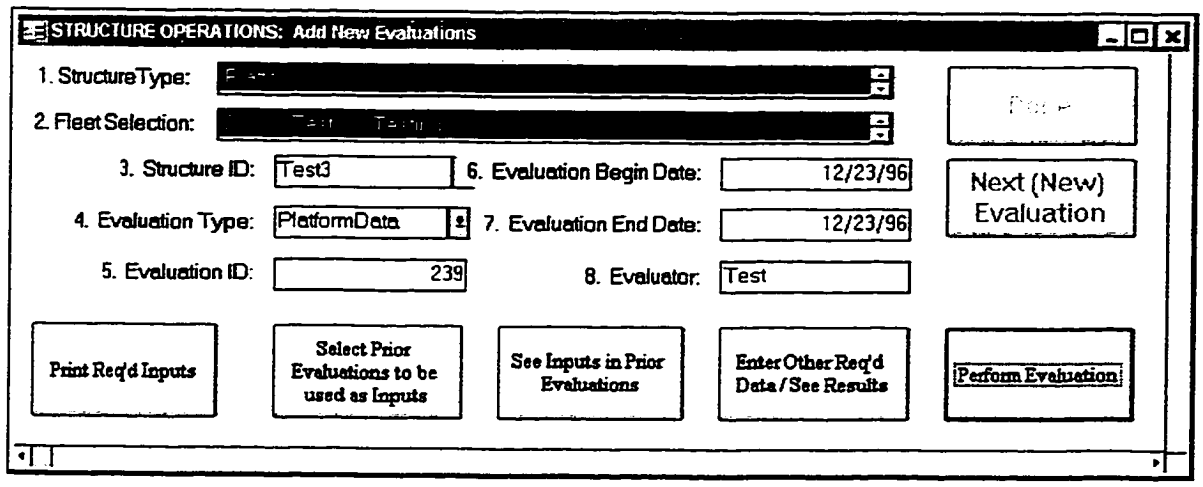


Figure 6-27: Adding a PlatformData Evaluation

2. Click the "Enter Other Req'd Data / See Results" button to open the "Data Entry / Results" form:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

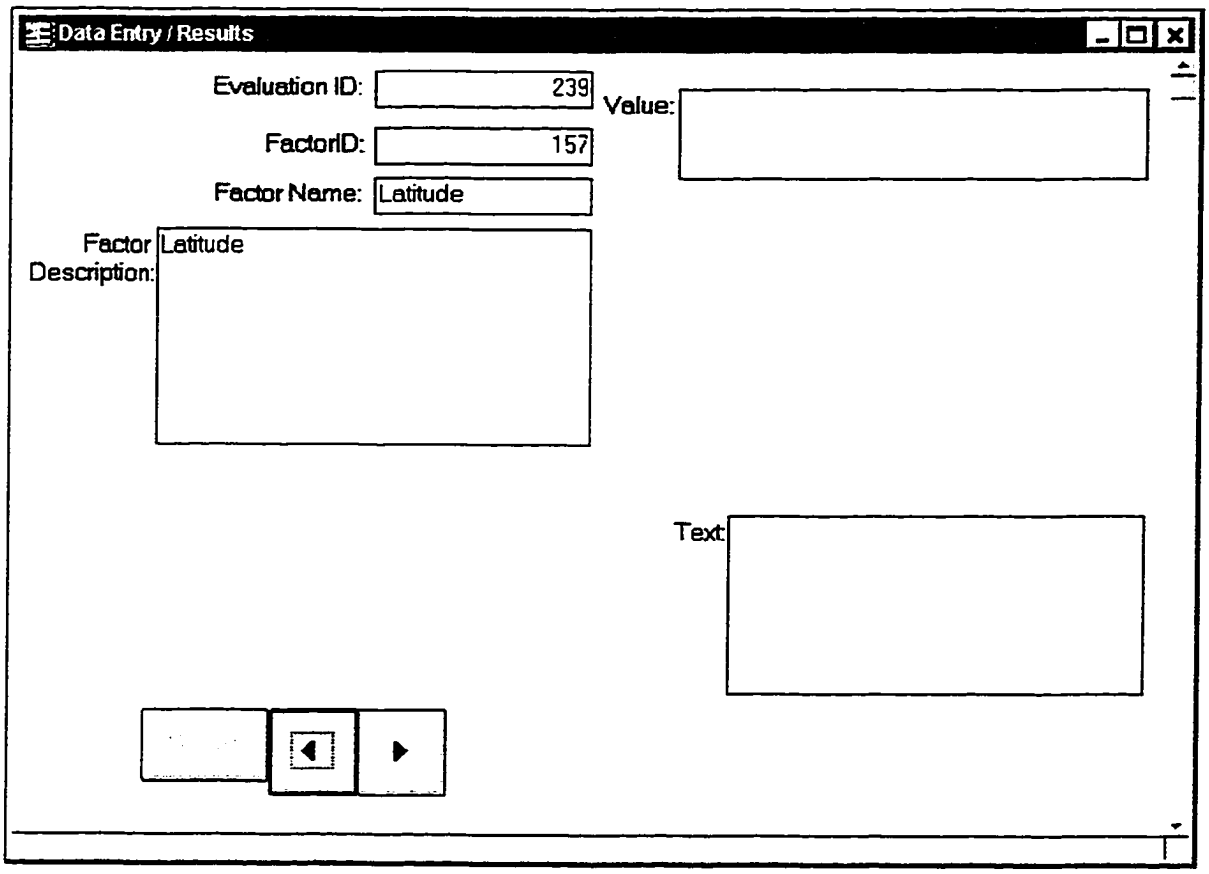


Figure 6-28: Entering Data for a PlatformData Evaluation

3. Use the arrow buttons to move among the available factors. Enter data not in the "Value" field, but in the data type field - in the illustration, the "Text" field.

4. Close the "Data Entry / Results" and "STRUCTURE OPERATIONS" forms when finished.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.3.3 Performing an "L1I1" Inspection

An L1I1 ("Level One Inspection, version 1") inspection evaluation is necessary to provide data to the L1C1, L1RSR1, L1Risk1, and L1Risk1_Fleet1 evaluations. Table 4-1 lists the collected data.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L1I1" as the evaluation type.

2. Click the "Enter Other Req'd Data / See Results" button to open the "Data Entry / Results" form. Use the arrow buttons to move among the available factors. Enter data not in the "Value" field, but in the data type field.

3. Close the "Data Entry / Results" and "STRUCTURE OPERATIONS" forms when finished.

6.3.4 Performing an "L1RSR1" Analysis

The logic of the L1RSR1 ("Level One Reserve Strength Ratio, version 1") evaluation type included with PIMS was originally developed by Bea and Craig [1993], and is based on the equation:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$RSR = \frac{R_1 \cdot R_2 \cdot R_3 \cdot R_4 \cdot R_5}{S_1 \cdot S_2 \cdot S_3 \cdot S_4}, \qquad (1)$$

where RSR is the Reserve Strength Ratio, and factors R_1 through S_4 are subjective factors obtained from a prior L1I1 evaluation. An L1RSR1 evaluation therefore requires an L1I1 evaluation be designated as an input. An L1RSR1 evaluation results in one output, RSR. (See Table 4-2.)

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L1RSR1" as the evaluation type.

2. Click on "Select Prior Evaluations to be Used as Inputs" to open the "Prior Evaluation Entry" form:

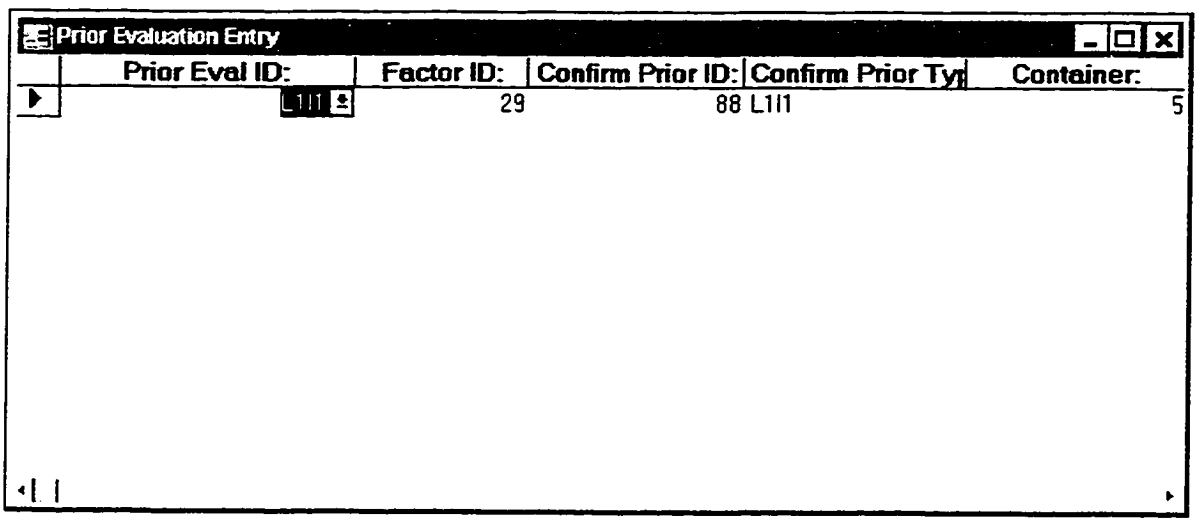


Figure 6-29: Selecting an L1I1 Inspection for Use with an L1RSR1 Evaluation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. Use the drop-down list to select one of the available L1I1 inspections. Use the upper-right control box to close the form.

4. Click on "See Inputs in Prior Evaluations" to see the input data. Close that form.

5. Click "Perform Evaluation" to perform the L1RSR1 analysis. Click on "Enter Other Req'd Data / See Results" to see the calculated value of RSR.

6.3.5 Performing an "L1C1" Evaluation

The L1C1 ("Level One Consequence, version 1") evaluation type included with PIMS is based on procedures developed by Aggarwal for platforms in the Gulf of Mexico [Aggarwal 1991]. It is a subjective approach, as illustrated in Figures 4-2 and 4-3.

An L1C1 evaluation requires that an L1I1 inspection be designated as an input. See Table 4-3 for more I/O information.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L1C1" as the evaluation type.

2. Click on "Select Prior Evaluations to be Used as Inputs" to open the "Prior Evaluation Entry" form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. Use the drop-down list to select one of the available L1I1 inspections. Use the upper-right control box to close the form.

4. Click on "See Inputs in Prior Evaluations" to see the input data. Close that form.

5. Click "Perform Evaluation" to perform the L1C1 analysis. Click on "Enter Other Req'd Data / See Results" to see the calculated consequence values.

6.3.6 Creating an "L1Risk1" Evaluation

The L1Risk1 ("Level One Risk, version 1") evaluation type included with PIMS is based on procedures developed by Aggarwal [1991], Bea and Craig [1993], and Staneff, Ibbs, and Bea [1994, 1995]. Incorporating the subjective L1RSR1 and L1C1 evaluation types makes it a subjective approach. Similarly, the L1RSR1, L1C1, and L1Risk1 evaluation types are all example methodologies and are not meant to be used to make decisions.

The L1Risk1 methodology plots consequences of failure against structural integrity to achieve an assessment of risk due to storm events. It does not combine consequences into one measure but plots the different categories - life, spillage, and economics - separately. To do this, the methodology converts the textual results of an L1C1 analysis as follows:

1. Loss of Life:
   - "Loss of life is not likely to occur" = 0.5
   - "Loss of life is likely to be very low" = 2.5

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- "Loss of life is likely to be moderate to very high" = 4.5
- Otherwise = 4.5

2. Spillage:
   - "Likelihood of environmental pollution is nil to very low" = 0.5
   - "Likelihood of environmental pollution is low" = 2.5
   - "Likelihood of environmental pollution is moderate to very high" = 4.5
   - Otherwise = 4.5
3. Economic Loss:
   - "Resource loss is likely to be low and property loss is likely to be very low to low" = 0.5
   - "Resource loss is likely to be moderate and property loss is likely to be very low to low" = 2.5
   - Otherwise = 4.5

These numeric consequence values are completely arbitrary, and are provided for illustration only. They are plotted separately along the vertical axis. The RSR result from an L1RSR1 analysis is plotted along the horizontal axis. Thus the three consequence results will form three points on the graph, which will be identified as "Acceptable," "Marginal," or "Not Acceptable" according to their location relative to the solid "Acceptable" and "Marginal" lines. (The L1Risk1 evaluation can serve as a template for a new L1Risk*x* evaluation class, using different equations for the "Acceptable" and "Marginal" lines – thus using a different safety standard. See Section 4.3.5.)

The user should not draw false conclusions from the Consequence Measure vs. RSR chart. This is possible because the chart apparently illustrates deterministic results. The user

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

must bear in mind that uncertainties exist in the assessment of RSR, in the assessment of consequence measures, and in the placing of the "Acceptable" and "Marginal" lines themselves. Thus the data points in the output charts should be interpreted as blurred circles or ovoids, possibly overlapping much thicker "Acceptable" or "Marginal" lines, and labeled as "probably acceptable" or "probably marginal."

An L1Risk1 evaluation requires that an L1C1 evaluation and an L1RSR1 evaluation be designated as inputs. See Table 4-4 for the object's attributes.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L1Risk1" as the evaluation type.

2. Click on "Select Prior Evaluations to be Used as Inputs" to open the "Prior Evaluation Entry" form.

3. Use the drop-down list on the first line to select one of the available L1C1 inspections. Use the drop-down list on the second line to select one of the available L1RSR1 evaluations. Use the upper-right control box to close the form.

4. Click on "See Inputs in Prior Evaluations" to see the input data. Close that form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. Click "Perform Evaluation" to perform the L1Risk1 analysis. Excel will open the file "L1Risk1.xls," import the appropriate data, and present the L1Risk1 Chart for inspection. To see the plotted points, it is occasionally necessary to manually move some of the data labels. Click the "Done" button on the Excel chart when finished.

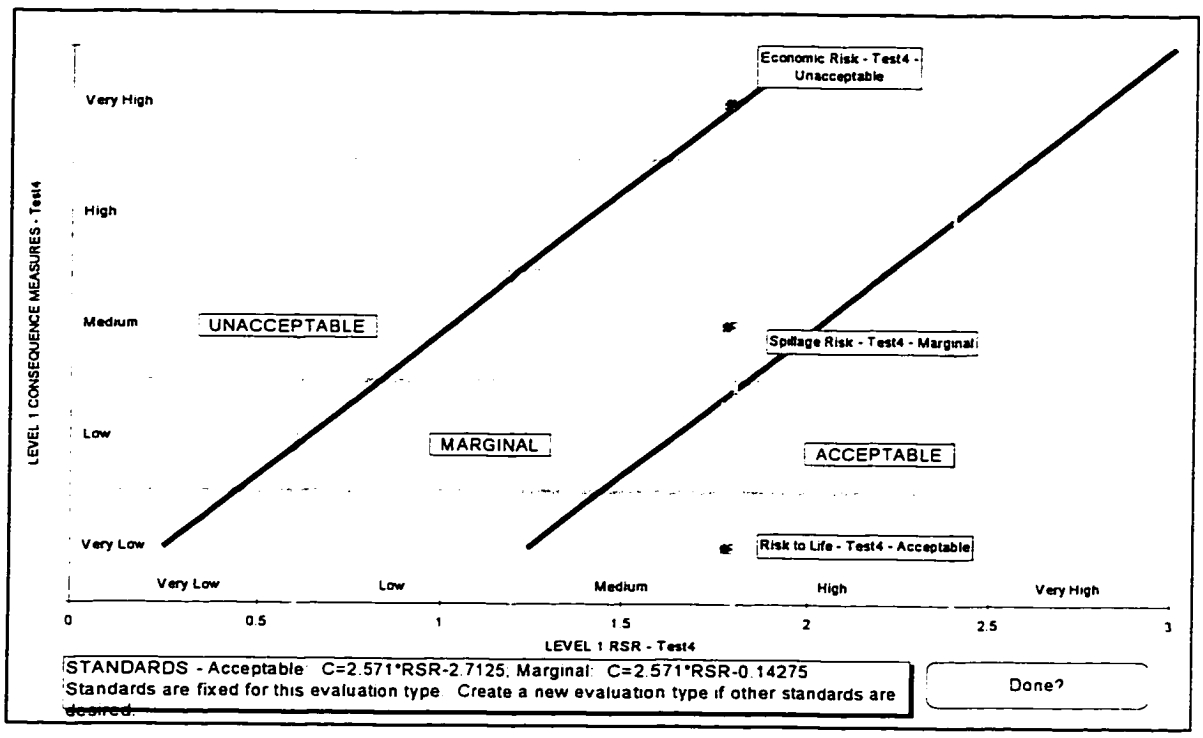


Figure 6-30: L1RSR1 Risk Evaluation Chart

6. Follow the instructions in the Excel dialog box: use <ALT><TAB> to switch back to Access and give it permission to close Excel. PIMS will then save the L1Risk1 outputs.

7. Click on "Enter Other Req'd Data / See Results". This opens the "Data Entry / Results" form. Use the arrow buttons to see the different L1Risk1 outputs. If the output

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

chart is not completely visible, use <edit><copy> to put it in memory, open a graphics editor such as Paintbrush, and paste the chart. The entire chart will then be visible.

6.3.7 Creating an "L1Risk1_Fleet1" Scenario

Like the L1Risk1 evaluation type, the L1Risk1_Fleet1 ("Level One Fleet Risk, version 1") scenario type depends heavily upon the results of the L1RSR1 and L1C1 evaluation types (which in turn depends on the L1I1 evaluation type). Rather than create an L1Risk1 risk assessment chart for a single platform, an L1Risk1_Fleet1 scenario creates a storm-wave induced risk assessment chart for an entire fleet of structures. As with the L1Risk1 evaluation type, L1Risk1_Fleet1 considers consequences categorically (and uses the same arbitrary weighting factors as L1Risk1). Also like the L1Risk1 evaluation type, the user must not be misled because of the false impression of deterministic results. Uncertainties may exist in the results which are not shown.

An L1Risk1_Fleet1 scenario requires that an L1C1 evaluation and an L1RSR1 evaluation for each platform in the selected fleet be designated as inputs. A bitmap of the L1Risk1_Fleet1 chart is produced as output (Table 4-5).

To proceed:

1. From the Main Menu, select Add - Scenarios - Individuals to open the "FLEET OPERATIONS: Add New Scenarios" form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

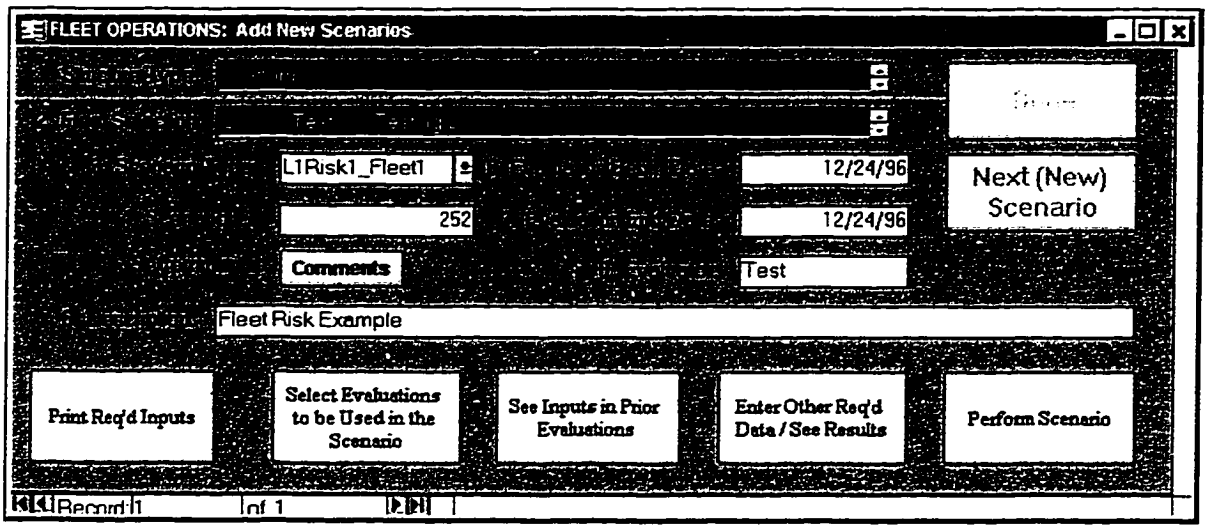


Figure 6-31: Adding an L1Risk1_Fleet1 Scenario

2. Pick a structure type, fleet, and scenario type ("L1Risk1_Fleet1").

3. To assign L1C1 and L1RSR1 evaluations to each of the fleet's platforms, click the "Select Evaluations to be Used in the Scenario".

4. This opens form "FLEET OPERATIONS: Register Evaluations with Scenario". Each platform in the fleet is presented twice in the form - once for assignment of an L1RSR1 evaluation, and once for assignment of an L1C1 evaluation. Move down the form, selecting input evaluations with the drop boxes as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

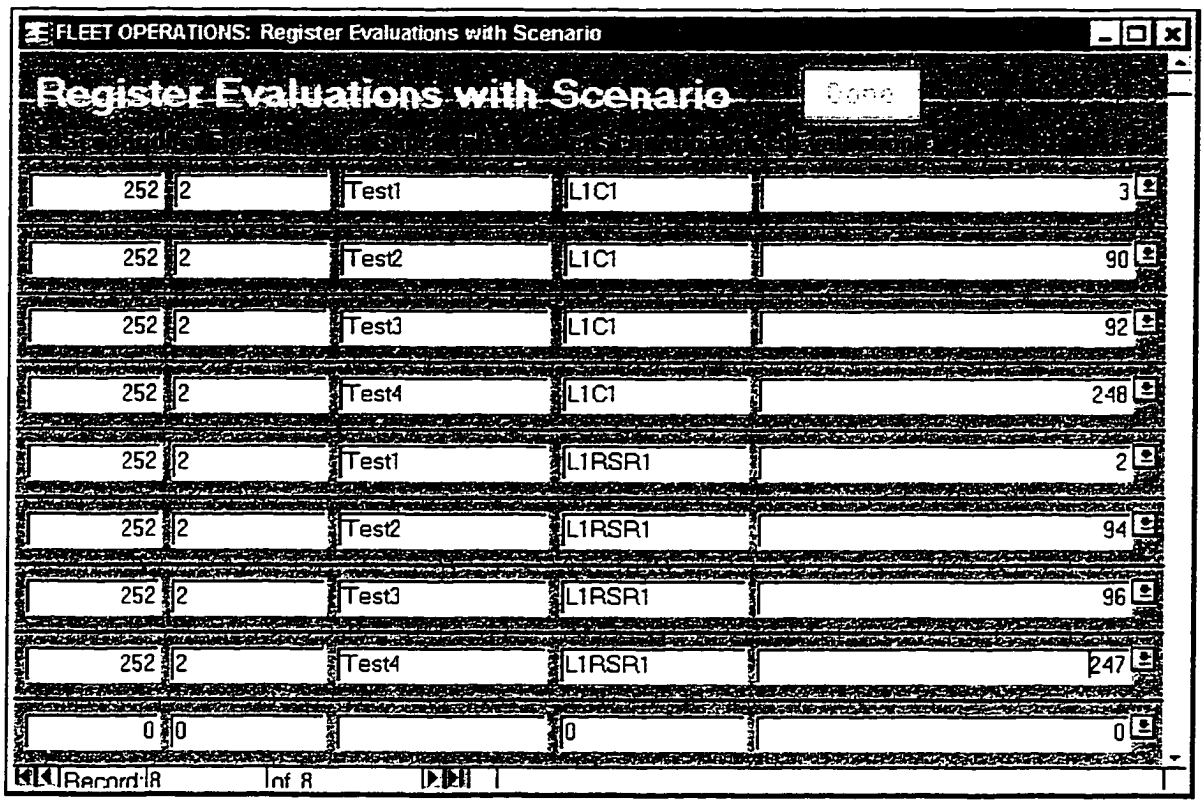


Figure 6-32: Picking L1C1 and L1RSR1 Evaluations For Use With L1Risk1_Fleet1 Scenario

5. Click on the Done button to close the form. Back in the "FLEET OPERATIONS: Add New Scenarios" form, click on "See Inputs in Prior Evaluations" to see the "Prior Scenario Data" form. Close that form when finished.

6. Back in the "FLEET OPERATIONS: Add New Scenarios" form, click on "Perform Scenario" to perform the actual scenario. Excel will open the files "L1fr3.xls" and "Fltrisk.xls," import the appropriate data, and present the L1Risk1_Fleet1 Chart for inspection.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

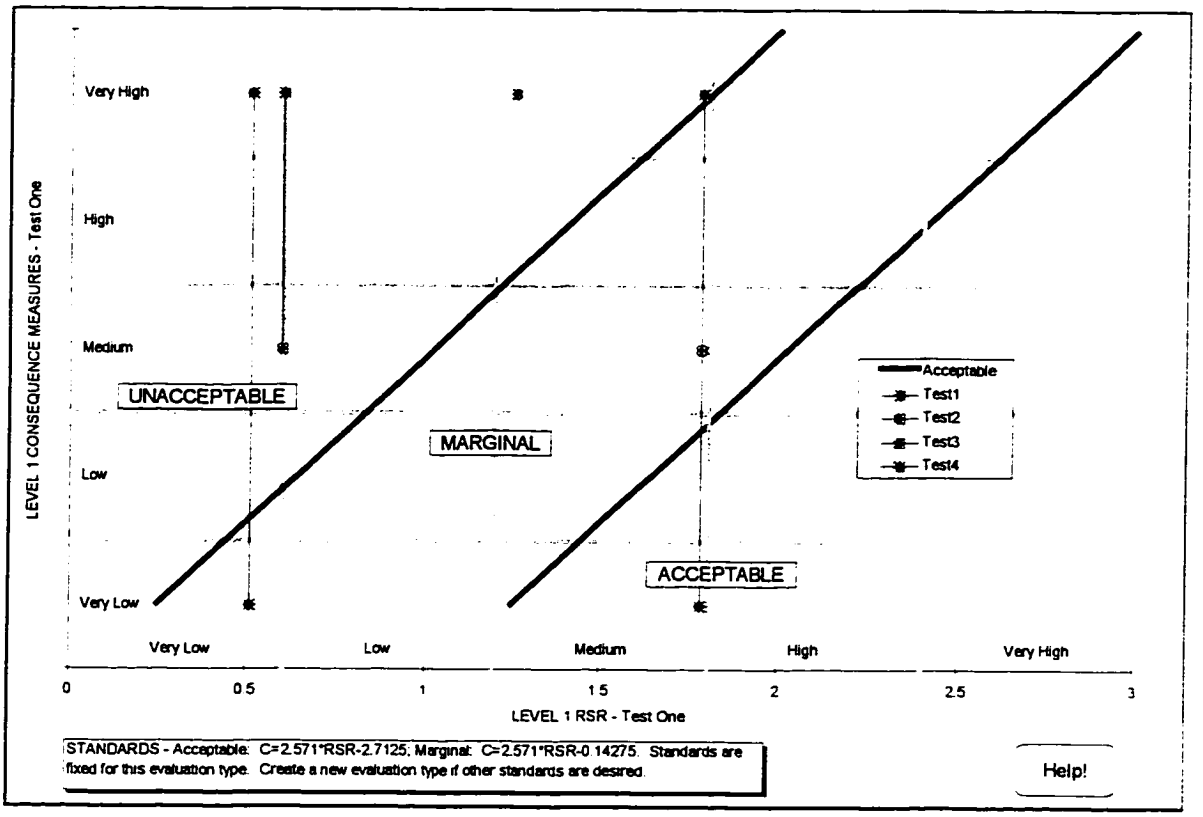


Figure 6-33: L1Risk1_Fleet1 Results Chart

7. This chart may hold up to 200 platforms. Each platform will be presented as a vertical line consisting of three points. Each of these points represents a consequence category (as in the L1Risk1 chart). If the points overlap (i.e., two or more consequence measures for a platform have the same value), only two or maybe just one of the points will be visible for that platform. It is occasionally necessary to move some of the data labels in order to see a particular series.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

8. Generally, platforms plotted towards the upper left of the L1Risk1_Fleet1 chart are less safe. These should be targeted for further analysis. To select one of these, double click a data series to show the platform's name and initiate a return to Access:

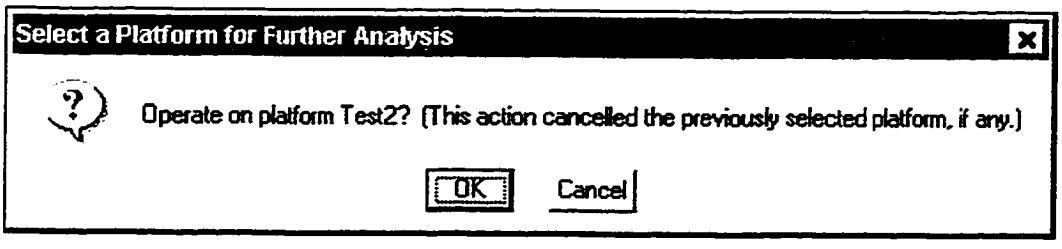


Figure 6-34: Selecting a Platform for Further Analysis

9. Close the dialog box in Excel and return to Access. OK the closing of Excel.

10. Your options now include:

a) Reviewing the information in the L1 inspection that led to these results for the selected platform, and

i) Creating a new L1 inspection for the platform, as well as new L1 structural and consequence evaluations, and a new L1 fleet risk scenario (this iterative approach retains the previous scenario for future reference); or

ii) Changing the present L1 inspection for the selected platform and then re-running its subsequent L1 structural and consequence evaluations, as well as the current L1 fleet risk scenario (this overwrites current data);

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

b) Performing an L2 evaluation on the platform;

c) Noting the situation and proceeding with other operations.

6.3.8 Creating an "L2I1" Inspection

An L2I1 ("Level Two Inspection, version 1") inspection evaluation is necessary to provide data to the L2RSR1 and L2RSR2 evaluations. An L2I1 inspection is hierarchical: it stores data not just about the structure, but about sections and subsections of the structure. Refer to Tables 4-7 through 4-13 for listings of collected data.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L2I1" as the evaluation type.

2. Click the "Perform Evaluation" button to begin the inspection. The "Input Type" dialog box will open, asking whether to input data manually, or import it from an existing Excel ULSLEA [Mortazavi 1996] file:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

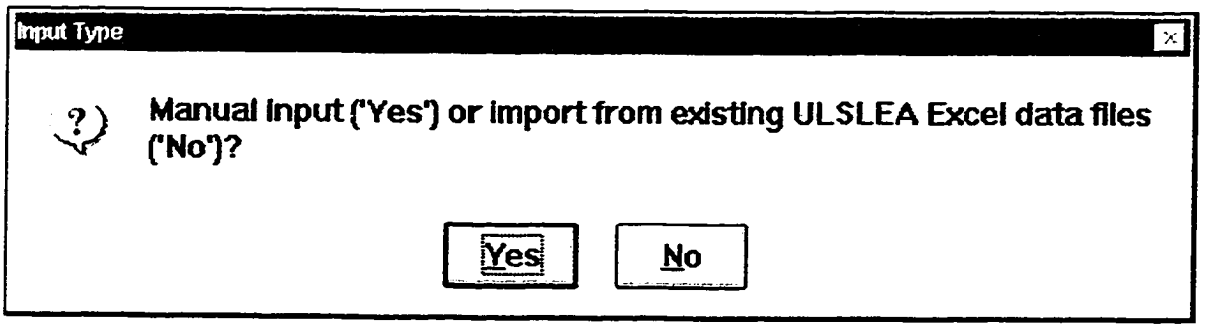


Figure 6-35: Selecting an L2I1 Input Method

3. Answer "No" to start the Excel import feature. When asked, input the DOS format (8.3) path and file name of the ULSLEA data file to be imported (if no such file is available, skip to #10 below). The full path and file name of the ULSLEA file must be in old-fashioned DOS notation; no long file names or "~" abbreviations are allowed.

4. Excel will open; within it, manually open the ULSLEA data file. Copy all the data in the ULSLEA file by going to cell A1 (or, in different notation, R1C1), holding the <SHIFT> key, and hitting <CTRL><END>, and then hitting <CTRL><INSERT>.

5. Move (via <CTRL><F6>, or by using the Window menu) to worksheet L2RSRin1.xls Sheet1. Go to cell A1, and press <SHIFT><INSERT> to paste the data. Next, from the Tools-Macro menu, run macro CountInputs. This will take a few minutes. (Alternatively, skip running the CountInputs macro - which runs as long as half an hour on a 486, and is not vitally important - and enter "1000" in the next step.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6. Excel will state the number of inputs in the ULSLEA file. Return to Access and input this number in the box provided.

7. PIMS will now import the data. A status bar across the bottom of the screen will indicate the progress of the work. This will also take a few minutes.

8. When finished, PIMS will close Excel without saving the ULSLEA file or the temporary file L2RSRIn1.xls.

9. Do not use the "Enter Other Req'd Data / See Results" button to see the inspection data; the hierarchical nature of the data will not make sense out of context of an L2RSR evaluation. Instead, click the "Perform Evaluation" button again, and choose manual input. The imported data will show in the procedure. Since the data are now in the system, there is no need to execute all of the manual input sub-procedures.

10. (From step #3) To input data manually, choose "Yes" from the "Input Type" dialog box. The "Choose a Procedure" form will open:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

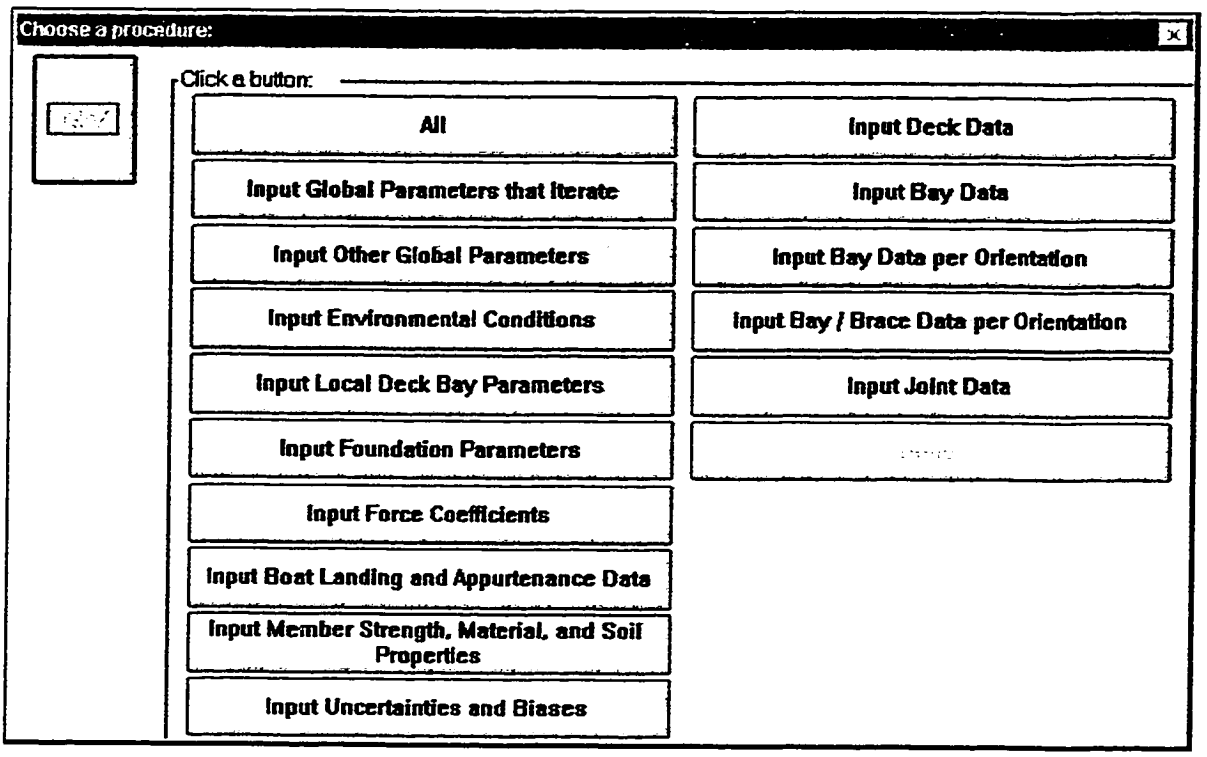


Figure 6-36: Manual L2I1 Inspection Data Entry Options

11. Select a category, select "Go!", then proceed with data entry. Start with the "Global Parameters" input categories, since these provide necessary information for later categories. Each category will ask the appropriate number of questions for the structure in question and then quit, returning to the "Choose a Procedure" form. Unfortunately, it is normal to receive an "Oops" error message after making the first input selection. Clear the message and continue.

12. When finished, select "Done" and then press "Go!".

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

13. Again, do not use the "Enter Other Req'd Data / See Results" button to see the inspection data.

6.3.9 Creating an "L2Engl" Evaluation

The L2Engl ("Level Two Engineering Assumption, version 1") evaluation type is an example of an "engineering assumption" evaluation type, the mechanism by which non-inspection, non-structural data can be accounted for in a subsequent evaluation. An L2Engl evaluation stores one value, the "L2RSR Prelim" setting. This setting tells an L2RSRx evaluation whether or not to calculate preliminary values according to the UL-SLEA methodology, and is strictly an engineering choice.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L2Engl" as the evaluation type.

2. Click on "Enter Other Req'd Data / See Results" to open the "Data Entry / Results" form. Enter "yes" or "no" and then close the form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.3.10 Creating an "L2VMax1" Evaluation

The L2VMax1 ("Level Two Maximum Wind Velocity, version 1") evaluation type is based on the Level One Maximum Wind Velocity sub-routine of the California Coastal Platform Information Management System [Staneff et al 1994]. It is included as an example of using Excel as an input form to PIMS, and should not be applied to Gulf of Mexico platforms.

An L2VMax1 evaluation develops and stores the expected maximum wind velocity, given the maximum wave height. The expected maximum wind velocity is a probabilistic value; an L2VMax1 evaluation also develops and stores the coefficient of variation of the maximum wind velocity (the COV reflects the user's confidence in the VMax value). The VMax chart used to determine the other two values is also stored.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L2VMax1" as the evaluation type.

2. Click on "Select Prior Evaluations to be Used as Inputs" to open the "Prior Evaluation Entry" form. Select a value for the L2I1 evaluation to be used as input, and close the "Prior Evaluation Entry" form (L2VMax1 uses the HMax value from L2I1).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. Click "Perform Evaluation" to perform the L2VMax1 analysis:

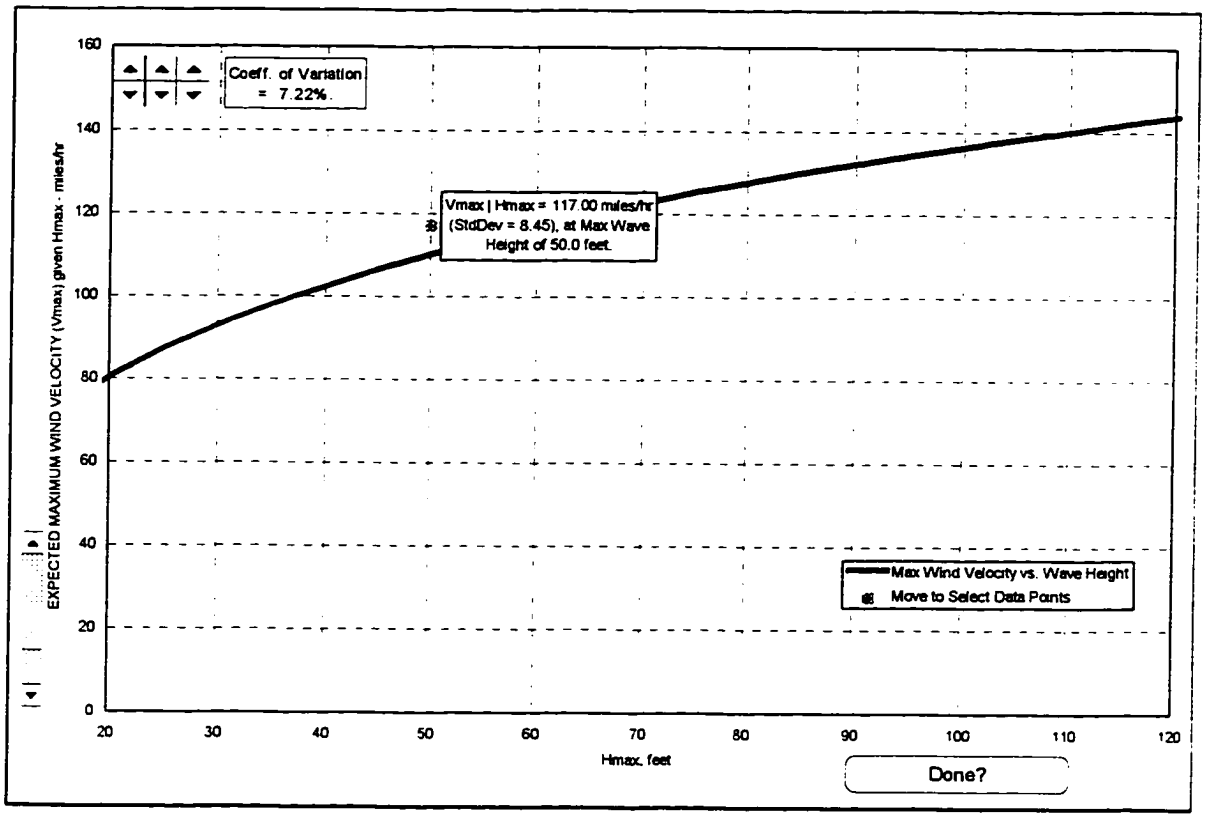


Figure 6-37: L2VMax1 Data Input Chart

4. Use the slide on the lower left of the Excel chart to move the VMax value to the curve. This well yield the correct value of VMax given HMax. Use the arrows at the upper left to adjust the value of the VMax COV. Changing values will be reflected in the data point's label. Click on Done when finished, and follow the subsequent instructions (the L2VMax1 evaluation type requires quite a bit more user assistance than do other evaluation types, but the instructions are self-explanatory).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. After Excel has closed, click on "Enter Other Req'd Data / See Results" to see the calculated values of VMax and VMax COV, as well as the VMax chart.

6.3.11 Creating an "L2RSRx" Evaluation

There are two L2RSRx ("Level Two Reserve Strength Ratio, version x") evaluation types included with PIMS: L2RSR1 follows the ULSLEA 2.0 methodology [Mortazavi 1995], and L2RSR2 follows the revised ULSLEA 2.1 methodology [Stear and Bea 1997]. Both are designed to provide "a simplified analytical method for (the) assessment of structural reliability of template-type offshore platforms" [Mortazavi 1995] due to storm waves. An L2RSRx evaluation requires inputs from an L2I1 inspection, a PlatformData inspection (for water depth), and an L2Eng1 inspection. An L2RSR2 evaluation also requires a VMax input from an L2VMax1 evaluation. Outputs are listed in Table 4-16, and include estimates of the probability of failure. These probabilities are not part of the original UL-SLEA methodologies, but are derived from their respective ULSLEA beta values:

$$P_f = 0.475e^{-(Beta^{1.6})}, Beta \geq 0 \quad \text{[Bea, 1990]}$$
$$P_f = 1 - \left(0.475e^{-(Beta^{1.6})}\right), Beta < 0 \qquad (2)$$

Since ULSLEA calculations are conditional on wave heights, these calculated probabilities of failure are also conditional on wave height (properly, "P_f | wave height"). These probabilities of failure are included for demonstration purposes only.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L2RSR1" as the evaluation type.

2. Click on "Select Prior Evaluations to be Used as Inputs" to open the "Prior Evaluation Entry" form:

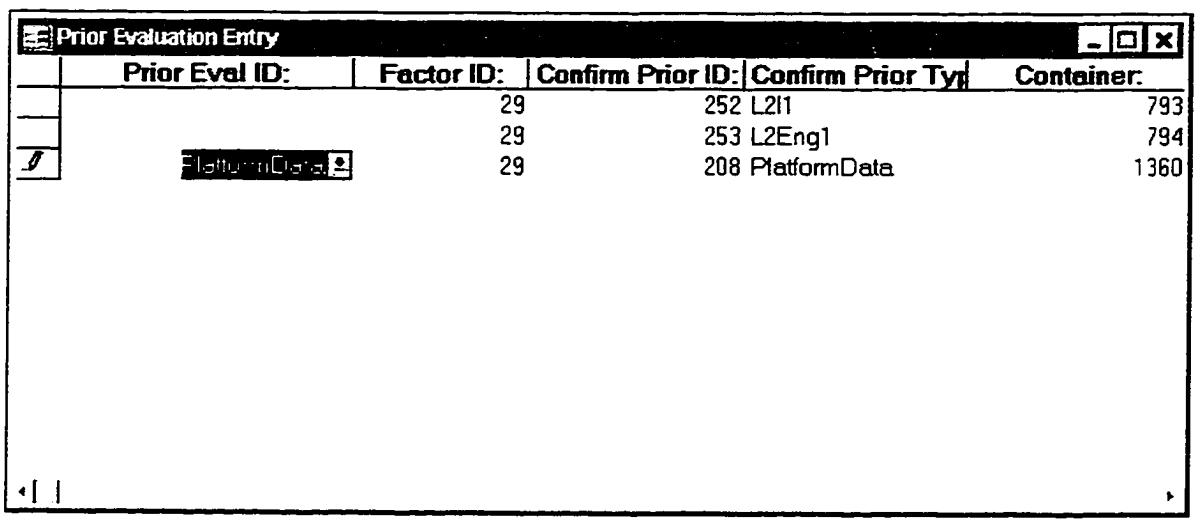


Figure 6-38: Selecting an Inspection, an Engineering Assumptions Evaluation, and a Platform Data Evaluation For Use With an L2RSR1 Evaluation

3. Assign evaluations for each of the necessary types: L2I1, L2Eng1, and PlatformData. Close the "Prior Evaluation Entry" form.

4. Click on "See Inputs in Prior Evaluations" to see the inputs to the evaluation. There will usually be more than a thousand inputs listed.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. Click the "Perform Evaluation" button to start the evaluation. No user input is required. The evaluation takes a few minutes.

6. Clicking the "Other Req'd Data / See Results" button will show the stored probabilities of failure as well as charts similar to those seen in Figures 4-8 through 4-13.

6.3.12 Creating an "L2C1" Evaluation

The L2C1 ("Level Two Consequence, version 1") evaluation type serves as a one-input inspection routine for the L2Risk1 evaluation type. That input is C_M, an estimate of the consequences of failure. C_M is intended to include all of the anticipated consequences of a failure, as expressed in dollars.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L2C1" as the evaluation type.

2. Click the "Enter Other Req'd Data / See Results" button to open the "Data Entry / Results" form. Enter a value for C_M.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.3.13 Creating an "L2Eng2" Evaluation

The L2Eng2 ("Level Two Engineering Assumption, version 2") evaluation type is an example of an "engineering assumption" evaluation type, the mechanism by which non-inspection, non-structural data can be accounted for in a subsequent evaluation. An L2Eng2 evaluation stores three values for later use in an L2Pf3 evaluation (see below for a description of the L2Pf3 evaluation type):

- "Number of Iterations" tells L2Pf3 how many times to iterate. A value between 500 and 1500 is suggested.

- "Mean" is the mean of the lognormal distribution of the wave heights to be generated by L2Pf3.

- "Standard Deviation" is the standard deviation of the lognormal distribution of the wave heights to be generated by L2Pf3.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L2Eng2" as the evaluation type.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. Click on "Enter Other Req'd Data / See Results" to open the "Data Entry / Results" form. Enter the number of iterations for L2Pf3 and the mean and standard deviation of the wave height distribution.

6.3.14 Creating an "L2Pf3" Evaluation

The L2Pf3 ("Level Two Probability of Failure, version 3") evaluation type is an example of handling uncertainties through iteration. It produces an estimate of the annual probability of failure of a platform due to storm waves. To do this, the evaluation first generates a lognormal distribution of wave heights according to the mean and standard deviations established in the prerequisite L2Eng2 evaluation. It then uses the L2RSR2 ULSLEA methodology [Stear and Bea 1997] to produce a probability of failure at each wave height. The estimated overall annual probability of failure due to storms (P_{fa}) is then the sum of the products of the probability of failure given wave height ($P_{f|H}$) and the probability of that wave height (P_H):

$$P_{fa} = \sum_1^N P_{f|H} \cdot P_H \,, \qquad (3)$$

where N is the number of iterations specified in L2Eng2. The greater the value of N, the better the estimate of P_{fa}. The mean and standard deviation values for wave height (H) should be the mean and standard deviation of the actual values of H, and not of the logs of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the values. The methodology used to generate the random lognormal distribution is based on that of Sprott [1994].

IMPORTANT: The ULSLEA 2.1 methodology as used in L2RSR2 and in L2Pf3 is designed for extreme wave heights, and performs poorly at lower wave heights. In tests, ULSLEA 2.1 produced a zero result for P_{flH} for all wave heights below 26 feet for one platform, and for all wave heights below 36 feet for another. Users wishing to investigate the effects of low wave heights upon the overall value of P_{fa} should use the included L2Pf1 evaluation type, which saves intermediate values to a temporary table. Those values can then be copied to a spreadsheet, charted, and examined for discontinuities.

An L2Pf3 evaluation requires that an L2I1 inspection, an L2Eng1 evaluation, a PlatformData evaluation, an L2VMax1 evaluation, and an L2Eng2 evaluation be designated as inputs. The sole result is an estimate of P_{fa}.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L2Pf3" as the evaluation type.

2. Click on "Select Prior Evaluations to be Used as Inputs" to open the "Prior Evaluation Entry" form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. Use the drop-down list on the first line to select one each of the available L2I1, L2Eng1, PlatformData, L2VMax1, and L2Eng2 evaluations. Use the upper-right control box to close the form.

4. Click on "See Inputs in Prior Evaluations" to see the input data. Close that form.

5. Click "Perform Evaluation" to perform the L2Pf3 analysis. Expect the procedure to take about 1.5 to 3 seconds per iteration. Click the "Done" button on the Excel chart when finished.

6.3.15 Creating an "L2Risk1" Evaluation

The L2Risk1 ("Level Two Risk, version 1") evaluation type included with PIMS is based on the risk evaluation methodology of Bea [1990]. The methodology utilizes the two components of risk - the likelihood of failure and the consequences of failure - to estimate a measure of risk. The likelihood of failure is represented by the annual probability of failure, P_{fa}, while the consequences of failure (C_M) are measured in dollars. The former is meant to encompass the entire spectrum of the causes of failure, while the latter is meant to encompass the entire spectrum of the costs of failure. C_M is the result of evaluation L2C1, and P_{fa} in this case is the result of evaluation L2Pf3. Unfortunately, this value of P_{fa} is an estimate only of structural failures due to wave impacts.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The L2Risk1 methodology plots the annual probability of failure against consolidated consequences of failure to achieve an assessment of risk. This is a subjective assessment, plotted deterministically on a chart that cannot convey uncertainties. Data points on the output chart should be interpreted as circles or ovoids, possibly overlapping much thicker "Acceptable" or "Marginal" lines, and labeled as "probably acceptable" or "probably marginal."

An L2Risk1 evaluation requires that an L2C1 and an L2Pf3 evaluation be designated as inputs. The L2Risk1 chart (Figure 6-39) and the assessment of "Acceptable," "Marginal," or "Unacceptable" are saved as outputs.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L2Risk1" as the evaluation type.

2. Click on "Select Prior Evaluations to be Used as Inputs" to open the "Prior Evaluation Entry" form.

3. Use the drop-down list on the first line to select one of the available L2C1 evaluations. Use the drop-down list on the second line to select one of the available L2Pf3 evaluations. Use the upper-right control box to close the form.

4. Click on "See Inputs in Prior Evaluations" to see the input data. Close that form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. Click "Perform Evaluation" to perform the L2Risk1 analysis. Excel will open the file "L2Risk1.xls," import the appropriate data, and present the L2Risk1 Chart for inspection. To see the plotted points, it is occasionally necessary to manually move some of the data labels. Click the "Done" button on the Excel chart when finished.

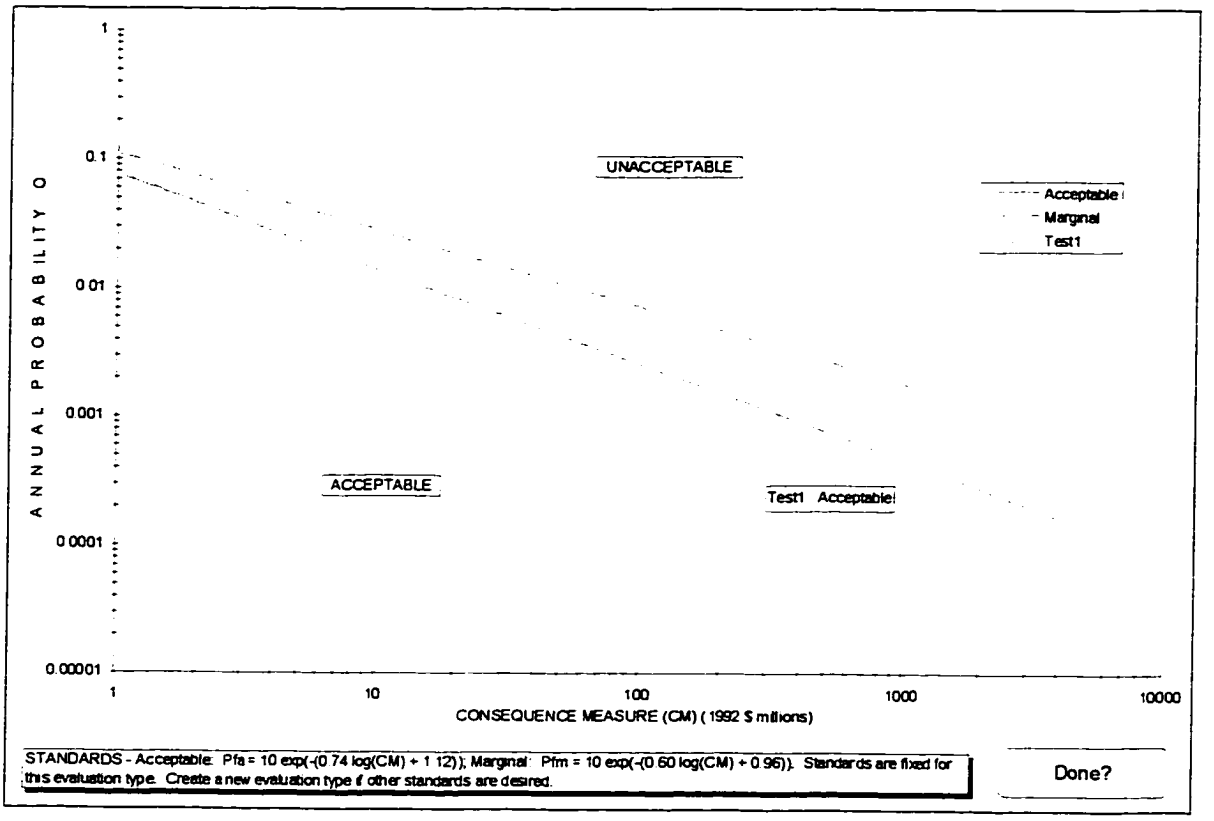


Figure 6-39: L2Risk1 Risk Evaluation Chart

6. Follow the instructions in the Excel dialog box: use <ALT> <TAB> to switch back to Access and give it permission to close Excel. PIMS will then save the L2Risk1 outputs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

7. Click on "Enter Other Req'd Data / See Results". This opens the "Data Entry / Results" form. Use the arrow buttons to see the different L2Risk1 outputs. If the output chart is not completely visible, use <edit> <copy> to put it in memory, open a graphics editor such as Paintbrush, and paste the chart. The entire chart will then be visible.

6.3.16 Creating an "L1HMax1" Evaluation

The L1Hmax1 ("Level One Maximum Wave Height, version 1") evaluation type, as included, essentially serves as a one-input inspection routine for the L1VMax1 evaluation type. That input is HMax, the maximum wave height.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L1HMax1" as the evaluation type.

2. Click the "Enter Other Req'd Data / See Results" button to open the "Data Entry / Results" form. Enter a value for HMax.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.3.17 Creating an "L1VMax1" Evaluation

The L1VMax1 ("Level One Maximum Wind Velocity, version 1") evaluation type is based on the Level One Maximum Wind Velocity sub-routine of the California Coastal Platform Information Management System [Staneff et al 1994]. It is included as an example of using Excel as an input form to PIMS, and should not be applied to Gulf of Mexico platforms. It is very similar to the L2VMax1 evaluation type.

An L1VMax1 evaluation develops and stores the expected maximum wind velocity, given the maximum wave height. The expected maximum wind velocity is a probabilistic value: an L1VMax1 evaluation also develops and stores the coefficient of variation of the maximum wind velocity (the COV reflects the user's confidence in the VMax value). The VMax chart used to determine the other two values is also stored.

To proceed:

1. Pick a structure type, fleet, and platform to be evaluated in the "STRUCTURE OPERATIONS: Add New Evaluations" form. Select "L1VMax1" as the evaluation type.

2. Click on "Select Prior Evaluations to be Used as Inputs" to open the "Prior Evaluation Entry" form. Select a value for the L1HVax1 evaluation to be used as input, and close the "Prior Evaluation Entry" form.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. Click "Perform Evaluation" to perform the L1VVax1 analysis:

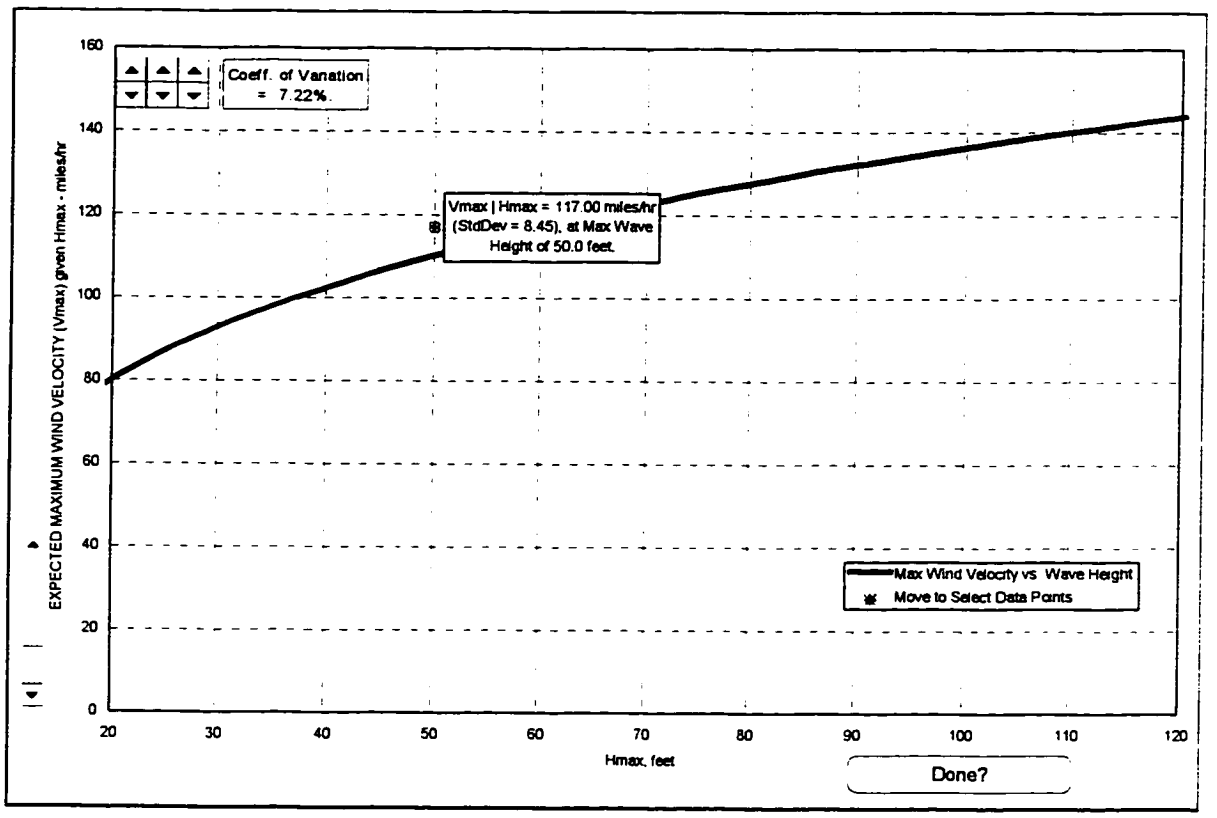


Figure 6-40: L1VMax1 Data Input Chart

4. Use the slide on the lower left of the Excel chart to move the VMax value to the curve. This well yield the correct value of VMax given HMax. Use the arrows at the upper left to adjust the value of the VMax COV. Changing values will be reflected in the data point's label. Click on Done when finished, and follow the subsequent instructions (the L1VMax1 evaluation type requires quite a bit more user assistance than do other evaluation types, but the instructions are self-explanatory).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. After Excel has closed, click on "Enter Other Req'd Data / See Results" to see the calculated values of VMax and VMax COV, as well as the VMax chart.

6.4 SUMMARY

PIMS - the Platform Information Management System - is an effective PC-based tool for the management and integration of information used in engineering analyses. Designed to be applied to infrastructure systems (fleets as well as individual structures), PIMS may be customized to meet users' needs. PIMS uses a combination of relational and object-oriented database technologies to provide flexibility while ensuring data integrity.

Sample engineering analyses are provided with PIMS to illustrate its applicability to the fleet of offshore oil platforms in the Gulf of Mexico.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

7 CONCLUSIONS

7.1 SUMMARY

In the quest to develop an information system for infrastructure management, this research developed a hybrid database schema, "Relational Object" technology (Chapter 3). Relational Object technology splits the database model into a relational substructure and an object-oriented superstructure, and uses the substructure to manage factors rather than sets of factors. Implemented in DAESSIM (the Decision, Analysis, and Engineering Support System for Infrastructure Management), Relational Object technology provides an effective tool for the management and integration of information used in engineering analyses. Designed to be applied to the analysis of risk for infrastructure systems (fleets as well as individual structures), DAESSIM serves as a shell that may be customized to meet users' needs.

That ability was demonstrated in Chapter 4 with PIMS (the Platform Information Management System), an instance (version) of DAESSIM. PIMS was created to show that DAESSIM incorporates the desired characteristics of an infrastructure information management system, as developed in Chapters 2 and 3. The methodologies, attributes, storage protocols, and interrelationships of the sample evaluation objects included in PIMS were discussed in the context of two major example types of "information flow" – 1) the information flow leading to a subjective assessment of the structural risk of an oil platform due to storm waves, and 2) that leading to an objective assessment of the reliability

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of an oil platform due to storm waves. These information flows were used to provide insight into the structure of DAESSIM.

Some of the DAESSIM characteristics demonstrated were:

- Storage and display of, and user interaction with, data and results.
- Ability to perform either a quick database search, or a simplified review of all structures, as dictated by available historical records and analysis method.
- Ability to compare results from different structures to determine relative degree of need (fleet risk).
- Ability to retrieve data for subsequent reassessment upon alteration of decision criteria or updating of analysis methodology.
- Modifiable decision criteria/safety standards.
- Modularity, for updating/replacing analysis procedures (technology transfer).
- Decision support functionality without usurping the decision maker's role.
- Ability to handle biases and/or uncertainties, as dictated by available analysis methods.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Extensibility for handling unknown and unanticipated (at design time) analysis types and structure types.

- Flexibility of handling non-standard hierarchical schemes.

- Non-proprietary format.

- Data integrity.

- *Ad hoc* query capability.

- Referential integrity.

- Interoperability with existing or external applications.

- Ability to track structure changes with time.

Perhaps the ultimate use to which an infrastructure information management system can aspire is as a risk screening system. Chapter 5 developed a sample 4-level screening system for later integration with DAESSIM. The system allows the user to employ common sense and economics to move to another structure within a fleet, to iterate a present analysis on the present structure using different assumptions, or to decide whether to accept a "safe" result for a structure. The latter, a recognition of the significant potential effects of even a small probability that an unsafe structure has been declared "safe," was analyzed in detail with a decision tree model. It was found to be both possible and neces-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

sary to calculate, rather than estimate, conditional probabilities for a multi-level screening system (starting with a small number of initial estimates of test quality). The ability to skip from an *i*-level test to an *i*+2 or *i*+3 level test was found to be critical in the decision model, as was the need to account for the time lag between screens. The assumption that screening system tests should decrease in conservatism with increasing complexity and cost was challenged, and found to be incorrect in one case. Given the example data used for demonstrating the model, the most influential factors upon the decision whether or not to continue testing after receiving a Level One "safe" result were found to be failure cost, the quality of a Level Four test, and the number of years until the next screening.

Finally, Chapter 6 provided detailed instructions for the operation and customization of PIMS.

7.2 FUTURE WORK

Opportunities abound for further work on converting the DAESSIM prototype into a practical implementation, and on application programs to serve as DAESSIM evaluation types.

7.2.1 DAESSIM Implementation Issues

On the implementation side, the top priority of future developers should be to scale up DAESSIM's relational object substructure from Microsoft Access™ to an industrial-strength database management system. Moving DAESSIM's programming language to a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

compiled language that supports object inheritance would tighten up the superstructure. Centralizing the database (through a client-server setup, a web site, or some other methodology) will be necessary to ensure data integrity (as is the case with all mission-critical databases). Such a centralized system could be located on a corporate server for use by its client engineering office(s), or on a regulator's computer for analysis and submittal use by industry (hierarchical object-management tables would reside on the central server to provide compatibility, but factor data would stay on the client systems until officially submitted).

For example, a workable arrangement might be for a regulator's central database to serve industrial client computers, which themselves would act as servers for portable computers with wireless links. A field engineer could then enter on-site inspection data immediately, handing it over to a remote engineer for analysis. Once sufficient analysis iterations, repairs, and re-inspections have been carried out (perhaps while the field engineer is still on-site), appropriate data would be submitted by the corporate system to the regulator's server.

One of the biggest disadvantages of the present state of relational object technology is the number of null values in the FACTOR_DATA table - and, consequently, the wasted memory and disk space. A single field with a flexible data type, incorporating all of the data types required by DAESSIM, would solve this problem. It is possible that some of the new "object relational" or "relational extension" database products will provide such capabilities.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A disturbing trend in computing, which seems to migrating to the mainframe arena, is the use of external programs to perform certain functions. Later, when an external component is upgraded, the systems are no longer necessarily compatible. For example, DAESSIM uses Microsoft Excel™ version 7 for graphing operations, but cannot use version 8. An industrial implementation of DAESSIM must not submit to the temptation to use external components – once one user "upgrades," the whole system would be disabled.

7.2.2 DAESSIM Applications

Engineers will continue to find new things to analyze, and to develop new methods of analysis. DAESSIM's greatest advantage is that it removes from the engineer the distraction of designing and implementing a data structure for each new type of analysis.

Oil platforms, dams, buildings, wharves, piers, pipelines, railroad bridges, and highway bridges are just a few of the large civil engineering structures that can be managed by DAESSIM. Nor is DAESSIM confined to civil engineering – by stretching the "structure" analogy, DAESSIM can be used for such "trivial" tasks as document management or research note collection.

In its present implementation, DAESSIM's uses as an analytical tool are limited primarily by the evaluation types available to it (as included in PIMS). There is no reason why

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

new evaluation types cannot be written for DAESSIM incorporating advanced methodologies, such as:

- Multiple causes of structure failure (e.g., operational error, seismic events)
- Multiple levels of failure (e.g., minor damage, major damage, collapse)
- Cost/benefit economics for repair alternatives
- Advanced structural analysis routines
- Multiple types of risks (e.g., risks to life, risks to the environment)
- Statistical analysis packages
- Advanced risk screening cycles (i.e., Model 5t of Chapter 5)
- Broad-based decision analysis packages

Future work might also concentrate on functions to be included in DAESSIM evaluations. For example, the existing L2Pf3 evaluation type mentioned briefly in Chapter 6 both randomly generates a log-normal distribution and iterates an L2RSR2 routine to produce a probabilistic result. This can form the basis for Farmer Curve analyses. A future function might enable a screening cycle to perform background iterations on a Level i test while a more complex Level $i+n$ test is being conducted. Then, if the result of the more complex test is "unacceptable," the many Level i trials can be inspected for possible

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

design/repair alternatives. A simple genetic algorithm running within user-defined constraints might work well in this capacity. Other future work might ease the process by which alternative safety standards are developed in routines such as L1Risk1_Fleet1.

7.2.3 Comparison with STEP

International Standards Organization (ISO) standard 10303 is known as STEP, the Standard for the Exchange of Product Model Data. STEP is an attempt to provide universal data files to every branch of industry, using object-oriented database technology. STEP takes an aggressive stance towards defining standard data packages ("entities") to represent every possible "thing," and then providing the standards to the engineering, shipping, manufacturing, software, etc., industries. Good success has been achieved in standardizing software formats like graphics files and moderate success in manufactured pieces like pumps; little progress has been made with customized pieces like civil engineering structures. It's difficult to pre-define, in a traditional database context, every possible brace geometry in an oil platform or every variation of CMU wall in a building. DAESSIM's relational-object technology, in contrast, is designed specifically to deal with such problems.

There is a major philosophical difference between the STEP initiative and DAESSIM: STEP must identify and define all entities before working with them, while DAESSIM takes a *laissez-faire* approach, passively sitting there until someone shows enough interest in an entity to define the necessary factors.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Implementation of DAESSIM as an open standard for data exchange would necessarily be based on a central server, possibly a web site maintained by a university. Evaluation types and object data structures (through the hierarchy tables) would be registered and stored on the server, while factor values would be kept only on individual clients' machines. After "priming the pump" to develop interest, perhaps with a student project to port a structural analysis program to the DAESSIM data format, private companies would begin to register their applications as DAESSIM evaluation types (indeed, software companies could advertise their products in this fashion). The first step in a software developer's DAESSIM integration process would be to check to see if necessary objects (e.g., CMU walls) have already been defined by another developer. The public "factor registration" tables would thus become an evolving standard for data exchange, fulfilling the role of the STEP model without waiting to pre-define every item in the world.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

REFERENCES

[Aggarwal 1991] Aggarwal, Rajiv K. "Methodology for Assessment by Regulatory Bodies of the Safety of Existing Steel Offshore Platforms," Berkeley: University of California, Dissertation, 1991.

[API 1993] American Petroleum Institute. "Recommended Practice for Planning, Designing and Constructing Fixed Offshore Platforms (RP 2A)," Washington, DC: American Petroleum Institute, 1993.

[API 1994] American Petroleum Institute (API). "API RP2A-WSD 20th Edition Draft Section 17.0: Assessment of Existing Platforms," Washington, DC: American Petroleum Institute, April 29, 1994.

[Arockiasamy et al 1993] Arockiasamy, M., M. Sawka, V. Sinha, and M. El Shahawy. "Knowledge-Based Expert System Approach to Analysis and Rating of Florida Bridges" in "Bridge Management 2: Inspection, Maintenance, Assessment and Repair," J. E. Harding, G. A. R. Parke, and M. J. Ryall, ed. London: Thomas Telford, 1993.

[Bea 1990] Bea, R. G. "Reliability Based Design Criteria for Coastal and Ocean Structures," Barton, Australia: The Institution of Engineers, 1990.

[Bea 1993] Bea, Robert G. "Reliability Based Requalification Criteria for Offshore Platforms." Glasgow, Scotland: *Proceedings of the 12th International Conference on Offshore Mechanics and Arctic Engineering (OMAE)*, 1993: 351-361.

[Bea and Craig 1993] Bea, R. G., and M. J. K. Craig. "Developments in the Assessment and Requalification of Offshore Platforms," Houston: *Proceedings of the Offshore Technology Conference*, OTC 7138, 1993.

[Beech 1997] Beech, David. "Can SQL3 Be Simplified?" *Database Programming & Design*, 10(1), January 1997: 46-50.

[Beeri 1994] Beeri, Catriel. "Query Languages For Models With Object-Oriented Features," in "Advances in Object-Oriented Database Systems," Dogac, Asuman, M. Tamer Ozsu, Alexandros Biliris, and Timos Sellis, ed. Berlin, Germany: Springer-Verlag, 1994.

[Bresler 1977] Bresler, Boris. "Evaluation of Earthquake Safety of Existing Buildings," in "Developing Methodologies for Evaluating the Earthquake Safety of Existing Buildings," Berkeley: University of California Earthquake Engineering Research Center, *Report No. UCB/EERC-77/06 to the National Science Foundation*, 1977.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[Choi 1992] Choi, Kwang C. "An Object-Oriented Historical Cost Information Base to Support Concurrent Engineering," Berkeley: University of California, Dissertation, 1992.

[Chow 1987] Chow, Tat-Chi. "Forecasting Severe Nuclear Accidents," Berkeley: University of California, Dissertation, 1987.

[Chow 1990] Chow, Tat-Chi, Robert M. Oliver, and G. Anthony Vignaux. "A Bayesian Escalation Model to Predict Nuclear Accidents and Risk," *Operations Research*, 38(2), 1990: 265-277.

[Date, 1995a] Date, C. J. "An Introduction to Database Systems," Reading, Massachusetts: Addison Wesley, 1995.

[Datta and Oliver 1992] Datta, K., and R. M. Oliver. "A Model to Predict Mid-Air and Near-Mid-Air Collision," *Journal of Forecasting*, Vol. 11, 1992: 207-223.

[Davis 1996] Davis, Judith R. "Creating An Extensible, Object-Relational Data Management Environment: IBM's DB2 Universal Database," DataBase Associates International: *Report Prepared for IBM Corporation*, November 1996.

[Dogac et al 1994] Dogac, Asuman, M. Tamer Ozsu, Alexandros Biliris, and Timos Sellis. Foreword. "Advances in Object-Oriented Database Systems," Berlin, Germany: Springer-Verlag, 1991

[Goldberg and Robson 1983] Goldberg, Adele, and David Robson. "Smalltalk-80: The Language and its Implementation," Reading, Massachusetts: Addison Wesley, 1995.

[Goldstein 1982] Goldstein, Bernard D. "The Problem with the Margin of Safety: Toward the Concept of Protection," *Risk Analysis*, 10, 1990: 7-10.

[Henderson 1982] Henderson, Hazel. "Thinking Globally, Acting Locally: the Politics and Ethics of the Solar Age," Berkeley: University of California Press, 1982.

[Hillebrand et al 1994] Hillebrand, Gerd. "Functional programming formalisms for OODBMS methods," in "Advances in Object-Oriented Database Systems," Dogac, Asuman, M. Tamer Ozsu, Alexandros Biliris, and Timos Sellis, ed. Berlin, Germany: Springer-Verlag, 1994.

[Hillier and Lieberman 1990] Hillier, Frederick S., and Gerald J. Lieberman. "Operations Research," New York: McGraw-Hill, 1990.

[Informix 1997] Informix Corp. "Developing DataBlade Modules for Informix – Universal Server," white paper: *http://www.informix.com/informix/corpinfo/zines/whitpprs/databld/21360f0.htm*, 1997.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[Jennings 1994] Jennings, Roger. "Access 2 Developer's Guide, Second Edition," Indianapolis: Sams Publishing, 1994.

[Keeler 1996] Keeler, Michael. "Database of All Trades," *Database Programming & Design*, 9(11), November 1996: 34-42.

[Marshall and Oliver 1995] Marshall, Kneale T., and Robert M. Oliver. "Decision Making and Forecasting," New York: McGraw-Hill, 1995.

[McCann et al 1983] McCann, Jr., M.W., Franzini, J.B., Kavazanjian, Jr., E., and Shah, H., "Preliminary Safety Evaluation Of Existing Dams," Palo Alto: John A. Blume Earthquake Engineering Center, Department of Civil Engineering, Stanford University, *Report Nos. 69 and 70 to the Federal Emergency Management Agency*, 1985.

[McFadden and Hoffer 1994] McFadden, Fred R., and Jeffrey A. Hoffer. "Modern Database Management," Redwood City, CA: Benjamin/Cummings, 1994.

[Miyamoto et al 1993] Miyamoto, A., H. Morikawa, M. Kushida, and T. Tokuyama. "A Knowledge-Based Expert System Application in Structural Safety Assessment of Concrete Bridges," in "Bridge Management 2: Inspection, Maintenance, Assessment and Repair," J. E. Harding, G. A. R. Parke, and M. J. Ryall, Ed. London: Thomas Telford, 1993.

[MMS 1995] United States Minerals Management Service, Department of the Interior, "STRU3394.dbf" (Platforms Database), private correspondence.

[Mortazavi 1995] Mortazavi, Mehrdad. "A Probabilistic Screening Methodology for Use in Assessment and Requalification of Steel, Template-Type Offshore Platforms," Berkeley: University of California, Dissertation, 1995.

[OECD 1992] OECD. "Bridge Management," Paris: Organisation for Economic Co-operation and Development, *Report prepared by an OECD Scientific Expert Group*, 1992.

[Okada and Bresler 1976] Okada, Tsuneo and Boris Bresler. "Strength and Ductility Evaluation of Existing Low-Rise Reinforced Concrete Buildings - Screening Method," Berkeley: University of California Earthquake Engineering Research Center, *Report No. EERC 76-1 to National Science Foundation*, 1976.

[Oliver and Yang 1990] Oliver, Robert M., and H. J. Yang. "Bayesian Updating of Event Tree Parameters to Predict High Risk Incidents," in "Influence Diagrams, Belief Nets and Decision Analysis," R. M. Oliver and J. Q. Smith, Ed. New York: John Wiley & Sons, 1990.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[Puskar et al 1994] Puskar, F. J., R. K. Aggarwal, C. A. Cornell, Fred Moses, and Charles Petrauskas. "A Comparison of Analytically Predicted Platform Damage to Actual Platform Damage During Hurricane Andrew," Houston: *Proceedings of the 26th Annual Offshore Technology Conference*, OTC 7473, 1994.

[Reid 1989] Reid, Stuart G. "Risk Assessment," Sydney: University of Sydney, NSW, Australia, School of Civil and Mining Engineering, *Research Report no. R 591*, 1989.

[Rhoads 1985] Rhoads, Steven E. "The Economist's View of the World: Government, Markets, & Public Policy," Cambridge: Cambridge University Press, 1985.

[Slade 1994] Slade, Cynthia. "Inventory of Dams in Northern California," Berkeley: University of California at Berkeley, *Report to the Bureau of Indian Affairs*, 1994.

[Sprott 1994] Sprott, Julien C. "Numerical Recipes Routines and Examples in BASIC," Cambridge: Cambridge University Press, 1994.

[Staneff and Ibbs 1994] Staneff, Stephen T., and C. William Ibbs, "Risk Evaluation for Marine Structures in a Regulatory Context," Houston: American Society of Mechanical Engineers, *Proceedings of the 13th International Conference on Offshore Mechanics and Arctic Engineering, Vol. II: Safety and Reliability*, 1994.

[Staneff et al 1994] Staneff, Stephen T., C. William Ibbs, and Robert G. Bea. "California Coastal Platform Information Management System - Software Documentation," Berkeley: University of California, *California Sea Grant Report No. R/OE-19*, 1994.

[Staneff et al 1995] Staneff, Stephen T., C. William Ibbs and Robert G. Bea. "A Risk Management System for Infrastructure Condition Assessment," *Journal of Infrastructure Systems*, American Society of Civil Engineers, New York, 1(4), 1995: 221-229.

[Stear and Bea 1997] Stear, J., and Robert G. Bea. ULSLEA 2.1. In publication.

[Stein and Maier 1988] Stein, Jacob, and David Maier. "Concepts in Object-Oriented Data Management," *Database Programming and Design*, 1(4), April 1988.

[Thurston et al 1986] Thurston, Howard M., Weimin Dong, Auguste C. Boissonnade, Farrokh Neghabat, James M. Gere, and Haresh C. Shah. "Risk Analysis And Seismic Safety Of Existing Buildings," Palo Alto: John A. Blume Earthquake Engineering Center, Department of Civil Engineering, Stanford University, *Report No. 81 to the National Science Foundation*, 1986.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[Weissmann et al 1989] Weissmann, Jose, W. R. Hudson, Ned H. Burns, and Robert Harrison. "A Bridge Management System Module for the Selection of Rehabilitation and Replacement Projects," Austin, Texas: Center for Transportation Research, University of Texas at Austin, *Research Report 439-4F*, 1989.

[Wenk 1986] Wenk, Edward Jr. "Tradeoffs: Imperatives of Choice in a High-Tech World," Baltimore: Johns Hopkins University Press, 1986.

[Winner 1986] Winner, Langdon. "The Whale and the Reactor," Chicago: University of Chicago Press, 1986.

[Wright et al 1973] Wright, R., et. al., editors, "Building Practices for Disaster Mitigation," Washington, DC: National Bureau of Standards, *Proceedings of a Workshop held at National Bureau of Standards, Boulder, Colorado, August 28 – September 1, 1972*, N. B. S. Building Science Series 46, N. B. S., Washington, D. C., 1973.

[Zdonik 1994] Zdonik, Stanley B. "What Makes Object-Oriented Database Management Systems Different?" in "Advances in Object-Oriented Database Systems," Dogac, Asuman, M. Tamer Ozsu, Alexandros Biliris, and Timos Sellis, ed. Berlin, Germany: Springer-Verlag, 1994.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

GLOSSARY

Attribute	A characteristic of an entity; represented by a field within a relational database
Blob	Binary Large Object data type – a method of storing bitmaps and other non-traditional data types
C1	Cost of Level 1 test (all test costs, including inspections)
C2	Cost of Level 2 test (all test costs, including inspections)
C3	Cost of Level 3 test (all test costs, including inspections)
C4	Cost of Level 4 test (all test costs, including inspections)
Candidate Key	An attribute that may serve as a primary key
Closure	The property whereby the results of one mathematical operation may be used in another
Combination Primary Key	A primary key consisting of more than one attribute
Contingency Table	Two-way contingency table of event outcomes – a table of counts of event occurrences vs. predictions
D	Yearly number of failures (expected)
DAESSIM	Decision, Analysis, and Engineering Support System for Infrastructure Management
Data Model	A concept guiding the means by which data are managed
Data Schema	A schema within the confines of a data model by which data are managed
Database	An implemented data schema within a DBMS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

DBMS	Database Management System – a software implementation of a data model
Encapsulation	In OO databases, the property by which all data pertaining to an object are contained within it; encapsulated data is hidden from the outside world and does not have to be revealed in order to be used (via methods)
Entity	A thing about which data is collected
Evaluation	An analysis performed on a structure; an instance of an evaluation type
Evaluation Type	A class of evaluation
F	Cost of failure (all costs generated by a structure's failure)
F Matrix	Forecast probability ("likelihood") matrix
Factor	An evaluation's input or output; a variable to be stored
FFP	Fit For Purpose, or Fitness For Purpose
Field	A column of a table
Flat File Database	A database made of a single table
Fleet	A collection of structures
Foreign Key	A candidate key in one relation that maintains a relationship by referencing a non-candidate key in another relation
Form	A database feature for user interaction
Hierarchical Data Model	A data model in which data are arranged in a top-down hierarchical progression, featuring nested records

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

HMax	Expected maximum wave height
Impedance Mismatch	In computer programming, a condition wherein the structure of one language is incompatible with the structure of another
IMR	Inspection, Maintenance, and Repair program
Inheritance	The property by which an object gains all the properties of its ancestors
Instance	A materialization or occurrence referring to a specific entity
Integrity Constraint	A rule or feature of a data schema enforcing the integrity of data
IRR	Internal Annual Rate of Return
L1	Level One
L1(1,1)	Count of safe structures labeled "safe" by Level 1 test
L1(1,2)	Count of unsafe structures labeled "safe" by Level 1 test
L1C1 Evaluation Type	A sample evaluation type designed to perform Level One consequence analysis for single platforms, based on [Aggarwal 1991, Staneff et al 1994]
L1HMax1 Evaluation Type	A sample evaluation type designed to store maximum wave height data for single platforms, based on [Staneff et all 1994]
L1I1 Evaluation Type	A sample evaluation type designed to collect Level One inspection data for single platforms, based on [Bea and Craig 1993, Aggarwal 1991]
L1Risk1 Evaluation Type	A sample evaluation type designed to perform Level One risk analysis for single platforms, based on [Staneff et al 1994]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

L1Risk1_Fleet1 Evaluation Type	A sample evaluation type designed to perform Level One risk analysis for platforms fleets
L1RSR1 Evaluation Type	A sample evaluation type designed to perform Level One structural analysis for single platforms, based on [Bea and Craig 1993]
L1VMax1 Evaluation Type	A sample evaluation type designed to perform Level One maximum wind velocity analysis for single platforms, based on [Staneff et al 1994]
L2	Level Two
L2(1,1)	Count of safe structures labeled "safe" by Level 2 test
L2(1,2)	Count of unsafe structures labeled "safe" by Level 2 test
L2Eng1 Evaluation Type	A sample evaluation type designed to collect engineering assumptions for Level Two structural analyses for single platforms, based on [Mortazavi 1995]
L2I1 Evaluation Type	A sample evaluation type designed to collect Level Two inspection data for single platforms, based on [Mortazavi 1995]
L2RSR1 Evaluation Type	A sample evaluation type designed to perform Level Two structural analyses for single platforms, based on [Mortazavi 1995]
L2RSR2 Evaluation Type	Another sample evaluation type designed to perform Level Two structural analyses for single platforms, based on [Stear 1997]
L3	Level Three
L3(1,1)	Count of safe structures labeled "safe" by Level 3 test
L3(1,2)	Count of unsafe structures labeled "safe" by Level 3 test
L4	Level Four
L4(1,1)	Count of safe structures labeled "safe" by Level 4 test

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

L4(1,2)	Count of unsafe structures labeled "safe" by Level 4 test
Message	In OO databases, a request by one object to use a method of another
Method	In OO databases, the means of retrieving an object's encapsulated data; the method is itself encapsulated in that object
N	Number of structures in a fleet
Network Data Model	A modified hierarchical data model in which an record could be associated with any other
Object	An entity in an object-oriented database that encapsulates the data that is collected about it
Object Class	A classification or group of objects having similar characteristics
Object Instance	An object with data referring to a specific entity; one occurrence or materialization of an object class
Object-Relational Data Model	A still-evolving data model under which one or more fields may be declared as complex domains (such as an object)
OO	Object-oriented
OODBMS	Object-Oriented Database Management System – a DBMS that consists of objects; all data are encapsulated and retrieved only via methods
P Matrix	Decision probability matrix
P1	P(U|L1 "S") (probability that a structure is actually unsafe, given a Level 1 test result of "safe")
P11	P(L1 "U") (probability of a Level 1 test result of "unsafe")
P12	P(L1 "M") (probability of a Level 1 test result of "marginal")

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

P2	P(U|L2 "S") (probability that a structure is actually unsafe, given a Level 2 test result of "safe")
P21	P(L2 "U"|L1 "S") (probability of a Level 2 test result of "unsafe" given a Level 1 test result of "safe")
P22	P(L2 "M"|L1 "S") (probability of a Level 2 test result of "marginal" given a Level 1 test result of "safe")
P23	P(L2 "U"|L1 "M") (probability of a Level 2 test result of "unsafe" given a Level 1 test result of "marginal")
P24	P(L2 "M"|L1 "M") (probability of a Level 2 test result of "marginal" given a Level 1 test result of "marginal")
P25	P(L2 "U"|L1 "U") (probability of a Level 2 test result of "unsafe" given a Level 1 test result of "unsafe")
P26	P(L2 "M"|L1 "U") (probability of a Level 2 test result of "marginal" given a Level 1 test result of "unsafe")
P3	P(U|L3 "S") (probability that a structure is actually unsafe, given a Level 3 test result of "safe")
P31	P(L3 "U"|L2 "S") (probability of a Level 3 test result of "unsafe" given a Level 2 test result of "safe")
P32	P(L3 "M"|L2 "S") (probability of a Level 3 test result of "marginal" given a Level 2 test result of "safe")
P33	P(L3 "U"|L2 "M") (probability of a Level 3 test result of "unsafe" given a Level 2 test result of "marginal")
P34	P(L3 "M"|L2 "M") (probability of a Level 3 test result of "marginal" given a Level 2 test result of "marginal")
P35	P(L3 "U"|L2 "U") (probability of a Level 3 test result of "unsafe" given a Level 2 test result of "unsafe")

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

P36	P(L3 "M"|L2 "U") (probability of a Level 3 test result of "marginal" given a Level 2 test result of "unsafe")
P4	P(U|L4 "S") (probability that a structure is actually unsafe, given a Level 4 test result of "safe")
P41	P(L4 "U"|L3 "S") (probability of a Level 4 test result of "unsafe" given a Level 3 test result of "safe")
P42	P(L4 "U"|L2 "M") (probability of a Level 4 test result of "unsafe" given a Level 3 test result of "marginal")
P43	P(L4 "U"|L2 "U") (probability of a Level 4 test result of "unsafe" given a Level 3 test result of "unsafe")
Pf	Probability of failure
PFIR	Probability of Failure Annual Inflation Rate (the rate at which the probability of failure of a structure might be expected to grow over time, due to wear and tear, fatigue, and other aspects of normal aging)
PIMS	Platform Information Management System (an instance of DAES-SIM)
PlatformData Evaluation Type	A sample evaluation type designed to collect basic data for platforms
Primary Key	A unique attribute or combination of attributes used to specify a unique entity instance (row) in a relation
Query	A view of data in a database
R	Cost of repair (includes all repair costs)
Relation	A table within a relational database (as opposed to a table within a flat file database) consisting of tuples (rows) and attributes (fields or columns)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Relational Database	A data model that represents data using tables having relationships with each other
Relational Object Data Model	A relational model managing attributes rather than sets of attributes, and managing objects possessing exposed (non-encapsulated) data. Example: DAESSIM
Relationship	An integrity constraint between two or more tables or entities
Row Instance	A row in a table with data referring to a specific entity; one occurrence or materialization of a row
RSR	Reserve strength ratio
Safe	Condition wherein a standard of safety is truly met
"Safe"	Test result indicating that a standard of safety has been met
Scenario	An evaluation performed on a fleet; an instance of a scenario type
Scenario Type	A class of scenario
Table	An organization of data consisting of rows and columns
Tuple	A set of data within a relation; analogous to a row within a table
View Or Query	A virtual table constructed by mathematical operations on one or more tables in a relational database
VMax	Expected maximum wind velocity

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A REVIEW OF DATABASE TECHNOLOGY

A.1 FUNDAMENTALS OF RELATIONAL DATABASE MANAGEMENT TECHNOLOGY

A.1.1 Basic Definitions

An overview of database principles naturally involves the use of database terminology. In this appendix, as many terms as possible will be functionally described, so as not to unduly distract from the subject of database design. Four terms, however, must be dealt with immediately:

Data model - a data model is a concept guiding the general means by which data will be managed. It is a combination of three components [Date 1995a]:

1. A collection of data object types, which form the basic building blocks for any database that conforms to the model

2. A collection of general integrity rules, which constrain the set of occurrences of those object types that can legally appear in any such database

3. A collection of operators, which can be applied to such object occurrences for retrieval and other purposes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Data structure - a data structure is a schema, within the confines of a data model, by which data will be managed.

Database Management System (DBMS) - a database management system is a software implementation of a data model that serves as an environment in which a data structure may be implemented. A good DBMS also insulates users from the inner workings of electronic data management.

Database - an implemented data structure within a DBMS, containing or capable of containing data. The database allows data to be used, maintained, controlled and transformed.

A.1.1 Flat File Databases

Database systems are often explained as being similar to computerized rolodexes, where each rolodex is represented as a *table*. The table is made up of *rows* and *columns*; each card in the rolodex corresponds to a row ('record') in the table, and each type of information on the cards (first name, last name, state, etc) corresponds to a column ('field') in the table. This is an example of a *flat file* (Figure A-1), one of the precursors of relational databases. Even such a simple database may offer significant advantages over a spreadsheet: it can be constructed so that only numbers are accepted in the Birth year column, so that only state abbreviations are accepted in the State column, or so that the same information is never entered twice. Such rules are termed *integrity constraints*.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

VICE PRESIDENT

First Name	Last Name	State	Birth
John	Adams	MA	1735
Thomas	Jefferson	VA	1743
Aaron	Burr	NJ	1756
George	Clinton	NY	1739
Edbridge	Gerry	MA	1744

Key:

TABLE NAME

Field Name	Field Name	Field Name
Datum	Datum	Datum
Datum	Datum	Datum
Datum	Datum	Datum
Datum	Datum	Datum
Datum	Datum	Datum

Figure A-1: Flat-File Database

A.1.2 Relational Databases

Given table VICE_PRESIDENT in Figure A-1, one may wish to know the present-day population of each vice president's home state. While this information could be stored in additional columns in the table in Figure A-1, it wouldn't be efficient (see Section A.1.3). A better alternative is shown in Figure A-2:

VICE PRESIDENT

First Name	Last Name	State	Birth
John	Adams	MA	1735
Thomas	Jefferson	VA	1743
Aaron	Burr	NJ	1756
George	Clinton	NY	1739
Edbridge	Gerry	MA	1744

STATE

State	Population
MA	6016425
NJ	7730188
NY	17990455
VA	6187358

Figure A-2: Relational Database

The line between the two tables in Figure A-2 indicates that the STATE table is in a *one-to-many relationship* with the VICE_PRESIDENT table: the State field of the VICE_PRESIDENT table is constrained such that any value entered into it must already

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

exist in the State field of the STATE table; and any entry in the State column of STATE can be used more than once in VICE_PRESIDENT. The two tables together form a *relational database*. When used within a *relational database management system* (RDBMS), means are provided by which to access the data in different combinations. For example, one might choose to see a derived table showing last names, states, and populations, sorted by populations and then by last names. If the RDBMS utilized the industry standard query language SQL ('structured query language'), the instruction to the RDBMS would be

```
SELECT DISTINCTROW VICE_PRESIDENT.State, STATE.Population,
VICE_PRESIDENT.[Last Name]

FROM States INNER JOIN VICE_PRESIDENT ON STATE.State
VICE_PRESIDENT.State

ORDER BY STATE.Population, VICE_PRESIDENT.[Last Name];
```

And the resulting derived table would be (Figure A-3):

RESULT_ONE

Last Name	*State*	*Population*
Adams	MA	6016425
Gerry	MA	6016425
Jefferson	VA	6187358
Burr	NJ	7730188
Clinton	NY	17990455

Figure A-3: Table Derived from Figure A-2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

If any data is changed or added in this derived table (or *query*, or *view*), it is also changed or added in the underlying tables. Also, this query may be used in further query operations: a property called *closure*. However, the query is a direct and dynamic result of the combination of the VICE_PRESIDENT and STATE tables - changing one of them also changes the query results. This is similar to the mathematical sequence "A=2; B=3; C=A+B": changing the value of A from 2 to 10 changes C from 5 to 13. Since relational databases are based on mathematical set theory, each table may be thought of as a variable and each query may be thought of as a result.

Commercial RDBMS's generally use the terms table, row, and field (or column); relational theorists are more likely to use the equivalent terms *relation*, *tuple* (rhymes with 'couple'), and *attribute*.

To use the database shown in Figure A-2 to store President / Vice President combinations, add two more tables:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

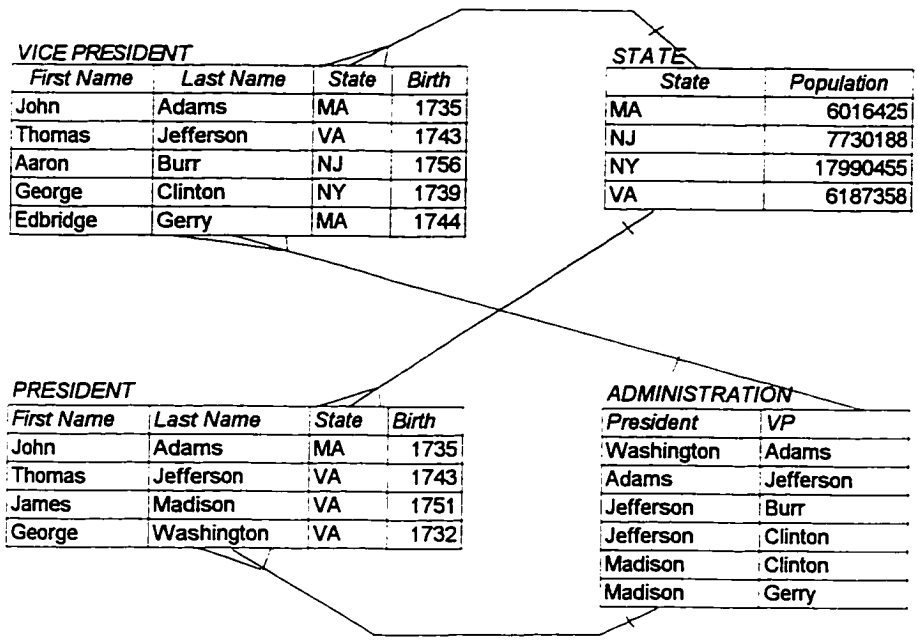


Figure A-4: Many-to-Many Relationship

In Figure A-4, the President field of table ADMINISTRATION is in a many-to-one relationship with the Last_Name field of PRESIDENT. The VP field of ADMINISTRATION is similarly in a many-to-one relationship with the Last_Name field of VICE_PRESIDENT. Thus, ADMINISTRATION allows the entries in PRESIDENT's Last_Name field to be matched many times with many entries from VICE_PRESIDENT's Last_Name field. This is called a *many-to-many relationship* between tables PRESIDENT and VICE_PRESIDENT.

Figure A-5 shows the example relational database in a somewhat more abstract format. Called an *entity-relationship diagram*, this format is intended to bring clarity to the data

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

modeling process by backing away from the table paradigm and dealing with entities, relationships, and attributes. In the format used in Figure A-5 [McFadden and Hoffer 1994], an *entity type* or *entity class* (rectangle) is a collection of things (*entity instances*) that share common characteristics. Here, the entity class PRESIDENT contains things (historical Presidents) that have *attributes* (ovals) in common - each has a First_Name, a Last_Name, and a Birth year. The PRESIDENT entity class also has the attribute State, but this is represented as an attribute of the STATE entity class. PRESIDENT is shown in a From *relationship* (diamond) with STATE; this is a one-to-many relationship (actually, as shown by the one horizontal and 'many' slanted lines at the PRESIDENT end of the link, there may be one or more President entity instances for each State instance; and as shown by the two vertical lines at the State end, there may be one and only one State for each President). The VICE_PRESIDENT entity class is shown in a many-to-many (actually, one-or-many to zero-or-many) relationship with PRESIDENT; each Vice President entity instance must have a corresponding President or Presidents, but each President may have zero, one, or many Vice Presidents.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

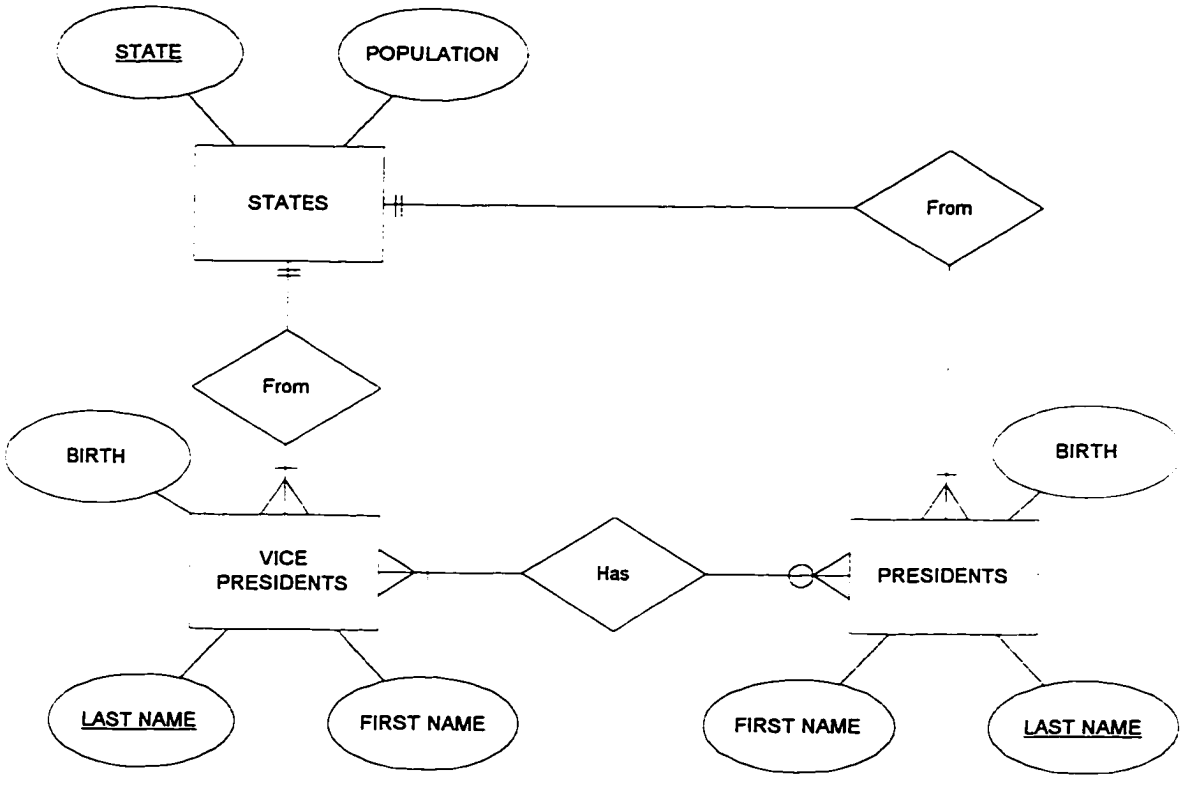


Figure A-5: Entity-Relationship Diagram of Example Database

Figure A-5 shows the complete database of Figure A-2 and A-4 without showing an ADMINISTRATION entity. The ADMINISTRATION table is necessary to implement the many-to-many relationship between PRESIDENT and VICE_PRESIDENT, but is not part of the conceptual design.

Some of the attributes in Figure A-5 have underlined text. This indicates that the attribute is a *primary key* - a unique attribute or combination of attributes whose value can be used to specify a unique entity instance. For example, STATE has the primary key State. As

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

long as no two states have the same name (or, per Figure A-2, two-letter abbreviation), one can discover the other attributes of a State instance by searching for its name in the State field. Similarly, PRESIDENT has the primary key Last_Name. However, if two President instances happen to have the same last name, the second instance will be rejected by the RDBMS. A better solution might be to use both Last_Name and First_Name as primary keys, but this also breaks down (e.g., John vs. John Quincy Adams). The database designer would probably end up using First_Name, Last_Name, and the new attribute Middle_Name as a *combination primary key* for PRESIDENT (as well as for VICE_PRESIDENT).

Any attribute that might serve as a primary key is called a *candidate key*. Any candidate key (and only a candidate key) may serve as a *foreign key*. In an RDBMS implementation, a foreign key helps maintain the relationship with another table by referencing a similar field (not a candidate key) in that other table. In Figure A-2, field State in the VICE_PRESIDENT table references candidate key (and primary key) State in the STATE table. Here, State in the STATE table is a foreign key. This mechanism isn't needed in an entity-relationship diagram, so the State attribute is shown only with the STATE entity class. (Admittedly, the McFadden and Hoffer E-R format might be clarified by the explicit representation of foreign keys. Figure A-6 shows a possible method.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

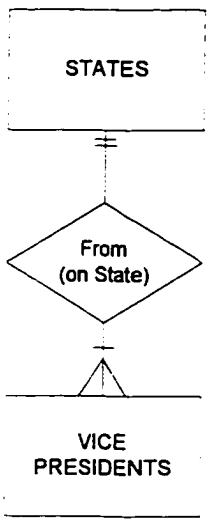


Figure A-6: E-R Foreign Key Declaration (Modified Detail of Figure A-5)

A.1.3 Data Integrity and Normalization

Referring back to Figure A-1 and A-3, if state information had been contained in the same table as vice president information, then the population of the state of Massachusetts would have to have been entered twice (once for each vice president). Any time information is entered twice, an error is possible and data processing inefficiencies can result. There are three broad categories of serious data errors that any database design - regardless of technology - should avoid:

- Insertion anomalies - entering data for one thing or type of thing should not require that data be entered for another thing or type of thing.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Deletion anomalies - deleting data pertaining to one thing or type of thing should not delete data pertaining to another thing or type of thing.

- Modification anomalies - data pertaining to a thing or type of thing should be entered only once.

To avoid these errors, relational tables should have the following properties [McFadden and Hoffer 1994]:

- Entries in columns should be atomic - there can be no multi-valued elements or repeating groups

- Entries in columns should be from the same domain - e.g., social security numbers should not be entered in the same column ('field') as proper names

- Each row should be unique - this is enforced by primary keys, although primary keys are not absolutely required by current standards

- Sequence of columns (e.g., from left to right) should be insignificant

- Sequence of rows (e.g., from top to bottom) should be insignificant

Enforcement of the error-avoidance properties is accomplished by designing relational data models in accordance with *normal form*. As commonly accepted, there are six 'nor-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

mal forms' for relational tables. Relational database designers take great pains to adhere to the standards imposed by the first three normal forms, and often attempt to abide by the others as well. The first three are:

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)

The latter three are:

- Bryce-Codd normal form (BCNF)
- Fourth normal form (4NF)
- Fifth normal form (5NF) [McFadden and Hoffer 1994]

First normal form specifies that there can be no repeating groups in a table. For example, TABLE_1A does not adhere to first normal form, while TABLE_1B does:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

TABLE 1A

Structure ID	Struct Name	Eval ID	Evaluator	Date Completed
1	Alpha	1	Smith	1/1/94
		3	Jones	2/1/94
2	Beta	4	Smith	3/1/94
3	Gamma	5	Chen	1/1/95
		2	Doe	1/2/94

TABLE 1B

Structure ID	Eval ID	Date Completed	Struct Name	Evaluator
1	1	1/1/94	Alpha	Smith
1	3	2/1/94	Alpha	Jones
2	4	3/1/94	Beta	Smith
3	5	1/1/95	Gamma	Chen
3	2	1/2/94	Gamma	Doe

Figure A-7: Example of First Normal Form

The designer of TABLE_1A evidently intended for Structure_ID to be the primary key. The designer of TABLE_1B recognized that Eval_ID contained the truly unique values - each evaluation denoted by Eval_ID was performed on a unique structure on a certain date.

Second normal form specifies that a table be in first normal form and that all non-key attributes be fully (not just partially) dependent on the primary key. If the primary key is a composite of several fields, each non-key field must be fully dependent on the entire primary key, not just some of its constituent fields. Note in TABLE_1B that, if Eval_ID is the primary key, then Struct_Name is dependent not on Eval_ID but on Structure_ID, which is dependent on Eval_ID (or, Structure_ID is dependent on Struct_Name while Struct_Name is dependent on Eval_ID). TABLE_1B should be broken into two, TABLE_2A and TABLE_2B:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

TABLE 2A

Eval ID	Evaluator	Date Completed
1	Smith	1/1/94
3	Jones	2/1/94
4	Smith	3/1/94
5	Chen	1/1/95
2	Doe	1/2/94

TABLE 2B

Structure ID	Struct Name	Eval ID
1	Alpha	1
1	Apha	3
2	Beta	4
3	Gamma	5
3	Gamma	2

Figure A-8: Example of Second Normal Form

If it is desired to see the information of TABLE_2A and TABLE_2B as *presented* in TABLE_1B, a query can be performed that joins TABLE_2A and TABLE_2B on Eval_ID. However, under the conditions of second normal form, the underlying data will be *stored* as TABLE_2A and TABLE_2B.

Third normal form requires that a table be in second normal form and that no transitive dependencies exist. TABLE_2B above gives such an example, where Eval_ID is the primary key - Struct_Name is dependent on Structure_ID (or vice versa). TABLE_2B should be split into TABLE_3A and TABLE_3B:

TABLE 3A

Structure ID	Eval ID
1	1
1	3
2	4
3	5
3	2

TABLE 3B

Structure ID	Struct Name
1	Alpha
2	Beta
3	Gamma

Figure A-9: Example of Third Normal Form

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Again, if it is desired to see the information of TABLE_3A and TABLE_3B as *presented* in TABLE_2B, a query can be performed that joins TABLE_3A and TABLE_3B on Structure_ID. Note that this organization of the data allows errors: for example, TABLE_1B and TABLE_2B might allow different names to be assigned to the same structure.

Bryce-Codd normal form requires that a table be in third normal form and that each determinant (each field which may determine the content of another) be a candidate for use as a primary key. Thus, if a non-key field can be considered as determined by another non-key field (a field not in the primary key that may have repeating information) then the table is not in Bryce-Codd normal form.

Fourth normal form requires that a table be in Bryce-Codd normal form and that there be no multi-valued dependencies - that no field depend on any combination of other non-key fields.

Fifth normal form requires that a table possess no join dependencies. A table with join dependencies cannot be split into two or more tables and then recombined to form the original table.[1]

---

[1] An alternative definition of fifth normal form has been proposed by Date [1995]: "*3NF and all keys simple implies 5NF.* More precisely, if table T (a) is in 3NF and (b) is such that every (candidate) key consists of exactly one column, then T is automatically in 5NF."

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The reader is referred to McFadden and Hoffer [1994] for examples of the latter three types of normal form.

A.1.4 Advantages and Disadvantages of Relational Systems

Advantages of relational systems:

- Based on mathematical theory - operations are consistent and predictable.

- Primarily non-proprietary - SQL is an industry standard, though its implementation varies by product; table structures are generally proprietary but translation protocols are common.

- Data integrity - data integrity is provided through the design of the components (normalization), as well as provisions for functions that automatically constrain new data (e.g., a field may be defined with an integrity constraint such that all values must be greater than zero - thus, no zero or less than zero values may ever be entered in that field).

- *Ad hoc* query capability - ensured by the theoretical foundation of the relational model.

- Referential integrity - a form of data integrity, referential integrity means that if entity instance A refers to and depends on entity instance B, and B is deleted, then the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RDBMS will automatically delete A (and all other entity instances that are defined as depending on B).

- Closure - the output of an operation can serve as the input of another. Though inherent in the full relational model, closure has not yet been fully implemented in commercial systems.

- Declarative support - once a rule has been declared (such as a business rule or an integrity constraint), the RDBMS will automatically enforce that rule without requiring user or programmatic intervention. In contrast, object-oriented database systems provide only procedural support, in which rules are enforced only through the continuous execution of programs (which must check for all possible errors, all of which must have been thoroughly explored in advance).

- Transactions (and performance in general) - transactions (insert, delete, or change operations) are very quick due to relational models' atomic data, set theory foundations, and declarative support.

Disadvantages of relational systems:

- Lack of support for user-defined and abstract data types (drawings, images, video, etc) - even when supported, such data types (domains) are proprietary. Since data types in current RDBMS's are restricted, the management of complex data types is awkward.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Date [1995a] gives the example of a database that stores rectangles by storing their coordinates (i.e., (x1, y1), (x2, y2)) in separate columns - the task of determining which rectangles overlap a given area is complex. This is an issue of performance optimization, not of capability. (A *data type* restricts the kind of information allowed in a field. For instance, a field defined as 'text 50' accepts only text that's 50 characters long or less. A *domain* in the present context is a set of allowable values, all taken from the same type, upon which a field may be defined. A field may be defined on a currency domain, where that domain is defined as all numbers of the double-precision type having only two decimal places; or a field may be defined directly on the double-precision type.)

- Non-intuitive representation - the real world consists of objects which interact with, contain, or are contained by other objects. In contrast, a computer stores data in long series of bits. A database system needs to present a user interface that somehow translates between the two. The relational model biases the translation towards the computer's viewpoint.

- Lack of flexibility - adding new attributes or entity classes (tables) requires users to modify the basic structure of the database: not just adding a new column or table, but altering integrity rules and relationships to suit the new schema. (However, Date [1995a] argues that relational data models are easier to change than object-oriented data models.)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Business rule support - an aspect of #2, business rules (more generally, *behavioral knowledge* or *domain-specific generic knowledge* [Dogac et al 1994] capture the ways in which objects interact. The relational model's support for business rule specification is awkward.

- Impedance mismatch - relational query languages such as SQL manipulate programmatic objects a set at a time. Application programming languages do it a record at a time (at best). Since SQL and other query languages are inadequate for application support, both types of languages are usually required in an RDBMS. The conflict between set-at-a-time and record-at-a-time variables is called *impedance mismatch*. Impedance mismatch becomes particularly critical when attempting to run applications - since messaging facilities (discussed in Section A.2) do not exist in relational databases, values must be passed back and forth between SQL sub-routines and application languages.

A.2 FUNDAMENTALS OF OBJECT-ORIENTED DATABASE MANAGEMENT TECHNOLOGY

Object-oriented database technology, an alternative approach to relational technology, has its roots in the simulation application languages of the late 1960's and early 1970's [Date 1995a, p. 649]. Development work began in earnest in the 1980's [Goldberg and Robson, 1983] with the Smalltalk language, and built upon advances in the related (but far from

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

identical) field of object-oriented programming. The technology now receives much theoretical and practical attention, and several database management products are on the market.

A.2.1 Object-Oriented Terminology

Suppose a computer user has a personal computer (PC) using an operating system capable of running several application programs simultaneously (or nearly so). On this computer are two applications (other than the operating system), a spreadsheet program (e.g., Lotus 1-2-3 or Quattro Pro) and a symbolic logic program (e.g., Mathematica or Maple). Both application programs contain (or have access to) many files. Some of these files are templates - files that serve as models for other files but are themselves left unchanged - and some of these files execute instructions (subroutines) and/or contain data. Using a communications protocol such as OLE (object linking and embedding, an increasingly common operating system feature), the two application programs can send messages to each other - if the spreadsheet needs a certain computation performed by the symbolic logic (s.l.) program, it can start the s.l. program, open a file, specify a subroutine, supply it with data, run the subroutine, accept the results, and close the s.l. program. Similarly, the s.l. program can use the spreadsheet as, for instance, a charting engine. Neither of the application programs, nor any of their files and subroutines, have direct access to data stored by the other application - data must be 'requested' from the other application (on this hypothetical computer, operating system functions like DDE and 'poke' do not exist), usu-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ally programmatically. There is no 'manager' monitoring operations - without intimate knowledge of the computer and diligence in documenting its operations, the user may become overwhelmed with files and data. (Note that this is the state of much of the engineering world today - applications with application-specific data residing on an engineer's computer (or scattered across many such computers) - and is one of the major problems leading to this research.)

This hypothetical computer's operating system may be considered, very loosely, an 'object-oriented database.' Each of the application programs is an *object* (everything in the universe is an object), and each is an *object class*. Each of the templates (explicit or implicit) is derived from its parent object and is thus an object class (each is also an object). Each file is derived from its parent template (object) and is also an object class when used to operate on data transiently, or is an *object instance* when used to permanently contain data or results (regardless of whether it's an object class or an object instance, a file is still an object). Each derived file *inherits* the features of its parent template. Each such file has a name, its *object ID*. Each input datum is known as an *attribute*. Each output datum is known as a *method*. Data are fully stored within their files (object instances) and cannot be accessed externally, and are thus *encapsulated*. The communications between objects are known as *messages*. The structural progression from upper-most object class to bottom-most object instance is known as a *class hierarchy* (although this is an artificial structure, imposed and maintained via software pointers - physical storage mechanisms dictate that objects be stored as linear files). (For those familiar with programming con-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ventions, Date [1995a] has compared OO (object-oriented) terminology to that of its programming roots, and lists the following comparisons: [OO term ↔ programming term]: object ↔ variable (if mutable), or value (if immutable); object class ↔ data type; method ↔ function; message ↔ call.)

Unlike the relational model, there is no standard for an object-oriented data model [Date 1995a, Zdonik 1994] - rather, there are a number of proprietary models. Commonly, however, an object-oriented database management system implementation mirrors the PC example given above, but with significant constraints mandated by the needs of a data model (section A.1.1). Among these: rather than rely on an all-purpose operating system, an OODBMS is itself an application program and establishes a dedicated environment; object classes and object instances within the OODBMS are essentially small pieces of application code that adhere to proprietary standards to ensure intra-product compatibility; the messaging system is provided by the OODBMS rather than by the operating system.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Objects are commonly represented as in Figure A-10 below [McFadden and Hoffer 1994]:

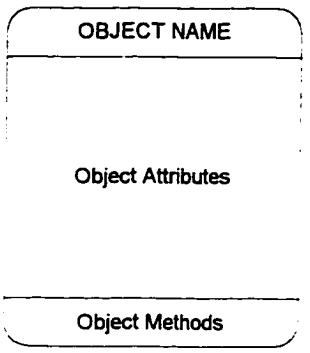


Figure A-10: Object Representation

In attempting to translate between the entities of the real world and the bits and bytes of the computer, object-oriented system user interfaces are biased toward real-world representations. Nevertheless, the object (as shown in Figure A-10) is still stored as a linear file. The representational abstraction is seen as a benefit by some observers and a detriment by others.

Inheritance is illustrated in Figure A-11. The object class INSPECTION has two subclasses, PRODUCTION and STRUCTURAL. Because of inheritance, all properties (attributes and methods) of a class become the properties of its subclasses. Each subclass also has whatever properties that have been uniquely assigned to it. Here, all production inspections have the Inspection ID, Date Performed, Daily Production Natural Gas, and Daily Production Crude attributes, as well as the Calculate Inspection Age and Calculate

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Yearly Crude methods. All structural inspections have their unique Height, Length, and Material Strength attributes, and the Calculate Structural Strength method, but they also have the Inspection ID, Date Performed, and Calculate Inspection Age properties. Inheritance is also a characteristic of strong user-defined data types (or domains) in relational systems.

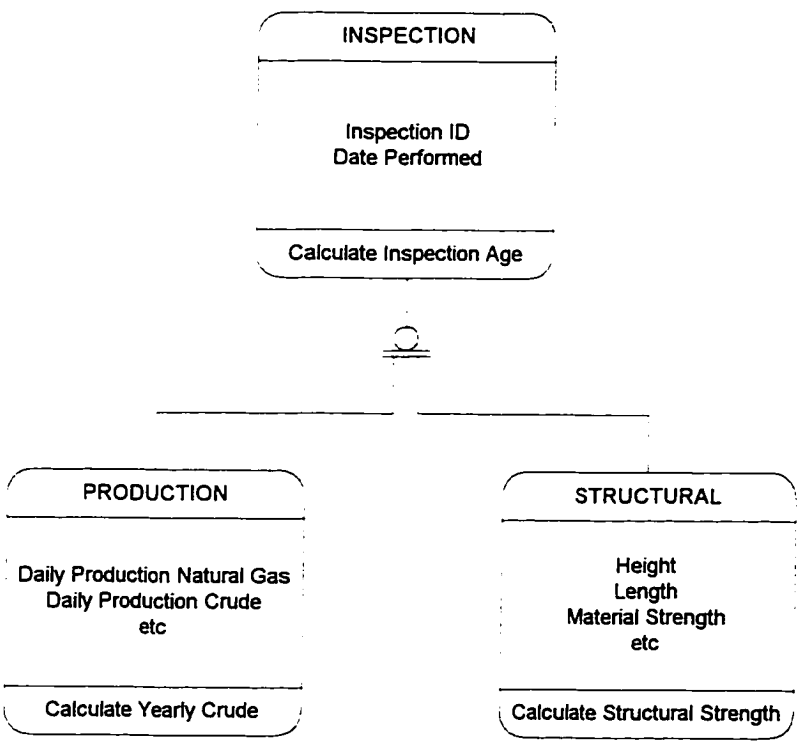


Figure A-11: Inheritance

In Figure A-12, the STRUCTURE object is in a one-to-many (actually, "one and only one" to "one or many") relationship with the INSPECTION object. The user has sent a message (#1) to the STRUCTURE object, asking it to execute its Yearly Crude Produc-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

tion method ("what is the total yearly production of crude for all structures?"). The STRUCTURE object contains more than one instance; each of these instances represents an individual structure. Each of these instances sends a message (#2) to the PRODUCTION object, asking its instances to respond with the results of their Calculate Yearly Crude methods (messages #3). The STRUCTURE object then responds to the user with the yearly production figure (message #4). Note that the user cannot ask, "what was the yearly production of natural gas?" or "what is the combined height of all structures?" (or other *ad hoc* queries), because the system designer has not included methods for these purposes and because all data are contained within objects (encapsulated). The lack of *ad hoc* query ability is a serious disadvantage of OO designs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

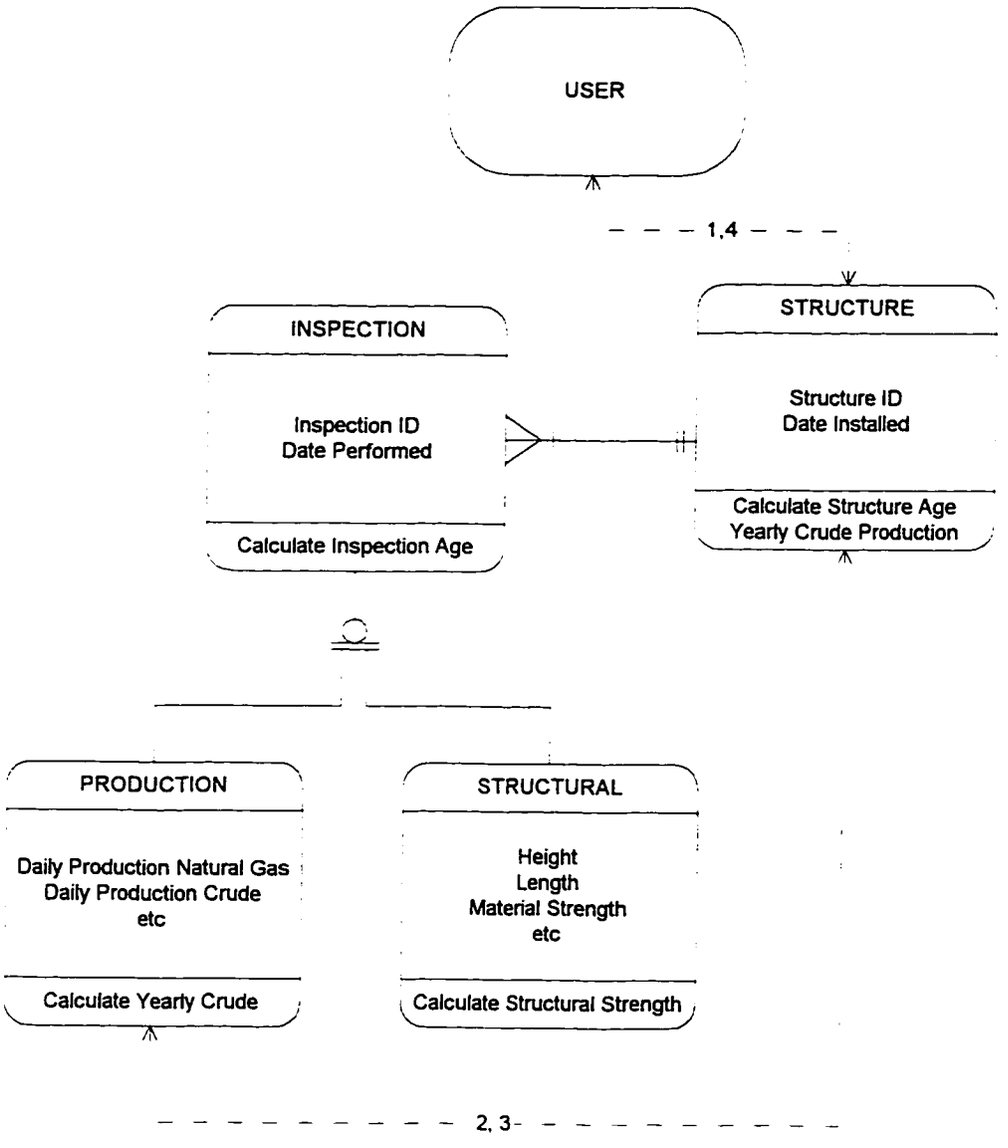


Figure A-12: Messaging

Despite claims to the contrary [Stein and Maier 1988, Choi 1992], or complete non-consideration of the matter [Dogac et al 1994], normalization is essential to a properly designed object-oriented database (or any type of database) [Date 1995a]. The tendency

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of object-oriented models toward nested or hierarchical systems leads the unaware into a structure that requires entering the same data many times - a root cause of data quality instability. Normalization methods, however, are not readily available for OO systems.

A.2.1 Advantages and Disadvantages of Object-Oriented Systems

Advantages of Object-Oriented Data Models:

- Support for custom and complex data types - but this can be enabled in relational systems through user-defined data types (domains).

- Inheritance - again, this can be enabled in relational systems through user-defined data types.

- Interoperability - since object classes are essentially units of program code, the potential exists for easy integration of OO systems into external applications.

- Object ID's - each object is uniquely, externally identified.

- One language for data manipulation and programming - so, only one language (albeit proprietary) to learn.

- Extensibility - subclasses may easily be added (by a qualified database administrator). For example, in Figure A-11, if the STRUCTURAL subclass had not existed in the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

original design, it could easily have been added later. Such an addition to a relational system would require more work and forethought.

- Joins - high orders of joins may be possible ([Choi 1992] cites 10-way joins; but Date [1995a] states that OODBMS's do not support joins, and that their equivalent 'path' expressions do not create new objects - a violation of the desirable closure property). (In a relational context, a "*join* operation joins together two tables on the basis of common values in a common column" [Date 1995a].)

Disadvantages of Object-Oriented Data Models:

- Proprietary systems - there is no standard data model for object-oriented databases, nor for exchanging data.

- No underlying mathematical foundation [Beeri 1994, Hillebrand et al 1994, Date 1995a, Choi 1992] - OO structures are not logic-based, but symbolic representations of a world-view.

- No support for views (queries) - due to the lack of a mathematical foundation.

- No support for closure - due to the lack of a mathematical foundation.

- Lengthy transactions - objects are 'checked out' while transactions are in progress. Thus some transactions (e.g., recalculation of CAD drawing objects) may take a very

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

long time. This is more a criticism of large and complex data types than it is of OO systems in particular.

- Object ID's - despite the existence of external object ID's, objects involved in relations still require internal keys [Date 1995a]. This often means that identification is provided twice for the same set of data, a violation of basic design principles.

- Encapsulation - *ad hoc* queries are impossible, and unforeseen requirements thus remain unfulfilled.

- Representational abstraction - data is apparently organized similarly to real-world objects (often cited as an advantage). However, the translation to files and bytes/bits may result in normalization errors or other modeling errors.

- One language for data manipulation and programming - relational systems use Structured Query Language (SQL) for data access and a higher-level language (e.g., C, C++, Basic, etc) for programming tasks. This leads to *impedance mismatch* - SQL handles data in sets, while higher-level languages handle data in single variables. The object-oriented answer to this is to use one language, but unfortunately the language chosen usually cannot handle sets at all [Date 1995a]. If one language is to be chosen, it should be one that handles and transfers not only sets but arrays.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Data integrity - integrity rules are potentially violated by built-in methods. For example, if new INSPECTION instances in Figure A-11 are required to be dated after 1/1/97, a method must be defined incorporating this rule to each new instance. But an 'Add New' method is built into every object by most OODBMS's, and does not contain this rule.

A.3 OTHER DATABASE TECHNOLOGIES

Other data models exist. *Hierarchical* models and *network* models were both precursors to relational models. Hierarchical models arranged data in a top-down hierarchical progression (tree structure), with records nested inside of other records. Network models loosened up the hierarchical model so that any record could be associated with any other record. Both models proved difficult to manage. (Note that the subject of this dissertation bears no relationship to IBM's hierarchical IMS product (see [McFadden and Hoffer 1994].)

Object-relational database management systems (ORDBMS's) are starting to appear. These attempt to meld the advantages of object-oriented systems with those of relational systems. Since the advantages of each are in dispute, the resulting systems may have differing approaches. The SQL3 standard addressing the matter is still being written [Beech 1997]. Date [1995a] advocates simply enabling user-defined data types (domains) within relational systems, and claims that a sufficiently defined data type essentially is an object.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Examples of systems arriving on the market, which may or may not follow either the SQL3 or the Date approach, are IBM's DB2 Universal Database [Davis 1996] and Informix's Universal Server / DataBlades [Informix 1997; Keeler 1996]. For the time being, object-relational systems may be considered as relational systems in which one or more fields may be declared as complex domains. Since these domains may encapsulate data, object-relational systems risk violating the atomic data properties of relational systems.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

B DAESSIM ENTITY-RELATIONSHIP DIAGRAM

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

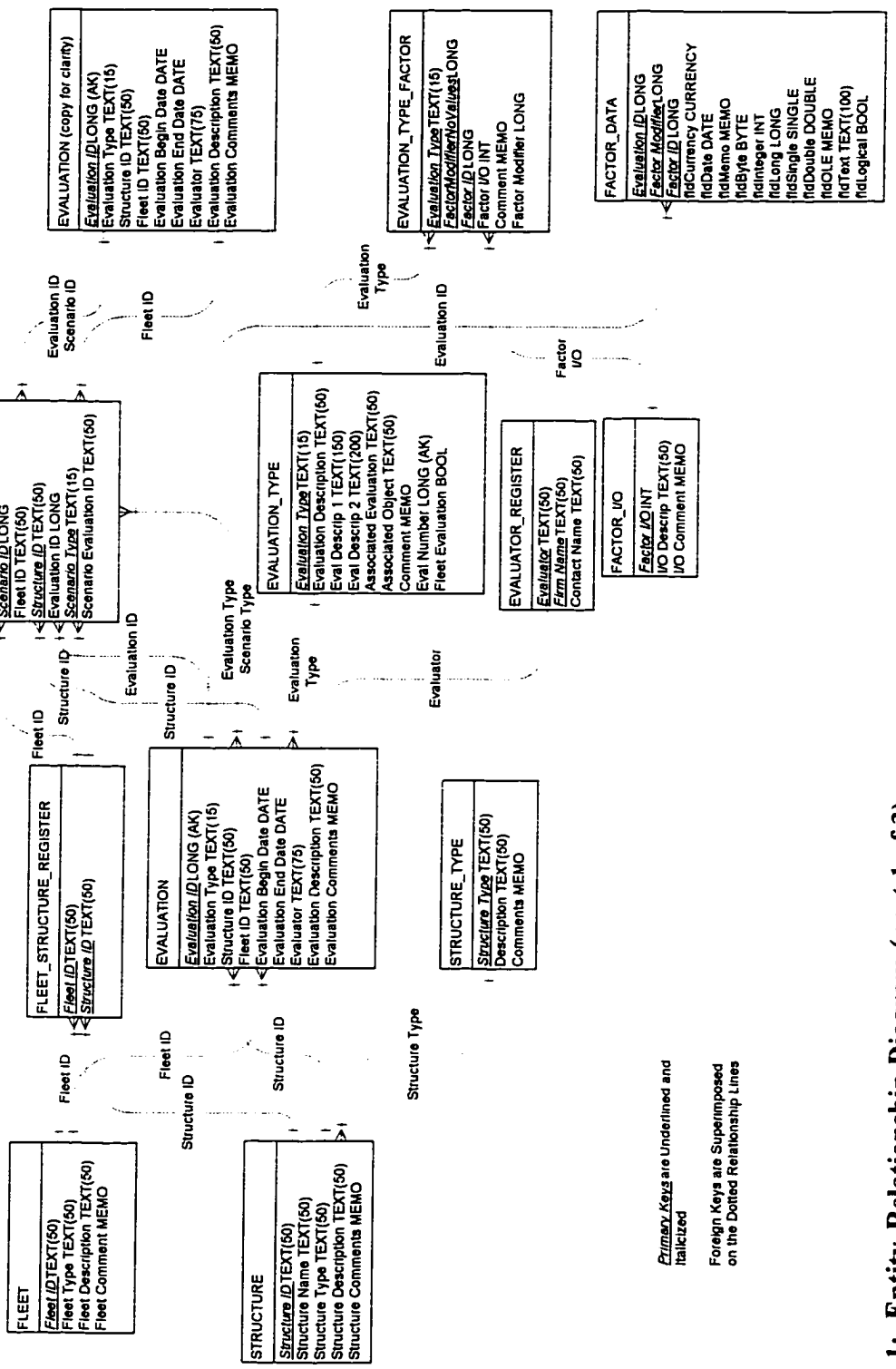


Figure B-1: Entity-Relationship Diagram (part 1 of 3)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

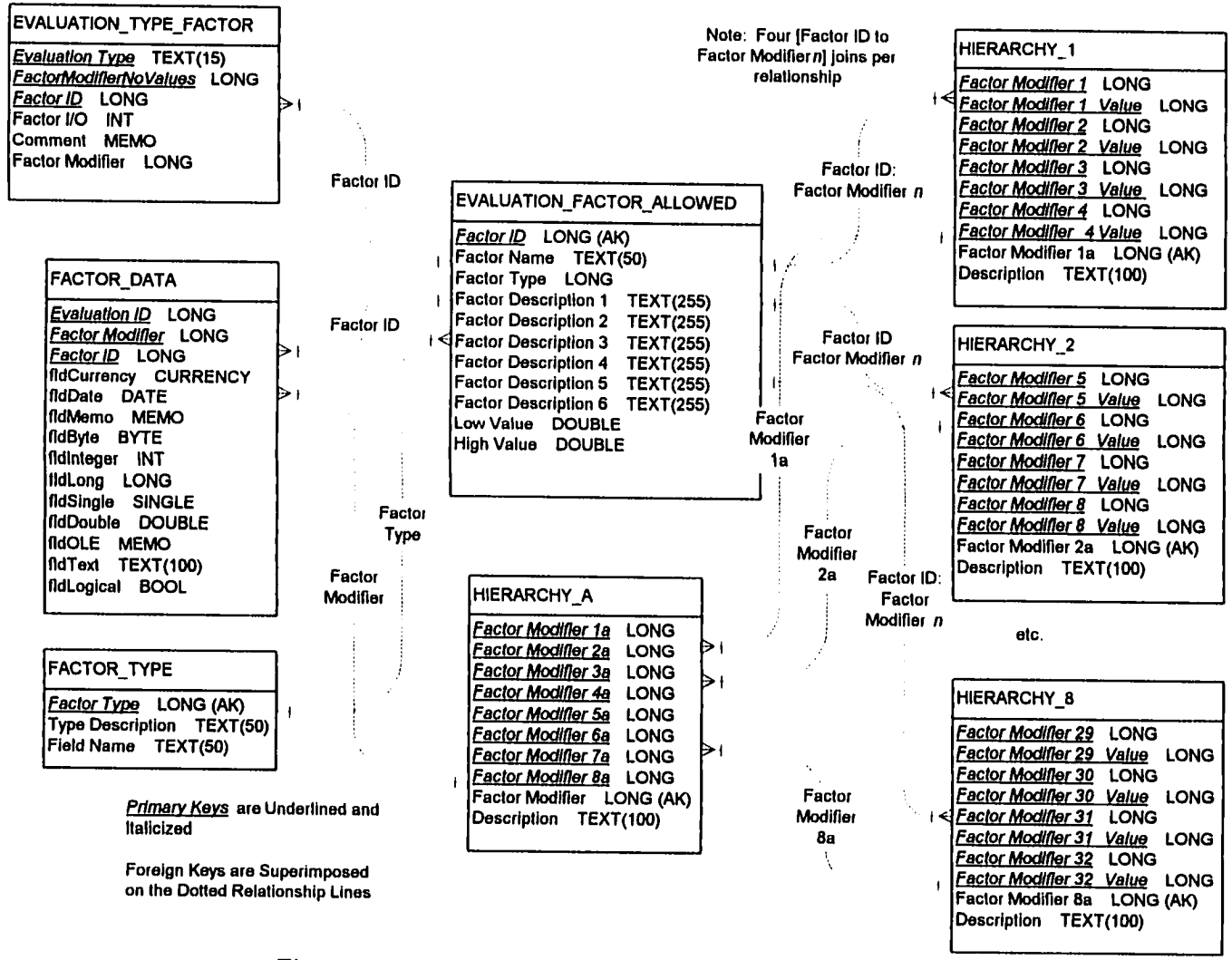


Figure B-2: Entity-Relationship Diagram (part 2 of 3)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

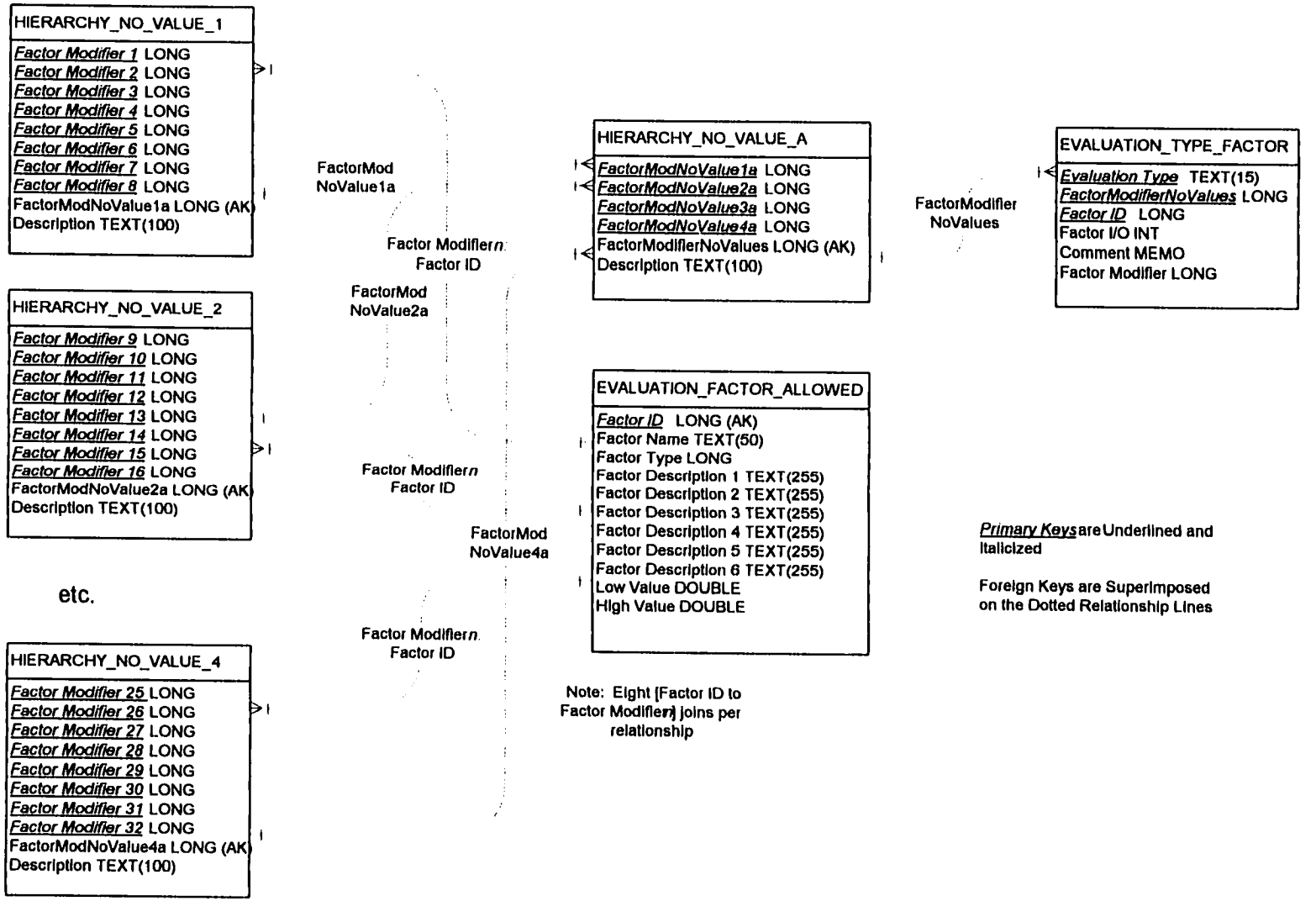


Figure B-3: Entity-Relationship Diagram (part 3 of 3)

C DAESSIM TABLE DEFINITIONS

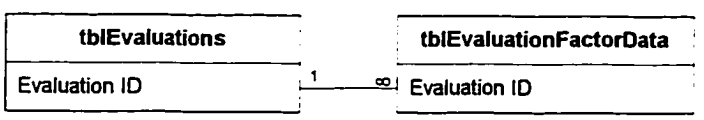
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Evaluation ID	Number (Long)	4
Factor Modifier	Number (Long)	4
Factor ID	Number (Long)	4
fldCurrency	Currency	8
fldDate	Date/Time	8
fldMemo	Memo	-
fldByte	Number (Byte)	1
fldInteger	Number (Integer)	2
fldLong	Number (Long)	4
fldSingle	Number (Single)	4
fldDouble	Number (Double)	8
fldOLE	OLE Object	-
fldText	Text	100
fldLogical	Yes/No	1

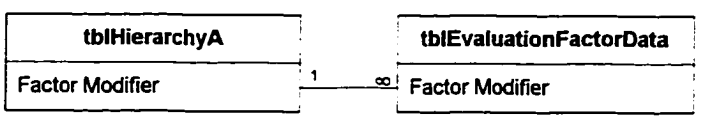
Relationships

Reference1



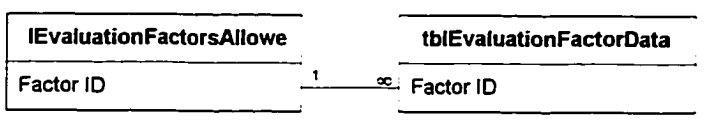
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference28



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference30



Attributes: One to Many, Enforced

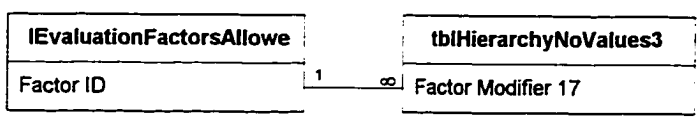
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor ID	Number (Long)	4
Factor Name	Text	50
Factor Type	Number (Long)	4
Factor Description 1	Text	255
Factor Description 2	Text	255
Factor Description 3	Text	255
Factor Description 4	Text	255
Factor Description 5	Text	255
Factor Description 6	Text	255
Low Value	Number (Double)	8
High Value	Number (Double)	8

Relationships

Reference100

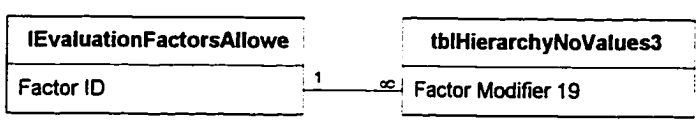


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference101

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

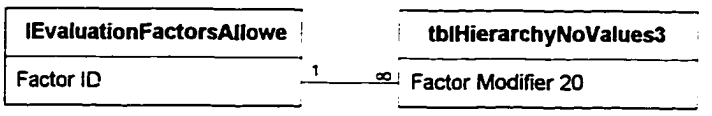
Reference102



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

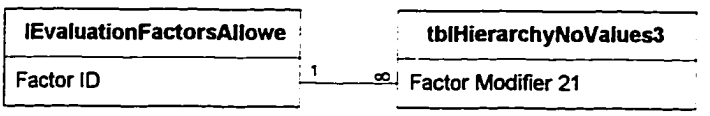
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference103



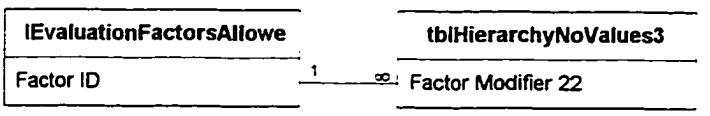
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference104



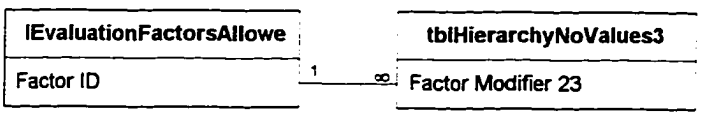
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference105



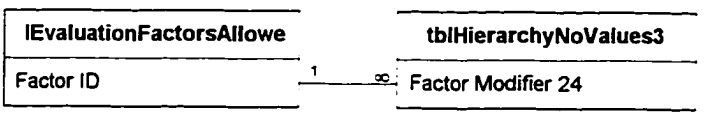
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference106



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

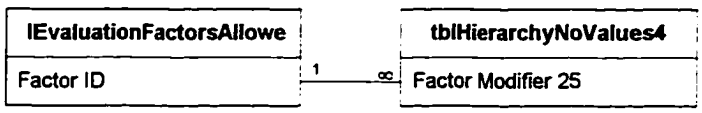
Reference107



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference108

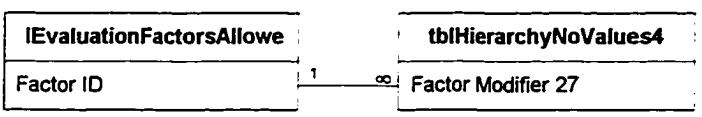


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference110

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference111

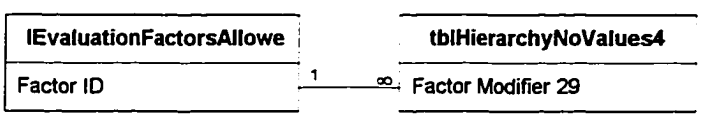


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference112

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

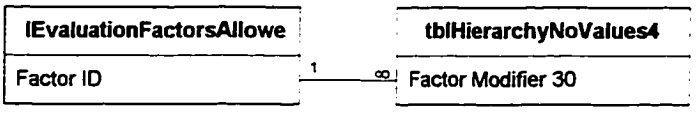
Reference113



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

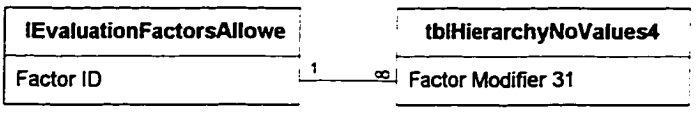
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference114



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference115

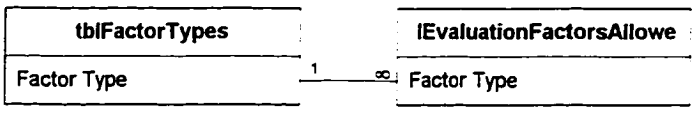


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference116

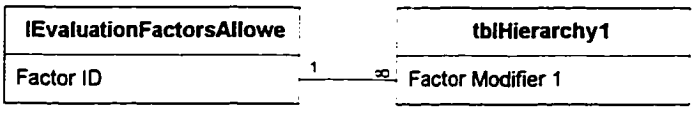
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference26



Attributes: One to Many, Enforced

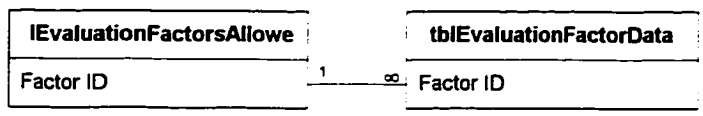
Reference29



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference30

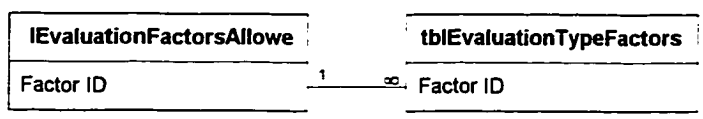


Attributes: One to Many, Enforced

Reference31

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference33

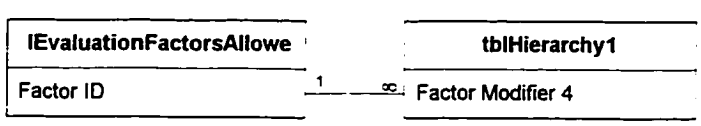


Attributes: One to Many, Enforced

Reference37

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

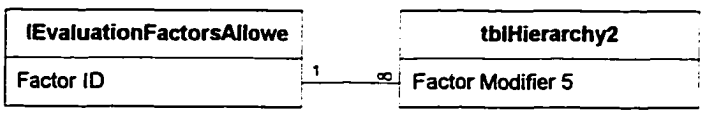
Reference38



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference40

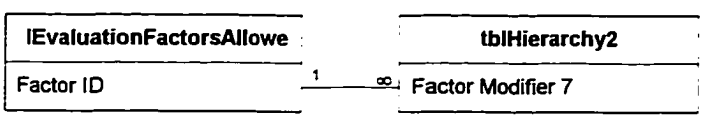


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference42

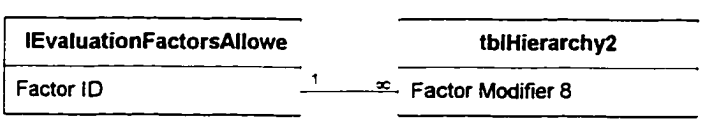
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference44



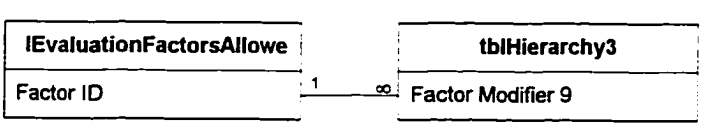
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference46



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

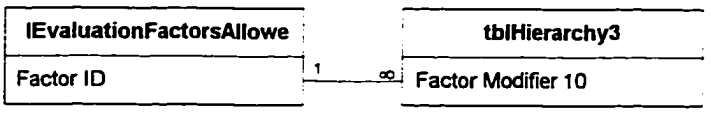
Reference48



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

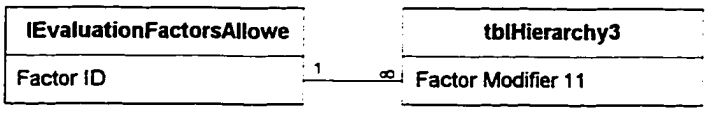
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference50



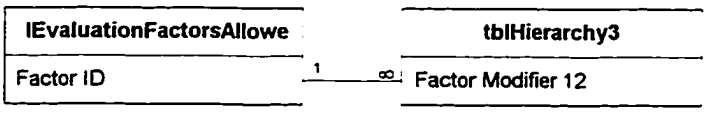
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference51



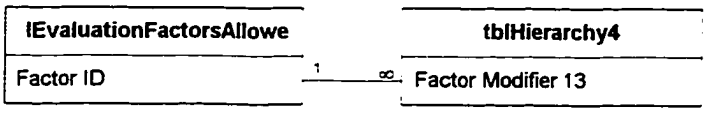
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference52



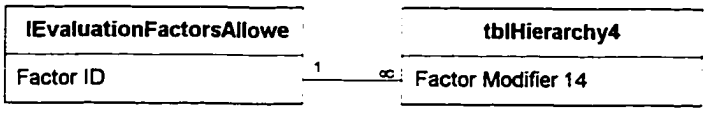
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference53



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

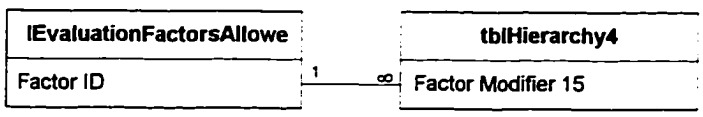
Reference54



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

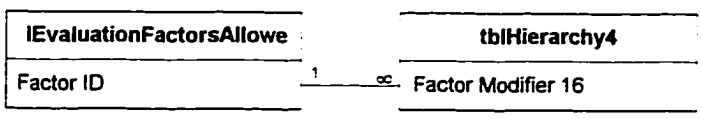
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference55



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference59

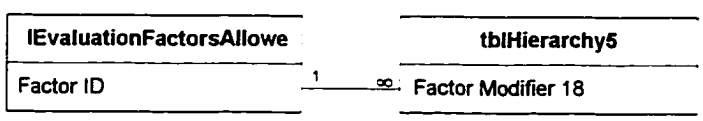


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference61

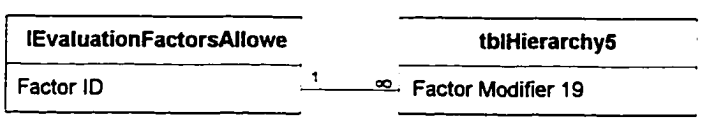
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference62



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

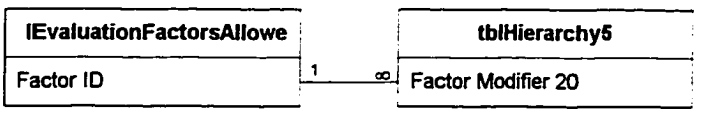
Reference63



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference64



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference65

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

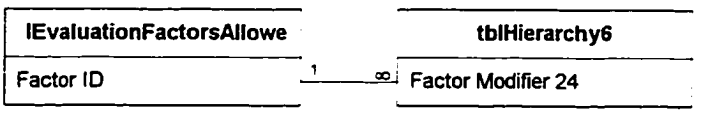
Reference66

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference67

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference68



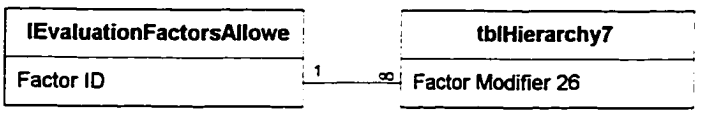
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference69

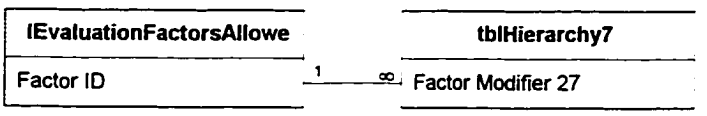
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference70



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference72

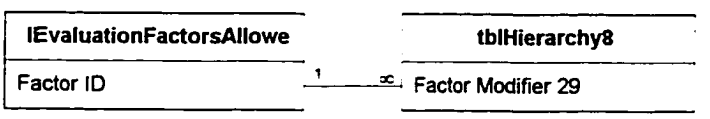


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference73

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

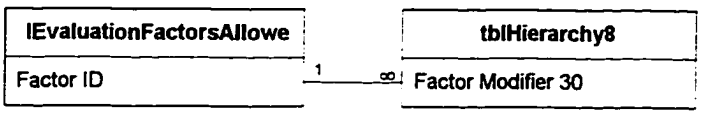
Reference75



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

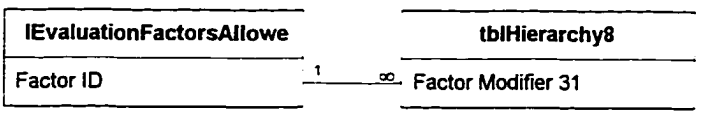
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference76



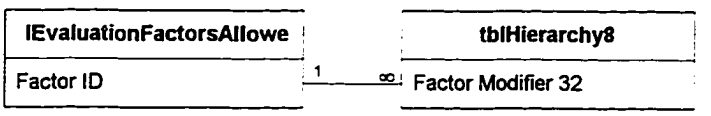
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference77



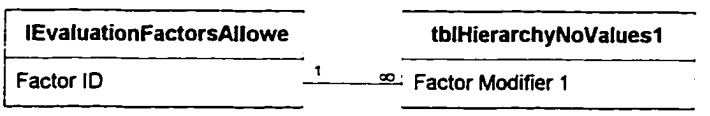
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference78



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference80



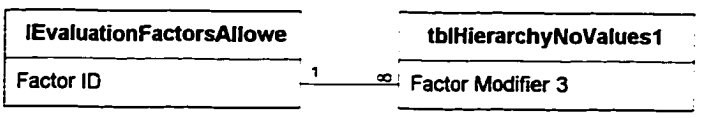
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference81

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

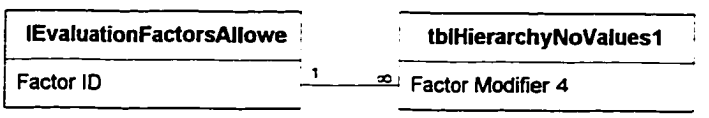
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference82



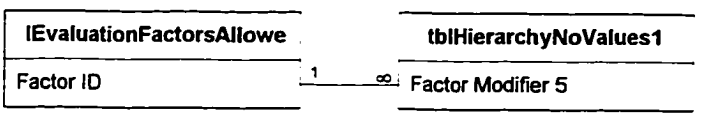
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference83



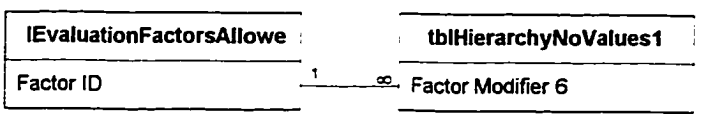
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference85



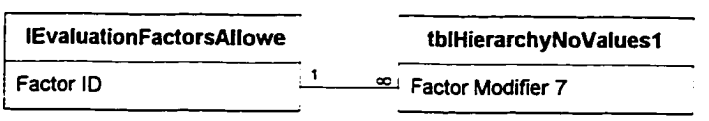
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference86



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

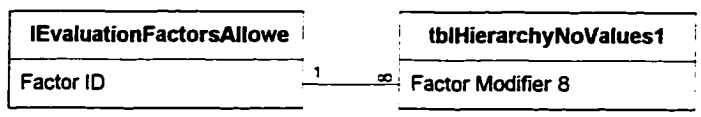
Reference87



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

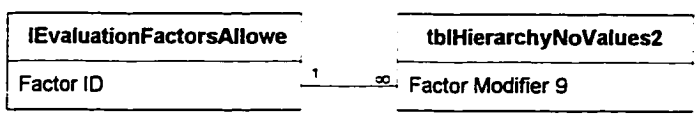
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference88



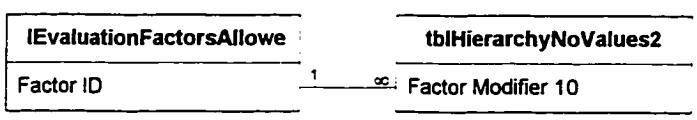
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference90



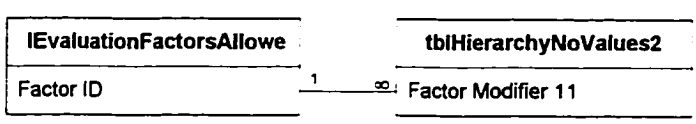
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference91



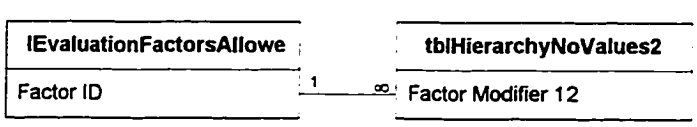
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference92



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference93



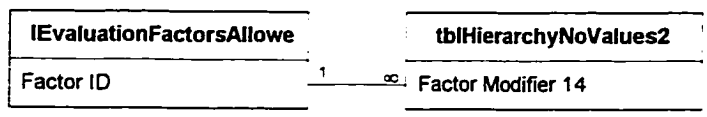
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference95

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference96

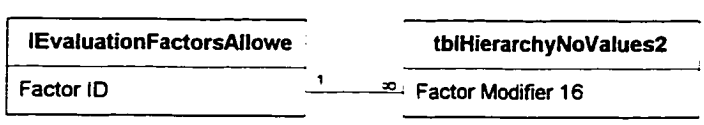


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference97

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference98



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

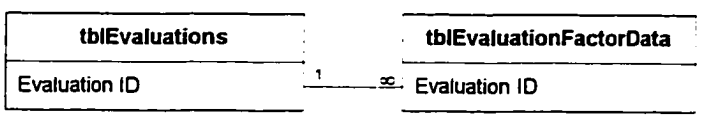
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Evaluation ID	Number (Long)	4
Evaluation Type	Text	15
Structure ID	Text	50
Fleet ID	Text	50
Evaluation Begin Date	Date/Time	8
Evaluation End Date	Date/Time	8
Evaluator	Text	75
Evaluation Description	Text	50
Evaluation Comments	Memo	-

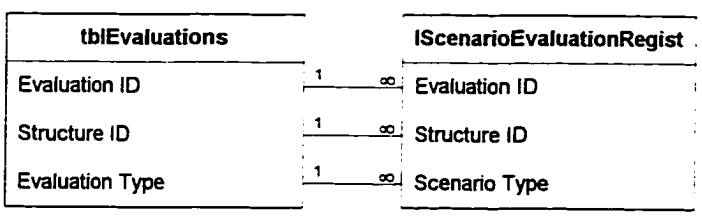
Relationships

Reference1



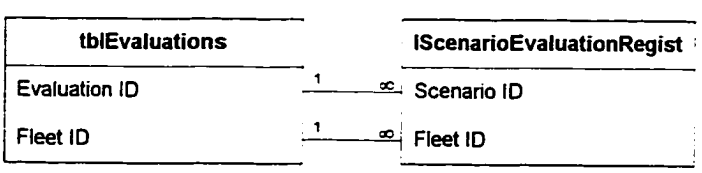
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference21



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

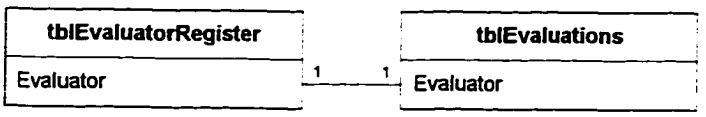
Reference3



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

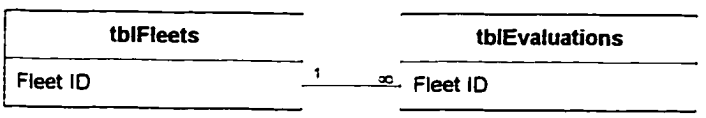
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference35



Attributes: One to One, Not Enforced

Reference6

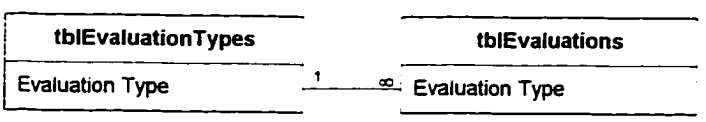


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference8

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference9



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Evaluation Type	Text	15
FactorModifierNoValues	Number (Long)	4
Factor ID	Number (Long)	4
Factor I/O	Number (Integer)	2
Comment	Memo	-
Factor Modifier	Number (Long)	4

Relationships

Reference

Attributes: One to Many, Enforced

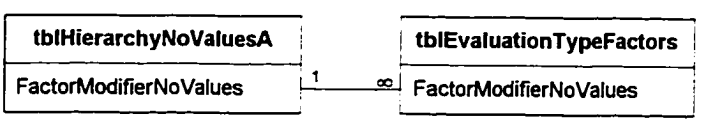
Reference2

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference33

Attributes: One to Many, Enforced

Reference60



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

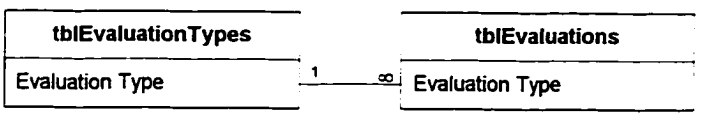
Name	Type	Size
Evaluation Type	Text	15
Evaluation Description	Text	50
Eval Descrip 1	Text	150
Eval Descrip 2	Text	200
Associated Evaluation	Text	50
Associated Object	Text	50
Comment	Memo	-
Eval Number	Number (Long)	4
Fleet Evaluation	Yes/No	1

Relationships

Reference2

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference9



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

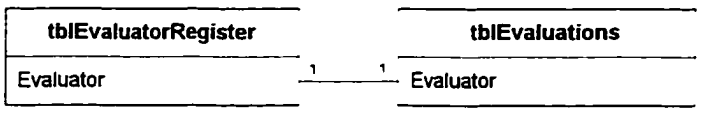
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Evaluator	Text	50
Firm Name	Text	50
Contact Name	Text	50

Relationships

Reference35



Attributes: One to One, Not Enforced

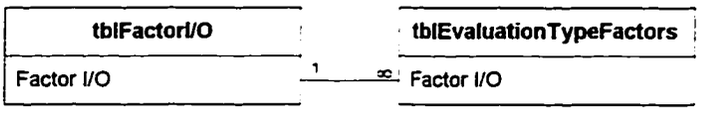
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor I/O	Number (Integer)	2
I/O Descrip	Text	50
I/O Comment	Memo	-

Relationships

Reference



Attributes: One to Many, Enforced

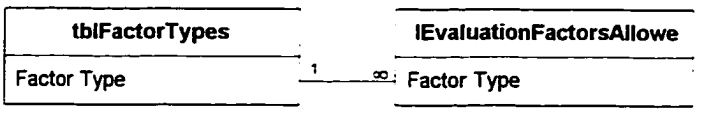
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Type	Number (Long)	4
Type Description	Text	50
Field Name	Text	50

Relationships

Reference26



Attributes: One to Many, Enforced

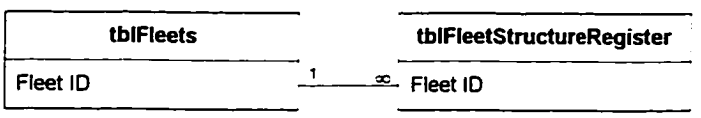
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Fleet ID	Text	50
Fleet Type	Text	50
Fleet Description	Text	50
Fleet Comment	Memo	-

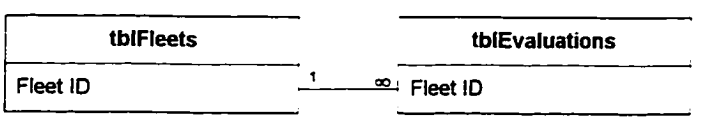
Relationships

Reference4



Attributes: One to Many, Enforced

Reference6



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

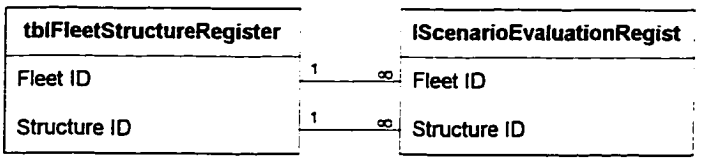
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Fleet ID	Text	50
Structure ID	Text	50

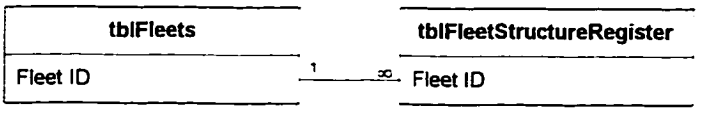
Relationships

Reference22



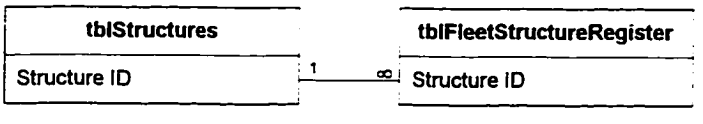
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference4



Attributes: One to Many, Enforced

Reference5



Attributes: One to Many, Enforced

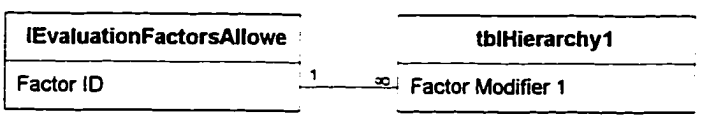
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 1	Number (Long)	4
Factor Modifier 1 Value	Number (Long)	4
Factor Modifier 2	Number (Long)	4
Factor Modifier 2 Value	Number (Long)	4
Factor Modifier 3	Number (Long)	4
Factor Modifier 3 Value	Number (Long)	4
Factor Modifier 4	Number (Long)	4
Factor Modifier 4 Value	Number (Long)	4
Factor Modifier 1a	Number (Long)	4
Description	Text	100

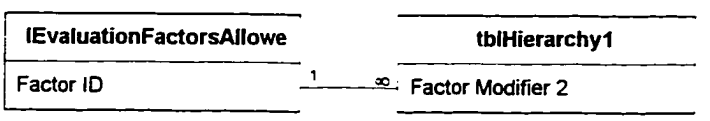
Relationships

Reference29



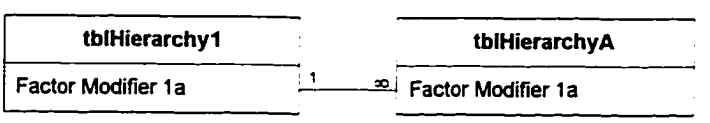
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference31



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference32



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference37

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference38

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

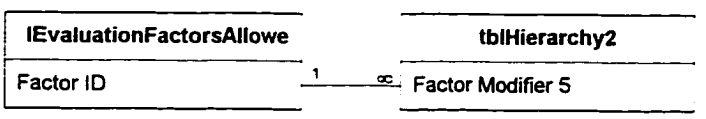
Name	Type	Size
Factor Modifier 5	Number (Long)	4
Factor Modifier 5 Value	Number (Long)	4
Factor Modifier 6	Number (Long)	4
Factor Modifier 6 Value	Number (Long)	4
Factor Modifier 7	Number (Long)	4
Factor Modifier 7 Value	Number (Long)	4
Factor Modifier 8	Number (Long)	4
Factor Modifier 8 Value	Number (Long)	4
Factor Modifier 2a	Number (Long)	4
Description	Text	100

Relationships

Reference39

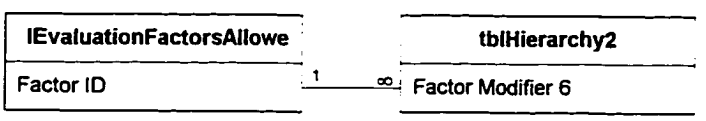
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference40



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference42



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference44

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference46

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

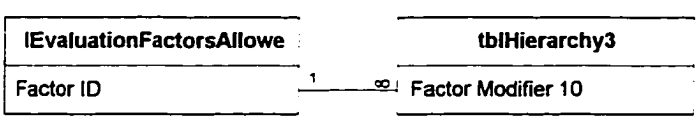
Name	Type	Size
Factor Modifier 9	Number (Long)	4
Factor Modifier 9 Value	Number (Long)	4
Factor Modifier 10	Number (Long)	4
Factor Modifier 10 Value	Number (Long)	4
Factor Modifier 11	Number (Long)	4
Factor Modifier 11 Value	Number (Long)	4
Factor Modifier 12	Number (Long)	4
Factor Modifier 12 Value	Number (Long)	4
Factor Modifier 3a	Number (Long)	4
Description	Text	100

Relationships

Reference48

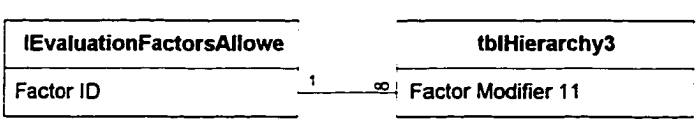
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference50



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

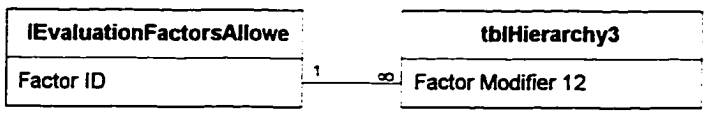
Reference51



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

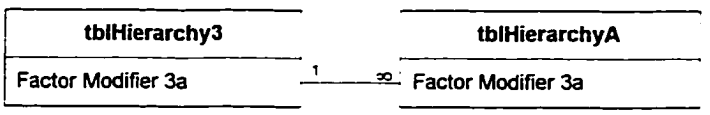
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference52



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference74



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

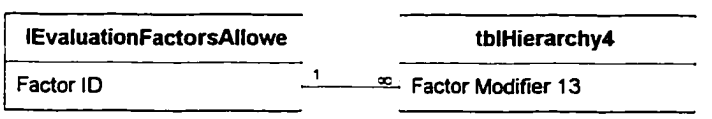
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 13	Number (Long)	4
Factor Modifier 13 Value	Number (Long)	4
Factor Modifier 14	Number (Long)	4
Factor Modifier 14 Value	Number (Long)	4
Factor Modifier 15	Number (Long)	4
Factor Modifier 15 Value	Number (Long)	4
Factor Modifier 16	Number (Long)	4
Factor Modifier 16 Value	Number (Long)	4
Factor Modifier 4a	Number (Long)	4
Description	Text	100

Relationships

Reference53

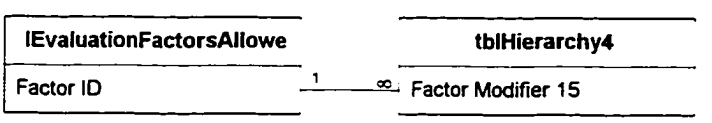


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference54

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference55



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference59

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference79

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

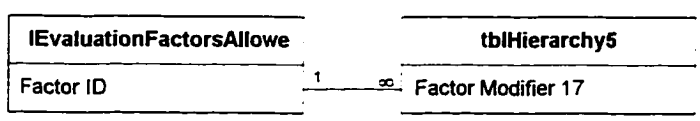
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 17	Number (Long)	4
Factor Modifier 17 Value	Number (Long)	4
Factor Modifier 18	Number (Long)	4
Factor Modifier 18 Value	Number (Long)	4
Factor Modifier 19	Number (Long)	4
Factor Modifier 19 Value	Number (Long)	4
Factor Modifier 20	Number (Long)	4
Factor Modifier 20 Value	Number (Long)	4
Factor Modifier 5a	Number (Long)	4
Description	Text	100

Relationships

Reference61

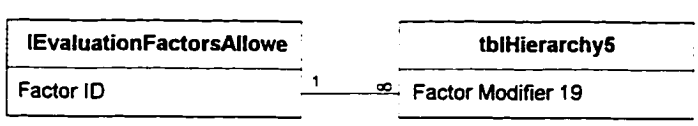


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference62

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

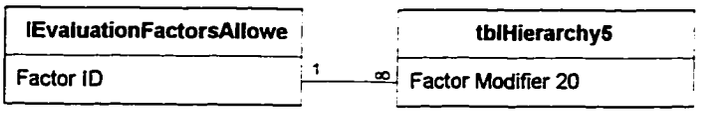
Reference63



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

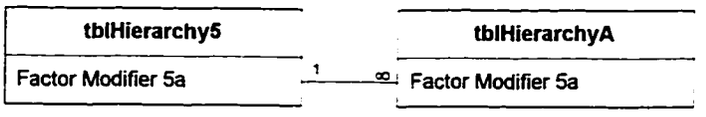
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference64



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference84



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

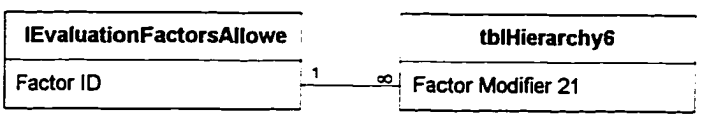
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 21	Number (Long)	4
Factor Modifier 21 Value	Number (Long)	4
Factor Modifier 22	Number (Long)	4
Factor Modifier 22 Value	Number (Long)	4
Factor Modifier 23	Number (Long)	4
Factor Modifier 23 Value	Number (Long)	4
Factor Modifier 24	Number (Long)	4
Factor Modifier 24 Value	Number (Long)	4
Factor Modifier 6a	Number (Long)	4
Description	Text	100

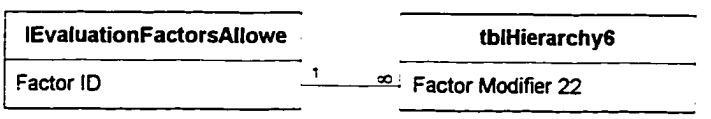
Relationships

Reference65



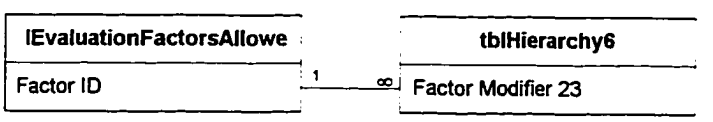
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference66



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference67



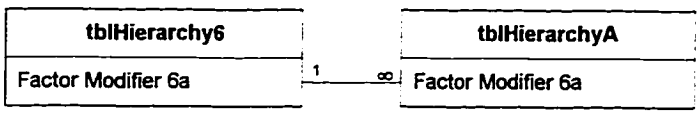
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference68

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference89



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 25	Number (Long)	4
Factor Modifier 25 Value	Number (Long)	4
Factor Modifier 26	Number (Long)	4
Factor Modifier 26 Value	Number (Long)	4
Factor Modifier 27	Number (Long)	4
Factor Modifier 27 Value	Number (Long)	4
Factor Modifier 28	Number (Long)	4
Factor Modifier 28 Value	Number (Long)	4
Factor Modifier 7a	Number (Long)	4
Description	Text	100

Relationships

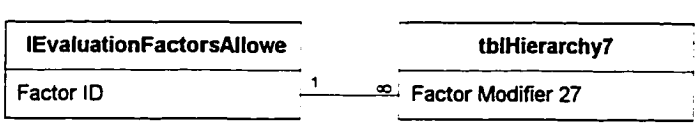
Reference69

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference70

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

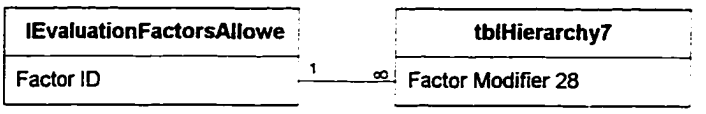
Reference72



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

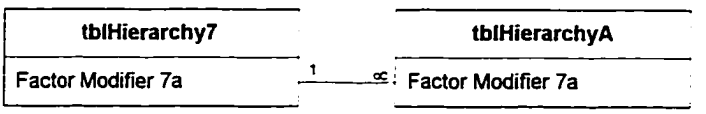
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference73



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference94



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

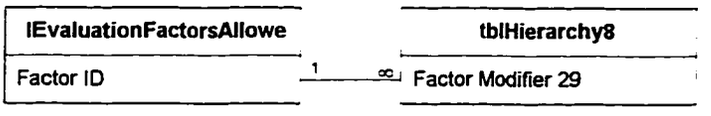
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 29	Number (Long)	4
Factor Modifier 29 Value	Number (Long)	4
Factor Modifier 30	Number (Long)	4
Factor Modifier 30 Value	Number (Long)	4
Factor Modifier 31	Number (Long)	4
Factor Modifier 31 Value	Number (Long)	4
Factor Modifier 32	Number (Long)	4
Factor Modifier 32 Value	Number (Long)	4
Factor Modifier 8a	Number (Long)	4
Description	Text	100

Relationships

Reference75

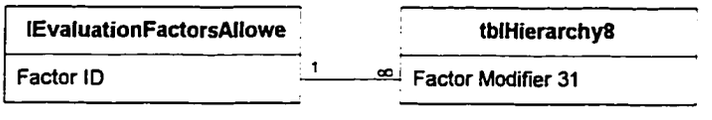


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference76

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

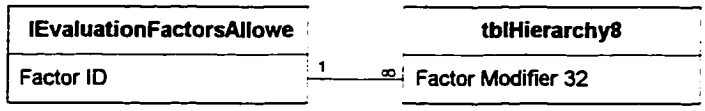
Reference77



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

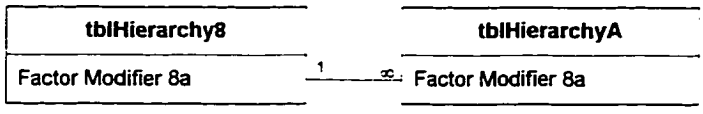
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference78



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference99



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

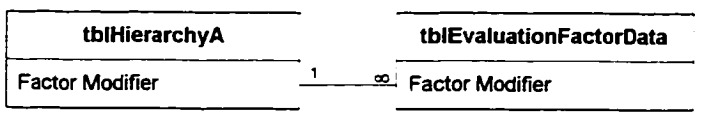
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 1a	Number (Long)	4
Factor Modifier 2a	Number (Long)	4
Factor Modifier 3a	Number (Long)	4
Factor Modifier 4a	Number (Long)	4
Factor Modifier 5a	Number (Long)	4
Factor Modifier 6a	Number (Long)	4
Factor Modifier 7a	Number (Long)	4
Factor Modifier 8a	Number (Long)	4
Factor Modifier	Number (Long)	4
Description	Text	100

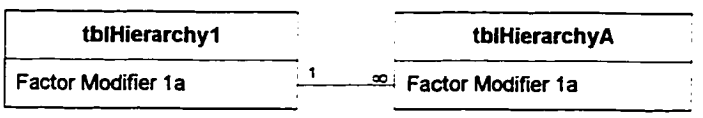
Relationships

Reference28



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference32



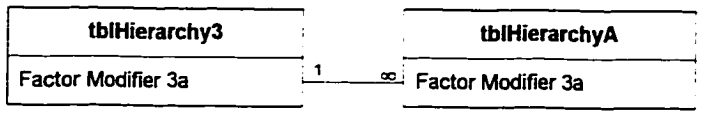
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference39

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

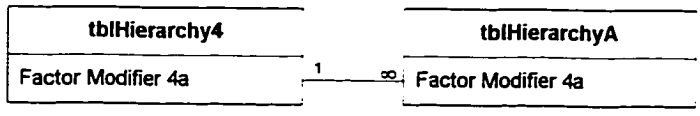
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference74



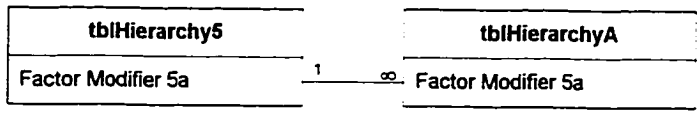
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference79



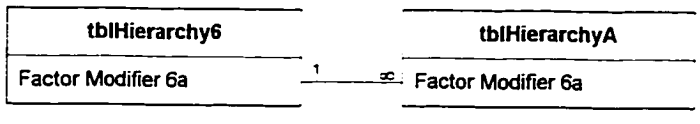
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference84



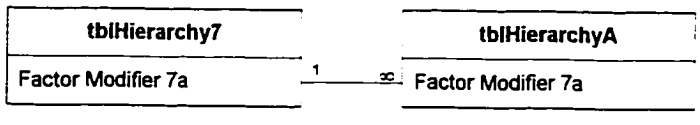
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference89



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

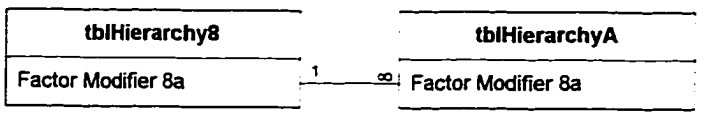
Reference94



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference99



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

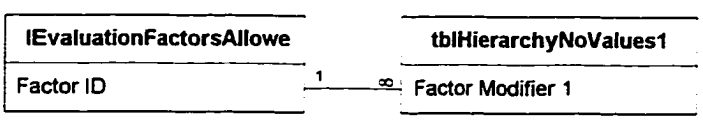
Name	Type	Size
Factor Modifier 1	Number (Long)	4
Factor Modifier 2	Number (Long)	4
Factor Modifier 3	Number (Long)	4
Factor Modifier 4	Number (Long)	4
Factor Modifier 5	Number (Long)	4
Factor Modifier 6	Number (Long)	4
Factor Modifier 7	Number (Long)	4
Factor Modifier 8	Number (Long)	4
FactorModNoValue1a	Number (Long)	4
Description	Text	100

Relationships

Reference117

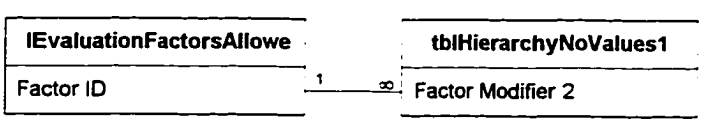
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference80



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

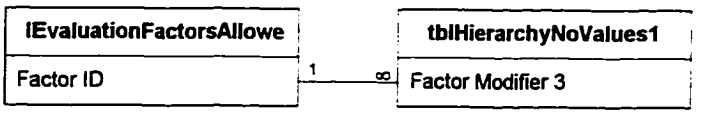
Reference81



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

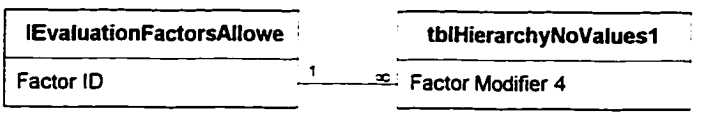
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference82



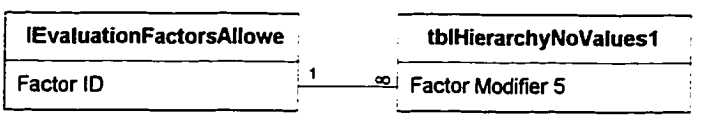
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference83



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference85

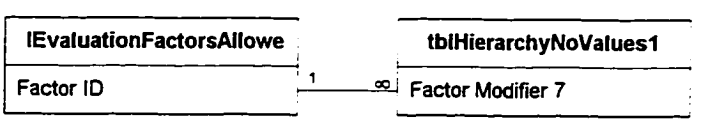


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference86

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

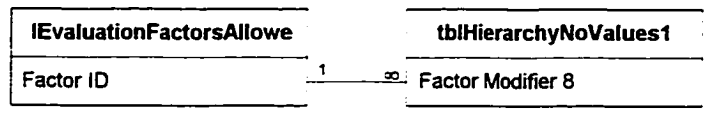
Reference87



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference88



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

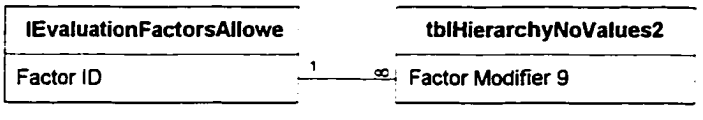
Name	Type	Size
Factor Modifier 9	Number (Long)	4
Factor Modifier 10	Number (Long)	4
Factor Modifier 11	Number (Long)	4
Factor Modifier 12	Number (Long)	4
Factor Modifier 13	Number (Long)	4
Factor Modifier 14	Number (Long)	4
Factor Modifier 15	Number (Long)	4
Factor Modifier 16	Number (Long)	4
FactorModNoValue2a	Number (Long)	4
Description	Text	100

Relationships

Reference109

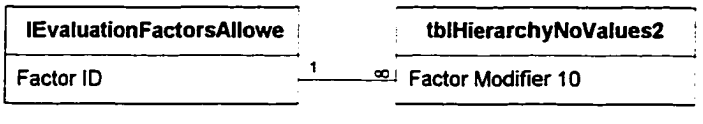
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference90



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

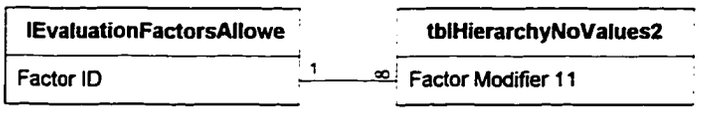
Reference91



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

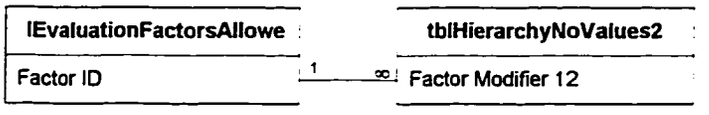
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference92



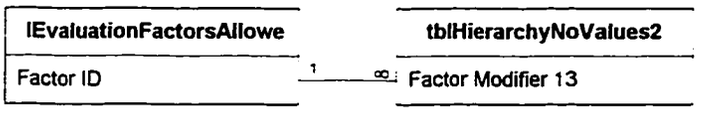
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference93



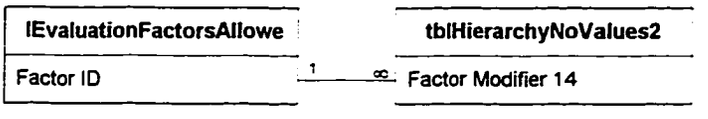
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference95



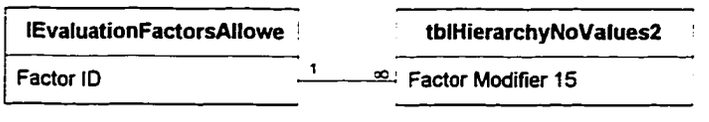
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference96



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

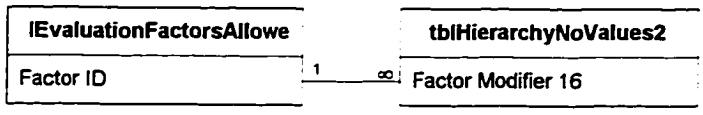
Reference97



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference98



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 17	Number (Long)	4
Factor Modifier 18	Number (Long)	4
Factor Modifier 19	Number (Long)	4
Factor Modifier 20	Number (Long)	4
Factor Modifier 21	Number (Long)	4
Factor Modifier 22	Number (Long)	4
Factor Modifier 23	Number (Long)	4
Factor Modifier 24	Number (Long)	4
FactorModNoValue3a	Number (Long)	4
Description	Text	100

Relationships

Reference100

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference101

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference102

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

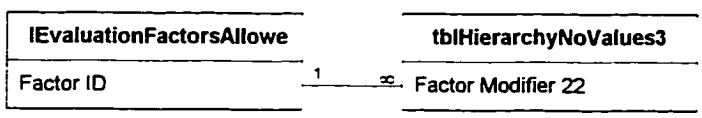
Reference103

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference104

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference105

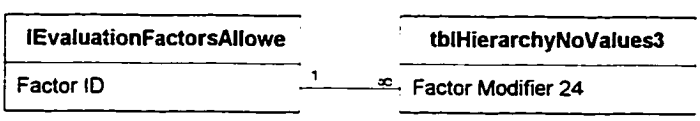


Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference106

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

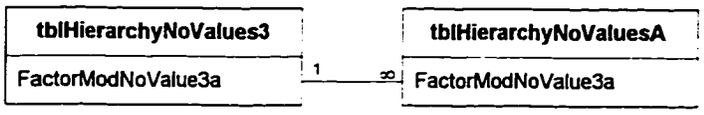
Reference107



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference118



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Factor Modifier 25	Number (Long)	4
Factor Modifier 26	Number (Long)	4
Factor Modifier 27	Number (Long)	4
Factor Modifier 28	Number (Long)	4
Factor Modifier 29	Number (Long)	4
Factor Modifier 30	Number (Long)	4
Factor Modifier 31	Number (Long)	4
Factor Modifier 32	Number (Long)	4
FactorModNoValue4a	Number (Long)	4
Description	Text	100

Relationships

Reference108

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference110

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference111

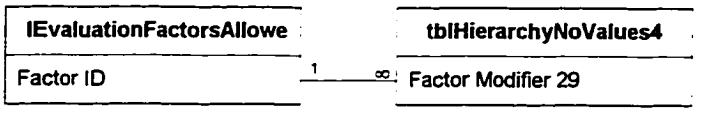
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference112

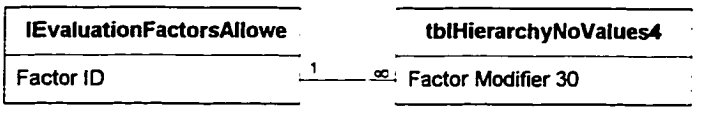
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference113



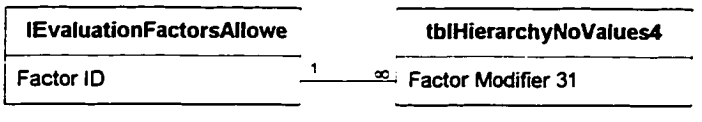
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference114



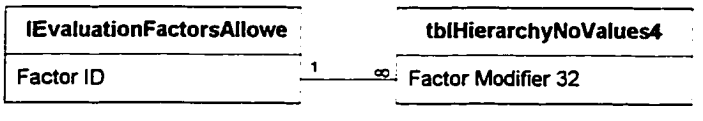
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference115



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

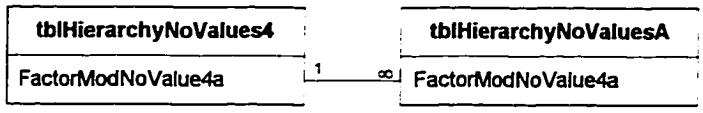
Reference116



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference127



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
FactorModNoValue1a	Number (Long)	4
FactorModNoValue2a	Number (Long)	4
FactorModNoValue3a	Number (Long)	4
FactorModNoValue4a	Number (Long)	4
FactorModifierNoValues	Number (Long)	4
Description	Text	100

Relationships

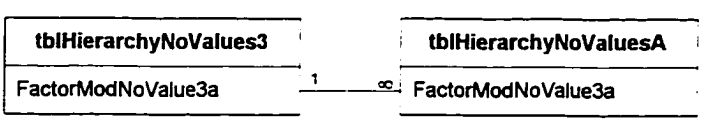
Reference109

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference117

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference118



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference127

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference60

Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Operator	Text	50
Firm Name	Text	50
Contact Name	Text	50

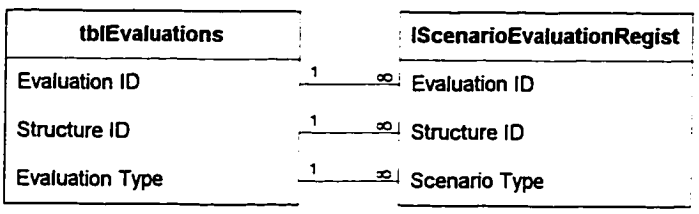
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Scenario ID	Number (Long)	4
Fleet ID	Text	50
Structure ID	Text	50
Evaluation ID	Number (Long)	4
Scenario Type	Text	15
Scenario Evaluation ID	Text	50

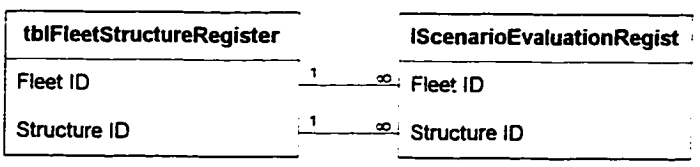
Relationships

Reference21



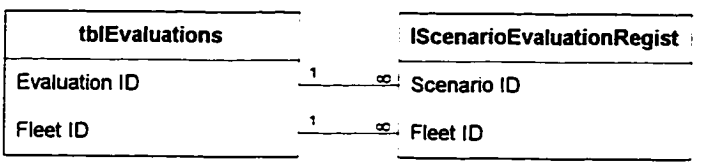
Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference22



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference3



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

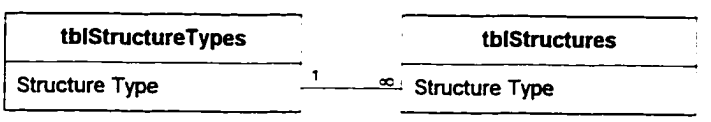
Name	Type	Size
Structure ID	Text	50
Structure Name	Text	50
Structure Type	Text	50
Structure Description	Text	50
Structure Comments	Memo	-

Relationships

Reference5

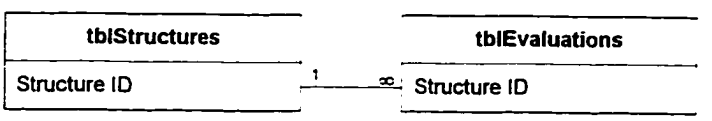
Attributes: One to Many, Enforced

Reference7



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reference8



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

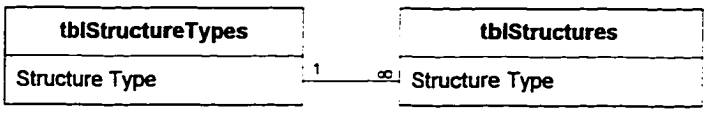
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Columns

Name	Type	Size
Structure Type	Text	50
Description	Text	50
Comments	Memo	-

Relationships

Reference7



Attributes: One to Many, Enforced, Cascade Updates, Cascade Deletes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

D SAMPLE VALUES OF P(U|"S") AND CORRELATION

Given the number of structures in a fleet, the number of those failures which can be expected to fail each year, an estimate of the number of structures which a certain test will label as "safe" when actually unsafe (quantity (1,2)), and an estimate of the number of structures which that test will label as "safe" when actually safe (quantity (1,1)): the probability P(U|"S") of any unsafe structure being labeled "safe" by the test can be estimated, as well as test correlation ρ_T .

Equations (3) and (4) of Chapter 5 were used to produce such estimates for the following table. More are given in Table 2 of Chapter 5.

Number of Structures	Number of Failures	(1,2): unsafe & tested "safe"	(1,1): safe & tested "safe"	P(U|"S")	ρ_T
1000	5	0	200	0.0000	0.0354
1000	5	0	500	0.0000	0.0709
1000	5	0	950	0.0000	0.3090
1000	5	0	995	0.0000	1.0000
1000	5	2	200	0.0099	-0.0350
1000	5	2	500	0.0040	0.0145
1000	5	2	950	0.0021	0.1831
1000	5	2	995	0.0020	0.7738
1000	5	4	200	0.0196	-0.1048
1000	5	4	500	0.0079	-0.0420
1000	5	4	950	0.0042	0.0521
1000	5	4	995	0.0040	0.4463
1000	10	0	200	0.0000	0.0503
1000	10	0	500	0.0000	0.1005
1000	10	0	950	0.0000	0.4381
1000	10	0	990	0.0000	1.0000
1000	10	4	200	0.0196	-0.0489
1000	10	4	500	0.0079	0.0209
1000	10	4	950	0.0042	0.2658
1000	10	4	990	0.0040	0.7730
1000	10	8	200	0.0385	-0.1466

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Number of Structures	Number of Failures	(1,2): unsafe & tested "safe"	(1,1): safe & tested "safe"	P(U|"S")	ρ_T
1000	10	8	500	0.0157	-0.0587
1000	10	8	950	0.0084	0.0792
1000	10	8	990	0.0080	0.4454
1000	50	0	200	0.0000	0.1147
1000	50	0	500	0.0000	0.2294
1000	50	0	950	0.0000	1.0000
1000	50	0	950	0.0000	1.0000
1000	50	20	200	0.0909	-0.0997
1000	50	20	500	0.0385	0.0551
1000	50	20	950	0.0206	0.7666
1000	50	20	950	0.0206	0.7666
1000	50	40	200	0.1667	-0.3008
1000	50	40	500	0.0741	-0.1197
1000	50	40	950	0.0404	0.4381
1000	50	40	950	0.0404	0.4381
2000	5	0	400	0.0000	0.0250
2000	5	0	1000	0.0000	0.0501
2000	5	0	1900	0.0000	0.2182
2000	5	0	1995	0.0000	1.0000
2000	5	2	400	0.0050	-0.0249
2000	5	2	1000	0.0020	0.0101
2000	5	2	1900	0.0011	0.1278
2000	5	2	1995	0.0010	0.7742
2000	5	4	400	0.0099	-0.0746
2000	5	4	1000	0.0040	-0.0298
2000	5	4	1900	0.0021	0.0356
2000	5	4	1995	0.0020	0.4468
2000	10	0	400	0.0000	0.0354
2000	10	0	1000	0.0000	0.0709
2000	10	0	1900	0.0000	0.3090
2000	10	0	1990	0.0000	1.0000
2000	10	4	400	0.0099	-0.0350
2000	10	4	1000	0.0040	0.0145
2000	10	4	1900	0.0021	0.1831
2000	10	4	1990	0.0020	0.7738
2000	10	8	400	0.0196	-0.1048
2000	10	8	1000	0.0079	-0.0420
2000	10	8	1900	0.0042	0.0521
2000	10	8	1990	0.0040	0.4463
2000	50	0	400	0.0000	0.0801
2000	50	0	1000	0.0000	0.1601
2000	50	0	1900	0.0000	0.6980
2000	50	0	1950	0.0000	1.0000
2000	50	20	400	0.0476	-0.0747
2000	50	20	1000	0.0196	0.0352
2000	50	20	1900	0.0104	0.4576
2000	50	20	1950	0.0102	0.7707
2000	50	40	400	0.0909	-0.2242

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Number of Structures	Number of Failures	(1,2): unsafe & tested "safe"	(1,1): safe & tested "safe"	P(U|"S")	ρ_T
2000	50	40	1000	0.0385	-0.0897
2000	50	40	1900	0.0206	0.1596
2000	50	40	1950	0.0201	0.4427
4000	5	0	800	0.0000	0.0177
4000	5	0	2000	0.0000	0.0354
4000	5	0	3800	0.0000	0.1542
4000	5	0	3995	0.0000	1.0000
4000	5	2	800	0.0025	-0.0176
4000	5	2	2000	0.0010	0.0071
4000	5	2	3800	0.0005	0.0898
4000	5	2	3995	0.0005	0.7744
4000	5	4	800	0.0050	-0.0529
4000	5	4	2000	0.0020	-0.0212
4000	5	4	3800	0.0011	0.0247
4000	5	4	3995	0.0010	0.4470
4000	10	0	800	0.0000	0.0250
4000	10	0	2000	0.0000	0.0501
4000	10	0	3800	0.0000	0.2182
4000	10	0	3990	0.0000	1.0000
4000	10	4	800	0.0050	-0.0249
4000	10	4	2000	0.0020	0.0101
4000	10	4	3800	0.0011	0.1278
4000	10	4	3990	0.0010	0.7742
4000	10	8	800	0.0099	-0.0746
4000	10	8	2000	0.0040	-0.0298
4000	10	8	3800	0.0021	0.0356
4000	10	8	3990	0.0020	0.4468
4000	50	0	800	0.0000	0.0563
4000	50	0	2000	0.0000	0.1125
4000	50	0	3800	0.0000	0.4904
4000	50	0	3950	0.0000	1.0000
4000	50	20	800	0.0244	-0.0543
4000	50	20	2000	0.0099	0.0236
4000	50	20	3800	0.0052	0.3012
4000	50	20	3950	0.0050	0.7726
4000	50	40	800	0.0476	-0.1630
4000	50	40	2000	0.0196	-0.0653
4000	50	40	3800	0.0104	0.0919
4000	50	40	3950	0.0100	0.4450
8000	5	0	1600	0.0000	0.0125
8000	5	0	4000	0.0000	0.0250
8000	5	0	7600	0.0000	0.1090
8000	5	0	7995	0.0000	1.0000
8000	5	2	1600	0.0012	-0.0125
8000	5	2	4000	0.0005	0.0050
8000	5	2	7600	0.0003	0.0633
8000	5	2	7995	0.0003	0.7745
8000	5	4	1600	0.0025	-0.0374

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

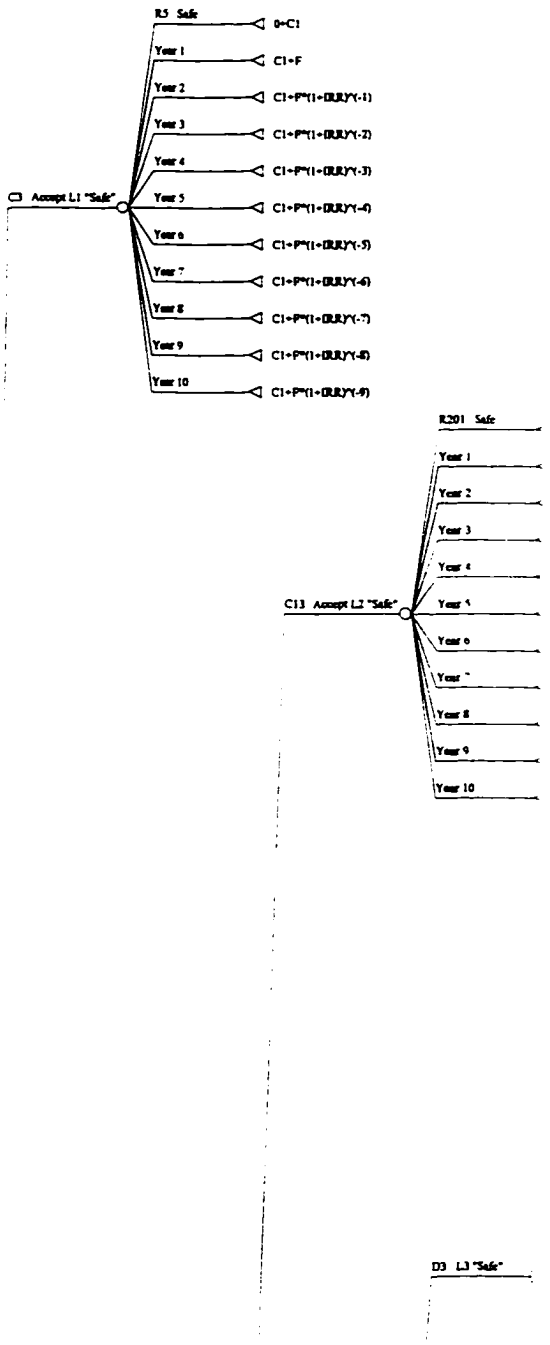
Number of Structures	Number of Failures	(1,2): unsafe & tested "safe"	(1,1): safe & tested "safe"	P(U|"S")	ρ_T
8000	5	4	4000	0.0010	-0.0150
8000	5	4	7600	0.0005	0.0174
8000	5	4	7995	0.0005	0.4471
8000	10	0	1600	0.0000	0.0177
8000	10	0	4000	0.0000	0.0354
8000	10	0	7600	0.0000	0.1542
8000	10	0	7990	0.0000	1.0000
8000	10	4	1600	0.0025	-0.0176
8000	10	4	4000	0.0010	0.0071
8000	10	4	7600	0.0005	0.0898
8000	10	4	7990	0.0005	0.7744
8000	10	8	1600	0.0050	-0.0529
8000	10	8	4000	0.0020	-0.0212
8000	10	8	7600	0.0011	0.0247
8000	10	8	7990	0.0010	0.4470
8000	50	0	1600	0.0000	0.0397
8000	50	0	4000	0.0000	0.0793
8000	50	0	7600	0.0000	0.3457
8000	50	0	7950	0.0000	1.0000
8000	50	20	1600	0.0123	-0.0390
8000	50	20	4000	0.0050	0.0163
8000	50	20	7600	0.0026	0.2060
8000	50	20	7950	0.0025	0.7736
8000	50	40	1600	0.0244	-0.1169
8000	50	40	4000	0.0099	-0.0468
8000	50	40	7600	0.0052	0.0593
8000	50	40	7950	0.0050	0.4461

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

E FOUR-LEVEL DECISION TREE ("MODEL 5T") INCORPORATING LEVEL-SKIPPING AND TIME

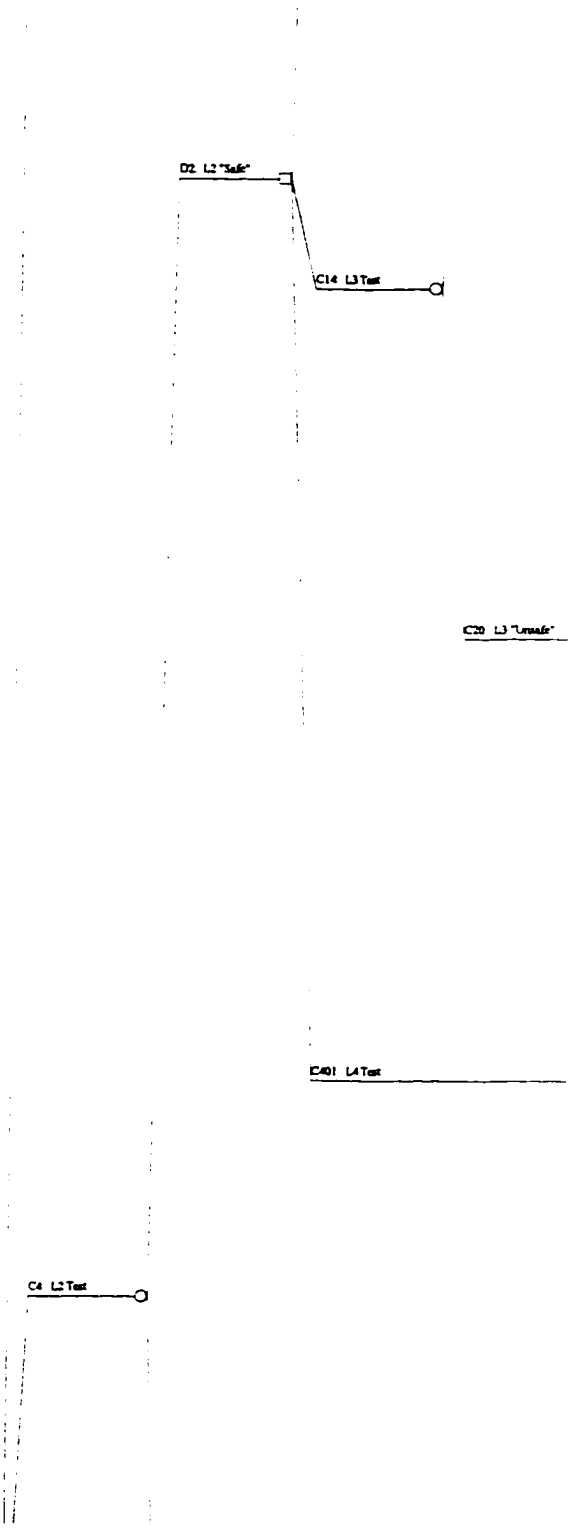
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Column 1, Row 1, Page 1



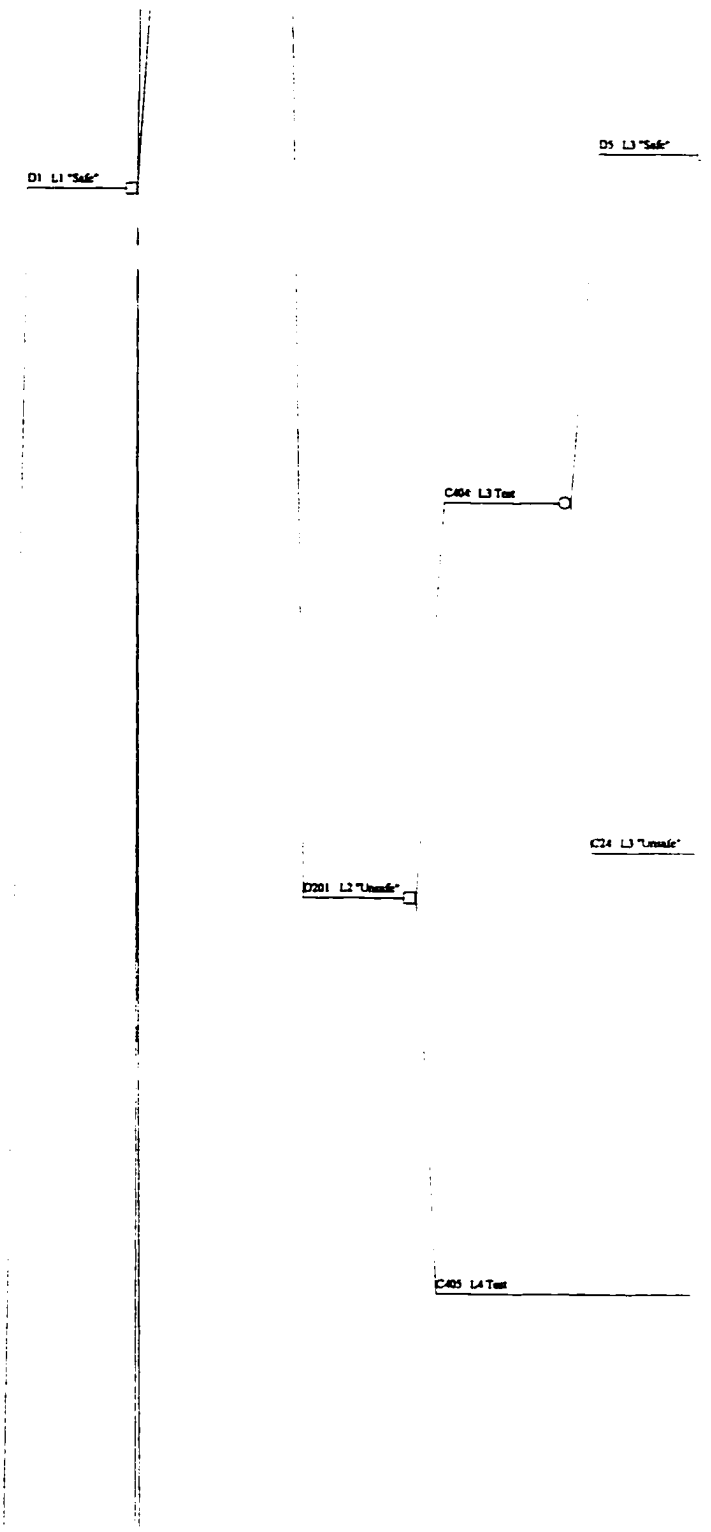
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Column 1, Row 2, Page 2



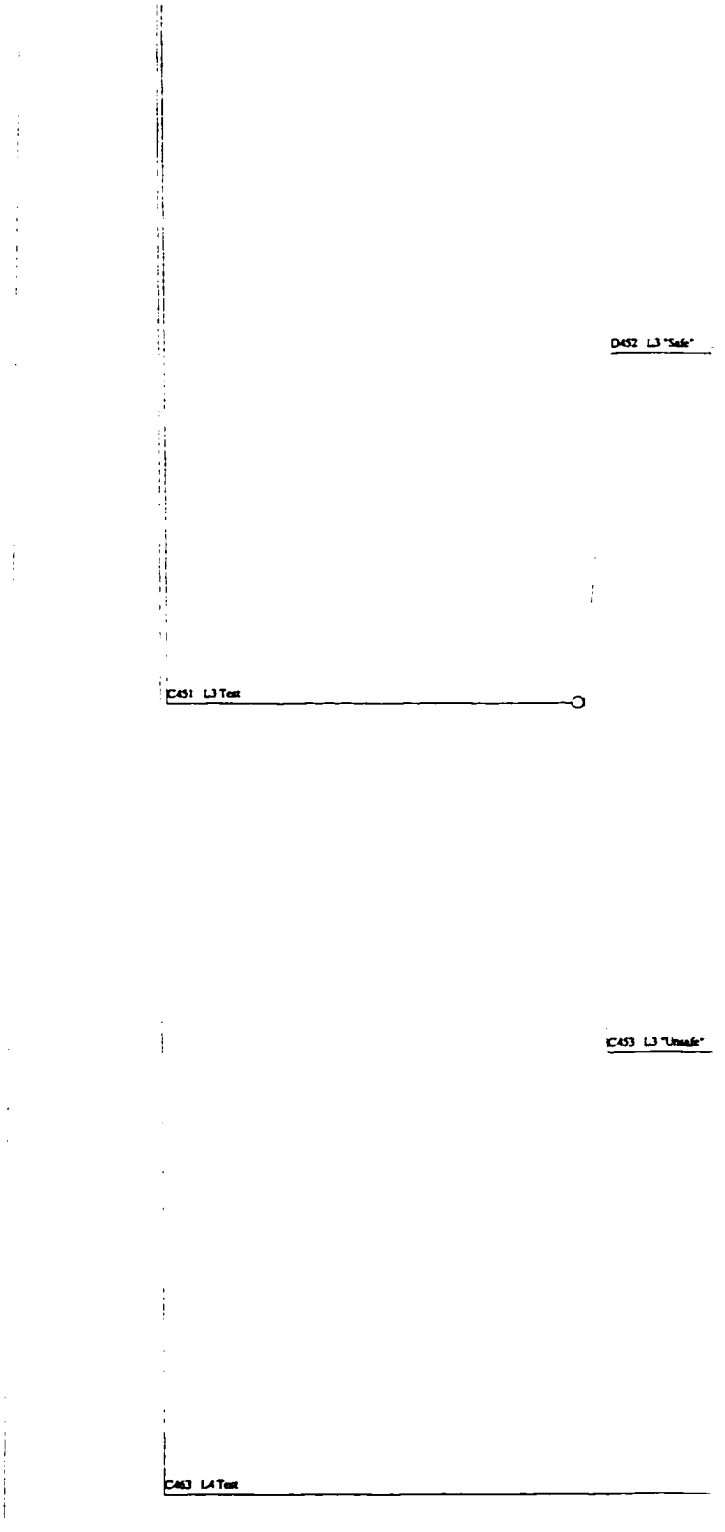
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Column 1, Row 3, Page 3

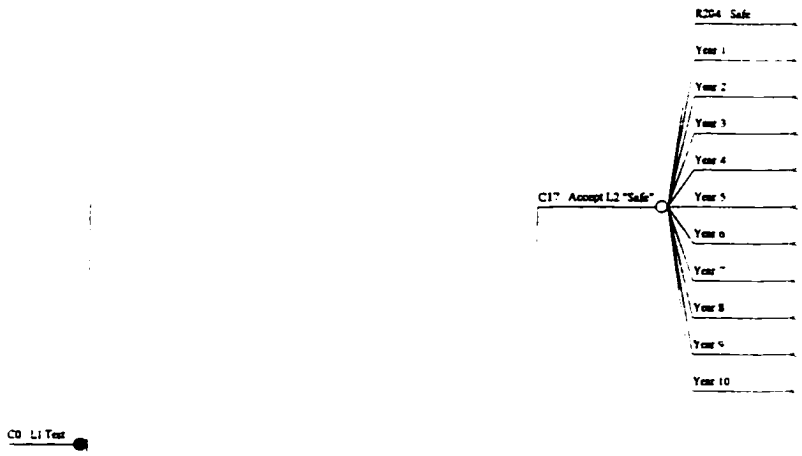


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Column 1, Row 4, Page 4

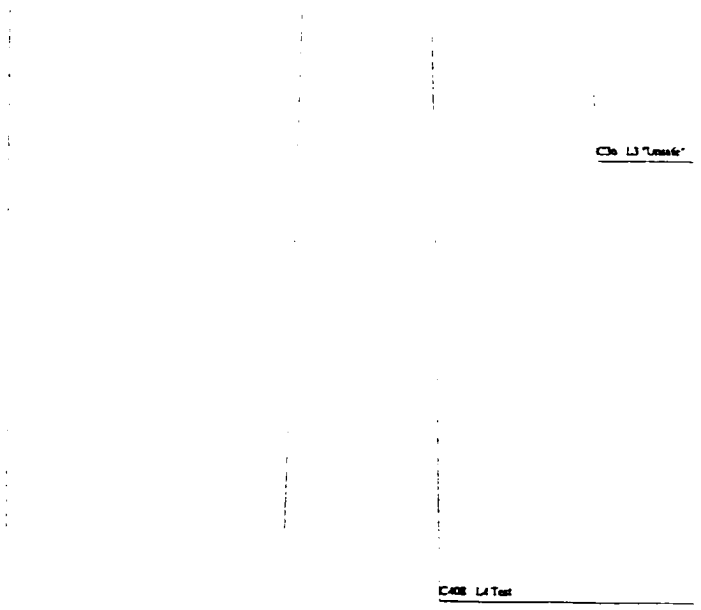


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



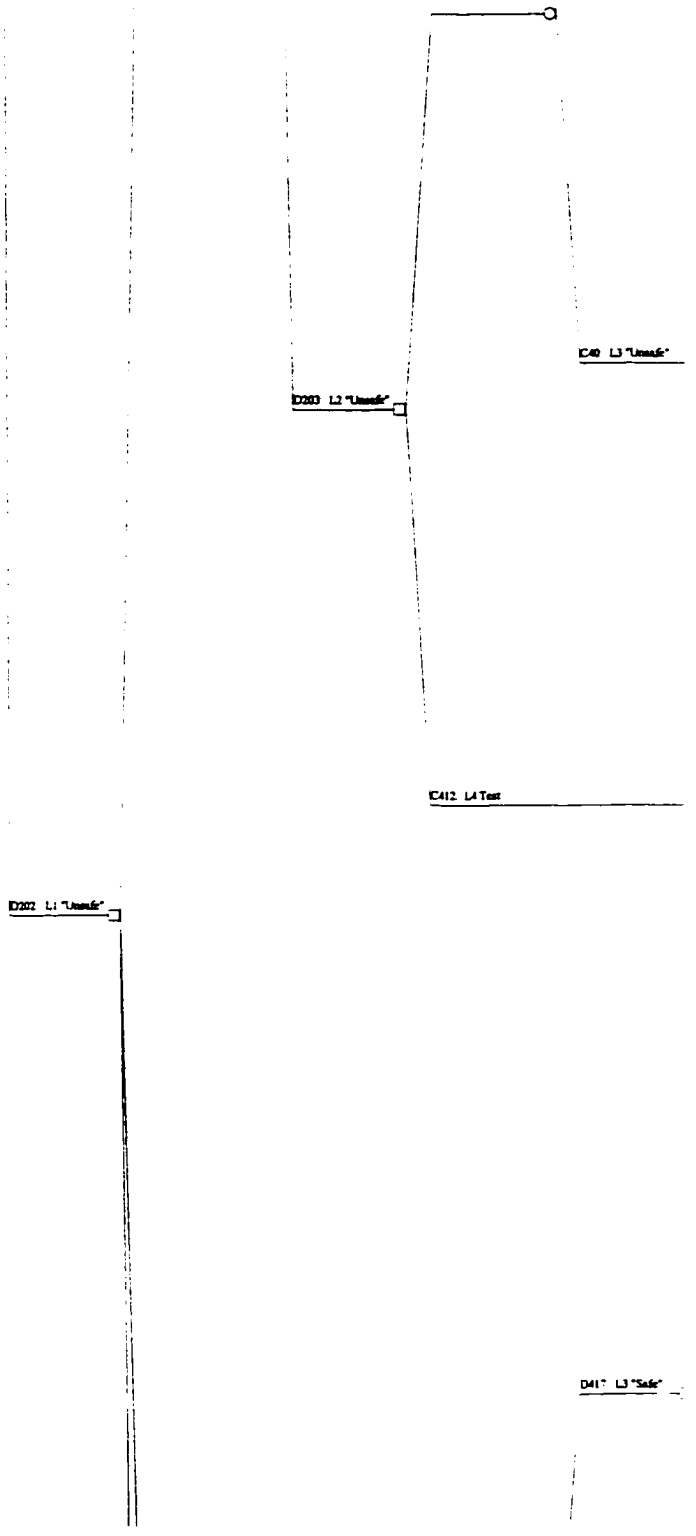
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Column 1, Row 6, Page 6



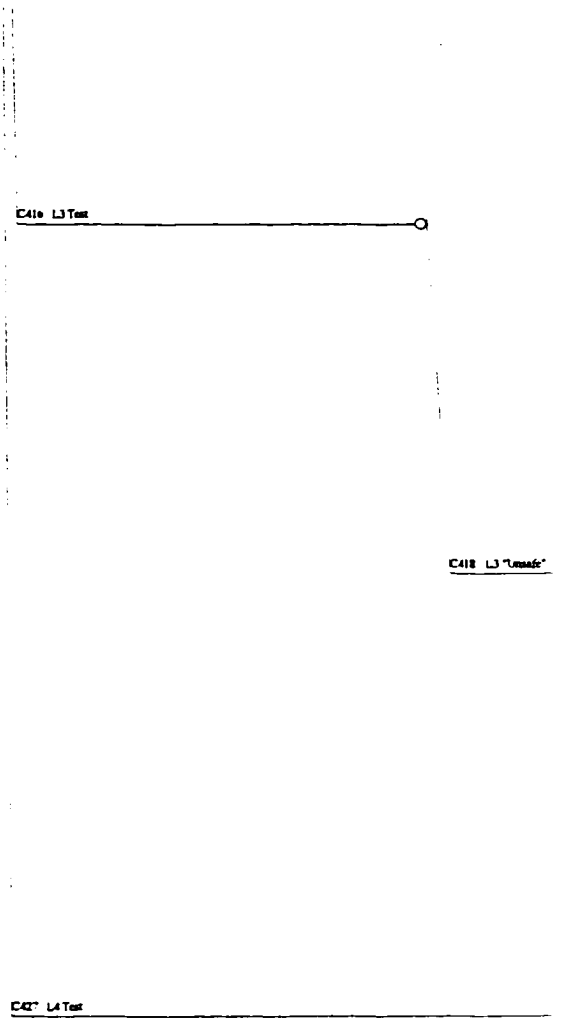
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Column 1, Row 7, Page 7

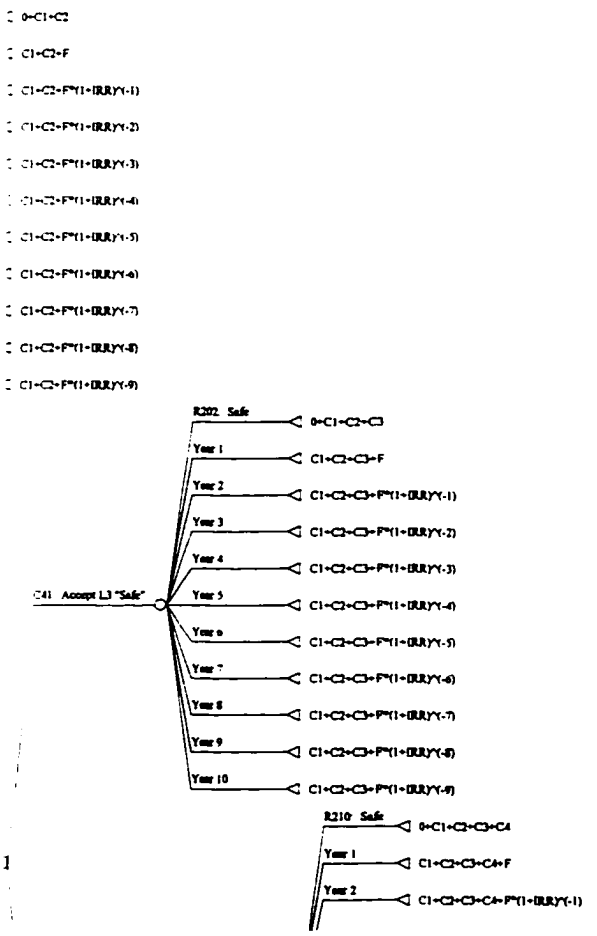


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

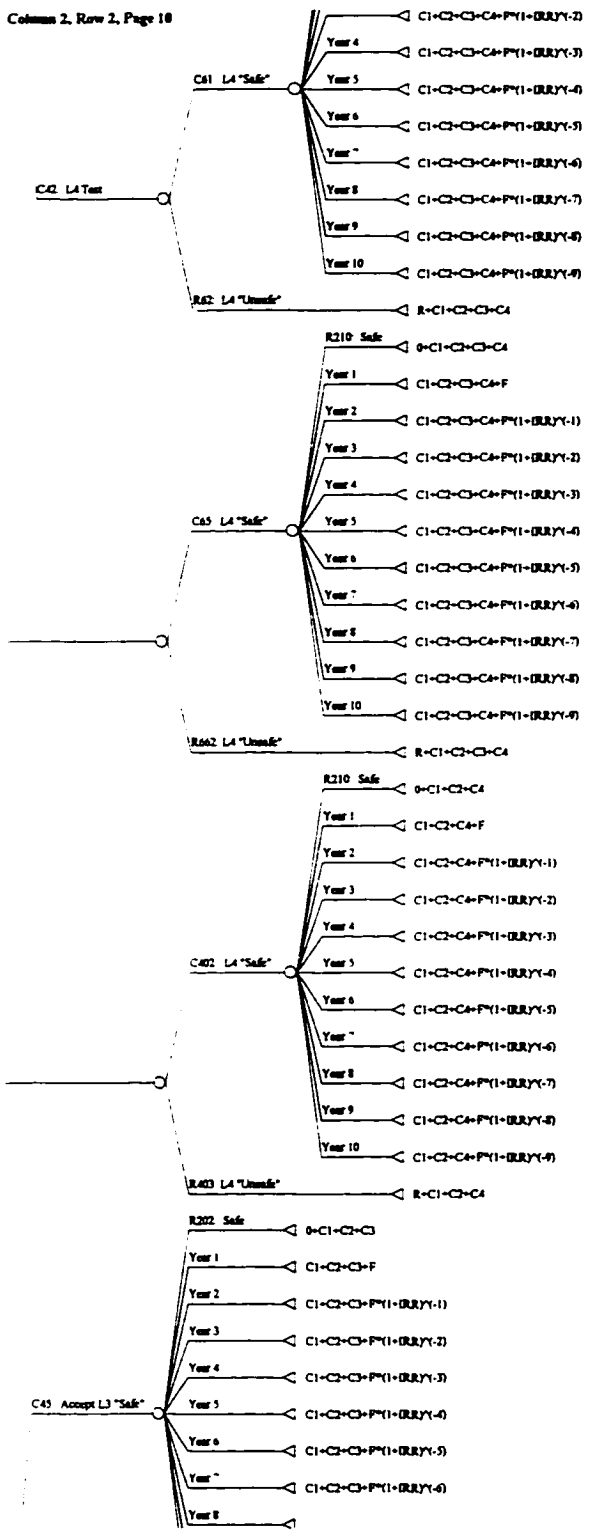
Column 1, Row 8, Page 8



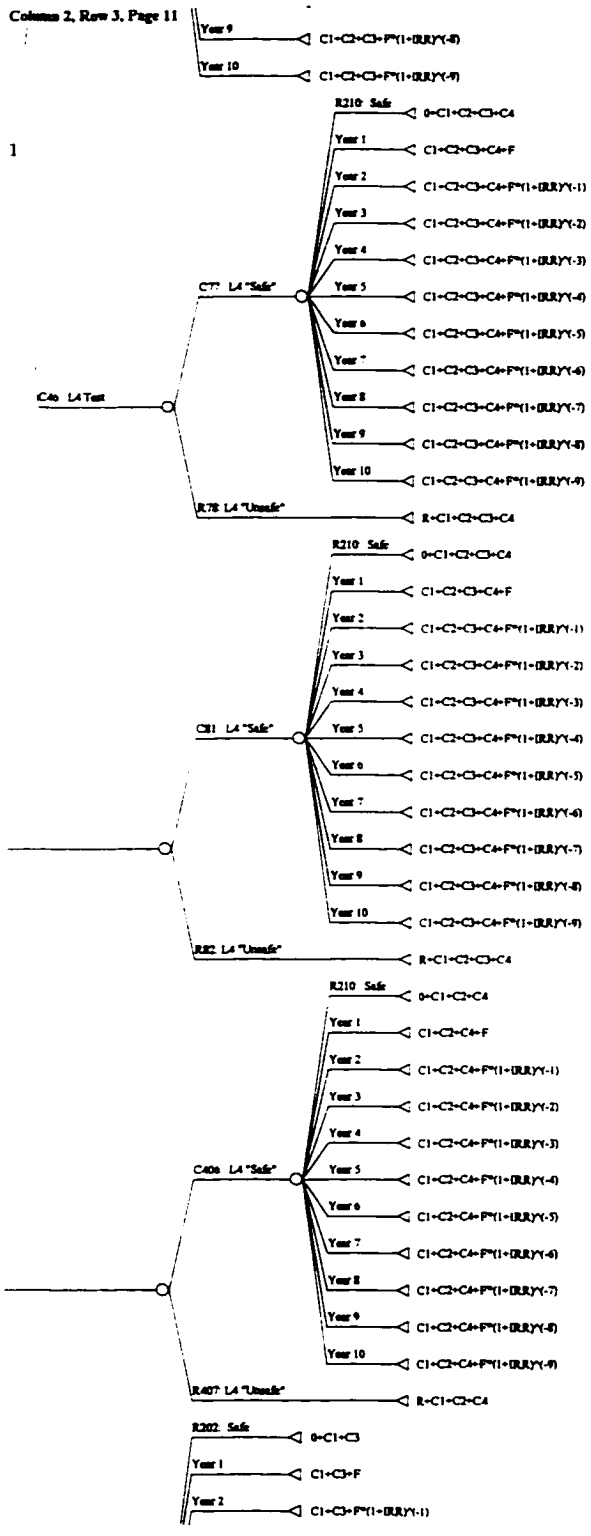
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



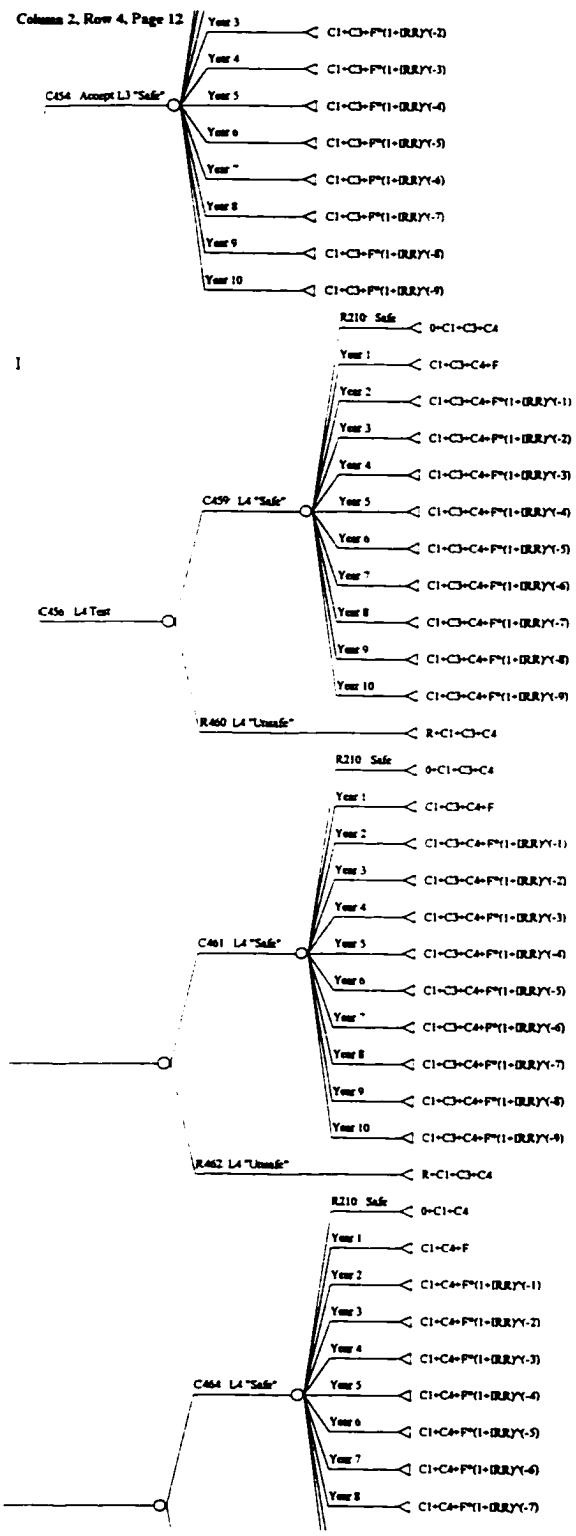
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



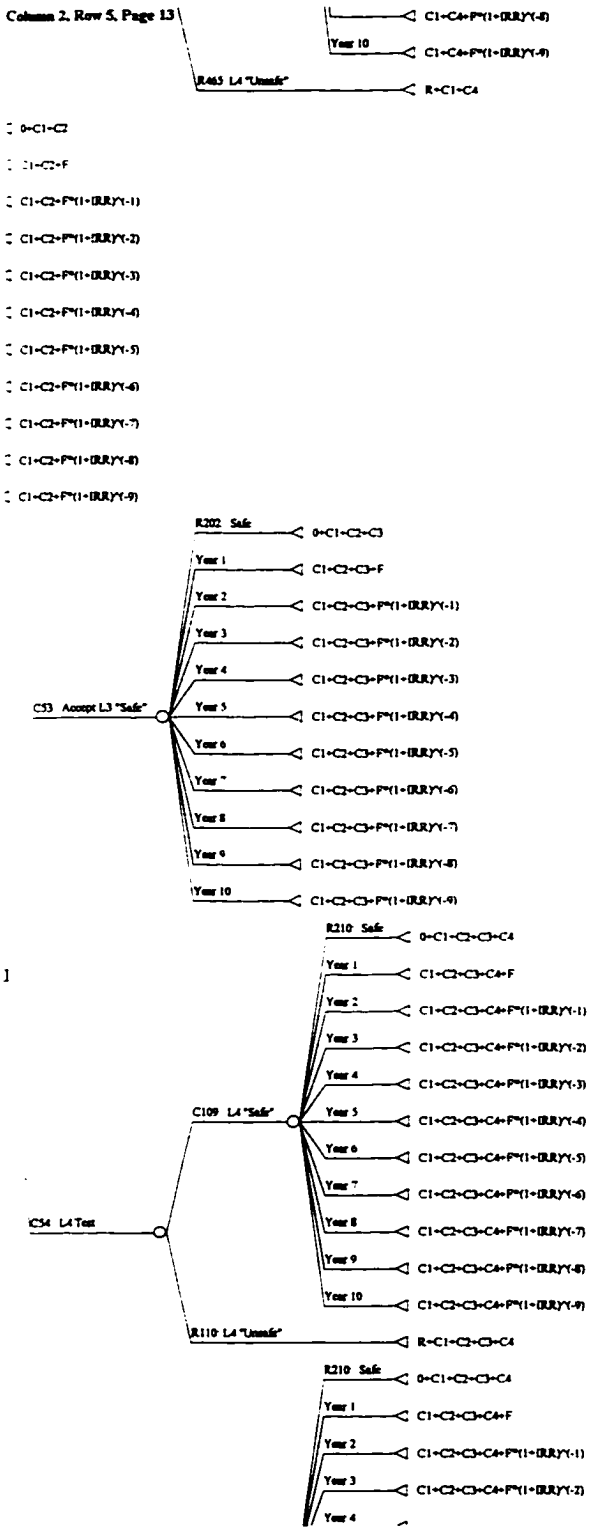
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



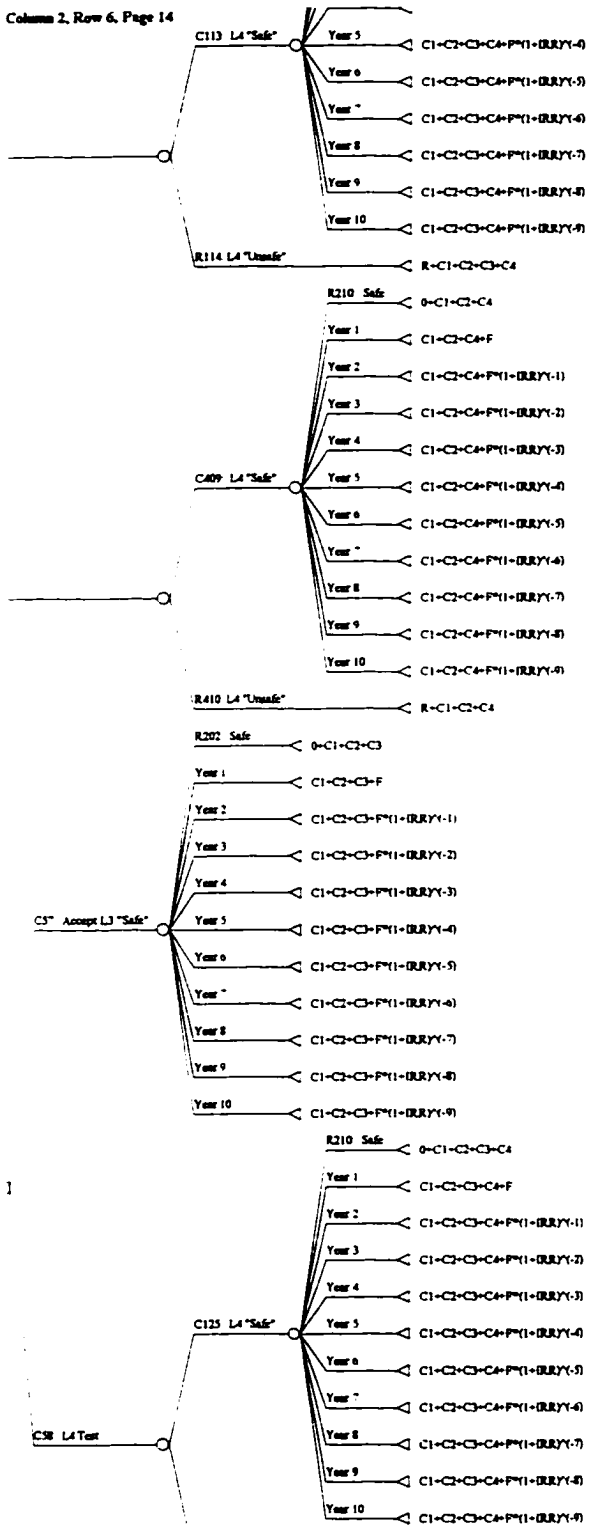
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



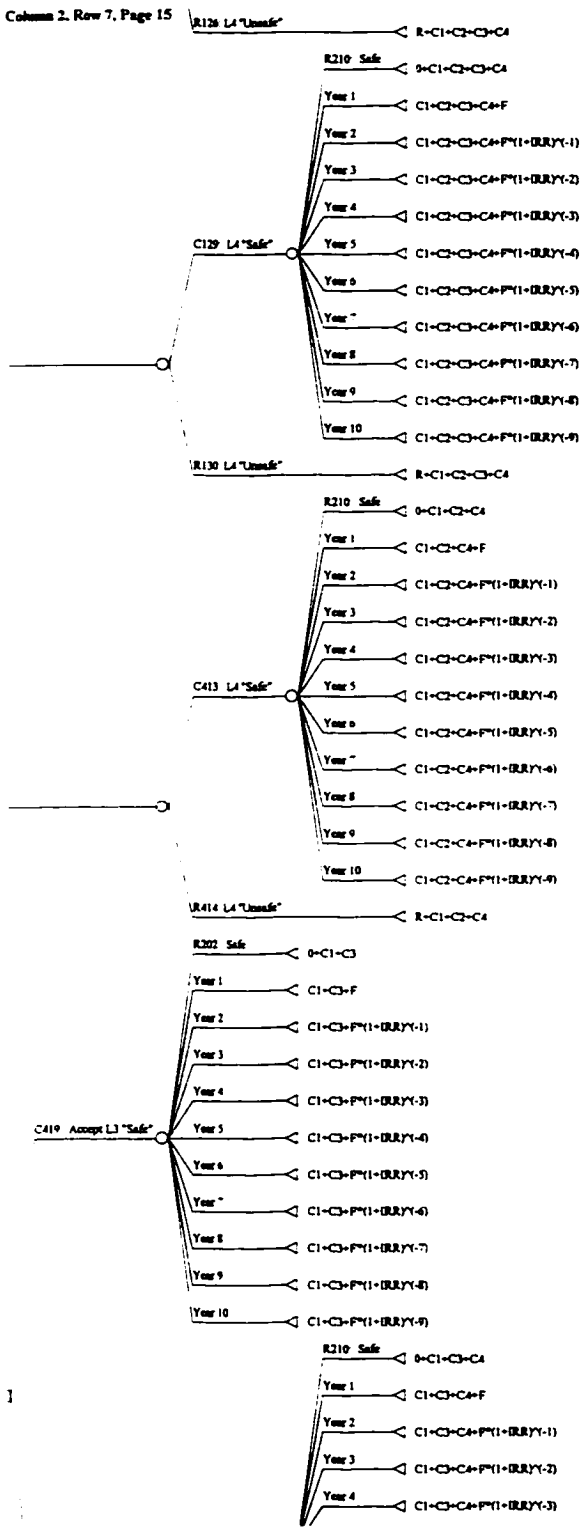
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



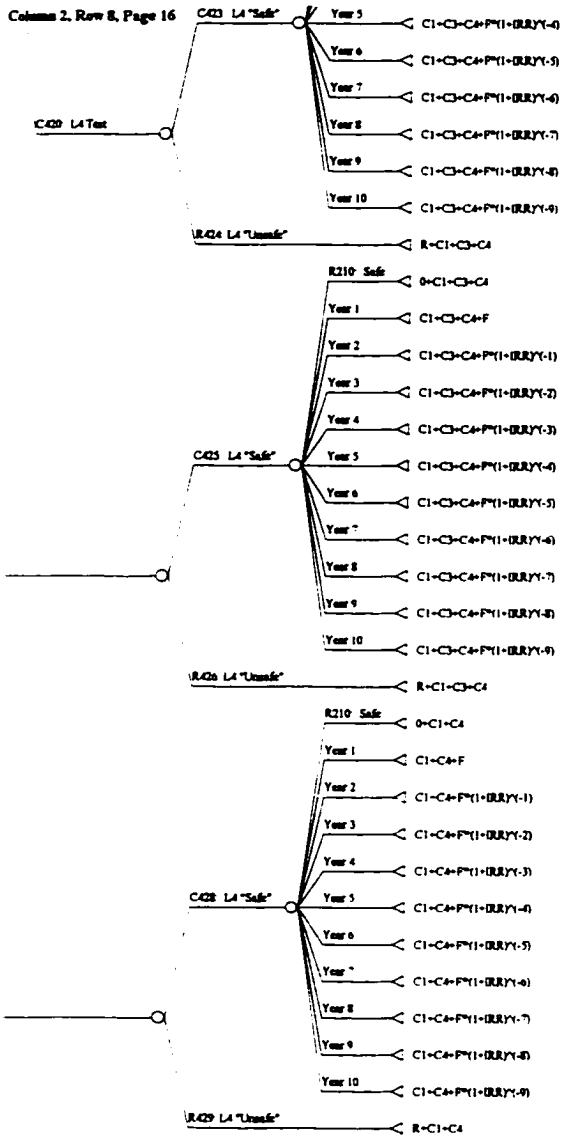
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

F SAMPLE CODE FOR MULTIPLE HIERARCHY EVALUATIONS

This code is meant to demonstrate the principles behind dealing with hierarchies in DAESSIM. An application built from this sample code may experience difficulties. Refer to the "evalPlatformsL2I1" and "evalPlatformsL2RSR1" modules, and to the Microsoft Access help file, for assistance.

INSPECTION ROUTINE: evalDeleteMeCCMSI1 Code

```
Function EstCostCCMSI1 ()
'  *** Note that input boxes select the factors they display rather than working
'  *** off a table (by contrast, the L1 generic inspection routine works straight
'  *** off tblEvaluationFactorData).
   Dim x As Variant
   Dim y As Variant
   Dim z As Variant
   Dim CallingForm As Form
   Dim InfoFormName As String
   Dim InputFormName As String
   Dim InputChoice As Integer
   Dim MenuFormName As String
   Dim thisEvalIDToUse As Integer

   On Error GoTo ErrorHandlerEstCostCCMSI1

' Get the name of the form calling this procedure
   Set CallingForm = screen.activeform
   CallingFormName = CallingForm.name

' Set the name of the form to use for input
   InputFormName = "frmL2RSR1Start"

' Open the information box
   InfoFormName = "frmEvalDescription1"
   DoCmd OpenForm InfoFormName
   forms(InfoFormName).txtEvalDescrip2.caption = DLookup("[Eval Descrip 2]", "tblEvaluationTypes", "[Evaluation
Type] = 'EstCostCCMSI1'")
   forms(InfoFormName).txtMessage1.caption = "Processing..."
   DoCmd RepaintObject A_FORM, InfoFormName

EstCostCCMSI1DoInput:
' Open the input switchboard - necessary for complex procedures
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    Pass4 = 0       'Initialize the output of the dialog box
    ReDim Pass(4, 20, 4)
    Pass(2, 1, 3) = "Choose a procedure:"
    Pass(2, 2, 3) = "Click a button:"
    Pass(2, 3, 3) = "CCMS Cost Estimate Inputs"
    Pass(2, 4, 3) = ""
    Pass(2, 5, 3) = ""
    Pass(2, 6, 3) = ""
    Pass(2, 7, 3) = ""
    Pass(2, 8, 3) = ""
    Pass(2, 9, 3) = ""
    Pass(2, 10, 3) = ""
    Pass(2, 11, 3) = ""
    Pass(2, 12, 3) = ""
    Pass(2, 13, 3) = ""
    Pass(2, 14, 3) = ""
    Pass(2, 15, 3) = ""
    Pass(2, 16, 3) = ""
    Pass(2, 17, 3) = ""
    Pass(2, 18, 3) = "Done"

    MenuFormName = "frmInputSwitchboard"
    DoCmd OpenForm MenuFormName, , , , , A_DIALOG
    Select Case Pass4
     Case 1
       x = EstCostCCMSI1_2(CallingFormName, InfoFormName, InputFormName)   'Get Number of Floors
       x = EstCostCCMSI1_3(CallingFormName, InfoFormName, InputFormName)   'Get numbers of points
       GoTo EstCostCCMSI1DoInput
     Case Else
     End Select

' Close the information box
    'Forms(InfoFormName).txtMessage1.visible = False  ' Comment out if not using a custom info form.
    DoCmd Close A_FORM, InfoFormName

' Get out of here
    Set CallingForm = Nothing
    CallingFormName = ""
    Exit Function

ErrorHandlerEstCostCCMSI1:
    MsgBox "There has been a fatal error in the EstCostCCMSI1 evaluation."
    ' Don't repaint frmEvalDescription, because we want to see where it stopped!
    ' forms![frmEvalDescription1].txtMessage1.caption = "Fatal error!"
    ' DoCmd RepaintObject A_FORM, "frmEvalDescription1"
    Exit Function

End Function

*************************************************************************

Function EstCostCCMSI1_2 (CallingFormName, InfoFormName, InputFormName)
    Dim MyDB As Database
    Dim MyRecords As Recordset
    Dim myQuery As QueryDef
    Dim myQuery1 As QueryDef
    Dim thisEvalInputs As Recordset
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    Dim numFactorTypes As Integer
    Dim i As Integer
    Dim thisFactorID() As Integer
    Dim thisFactorDescrip() As String
    Dim temp As Double
    Dim ExistingValue() As Variant
    Dim LowValue() As Double
    Dim HighValue() As Double

    forms(CallingFormName).Refresh    ' Saves the record currently being worked on in the active form.  Was DoCmd
DoMenuItem A_FORMBAR, A_FILE, A_SAVERECORD, but this relies on keeping track of forms.
    DoCmd SetWarnings False    ' Turns off annoying warnings

    forms(InfoFormName).txtMessage1.caption = "Number of floors... "
    DoCmd RepaintObject A_FORM, InfoFormName

    '1.  Get the variables that iterate - like, "how many bays are in this platform?"
    On Error GoTo Error_EstCostCCMSI1_2_1
    'Go to the tables and find out what the iterative values are:
    ' Note that values of factor modifiers will always be integers - or else!
    'Look for FactorModifierNoValues = "3":  this is the "iterator" and denotes factors that need to be input first
    ' Note that not all "iterators" will actually iterate - some are just required for later information,
    ' depending on the vagaries of the programmer.  For example, "Platform Type" is needed to tell L2RSR1
    ' how many legs a platform has - while "legs" may be part of an iterative loop, "Platform Type" is not.
    'Note that there's a better, unimplemented way - ignore this routine, and just input iterated factors until finished.
Use a code to denote the end of input.  Then the values that were collected here are simply the max values of the
inputted factors.
    DoCmd RunSQL "SELECT DISTINCTROW tblEvaluationTypeFactors.[Factor ID], tblEvaluationTypeFac-
tors.FactorModifierNoValues INTO thisEvalInputs FROM tblEvaluationTypeFactors WHERE ((tblEvaluationType-
Factors.[Evaluation Type]=forms![" & CallingFormName & "]![ctlEvalTypeToUse].value) AND (tblEvaluationType-
Factors.FactorModifierNoValues=3));"
    Set MyDB = dbengine.workspaces(0).databases(0)
    Set MyRecords = MyDB.OpenRecordset("thisEvalInputs")    ' Opens up the table created in the above RunSQL
statement
    numFactorTypes = MyRecords.recordcount    ' How many records were found?
    If numFactorTypes = -1 Then    ' Big trouble
      MsgBox "tblEvaluationTypeFactors no longer exists.  Seek help."
      DoCmd SetWarnings True
      Exit Function
     End If
    If numFactorTypes = 0 Then
      ' close down the function
      MsgBox "No data input (apart from that for prior evaluations) is required for this procedure.  Proceed to the
""Perform Evaluation"" button .  (If this seems incorrect, inspect tblEvaluationTypeModifiersClassN for factor
types.)"
      DoCmd SetWarnings True
      Exit Function
     End If
   For i = 1 To numFactorTypes    ' Loop 1
      ReDim thisFactorID(numFactorTypes) As Integer    ' Re-setting the array to an appropriate size.
      ReDim thisFactorDescrip(numFactorTypes) As String  ' Re-setting the array to an appropriate size.
      If i = 1 Then MyRecords.MoveFirst Else MyRecords.MoveNext    ' Move within the recordset.
      thisFactorID(i) = MyRecords![Factor ID]    ' Get values for matching Factor ID for each retrived Factor Modi-
fier.
      On Error Resume Next
      If i = 1 Then Set myQuery = MyDB.CreateQueryDef("aQryTemp", "SELECT DISTINCTROW thisEvalIn-
puts.[Factor ID], tblEvaluationFactorsAllowed.[Factor Description 1], tblEvaluationFactorsAllowed.[Factor Name]
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
FROM thisEvalInputs INNER JOIN tblEvaluationFactorsAllowed ON thisEvalInputs.[Factor ID] = tblEvaluation-
FactorsAllowed.[Factor ID];")
    On Error GoTo Error_EstCostCCMSI1_2_1
    ' In the above, create query to get descriptions (where text for the dialog box is held) for the factors.
    ' Below, get that text.
    thisFactorDescrip(i) = DLookup("[Factor Description 1]", "aQryTemp", "[Factor ID]=" & thisFactorID(i))
    On Error GoTo Error_EstCostCCMSI1_2_2
    Pass3 = 101
    Pass2 = thisFactorDescrip(i)   ' Send the text to the dialog box.
    'If there's an existing value, insert it into the dialog box
    ReDim ExistingValue(1, 1, 1)
    'MsgBox ExistingValue(1, 1, 1) & "=ExistingValue(1,1,1) (before DLookUp - should be null!)"
    ExistingValue(1, 1, 1) = DLookup("[fldInteger]", "tblEvaluationFactorData", "[Evaluation ID] = " &
forms(CallingFormName)![ctlEvalIDToUse] & " AND [Factor ID] = " & thisFactorID(i) & " AND [Factor Modifier] =
1")
    'MsgBox ExistingValue(1, 1, 1) & "=ExistingValue(1,1,1) (this is after the DLookUp statement)"
    If IsNull(ExistingValue(1, 1, 1)) Then
     Else
      Pass1 = ExistingValue(1, 1, 1)
     End If
EstCostCCMSI1_2:
    DoCmd OpenForm InputFormName, , , , , A_DIALOG     ' Open the dialog box.
    DoCmd RepaintObject A_FORM, InfoFormName
    temp = Pass1       ' Get result of dialog box.
    If IsNull(temp) Then
      MsgBox "Null values are not allowed"
      GoTo EstCostCCMSI1_2
     Else
     End If

    'MsgBox Temp & " = Temp"
    ' Below, add record to tblEvaluationFactorData with values of Factor ID from dialog boxes for the retrieved fac-
tor modifiers.
    ' Now either insert the new value or update the existing value
    On Error GoTo Error_EstCostCCMSI1_2_3
    If IsNull(ExistingValue(1, 1, 1)) Then
      DoCmd RunSQL "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor ID], [Factor Modi-
fier],[fldInteger]) VALUES (forms![" & CallingFormName & "]![ctlEvalIDToUse], " & thisFactorID(i) & ",1, " &
temp & ");"
      DoCmd RunSQL "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor ID], [Factor Modi-
fier],[fldInteger]) VALUES (forms![" & CallingFormName & "]![ctlEvalIDToUse], 196,1, 2);"  'Set number of Panel
Types to 2
      DoCmd RunSQL "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor ID], [Factor Modi-
fier],[fldInteger]) VALUES (forms![" & CallingFormName & "]![ctlEvalIDToUse], 197,1, 2);"  'Set number of Point
Types to 2
     Else
      DoCmd RunSQL "UPDATE tblEvaluationFactorData SET tblEvaluationFactorData.fldInteger = " & temp & "
WHERE ((tblEvaluationFactorData.[Evaluation ID]=forms![" & CallingFormName & "]![ctlEvalIDToUse]) AND
(tblEvaluationFactorData.[Factor Modifier]=1) AND (tblEvaluationFactorData.[Factor ID]=" & thisFactorID(i) &
"));"
      DoCmd RunSQL "UPDATE tblEvaluationFactorData SET tblEvaluationFactorData.fldInteger = 2 WHERE
((tblEvaluationFactorData.[Evaluation ID]=forms![" & CallingFormName & "]![ctlEvalIDToUse]) AND (tblEvalua-
tionFactorData.[Factor Modifier]=1) AND (tblEvaluationFactorData.[Factor ID]=196));"     'Set number of Panel
Types to 2
      DoCmd RunSQL "UPDATE tblEvaluationFactorData SET tblEvaluationFactorData.fldInteger = 2 WHERE
((tblEvaluationFactorData.[Evaluation ID]=forms![" & CallingFormName & "]![ctlEvalIDToUse]) AND (tblEvalua-
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
tionFactorData.[Factor Modifier]=1) AND (tblEvaluationFactorData.[Factor ID]=197));"    'Set number of Point Types
to 2
        End If
    Next i   ' Loop 1

    DoCmd DeleteObject A_QUERY, "aQryTemp"
    forms(InfoFormName).txtMessage1.caption = "Floor count completed... "
    DoCmd RepaintObject A_FORM, InfoFormName

    DoCmd SetWarnings True

    ' Hey, delete the thisEvalInputs table!
    Set MyRecords = Nothing
    DoCmd DeleteObject A_TABLE, "thisEvalInputs"
    Application.Echo True
    Exit Function

Error_EstCostCCMSI1_2_1:
    MsgBox "Oop!"
    MsgBox "Error in getting allowable factor types for this evaluation type.  Inspect tblEvaluations and tblEvalua-
tionTypeFactors.  Terminating."
    DoCmd DeleteObject A_QUERY, "aQryTemp"
    DoCmd SetWarnings True
    Exit Function

Error_EstCostCCMSI1_2_2:
    MsgBox Pass2
    MsgBox "Error in adding new data.  Check that there are no duplicated primary keys.  Otherwise, check tblE-
valuationFactorData.  Terminating."
    DoCmd DeleteObject A_QUERY, "aQryTemp"
    DoCmd SetWarnings True
    Exit Function

Error_EstCostCCMSI1_2_3:
    MsgBox "Error in adding new data.  Inputs should be integers - are they?  Terminating."
    DoCmd DeleteObject A_QUERY, "aQryTemp"
    DoCmd SetWarnings True
    Exit Function

End Function

*************************************************************************

Function EstCostCCMSI1_3 (CallingFormName, InfoFormName, InputFormName)
    Dim LoopFactorID4 As Integer
    Dim LoopFactorID3 As Integer
    Dim LoopFactorID2 As Integer
    Dim LoopFactorID1 As Integer
    Dim InfoFormCaption As String
    Dim x As Variant
    Dim thisEvalIDToUse As Integer
    Dim LoopStart4 As Integer
    Dim LoopStart3 As Integer
    Dim LoopStart2 As Integer
    Dim LoopStart1 As Integer

'   *List all the inputs here:
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    'Messages to users
    InfoFormCaption = "Point Counts:  "
    'Specify factor modifiers
    LoopFactorID4 = 1
    LoopFactorID3 = 195
    LoopFactorID2 = 196
    LoopFactorID1 = 197
    '  Specify the starting integer of the inner loop - default is "1"
    LoopStart4 = 1
    LoopStart3 = 1
    LoopStart2 = 0
    LoopStart1 = 1
    'Specify the number of factors to be input
    Pass3 = 1
    ReDim thisFactorID(Pass3)
    'Specify the individual factors to input - use exactly "Pass3" inputs, no more, no less
    thisFactorID(1) = 198

    'Do Admin
    x = InputTier1A_Admin((CallingFormName), (InfoFormName), (InfoFormCaption))
    thisEvalIDToUse = forms(CallingFormName)![ctlEvalIDToUse]

On Error GoTo 0
DoCmd SetWarnings True
Stop
    'Do it!
    x = InputTier1A(LoopFactorID4, LoopFactorID3, LoopFactorID2, LoopFactorID1, thisEvalIDToUse, (InfoForm-
Name), (InputFormName), LoopStart4, LoopStart3, LoopStart2, LoopStart1)

End Function
*****************************************************************************************
```

EVALUATION ROUTINE: EVALDELETEMECCMS1 CODE:

```
Function EstCostCCMS1 ()

    Dim x, y, z As Variant
    Dim SystemCost As Double
    Dim thisEvalIDToUse As Long
    Dim Continue As Integer
    Dim EvalIDSelected As Long
    Dim InfoFormName As String
    Dim PriorEvalID1 As Long
    Dim PriorEvalID2 As Long
    Dim PriorEvalID3 As Long
    Dim StructureID As String

'   On Error GoTo ErrorHandlerEstCostCCMS1

' Get the name of the form calling this procedure
    Set callingForm = screen.activeform
    CallingFormName = callingForm.name
    If CallingFormName = "frmDataRetrieval1" Then
       Continue = MsgBox("If you continue, previous results will be overwritten.  Continue?", 4, "WARNING")
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      If Continue = 6 Then
        GoTo EstCostCCMS1_1
      ElseIf Continue = 7 Then
        'MsgBox "OK, we're stopping"
        DoCmd SetWarnings True
        Exit Function
      Else GoTo ErrorHandlerEstCostCCMS1
      End If
    Else
    End If

' Open the information box
EstCostCCMS1_1:
  InfoFormName = "frmEvalDescription1"
  DoCmd OpenForm InfoFormName
  forms(InfoFormName).txtEvalDescrip2.caption = DLookup("[Eval Descrip 2]", "tblEvaluationTypes", "[Evaluation
Type] = 'EstCostCCMS1'")
  forms(InfoFormName).txtMessage1.caption = "Processing..."
  DoCmd RepaintObject A_FORM, InfoFormName

' Get the ID of the evaluation being run:
  EvalIDSelected = Eval("forms![" & CallingFormName & "]![ctlEvalIDToUse]")

' Get ID's of inspections, structure name
  PriorEvalID1 = DLookup("[fldLong]", "tblEvaluationFactorData", "[Evaluation ID]= " & EvalIDSelected & " and
[Factor ID]=" & 29 & " and [Factor Modifier] = " & 1384)
  StructureID = forms(CallingFormName)![ctlStructureIDToUse]

' Following calls the main subprocedures:
  SystemCost = EstCostCCMS1_M1(EvalIDSelected, CallingFormName, InfoFormName, PriorEvalID1, PriorE-
valID2, PriorEvalID3, StructureID)
  'MsgBox "EstCostCCMS1_M1 is done."

' Save the results:
  z = EstCostCCMS1_Save(EvalIDSelected, InfoFormName, PriorEvalID1, SystemCost)

' Close the information box
  DoCmd Close A_FORM, "frmEvalDescription1"

' Get out of here
Exit_EstCostCCMS1:
  DoCmd SetWarnings True
  On Error GoTo 0
  Exit Function

ErrorHandlerEstCostCCMS1:
  MsgBox "There has been a fatal error in the EstCostCCMS1 evaluation.  Terminating."
  GoTo Exit_EstCostCCMS1

End Function
*****************************************************************************

Function EstCostCCMS1_M1 (EvalIDSelected As Long, CallingFormName As String, InfoFormName As String,
PriorEvalID1 As Long, PriorEvalID2 As Long, PriorEvalID3 As Long, StructureID As String)
This procedure uses an indexed seek - much faster than find or DLookUp methods!

Dim FieldName() As String
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim NFloors, N1, N2 As Integer
Dim NPanels(), NPanels1(), NPanels2() As Integer
Dim PointCost As Double
Dim PanelCost As Double
Dim SystemCost As Double
Dim thisFM1a As Integer
Dim thisFM As Integer
Dim MyDB As Database
Dim thisEvalInputs As Recordset
Dim MyRecords As Recordset
Dim numInputs As Double
Dim Criterion As String
Dim MyIndex As Index
Dim MyField As Field
Dim MyTableDef As TableDef

'Use PriorEvalID1 to copy inspection records to temp recordset and save time
   Set MyDB = dbengine.workspaces(0).databases(0)
   MyDB.Execute ("SELECT DISTINCTROW [Evaluation ID], [Factor Modifier], [Factor ID], fldCurrency, fldDate,
fldMemo, fldByte, fldInteger, fldLong, fldSingle, fldDouble, fldOLE, fldText, fldLogical INTO thisEvalInputs FROM
tblEvaluationFactorData WHERE (([Evaluation ID]=" & PriorEvalID1 & ") );")

'Set about the business of indexing table thisEvalInputs
   Set MyTableDef = MyDB.TableDefs("thisEvalInputs")
   Set MyIndex = MyTableDef.CreateIndex("thisEvalIndex")
   Set MyField = MyIndex.CreateField("Factor ID")
   MyIndex.primary = True
   MyIndex.required = True
   MyIndex.fields.Append MyField
   Set MyField = MyIndex.CreateField("Factor Modifier")
   MyIndex.fields.Append MyField
   MyTableDef.Indexes.Append MyIndex
   Set MyTableDef = Nothing
   Set MyRecords = MyDB.OpenRecordset("thisEvalInputs")
   MyRecords.index = "thisEvalIndex"

'Count the records returned & display - a nicety
   numInputs = MyRecords.recordcount    ' How many records were found?
   DoCmd SetWarnings True
   forms(InfoFormName).txtMessage1.caption = "Retrieving data (" & numInputs + 1 & " inputs). . ."
   DoCmd RepaintObject A_FORM, InfoFormName

' Get names of fields for DLookUp statements for each factor
   ReDim FieldName(198 + 1)
   For i = 1 To 198
      On Error Resume Next
      FieldName(i) = FindFactorFieldName(i)
    Next i
    On Error GoTo 0

'Commence with the cost estimating code

   MyRecords.Seek "=", 195, 1
   NFloors = MyRecords(FieldName(195))
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    ReDim Points(NFloors + 1, 2, 2)
    ReDim Panels1(NFloors + 1)
    ReDim Panels2(NFloors + 1)
    ReDim Panels(NFloors + 1)
    For i = 1 To NFloors
      For j = 1 To 2
        For k = 1 To 2
          thisFM1a = DLookup("[Factor Modifier 1a]", "tblHierarchy1", "[Factor Modifier 1]=197 AND [Factor
Modifier 1 Value]= k AND [Factor Modifier 2]=196 AND [Factor Modifier 2 Value]=" & j & " AND [Factor Modi-
fier 3]=195 AND [Factor Modifier 3 Value]=" & i & " AND [Factor Modifier 4]=1 AND [Factor Modifier 4
Value]=1")
          thisFM = DLookup("[Factor Modifier]", "tblHierarchyA", "[Factor Modifier 1a]=" & thisFM1a & " AND
[Factor Modifier 2a]=1  AND [Factor Modifier 3a]=1 AND [Factor Modifier 4a]=1 AND [Factor Modifier 5a]=1
AND [Factor Modifier 6a]=1 AND [Factor Modifier 7a]=1 AND [Factor Modifier 8a]=1 ")
          MyRecords.Seek "=", 198, thisFM
          Points(i, j, k) = MyRecords(FieldName(198))
        Next k
      Next j
    Next i


  For i = 0 To NFloors              'Floors, from the basement to the maximum
    For j = 1 To 2                  'Panel type
      N1 = Points(i, j, 1)      'Get number of inputs per panel type per floor
      N2 = Points(i, j, 2)      'Get number of outputs per panel type per floor
      If Points(i, j, 1) > 0 Then                       'Find number of panels req'd
        If Int(Points(i, j, 1) / 64) >= .5 Then
          NPanels1(j) = NPanels(j) + Int(Points(i, j, 1) / 64)
        Else
          NPanels1(j) = NPanels(j) + Int(Points(i, j, 1) / 64) + 1
        End If
      Else
      End If
      If Points(i, j, 2) > 0 Then
        If Int(Points(i, j, 2) / 64) >= .5 Then
          NPanels2(j) = NPanels(j) + Int(Points(i, j, 2) / 8)
        Else
          NPanels2(j) = NPanels(j) + Int(Points(i, j, 2) / 8) + 1
        End If
      Else
      End If
      NPanels(j) = IMSMathMax(NPanels1(j), NPanels2(j))
      PointCost = PointCost + Points(i, j, 1) * 50 + Points(i, j, 2) * 150
    Next j
  Next i
  PanelCost = NPanels(1) * 1000 + NPanels(2) * 5000
  SystemCost = PanelCost + PointCost
  EstCostCCMS1_M1 = SystemCost

  ' Hey, delete the thisEvalInputs table!
  Set MyRecords = Nothing
  DoCmd DeleteObject A_TABLE, "thisEvalInputs"

End Function
*****************************************************************************************************
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
Function EstCostCCMS1_Save (EvalIDSelected As Long, InfoFormName As String, PriorEvalID1 As Long, SystemCost As Double)
   Dim FactorID1 As Integer
   Dim x As Variant
' Create an SQL update query to change the existing value of RSR to the new value
   DoCmd SetWarnings False
   forms![frmEvalDescription1].txtMessage1.caption = "Saving result . . ."
   DoCmd RepaintObject A_FORM, "frmEvalDescription1"
   '* Use the following for calling from anywhere
   ' Get Eval ID value
   EvalIDSelected = Eval("forms![" & CallingFormName & "]![ctlEvalIDToUse]")
   ' If user didn't use "Enter other req'd data / See results" button, then
   ' a record wasn't added to hold RSR result.  This checks for that and adds
   ' if necessary.
   FactorID1 = FindFactorID("Cost", EvalIDSelected)
   x = DLookup("[fldLong]", "tblEvaluationFactorData", "[Evaluation ID] = " & EvalIDSelected & " AND [Factor ID] = " & FactorID1)
   If IsNull(x) Then GoTo Error_EstCostCCMS11Save_1
Resume_EstCostCCMS11Save_1:
   On Error GoTo Error_EstCostCCMS11Save_2
   DoCmd RunSQL "UPDATE DISTINCTROW (tblEvaluationFactorsAllowed INNER JOIN tblEvaluationFactorData ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationFactorData.[Factor ID]) INNER JOIN tblEvaluationTypeFactors ON tblEvaluationFactorsAllowed.[Factor ID] = tblEvaluationTypeFactors.[Factor ID] SET tblEvaluationFactorData.fldDouble = " & SystemCost & " WHERE ((tblEvaluationFactorData.[Evaluation ID]=" & EvalIDSelected & ") AND (tblEvaluationFactorData.[Factor ID]=" & FactorID1 & ") AND (tblEvaluationTypeFactors.[Factor I/O]=2));"
   DoCmd SetWarnings True
   Exit Function

Error_EstCostCCMS11Save_1:
   'Adds record to hold results:
   'MsgBox "Inserting record, from error routine."
   DoCmd SetWarnings False
   DoCmd RunSQL "INSERT INTO tblEvaluationFactorData ([Evaluation ID], [Factor ID], [fldLong]) VALUES (" & EvalIDSelected & ", " & FactorID1 & ", ""0"");"
   x = DLookup("[fldLong]", "tblEvaluationFactorData", "[Evaluation ID] = " & EvalIDSelected & " AND [Factor ID] = " & FactorID1)
   GoTo Resume_EstCostCCMS11Save_1

Error_EstCostCCMS11Save_2:
   MsgBox "EstCostCCMS11Save update statement failed.  Terminating."
   Exit Function

End Function
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.