NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

REENGINEERING THE UNITED STATES MARINE CORPS' RECRUIT DISTRIBUTION MODEL (RDM)

by

Kevin J. Snoap

September 1998

Thesis Advisor:	Hemant K. Bhargava
Associate Advisor:	Suresh Sridhar

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1998	Master's Thesis

4. TITLE AND SUBTITLE : REENGINEERING THE UNITED STATES MARINE CORPS' RECRUIT DISTRIBUTION MODEL (RDM)

5. FUNDING NUMBERS

6. AUTHOR(S)
Kevin J. Snoap

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The United States Marine Corps accomplishes its mission "to put the right Marine in the right place at the right time with the right skills and quality of life" in a variety of ways. One of the information systems assisting the Marine Enlisted Assignments branch is the Recruit Distribution Model (RDM). This thesis proposes changes to the RDM user interface, data management, assignment model, and analysis capability. With the use of business process reengineering, process modeling, mathematical modeling, and database design a fully functional prototype has been developed to address each identified change proposal. This reengineered system includes numerous innovations such as an intuitive navigational scheme using switchboards, and the elimination of manual data entry for data already available in the system. It also provides a number of significant contributions beneficial to the USMC. For instance, the reengineered system allows the user to objectively analyze different results by comparing four different objective measures, and its mathematical model uses commercial-off-the-shelf products eliminating a proprietary solver. All these changes will empower managers to effectively and efficiently manage the assignment of recruits in order to meet the challenges of the 21st century.

14. SUBJECT TERMS	15. NUMBER OF PAGES
USMC, Databases, Manpower Assignment, Models, Decision Support Systems, Graphical User Interface	100
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

REENGINEERING THE UNITED STATES MARINE CORPS' RECRUIT DISTRIBUTION MODEL (RDM)

Kevin J. Snoap
Lieutenant, United States Navy
B.S., University of Texas at Austin, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1998

Author: Kevin J. Snoap
Kevin J. Snoap

Approved by: H.K. Bhargava
Hemant K. Bhargava, Advisor

Suresh Sridhar, Associate Advisor

Rueben T. Harris, Chairman, Department of Systems Management

ABSTRACT

The United States Marine Corps accomplishes its mission "to put the right Marine in the right place at the right time with the right skills and quality of life" in a variety of ways. One of the information systems assisting the Marine Enlisted Assignments branch is the Recruit Distribution Model (RDM). This thesis proposes changes to the RDM user interface, data management, assignment model, and analysis capability. With the use of business process reengineering, process modeling, mathematical modeling, and database design a fully functional prototype has been developed to address each identified change proposal. This reengineered system includes numerous innovations such as an intuitive navigational scheme using switchboards, and the elimination of manual data entry for data already available in the system. It also provides a number of significant contributions beneficial to the USMC. For instance, the reengineered system allows the user to objectively analyze different results by comparing four different objective measures, and its mathematical model uses commercial-off-the-shelf products eliminating a proprietary solver. All these changes will empower managers to effectively and efficiently manage the assignment of recruits in order to meet the challenges of the 21st century.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. THESIS PURPOSE

The main purpose of this thesis was to reengineer a United States Marine Corps' Manpower Assignment model concerned with the distribution of recruits to schools. This model is called the Recruit Distribution Model (RDM) [Ref. 1]. Throughout this thesis, the RDM is addressed as either RDM or the old system.

Additionally, an important purpose of this thesis was to build a functional prototype of the reengineered RDM. The new system is called the Recruit Distribution Decision Support System (RDdss). It demonstrates the functionality of the reengineered RDM. Throughout this thesis, the RDdss is addressed as either RDdss or the new system.

The majority of this thesis is devoted to a discussion of the RDdss. As necessary, the RDM is discussed. The following section is a general discussion designed to set the stage for understanding the rest of the thesis. It provides a problem description of recruit distribution in the United States Marine Corps (USMC). The last section of this chapter discusses the significant contributions of this thesis.

B. GENERAL PROBLEM DESCRIPTION

Recruit distribution in the USMC is the process that assigns recruits to an entry level school (ELS) leading to a military occupational specialty (See Figure 1). These assignments are made about 48 times a year, during the last week of Marine Corps Recruit Depot (MCRD) training. In this ending period of the MCRD, the recruits are facing the "Crucible," which is the final wicket a recruit endures before officially becoming a Marine. Consequently, the use of the titles recruit and Marine are used interchangably in this paper.

For at least two reasons, this assignment process is a critical manpower function. First, a Marine's military occupational specialty (MOS) ultimately determines

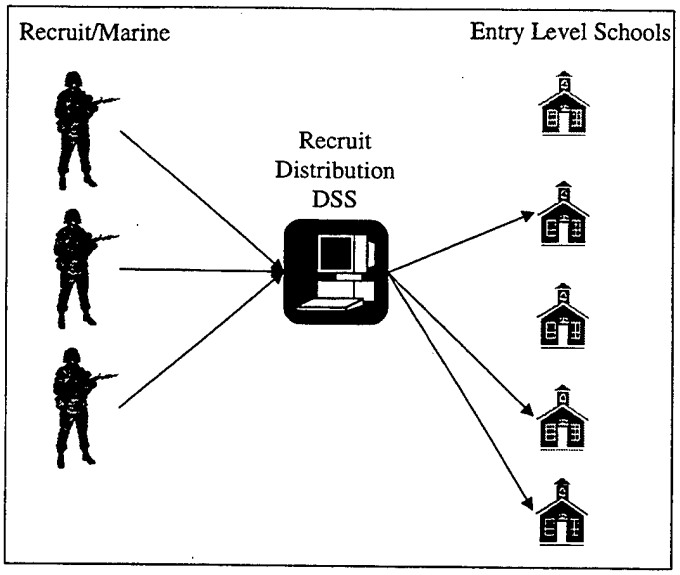


Figure 1. Assigning Marines to Schools

the member's career. Therefore, it is in the best interest of both the Marine and the USMC that a school assignment matching the Marine's desire is made. Second, the success obtained by a service member during his or her time in the USMC is partially based upon successful completion of their ELS, where a pattern of success is established. Therefore, a school assignment maximizing the chances of the Marine completing their training is important. So, fulfilling the Marine's desire and matching him or her to an MOS (i.e. ELS) that fits their personal characteristics is critical to the overall health of the USMC, making the assignment process a critical manpower function [Ref. 2].

Fulfilling the desire of the Marine is accomplished through the use of a contract guarantee. This is called a program enlisted for (PEF), and is specified during the recruiting process. For instance, a PEF = 19 is the "Tank and Assault Amphibian Option." There are currently two ELS's associated with this, "M1A1 Tank Crewman" and "Assault Amphibian Crewman." So, a Marine who chose the PEF = 19 is a possible candidate for these two schools and no others.

Once the Marine's school options are known, a Marine-to-school fit is deter-

mined for each of these schools (See Chapter III, Section B for details). This fitness determination is partly made by looking at each ELS's minimum eligibility requirements, which are call mandatory properties. This term "property" is used as defined by Webster's Ninth New Collegiate Dictionary: "a quality or trait belonging and especially peculiar to an individual or thing." Examples are Age > 18, Clerical >= 80, and Electrical >= 95. The meaning of the first example is obvious. The other two are based on test scores from the Armed Services Vocational Aptitude Battery (ASVAB) test.

In addition to the mandatory properties, most schools also specify desirable properties. A desired property is the same as a mandatory property, except they are not prerequisites for attending the school. For example, the Traffic Management Coordination school desires Marines with a Clerical score of at least 100. So, a desired property of Clerical >= 100 is specified for this ELS.

By using the information obtained from the PEF, mandatory and desired properties, a fitness matrix is generated. This shows the fit of every Marine to every school he or she is eligible for. Since there are about 100 schools and on average 700 Marines considered for every run, this matrix has the potential of 70,000 matches.

However, the matrix size is actually bigger. This is because each of the schools is broken down by classes. Some ELS's have a class starting each week, others every month, and others every quarter. Over a given year, these classes total about 1,800. Following the practice of the USMC, only the classes over the next 3-4 months are considered during the assignment process (see Figure 2). Therefore, the fitness matrix is increased to a potential size of 350,000.

The fact that the school classes start at different times throughout the year and that the assignment process is only conducted 48 times a year (normally this occurs on friday) causes the "3-month look ahead" in the model and has implications important to the USMC. The concern is school seats which never get filled. Each seat is prepaid, guaranteeing its availability to the USMC. This means every vacant

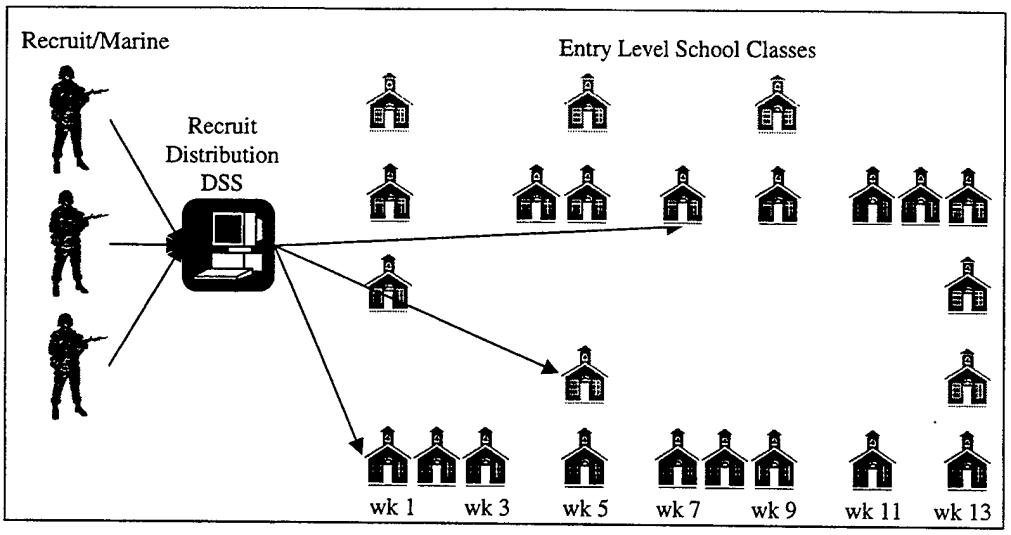


Figure 2. Assigning Marines to School Classes

spot is potentially a lost resource to the USMC. Therefore, in addition to the Marine-to-school fitness concern, there is also the concern of filling all available school seats before their report date is passed.

Another concern is the problem of unassigned Marines. There are numerous reasons why they may not get assigned. Maybe the only school class the Marine was eligible for is already full. Or, possibly he or she is not qualified for any of the school's promised by their PEF. A third possibility is errors in the data provided to the assignment process. Regardless of the cause, it requires identification and corrective action.

Finally, a discussion of the internal and external stakeholders associated with the USMC recruit distribution is given. Internally, there are the USMC and the recruits. Since their concerns were discussed above, they are not given any further consideration. Externally, there is the contractor Decision Support Associates, Inc. (DSAI) who has maintained and upgraded the RDM for over 30 years. They also maintain about eight other major systems for the USMC. For their services, the USMC pays a significant amount of money each year. Additionally, DSAI has proprietary software in some of these systems, which has locked-in the USMC to this

company. Besides the contractor, there is one last significant stakeholder, the American taxpayer. In a time of increasing fiscal constraints and shrinking budgets, it is imperative that wise decisions are made in regards to assigning Marines to ELS's. The American taxpayers expect nothing less.

C. SIGNIFICANT CONTRIBUTIONS

Four significant contributions beneficial to the USMC have been made in this thesis. A detailed discussion of these is given throughout the next three chapters. Here, we list and summarize the contibutions.

- Analysis and articulation of the recruit distribution process - by interviewing USMC and DSAI personnel, reviewing available documentation, and operating the RDM we were able to articulate an understanding of the recruit distribution process using IDEF process modeling. Additional data modeling was articulated in a third normal form relational schema [Ref. 3].
- Development of a mathematical model - by analyzing the assignment process, criteria, and constraints, USMC policies and objectives, a mathematical model for the new system was developed.
- Fully functional prototype - an intuitive and easy to understand new system that seamlessly interfaces with the old system was built. It provides an objective means of comparing assignment results of both systems, and was developed using commercial-off-the-shelf (COTS) software applications.
- Elimination of the proprietary solver and associated contractor lock-in - this was accomplished by replacing the proprietary solver with two COTS applications.

II. REENGINEERING MOTIVATIONS AND PROCESS VIEW

The RDM has a number of limitations which motivated the USMC to reengineer it. We start this chapter by listing and describing each identified limitation. Then, to develop a better understanding of recruit distribution in the USMC, two different process views are examined. The first is concerned with the RDM operating environment, and the second is concerned with the first level of the RDdss IDEF0 model.

A. LIMITATIONS OF OLD SYSTEM

Throughout this reengineering effort, a number of limitations of the old system have become apparent. Many of these limits were recognized earlier, and were part of the USMC's motivation to reengineer the RDM. The following is a list of the identified limitations of the old system.

- Navigation
- Data management
- Assignment procedure
- Transaction processing approach

A description of each limitation is now given.

1. Navigation

Navigating through the RDM is not intuitive. It is neither obvious where one should start or where one should go. Navigation is accomplished by initially selecting an option from one of the main display's drop down menus. This normally results in a window appearing on the computer's desktop. Then, by pointing and clicking on the displayed window's buttons, further navigation is accomplished. After working with the RDM for a number of hours, we were able to navigate through the application

to find and display specific windows. However, after a week or two of not using the system, we had difficulty finding our way around again. In the RDdss, we have created an intuitive navigational scheme using switchboards.

2. Data Management

Data management in the RDM is poor, leading to the introduction of numerous errors. The biggest problem in this regard deals with data entry. The RDM violates the basic rule of never requiring the user to enter data already in the system [Ref. 4]. For example, creating a new school in the RDM requires the user to type data into seven different data fields. Only the course identification number has been automated by a drop down list. In the RDdss, when creating a new school we have reduced the manual data entry to one field.

3. Assignment Procedure

There are at least five limitations associated with the old system's assignment procedure. It

- is encoded into proprietary software,
- examines schools sequentially, rather than globally,
- makes assignments based on school priority rather than weights,
- attempts to maximize fill rather than fit-and-fill, and
- is relatively inflexible.

The solver performing these assignments was designed back in the 1950's, where the major concern was speed and using the minimal amount of memory. It is written in Fortran, and is proprietary code owned by DSAI. It does not search repetitively for an optimal solution by trying to maximize or minimize an objective function. Instead, it maximizes the fill of prioritized school seats. In the RDdss, we use a well know algorithm called CPlex [Ref. 5]. After conducting around 600 iterations of comparing 344 Marines to 576 school classes, it produces an *optimal* solution. Its

objective function was written to make the assignment procedure flexible. It allows the RDdss manager to "game" the system by making fit-and-fill trade-offs, until a "good" solution is found.

4. Transaction Processing Approach

Finally, the old system follows a transaction processing approach, vice a decision support approach [Ref. 6]. This is partially due to the inflexibility of the solver. However, another contributor is the extreme difficulty in comparing one run to another. Other than providing a means for manually computing the numerical difference in Marines assigned, the RDM provides little insight or information views for comparing runs. This is because there is no way to objectively compare one run to another. As it is now, if the RDM manager launches the assignment model and everybody is assigned, the result becomes the approved assignments. In the RDdss, we have created an entire process devoted to providing insightful analysis of a given run (See Chapter IV, Section D for details). Its purpose is to support the making of wise assignment decisions.

B. A PROCESS VIEW

To develop a better understanding of recruit distribution in the USMC, two different process views are now examined. The first is concerned with the current operating environment of the old system. The goal is to develop a big picture view of this process. For those interested in further study of the old system, Appendix A contains the RDM business process IDEF0 model.

The second process view examined concerns the first level of the RDdss business process IDEF0 model. The goal is to develop a better understanding of the new system, without going into great detail. This lays a good foundation for the RDdss discussions in the remainder of this thesis. Appendix B contains the entire IDEF0 model of the new system.

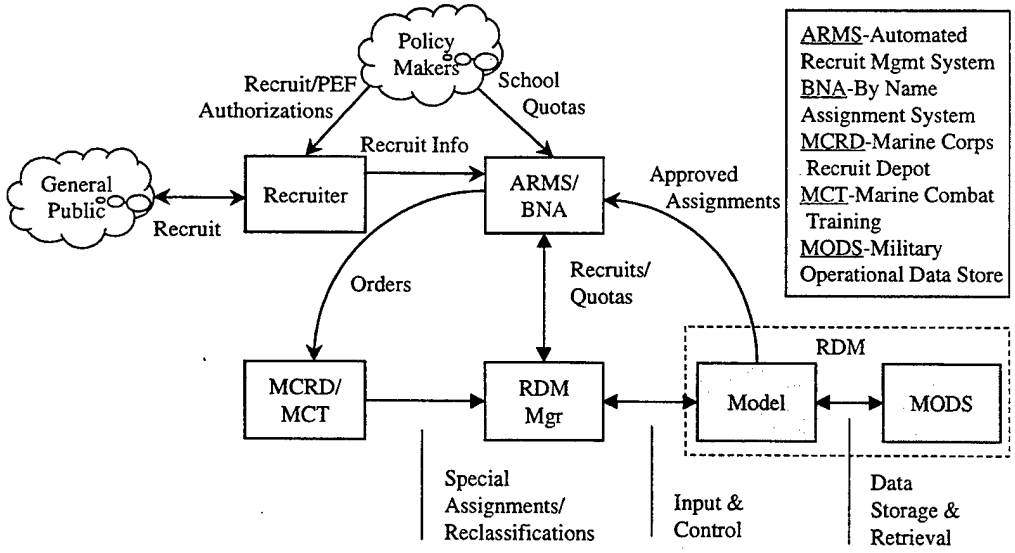


Figure 3. Current Recruit Distribution Operating Environment

1. Recruit Distribution Model Operating Environment

Figure 3 graphically depicts the operating environment of the USMC recruit distribution. At the top are the policy makers, who forecast and decide how many recruits and corresponding school seats are needed over the next few years. In addition to this, they determine how many PEF's or school guarantees to make available for a given year. Currently, about 65-70% of the recruits enter the USMC with a quarantee. The recruit and PEF authorizations for the following year are given to the USMC recruiters, and the school seats or quotas determined for the following years is input into the By Name Assignment (BNA) system.

The recruiters use the recruit and PEF authorizations in recruiting from the general public. Once a potential candidate is found they take the ASVAB, if they have not already done so. Based on the results, the recruiter can offer different quarantees. Once the candidate signs a contract with the USMC, their test scores, PEF, and other personal information such as age and height are entered into the Automated Recruit Management System (ARMS).

Both the ARMS and BNA systems utilize large main frame databases. They

serve as central repositories for maintaining data on many aspects dealing with the USMC. Keeping this information stored in one location ensures everyone is using the same data, providing consistency for all users. One user is the RDM manager, who retrieves recruit and school information from these systems for use in the RDM.

Additionally, the RDM receives data from two other sources. The MCRD instructors provide special assignment inputs. These are personnel identified as having the talents or abilities well suited for a particular school. The other data comes from the MCT. They provide the RDM manager with reclassification information. For instance, a Marine is reclassified if he or she is injured during MCT and is unable to make the start date of their assigned ELS.

All this recruit and school data is input into the RDM, where it is stored in the Military Operational Data Store (MODS). The model is then run. Once a satisfactory set of assignments is obtained, the RDM manager uploads the approved assignments to the ARMS and BNA systems. From this assignment information, the MCRD generates orders for the Marines graduating from the MCRD.

2. New System Level 0 Diagram

Focus is now shifted to the examination of the first process level of the RDdss. Looking any deeper will provide more detail than is necessary at this point. As mentioned earlier, the intent is to lay the foundation for the RDdss discussions in the remainder of this thesis.

Figure 4 shows the level 0, or context diagram of the RDdss. The program used to generate this illustration was BPWin, which is based on IDEF0 modeling. Consequently, all arrows entering from the left are considered "inputs," arrows entering from the top are "controls," arrows exiting to the right are "outputs," and arrows entering from the bottom are "mechanisms." A convenient acronym for this is ICOM [Ref. 7].

- I = Input: something consumed in the process
- C = Control: a constraint on the operation of the process

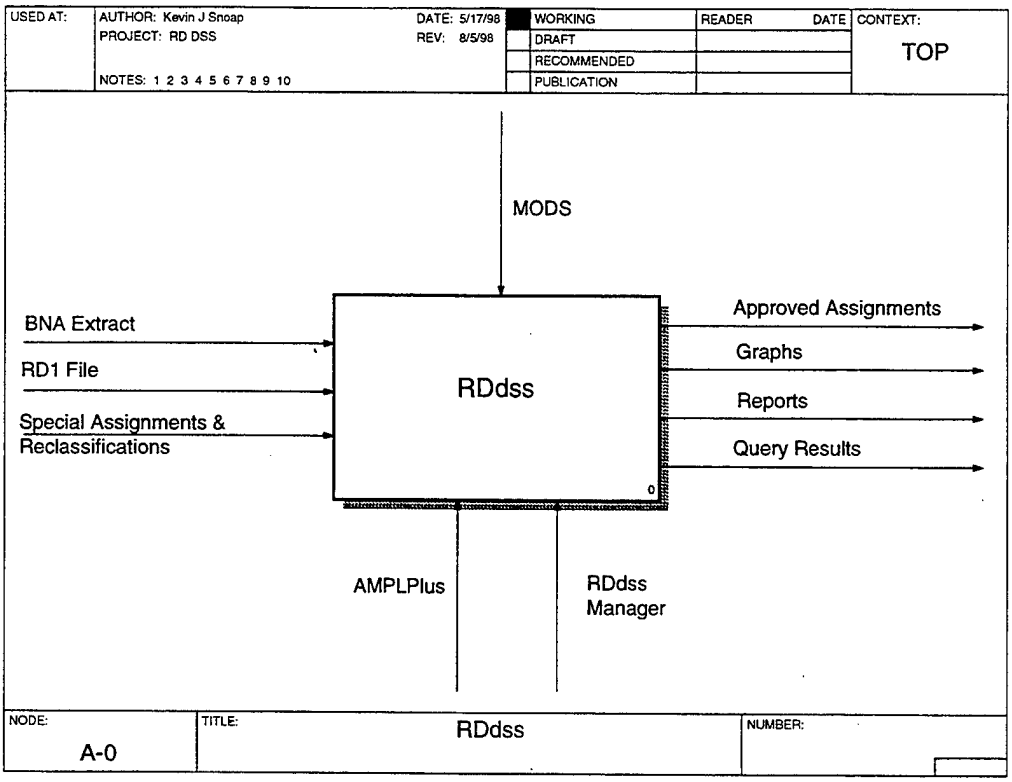


Figure 4. RDdss Process, Level 0 Diagram

- O = Output: something resulting from the process
- M = Mechanism: something used to perform the process, but is not itself consumed

There are three inputs to the RDdss, the BNA extract, the RD1 file, and the special assignments/reclassifications. The first of these contains a set of school class quotas covering a 120 day period. This comes to the RDdss manager as a fixed-width delimited text file. The RD1 file contains the data for the recruits soon to graduate from the MCRD. It also is a fixed-width delimited text file. The special assignments and reclassifications are the recruits identified with special abilities or talents and the Marines requiring reassignment to a different school class, respectively. This last data comes to the RDdss manager as a memo and not a text file.

The only control is the database (or Military Operational Data Store) used to store the data for the RDdss. Among other things, the MODS contains the informa-

tion specifying the eligibility requirements for each of the schools. This is why it is considered a control. It contains the data controlling or constraining the assignment of Marines to schools.

Besides the approved assignments that were mentioned earlier, the output consists of graphs, reports, and query results. Each graph was designed to provide insight into the current assignment result, plus provide a means for comparing different runs. The reports provide a print-out of information the RDdss manager might find useful. For instance, an approved assignment report and an unassigned Marine report are both available. The query results provide different views of the data in the MODS. For instance, one of the queries provides a listing of the fitness scores of all schools a Marine is eligible for.

Finally, the two mechanisms of the RDdss are the RDdss manager and the commercial-off-the-shelf (COTS) application AMPLPlus. Both of these entites are necessary to make the system work, and are not consumed by the process itself. This is why they are considered mechanisms. The RDdss manager is the one who points and clicks on the buttons of the RDdss, making it operate. AMPLPlus is the application containing the mathematical model used in making optimal Marine-to-school assignments. It interfaces with the CPlex algorithm that actually performs the optimization.

III. MODELING THE RECRUIT DISTRIBUTION PROBLEM

We model the recruit distribution problem as an assignment problem. Various submodels are developed to compute input paramters and perform other preprocessing steps for the assignment model, which optimally assigns recruits to a school class. Following is a discussion of important points concerning the recruit distribution problem.

1. Schools and Classes: there are about 130 schools per fiscal year, which are broken down by classes (about 1800 classes per fiscal year). Classes of the same school, commencing on different dates, are identical. However, a class commencing soon after a recruit is available for training has greater utility than another class starting at a later date. On average, there are 600 classes in a run.
2. Program Enlisted For (PEF)
   (a) 65-70% of enlistees enter the USMC under a guarantee (or PEF). The remaining recruits enter under an open contract (PEF=00).
   (b) A recruit can be assigned only to schools associated with his or her PEF. Some PEF's are associated with one school, while others are associated with many schools.

A. OPTIMIZATION OBJECTIVES: FIT AND FILL

Our approach in the assignment model is to optimize by looking at both fit and fill. The importance of both these objectives was discussed in Chapter II, Section B. The idea is to allow the RDdss manager to make a tradeoff between fit and fill.

The high-level objective function is:

- Maximize

$$K_1 \cdot Fitness - K_2 \cdot Penalty$$

Where K_1 is the fitness coefficient and K_2 is the fill coefficient.

By assigning different values to these coefficients, the RDdss manager is able to make trade-offs between fit and fill. This capability makes the assignment model flexible. It provides the RDdss manager a means for "gaming" the system.

B. MEASURING FITNESS

The first half of the high-level objective function deals with maximizing the fitness of Marines to schools. Therefore, we cover this topic now. Following is a discussion of important points concerning the measuring of fitness.

1. Marines are only eligible for assignment to schools corresponding to their PEF guarantees.

2. Associated with each school are specific mandatory properties, or minimum eligibility requirments (e.g. AGE $>$ 18).

3. Eligibilty of a Marine to a school is pre-determined by comparing a Marine's attributes (e.g. Age = 20) to the school's mandatory properties. A Marine is eligible only if he or she meets all the minimum school requirements.

4. Many schools also have desirable properties associated with them. This is a property which is desired in a Marine (e.g. Height $>$ 65 inches), but are not prerequisties for attending the school. Since some desirable properties are more desirable than others, a means of distinquishing the properties is necessary. We have followed the Marine Corps practice of using 6 levels of desirability. Level 1 properties are the most desired, followed by level 2, etc.

5. Let $\mathcal{P}(s)$ denote the set of desirable properties for school s, and let $level(p)$ denote the desirability level of property p. Let $possesses_{(m,p)} = 1$ if Marine m possesses property p (0 otherwise). Further, let $Score_{level(p)}$ (> 0) be the score assigned for possessing a $level(p)$ property, where $Score_{level(p)}$ is calculated using an exponential function inversely proportional to $level(p)$. The scores for levels 1 through 6 are shown in Figure 5.

6. The total fitness score $(fit_{m,s})$ of Marine m to school s is composed of two parts: the score $(ManScore_{m,s})$ for possessing mandatory properties, and the score $(DesScore_{m,s})$ for possessing some or all of the desirable properties.

   Our procedure for computing fitness scores is designed in a manner that, for each school, the average fitness score—with the average computed over all qualified Marines, i.e. Marines who have met the mandatory properties—is constant (i.e. 100). The reason we did this was to ensure each school

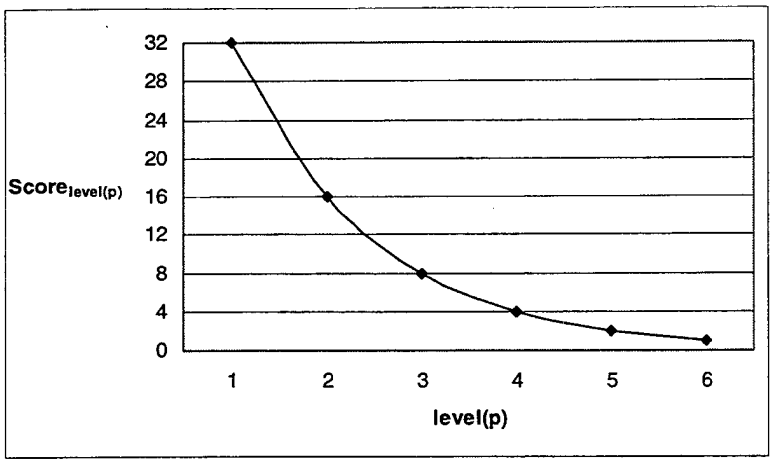


Figure 5. Exponential Function for Calculating $Score_{level(p)}$

received equal treatment in the model, regardless of the number of desirable properties specified for each school. Otherwise, all other things being equal, the school with the greatest number of desirable properties would receive the most assignments.

7. For each school s, percentage weights $ManWt_s$ and $DesWt_s$ are assigned, respectively, for the mandatory and desirable properties. For example, for school s_1, a weight of 70% may be assigned to the mandatory properties and 30% for the desirable properties.

8. A Marine possessing all mandatory properties for school s is given an initial score $ManScore_{m,s} = ManWt_s \cdot 100$. Marines who do not possess all mandatory properties for a school are given an overall fitness score of 0 for that school.

9. Of all the Marines possessing the mandatory properties, those possessing desirable properties are awarded additional points, which are weighted by the level of the property. The Marine's score for desirable properties is computed as follows:

   - The absolute score $Abs_{m,s}$, for Marine m and school s, based on desirable properties is

$$Abs_{m,s} = \sum_{p \in \mathcal{P}(s)} Score_{level(p)} \cdot possesses(m,p)$$

   The average of these absolute scores is computed over all qualified Marines. Let $AveAbs_s$ denote the average absolute score for school s.

- Each Marine's fitness score for desirable properties is then computed as a fraction of this average, and normalized by multiplying with the percentage weight for desirable properties. That is, $DesScore_{m,s}$ is a weighted relative score:

$$DesScore_{m,s} = (\frac{Abs_{m,s}}{AveAbs_s}) \cdot DesWt_s \cdot 100$$

The overall average of all these scores is, due to the above construction, $DesWt_s \cdot 100$.

10. The final fitness score, $fit_{m,s}$ is simply the sum $ManScore_{m,s} + DesScore_{m,s}$. It may be seen that, for each school, this number averages (over all the qualified Marines for that school) to 100.

11. As mentioned earlier, schools are broken down by classes that are identical, except for their start dates. Therefore, let $fit_{m,s} = fit_{m,c}$ for each class c of school s.

C. MEASURING FILL

The second half of the objective function deals with maximizing the fill of school seats. We now turn our attention to this topic.

As we discussed in Chapter II, school seats are paid for in advance. This is to guarantee the seat is available to the USMC. Consequently, every vacant spot is potentially a lost resource. Since the model is run about once every week, the biggest concern is the unfilled seats having report dates within the next seven days. Conversely, a seat having a report date in three months is not so critical.

To capture the essence of filling school seats with early report dates, we use a penalty function. The idea is simple. School seats having an early report date get a high penalty, and those having a late report date get a low penalty. A number of functions would have worked for this. We have graphed two candidates in Figure 6. The first is a linear function and the second a large-step function. Each of these was tried in the prototype. In our opinion the latter one is the best choice. It gives a lot of emphasis to the first week, plus it treats all schools having report dates within the same week equally. It is the function currently implemented in the rddss.

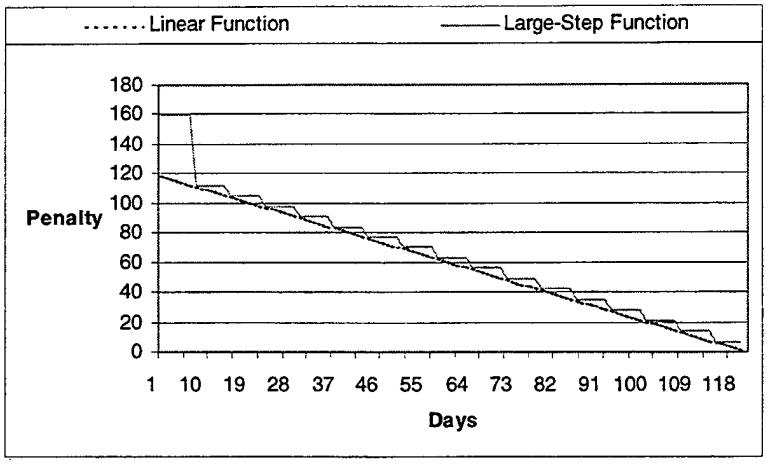


Figure 6. Two Candidate Penalty Functions

D. ASSIGNMENT MODEL

We are now ready to formally describe the assignment model. We will first specify the model's assumptions. Then, we will list the model's notation, followed by its objective function and constraints.

1. Assumptions

1. Each school is unique and identifies a single course of instruction leading to an MOS.

2. In any given run, a school may offer several classes that start on different dates. It is more desirable to fill seats in classes starting earlier. Such classes have a higher penalty per unit shortfall than similar classes starting at a later date.

3. All Marines available for assignment are graduating from the MCRD on the same date.

4. A Marine's eligibility for a school, as well as the fit for each school, is determined by the model preprocessor, taking into account the PEF guarantees, and the mandatory and desirable properties.

5. A Marine is assigned to a school corresponding to their PEF code, or not at all.

6. It is better to leave a Marine unassigned than to overfill a school.

7. The demand for the number of Marines to be trained in each school class is determined by looking at the BNA extract. This demand statement is already constrained by the capacity constraint at each school.

2. Notation

- Sets

  $\mathcal{M}$: Marines

  $\mathcal{C}$: Classes

- Exogenous Variables (Parameters in AMPL)

  $fit_{m,c}$: the desirability of assigning Marine m to class c (note: the fitness score is zero for Marine-class pairs where either the Marine does not meet the classes mandatory properties, or where the class does not fulfill the Marine's PEF guarantee.)

  $demand_c$: demand for Marines to be trained at class c

  $penalty_c$: penalty for each unit of demand not met (higher value means it is more critical to fill the class)

- Decision Variables

  $x_{m,c}$ (binary integer): 1, if Marine is assigned to the given class; 0 otherwise

3. Objective

Maximize the Total Utility: (Total Reward - Total Penalty)

$$TU = k_1 \left(\sum_{c \in \mathcal{C}} \sum_{m \in \mathcal{M}} fit_{m,c} \cdot x_{m,c} \right) - k_2 \left(\sum_{c \in \mathcal{C}} penalty_c \cdot (demand_c - \sum_{m \in \mathcal{M}} x_{m,c}) \right) \quad \text{(III.1)}$$

4. Constraints

- assignmentLimit: a Marine is assignable to at most one school

$$\sum_{c \in \mathcal{C}} x_{m,c} \leq 1 \qquad \forall m \qquad \text{(III.2)}$$

- eligibility: a Marine is only assignable to a class they are fit for (this prevents assigning Marines with a fitness score of zero)

$$x_{m,c} \leq fit_{m,c} \qquad \forall m \; \forall c \qquad \text{(III.3)}$$

- capacity: since penalties apply only if there is a positive shortfall, the objective function would be non-linear. To avoid that, we assume we will never oversupply Marines to schools, resulting in:

$$\sum_{m \in \mathcal{M}} x_{m,c} \leq demand_c \qquad \forall c \tag{III.4}$$

IV. A DECISION SUPPORT SYSTEM FOR RECRUIT DISTRIBUTION

A. ARCHITECTURE AND IMPLEMENTATION

Following is a discussion of the RDdss architecture and its implementation. First, the architectural components are discussed. Then, the steps taken to build the RDdss are described.

1. Architecture

The architecture of the RDdss is depicted in Figure 7. This illustration shows how the six major components of the system are related. We will describe this architecture by examining each component in the following order: switchboard, relational database, preprocessor, assignment model, solver, and analyzer.

The switchboard is one of two components providing an interface between the user and the RDdss. It is the mechanism by which the user controls the operation of the system. Figure 8 illustrates the main switchboard of the RDdss. It is displayed

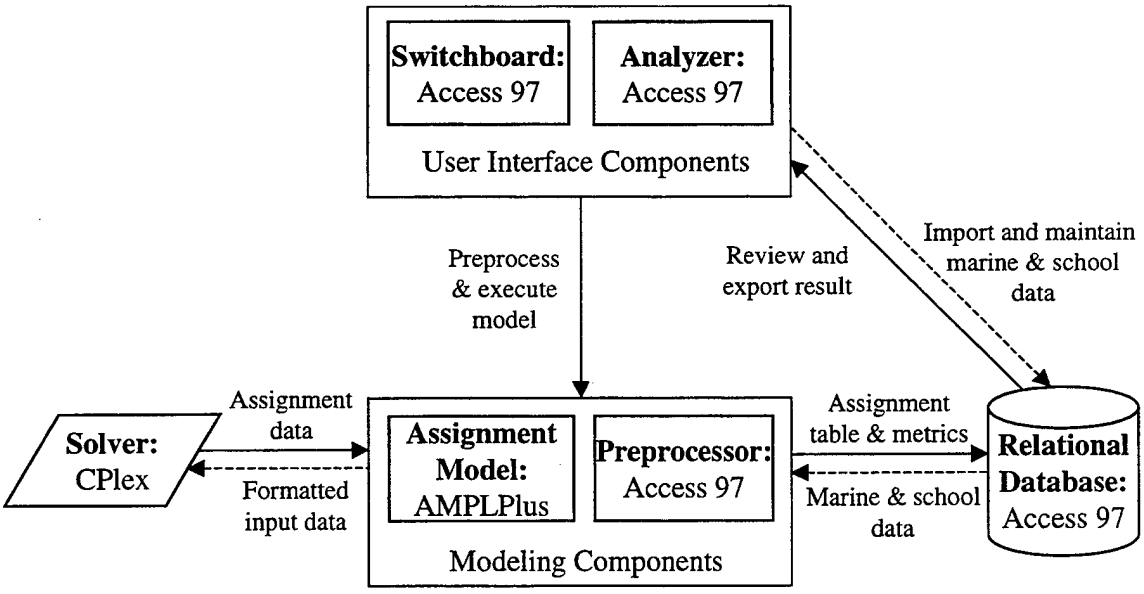


Figure 7. RDdss Architecture

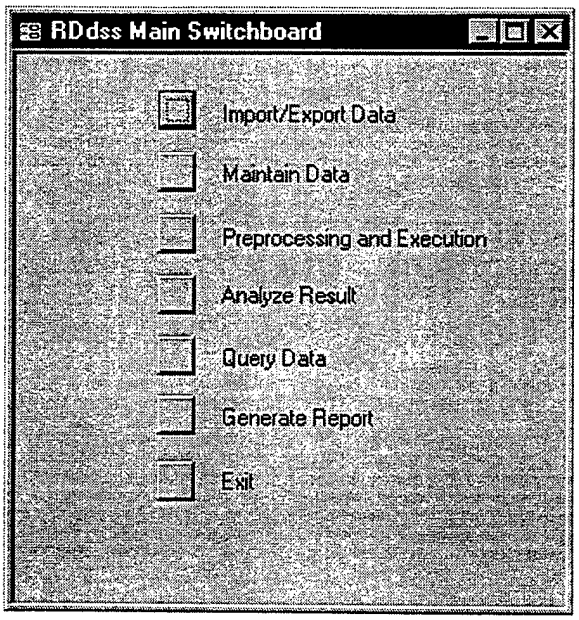


Figure 8: RDdss Main Switchboard

when the application is started. As this illustration depicts, the user can perform the following functions by simply pointing and clicking on the appropriate button: import and export data, maintain data, preprocessing and execution of the assignment model, analyze results, query data, and generate reports. All the RDdss switchboards were generated using Access 97.

The relational database technology was used in the RDdss [Ref. 8]. The actual table relationships are shown in Figure 9. These tables were developed in Access 97, which uses the relational technology. In addition to a few flat files used to communicate with the assignment model, all the data for the RDdss is stored in this database.

The preprocessor is one of the two modeling components. It consists of Access 97 Visual Basic for Applications (VBA) code that computes information necessary for the assignment model [Ref. 9]. It determines the demand of each class, the appropriate penalty associated with each class, and the Marine-to-class fitness matrix. The code for these assignment model inputs is found in Appendix D, lines 452-516 and 4305-4812.

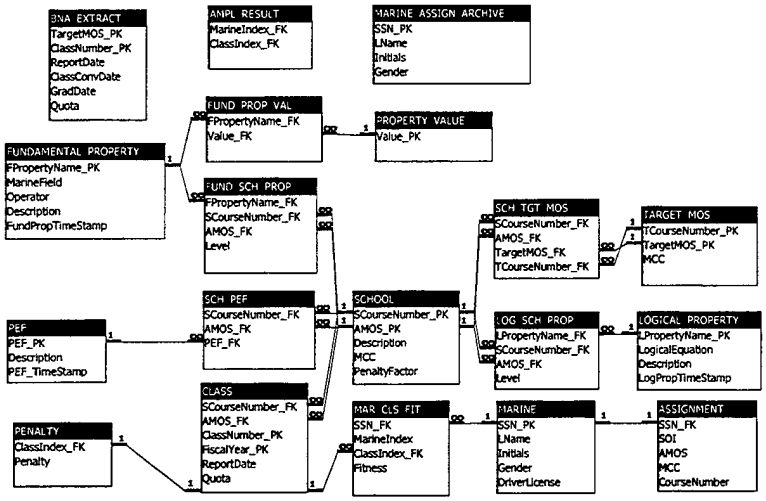


Figure 9. RDdss Relational Schema

The assignment model is the other component of the modeling components. It has been modeled in AMPL [Ref. 10], a language and a computing environment for expressing, solving, and analyzing mathematical programming problems (minimizing or maximizing a function of decision variables, subject to constraints on the variables). A complete discussion of the assignment model's math model is available in Chapter III, Section D.

The next component is the solver. It communicates directly with AMPL, the modeling language of AMPLPlus. As mentioned in Chapter II, it uses an algorithm called CPlex, which is an optimization package for linear, network and integer programming. Working in conjunction with AMPLPlus, the solver finds an optimal solution.

The final component is the analyzer. It is the other component making up the user interface. This is where the RDdss manager analyzes assignment results, seeking insights for developing a "good" solution. Insight is developed mostly by the graphs and numerical information available in the analyzer. These were generated in Access 97 using its report generator and VBA code. The programming used in the analyzer is shown in Appendix D and includes lines 1-310, 2156-2214, and 3756-3827.

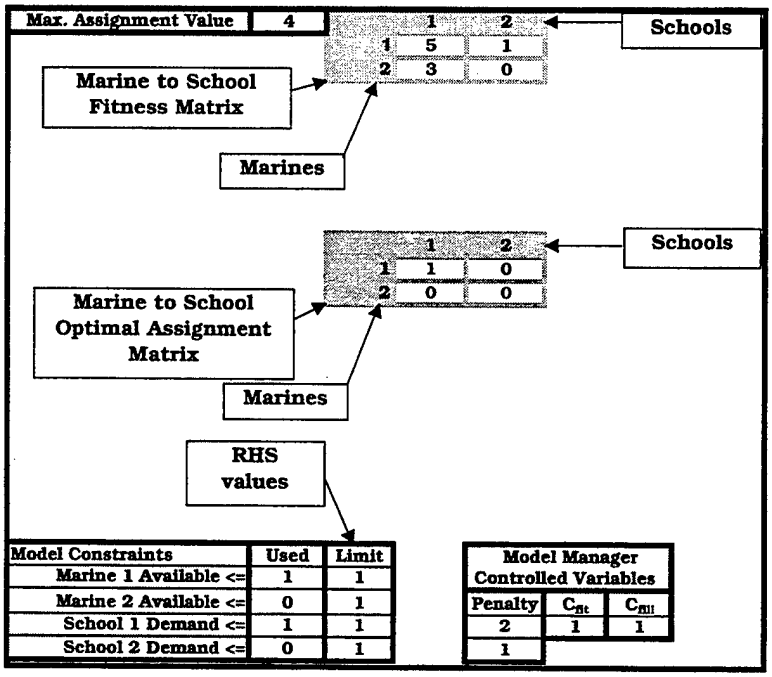


Figure 10. RDdss Demonstration Solver

2. Implementation

Building the RDdss was an iterative process. Some steps took numerous iterations, and some were straightforward. Below is a listing of the 14 major steps it took to build the RDdss.

1. Analyzing the current system was the first step. This included 1) interviewing USMC and DSAI personnel, 2) reviewing available documentation, and 3) operating the RDM.

2. After gathering the current system data, it was necessary to develop a logical understanding of the system. This was accomplished by modeling it with BPWin. Appendix B is the result of this step.

3. With this system understanding, it was possible to envision how the new system would make assignments. Using standard optimization techniques [Ref. 11], a small-scale demonstration solver was built using Excel 97. It is shown in Figure 10.

4. Combining the system understanding and the data requirements for the demonstration solver, it was possible to envision a data structure for the reengineered system. This lead to the development of a third normal form relational schema for the RDdss [Ref. 3].

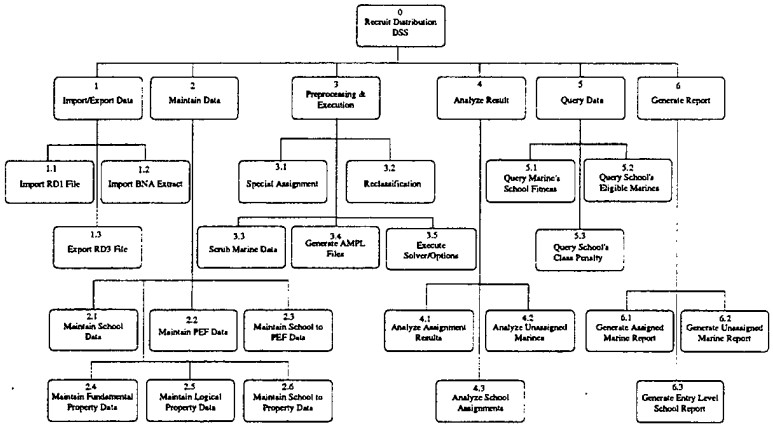


Figure 11. RDdss Decomposition Diagram

5. Using the relational schema as a model, the RDdss tables were built in Access 97. The results of this were seen earlier in Figure 9.

6. As the vision for the reengineered system came more into focus, an intuitive navigational scheme was conceived. This lead to the development of a decomposition diagram for the RDdss. It is demonstrated in Figure 11.

7. Using the decomposition diagram as a model, the user interface was built for the RDdss in Access 97. The Main Switchboard was seen earlier in Figure 8.

8. With the switchboards now in place, VBA coding was commenced [Ref. 9]. This made each switchboard functional and useful.

9. Building upon the concept of the demonstration solver from step 3, a formal mathematical assignment model for the RDdss was developed. This was introduced in Chapter III, Section D.

10. The COTS applications, AMPLPlus and CPlex were installed on the personal computer containing the RDdss.

11. With AMPLPlus now installed, it was possible to code the assignment model into the application. This made it possible to verify our design.

12. Manually creating input files for AMPLPlus is time consuming. So, the preprocessor was coded. This automated the generation of the class demand, class penalty, and Marine-to-class fitness matrix.

13. Further automation of the RDdss was accomplished by interfacing Access 97 and AMPLPlus. This required both VBA and AMPL coding. The end result was a seamless operation of the two applications.

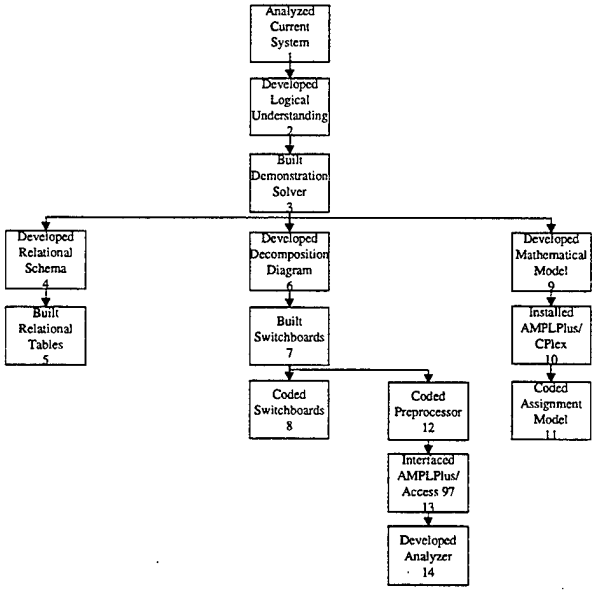


Figure 12. Actual RDdss Build Sequence

14. Finally, much thought was given to ways of gaining insight into the assignment results. This lead to the development of the analyzer, which was discussed briefly in the previous section.

Some of these steps were performed in parallel. The actual order followed in building the RDdss are depicted in Figure 12.

B. USING THE RECRUIT DISTRIBUTION DECISION SUPPORT SYSTEM

In this section, we will discuss using the RDdss. Three aspects are covered, 1) setting up a run, 2) model execution, and 3) customizing a run and results. Each is discussed in the order given here.

1. Setting up a Run

Setting up a run of the RDdss is straightforward and easy to do. Figure 13 graphically illustrates the steps involved. Starting with step 1 on the graph and working through to step 3d(iii), we will discuss a run set-up.

The RDdss manager starts at the top of the Main Switchboard. Here, he or

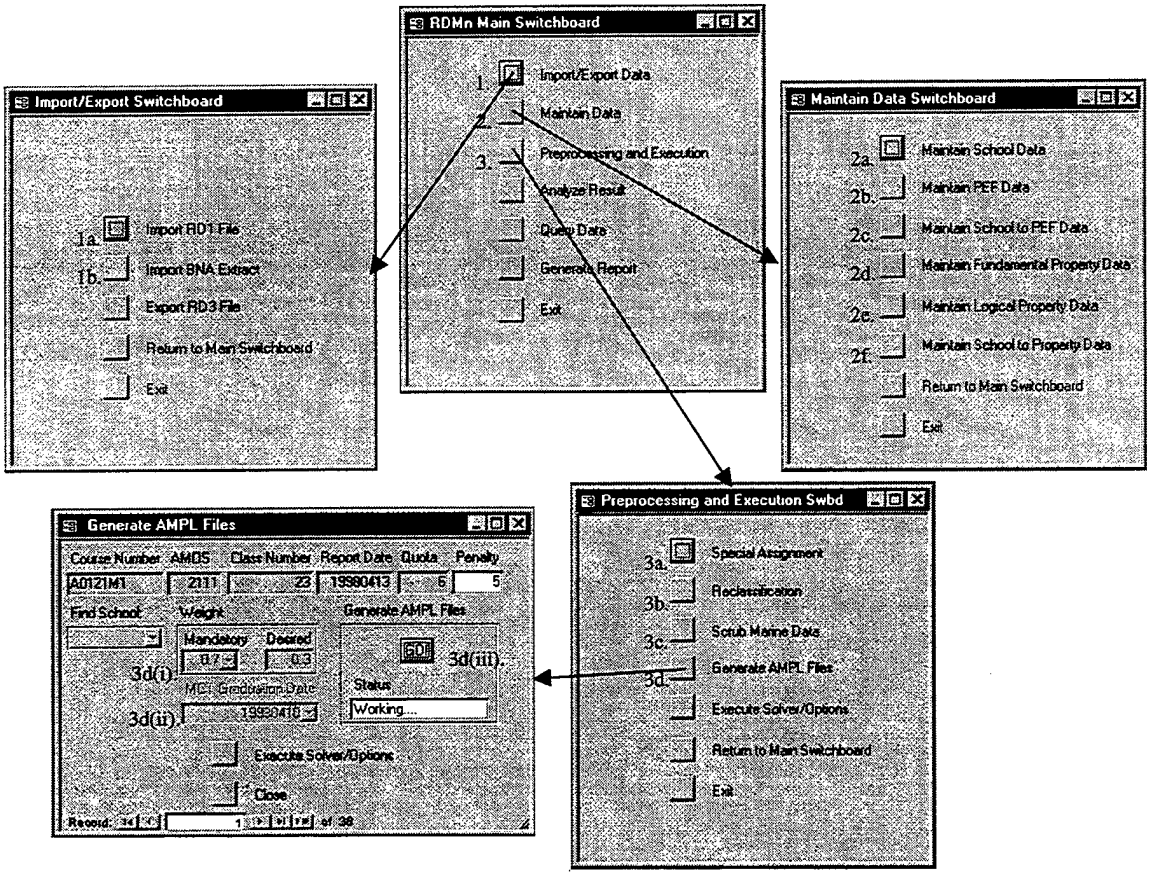


Figure 13. Setting up a Model Run

she clicks on the Import/Export Data button. This will cause the Main Switchboard to close and the Import/Export Switchboard to open. From this switchboard, the RDdss manager will import the RD1 file and the BNA extract into the RDdss MODS. This is easily accomplished by clicking on the corresponding buttons for importing these files. Finally, the RDdss manager will click on the "Return to Main Switchboard" button, causing the Import/Export Switchboard to close and the Main Switchboard to reopen.

The next step is data maintenance. This includes updating, editing, and/or deleting of school related information. The RDdss manager gets to the Maintain Data Switchboard by clicking on the appropriate button on the Main Switchboard.

From the Maintain Data Switchboard, the RDdss manager will start at the

top and work his or her way down to the bottom. The various buttons on this switchboard open forms where data maintenance is performed. For instance, the first button opens the Maintain School Data form, allowing the RDdss manager to create, edit, and/or delete a school. The other buttons allow similar operations with the PEFs, and fundamental and logical properties. After the RDdss manager has finished maintaining the data, he or she returns to the Main Switchboard by clicking on the "Return to Main Switchboard" button.

After data maintenance, the RDdss manager is ready to perform data preprocessing in prepartion for a model run. By clicking on the Preprocessing and Execution button, the Main Switchboard closes and the Preprocessing and Execution Switchboard opens. From here, as on all other switchboards, the RDdss manager starts at the top and works down.

The first three preprocessing buttons directly effect the generation of the class demand, class penalty, and Marine-to-class fitness matrix. This is why they come before the fourth button, Generate AMPL Files. Preprocessing in the correct order will reduce the amount of time spent setting up the run, by minimizing the need to generate a new class demand, class penalty, and Marine-to-class fitness matrix.

The first preprocessing button concerns special assignments. Specially assigning a recruit affects the AMPL input in two ways. First, the respective member is not included in the Marine-to-class fitness matrix, since they already have an assignment. Second, the class demand for the class the recruit is assigned to must decrease by one. Otherwise, it may become overfilled.

The second preprocessing button concerns reclassification of a Marine. Reclassified Marines affect the AMPL input in two ways. First, the class demand of the class they are no longer assigned to must increase by one. This provides an opportunity to fill the empty seat during a subsequent run. Second, the class demand of the class the Marine has been reassigned to must decrease by one. Otherwise, it may become overfilled.

The third preprocessing button concerns data scrubbing, which also affects the AMPL input. It is an attempt to identify and correct errors relating to the Marines. An example is a Marine who has been given an unknown PEF. Since it is not identified by the RDdss, the member will receive a zero fitness score for every class. This will prevent the Marine from getting an assignment.

Following the completion of special assignments, reclassifications and data scrubbing, the RDdss manager is ready to generate the AMPL input files. After going to the Generate AMPL Files form, the first concern is specifying the weights for the mandatory and desired properties. This is specified by the RDdss manager at step 3d(i). What this does is indicate how important the mandatory properties are with respect to the desired properties. For instance, if the RDdss manager decided to give a weight of 1.0 to the Mandatory selection, the Desired selection would automatically fill in with 0.0. This would indicate the desired properties are of no consequence. The default for these two selections is 0.7 for Mandatory, and 0.3 for Desired.

Next, at step 3d(ii), the RDdss manager selects the MCT graduation date desired. This is selected from the drop down list. Then, the RDdss manager pushes the "GO!" button. The system will now generate the AMPL input files, completing the run set-up.

2. Model Execution

Execution of the assignment model takes place after the run set-up has been completed. Figure 14 graphically illustrates the steps taken to execute the model. A discussion of this is now given.

From the Main Switchboard, the RDdss manager points and clicks on the Preprocessing and Execution button. This takes him or her to the corresponding switchboard. From here, the Execute Solver/Options selection is made. This opens up the form where the model is executed from.

The Execute Solver/Options form provides the RDdss manager the ability to set the fit and fill coefficients for the objective function. The default values are 1 for

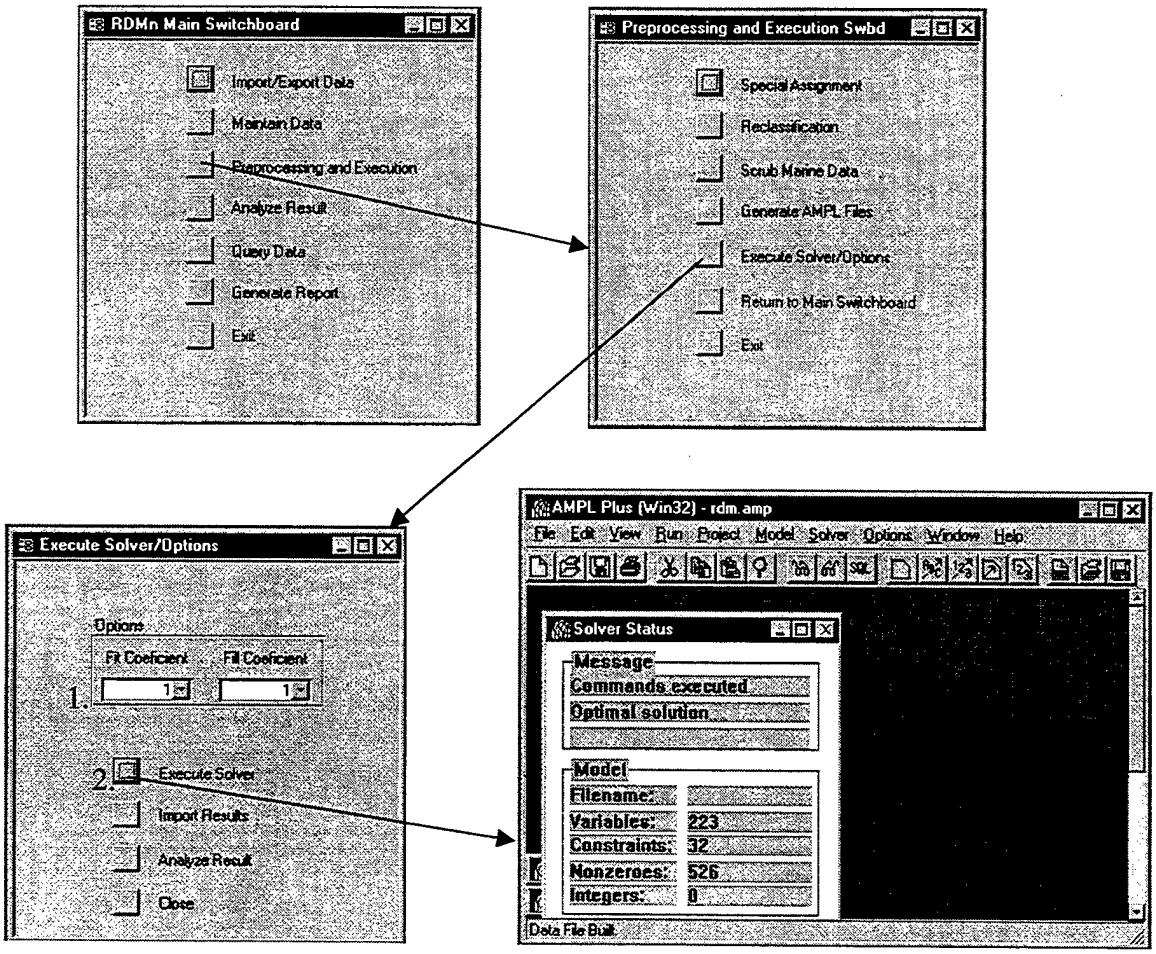


Figure 14. Model Execution

each coefficient. The RDdss manager can change either of these with a whole number from 0 - 100. The significance of these numbers is discussed in detail later in this chapter. Once the desired coefficients have been chosen, the Execute Solver button is pushed.

By selecting the Execute Solver button, AMPLPlus is launched. This application will run without user interaction. First, it will import the RDdss mathematical assignment model. Next, the specified fit and fill coefficients are retrieved. Finally, the preprocessed data generated during the run set-up are imported. Using this input data, AMPLPlus works in conjunction with the solver and generates an optimal assignment solution.

With the generation of an optimal assignment solution, the results are sent to an output file called "rdm.out." AMPLPlus, having completed its task, now closes. This indicates an optimal solution was found and is available for importing. So, the RDdss manager now imports the results by pushing the Import Results button on the Execute Solver/Option form.

3. Customizing a Run and Results

As mentioned in the previous subsection, the RDdss manager can set the fit and fill coefficients of the objective function. This is a key capability of the new system. It is the means by which the RDdss manager can customize a run and "game" the RDdss. This "gaming" of the new system is essential for determining a "good" assignment result.

For instance, if the fit coefficient is set to 0, and the fill coefficient is set to a number > 0, say 1, the assignment model will maximize the fill of schools having early report dates. This has both advantages and disadvantages. One advantage is the average wait time for Marines to attend a school is minimized. Another is that the number of unfilled school seats, having a report date between the current and next scheduled model run, is minimized. However, a disadvantage is the average Marine-to-school fitness score is guaranteed to be the lowest score of all runs.

Conversely, if the fit coefficient is set to a number > 0, say 1, and the fill coefficient is set to 0, the assignment model will maximize the average Marine-to-school fitness. An advantage here is the increased chance of each Marine successfully completing their ELS, establishing a pattern of success early in theri career. However, one disadvantage is the wait time for a Marine to attend a school is guaranteed to be the longest. Also, the number of unfilled school seats, having a report date between the current and next scheduled model run, is guaranteed to be the greatest.

Neither of the above solutions is ideal. A better solution is some where between these two extremes. By changing the fit and fill coefficients of the objective function, the RDdss manager has a means for customizing each run to ensure a "good"

assignment result is found.

C. INNOVATIONS IN USER INTERACTION

A number of user interaction innovations have been made in the RDdss. Two of them have already been discussed. These were the intuitive navigational scheme using the switchboards, and the elimination of manually entering data already available in the system. One more note worthy innovation is now introduced. It is a date change innovation.

The date change innovation concerns changing the graduation date of the Marines in the downloaded RD1 file. This is necessary for generating correct assignment results. The date in the downloaded RD1 file is based on graduation from the MCRD. However, the model needs to know when the Marines are available for assignment. Availability is based on the MCT graduation date.

The old system solution to this problem took three steps. First, the RDM manager used a calendar to determine the MCT graduation date. This is normally 27 days following completion of MCRD, and is always on a friday. So, the RDM manager will find the MCRD graduation date on the calendar and count 27 days. The friday nearest to this day is the day desired. Second, this MCT graduation date, along with the RD1 file, is sent to another branch in the USMC where the date change is made. Third, the file is returned to the RDM manager, who imports the file into the RDM.

The reengineered system has streamlined this process. Within the RDdss, the RDdss manager makes the date change. It is accomplished with the push of a couple buttons.

Figure 15 shows how these changes are made. First, the RDdss manager selects the MCRD graduation date requiring a change. Once this is selected, a calendar appears. Its purpose is to help the RDdss manager correctly identify the MCT graduation date. Next, the "=" button is pushed. Its purpose is to add the number of

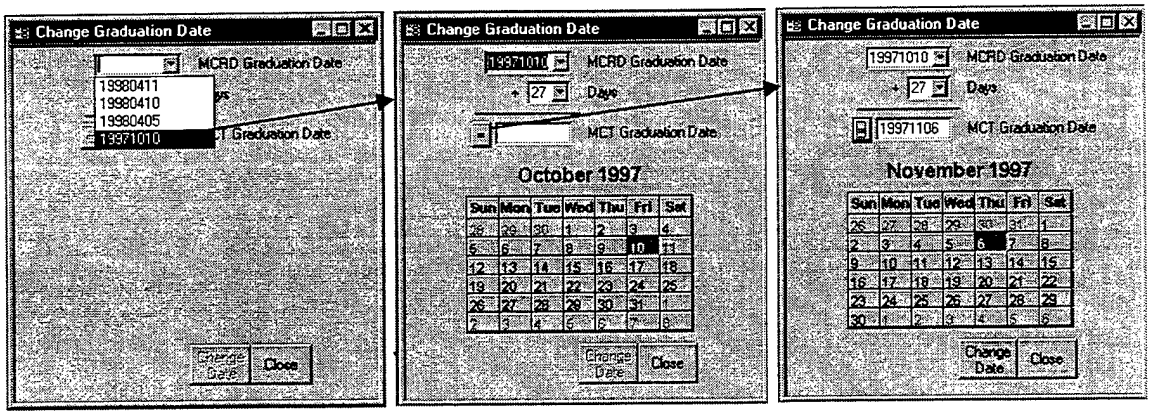


Figure 15. Changing a Graduation Date

days specified in the "Days" field (default is 27), to the MCRD graduation date. This updates the calender as shown on the far right. When the correct date is found, the "Change Date" button is pushed. This makes the desired change to the data in the RDdss database.

D. INNOVATIONS IN ANALYZING MODEL RESULTS

As discussed earlier, the RDdss manager has the capability of "gaming" the system by using different fit and fill coefficients. However, having this ability is not enough. The RDdss manager still needs a means to compare one run against another.

It is hard, if not impossible, to estimate the quality of a solution simply by looking at a list of assignments. The actual assignment results depend on the values of the fit and fill coefficients, which cannot be set at the ideal levels without understanding the impact on solution quality. So, analyzing run results to get insight and develop a "good" solution is now discussed.

Four measures are considered particularly useful in evaluating alternative solutions.

1. Fitness premium. The Marine-to-school fitness function is defined in a way that for each school, the average fitness over all Marines *eligible* for that school is 100. So, for any school, if the average fitness computed over Marines *assigned*

to that school is greater than (or less than) 100, the difference represents a positive (or negative) premium.

2. Unfilled seats in first week. Since the model is run nearly every week, seats left unfilled in classes starting in the first week will remain unfilled. This represents wasted resources.

3. Average wait time for Marines. Experience has shown that the longer a Marine waits before attending their ELS, the greater the chance of disciplinary problems. Therefore, it is generally better to keep this wait period short.

4. Unassigned Marines. A purpose of the system is to assign each Marine to a school. Therefore, knowing the number of unassigned Marines is an important measure.

To provide the RDdss manager insight into the assignment results, these measures have been incorporated into output representations, including summary statistics and detailed graphs. Each output provides both numerical and visual information. The numbers are used for quantitative comparisons. The graphical information provides a big picture view for use in a qualitative comparison.

As an example, consider the two fitness graphs in Figure 16. Both of these were produced using an RD1 file of 344 Marines and an BNA extract with school data covering a 120 day period. Below each graph is amplifying and numerical information. The bar chart on the left has a fit coefficient of 0 and a fill coefficient of 1. Additionally, its fitness premium is 28 (=128-100). The bar chart on the right has a coefficient of 1 for both fit and fill. Its fitness premium is 38 (=138-100). The fit numbers are quantitatively comparable. The fitness premium for the run on the right is 36% (=100% x [38-28]/28) better than the run on the left.

The above numerical analysis is also supported by visually comparing the bar charts. Notice the large number of blips falling below the fitness score of 100 on the left graph. Each blip represents a Marine, and there are actually 44 of them. Conversely, notice the comparatively small number of blips below 100 on the right graph. There are 21 of them. The visual comparison of the two graphs reveals the

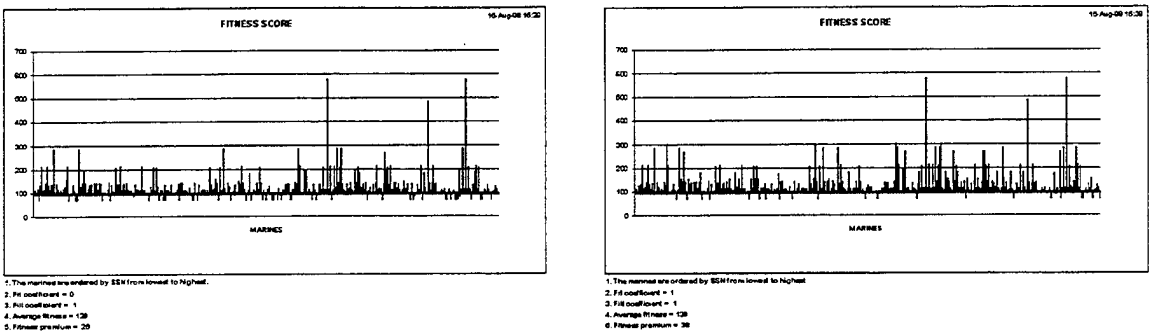


Figure 16. Two alternative results: Fit

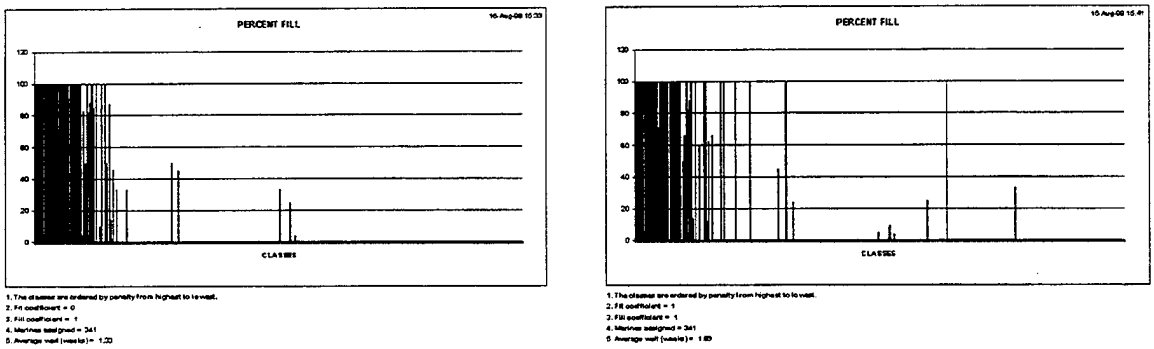


Figure 17. Two alternative results: Fill

qualitative difference. The graph on the right indicates a higher quality Marine-to-school fit. Of course, this supports the numerical comparison performed earlier.

As another example, consider the two fill graphs in Figure 17. These are based on the two model runs just described. In other words, the left graph has a fit coefficient of 0 and a fill coefficient of 1, and the right graph has a coefficient of 1 for both fit and fill.

Visually, these two graphs display an obvious difference. To appreciate how they are different requires an understanding of the significance of the blips. As note 1 of each graph indicates, "The classes are ordered by penalty from highest to lowest." Recall from Chapter III, this means the ordering is by class report date, starting with the earliest date. In other words, the bar chart on the left indicates almost every Marine will start school within the first three weeks. As discussed earlier, this is desirable. The bar chart on the right has its blips more spread out. So, by visually

comparing the two graphs, the left one has a higher quality with respect to minimizing wait time.

This is easily validated by the numbers given in note 5 of each graph. The average wait of a Marine from the left bar chart is 1.33 weeks, and the average wait from the other bar chart is 1.69 weeks. Now we quantitatively compare these values. The results from the left graph are 27% (=100% x [1.69-1.33]/1.33) better than the right.

In the following table, the results from three runs are given. The measures on the left are the four measures described earlier. Each run used the same 344 Marines and 120 days worth of school classes. The information for the first and third runs is from the two model runs discussed above. The second run was determined by "gaming" the system. In other words, we tried numerous combinations of the fit and fill coefficients until a "good" solution was found. Run 2 was the result. Understand though, we are not saying "best" solution. Only the USMC can determine what is best for them.

	Run 1 (fit=0; fill=1)	Run 2 (fit=1; fill=11)	Run 3 (fit=1; fill=1)
Unfilled Seats (1st week)	23	23	23
Unassigned Marines	3	3	3
Average Wait (weeks)	1.33	1.33	1.69
Fitness Premium	28	31	38

The reason we feel Run 2 is a "good" solution is based on a quantitative comparison of the measures and one additional piece of information. It is the fact the USMC feels it is more important to minimize wait time than to maximize the Marine-to-school fit. The remainder of this section explains how we decided Run 2 was a "good" solution.

Starting with the values for unfilled seats, it is obvious all three runs are equal. The same is true for the unassigned Marines measure. So, these two measures provide no insight for comparison purposes.

Concerning the average wait, the values of Runs 1 and 2 are equal. Further, their 1.33 value is numerically smaller than the 1.69 value of Run 3. This is good for Runs 1 and 2. As was proven earlier, a value of 1.33 is 27% better than 1.69.

Finally, there is the fitness premium measure. The first observation made is that Run 1 is a candidate for elimination. The reason is as follows. Run 2 has the same score as Run 1 for the first three measures in the table, but it shows an 11% (=100%x[31-28]/28) increase in the fitness premium. So, Run 1 is eliminated from the competition.

This leaves the last two runs. Run 3 shows a 23% (=100%x[38-31]/31) increase in fitness premium over Run 2. This creates a conflict. Run 3 has a 27% increase in average wait time, which is bad. Conversly, it has a 23% increase in fitness premium, which is good. To decide which solution is better, more information is necessary. This comes from the fact mentioned earlier, that the USMC feels it is more important to minimize wait time than to maximize the Marine-to-school fit.

We can now conclude Run 2 is a better solution than Run 3. Here is why. Run 2 minimizes average wait by 27% by giving up a 23% increase in Marine-to-school fit. This agrees with the USMC desire to more strongly emphasize a minimum waiting time. So, Run 2 is a better solution than Run 3. Since it also satisfies the known desires of the USMC, we feel it is a "good" solution as well.

E. OBJECTIVE COMPARISON OF OLD AND NEW SYSTEM SOLUTIONS

In the previous section, we described how a set of solutions generated by the RDdss are compared to determine a "good" solution. Using the same approach, we now demonstrate how to objectively compare the results from the old system with the new system. This provides the USMC a means of determining which system's solution is better, if either. Comparison of the new and old system results has been made easy by automating the importing of the RDM result into the RDdss.

We have summarized the important data for the old and new systems in the following table, based on the same 344 Marines and 120 days of school classes discussed earlier. Since the assignment algorithm for the RDM was designed to maximize fill, a run from the RDdss based strictly on fill has also been included. It is the run with a fit coefficient of 0, and a fill coefficient of 1. The last column, labeled "Run 2," is a "good" solution we found by gaming the RDdss.

	Run 1 (fit=0; fill=1)	RDM	Run 2 (fit=1; fill=10)
Unfilled Seats (1st week)	23	23	23
Unassigned Marines	3	3	3
Average Wait (weeks)	1.33	1.34	1.34
Fitness Premium	28	29	32

We start by objectively comparing the RDdss fill-based result with the RDM result. They are labeled "Run 1" and "RDM," respectively. As we shall prove, the RDdss has done a better job at filling early school seats than the RDM.

Of the four measures, only one provides any useful insight for this comparison. It is the average wait measure. The reason the other three do not apply is as follows. The first two measures have the same values, so they provide no insight. The last measure is not too meaningful, since Run 1 is strictly based on maximizing fill. In other words, it disregards fit because its fit coefficient is set to 0. This leaves the average wait measure.

A quantitative comparison of the average wait shows the RDdss solution is 1% (100%x[1.34-1.33]/1.33) better than the RDM solution. This is explainable by considering the approach used in each model. The RDM assignment algorithm maximizes the fill of early school seats by looking at each school in a sequential fashion. In contrast, the RDdss conducts a global comparison of Marines and schools. In other words, it goes through numerous iterations until it finds an optimal fill solution. This show that the RDM assignment algorithm is only closely approximating the optimized assignment result.

Finally, we "gamed" the new system until a "good" solution was found. This is Run 2. It has a fit coefficient of 1 and a fill coefficient of 10.

The objective comparison of the RDM and Run 2 solutions is straigtforward. Since the first three measures are the same for both results, they provide no insights. This leaves only the fitness premium measure. A quantitative comparison of this measure shows Run 2 has a 10% (100%x[32-29]/29) better solution than RDM. This shows the benefit of "gaming" the RDdss.

F. SUMMARY

This relatively long chapter covered some important points related specifically to the RDdss. The architectural components of the new system were discussed first, followed by the steps taken to actually build this DSS. Then, from an operational point of view, using the RDdss was discussed. This included setting up a run, executing the model, and customizing a run for determining a "good" solution. Next, two innovations were discussed in great detail. These included the date change innovation and the analyzing model results innovation. Finally, a means of objectively comparing the solution from the old and new systems was discussed.

We truly believe the innovations and capabilites introduced by this prototype can greatly benefit the USMC's Marine Enlisted Assignments branch. It can help the USMC to more effectively accomplish its mission as stated by the Commandant of the Marine Corp, "to put the right Marine in the right place at the right time with the right skills and quality of life."

V. CONCLUSION

This concluding chapter has three short sections, followed by a final comment. In the first section, we reiterate the significant contributions made in the thesis. This is followed by a brief description of some lessons we learned. Then, some recommended improvements for the prototype are given. Following this last section, we comment on how this thesis work might benefit the other branches of the United States military.

A. SIGNIFICANT CONTRIBUTIONS

Four significant contributions beneficial to the USMC were made in this thesis. These were discussed in detail in the previous three chapters. A list and summary of the contributions is given here.

- Analysis and articulation of the recruit distribution process - by interviewing USMC and DSAI personnel, reviewing available documentation, and operating the RDM we were able to articulate an understanding of the recruit distribution process using IDEF process modeling. Additional data modeling was articulated in a third normal form relational schema.
- Development of a mathematical model - by analyzing the assignment process, criteria, and constraints, USMC policies and objectives, a mathematical model for the new system was developed.
- Fully functional prototype - an intuitive and easy to understand new system that seamlessly interfaces with the old system was built. It provides an objective means of comparing assignment results of both systems, and was developed using COTS software applications.
- Elimination of the proprietary solver and associated contractor lock-in - this was accomplished by replacing the proprietary solver with two COTS applications.

B. LESSONS LEARNED

Our original goal was to build an As-Is IDEF0 model of the RDM, about two months after starting the thesis. This worked out fine. The next goal was to take

another six weeks to develop a To-Be IDEF0 model [Ref. 1]. This did not work out, for two reasons. First, our understanding of the entire recruit distribution process was still not mature enough to build the To-Be model. Nearly half a year was spent just thinking about its many details. Second, it is nearly impossible to envision the new system without first knowing the capabilities and limitations of the application used to build the prototype. One extreme envisions the impossible, while the other barely touches the reengineered system's potential. *The lesson learned from this was that properly reengineering a process requires two things. A detailed understanding of the current process environment, and a good working knowledge of the capabilities and limitations of the application used to build the prototype.*

Getting feedback from the Marine Enlisted Assignments branch was not always easy. Correspondence normally occured by e-mail. There was an inverse relationship between the number of questions asked and the number of answers given. As the one increased, the other decreased. *So, another lesson learned was that it is best to limit each correspondance to a question or two.*

While writing code, it is best to simultaneously document the code's purpose. Going back and trying to provide documentation at the end is too frustrating. This requires a considerable amount of discipline, but is well worth the effort. *The lesson learned here is document as you code.*

C. PROTOTYPE IMPROVEMENTS

A big part of this thesis involved the building and refinement of the prototype. The following topics are areas where the RDdss could use improvements.

1. Speed Improvements

a. Preprocessing Data

Of all the steps necessary to complete a run of the RDdss, the preprocessing step is the most time consuming. For the 344 Marines and approximately 500 school classes analyzed in this thesis, it takes about 30 minutes to preprocess

this data. This is about ten times longer than the assignment model takes to find an optimal solution. As the size of the preprocessing data is doubled, so is the time needed to preprocess it. Finding a way to speed up this process is desirable. One possibility is to find a way to perform all the preprocessing in main memory, with only two accesses to the hard drive. The first access is to get the necessary Marine and school data. The second is to write the preprocessed results back to the hard drive.

b. Analyzing Result

Compared to all other forms and switchboards in the RDdss, opening the "analyze results" forms is the most time intensive. With the 344 Marines and approximately 500 school classes, it takes just under a minute for any of these three to open. This feels like an eternity when you are in a hurry. The reason it takes so long is the creation of temporary tables, numerous calculations and comparisons, and retrieving of data from flat files. Finding a way to reduce this time is desirable. One possibility is to create another form which would contain only the four objective measures. Currently, the RDdss manager must go to two different forms for this information. Each containing information and displays in addition to the objective measures, which slow them down.

2. Multiple Solution Storage

Currently, the prototype will only save the results of one solution. This means the RDdss manager must either print-out hard copies of the graphs, or manually write down each solution's objective measures for comparison with other runs. An improvement here should allow the storage of at least three solutions. This would permit the RDdss manager to easily compare each solution. He or she could then eliminate one or two undesirable solutions, while continuing to search for a "good" one.

3. Administration Switchboard

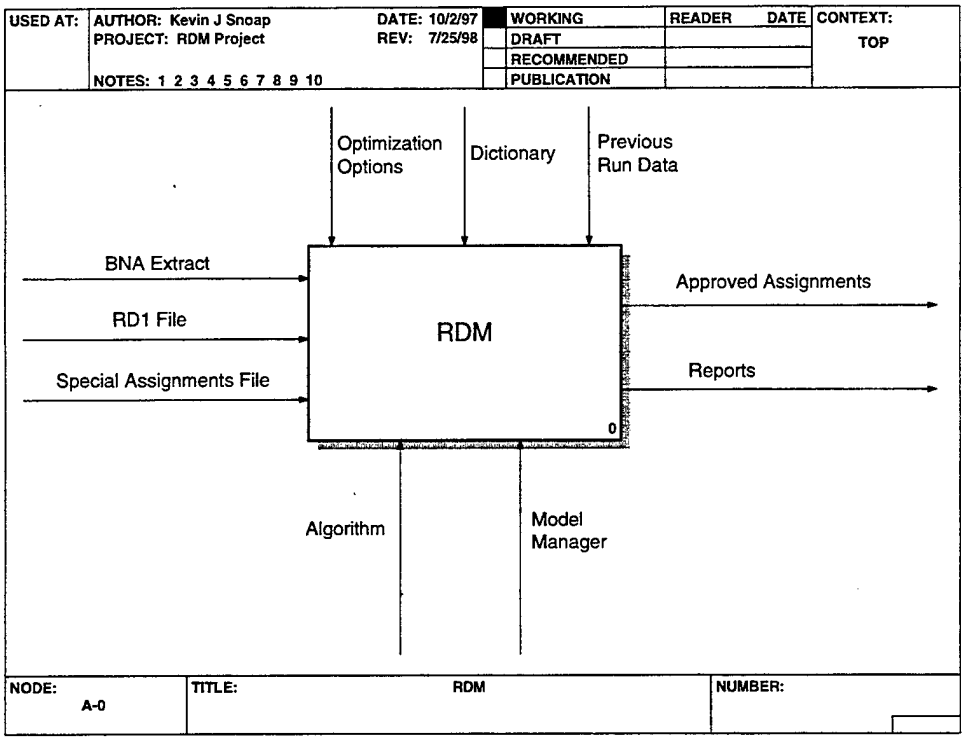
To look at all the schools or properties in one display, the RDdss manager must open up the associated school or property table. A reason for doing this is system administration. For instance, if a supervisor is auditing the RDdss's property data, looking at all the data at one time is convenient. A simple way of including this feature is by adding another switchboard. The switchboard's buttons would have VBA code associated with them, causing the desired tables to open.

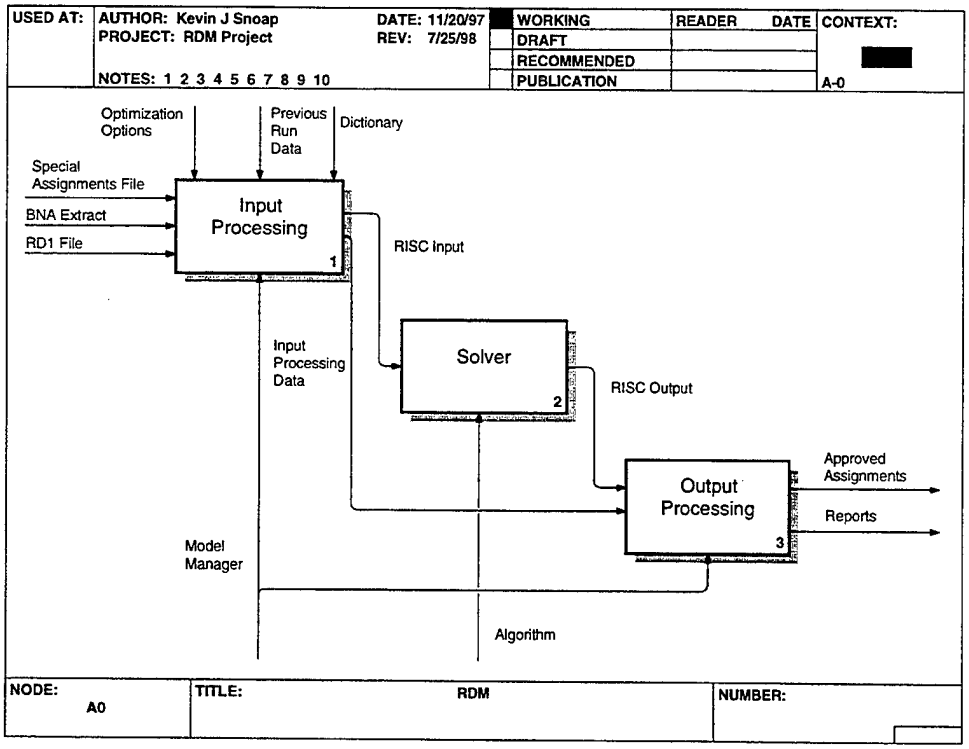
Any benefits derived from this thesis work were specifically aimed at the USMC. However, it does not have to end there. Like the USMC, other branches of the United States military must deal with the problem of how best to assign their service members to schools. Since all the services require their enlisted personnel to take the ASVAB test, the approach used in our prototype might prove an ideal solution. Further, our efforts may also contribute to the reengineering of other assignment models. So, this thesis work could actually benefit all branches of the United States military.

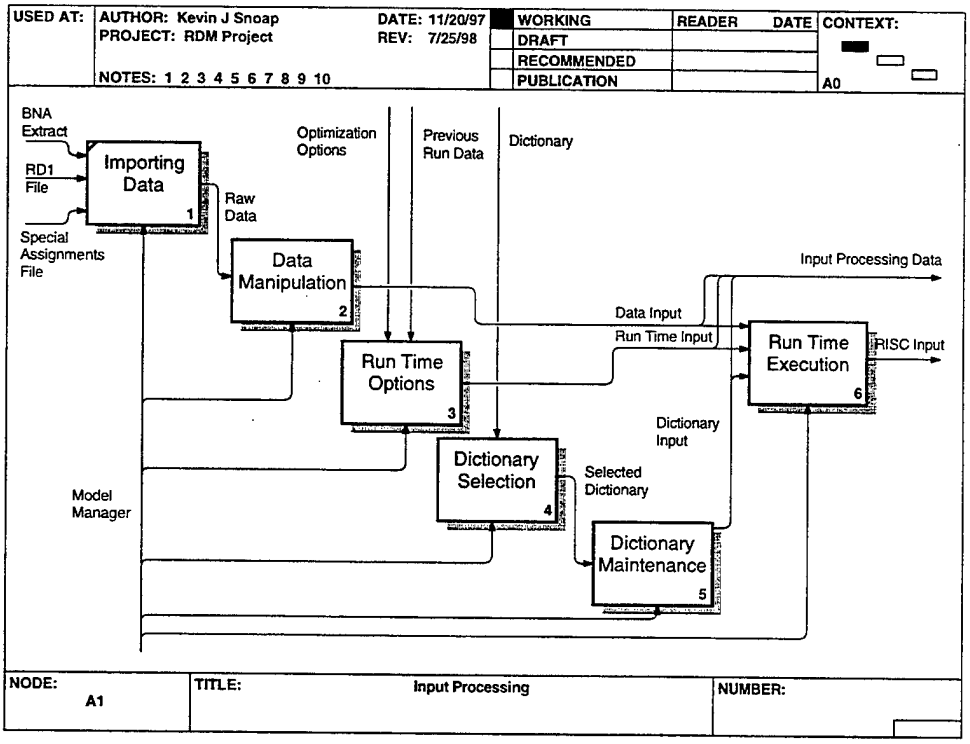
APPENDIX A. ACRONYMS

Acronym=>	Meaning
AMPL	A Modeling Language for Mathematical Programming
ARMS	Automated Recruit Management System
ASVAB	Armed Services Vocational Aptitude Battery
BNA	By Name Assignment System
BPWin	Business Process for Windows
COTS	Commercial-off-the-shelf
CPlex	Optimization Package for Complex Linear, Network and Integer Programming
DSAI	Decision Support Associates, Inc.
DSS	Decision Support System
ELS	Entry Level School
ICOM	Input, Control, Output, Mechanism
IDEF	Defense Institute Modeling
IDEF0	Business Process Modeling
MCRD	Marine Corps Recruit Depot
MCT	Marine Combat Training
MODS	Military Operational Data Store
MOS	Military Occupational Specialty
PEF	Program Enlisted For
RD1	Recruit Distribution Input File
RD3	Recruit Distribution Output File
RDdss	Recruit Distribution Decision Support System
RDM	Recruit Distribution Model
USMC	United States Marine Corps
VBA	Visual Basic for Applications

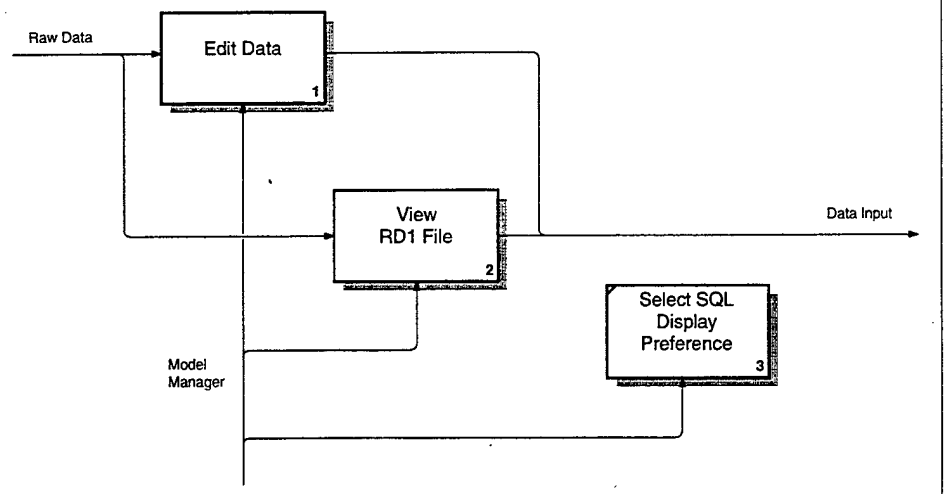
APPENDIX B. AS-IS RD BUSINESS PROCESS (IDEF0) MODEL




USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 11/20/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A-0
Optimization Options
Previous Run Data
Dictionary
Special Assignments File
BNA Extract
RD1 File
Input Processing
1
RISC Input
Input Processing Data
Solver
2
RISC Output
Output Processing
3
Approved Assignments
Reports
Model Manager
Algorithm
NODE:
A0
TITLE:
RDM
NUMBER:

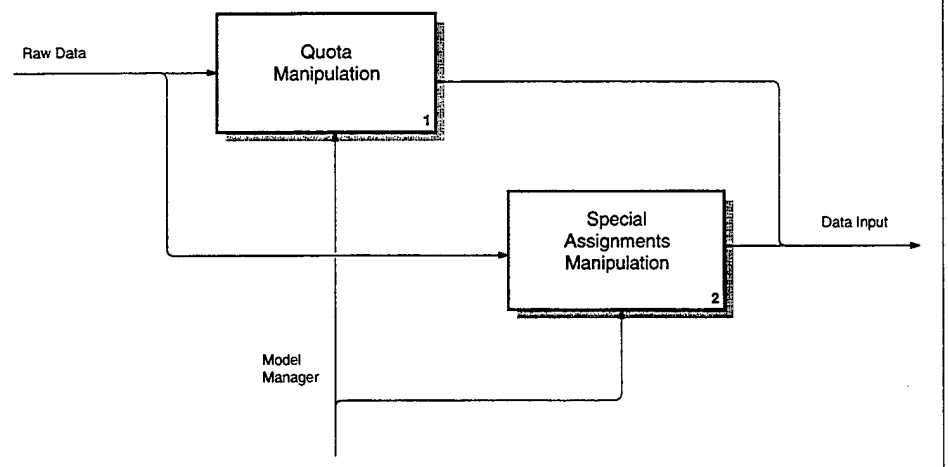

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 11/20/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A0
BNA Extract
RD1 File
Special Assignments File
Importing Data
1
Raw Data
Optimization Options
Previous Run Data
Dictionary
Data Manipulation
2
Input Processing Data
Data Input
Run Time Input
Run Time Options
3
Run Time Execution
6
RISC Input
Dictionary Input
Dictionary Selection
4
Selected Dictionary
Model Manager
Dictionary Maintenance
5
NODE:
A1
TITLE:
Input Processing
NUMBER:

USED AT:	AUTHOR: Kevin J Snoap	DATE: 12/11/97	WORKING	READER DATE	CONTEXT:
	PROJECT: RDM Project	REV: 7/25/98	DRAFT		
			RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		PUBLICATION		A1

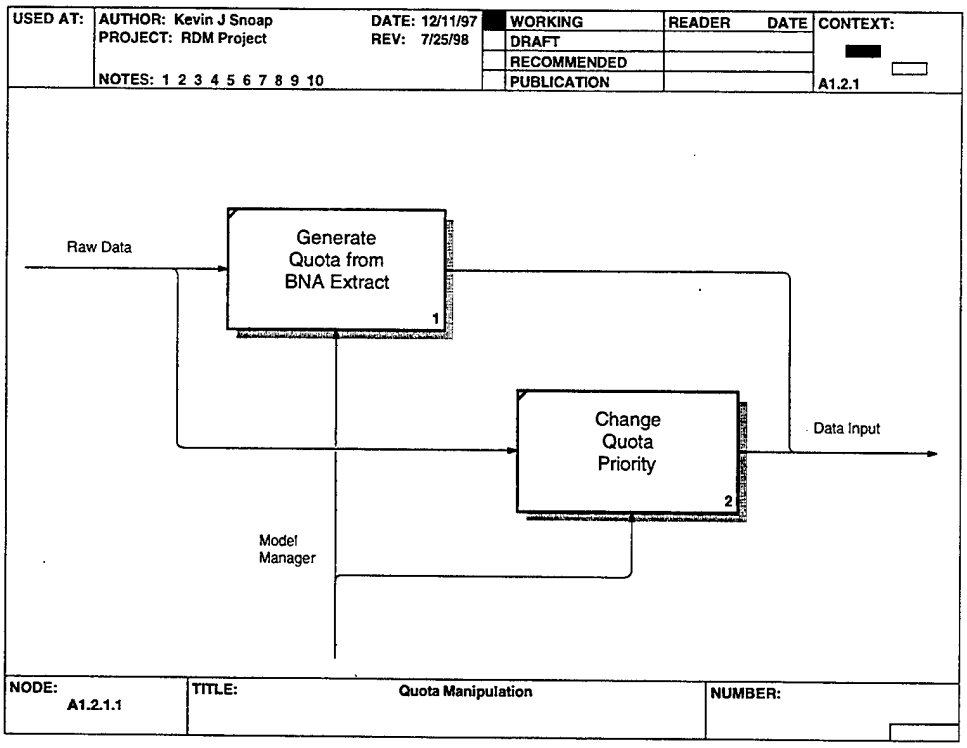


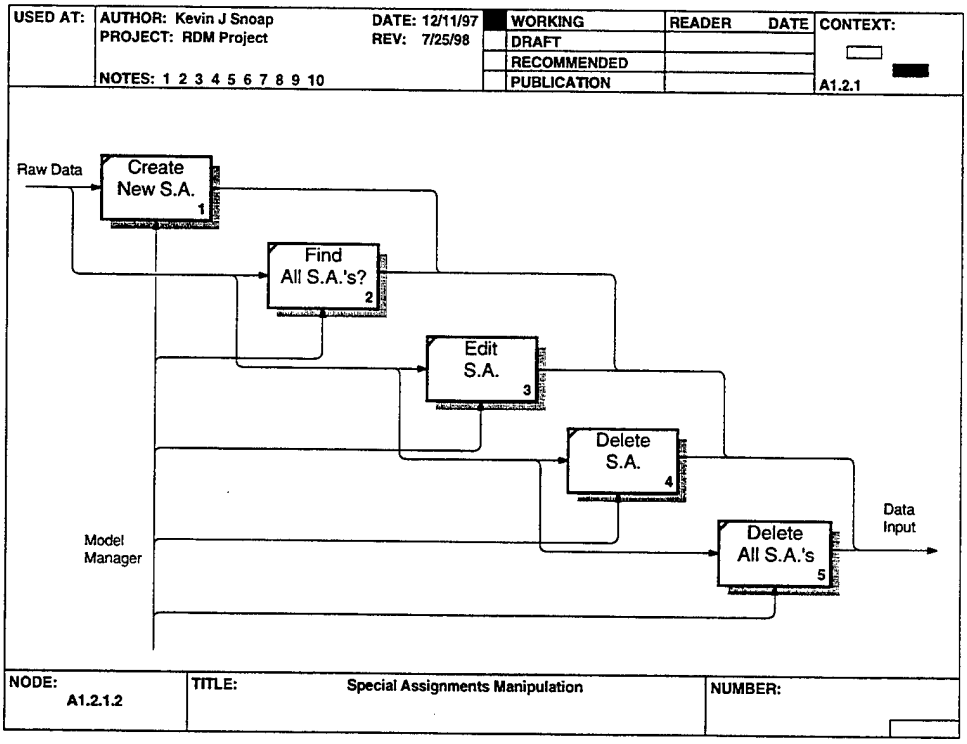
NODE: A1.2	TITLE: Data Manipulation	NUMBER:

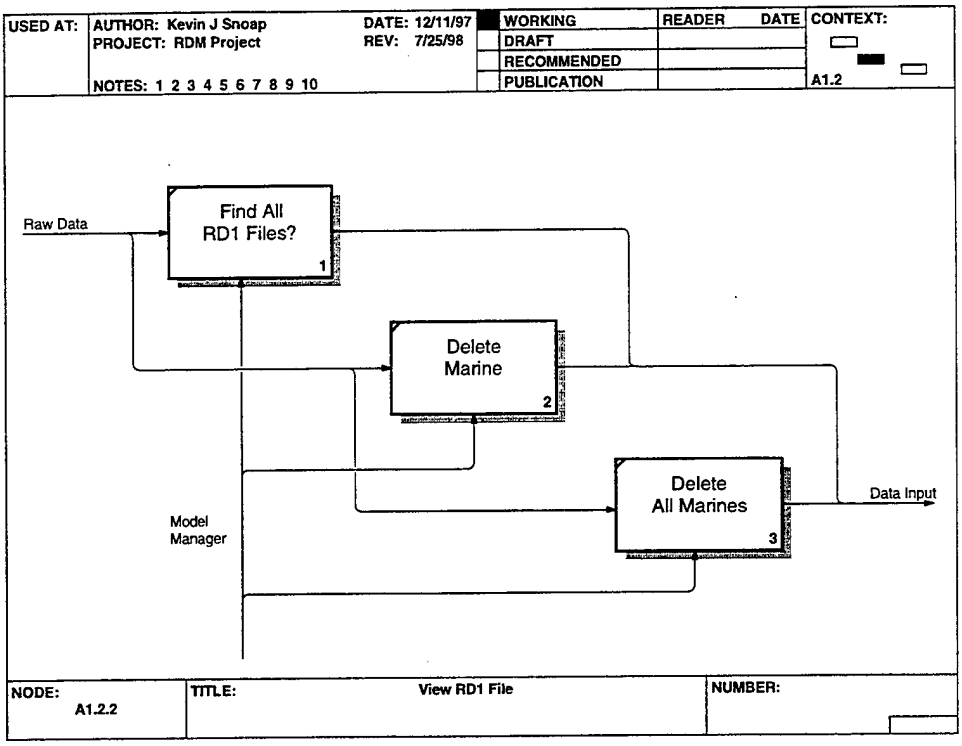
USED AT:	AUTHOR: Kevin J Snoap	DATE: 12/11/97	WORKING	READER DATE	CONTEXT:
	PROJECT: RDM Project	REV: 7/25/98	DRAFT		
			RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		PUBLICATION		A1.2

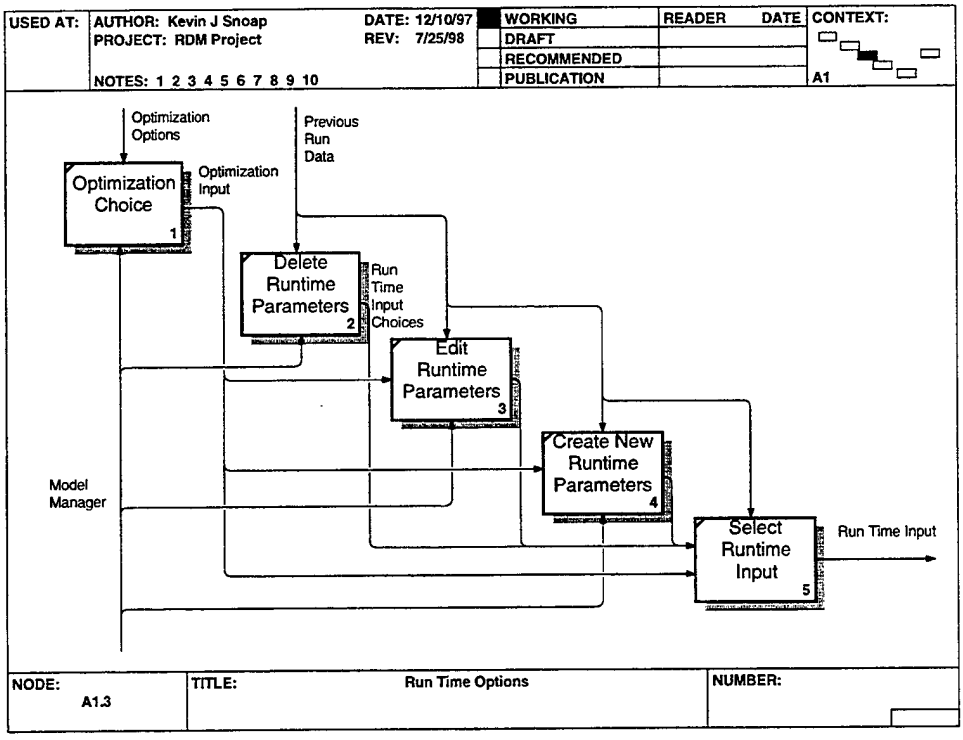


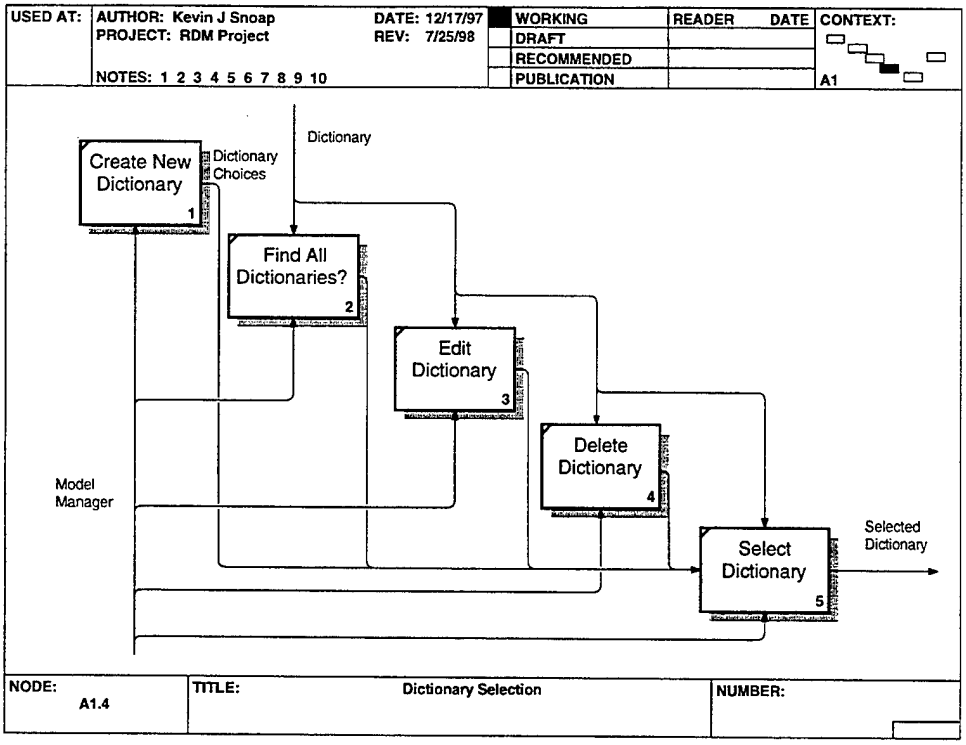
NODE: A1.2.1	TITLE: Edit Data	NUMBER:

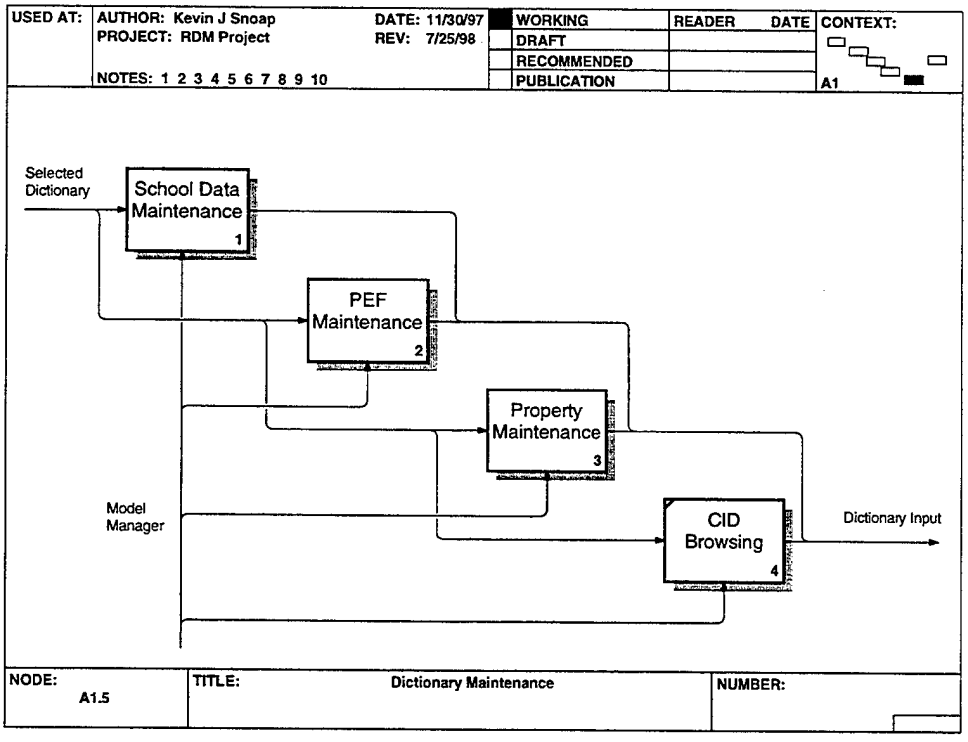

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/11/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1.2.1
Raw Data
Generate Quota from BNA Extract
1
Change Quota Priority
2
Data Input
Model Manager
NODE:
A1.2.1.1
TITLE:
Quota Manipulation
NUMBER:

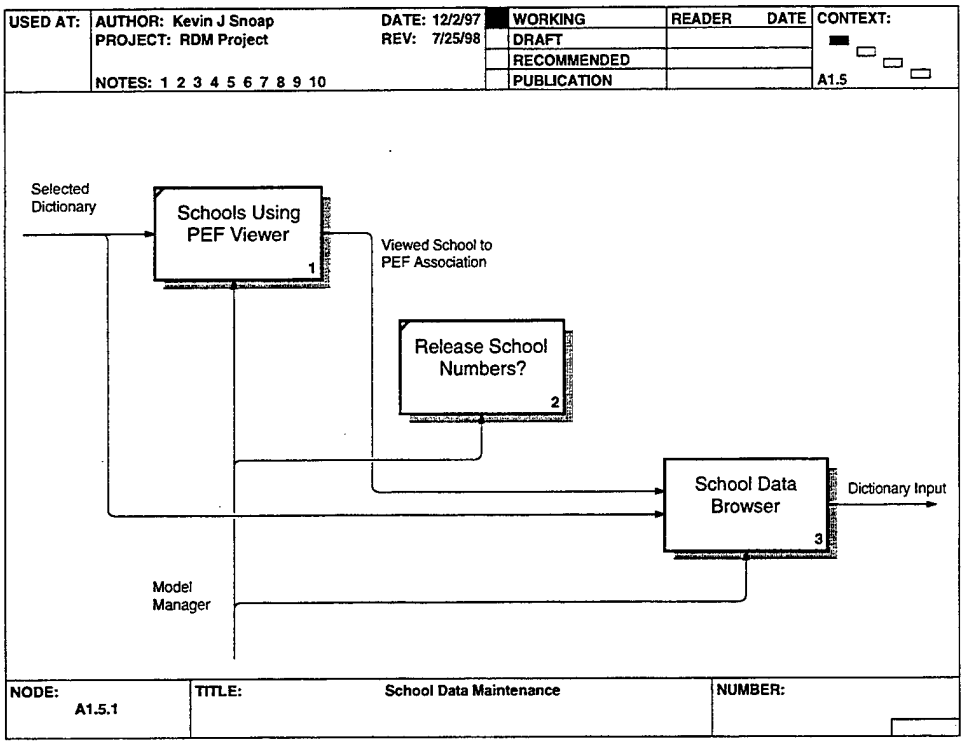

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/11/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1.2.1
Raw Data
Create New S.A.
1
Find All S.A.'s?
2
Edit S.A.
3
Delete S.A.
4
Delete All S.A.'s
5
Data Input
Model Manager
NODE:
A1.2.1.2
TITLE:
Special Assignments Manipulation
NUMBER:

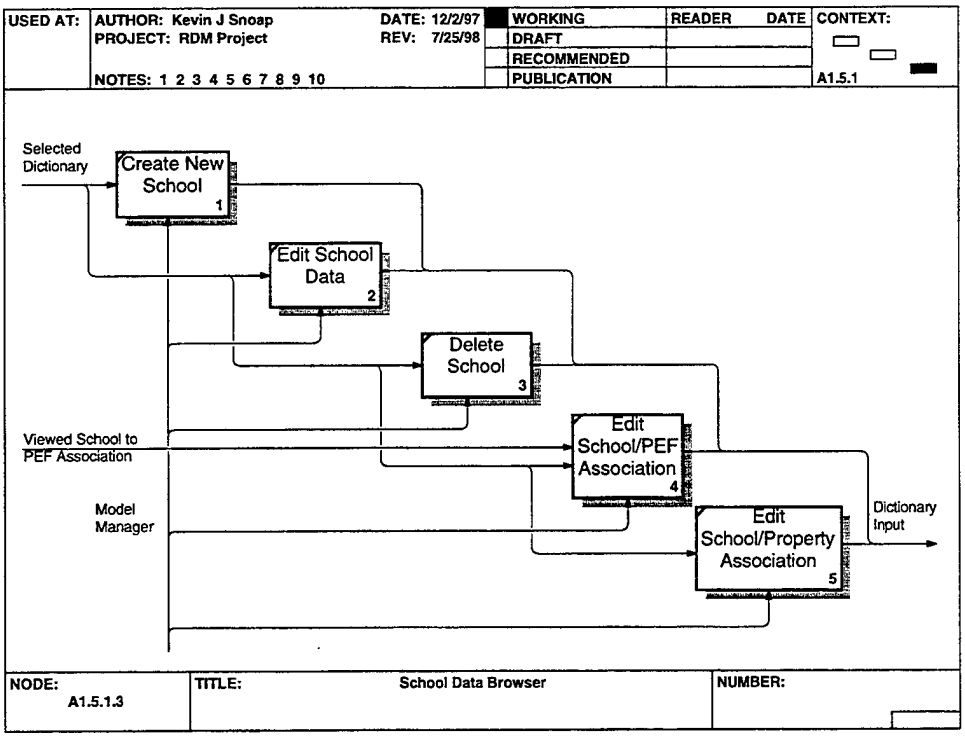

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/11/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1.2
Raw Data
Find All RD1 Files?
1
Delete Marine
2
Delete All Marines
3
Data Input
Model Manager
NODE:
A1.2.2
TITLE:
View RD1 File
NUMBER:

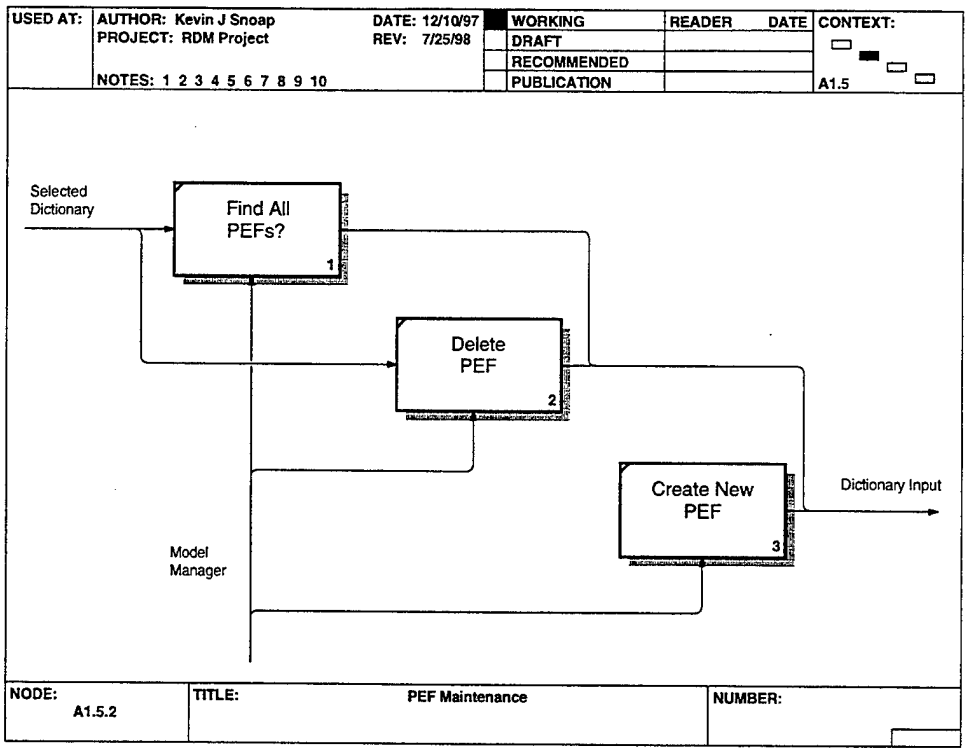

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/10/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1
Optimization Options
Previous Run Data
Optimization Choice
1
Optimization Input
Delete Runtime Parameters
2
Run Time Input Choices
Edit Runtime Parameters
3
Create New Runtime Parameters
4
Select Runtime Input
5
Run Time Input
Model Manager
NODE:
A1.3
TITLE:
Run Time Options
NUMBER:

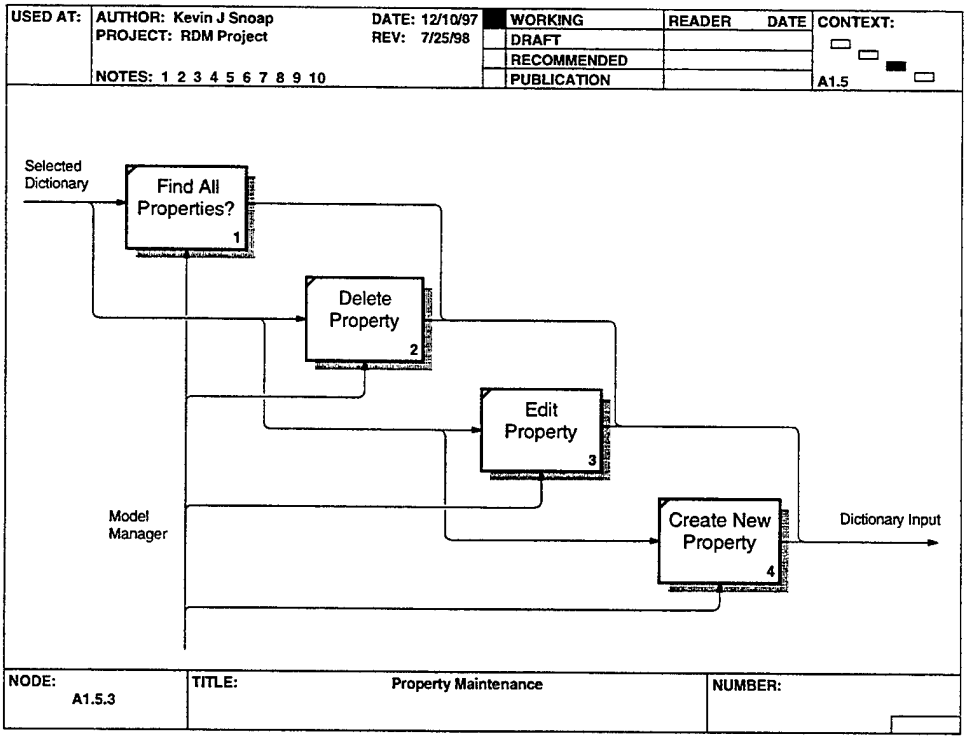

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/17/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1
Create New Dictionary 1
Dictionary Choices
Dictionary
Find All Dictionaries? 2
Edit Dictionary 3
Delete Dictionary 4
Select Dictionary 5
Model Manager
Selected Dictionary
NODE: A1.4
TITLE: Dictionary Selection
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 11/30/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1
Selected Dictionary
School Data Maintenance 1
PEF Maintenance 2
Property Maintenance 3
CID Browsing 4
Model Manager
Dictionary Input
NODE: A1.5
TITLE: Dictionary Maintenance
NUMBER:

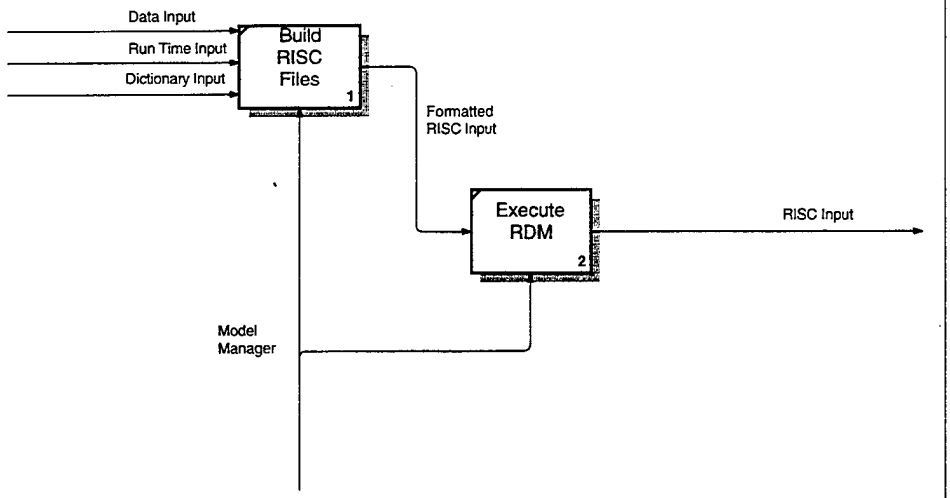

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/2/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1.5
Selected Dictionary
Schools Using PEF Viewer
1
Viewed School to PEF Association
Release School Numbers?
2
School Data Browser
3
Dictionary Input
Model Manager
NODE:
A1.5.1
TITLE:
School Data Maintenance
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/2/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1.5.1
Selected Dictionary
Create New School
1
Edit School Data
2
Delete School
3
Viewed School to PEF Association
Edit School/PEF Association
4
Model Manager
Edit School/Property Association
5
Dictionary Input
NODE:
A1.5.1.3
TITLE:
School Data Browser
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/10/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1.5
Selected Dictionary
Find All PEFs?
1
Delete PEF
2
Create New PEF
3
Dictionary Input
Model Manager
NODE:
A1.5.2
TITLE:
PEF Maintenance
NUMBER:

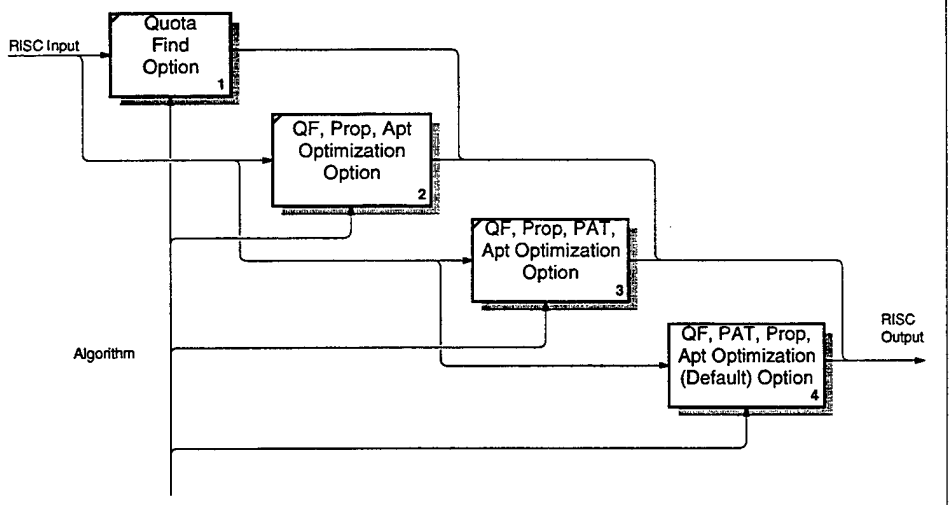

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/10/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A1.5
Selected Dictionary
Find All Properties?
1
Delete Property
2
Edit Property
3
Create New Property
4
Dictionary Input
Model Manager
NODE:
A1.5.3
TITLE:
Property Maintenance
NUMBER:

USED AT:	AUTHOR: Kevin J Snoap PROJECT: RDM Project NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 12/10/97 REV: 7/25/98	WORKING DRAFT RECOMMENDED PUBLICATION	READER DATE	CONTEXT: A1

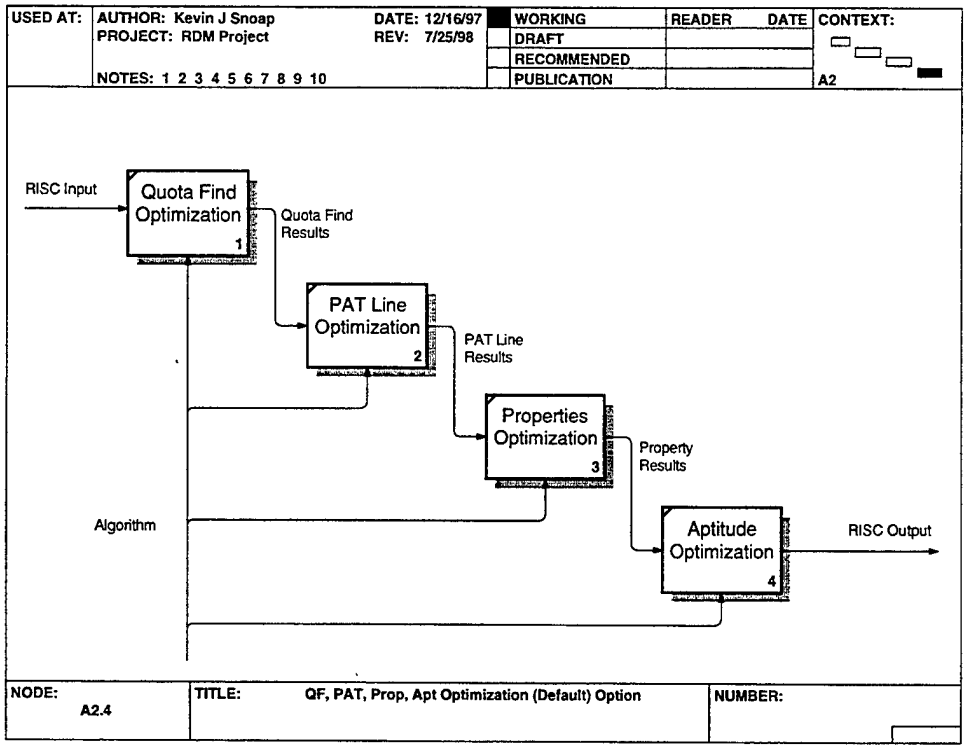


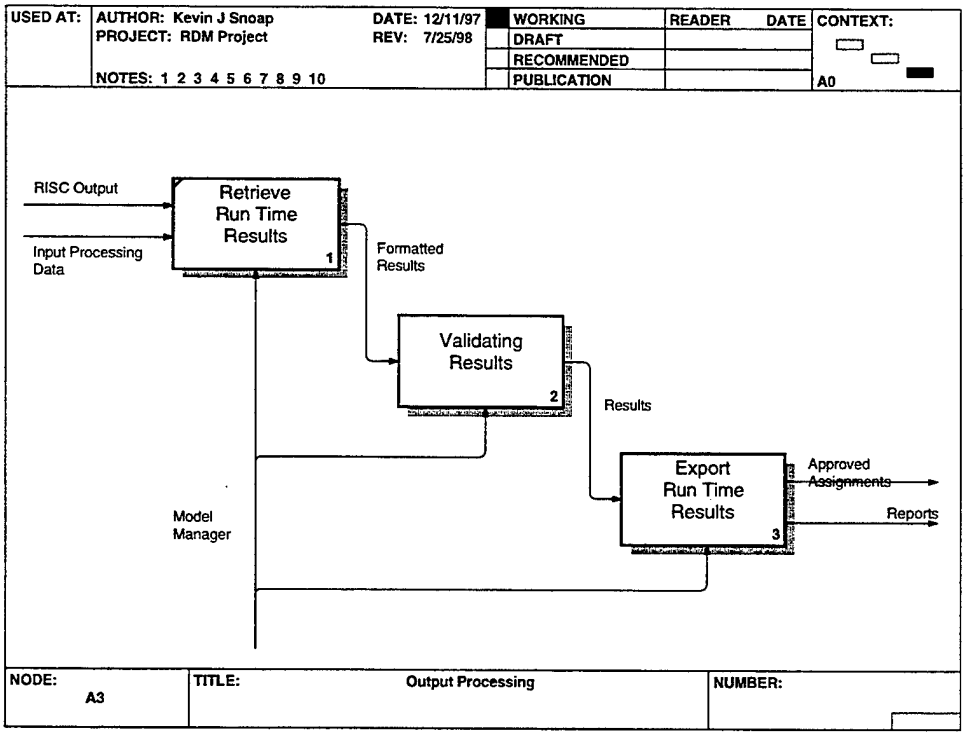
NODE: A1.6	TITLE: Run Time Execution	NUMBER:

USED AT:	AUTHOR: Kevin J Snoap PROJECT: RDM Project NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 12/15/97 REV: 7/25/98	WORKING DRAFT RECOMMENDED PUBLICATION	READER DATE	CONTEXT: A0

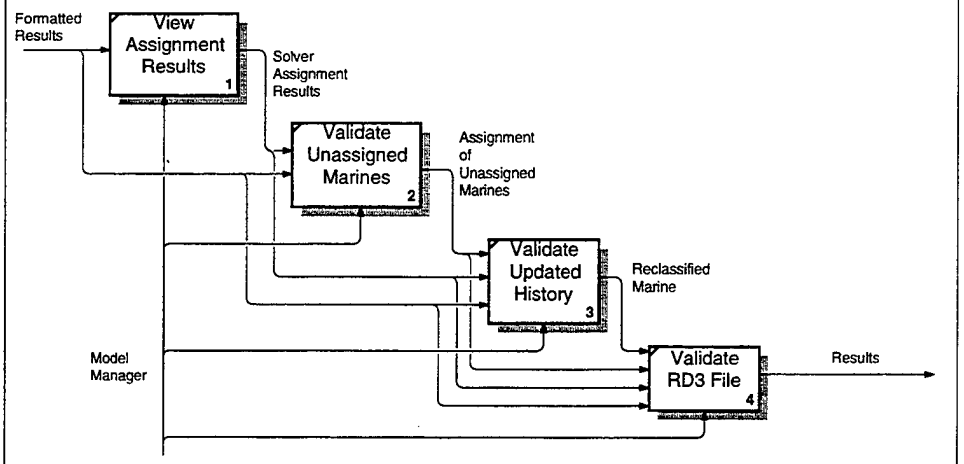


NODE: A2	TITLE: Solver	NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/16/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A2
RISC Input
Quota Find Optimization 1
Quota Find Results
PAT Line Optimization 2
PAT Line Results
Properties Optimization 3
Property Results
Aptitude Optimization 4
RISC Output
Algorithm
NODE: A2.4
TITLE: QF, PAT, Prop, Apt Optimization (Default) Option
NUMBER:

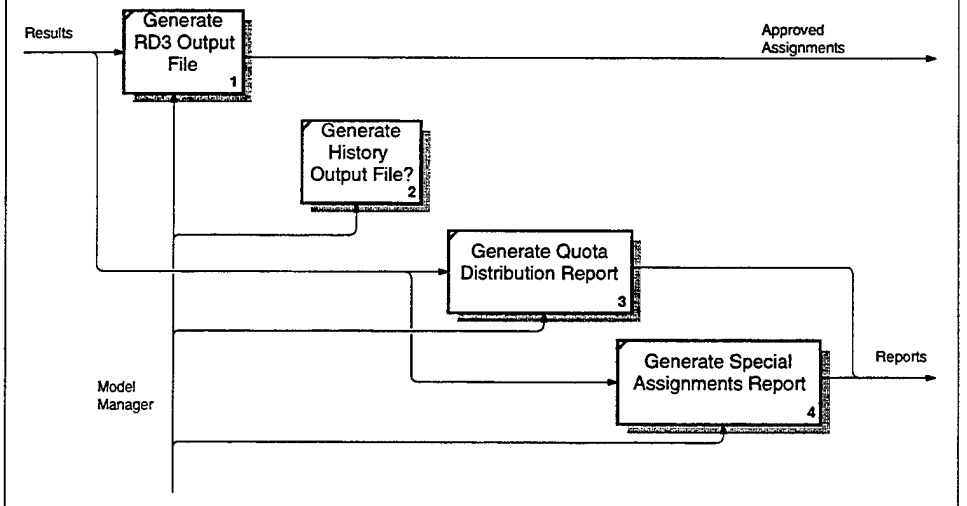

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
DATE: 12/11/97
REV: 7/25/98
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT:
A0
RISC Output
Input Processing Data
Retrieve Run Time Results 1
Formatted Results
Validating Results 2
Results
Export Run Time Results 3
Approved Assignments
Reports
Model Manager
NODE: A3
TITLE: Output Processing
NUMBER:

USED AT:	AUTHOR: Kevin J Snoap	DATE: 12/11/97	WORKING	READER DATE	CONTEXT:
	PROJECT: RDM Project	REV: 7/25/98	DRAFT		
			RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		PUBLICATION		A3



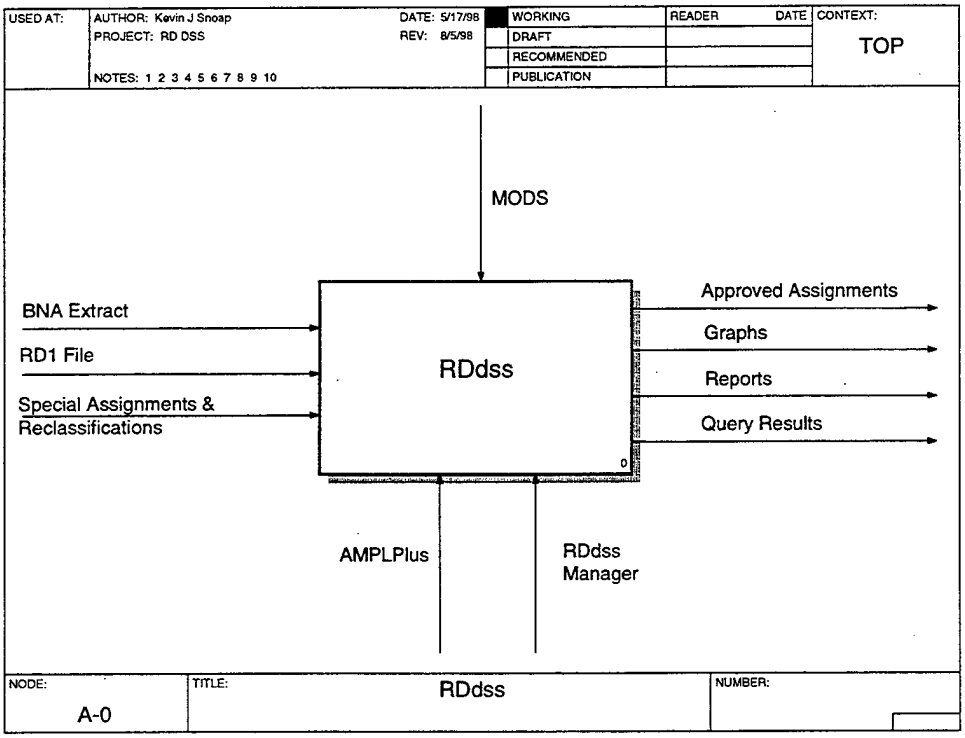
NODE: A3.2	TITLE: Validating Results	NUMBER:

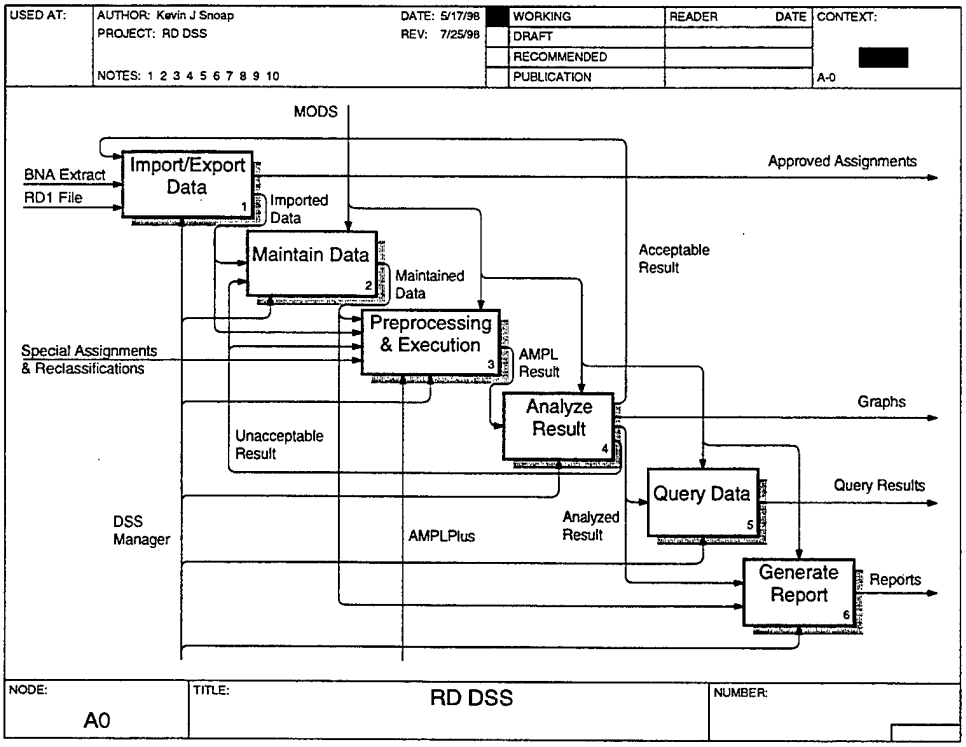
USED AT:	AUTHOR: Kevin J Snoap	DATE: 12/11/97	WORKING	READER DATE	CONTEXT:
	PROJECT: RDM Project	REV: 7/25/98	DRAFT		
			RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		PUBLICATION		A3

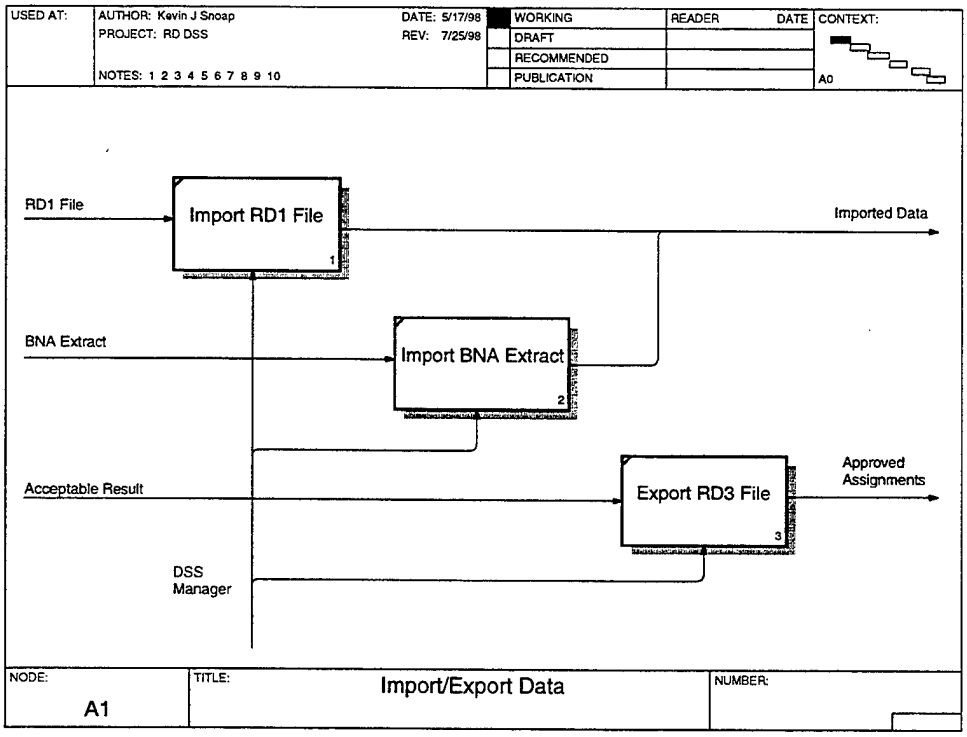


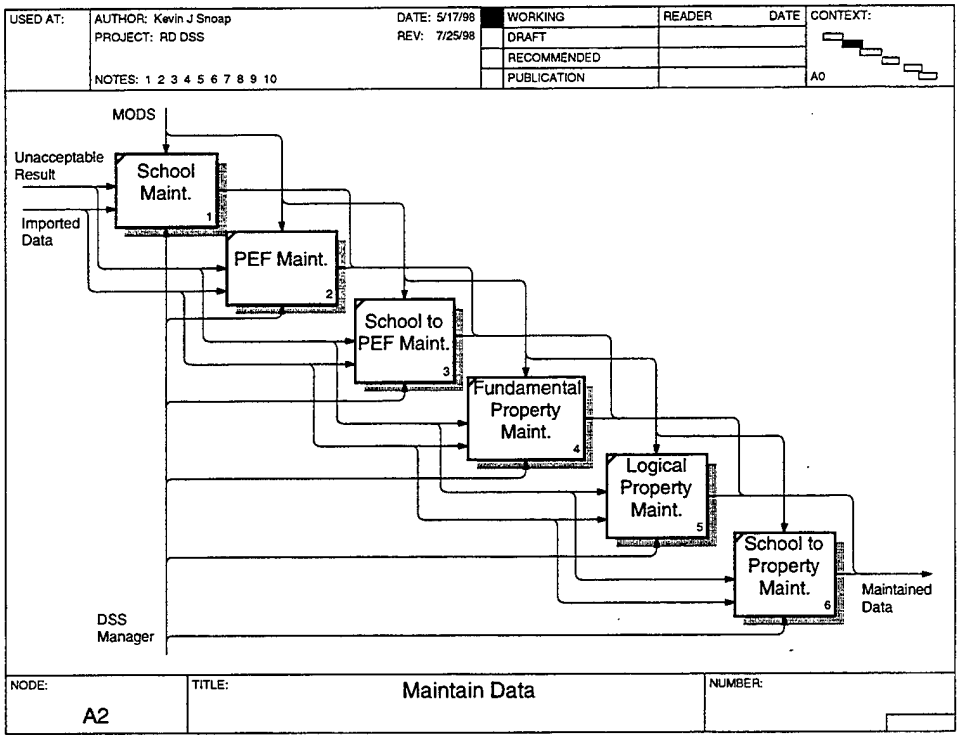
NODE: A3.3	TITLE: Export Run Time Results	NUMBER:

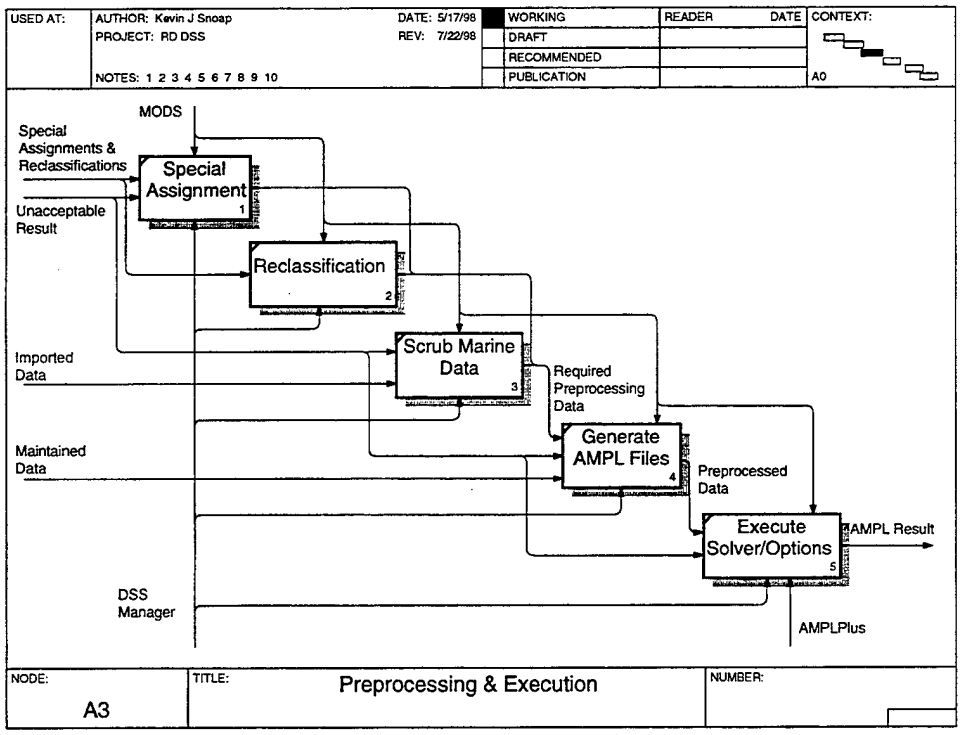
APPENDIX C. TO-BE RD BUSINESS PROCESS (IDEF0) MODEL

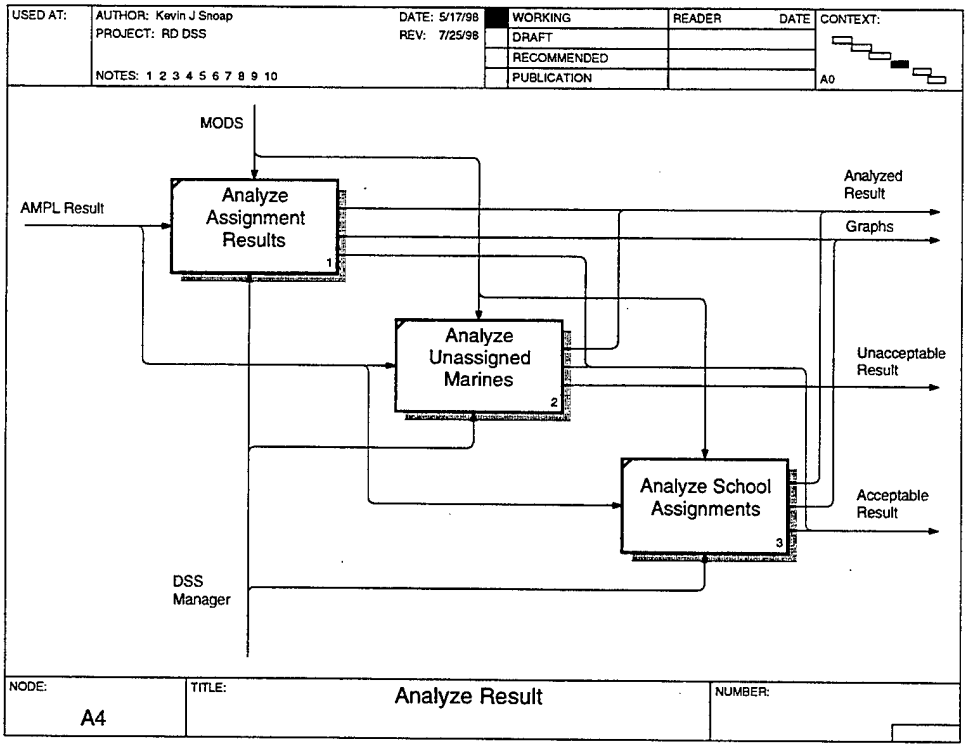


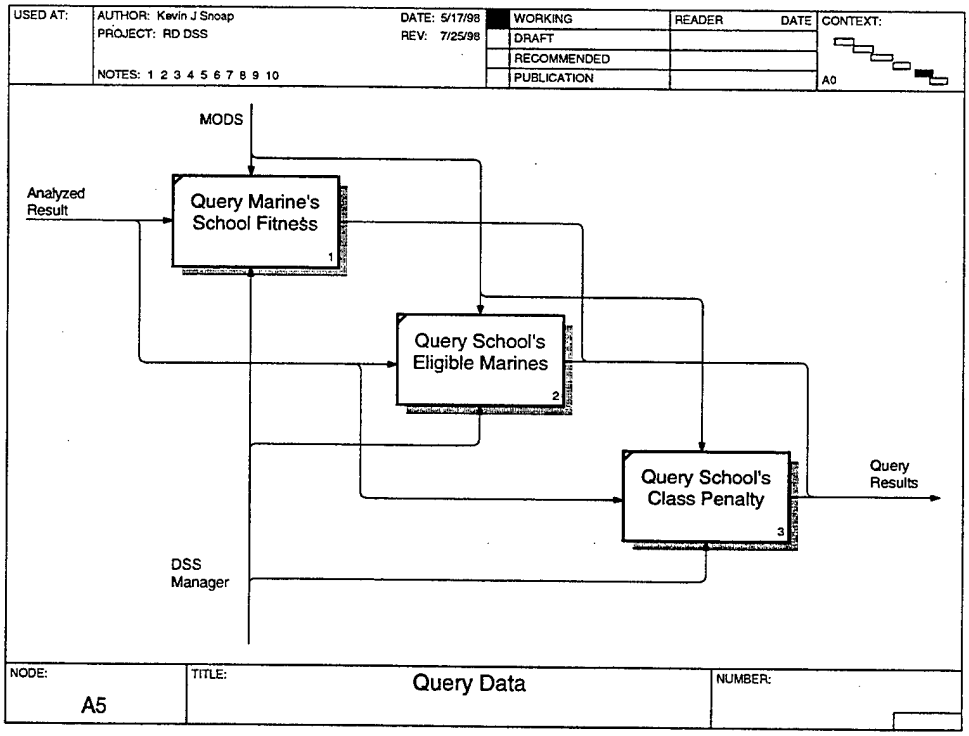

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A-0
MODS
BNA Extract
RD1 File
Import/Export Data
1
Imported Data
Approved Assignments
Maintain Data
2
Maintained Data
Acceptable Result
Preprocessing & Execution
3
Special Assignments & Reclassifications
AMPL Result
Analyze Result
4
Graphs
Unacceptable Result
Query Data
5
Query Results
DSS Manager
AMPLPlus
Analyzed Result
Generate Report
6
Reports
NODE:
A0
TITLE:
RD DSS
NUMBER:

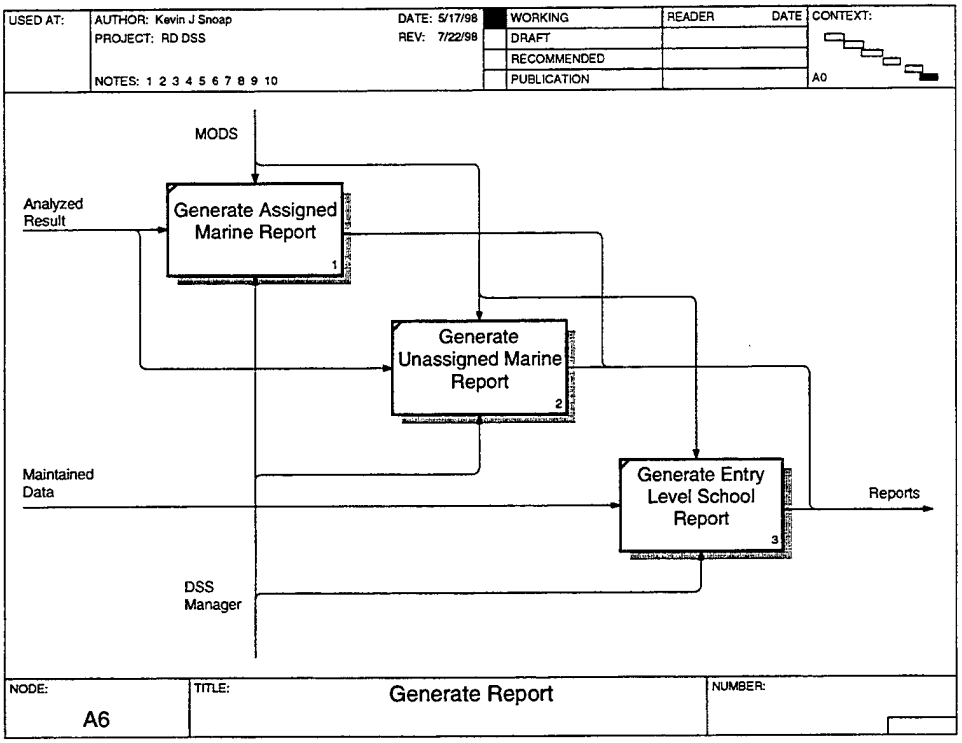

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
RD1 File
Import RD1 File
1
Imported Data
BNA Extract
Import BNA Extract
2
Acceptable Result
Export RD3 File
3
Approved Assignments
DSS Manager
NODE:
A1
TITLE:
Import/Export Data
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
Unacceptable Result
Imported Data
School Maint.
1
PEF Maint.
2
School to PEF Maint.
3
Fundamental Property Maint.
4
Logical Property Maint.
5
School to Property Maint.
6
Maintained Data
DSS Manager
NODE:
A2
TITLE:
Maintain Data
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/22/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
Special Assignments & Reclassifications
Special Assignment
1
Unacceptable Result
Reclassification
2
Imported Data
Scrub Marine Data
3
Required Preprocessing Data
Maintained Data
Generate AMPL Files
4
Preprocessed Data
Execute Solver/Options
5
AMPL Result
DSS Manager
AMPLPlus
NODE:
A3
TITLE:
Preprocessing & Execution
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
AMPL Result
Analyze Assignment Results
1
Analyze Unassigned Marines
2
Analyze School Assignments
3
Analyzed Result
Graphs
Unacceptable Result
Acceptable Result
DSS Manager
NODE:
A4
TITLE:
Analyze Result
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
Analyzed Result
Query Marine's School Fitness
1
Query School's Eligible Marines
2
Query School's Class Penalty
3
Query Results
DSS Manager
NODE:
A5
TITLE:
Query Data
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/22/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
Analyzed Result
Generate Assigned Marine Report
1
Generate Unassigned Marine Report
2
Maintained Data
Generate Entry Level School Report
3
Reports
DSS Manager
NODE:
A6
TITLE:
Generate Report
NUMBER:

APPENDIX D. RDDSS VBA CODE

```
Form: frmAnalyzeResult
  Code
    VAttribute VB_Name = "Form_frmAnalyzeResult"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Dim booSchoolAssignment As Boolean


    Private Sub btnExit_Click()
    On Error GoTo Err_btnExit_Click


        DoCmd.Close

    Exit_btnExit_Click:
        Exit Sub

    Err_btnExit_Click:
        MsgBox Err.Description
        Resume Exit_btnExit_Click


    End Sub

    Private Sub btnSchoolAssignments_Click()
    On Error GoTo Err_btnSchoolAssignments_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Set the flag
        booSchoolAssignment = True

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmSchoolAssignments"
        DoCmd.OpenForm stDocName, , , stLinkCriteria


    Exit_btnSchoolAssignments_Click:
        Exit Sub

    Err_btnSchoolAssignments_Click:
        MsgBox Err.Description
        Resume Exit_btnSchoolAssignments_Click

    End Sub

    Private Sub btnSolverAssignments_Click()
    On Error GoTo Err_btnSolverAssignments_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmAssignmentResult"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_btnSolverAssignments_Click:
        Exit Sub

    Err_btnSolverAssignments_Click:
        MsgBox Err.Description
        Resume Exit_btnSolverAssignments_Click

    End Sub

    Private Sub btnUnassigedMarines_Click()
    On Error GoTo Err_btnUnassigedMarines_Click


        Dim strSQL As String
        Dim rec As Recordset
        Dim db1 As Database

        Set db1 = CurrentDb()

        ' If the number of unassigned marines = 0, then disable the
    unassigned marines button
        strSQL = "SELECT Count(PEF) AS TotalUnassigned FROM
        Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        If rec!TotalUnassigned = 0 Then
            MsgBox ("                    All Marines were assigned.")
        Else

            Dim stDocName As String
            Dim stLinkCriteria As String

            ' Close the current form
            DoCmd.Close
            ' Open specified form
            stDocName = "frmUnassignedMarines"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

        End If


    Exit_btnUnassigedMarines_Click:
        Exit Sub

    Err_btnUnassigedMarines_Click:
        MsgBox Err.Description
        Resume Exit_btnUnassigedMarines_Click


    End Sub

    Private Sub Command8_Click()
    On Error GoTo Err_Command8_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmRDM_Main_Switchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria


    Exit_Command8_Click:
        Exit Sub

    Err_Command8_Click:
        MsgBox Err.Description
        Resume Exit_Command8_Click


    End Sub


    Private Sub Form_Deactivate()

    If booSchoolAssignment Then

        Dim strSQL As String
        Dim i As Integer, j As Integer
        Dim rec As Recordset, rec1 As Recordset, rec2 As Recordset
        Dim db1 As Database

        Set db1 = CurrentDb()

        ' Create a table to get the fit and fill weights used during the
        Application.SetOption "Confirm Action Queries", False
        ' Delete the table if it already exists
        DeleteTable


        db1.Execute "CREATE TABLE PERCENTAGE (CourseNumber TEXT, AMOS
    TEXT, Quota INTEGER, Assigned INTEGER, Percentage INTEGER);"
        Application.SetOption "Confirm Action Queries", True


        strSQL = "SELECT * FROM PERCENTAGE;"
        Set rec = db1.OpenRecordset(strSQL, dbOpenDynaset)


        strSQL = "SELECT * FROM qryTotalQuotaForRun ORDER BY
        Set rec1 = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        strSQL = "SELECT * FROM qryTotalQuotaFilledForRun ORDER BY
    SCourseNumber_FK;"
        Set rec2 = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        ' Ensure there are values in the records.
        If rec1.EOF = False Then
            If rec2.EOF = False Then
                rec1.MoveLast
                rec1.MoveFirst
                rec2.MoveLast
                rec2.MoveFirst
                ' Fill the PENALTY table with the quota values for the
                For i = 1 To rec1.RecordCount
                    rec.AddNew
                    rec!CourseNumber = rec1!SCourseNumber_FK
                    rec!AMOS = rec1!AMOS_FK
                    rec!Quota = rec1!SumOfQuota
                    rec.Update
                    rec1.MoveNext
                Next i
                rec.MoveFirst
                ' Update the PENALTY table with the percentage of quota
                For i = 1 To rec.RecordCount
                    rec2.MoveFirst
                    For j = 1 To rec2.RecordCount
                        If rec!CourseNumber = rec2!SCourseNumber_FK Then
                            If rec!AMOS = rec2!AMOS_FK Then
                                rec.Edit
                                rec!Assigned = rec2!CountOfFiscalYear_PK
                                rec!Percentage = Int((rec!Assigned /
    rec!Quota) * 100)
                                rec.Update
                                Exit For
                            End If
                        End If
                        rec2.MoveNext
                        ' If no match was found, then the percent fill is
                        If j = rec2.RecordCount Then
                            rec.Edit
                            rec!Assigned = 0
                            rec!Percentage = 0
                            rec.Update
                        End If
                    Next j
                    rec.MoveNext
                Next i

            End If

        End If

        rec.Close
        rec1.Close
        rec2.Close
        db1.Close

        ' Reset the flag
        booSchoolAssignment = False
    Else
        ' Delete the PERCENTAGE table
        DeleteTable
    End If

    End Sub


Form: frmAssignmentResult
  Code
    Attribute VB_Name = "Form_frmAssignmentResult"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit


    Private Sub Form_Current()

        Dim strSQL As String
        Dim rec As Recordset
        Dim db1 As Database


        Set db1 = CurrentDb()

        ' Calculates the high, average, and low fitness scores. Also, it
    calculates the number of solver assignments.
        strSQL = "SELECT Avg(Fitness) AS Average, Max(Fitness) As Maximum,
    Min(Fitness) As Minimum, Count(Fitness) As TotalAssigned FROM
    qryAssignmentResult;"
        Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        Me.txtFitnessAvg = Int(rec!Average)
        Me.txtFitnessMax = rec!Maximum
        Me.txtFitnessMin = rec!Minimum
        Me.txtSolverAssignments = rec!TotalAssigned


        ' Calculates the number of manual assignments, and the total number
    of assignments.
        strSQL = "SELECT Count(AssignmentType) AS TotalManualAssigned FROM
    qryManuallyAssignedMarines;"
        Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        Me.txtManualAssignments = rec!TotalManualAssigned
```

```
261  Me.txtSolverAssignments
262
263      ' Calculates the number of unassigned marines, and the total number
264  of assignable marines.
265      strSQL = "SELECT Count(PEF) AS TotalUnassigned FROM
266      Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
267      Me.txtTotalUnassigned = rec!TotalUnassigned
268      Me.txtTotalAssignable = rec!TotalUnassigned +
269
270      ' Create a table to get the fit and fill weights used during the run
271      Application.SetOption "Confirm Action Queries", False
272      db1.Execute "CREATE TABLE CONSTANTS (ConstantName TEXT, intValue
273      DoCmd.TransferText acImportFixed, "Rdm_Data Export Specification",
274  CONSTANTS, "C:\ToBeRdm\AmplData\rdm.dat"
275      Application.SetOption "Confirm Action Queries", True
276
277      strSQL = "SELECT IntValue FROM CONSTANTS;"
278      Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
279      Me.txtFitWeight = rec!IntValue
280      rec.MoveNext
281      rec.MoveNext
282      Me.txtFillWeight = rec!IntValue
283      rec.Close
284
285      ' Remove the created table
286      db1.Execute "DROP TABLE CONSTANTS;"
287      db1.Close
288
289
290  End Sub
291  Private Sub Close_Click()
292  On Error GoTo Err_Close_Click
293
294      Dim stDocName As String
295      Dim stLinkCriteria As String
296
297      ' Close the current form
298      DoCmd.Close
299      ' Open specified form
300      stDocName = "frmAnalyzeResult"
301      DoCmd.OpenForm stDocName, , , stLinkCriteria
302
303  Exit_Close_Click:
304      Exit Sub
305
306  Err_Close_Click:
307      MsgBox Err.Description
308      Resume Exit_Close_Click
309
310  End Sub
311
312 Form: frmChangeGradDate
313     Code
314     Attribute VB_Name = "Form_frmChangeGradDate"
315     Attribute VB_Creatable = True
316     Attribute VB_PredeclaredId = True
317     Attribute VB_Exposed = False
318     Option Compare Database
319     Option Explicit
320
321     Private Sub btnChangeDate_Click()
322
323         ' Change the specified graduation date in the MARINE table
324         Application.SetOption "Confirm Action Queries", False
325         DoCmd.OpenQuery "qryChangeGradDate"
326         Application.SetOption "Confirm Action Queries", True
327
328         ' Update the drop down list for MCRD Grad Date
329         Me!cmbMCRDGradDate.RowSource = "SELECT DISTINCTROW
330     [qryMCTGradDate].[GradDate] FROM [qryMCTGradDate];"
331
332         ' Force user to refresh data before performing another date change
333         Me!cmbDays.Enabled = False
334         Me!txtMCTGradDate.Enabled = False
335         Me!btnEquals.Enabled = False
336
337         ' Provide a message indicating the change was made
338         MsgBox "Date change was successful"
339
340     End Sub
341
342     Private Sub btnEquals_Click()
343
344         Dim varDate As Variant
345         Dim intDays As Integer
346
347         intDays = Me.cmbDays
348         varDate = Me.cmbMCRDGradDate
349
350         If IsNull(varDate) = False And IsNull(intDays) = False Then
351
352             ' Format this date for use in the DateAdd function
353             varDate = Format(varDate, "#####/##/##")
354
355             ' Add 120 days to this date
356             varDate = DateAdd("d", intDays, varDate)
357
358             ' Change calendar to this date
359             Me!ctlCalendar.Value = varDate
360
361             ' Convert the new date back into the original format
362             varDate = Format(varDate, "yyyymmdd")
363
364             ' Place the new date in the "upper date bound" text box on the
365     calling form.
366             Me.txtMCTGradDate = varDate
367
368             ' Allows user to change the graduation date in the MARINE table
369             Me!btnChangeDate.Enabled = True
370         End If
371
372     End Sub
373
374     Private Sub cmbMCRDGradDate_AfterUpdate()
375
376         Dim varDate As Variant
377
378         varDate = Me.cmbMCRDGradDate
379         Me!txtMCTGradDate.Value = " "
380
381         If IsNull(varDate) = False Then
382
383             ' Format this date for use in the DateAdd function
384             varDate = Format(varDate, "#####/##/##")
385
386             ' Change calendar to this date
387             Me!ctlCalendar.Value = varDate
388             Me!ctlCalendar.Visible = True
389             Me!cmbDays.Enabled = True
390             Me!txtMCTGradDate.Enabled = True
391             Me!btnEquals.Enabled = True
392
393         End If
394
395     End Sub
396
397     Private Sub Form_Current()
398
399         Me!ctlCalendar.Visible = False
400         Me!cmbDays.Enabled = False
401         Me!txtMCTGradDate.Enabled = False
402         Me!btnEquals.Enabled = False
403         Me!btnChangeDate.Enabled = False
404
405     End Sub
406     Private Sub Close_Click()
407     On Error GoTo Err_Close_Click
408
409         Dim stDocName As String
410         Dim stLinkCriteria As String
411
412         ' Close the current form
413         DoCmd.Close
414         ' Open specified form
415         stDocName = "frmImport/ExportSwitchboard"
416         DoCmd.OpenForm stDocName, , , stLinkCriteria
417
418     Exit_Close_Click:
419         Exit Sub
420
421     Err_Close_Click:
422         MsgBox Err.Description
423         Resume Exit_Close_Click
424
425     End Sub
426
427 Form: frmClass
428     Code
429     Attribute VB_Name = "Form_frmClass"
430     Attribute VB_Creatable = True
431     Attribute VB_PredeclaredId = True
432     Attribute VB_Exposed = False
433     Option Compare Database
434     Option Explicit
435
436     Private Sub btnFindSchool_Click()
437     On Error GoTo Err_btnFindSchool_Click
438
439
440         Screen.PreviousControl.SetFocus
441         DoCmd.DoMenuItem acFormBar, acEditMenu, 10, , acMenuVer70
442
443     Exit_btnFindSchool_Click:
444         Exit Sub
445
446     Err_btnFindSchool_Click:
447         MsgBox Err.Description
448         Resume Exit_btnFindSchool_Click
449
450     End Sub
451
452 Form: frmClassQuotaPenaltyAndFit
453     Code
454     Attribute VB_Name = "Form_frmClassQuotaPenaltyAndFit"
455     Attribute VB_Creatable = True
456     Attribute VB_PredeclaredId = True
457     Attribute VB_Exposed = False
458     Option Compare Database
459     Option Explicit
460
461     Private Sub btnGo_Click()
462
463         Dim varUDateBound
464         Dim lngHour As Long, lngMin As Long, lngSec As Long
465         Dim lngStart As Long, lngEnd As Long
466         Dim strSQL As String
467
468         ' Starts the timer for displaying the time taken for this subroutine
469         lngStart = Timer
470
471
472         ' Prevents the user from going back to change the MCTGradDate until a
473     fitness file is
474         ' made, or the reset button is pushed.
475         Me!MCTGradDate.Enabled = False
476         varUDateBound = Me.UDateBound
477
478         ' Generate the quota and penalty files
479         Quota_Penalty (varUDateBound)
480
481         ' Update the quota/penalty records on this form
482         strSQL = "SELECT DISTINCTROW [CLASS].[SCourseNumber_FK], " & _
483         "[CLASS].[AMOS_FK],[CLASS].[ClassNumber_PK], [CLASS].[ReportDate], "
484         "[CLASS].[Quota], [CLASS].[ClassIndex], [PENALTY].[ClassIndex_FK], "
485         "[PENALTY].[Penalty] FROM ([CLASS] INNER JOIN [PENALTY] ON " & _
486         "[CLASS].[ClassIndex] =[PENALTY].[ClassIndex_FK]);"
487         Me.RecordSource = strSQL
488
489         ' update status of status text box
490         Me!Status = "Working...."
491
492         ' Call the Fitness procedure
493         Fitness
494
495         ' update status of status text box
496         Me!Status = "Files generated!"
497         DoCmd.Beep
498
499         ' Records the stop time for running this subroutine
500         lngEnd = Timer
501         ' Determine total secs
502         lngSec = lngEnd - lngStart
503         ' Determine total hours
504         lngHour = lngSec \ 3600
505         ' Determine total mins
506         lngMin = (lngSec - lngHour * 3600) \ 60
507         ' Determine the secs remaining
508         lngSec = (lngSec - (lngMin * 60) - (lngHour * 3600)) \ 1
509         MsgBox "This operation took " & lngHour & " hour(s), " & lngMin & "
510     min(s), " & _
511         "and " & lngSec & " secs.", vbOKOnly, "Recruit Distribution Model"
512
513         Me.btnPrepareAndExecuteSolver.SetFocus
514
515
516     End Sub
517
518     Private Sub btnGo_MouseDown(Button As Integer, Shift As Integer, X As
519     Y As Single)
520
```

```
    Me!Status = "Working...."

End Sub

Private Sub btnPrepareAndExecuteSolver_Click()
On Error GoTo Err_btnPrepareAndExecuteSolver_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmPrepareAndExecuteSolver"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_btnPrepareAndExecuteSolver_Click:
    Exit Sub

Err_btnPrepareAndExecuteSolver_Click:
    MsgBox Err.Description
    Resume Exit_btnPrepareAndExecuteSolver_Click

End Sub

Private Sub btnReturnPrevious_Click()
On Error GoTo Err_btnReturnPrevious_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmPreprocessing&ExecutionSwitchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_btnReturnPrevious_Click:
    Exit Sub

Err_btnReturnPrevious_Click:
    MsgBox Err.Description
    Resume Exit_btnReturnPrevious_Click

End Sub

Private Sub cmbCourseNumberFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SCourseNumber_FK] = " & Chr(34) &
Me![cmbCourseNumberFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbCourseNumberFind] = Null
    Me.Penalty.SetFocus

End Sub

Private Sub Form_Current()

    ' Prevents the user from creating the AMPL files until a MCT
    ' graduation date is entered
    If IsNull(Me.MCTGradDate.Value) Then
        Me!btnGo.Enabled = False
        Me!Status.Enabled = False

    ' Allows the user to create the quota and penalty files
    Else
        Me!btnGo.Enabled = True

    End If
    ' update the status of the status text box
    Me!Status = " "

End Sub

Private Sub MCTGradDate_AfterUpdate()

    Dim varDate As Variant

    varDate = Me.MCTGradDate

    If IsNull(varDate) = False Then

        Me!btnGo.Enabled = True
        Me!Status.Enabled = True

        ' Format this date for use in the DateAdd function
        varDate = Format(varDate, "########")

        ' Add 120 days to this date
        varDate = DateAdd("d", 120, varDate)

        ' Convert the new date back into the original format
        varDate = Format(varDate, "yyyymmdd")

        ' Place the new date in the "upper date bound" text box on the
calling form.
        Me.UDateBound = varDate

    Else
        Me!btnGo.Enabled = False
        Me!Status.Enabled = False
    End If

End Sub


Form: frmFind
Code
    Attribute VB_Name = "Form_frmFind"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub btnFind_Click()
    On Error GoTo Err_btnFind_Click

    'Screen.PreviousControl.SetFocus
    Forms!frmSchoolToPropertyMaint.SetFocus
    DoCmd.FindRecord Me.cmbCourseNumber, , True, , True, acAll
    'DoCmd.DoMenuItem acFormBar, acEditMenu, 10, , acMenuVer70

    Exit_btnFind_Click:
        Exit Sub

    Err_btnFind_Click:
        MsgBox Err.Description
        Resume Exit_btnFind_Click

    End Sub


    Private Sub cmbFPropertyNameFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[FPropertyName_PK] = " & Chr(34) &
    Me![cmbFPropertyNameFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbFPropertyNameFind] = Null
        Me.FPropertyName_PK.SetFocus

    End Sub

    Private Sub Command11_Click()
        ' Forces user to enter data for property name and description
    If IsNull(Me.FPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!MarineField.Enabled = False
        Me!Operator.Enabled = False
        Me!Label10.Visible = False
        Me!subfrmFundamentalProperty.Visible = False
        Me!subfrmFundamentalPropertyList.Visible = False
        Me!cmbPotentialValue.Visible = False
        MsgBox "You must provide a 'Property Name' and 'Description'
    before building the fundamental equation."
    ' Allows the user to enter values for the fundamental equation
    Else
        Me!MarineField.Enabled = True
        Me!Operator.Enabled = True
        Me!Label10.Visible = True
        If Forms!frmFundamentalProperty.Operator.Value = "in" Then
            Me!subfrmFundamentalProperty.Visible = False
            Me!subfrmFundamentalPropertyList.Visible = True

        Else
            Me!cmbPotentialValue.Visible = True
            Me!subfrmFundamentalProperty.Visible = True
            Me!subfrmFundamentalPropertyList.Visible = False
        End If
    End If

    End Sub

    Private Sub Description_AfterUpdate()

        Dim strSQL As String

        ' Forces user to enter data for property name and description
    If IsNull(Me.FPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!MarineField.Enabled = False
        Me!Operator.Enabled = False
        Me!Label10.Visible = False
        Me!subfrmFundamentalProperty.Visible = False
        Me!subfrmFundamentalPropertyList.Visible = False

    ' Allows the user to enter values for the fundamental equation
    Else
        Me!MarineField.Enabled = True
        Me!Operator.Enabled = True
        Me!Label10.Visible = True
        If Forms!frmFundamentalProperty.Operator.Value = "in" Then
            strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM
    Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK.RowSo
    urce = strSQL
            Me!subfrmFundamentalProperty.Visible = False
            Me!subfrmFundamentalPropertyList.Visible = True

        Else
            strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM
    Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSou
    rce = strSQL
            Me!subfrmFundamentalProperty.Visible = True
            Me!subfrmFundamentalPropertyList.Visible = False
        End If
    End If

    End Sub

    Private Sub Form_BeforeUpdate(Cancel As Integer)

        [FundPropTimeStamp] = Now

    End Sub
    Private Sub Form_Current()

        Dim strSQL As String

        'strSQL = "SELECT MarineField, Operator FROM FUNDAMENTAL_PROPERTY
    WHERE FPropertyName_PK = " & FundPropName

```

```
781   117     ' Forces user to enter data for property name and description
782   118   If IsNull(Me.FPropertyName_PK.Value) Or IsNull(Me.Description.Value)
783   119     Me!MarineField.Enabled = False
784   120     Me!Operator.Enabled = False
785   121     Me!Label10.Visible = False
786   122     Me!subfrmFundamentalProperty.Visible = False
787   123     Me!subfrmFundamentalPropertyList.Visible = False
788   124
789   125   ' Allows the user to enter values for the fundamental equation
790   126   Else
791   127     Me!MarineField.Enabled = True
792   128     Me!Operator.Enabled = True
793   129     Me!Label10.Visible = True
794   130     If Forms!frmFundamentalProperty.Operator.Value = "in" Then
795   131       ' Update the Potential Value list
796   132       Me!subfrmFundamentalProperty.Visible = False
797   133       Me!subfrmFundamentalPropertyList.Visible = True
798   134       strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM
799   135
800   Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK.RowSo
801   urce = strSQL
802   136     Else
803   137       ' Update the Potential Value list
804   138       strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM
805   139
806   Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSou
807   rce = strSQL
808   140       Me!subfrmFundamentalProperty.Visible = True
809   141       Me!subfrmFundamentalPropertyList.Visible = False
810   142     End If
811   143
812   144     ' Set focus to the property name. This prevents problem
813   associated with entering the "Operator" object.
814   145     FPropertyName_PK.SetFocus
815   146   End If
816   147
817   148
818   149
819   150 End Sub
820   151
821   152 Private Sub FPropertyName_PK_AfterUpdate()
822   153
823   154   Dim strSQL As String
824   155
825   156     ' Forces user to enter data for property name and description
826   157   If IsNull(Me.FPropertyName_PK.Value) Or IsNull(Me.Description.Value)
827   158     Me!MarineField.Enabled = False
828   159     Me!Operator.Enabled = False
829   160     Me!Label10.Visible = False
830   161     Me!subfrmFundamentalProperty.Visible = False
831   162     Me!subfrmFundamentalPropertyList.Visible = False
832   163
833   164   ' Allows the user to enter values for the fundamental equation
834   165   Else
835   166     Me!MarineField.Enabled = True
836   167     Me!Operator.Enabled = True
837   168     Me!Label10.Visible = True
838   169     If Forms!frmFundamentalProperty.Operator.Value = "in" Then
839   170       strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM
840   171
841   Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK.RowSo
842   urce = strSQL
843   172       Me!subfrmFundamentalProperty.Visible = False
844   173       Me!subfrmFundamentalPropertyList.Visible = True
845   174
846   175     Else
847   176       strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM
848   177
849   Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSou
850   rce = strSQL
851   178       Me!subfrmFundamentalProperty.Visible = True
852   179       Me!subfrmFundamentalPropertyList.Visible = False
853   180     End If
854   181   End If
855   182
856   183 End Sub
857   184
858   185 Private Sub MarineField_AfterUpdate()
859   186
860   187   Dim strSQL As String
861   188
862   189   ' Update the Potential Value list
863   190   strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM MARINE;"
864   191
865   Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK.RowSo
866   urce = strSQL
867   192
868   Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSou
869   rce = strSQL
870   193
871   194 End Sub
872   195
873   196 Private Sub Operator_AfterUpdate()
874   197
875   198   If Forms!frmFundamentalProperty.Operator.Value = "in" Then
876   199     Me!subfrmFundamentalProperty.Visible = False
877   200     Me!subfrmFundamentalPropertyList.Visible = True
878   201   Else
879   202     Me!subfrmFundamentalProperty.Visible = True
880   203     Me!subfrmFundamentalPropertyList.Visible = False
881   204   End If
882   205
883   206 End Sub
884   207 Private Sub btnDelete_Click()
885   208 On Error GoTo Err_btnDelete_Click
886   209
887   210
888   211   DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
889   212   DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70
890   213
891   214 Exit_btnDelete_Click:
892   215   Exit Sub
893   216
894   217 Err_btnDelete_Click:
895   218   MsgBox Err.Description
896   219   Resume Exit_btnDelete_Click
897   220
898   221 End Sub
899   222 Private Sub Command14_Click()
900   223 On Error GoTo Err_Command14_Click
901   224
902   225   Dim stDialStr As String
903   226   Dim PrevCtl As Control
904   227   Const ERR_OBJNOTEXIST = 2467
905   228   Const ERR_OBJNOTSET = 91
906   229
907   230   Set PrevCtl = Screen.PreviousControl
908   231
909   232   If TypeOf PrevCtl Is TextBox Then
910   233     stDialStr = IIf(VarType(PrevCtl) > V_NULL, PrevCtl, "")
911   234   ElseIf TypeOf PrevCtl Is ListBox Then
912   235     stDialStr = IIf(VarType(PrevCtl) > V_NULL, PrevCtl, "")
913   236   ElseIf TypeOf PrevCtl Is ComboBox Then
914   237     stDialStr = IIf(VarType(PrevCtl) > V_NULL, PrevCtl, "")
915   238   Else
916   239     stDialStr = ""
917   240   End If
918   241
919   242   Application.Run "utility.wlb_AutoDial", stDialStr
920   243
921   244 Exit_Command14_Click:
922   245   Exit Sub
923   246
924   247 Err_Command14_Click:
925   248   If (Err = ERR_OBJNOTEXIST) Or (Err = ERR_OBJNOTSET) Then
926   249     Resume Next
927   250   End If
928   251   MsgBox Err.Description
929   252   Resume Exit_Command14_Click
930   253
931   254 End Sub
932   255
933   256 Private Sub Operator_Enter()
934   257
935   258   If Me.Operator.Value = "in" Then
936   259     MsgBox "The 'in' operator is not changeable. It is removed by
937   deleting the property.", vbInformation, "Unauthorized Action"
938   260     subfrmFundamentalPropertyList.SetFocus
939   261   End If
940   262
941   263 End Sub
942
943 Form: frmGenerateReport
944   Code
945   1   Attribute VB_Name = "Form_frmGenerateReport"
946   2   Attribute VB_Creatable = True
947   3   Attribute VB_PredeclaredId = True
948   4   Attribute VB_Exposed = False
949   5   Option Compare Database
950   6   Option Explicit
951   7
952   8   Private Sub btnAssignmentReport_Click()
953   9
954   10    DoCmd.OpenReport "rptassignedMarines", acViewPreview
955   11
956   12  End Sub
957   13
958   14  Private Sub btnEntryLevelSchool_Click()
959   15
960   16    DoCmd.OpenReport "rptEntryLevelSchool", acViewPreview
961   17
962   18
963   19  End Sub
964   20
965   21  Private Sub btnExit_Click()
966   22  On Error GoTo Err_btnExit_Click
967   23
968   24
969   25    DoCmd.Close
970   26
971   27  Exit_btnExit_Click:
972   28    Exit Sub
973   29
974   30  Err_btnExit_Click:
975   31    MsgBox Err.Description
976   32    Resume Exit_btnExit_Click
977   33
978   34
979   35  End Sub
980   36
981   37  Private Sub btnUnassignedMarineReport_Click()
982   38
983   39    DoCmd.OpenReport "rptUnassignedMarines", acViewPreview
984   40
985   41  End Sub
986   42
987   43  Private Sub Command8_Click()
988   44  On Error GoTo Err_Command8_Click
989   45
990   46    Dim stDocName As String
991   47    Dim stLinkCriteria As String
992   48
993   49    ' Close the current form
994   50    DoCmd.Close
995   51    ' Open specified form
996   52    stDocName = "frmRDM_Main_Switchboard"
997   53    DoCmd.OpenForm stDocName, , , stLinkCriteria
998   54
999   55  Exit_Command8_Click:
1000    56    Exit Sub
1001    57
1002    58  Err_Command8_Click:
1003    59    MsgBox Err.Description
1004    60    Resume Exit_Command8_Click
1005    61
1006    62  End Sub
1007
1008  Form: frmImport/ExportSwitchboard
1009    Code
1010    1   Attribute VB_Name = "Form_frmImport/ExportSwitchboard"
1011    2   Attribute VB_Creatable = True
1012    3   Attribute VB_PredeclaredId = True
1013    4   Attribute VB_Exposed = False
1014    5   Option Compare Database
1015    6   Option Explicit
1016    7
1017    8   Private Sub btnExit_Click()
1018    9   On Error GoTo Err_btnExit_Click
1019    10
1020    11
1021    12    DoCmd.Close
1022    13
1023    14  Exit_btnExit_Click:
1024    15    Exit Sub
1025    16
1026    17  Err_btnExit_Click:
1027    18    MsgBox Err.Description
1028    19    Resume Exit_btnExit_Click
1029    20
1030    21
1031    22  End Sub
1032    23
1033    24  Private Sub btnExportRD3_Click()
1034    25
1035    26    Application.SetOption "Confirm Action Queries", False
1036    27    DoCmd.SetWarnings (False)
1037    28    ' Exports assignments to RD3.txt file
1038    29    ExportRD3
1039    30    ' Archives data from the Marine and assignment tables, having report
1040  dates less than the current date
```

```
    ArchiveData
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

End Sub


Private Sub Command2_Click()

    Dim stDocName As String
    Dim stLinkCriteria As String

    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)
    ' Imports the RD1 file
    ImportRD1
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmChangeGradDate"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

End Sub

Private Sub Command4_Click()

    ' Imports BNA.txt file
    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)
    DoCmd.RunSQL "DELETE * FROM BNA_EXTRACT;"
    ImportBNA
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

End Sub

Private Sub Command8_Click()

On Error GoTo Err_Command8_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmRDM_Main_Switchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command8_Click:
    Exit Sub

Err_Command8_Click:
    MsgBox Err.Description
    Resume Exit_Command8_Click

End Sub

Form: frmLogicalProperty
Code
Attribute VB_Name = "Form_frmLogicalProperty"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub btnClose_Click()
On Error GoTo Err_btnClose_Click


    DoCmd.Close
    DoCmd.OpenForm "frmMaintenanceSwitchboard"

Exit_btnClose_Click:
    Exit Sub

Err_btnClose_Click:
    MsgBox Err.Description
    Resume Exit_btnClose_Click

End Sub

Private Sub cmbFundProp_Click()

    ' Adds the selected fundamental property to the logical equation
    Me.LogicalEquation.Value = Me.LogicalEquation.Value &
Me.cmbFundProp.Value & " "

End Sub
Private Sub cmbLogOper_Click()

    ' Adds the selected logical operator to the logical equation
    Me.LogicalEquation.Value = Me.LogicalEquation.Value &
Me.cmbLogOper.Value & " "

End Sub

Private Sub cmbLPropertyNameFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[LPropertyName_PK] = " & Chr(34) &
Me![cmbLPropertyNameFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbLPropertyNameFind] = Null
    Me.LPropertyName_PK.SetFocus

End Sub

Private Sub Command10_Click()

    ' Removes the last entry in the logical equation

    Dim strInName, strUndo  As String     ' input name
    Dim intSpacePos         As Integer    ' position of the space
    Dim i                   As Integer    ' loop index
    Dim booResult           As Boolean
    Dim lngConversion       As Long
    Dim varConv             As Variant


    ' get the logical expression
    strInName = Me.LogicalEquation.Value

    'Check for "Null" value
    If strInName <> "Null" Then

    ' declare the temporary string as an array
    ReDim strTemp(0) As String

    ' Find the position of the first space
    intSpacePos = InStr(strInName, " ")

    ' Loop until no more spaces are found. InStr returns 0 if the
space is not found.
    Do Until intSpacePos = 0
        ' redimension the array, adding another element
        ' UBound tells us how many elements there already are in the
        ReDim Preserve strTemp(UBound(strTemp) + 1)

        ' set the new element to the characters from the beginning of
        ' string up to the first space.  Left$ returns a number of
characters
        ' from the beginning of a string
        strTemp(UBound(strTemp)) = Trim$(Left$(strInName, intSpacePos

        ' now we've copied these characters into the name, they can
        ' removed from the original string.  Right$ returns a number
        ' characters from the end of the string.  It is the rightmost
        ' characters, therefore we subtract the position of the space
        ' the length of the string.  This gives us the number of
characters remaining.
        strInName = LTrim$(Right$(strInName, Len(strInName) -

        ' find the next space
        intSpacePos = InStr(strInName, " ") 'Or intSpacePos =
InStr(strInName, "(") 'Or intSpacePos = InStr(strInName, ")")

    Loop

    ' Remove the last element of the array, unless there is only one
element left
    If UBound(strTemp) > 1 Then
        ReDim Preserve strTemp(UBound(strTemp) - 1)

        ' Loop through the elements of the array and place them back
into the logical equation expression
        For i = 1 To (UBound(strTemp))
            strUndo = strUndo & strTemp(i) & " "
            Me.LogicalEquation.Value = strUndo
        Next i
    Else
        ' Set the logical equation equal to "Null"
        Me.LogicalEquation.Value = "( "
    End If

    ' Set logical equation expression equal to new value
End If

End Sub

Private Sub Command11_Click()

    ' Forces user to enter data for property name and description
If IsNull(Me.LPropertyName_PK.Value) Or IsNull(Me.Description.Value)
    Me!Command8.Enabled = False
    Me!Command9.Enabled = False
    Me!Command10.Enabled = False
    Me!cmbFundProp.Enabled = False
    Me!cmbLogOper.Enabled = False
    Me!LogicalEquation.Enabled = False
    MsgBox "You must provide a 'Property Name' and 'Description'
before building the logical equation."

    ' Enters a "( " be the first entry in the logical equation if it is
ElseIf IsNull(Me!LogicalEquation.Value) Then
    Me!Command8.Enabled = True
    Me!Command9.Enabled = True
    Me!Command10.Enabled = True
    Me!cmbFundProp.Enabled = True
    Me!cmbLogOper.Enabled = True
    Me!LogicalEquation.Enabled = True
    Me!LogicalEquation.Value = "( "

    ' Allows the user to enter values into the logical equation
Else
    Me!Command8.Enabled = True
    Me!Command9.Enabled = True
    Me!Command10.Enabled = True
    Me!cmbFundProp.Enabled = True
    Me!cmbLogOper.Enabled = True
    Me!LogicalEquation.Enabled = True
End If

End Sub

Private Sub Command8_Click()

    ' Adds the left bracket to the logical equation
    Me.LogicalEquation.Value = Me.LogicalEquation.Value & "( "

End Sub
Private Sub Command9_Click()

    ' Adds the right bracket to the logical equation
    Me.LogicalEquation.Value = Me.LogicalEquation.Value & ") "

End Sub
Private Sub btnDelete_Click()
On Error GoTo Err_btnDelete_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_btnDelete_Click:
    Exit Sub

Err_btnDelete_Click:
    MsgBox Err.Description
    Resume Exit_btnDelete_Click

End Sub
Private Sub Description_AfterUpdate()

    ' Forces user to enter data for property name and description
If IsNull(Me.LPropertyName_PK.Value) Or IsNull(Me.Description.Value)
    Me!Command8.Enabled = False
    Me!Command9.Enabled = False
    Me!Command10.Enabled = False
    Me!cmbFundProp.Enabled = False
    Me!cmbLogOper.Enabled = False
    Me!LogicalEquation.Enabled = False

    ' Enters a "( " be the first entry in the logical equation if it is
ElseIf IsNull(Me!LogicalEquation.Value) Then
```

```
1301      188      Me!Command8.Enabled = True
1302      189      Me!Command9.Enabled = True
1303      190      Me!Command10.Enabled = True
1304      191      Me!cmbFundProp.Enabled = True
1305      192      Me!cmbLogOper.Enabled = True
1306      193      Me!LogicalEquation.Enabled = True
1307      194      Me!LogicalEquation.Value = "( "
1308      195
1309      196    ' Allows the user to enter values into the logical equation
1310      197    Else
1311      198      Me!Command8.Enabled = True
1312      199      Me!Command9.Enabled = True
1313      200      Me!Command10.Enabled = True
1314      201      Me!cmbFundProp.Enabled = True
1315      202      Me!cmbLogOper.Enabled = True
1316      203      Me!LogicalEquation.Enabled = True
1317      204    End If
1318      205
1319      206 End Sub
1320      207
1321      208 Private Sub Form_BeforeUpdate(Cancel As Integer)
1322      209
1323      210    [LogPropTimeStamp] = Now
1324      211
1325      212 End Sub
1326      213
1327      214 Private Sub Form_Current()
1328      215
1329      216    ' Forces user to enter data for property name and description
1330      217    If IsNull(Me.LPropertyName_PK.Value) Or IsNull(Me.Description.Value)
1331      218      Me!Command8.Enabled = False
1332      219      Me!Command9.Enabled = False
1333      220      Me!Command10.Enabled = False
1334      221      Me!cmbFundProp.Enabled = False
1335      222      Me!cmbLogOper.Enabled = False
1336      223      Me!LogicalEquation.Enabled = False
1337      224
1338      225    ' Enters a "( " be the first entry in the logical equation if it is
1339      226    ElseIf IsNull(Me!LogicalEquation.Value) Then
1340      227      Me!LogicalEquation.Value = "( "
1341      228
1342      229    ' Allows the user to enter values into the logical equation
1343      230    Else
1344      231      Me!Command8.Enabled = True
1345      232      Me!Command9.Enabled = True
1346      233      Me!Command10.Enabled = True
1347      234      Me!cmbFundProp.Enabled = True
1348      235      Me!cmbLogOper.Enabled = True
1349      236      Me!LogicalEquation.Enabled = True
1350      237    End If
1351      238
1352      239 End Sub
1353      240 Private Sub LogicalEquation_Enter()
1354      241
1355      242    ' Prevents the user from modfing the logical equation directly
1356      243    MsgBox "The logical equation is only modifiable by the above logical
1357      equation builder"
1358      244    DoCmd.GoToControl "LPropertyName_PK"
1359      245
1360      246 End Sub
1361      247
1362      248 Private Sub LPropertyName_PK_AfterUpdate()
1363      249
1364      250    ' Forces user to enter data for property name and description
1365      251    If IsNull(Me.LPropertyName_PK.Value) Or IsNull(Me.Description.Value)
1366      252      Me!Command8.Enabled = False
1367      253      Me!Command9.Enabled = False
1368      254      Me!Command10.Enabled = False
1369      255      Me!cmbFundProp.Enabled = False
1370      256      Me!cmbLogOper.Enabled = False
1371      257      Me!LogicalEquation.Enabled = False
1372      258
1373      259    ' Enters a "( " be the first entry in the logical equation if it is
1374      260    ElseIf IsNull(Me!LogicalEquation.Value) Then
1375      261      Me!Command8.Enabled = True
1376      262      Me!Command9.Enabled = True
1377      263      Me!Command10.Enabled = True
1378      264      Me!cmbFundProp.Enabled = True
1379      265      Me!cmbLogOper.Enabled = True
1380      266      Me!LogicalEquation.Enabled = True
1381      267      Me!LogicalEquation.Value = "( "
1382      268
1383      269    ' Allows the user to enter values into the logical equation
1384      270    Else
1385      271      Me!Command8.Enabled = True
1386      272      Me!Command9.Enabled = True
1387      273      Me!Command10.Enabled = True
1388      274      Me!cmbFundProp.Enabled = True
1389      275      Me!cmbLogOper.Enabled = True
1390      276      Me!LogicalEquation.Enabled = True
1391      277    End If
1392      278
1393      279 End Sub
1394
1395  Form: frmMaintenanceSwitchboard
1396      Code
1397      1    Attribute VB_Name = "Form_frmMaintenanceSwitchboard"
1398      2    Attribute VB_Creatable = True
1399      3    Attribute VB_PredeclaredId = True
1400      4    Attribute VB_Exposed = False
1401      5    Option Compare Database
1402      6    Option Explicit
1403      7
1404      8    Private Sub btnExit_Click()
1405      9    On Error GoTo Err_btnExit_Click
1406      10
1407      11
1408      12      DoCmd.Close
1409      13
1410      14   Exit_btnExit_Click:
1411      15      Exit Sub
1412      16
1413      17   Err_btnExit_Click:
1414      18      MsgBox Err.Description
1415      19      Resume Exit_btnExit_Click
1416      20
1417      21
1418      22   End Sub
1419      23
1420      24   Private Sub Command0_Click()
1421      25   On Error GoTo Err_Command0_Click
1422      26
1423      27      Dim stDocName As String
1424      28      Dim stLinkCriteria As String
1425      29
1426      30      ' Close the current form
1427      31      DoCmd.Close
1428      32      ' Open specified form
1429      33      stDocName = "frmSchoolToPEF_Maint"
1430      34      DoCmd.OpenForm stDocName, , , stLinkCriteria
1431      35
1432      36   Exit_Command0_Click:
1433      37      Exit Sub
1434      38
1435      39   Err_Command0_Click:
1436      40      MsgBox Err.Description
1437      41      Resume Exit_Command0_Click
1438      42
1439      43   End Sub
1440      44
1441      45   Private Sub Command12_Click()
1442      46   On Error GoTo Err_Command12_Click
1443      47
1444      48      Dim stDocName As String
1445      49      Dim stLinkCriteria As String
1446      50
1447      51      ' Close the current form
1448      52      DoCmd.Close
1449      53      ' Open specified form
1450      54      stDocName = "frmSchools"
1451      55      DoCmd.OpenForm stDocName, , , stLinkCriteria
1452      56
1453      57   Exit_Command12_Click:
1454      58      Exit Sub
1455      59
1456      60   Err_Command12_Click:
1457      61      MsgBox Err.Description
1458      62      Resume Exit_Command12_Click
1459      63
1460      64   End Sub
1461      65
1462      66   Private Sub Command14_Click()
1463      67   On Error GoTo Err_Command14_Click
1464      68
1465      69      Dim stDocName As String
1466      70      Dim stLinkCriteria As String
1467      71
1468      72      ' Close the current form
1469      73      DoCmd.Close
1470      74      ' Open specified form
1471      75      stDocName = "frmPEF"
1472      76      DoCmd.OpenForm stDocName, , , stLinkCriteria
1473      77
1474      78   Exit_Command14_Click:
1475      79      Exit Sub
1476      80
1477      81   Err_Command14_Click:
1478      82      MsgBox Err.Description
1479      83      Resume Exit_Command14_Click
1480      84
1481      85   End Sub
1482      86
1483      87   Private Sub Command2_Click()
1484      88   On Error GoTo Err_Command2_Click
1485      89
1486      90      Dim stDocName As String
1487      91      Dim stLinkCriteria As String
1488      92
1489      93      ' Close the current form
1490      94      DoCmd.Close
1491      95      ' Open specified form
1492      96      stDocName = "frmFundamentalProperty"
1493      97      DoCmd.OpenForm stDocName, , , stLinkCriteria
1494      98
1495      99   Exit_Command2_Click:
1496      100     Exit Sub
1497      101
1498      102  Err_Command2_Click:
1499      103     MsgBox Err.Description
1500      104     Resume Exit_Command2_Click
1501      105
1502      106  End Sub
1503      107
1504      108  Private Sub Command4_Click()
1505      109  On Error GoTo Err_Command4_Click
1506      110
1507      111     Dim stDocName As String
1508      112     Dim stLinkCriteria As String
1509      113
1510      114     ' Close the current form
1511      115     DoCmd.Close
1512      116     ' Open specified form
1513      117     stDocName = "frmLogicalProperty"
1514      118     DoCmd.OpenForm stDocName, , , stLinkCriteria
1515      119
1516      120  Exit_Command4_Click:
1517      121     Exit Sub
1518      122
1519      123  Err_Command4_Click:
1520      124     MsgBox Err.Description
1521      125     Resume Exit_Command4_Click
1522      126
1523      127  End Sub
1524      128
1525      129  Private Sub Command6_Click()
1526      130  On Error GoTo Err_Command6_Click
1527      131
1528      132     Dim stDocName As String
1529      133     Dim stLinkCriteria As String
1530      134
1531      135     ' Close the current form
1532      136     DoCmd.Close
1533      137     ' Open specified form
1534      138     stDocName = "frmSchoolToPropertyMaint"
1535      139     DoCmd.OpenForm stDocName, , , stLinkCriteria
1536      140
1537      141  Exit_Command6_Click:
1538      142     Exit Sub
1539      143
1540      144  Err_Command6_Click:
1541      145     MsgBox Err.Description
1542      146     Resume Exit_Command6_Click
1543      147
1544      148  End Sub
1545      149
1546      150  Private Sub Command8_Click()
1547      151  On Error GoTo Err_Command8_Click
1548      152
1549      153     Dim stDocName As String
1550      154     Dim stLinkCriteria As String
1551      155
1552      156     ' Close the current form
1553      157     DoCmd.Close
1554      158     ' Open specified form
1555      159     stDocName = "frmRDM_Main_Switchboard"
1556      160     DoCmd.OpenForm stDocName, , , stLinkCriteria
1557      161
1558      162  Exit_Command8_Click:
1559      163     Exit Sub
1560      164
```

```
    165 Err_Command8_Click:
    166   MsgBox Err.Description
    167   Resume Ext_Command8_Click
    168
    169 End Sub

Form: frmPEF
   Code
     1  Attribute VB_Name = "Form_frmPEF"
     2  Attribute VB_Creatable = True
     3  Attribute VB_PredeclaredId = True
     4  Attribute VB_Exposed = False
     5  Option Compare Database
     6  Option Explicit
     7
     8  Private Sub cmbPefFind_AfterUpdate()
     9
    10    Dim R As Recordset
    11    Set R = Me.RecordsetClone
    12    R.FindFirst "[PEF_PK] = " & Chr(34) & Me![cmbPefFind] & Chr(34)
    13    Me.Bookmark = R.Bookmark
    14    Me![cmbPefFind] = Null
    15    Me.PEF_PK.SetFocus
    16
    17  End Sub
    18
    19  Private Sub Form_BeforeUpdate(Cancel As Integer)
    20
    21    [PEF_TimeStamp] = Now
    22
    23  End Sub
    24
    25  Private Sub Form_Current()
    26
    27    Dim strSQL As String
    28
    29    ' Update the PEF List
    30    strSQL = "SELECT DISTINCTROW [MARINE].[PEF]FROM MARINE LEFT JOIN PEF
ON [MARINE].[PEF] = [PEF].[PEF_PK]WHERE ([PEF].[PEF_PK] Is Null)ORDER BY
    31    Me.PEF_PK.RowSource = strSQL
    32
    33  End Sub
    34
    35  Private Sub PEF_PK_AfterUpdate()
    36
    37    Dim strSQL As String
    38
    39    ' Update the PEF List
    40    strSQL = "SELECT DISTINCTROW [MARINE].[PEF]FROM MARINE LEFT JOIN PEF
ON [MARINE].[PEF] = [PEF].[PEF_PK]WHERE ([PEF].[PEF_PK] Is Null)ORDER BY
    41    Me.PEF_PK.RowSource = strSQL
    42
    43  End Sub
    44  Private Sub btnDelete_Click()
    45  On Error GoTo Err_btnDelete_Click
    46
    47
    48    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    49    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70
    50
    51  Exit_btnDelete_Click:
    52    Exit Sub
    53
    54  Err_btnDelete_Click:
    55    MsgBox Err.Description
    56    Resume Exit_btnDelete_Click
    57
    58  End Sub
    59  Private Sub btnClose_Click()
    60  On Error GoTo Err_btnClose_Click
    61
    62
    63    DoCmd.Close
    64    DoCmd.OpenForm "frmMaintenanceSwitchboard"
    65
    66  Exit_btnClose_Click:
    67    Exit Sub
    68
    69  Err_btnClose_Click:
    70    MsgBox Err.Description
    71    Resume Exit_btnClose_Click
    72
    73  End Sub
    74
Form: frmPrepareAndExecuteSolver
   Code
     1  Attribute VB_Name = "Form_frmPrepareAndExecuteSolver"
     2  Attribute VB_Creatable = True
     3  Attribute VB_PredeclaredId = True
     4  Attribute VB_Exposed = False
     5  Option Compare Database
     6  Option Explicit
     7
     8  Private Sub btnAMPL_Click()
     9  On Error GoTo Err_btnAMPL_Click
    10
    11    AmplData
    12
    13    Dim stAppName As String
    14
    15    'stAppName = "C:\ToBeRdm\Amplwin\amplwin.exe"
    16    stAppName = "C:\ToBeRdm\Amplwin\amp.bat"
    17    Call Shell(stAppName, 0)
    18
    19    'MsgBox "Enter into the command line: 'include rdm.run'"
    20
    21  Exit_btnAMPL_Click:
    22    Exit Sub
    23
    24  Err_btnAMPL_Click:
    25    MsgBox Err.Description
    26    Resume Exit_btnAMPL_Click
    27
    28  End Sub
    29
    30  Private Sub btnAmplPlus_Click()
    31  On Error GoTo Err_btnAmplPlus_Click
    32
    33    AmplData
    34
    35    Dim stAppName As String
    36
    37    stAppName = "C:\AMPLPLUS\AMPLPLUS.EXE rdm.amp"
    38    Call Shell(stAppName, 1)
    39
    40  Exit_btnAmplPlus_Click:
    41    Exit Sub
    42
    43  Err_btnAmplPlus_Click:
    44    MsgBox Err.Description
    45    Resume Exit_btnAmplPlus_Click
    46
    47  End Sub
    48
    49
    50  Private Sub btnReturnPrevious_Click()
    51  On Error GoTo Err_btnReturnPrevious_Click
    52
    53
    54    Dim stDocName As String
    55    Dim stLinkCriteria As String
    56
    57    ' Close the current form
    58    DoCmd.Close
    59    ' Open specified form
    60    stDocName = "frmPreprocessing&ExecutionSwitchboard"
    61    DoCmd.OpenForm stDocName, , , stLinkCriteria
    62
    63  Exit_btnReturnPrevious_Click:
    64    Exit Sub
    65
    66  Err_btnReturnPrevious_Click:
    67    MsgBox Err.Description
    68    Resume Exit_btnReturnPrevious_Click
    69
    70
    71  End Sub
    72
    73  Private Sub Command4_Click()
    74    Application.SetOption "Confirm Action Queries", False
    75    DoCmd.SetWarnings (False)
    76    DoCmd.RunSQL "DELETE * FROM AMPL_RESULT;"
    77    Ampl_Result
    78    Application.SetOption "Confirm Action Queries", True
    79    DoCmd.SetWarnings (True)
    80
    81  End Sub
    82
    83  Private Sub Command7_Click()
    84  On Error GoTo Err_Command7_Click
    85
    86    Dim stDocName As String
    87    Dim stLinkCriteria As String
    88
    89    ' Close the current form
    90    DoCmd.Close
    91    ' Open specified form
    92    stDocName = "frmAnalyzeResult"
    93    DoCmd.OpenForm stDocName, , , stLinkCriteria
    94
    95  Exit_Command7_Click:
    96    Exit Sub
    97
    98  Err_Command7_Click:
    99    MsgBox Err.Description
   100    Resume Exit_Command7_Click
   101
   102 End Sub
   103
Form: frmPreprocessing&ExecutionSwitchboard
   Code
     1  Attribute VB_Name = "Form_frmPreprocessing&ExecutionSwitchboard"
     2  Attribute VB_Creatable = True
     3  Attribute VB_PredeclaredId = True
     4  Attribute VB_Exposed = False
     5  Option Compare Database
     6  Option Explicit
     7
     8  Private Sub btnExit_Click()
     9  On Error GoTo Err_btnExit_Click
    10
    11
    12    DoCmd.Close
    13
    14  Exit_btnExit_Click:
    15    Exit Sub
    16
    17  Err_btnExit_Click:
    18    MsgBox Err.Description
    19    Resume Exit_btnExit_Click
    20
    21
    22  End Sub
    23
    24  Private Sub btnReclassification_Click()
    25  On Error GoTo Err_btnReclassification_Click
    26
    27    Dim stDocName As String
    28    Dim stLinkCriteria As String
    29
    30    ' Close the current form
    31    DoCmd.Close
    32    ' Open specified form
    33    stDocName = "frmReclassification"
    34    DoCmd.OpenForm stDocName, , , stLinkCriteria
    35
    36  Exit_btnReclassification_Click:
    37    Exit Sub
    38
    39  Err_btnReclassification_Click:
    40    MsgBox Err.Description
    41    Resume Exit_btnReclassification_Click
    42
    43
    44  End Sub
    45
    46  Private Sub btnScrubMarineData_Click()
    47  On Error GoTo Err_btnScrubMarineData_Click
    48
    49    Dim stDocName As String
    50    Dim stLinkCriteria As String
    51
    52    ' Close the current form
    53    DoCmd.Close
    54    ' Open specified form
    55    stDocName = "frmScrubMarine"
    56    DoCmd.OpenForm stDocName, , , stLinkCriteria
    57
    58  Exit_btnScrubMarineData_Click:
    59    Exit Sub
    60
    61  Err_btnScrubMarineData_Click:
    62    MsgBox Err.Description
    63    Resume Exit_btnScrubMarineData_Click
    64
    65  End Sub
    66
    67  Private Sub btnSpecialAssignment_Click()
    68  On Error GoTo Err_btnSpecialAssignment_Click
    69
```

```
    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmSpecialAssignment"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_btnSpecialAssignment_Click:
    Exit Sub

Err_btnSpecialAssignment_Click:
    MsgBox Err.Description
    Resume Exit_btnSpecialAssignment_Click



End Sub

Private Sub Command2_Click()
On Error GoTo Err_Command2_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmclassQuotaPenaltyAndFit"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command2_Click:
    Exit Sub

Err_Command2_Click:
    MsgBox Err.Description
    Resume Exit_Command2_Click

End Sub
Private Sub Command4_Click()
    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)
    DoCmd.RunSQL "DELETE * FROM AMPL_RESULT;"
    Ampl_Result
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

End Sub
Private Sub Command5_Click()
On Error GoTo Err_Command5_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmPrepareAndExecuteSolver"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command5_Click:
    Exit Sub

Err_Command5_Click:
    MsgBox Err.Description
    Resume Exit_Command5_Click

End Sub
Private Sub Command8_Click()
On Error GoTo Err_Command8_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmRDM_Main_Switchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command8_Click:
    Exit Sub

Err_Command8_Click:
    MsgBox Err.Description
    Resume Exit_Command8_Click

End Sub

Form: frmRDM_Main_Switchboard
  Code
    Attribute VB_Name = "Form_frmRDM_Main_Switchboard"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit
    Private Sub Command0_Click()

    On Error GoTo Err_Command0_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmImport/ExportSwitchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Command0_Click:
        Exit Sub

    Err_Command0_Click:
        MsgBox Err.Description
        Resume Exit_Command0_Click

    End Sub
    Private Sub Command2_Click()
    On Error GoTo Err_Command2_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmMaintenanceSwitchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Command2_Click:
        Exit Sub

    Err_Command2_Click:
        MsgBox Err.Description
        Resume Exit_Command2_Click

    End Sub
    Private Sub Command4_Click()
    On Error GoTo Err_Command4_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmPreprocessing&ExecutionSwitchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Command4_Click:
        Exit Sub

    Err_Command4_Click:
        MsgBox Err.Description
        Resume Exit_Command4_Click

    End Sub
    Private Sub Command6_Click()
    On Error GoTo Err_Command6_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmAnalyzeResult"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Command6_Click:
        Exit Sub

    Err_Command6_Click:
        MsgBox Err.Description
        Resume Exit_Command6_Click

    End Sub
    Private Sub Command8_Click()
    On Error GoTo Err_Command8_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmGenerateReport"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Command8_Click:
        Exit Sub

    Err_Command8_Click:
        MsgBox Err.Description
        Resume Exit_Command8_Click

    End Sub
    Private Sub btnExit_Click()
    On Error GoTo Err_btnExit_Click


        DoCmd.Close

    Exit_btnExit_Click:
        Exit Sub

    Err_btnExit_Click:
        MsgBox Err.Description
        Resume Exit_btnExit_Click

    End Sub

Form: frmReclassification
  Code
    Attribute VB_Name = "Form_frmReclassification"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub cmbClassNumber_AfterUpdate()

        Dim strConvert As String

        Me.txtReportDate = Me.cmbClassNumber.Column(2)
        Me.txtMCC = Me.cmbClassNumber.Column(4)

        ' Convert the fiscal year to a two digit number for RD3 File
        strConvert = CStr(Me.cmbClassNumber.Column(3))
        strConvert = Right(strConvert, 2)
        Me.txtFY = Cint(strConvert)

        Me.txtAssignmentType = "R"

    End Sub

    Private Sub cmbClassNumber_Enter()

        Dim strSQL As String

        ' This query finds the class numbers, report dates and class
convening dates associated with the chosen school
        strSQL = "SELECT BNA_EXTRACT.ClassNumber_PK,
BNA_EXTRACT.ClassConvDate, BNA_EXTRACT.ReportDate,
BNA_EXTRACT.FiscalYear_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT INNER JOIN
SCH_TGT_MOS ON (BNA_EXTRACT.CourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK) AND (BNA_EXTRACT.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK) WHERE BNA_EXTRACT.CourseNumber_PK =
Forms!frmReclassification!cmbCourseNumber AND SCH_TGT_MOS.AMOS_FK =
Forms!frmReclassification!txtAMOS;"
        Me.cmbClassNumber.RowSource = strSQL

    End Sub
```

```

    Private Sub cmbCourseNumber_AfterUpdate()

        Me.txtAMOS = Me!cmbCourseNumber.Column(1)

    End Sub

    Private Sub cmbSSN_AfterUpdate()

        Me.txtSOI = Me!cmbSSN.Column(1)
        Me.txtGradDate = Me!cmbSSN.Column(2)

    End Sub

    Private Sub cmbSSN_Enter()

        Dim strSQL As String

        strSQL = "SELECT DISTINCTROW MARINE.SSN_PK, MARINE.SOI,
    MARINE.GradDate FROM MARINE LEFT JOIN ASSIGNMENT ON MARINE.SSN_PK =
    ASSIGNMENT.SSN_FK WHERE (((ASSIGNMENT.SSN_FK) Is Null));"
        Me.cmbSSN.RowSource = strSQL

    End Sub

    Private Sub cmbSSNFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[SSN_FK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbSSNFind] = Null
        Me.cmbSSN.SetFocus

    End Sub

    Private Sub Delete_Click()
    On Error GoTo Err_Delete_Click

        DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
        DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

    Exit_Delete_Click:
        Exit Sub

    Err_Delete_Click:
        MsgBox Err.Description
        Resume Exit_Delete_Click

    End Sub

    Private Sub Close_Click()
    On Error GoTo Err_Close_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close current form
        DoCmd.Close

        ' Open specified form
        stDocName = "frmPreprocessing&ExecutionSwitchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Close_Click:
        Exit Sub

    Err_Close_Click:
        MsgBox Err.Description
        Resume Exit_Close_Click

    End Sub


Form: frmSchoolAssignments
    Code
    Attribute VB_Name = "Form_frmSchoolAssignments"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit
    Private Sub Close_Click()
    On Error GoTo Err_Close_Click

    Dim stDocName As String
    Dim stLinkCriteria As String
    Dim db1 As Database

    Set db1 = CurrentDb()

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmAnalyzeResult"
    DoCmd.OpenForm stDocName, , , stLinkCriteria
    Forms!frmAnalyzeResult.SetFocus


    Exit_Close_Click:
        Exit Sub

    Err_Close_Click:
        MsgBox Err.Description
        Resume Exit_Close_Click

    End Sub

    Private Sub Form_Current()

        Dim strSQL As String
        Dim rec As Recordset
        Dim db1 As Database

        Set db1 = CurrentDb()

        ' Calculates the total quota, and number of schools for the run.
        strSQL = "SELECT Sum(SumOfQuota) AS TotalQuota, Count(SumOfQuota) AS
    NumberOfSchools FROM qryTotalQuotaForRun;"
        Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        Me.txtTotalQuota = rec!TotalQuota
        Me.txtNumberOfSchools = rec!NumberOfSchools

        ' Calculates the total number of assignements for the run.
        strSQL = "SELECT Sum(CountOfFiscalYear_PK) AS TotalAssigned FROM
    qryTotalQuotaFilledForRun;"
        Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        Me.txtTotalAssigned = rec!TotalAssigned

        Me.txtTotalPercentFill = Int((Me.txtTotalAssigned / Me.txtTotalQuota)

    End Sub

Form: frmSchools
    Code
    Attribute VB_Name = "Form_frmSchools"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub AMOS_PK_AfterUpdate()
        Dim rec As Recordset
        Dim db1 As Database
        Dim strSQL As String

        ' Update the drop down list for AMOS
        strSQL = "SELECT TargetMOS_PK FROM
    qryListUnusedTargetMOSFromTARGET_MOS WHERE TCourseNumber_PK = " &
        Me.AMOS_PK.RowSource = strSQL
        Me.lstTargetMOS_FK.RowSource = strSQL

        ' Update the AMOS_FK in the child relation SCH_TGT_MOS
        'Forms!frmSchools!subfrmSchools!AMOS_FK.Value = Me.AMOS_PK.Value

        ' Update the TargetMOS_FK in the child relation SCH_TGT_MOS
        'Set db1 = CurrentDb()
        'Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        'Forms!frmSchools!subfrmSchools!TargetMOS_FK.Value = rec!TargetMOS_PK

        ' Input the correct MCC value for the selected course
        strSQL = "SELECT MCC FROM TARGET_MOS WHERE TCourseNumber_PK = " &
    Me.SCourseNumber_PK.Value & ""
        Set db1 = CurrentDb()
        Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        Me.MCC.Value = rec!MCC

        ' Update the TCourseNumber_FK in the child relation SCH_TGT_MOS
        'Forms!frmSchools!subfrmSchools!TCourseNumber_FK.Value =
    Me.SCourseNumber_PK.Value

    End Sub

    Private Sub btnClear_Click()

        Dim db1 As Database
        Dim strSQL As String
        Dim i As Integer
        Dim rec As Recordset

        Set db1 = CurrentDb()

        strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
    TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
    SCH_TGT_MOS.TargetMOS_FK AND TARGET_MOS.TCourseNumber_PK =
    SCH_TGT_MOS.TCourseNumber_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
    Null)AND((SCH_TGT_MOS.TCourseNumber_FK) Is Null))ORDER BY
    TCourseNumber_PK;"
        Set rec = db1.OpenRecordset(strSQL, dbOpenDynaset)

        If rec.EOF = False Then
          rec.MoveLast
          rec.MoveFirst
        End If

        ' Clear unused course numbers from the target_mos table

        If rec.RecordCount > 0 Then
          For i = 1 To rec.RecordCount
            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False
            DoCmd.SetWarnings (False)

            strSQL = "DELETE * FROM TARGET_MOS WHERE
    TARGET_MOS.TCourseNumber_PK = " & rec!TCourseNumber_PK & ""
            db1.Execute strSQL

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True
            DoCmd.SetWarnings (True)
            rec.MoveNext
          Next i
        End If

        ' Update the Course Number List
        strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
    TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
    SCH_TGT_MOS.TargetMOS_FK AND TARGET_MOS.TCourseNumber_PK =
    SCH_TGT_MOS.TCourseNumber_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
    Null)AND((SCH_TGT_MOS.TCourseNumber_FK) Is Null))ORDER BY
    TCourseNumber_PK;"
        Me.SCourseNumber_PK.RowSource = strSQL

    End Sub

    Private Sub btnResetAllPenalties_Click()


        ' Disables the action query confirmation message
        Application.SetOption "Confirm Action Queries", False
        DoCmd.SetWarnings (False)

        DoCmd.RunSQL "UPDATE SCHOOL SET PenaltyFactor = 24;"

        ' Enables the action query confirmation message
        Application.SetOption "Confirm Action Queries", True
        DoCmd.SetWarnings (True)

        cmbPenaltyView = "1x"

    End Sub

    Private Sub cmbCourseNumberFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[SCourseNumber_PK] = " & Chr(34) &
    Me![cmbCourseNumberFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbCourseNumberFind] = Null
        Me.SCourseNumber_PK.SetFocus

    End Sub
    Private Sub cmbPenaltyView_AfterUpdate()
```

```
    Select Case cmbPenaltyView
        Case "4x"
            txtPenaltyFactor = 4 * 24
        Case "3x"
            txtPenaltyFactor = 3 * 24
        Case "2x"
            txtPenaltyFactor = 2 * 24
        Case "1x"
            txtPenaltyFactor = 1 * 24
        Case "1/2x"
            txtPenaltyFactor = 24 / 2
        Case "1/3x"
            txtPenaltyFactor = 24 / 3
        Case "1/4x"
            txtPenaltyFactor = 24 / 4
    End Select

    ' Save the update
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , , 1
End Sub
Private Sub Command14_Click()

    Dim strSQL As String

    ' Disables the action query confirmation message
    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)

    DoCmd.RunSQL "INSERT INTO TARGET_MOS ( TCourseNumber_PK,
TargetMOS_PK, MCC ) SELECT DISTINCT BNA_EXTRACT.CourseNumber_PK,
BNA_EXTRACT.TargetMOS_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT;"

    ' Enables the action query confirmation message
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK AND TARGET_MOS.TCourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
Null) AND ((SCH_TGT_MOS.TCourseNumber_FK) Is Null))ORDER BY
TCourseNumber_PK;"
    Me.SCourseNumber_PK.RowSource = strSQL

End Sub


Private Sub Command39_Click()
    Dim frm As Form, ctl As Control
    Dim varItm As Variant, intI As Integer
    Dim strSQL As String

    ' Ensures the current record is save to the SCHOOL table
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , , 1

    ' This code enters the selected Target MOS's into the SCH_TGT_MOS
    Set frm = Forms!frmSchools
    Set ctl = frm!lstTargetMOS_FK
    For Each varItm In ctl.ItemsSelected
        For intI = 0 To ctl.ColumnCount - 1

            ' This puts the value found in the list box into a text box,
making it readable by the query
            Forms!frmSchools!TargetValue.Value = ctl.Column(intI, varItm)

            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False

            ' Enters the value into the SCH_TGT_MOS table
            DoCmd.OpenQuery "qryUpdateSCH_TGT_MOS"

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True

        Next intI
    Next varItm

    ' Update the Target MOS list
    strSQL = "SELECT [TargetMOS_FK] FROM [SCH_TGT_MOS] WHERE
[SCourseNumber_FK] = '" & Forms!frmSchools!SCourseNumber_PK.Value & "'" &
" AND AMOS_FK = '" & Forms!frmSchools!AMOS_PK.Value & "'"
    Forms!frmSchools!lstTargetMOS_FK_History.RowSource = strSQL

    ' Update the Potential Target MOS list
    strSQL = "SELECT TargetMOS_PK FROM
qryListUnusedTargetMOSFromTARGET_MOS WHERE TCourseNumber_PK = '" &
    Me.lstTargetMOS_FK.RowSource = strSQL

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
Null))ORDER BY TCourseNumber_PK;"
    Me.SCourseNumber_PK.RowSource = strSQL

    ' Update time stamp
    [Sch_TimeStamp] = Now

End Sub


Private Sub Command40_Click()
    Dim frm As Form, ctl As Control
    Dim varItm As Variant, intI As Integer
    Dim strSQL As String

    ' This code Deletes the selected Target MOS's and its associated
entries from the SCH_TGT_MOS table
    Set frm = Forms!frmSchools
    Set ctl = frm!lstTargetMOS_FK_History
    For Each varItm In ctl.ItemsSelected
        For intI = 0 To ctl.ColumnCount - 1

            ' This puts the value found in the list box into a text box,
making it readable by the query
            Forms!frmSchools!TargetValue.Value = ctl.Column(intI, varItm)

            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False

            ' Deletes the record associated with the specified value from
the SCH_TGT_MOS table
            DoCmd.OpenQuery "qryDeleteSCH_TGT_MOS"

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True

        Next intI
    Next varItm

    ' Update the Target MOS list
    strSQL = "SELECT [TargetMOS_FK] FROM [SCH_TGT_MOS] WHERE
[SCourseNumber_FK] = '" & Forms!frmSchools!SCourseNumber_PK.Value & "'" &
" AND AMOS_FK = '" & Forms!frmSchools!AMOS_PK.Value & "'"
    Forms!frmSchools!lstTargetMOS_FK_History.RowSource = strSQL

    ' Update the Potential Target MOS list
    strSQL = "SELECT TargetMOS_PK FROM
qryListUnusedTargetMOSFromTARGET_MOS WHERE TCourseNumber_PK = '" &
    Me.lstTargetMOS_FK.RowSource = strSQL

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
Null))ORDER BY TCourseNumber_PK;"
    Me.SCourseNumber_PK.RowSource = strSQL

    ' Update time stamp
    [Sch_TimeStamp] = Now

End Sub


Private Sub Form_BeforeUpdate(Cancel As Integer)

    [Sch_TimeStamp] = Now

End Sub


Private Sub Form_Current()
    Dim strSQL As String

    ' Update the Potential Target MOS list
    strSQL = "SELECT TargetMOS_PK FROM
qryListUnusedTargetMOSFromTARGET_MOS WHERE TCourseNumber_PK = '" &
    Me.lstTargetMOS_FK.RowSource = strSQL

    ' Update the Target MOS List
    strSQL = "SELECT [TargetMOS_FK] FROM [SCH_TGT_MOS] WHERE
[SCourseNumber_FK] = '" & Me.SCourseNumber_PK.Value & "'" & " AND AMOS_FK
= '" & Me.AMOS_PK.Value & "'"
    Me.lstTargetMOS_FK_History.RowSource = strSQL

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK AND TARGET_MOS.TCourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
Null) AND ((SCH_TGT_MOS.TCourseNumber_FK) Is Null))ORDER BY
TCourseNumber_PK;"
    Me.SCourseNumber_PK.RowSource = strSQL


    ' Forces user to enter data for course number, before the Assigned
    If IsNull(Me.SCourseNumber_PK.Value) Then
        Me!AMOS_PK.Enabled = False
    Else
    ' Allows the user to enter an assigned MOS
        Me!AMOS_PK.Enabled = True
    End If

    ' Prevents the user from accidentally changing a course number
    If IsNull(SCourseNumber_PK) Then
        Me!SCourseNumber_PK.Locked = False
    Else
        Me!SCourseNumber_PK.Locked = True
    End If

        ' Prevents the user from accidentally changing an AMOS
    If IsNull(AMOS_PK) Then
        Me!AMOS_PK.Locked = False
    Else
        Me!AMOS_PK.Locked = True
    End If

    ' Update the penalty factor view
    Select Case txtPenaltyFactor
        Case 96
            cmbPenaltyView = "4x"
        Case 72
            cmbPenaltyView = "3x"
        Case 48
            cmbPenaltyView = "2x"
        Case 24
            cmbPenaltyView = "1x"
        Case 12
            cmbPenaltyView = "1/2x"
        Case 8
            cmbPenaltyView = "1/3x"
        Case 6
            cmbPenaltyView = "1/4x"
    End Select


End Sub


Private Sub SCourseNumber_PK_AfterUpdate()

    Dim rec As Recordset
    Dim db1 As Database
    Dim strSQL As String

    ' Update the drop down list for AMOS
    strSQL = "SELECT TargetMOS_PK FROM TARGET_MOS WHERE TCourseNumber_PK
= '" & Me.SCourseNumber_PK.Value & "'"
    Me.AMOS_PK.RowSource = strSQL
    Me.lstTargetMOS_FK.RowSource = strSQL

    ' Allows the user to enter an assigned MOS
    Me!AMOS_PK.Enabled = True

End Sub


Private Sub btnDelete_Click()
On Error GoTo Err_btnDelete_Click


    ' Disables the action query confirmation message
    Application.SetOption "Confirm Action Queries", False

    ' Deletes the record associated with the specified value from the
SCH_TGT_MOS table
    DoCmd.OpenQuery "qryDeleteSCH_TGT_MOSEntry"

    ' Deletes the record associated with the specified value from the
SCHOOL table
    DoCmd.OpenQuery "qryDeleteSCHOOLEntry"

    ' Enables the action query confirmation message
```

```
    336     Application.SetOption "Confirm Action Queries", True
    337
    338     DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    339     DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70
    340
341 Exit_btnDelete_Click:
    342     Exit Sub
    343
344 Err_btnDelete_Click:
    345     MsgBox Err.Description
    346     Resume Exit_btnDelete_Click
    347
348 End Sub
349 Private Sub btnClose_Click()
350 On Error GoTo Err_btnClose_Click
    351
    352
    353     DoCmd.Close
    354     DoCmd.OpenForm "frmMaintenanceSwitchboard"
    355
356 Exit_btnClose_Click:
    357     Exit Sub
    358
359 Err_btnClose_Click:
    360     MsgBox Err.Description
    361     Resume Exit_btnClose_Click
    362
363 End Sub
    1
Form: frmSchoolToPEF_Maint
    Code
    1     Attribute VB_Name = "Form_frmSchoolToPEF_Maint"
    2     Attribute VB_Creatable = True
    3     Attribute VB_PredeclaredId = True
    4     Attribute VB_Exposed = False
    5     Option Compare Database
    6     Option Explicit
    7
    8     Private Sub btnClose_Click()
    9     On Error GoTo Err_btnClose_Click
    10
    11
    12      DoCmd.Close
    13      DoCmd.OpenForm "frmMaintenanceSwitchboard"
    14
    15  Exit_btnClose_Click:
    16      Exit Sub
    17
    18  Err_btnClose_Click:
    19      MsgBox Err.Description
    20      Resume Exit_btnClose_Click
    21
    22
    23  End Sub
    24
    25  Private Sub btnDelete_Click()
    26  On Error GoTo Err_btnDelete_Click
    27
    28
    29      DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    30      DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70
    31
    32  Exit_btnDelete_Click:
    33      Exit Sub
    34
    35  Err_btnDelete_Click:
    36      MsgBox Err.Description
    37      Resume Exit_btnDelete_Click
    38
    39  End Sub
    40
    41  Private Sub cmbCourseNumberFind_AfterUpdate()
    42
    43      Dim R As Recordset
    44      Set R = Me.RecordsetClone
    45      R.FindFirst "[SCourseNumber_PK] = " & Chr(34) &
Me![cmbCourseNumberFind] & Chr(34)
    46      Me.Bookmark = R.Bookmark
    47      Me![cmbCourseNumberFind] = Null
    48      Me.SCourseNumber_PK.SetFocus
    49
    50  End Sub
    51
    52  Private Sub Form_Current()
    53      Dim strSQL As String
    54
    55      ' Update the Potential PEF list
    56      strSQL = "SELECT DISTINCTROW [PEF].[PEF_PK] FROM PEF LEFT JOIN
qrySch_Pef ON [PEF].[PEF_PK] = [qrySch_Pef].[PEF_FK] WHERE
([qrySch_Pef].[PEF_FK] Is Null) ORDER BY [PEF].[PEF_PK];"
    57      Me.ListUnselectedPEF.RowSource = strSQL
    58
    59      ' Update the Selected PEF List
    60      strSQL = "SELECT [PEF_FK] FROM qrySch_Pef;"
    61      Me.ListSelectedPEF.RowSource = strSQL
    62
    63  End Sub
    64
    65  Private Sub LeftArrow_Click()
    66      Dim frm As Form, ctl As Control
    67      Dim varItm As Variant, intI As Integer
    68      Dim strSQL As String
    69
    70      ' Ensures the current record is save to the SCHOOL table
    71      DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, ,
    72
    73      ' This code enters the selected Target MOS's into the SCH_TGT_MOS
    74      Set frm = Forms!frmSchoolToPEF_Maint
    75      Set ctl = frm!LstUnselectedPEF
    76      For Each varItm In ctl.ItemsSelected
    77          For intI = 0 To ctl.ColumnCount - 1
    78
    79              ' This puts the value found in the list box into a text box,
making it readable by the query
    80              Forms!frmSchoolToPEF_Maint!PEFValue.Value = ctl.Column(intI,
    81
    82              ' Disables the action query confirmation message
    83              Application.SetOption "Confirm Action Queries", False
    84
    85              ' Enters the value into the SCH_TGT_MOS table
    86              DoCmd.OpenQuery "qryUpdateSCH_PEF"
    87
    88              ' Enables the action query confirmation message
    89              Application.SetOption "Confirm Action Queries", True
    90
    91          Next intI
    92      Next varItm
    93
    94      ' Update the Potential PEF list
    95      strSQL = "SELECT DISTINCTROW [PEF].[PEF_PK] FROM PEF LEFT JOIN
qrySch_Pef ON [PEF].[PEF_PK] = [qrySch_Pef].[PEF_FK] WHERE
([qrySch_Pef].[PEF_FK] Is Null) ORDER BY [PEF].[PEF_PK];"
    96      Me.ListUnselectedPEF.RowSource = strSQL
    97
    98      ' Update the Selected PEF List
    99      strSQL = "SELECT [PEF_FK] FROM qrySch_Pef;"
    100     Me.ListSelectedPEF.RowSource = strSQL
    101
102 End Sub
    103
104 Private Sub RightArrow_Click()
    105     Dim frm As Form, ctl As Control
    106     Dim varItm As Variant, intI As Integer
    107     Dim strSQL As String
    108
    109     ' This code Deletes the selected Target MOS's and its associated
entries from the SCH_TGT_MOS table
    110     Set frm = Forms!frmSchoolToPEF_Maint
    111     Set ctl = frm!LstSelectedPEF
    112     For Each varItm In ctl.ItemsSelected
    113         For intI = 0 To ctl.ColumnCount - 1
    114
    115             ' This puts the value found in the list box into a text box,
making it readable by the query
    116             Forms!frmSchoolToPEF_Maint!PEFValue.Value = ctl.Column(intI,
    117
    118             ' Disables the action query confirmation message
    119             Application.SetOption "Confirm Action Queries", False
    120
    121             ' Deletes the record associated with the specified value from
the SCH_TGT_MOS table
    122             DoCmd.OpenQuery "qryDeleteSCH_PEF"
    123
    124             ' Enables the action query confirmation message
    125             Application.SetOption "Confirm Action Queries", True
    126
    127         Next intI
    128     Next varItm
    129
    130     ' Update the Potential PEF list
    131     strSQL = "SELECT DISTINCTROW [PEF].[PEF_PK] FROM PEF LEFT JOIN
qrySch_Pef ON [PEF].[PEF_PK] = [qrySch_Pef].[PEF_FK] WHERE
([qrySch_Pef].[PEF_FK] Is Null) ORDER BY [PEF].[PEF_PK];"
    132     Me.ListUnselectedPEF.RowSource = strSQL
    133
    134     ' Update the Selected PEF List
    135     strSQL = "SELECT [PEF_FK] FROM qrySch_Pef;"
    136     Me.ListSelectedPEF.RowSource = strSQL
    137
138 End Sub
    139
Form: frmSchoolToPropertyMaint
    Code
    1     Attribute VB_Name = "Form_frmSchoolToPropertyMaint"
    2     Attribute VB_Creatable = True
    3     Attribute VB_PredeclaredId = True
    4     Attribute VB_Exposed = False
    5     Option Compare Database
    6     Option Explicit
    7
    8     Private Sub btnClose_Click()
    9     On Error GoTo Err_btnClose_Click
    10
    11
    12      DoCmd.Close
    13      DoCmd.OpenForm "frmMaintenanceSwitchboard"
    14
    15  Exit_btnClose_Click:
    16      Exit Sub
    17
    18  Err_btnClose_Click:
    19      MsgBox Err.Description
    20      Resume Exit_btnClose_Click
    21
    22  End Sub
    23
    24
    25  Private Sub btnDelete_Click()
    26  On Error GoTo Err_btnDelete_Click
    27
    28
    29      DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    30      DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70
    31
    32  Exit_btnDelete_Click:
    33      Exit Sub
    34
    35  Err_btnDelete_Click:
    36      MsgBox Err.Description
    37      Resume Exit_btnDelete_Click
    38
    39  End Sub
    40
    41  Private Sub btnLevel0_Click()
    42  On Error GoTo Err_btnLevel0_Click
    43
    44      Dim lngRecordNum As Long
    45      Dim strSQL As String
    46
    47      lngRecordNum = Me.CurrentRecord
    48
    49      DoCmd.Close
    50
    51      DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    52      DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    53      Forms!frmSchoolToPropLevelMaint!ChosenLevel = 0
    54
    55      ' Update the selected property list
    56      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = " &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "" & " And
AMOS_FK = " & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "" & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "" & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= " & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "" & "
And AMOS_FK = " & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "" &
 And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & ""
    57      Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL
    58
    59
    60  Exit_btnLevel0_Click:
    61      Exit Sub
    62
    63  Err_btnLevel0_Click:
    64      MsgBox Err.Description
    65      Resume Exit_btnLevel0_Click
    66
    67  End Sub
    68
```

```
Private Sub btnLevel1_Click()
On Error GoTo Err_btnLevel1_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 1

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "' & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' &
 And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel1_Click:
    Exit Sub

Err_btnLevel1_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel1_Click

End Sub

Private Sub btnLevel2_Click()
On Error GoTo Err_btnLevel2_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 2

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "' & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' &
 And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel2_Click:
    Exit Sub

Err_btnLevel2_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel2_Click

End Sub

Private Sub btnLevel3_Click()
On Error GoTo Err_btnLevel3_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 3

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "' & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' &
 And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel3_Click:
    Exit Sub

Err_btnLevel3_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel3_Click

End Sub

Private Sub btnLevel4_Click()
On Error GoTo Err_btnLevel4_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 4

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "' & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' &
 And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel4_Click:
    Exit Sub

Err_btnLevel4_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel4_Click

End Sub

Private Sub btnLevel5_Click()
On Error GoTo Err_btnLevel5_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 5

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' &
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "' &
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' &
 And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel5_Click:
    Exit Sub

Err_btnLevel5_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel5_Click

End Sub

Private Sub btnLevel6_Click()
On Error GoTo Err_btnLevel6_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 6

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' &
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "' &
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "' & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "' &
 And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel6_Click:
    Exit Sub

Err_btnLevel6_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel6_Click

End Sub

Private Sub cmbCourseNumberFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SCourseNumber_PK] = " & Chr(34) &
Me![cmbCourseNumberFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbCourseNumberFind] = Null
    Me.SCourseNumber_PK.SetFocus

End Sub

Private Sub Form_Current()
    Dim strSQL, strInput1, strInput2 As String
    Dim intLevel As Integer

    ' Update the Level 0 list
    intLevel = 0
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "' & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "' & " And Level = " & intLevel & "' & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
 Me.SCourseNumber_PK.Value & "' & " And AMOS_FK = '" & Me.AMOS_PK.Value
& "' & " And Level = " & intLevel & "'"
    Me.lstLevel0.RowSource = strSQL

    ' Update the Level 1 list
    intLevel = 1
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "' & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "' & " And Level = " & intLevel & "' & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
 Me.SCourseNumber_PK.Value & "' & " And AMOS_FK = '" & Me.AMOS_PK.Value
& "' & " And Level = " & intLevel & "'"
    Me.lstLevel1.RowSource = strSQL

    ' Update the Level 2 list
    intLevel = 2
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "' & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "' & " And Level = " & intLevel & "' & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
 Me.SCourseNumber_PK.Value & "' & " And AMOS_FK = '" & Me.AMOS_PK.Value
```

```
3120      & "" & " And Level = " & intLevel & ""
3121      Me.lstLevel2.RowSource = strSQL
3122
3123      intLevel = 3
3124      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
3125     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3126      '" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "" & " UNION
3127     SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
3128      Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
3129     & "'" & " And Level = " & intLevel & ""
3130      Me.lstLevel3.RowSource = strSQL
3131
3132      intLevel = 4
3133      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
3134     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3135      '" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "" & " UNION
3136     SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
3137      Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
3138     & "'" & " And Level = " & intLevel & ""
3139      Me.lstLevel4.RowSource = strSQL
3140
3141      intLevel = 5
3142      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
3143     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3144      '" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "" & " UNION
3145     SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
3146      Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
3147     & "'" & " And Level = " & intLevel & ""
3148      Me.lstLevel5.RowSource = strSQL
3149
3150      intLevel = 6
3151      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
3152     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3153      '" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "" & " UNION
3154     SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
3155      Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
3156     & "'" & " And Level = " & intLevel & ""
3157      Me.lstLevel6.RowSource = strSQL
3158
3159
3160  End Sub
3161
3162  Form: frmSchoolToPropLevelMaint
3163    Code
3164      Attribute VB_Name = "Form_frmSchoolToPropLevelMaint"
3165      Attribute VB_Creatable = True
3166      Attribute VB_PredeclaredId = True
3167      Attribute VB_Exposed = False
3168      Option Compare Database
3169      Option Explicit
3170
3171      Private Sub btnClose_Click()
3172      On Error GoTo Err_btnClose_Click
3173
3174        Dim lngRecordNum As Long
3175
3176        lngRecordNum = Me.CurrentRecord
3177
3178        DoCmd.Close
3179
3180        DoCmd.OpenForm "frmSchoolToPropertyMaint"
3181        DoCmd.GoToRecord acDataForm, "frmSchoolToPropertyMaint", acGoTo,
3182    lngRecordNum
3183
3184    Exit_btnClose_Click:
3185        Exit Sub
3186
3187    Err_btnClose_Click:
3188        MsgBox Err.Description
3189        Resume Exit_btnClose_Click
3190
3191    End Sub
3192
3193    Private Sub btnDelete_Click()
3194    On Error GoTo Err_btnDelete_Click
3195
3196
3197        DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
3198        DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70
3199
3200    Exit_btnDelete_Click:
3201        Exit Sub
3202
3203    Err_btnDelete_Click:
3204        MsgBox Err.Description
3205        Resume Exit_btnDelete_Click
3206
3207    End Sub
3208
3209    Private Sub ChosenLevel_AfterUpdate()
3210
3211        Dim strSQL As String
3212
3213      ' Update the potential property list
3214      strSQL = "SELECT DISTINCTROW [qryUnionAllProperty].[FPropertyName_PK]
3215     FROM qryUnionAllProperty LEFT JOIN qryUnionPropertyByLevel ON
3216    [qryUnionAllProperty].[FPropertyName_PK] =
3217    [qryUnionPropertyByLevel].[FPropertyName_FK] WHERE
3218    ([qryUnionPropertyByLevel].[FPropertyName_FK] Is Null);"
3219      Me.lstLevelUnused.RowSource = strSQL
3220      ' Update the selected property list
3221      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
3222     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3223      '" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
3224      & " UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE
3225     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3226      '" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
3227
3228      Me.lstLevel0.RowSource = strSQL
3229
3230  End Sub
3231
3232  Private Sub cmbCourseNumberFind_AfterUpdate()
3233
3234      Dim R As Recordset
3235      Set R = Me.RecordsetClone
3236      R.FindFirst "[SCourseNumber_PK] = " & Chr(34) &
3237    Me![cmbCourseNumberFind] & Chr(34)
3238      Me.Bookmark = R.Bookmark
3239      Me![cmbCourseNumberFind] = Null
3240      Me.SCourseNumber_PK.SetFocus
3241
3242  End Sub
3243
3244  Private Sub Form_Current()
3245
3246      Dim strSQL As String
3247
3248      ' Update the potential property list
3249      strSQL = "SELECT DISTINCTROW [qryUnionAllProperty].[FPropertyName_PK]
3250     FROM qryUnionAllProperty LEFT JOIN qryUnionPropertyByLevel ON
3251    [qryUnionAllProperty].[FPropertyName_PK] =
3252    [qryUnionPropertyByLevel].[FPropertyName_FK] WHERE
3253    ([qryUnionPropertyByLevel].[FPropertyName_FK] Is Null);"
3254      Me.lstLevelUnused.RowSource = strSQL
3255      ' Update the selected property list
3256      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
3257     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3258      '" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
3259      & " UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE
3260     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3261      '" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
3262
3263      Me.lstLevel0.RowSource = strSQL
3264
3265  End Sub
3266
3267
3268  Private Sub LeftArrow_Click()
3269      Dim frm As Form, ctl As Control
3270      Dim varItm As Variant, intI As Integer
3271      Dim strSQL As String
3272
3273      ' Ensures the current record is save to the SCHOOL table
3274      DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, ,
3275
3276      ' This code enters the selected Target MOS's into the SCH_TGT_MOS
3277      Set frm = Forms!frmSchoolToPropLevelMaint
3278      Set ctl = frm!lstLevelUnused
3279      For Each varItm In ctl.ItemsSelected
3280          For intI = 0 To ctl.ColumnCount - 1
3281
3282              ' This puts the value found in the list box into a text box,
3283    making it readable by the query
3284              Forms!frmSchoolToPropLevelMaint!TargetValue.Value =
3285    ctl.Column(intI, varItm)
3286
3287              ' Disables the action query confirmation message
3288              Application.SetOption "Confirm Action Queries", False
3289              DoCmd.SetWarnings (False)
3290
3291              ' Enters the value into the SCH_TGT_MOS table
3292              DoCmd.OpenQuery "qryUpdateFUND_SCH_PROP"
3293              DoCmd.OpenQuery "qryUpdateLOG_SCH_PROP"
3294
3295              ' Enables the action query confirmation message
3296              Application.SetOption "Confirm Action Queries", True
3297              DoCmd.SetWarnings (True)
3298
3299          Next intI
3300      Next varItm
3301
3302      ' Update the potential property list
3303      strSQL = "SELECT DISTINCTROW [qryUnionAllProperty].[FPropertyName_PK]
3304     FROM qryUnionAllProperty LEFT JOIN qryUnionPropertyByLevel ON
3305    [qryUnionAllProperty].[FPropertyName_PK] =
3306    [qryUnionPropertyByLevel].[FPropertyName_FK] WHERE
3307    ([qryUnionPropertyByLevel].[FPropertyName_FK] Is Null);"
3308      Me.lstLevelUnused.RowSource = strSQL
3309      ' Update the selected property list
3310      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
3311     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3312      '" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
3313      & " UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE
3314     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3315      '" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
3316
3317      Me.lstLevel0.RowSource = strSQL
3318
3319  End Sub
3320
3321  Private Sub RightArrow_Click()
3322      Dim frm As Form, ctl As Control
3323      Dim varItm As Variant, intI As Integer
3324      Dim strSQL As String
3325
3326      ' This code Deletes the selected Target MOS's and its associated
3327    entries from the SCH_TGT_MOS table
3328      Set frm = Forms!frmSchoolToPropLevelMaint
3329      Set ctl = frm!lstLevel0
3330      For Each varItm In ctl.ItemsSelected
3331          For intI = 0 To ctl.ColumnCount - 1
3332
3333              ' This puts the value found in the list box into a text box,
3334    making it readable by the query
3335              Forms!frmSchoolToPropLevelMaint!TargetValue.Value =
3336    ctl.Column(intI, varItm)
3337
3338              ' Disables the action query confirmation message
3339              Application.SetOption "Confirm Action Queries", False
3340              DoCmd.SetWarnings (False)
3341
3342              ' Deletes the record associated with the specified value from
3343    the FUND or LOG SCH_PROP table
3344              DoCmd.OpenQuery "qryDeleteFUND_SCH_PROP"
3345              DoCmd.OpenQuery "qryDeleteLOG_SCH_PROP"
3346
3347              ' Enables the action query confirmation message
3348              Application.SetOption "Confirm Action Queries", True
3349              DoCmd.SetWarnings (True)
3350
3351          Next intI
3352      Next varItm
3353
3354      ' Update the potential property list
3355      strSQL = "SELECT DISTINCTROW [qryUnionAllProperty].[FPropertyName_PK]
3356     FROM qryUnionAllProperty LEFT JOIN qryUnionPropertyByLevel ON
3357    [qryUnionAllProperty].[FPropertyName_PK] =
3358    [qryUnionPropertyByLevel].[FPropertyName_FK] WHERE
3359    ([qryUnionPropertyByLevel].[FPropertyName_FK] Is Null);"
3360      Me.lstLevelUnused.RowSource = strSQL
3361      ' Update the selected property list
3362      strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
3363     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3364      '" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
3365      & " UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE
3366     SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
3367      '" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
3368
3369      Me.lstLevel0.RowSource = strSQL
3370
3371  End Sub
3372
3373  Form: frmScrubMarine
3374    Code
3375      Attribute VB_Name = "Form_frmScrubMarine"
3376      Attribute VB_Creatable = True
3377      Attribute VB_PredeclaredId = True
3378      Attribute VB_Exposed = False
3379      Option Compare Database
```

```
Option Explicit

Private Sub Close_Click()
On Error GoTo Err_Close_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close current form
    DoCmd.Close

    ' Open specified form
    stDocName = "frmPreprocessing&ExecutionSwitchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Close_Click:
    Exit Sub

Err_Close_Click:
    MsgBox Err.Description
    Resume Exit_Close_Click

End Sub

Private Sub cmbPefFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[PEF] = " & Chr(34) & Me![cmbPefFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbPefFind] = Null
    Me.SSN_PK.SetFocus

End Sub

Private Sub cmbSSNFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SSN_PK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbSSNFind] = Null
    Me.Close.SetFocus

End Sub

Private Sub Delete_Click()
On Error GoTo Err_Delete_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_Delete_Click:
    Exit Sub

Err_Delete_Click:
    MsgBox Err.Description
    Resume Exit_Delete_Click

End Sub

Private Sub Form_Current()
    Dim strSQL As String

    ' Update the PEF List
    strSQL = "SELECT DISTINCTROW [MARINE].[PEF]FROM MARINE LEFT JOIN PEF
ON [MARINE].[PEF] = [PEF].[PEF_PK]WHERE ([PEF].[PEF_PK] Is Null)ORDER BY
    Me.cmbPefFind.RowSource = strSQL

End Sub

Form: frmSpecialAssignment
    Code
    Attribute VB_Name = "Form_frmSpecialAssignment"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub cmbClassNumber_AfterUpdate()

        Dim strConvert As String

        Me.txtReportDate = Me.cmbClassNumber.Column(2)
        Me.txtMCC = Me.cmbClassNumber.Column(4)

        ' Convert the fiscal year to a two digit number for RD3 File
        strConvert = CStr(Me.cmbClassNumber.Column(3))
        strConvert = Right(strConvert, 2)
        Me.txtFY = CInt(strConvert)

        Me.txtAssignmentType = "S"

    End Sub

    Private Sub cmbClassNumber_Enter()

        Dim strSQL As String

        ' This query finds the class numbers, report dates and class
convening dates associated with the chosen school
        strSQL = "SELECT BNA_EXTRACT.ClassNumber_PK,
BNA_EXTRACT.ClassConvDate, BNA_EXTRACT.ReportDate,
BNA_EXTRACT.FiscalYear_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT INNER JOIN
SCH_TGT_MOS ON (BNA_EXTRACT.CourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK) AND (BNA_EXTRACT.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_PK) WHERE BNA_EXTRACT.CourseNumber_PK =
Forms!frmSpecialAssignment!cmbCourseNumber AND SCH_TGT_MOS.AMOS_FK =
Forms!frmSpecialAssignment!txtAMOS;"
        Me.cmbClassNumber.RowSource = strSQL

    End Sub

    Private Sub cmbCourseNumber_AfterUpdate()

        Me.txtAMOS = Me!cmbCourseNumber.Column(1)

    End Sub

    Private Sub cmbSSN_AfterUpdate()

        Me.txtSOI = Me!cmbSSN.Column(1)
        Me.txtGradDate = Me!cmbSSN.Column(2)

    End Sub

    Private Sub cmbSSN_Enter()

        Dim strSQL As String

        strSQL = "SELECT DISTINCTROW MARINE.SSN_PK, MARINE.SOI,
MARINE.GradDate FROM MARINE LEFT JOIN ASSIGNMENT ON MARINE.SSN_PK =
ASSIGNMENT.SSN_FK WHERE (((ASSIGNMENT.SSN_FK) Is Null));"
        Me.cmbSSN.RowSource = strSQL

    End Sub

    Private Sub cmbSSNFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[SSN_FK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbSSNFind] = Null
        Me.Close.SetFocus

    End Sub

    Private Sub Delete_Click()
    On Error GoTo Err_Delete_Click


        DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
        DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

    Exit_Delete_Click:
        Exit Sub

    Err_Delete_Click:
        MsgBox Err.Description
        Resume Exit_Delete_Click

    End Sub

    Private Sub Close_Click()
    On Error GoTo Err_Close_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close current form
        DoCmd.Close

        ' Open specified form
        stDocName = "frmPreprocessing&ExecutionSwitchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Close_Click:
        Exit Sub

    Err_Close_Click:
        MsgBox Err.Description
        Resume Exit_Close_Click

    End Sub

    Private Sub Form_Current()

        ' Prevents user from accidentially making special assignments to
marines assigned normally
        If IsNull(Me.cmbSSN.Value) = False Then
            Me!cmbSSN.Enabled = False
            Me!cmbCourseNumber.Enabled = False
            Me!cmbClassNumber.Enabled = False

        ' Allows the user to enter values for the fundamental equation
        Else
            Me!cmbSSN.Enabled = True
            Me!cmbCourseNumber.Enabled = True
            Me!cmbClassNumber.Enabled = True

        End If

    End Sub

    Private Sub Form_Open(Cancel As Integer)

        DoCmd.GoToRecord acDataForm, "frmSpecialAssignment", acNewRec

    End Sub

Form: frmSpecialAssignmentOfUnassignedMarine
    Code
    Attribute VB_Name = "Form_frmSpecialAssignmentOfUnassignedMarine"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub cmbClassNumber_AfterUpdate()

        Dim strConvert As String
        Dim strSSN As String
        Dim R As Recordset

        Me.txtReportDate = Me.cmbClassNumber.Column(2)
        Me.txtMCC = Me.cmbClassNumber.Column(4)

        ' Convert the fiscal year to a two digit number for RD3 File
        strConvert = CStr(Me.cmbClassNumber.Column(3))
        strConvert = Right(strConvert, 2)
        Me.txtFY = CInt(strConvert)

        Me.txtAssignmentType = "S"

        ' Keep track of the current SSN
        strSSN = Me.cmbSSN

        Me.txtAMOS = Me!cmbCourseNumber.Column(1)

        ' Updates the current form and subform
        Me.Requery
        Me.Repaint

        ' Moves the record back to the current SSN
        Set R = Me.RecordsetClone
        R.FindFirst "[SSN_FK] = " & Chr(34) & strSSN & Chr(34)
        Me.Bookmark = R.Bookmark

    End Sub

    Private Sub cmbClassNumber_Enter()

        Dim strSQL As String
```

```
3640      44      ' This query finds the class numbers, report dates and class
3641      convening dates associated with the chosen school
3642      45      strSQL = "SELECT BNA_EXTRACT.ClassNumber_PK,
3643      BNA_EXTRACT.ClassConvDate, BNA_EXTRACT.ReportDate,
3644      BNA_EXTRACT.FiscalYear_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT INNER JOIN
3645      SCH_TGT_MOS ON (BNA_EXTRACT.CourseNumber_PK =
3646      SCH_TGT_MOS.TCourseNumber_FK) AND (BNA_EXTRACT.TargetMOS_PK =
3647      SCH_TGT_MOS.TargetMOS_FK) WHERE BNA_EXTRACT.CourseNumber_PK =
3648      Forms!frmSpecialAssignmentOfUnassignedMarine!cmbCourseNumber AND
3649      SCH_TGT_MOS.AMOS_FK =
3650      46      Me.cmbClassNumber.RowSource = strSQL
3651      47
3652      48  End Sub
3653      49
3654      50  Private Sub cmbCourseNumber_AfterUpdate()
3655      51
3656      52      Dim strSSN As String
3657      53      Dim R As Recordset
3658      54
3659      55      strSSN = Me.cmbSSN
3660      56
3661      57      ' Keep track of the current SSN
3662      58      strSSN = Me.cmbSSN
3663      59
3664      60      Me.txtAMOS = Me!cmbCourseNumber.Column(1)
3665      61
3666      62      ' Updates the current form and subform
3667      63      Me.Requery
3668      64      Me.Repaint
3669      65
3670      66      ' Moves the record back to the current SSN
3671      67      Set R = Me.RecordsetClone
3672      68      R.FindFirst "[SSN_FK] = " & Chr(34) & strSSN & Chr(34)
3673      69      Me.Bookmark = R.Bookmark
3674      70
3675      71  End Sub
3676      72
3677      73  Private Sub cmbSSN_AfterUpdate()
3678      74
3679      75      Me.txtSOI = Me!cmbSSN.Column(1)
3680      76      Me.txtGradDate = Me!cmbSSN.Column(2)
3681      77
3682      78  End Sub
3683      79
3684      80  Private Sub cmbSSN_Enter()
3685      81
3686      82      Dim strSQL As String
3687      83
3688      84      strSQL = "SELECT DISTINCTROW MARINE.SSN_PK, MARINE.SOI,
3689      MARINE.GradDate FROM MARINE LEFT JOIN ASSIGNMENT ON MARINE.SSN_PK =
3690      ASSIGNMENT.SSN_FK WHERE (((ASSIGNMENT.SSN_FK) Is Null));"
3691      85      Me.cmbSSN.RowSource = strSQL
3692      86
3693      87  End Sub
3694      88
3695      89  Private Sub cmbSSNFind_AfterUpdate()
3696      90
3697      91      Dim R As Recordset
3698      92      Set R = Me.RecordsetClone
3699      93      R.FindFirst "[SSN_FK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
3700      94      Me.Bookmark = R.Bookmark
3701      95      Me![cmbSSNFind] = Null
3702      96      Me.Close.SetFocus
3703      97
3704      98  End Sub
3705      99
3706      100 Private Sub Delete_Click()
3707      101 On Error GoTo Err_Delete_Click
3708      102
3709      103
3710      104     DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
3711      105     DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70
3712      106
3713      107 Exit_Delete_Click:
3714      108     Exit Sub
3715      109
3716      110 Err_Delete_Click:
3717      111     MsgBox Err.Description
3718      112     Resume Exit_Delete_Click
3719      113
3720      114 End Sub
3721      115
3722      116 Private Sub Close_Click()
3723      117 On Error GoTo Err_Close_Click
3724      118
3725      119     ' Close current form
3726      120     DoCmd.Close
3727      121
3728      122 Exit_Close_Click:
3729      123     Exit Sub
3730      124
3731      125 Err_Close_Click:
3732      126     MsgBox Err.Description
3733      127     Resume Exit_Close_Click
3734      128
3735      129 End Sub
3736      130
3737      131 Private Sub Form_Current()
3738      132
3739      133         Me!cmbSSN.Enabled = True
3740      134         Me!cmbCourseNumber.Enabled = True
3741      135         Me!cmbClassNumber.Enabled = True
3742      136
3743      137
3744      138
3745      139 End Sub
3746      140
3747      141 Private Sub Form_Open(Cancel As Integer)
3748      142
3749      143
3750      144     DoCmd.GoToRecord acDataForm,
3751      frmSpecialAssignmentOfUnassignedMarine, acNewRec
3752      145
3753      146
3754      147 End Sub
3755
3756  Form: frmUnassignedMarines
3757      Code
3758      1   Attribute VB_Name = "Form_frmUnassignedMarines"
3759      2   Attribute VB_Creatable = True
3760      3   Attribute VB_PredeclaredId = True
3761      4   Attribute VB_Exposed = False
3762      5   Option Compare Database
3763      6   Option Explicit
3764      7
3765      8   Private Sub Close_Click()
3766      9   On Error GoTo Err_Close_Click
3767      10
3768      11      Dim stDocName As String
3769      12      Dim stLinkCriteria As String
3770      13
3771      14      ' Close the current form
3772      15      DoCmd.Close
3773      16      ' Open specified form
3774      17      stDocName = "frmAnalyzeResult"
3775      18      DoCmd.OpenForm stDocName, , , stLinkCriteria
3776      19
3777      20  Exit_Close_Click:
3778      21      Exit Sub
3779      22
3780      23  Err_Close_Click:
3781      24      MsgBox Err.Description
3782      25      Resume Exit_Close_Click
3783      26
3784      27  End Sub
3785      28
3786      29  Private Sub cmbSSNFind_AfterUpdate()
3787      30
3788      31      Dim R As Recordset
3789      32      Set R = Me.RecordsetClone
3790      33      R.FindFirst "[SSN_FK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
3791      34      Me.Bookmark = R.Bookmark
3792      35      Me![cmbSSNFind] = Null
3793      36      Me.Close.SetFocus
3794      37
3795      38  End Sub
3796      39
3797      40  Private Sub Form_Current()
3798      41
3799      42      Dim strSQL As String
3800      43      Dim rec As Recordset
3801      44      Dim db1 As Database
3802      45
3803      46      Set db1 = CurrentDb()
3804      47
3805      48      ' Calculates the number of unassigned marines.
3806      49      strSQL = "SELECT Count(PEF) AS TotalUnassigned FROM
3807      50      Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
3808      51      Me.txtTotalUnassigned = rec!TotalUnassigned
3809      52
3810      53  End Sub
3811      54  Private Sub btnAssign_Click()
3812      55  On Error GoTo Err_btnAssign_Click
3813      56
3814      57      Dim stDocName As String
3815      58
3816      59      stDocName = "frmSpecialAssignmentOfUnassignedMarine"
3817      60
3818      61      DoCmd.OpenForm stDocName
3819      62
3820      63  Exit_btnAssign_Click:
3821      64      Exit Sub
3822      65
3823      66  Err_btnAssign_Click:
3824      67      MsgBox Err.Description
3825      68      Resume Exit_btnAssign_Click
3826      69
3827      70  End Sub
3828
3829  Form: subfrmFundamentalProperty
3830      Code
3831      1   Attribute VB_Name = "Form_subfrmFundamentalProperty"
3832      2   Attribute VB_Creatable = True
3833      3   Attribute VB_PredeclaredId = True
3834      4   Attribute VB_Exposed = False
3835      5   Option Compare Database
3836      6   Option Explicit
3837      7   Private Sub cmbValue_FK_AfterUpdate()
3838      8
3839      9       Forms!frmFundamentalProperty![FPropertyName_PK].SetFocus
3840      10
3841      11  End Sub
3842      12
3843      13  Private Sub cmbValue_FK_BeforeUpdate(Cancel As Integer)
3844      14
3845      15      On Error GoTo Err_cmbValue_FK_BeforeUpdate
3846      16
3847      17          Dim db As Database
3848      18          Dim rec As Recordset
3849      19          Dim strSQL As String
3850      20          Dim Convert As Variant
3851      21
3852      22          ' Places value in a form agreeable to Access (e.g. "H" or "10")
3853      23          Convert = "" &
3854      Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK & ""
3855      24
3856      25          ' Opens the table PROPERTY_VALUE
3857      26          strSQL = "PROPERTY_VALUE"
3858      27          Set db = CurrentDb()
3859      28          Set rec = db.OpenRecordset(strSQL, dbOpenSnapshot)
3860      29
3861      30          ' Searches for the value entered by the user in the
3862      31          rec.FindFirst "Value_PK = " & Convert
3863      32
3864      33          ' If the value is not in the PROPERTY_VALUE table, it is added.
3865      34          If rec.NoMatch = True Then
3866      35
3867      36              ' Disables the action query confirmation message
3868      37              Application.SetOption "Confirm Action Queries", False
3869      38
3870      39              ' Enters the value into the PROPERTY_VALUE table
3871      40              DoCmd.OpenQuery "qryUpdateProperty_Value"
3872      41
3873      42              ' Updates the drop down list for the combobox
3874      43              strSQL = "SELECT DISTINCT " &
3875      Forms!frmFundamentalProperty!MarineField & " FROM MARINE;"
3876      44
3877      Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSou
3878      rce = strSQL
3879      45
3880      46              ' Enables the action query confirmation message
3881      47              Application.SetOption "Confirm Action Queries", True
3882      48          End If
3883      49
3884      50          rec.Close
3885      51
3886      52  Exit_cmbValue_FK_BeforeUpdate:
3887      53      Exit Sub
3888      54
3889      55  Err_cmbValue_FK_BeforeUpdate:
3890      56      MsgBox Err.Description
3891      57      Application.SetOption "Confirm Action Queries", True
3892      58      Resume Exit_cmbValue_FK_BeforeUpdate
3893      59
3894      60  End Sub
3895
3896
3897  Form: subfrmFundamentalPropertyList
3898      Code
3899      1   Attribute VB_Name = "Form_subfrmFundamentalPropertyList"
```

```
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub cmbValue_FK_BeforeUpdate(Cancel As Integer)
    On Error GoTo Err_cmbValue_FK_BeforeUpdate

    Dim db As Database
    Dim rec As Recordset
    Dim strSQL As String
    Dim Convert As Variant

    'Opens the subfrmFundamentalProperty so the following code will
    DoCmd.OpenForm "subfrmFundamentalProperty", , , , acHidden

    ' Places value in a form agreeable to Access (e.g. 'HT' or '10')
    Convert = "" & Forms![subfrmFundamentalProperty]!cmbValue_FK &
    ' Opens the table PROPERTY_VALUE
    strSQL = "PROPERTY_VALUE"

    Set db = CurrentDb()
    Set rec = db.OpenRecordset(strSQL, dbOpenSnapshot)

    rec.FindFirst "Value_PK = " & Convert
    If rec.NoMatch = True Then
        ' Disables the action query confirmation message
        Application.SetOption "Confirm Action Queries", False

        DoCmd.OpenQuery "qryUpdateProperty_Value"
        strSQL = "SELECT DISTINCT " &
Forms!frmFundamentalProperty!MarineField & " FROM MARINE;"
        Me.cmbValue_FK.RowSource = strSQL

        ' Enables the action query confirmation message
        Application.SetOption "Confirm Action Queries", True
    End If

    rec.Close

    'Closes the subfrmFundamentalProperty without saving changes to
    DoCmd.Close acForm, "subfrmFundamentalProperty"

Exit_cmbValue_FK_BeforeUpdate:
    Exit Sub

Err_cmbValue_FK_BeforeUpdate:
    MsgBox Err.Description
    Application.SetOption "Confirm Action Queries", True
    Resume Exit_cmbValue_FK_BeforeUpdate

End Sub

Private Sub Value_FK_BeforeUpdate(Cancel As Integer)
    On Error GoTo Err_Value_FK_BeforeUpdate

    Dim db As Database
    Dim rec As Recordset
    Dim strSQL As String
    Dim Convert As Variant

    ' Places value in a form agreeable to Access (e.g. 'HT' or '10')
    Convert = "" &
Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK & ""
    ' Opens the table PROPERTY_VALUE
    strSQL = "PROPERTY_VALUE"

    Set db = CurrentDb()
    Set rec = db.OpenRecordset(strSQL, dbOpenSnapshot)

    rec.FindFirst "Value_PK = " & Convert
    ' This If Then statement checks to see if the entered value is in
    ' PROPERTY_VALUE table. If not, it is added to the table.
    If rec.NoMatch = True Then
        ' Disables the action query confirmation message
        Application.SetOption "Confirm Action Queries", False

        DoCmd.OpenQuery "qryUpdateProperty_ValueList"

        ' Enables the action query confirmation message
        Application.SetOption "Confirm Action Queries", True
    End If

    rec.Close

Exit_Value_FK_BeforeUpdate:
    Exit Sub

Err_Value_FK_BeforeUpdate:
    MsgBox Err.Description
    Application.SetOption "Confirm Action Queries", True
    Resume Exit_Value_FK_BeforeUpdate

End Sub


Module: modFitnessDetermination
Code
Attribute VB_Name = "modFitnessDetermination"
Option Compare Database
Option Explicit

Dim lngStart, lngEnd As Long

Public Function FundPropTest(FundPropName As String, SSN As String) As

    Dim rec As Recordset, rec2 As Recordset, rec3 As Recordset
    Dim strMarField As String, strFundOperator As String, strSQL As
    Dim strMarValueSQL As String, strFundValueSQL As String, strInput As
    Dim varFundValue, varMarValue
    Dim db1 As Database

    FundPropName = "" & FundPropName & ""    'This expression
properly formates the argument FundPropName, by placing a ' in front of
and behind the value found in FundPropName.

    Set db1 = CurrentDb()

    strSQL = "SELECT MarineField, Operator FROM FUNDAMENTAL_PROPERTY
WHERE FPropertyName_PK = " & FundPropName
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    strMarField = rec!MarineField

    strFundOperator = rec!Operator

    strInput = "" & SSN & ""
    strMarValueSQL = "SELECT " & strMarField & " AS [Value] FROM MARINE
WHERE SSN_PK = " & strInput
    Set rec2 = db1.OpenRecordset(strMarValueSQL, dbOpenSnapshot)
    varMarValue = rec2!Value

    strFundValueSQL = "SELECT Value_FK FROM FUND_PROP_VAL WHERE
FPropertyName_FK = " & FundPropName
    Set rec3 = db1.OpenRecordset(strFundValueSQL, dbOpenSnapshot)
    varFundValue = rec3!Value_FK

    ' Convert varFundValue and varMarValue to a numeric, if possible
    If IsNumeric(varFundValue) And IsNumeric(varMarValue) Then
        varFundValue = CInt(varFundValue)
        varMarValue = CInt(varMarValue)
    Else
        varFundValue = CStr(varFundValue)
        varMarValue = CStr(varMarValue)
    End If


    Select Case strFundOperator
        Case "="
            If varMarValue = varFundValue Then
                FundPropTest = True
            End If
        Case "not ="
            If varMarValue <> varFundValue Then
                FundPropTest = True
            End If
        Case "<"
            If varMarValue < varFundValue Then
                FundPropTest = True
            End If
        Case ">"
            If varMarValue > varFundValue Then
                FundPropTest = True
            End If
        Case "<="
            If varMarValue <= varFundValue Then
                FundPropTest = True
            End If
        Case ">="
            If varMarValue >= varFundValue Then
                FundPropTest = True
            End If
        Case "in"
            Do Until rec3.EOF
                varFundValue = rec3!Value_FK
                If IsNumeric(varFundValue) And IsNumeric(varMarValue)
                    varFundValue = CInt(varFundValue)
                    varMarValue = CInt(varMarValue)
                Else
                    varFundValue = CStr(varFundValue)
                    varMarValue = CStr(varMarValue)
                End If
                If varMarValue = varFundValue Then
                    FundPropTest = True
                End If
                rec3.MoveNext
            Loop
    End Select
    rec2.Close
    rec3.Close
    db1.Close

End Function
Public Function LogOrTest(Input1 As Boolean, Input2 As Boolean) As

If Input1 Or Input2 Then
    LogOrTest = True

Else: LogOrTest = False
End If

End Function
Public Function LogAndTest(Input1 As Boolean, Input2 As Boolean) As

If Input1 And Input2 Then
    LogAndTest = True
Else: LogAndTest = False
End If


End Function
Public Function LogPropTest(LogPropName As String, SSN As String) As
    Dim intNewUBound As Integer, j As Integer, k As Integer, l As
Integer, m As Integer
    Dim booOrOperator As Boolean, booAndOperator As Boolean,
booNotOperator As Boolean
    Dim strLogicalEquation As String, strSQL As String, strInput As
    Dim strWorkingEqn() As String
    Dim intSpacePos As Integer      ' position of the space
    Dim db1 As Database
    Dim rec As Recordset


    Set db1 = CurrentDb()

    ' Get the logical equation from the LOGICAL_PROPERTY table
    strInput = "" & LogPropName & ""
    strSQL = "SELECT LogicalEquation FROM LOGICAL_PROPERTY WHERE
LPropertyName_PK = " & strInput & ""
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    strLogicalEquation = rec!LogicalEquation

    ' Parse the string into an array
    ' declare the temporary string as an array
    ReDim strWorkingEqn(0) As String

    ' Find the position of the first space
    intSpacePos = InStr(strLogicalEquation, " ")
    ' loop until no more spaces are found
    ' InStr returns 0 if the space is not found
    Do Until intSpacePos = 0
        ' redimension the array, adding another element
        ' UBound tells us how many elements there already are in the
        ReDim Preserve strWorkingEqn(UBound(strWorkingEqn) + 1)

        ' set the new element to the characters from the beginning of the
        ' string up to the first space.  Left$ returns a number of
        ' from the beginning of a string
        strWorkingEqn(UBound(strWorkingEqn)) =
Trim$(Left$(strLogicalEquation, intSpacePos - 1))

        ' now we've copied these characters into the name, they can be

        ' removed from the original string.  Right$ returns a number of
        ' characters from the end of the string.  It is the rightmost N
```

```
4160    145     ' characters, therefore we subtract the position of the space
4161    146     ' the length of the string.  This gives us the number of
4162    characters remaining.
4163    147         strLogicalEquation = LTrim$(Right$(strLogicalEquation,
4164    Len(strLogicalEquation) - intSpacePos))
4165    148
4166    149     ' find the next space
4167    150     intSpacePos = InStr(strLogicalEquation, " ")
4168    151 Loop
4169    152
4170    153 For j = 1 To (UBound(strWorkingEqn))
4171    154     If strWorkingEqn(j) = "(" Then
4172    155
4173    156 Line1:
4174    157         For k = (j + 1) To (UBound(strWorkingEqn))
4175    158             Select Case strWorkingEqn(k)
4176    159                 Case ")"
4177    160                     For i = (j + 1) To (k - 1)
4178    161                         Select Case strWorkingEqn(i)
4179    162                             Case "Or"
4180    163                                 booOrOperator = True
4181    164                             Case "And"
4182    165                                 booAndOperator = True
4183    166                             Case "Not"
4184    167                                 booNotOperator = True
4185    168                             Case "True"
4186    169                                 If strWorkingEqn(i) = "(" Then
4187    170                                     If booNotOperator Then
4188    171                                         strWorkingEqn(i) = "False"
4189    172                                         booNotOperator = False
4190    173                                     Else
4191    174                                         strWorkingEqn(i) = "True"
4192    175                                     End If
4193    176                                 Else    ' Executed after the first
4194    bracketed property is tested and stored in the location of "("
4195    177                                     If booNotOperator Then
4196    178                                         strWorkingEqn(i) = "False"
4197    179                                         booNotOperator = False
4198    180                                     End If
4199    181                                     If booOrOperator Then
4200    182                                         strWorkingEqn(j) =
4201    LogOrTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(i)))
4202    183                                         booOrOperator = False
4203    184                                     ElseIf booAndOperator Then
4204    185                                         strWorkingEqn(j) =
4205    LogAndTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(i)))
4206    186                                         booAndOperator = False
4207    187                                     End If
4208    188                                 End If
4209    189                             Case "False"
4210    190                                 If strWorkingEqn(i) = "(" Then
4211    191                                     If booNotOperator Then
4212    192                                         strWorkingEqn(j) = "True"
4213    193                                         booNotOperator = False
4214    194                                     Else
4215    195                                         strWorkingEqn(j) = "False"
4216    196                                     End If
4217    197                                 Else    ' Executed after the first
4218    bracketed property is tested and stored in the location of "("
4219    198                                     If booNotOperator Then
4220    199                                         strWorkingEqn(i) = "True"
4221    200                                         booNotOperator = False
4222    201                                     End If
4223    202                                     If booOrOperator Then
4224    203                                         strWorkingEqn(j) =
4225    LogOrTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(i)))
4226    204                                         booOrOperator = False
4227    205                                     ElseIf booAndOperator Then
4228    206                                         strWorkingEqn(j) =
4229    LogAndTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(i)))
4230    207                                         booAndOperator = False
4231    208                                     End If
4232    209                                 End If
4233    210                             Case ")"    'Do nothing
4234    211                             Case Else
4235    212                                 If strWorkingEqn(i) = "(" Then
4236    213                                     If booNotOperator Then
4237    214                                         strWorkingEqn(j) = Not
4238    (FundPropTest(strWorkingEqn(i), SSN))
4239    215                                         booNotOperator = False
4240    216                                     Else
4241    217                                         strWorkingEqn(j) =
4242    FundPropTest(strWorkingEqn(i), SSN)
4243    218                                     End If
4244    219                                 Else    ' Executed after the first
4245    bracketed property is tested and stored in the location of "("
4246    220                                     If booNotOperator Then
4247    221                                         strWorkingEqn(i) = Not
4248    (FundPropTest(strWorkingEqn(i), SSN))
4249    222                                         booNotOperator = False
4250    223                                     Else
4251    224                                         strWorkingEqn(i) =
4252    FundPropTest(strWorkingEqn(i), SSN)
4253    225                                     End If
4254    226                                     If booOrOperator Then
4255    227                                         strWorkingEqn(j) =
4256    LogOrTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(i)))
4257    228                                         booOrOperator = False
4258    229                                     ElseIf booAndOperator Then
4259    230                                         strWorkingEqn(j) =
4260    LogAndTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(i)))
4261    231                                         booAndOperator = False
4262    232                                     End If
4263    233                                 End If
4264    234                         End Select
4265    235                     Next i
4266    236
4267    237                     ' Must transform the working equation to account
4268    for the bracketed expression evaluation.
4269    238
4270    239                     intNewUBound = UBound(strWorkingEqn) - (k - j)
4271    240                     If intNewUBound <> 1 Then
4272    241                         ReDim strTransformedEqn(1 To intNewUBound)
4273    242                         For m = 1 To j
4274    243                             strTransformedEqn(m) = strWorkingEqn(m)
4275    244                         Next m
4276    245                         For m = (k + 1) To UBound(strWorkingEqn)
4277    246                             strTransformedEqn(m - k + j) =
4278    247                         Next m
4279    248                         ReDim strWorkingEqn(1 To intNewUBound)
4280    249                         For m = 1 To intNewUBound
4281    250                             strWorkingEqn(m) = strTransformedEqn(m)
4282    251                         Next m
4283    252                         j = 1   'Set j back to a value of 1.
4284    253                         GoTo Line1    'Jumps back to For j = 1 to
4285    (UBound(strWorkingEqn))loop
4286    254                     Else
4287    255                         LogPropTest = strWorkingEqn(j)
4288    256                         Erase strWorkingEqn
4289    257                         Erase strTransformedEqn
4290    258                         Exit Function
4291    259                     End If
4292    260                 Case "("
4293    261                     j = k
4294    262                     GoTo Line1    'Jumps back to For j = 1 to
4295    (UBound(strWorkingEqn))loop
4296    263                 'Exit For    'Exits the For k = (j + 1) to
4297    (UBound(strWorkingEqn)) loop
4298    264             End Select
4299    265         Next k
4300    266     End If
4301    267 Next j
4302    268
4303    269 End Function
4304    270
4305    271 Public Sub Fitness()
4306    272     Dim strSQL As String, strClassNumber As String, strAMOS As String
4307    273     Dim strInput1 As String, strInput2 As String
4308    274     Dim i As Integer, j As Integer, k As Integer, l As Integer
4309    275     Dim intNormalFactor As Integer, intFit As Integer, intInput As
4310    276     Dim intLevel1 As Integer, intLevel2 As Integer, intLevel3 As Integer
4311    277     Dim intLevel4 As Integer, intLevel5 As Integer, intLevel6 As Integer
4312    278     Dim sngDesiredWeight As Single, sngEpsilon As Single, sngSumOfFit As
4313    279     Dim sngCount As Single, sngAvgFit As Single, sngFitness As Single
4314    280     Dim sngMandFitValue As Single
4315    281     Dim lngGradDate As Long
4316    282     Dim db1 As Database
4317    283     Dim recClass As Recordset, recMarine As Recordset, recMarClsFit As
4318    284     Dim recPEF As Recordset, recFDesired As Recordset, recLDesired As
4319    285     Dim recFMand As Recordset, recLMand As Recordset
4320    286     Dim booMandProp As Boolean, booPEF As Boolean, booMandatory As
4321    287     Dim varRetVal As Variant
4322    288     Dim frmCurrent As Form_frmClassQuotaPenaltyAndFit
4323    289
4324    290     Set frmCurrent = Forms!frmClassQuotaPenaltyAndFit
4325    291     Set db1 = CurrentDb()
4326    292
4327    293     ' Create TEMP_FIT table
4328    294     Application.SetOption "Confirm Action Queries", False
4329    295
4330    296     lngGradDate = frmCurrent!MCTGradDate
4331    297
4332    298
4333    299     ' Create 2 TEMP_FIT tables from MAR_CLS_FIT table
4334    300     db1.Execute "SELECT SSN_FK, MarineIndex, ClassIndex_FK, Fitness INTO
4335    TEMP_FIT FROM MAR_CLS_FIT;"
4336    301     db1.Execute "SELECT SSN_FK, MarineIndex, ClassIndex_FK, Fitness INTO
4337    TEMP_FIT1 FROM MAR_CLS_FIT;"
4338    302
4339    303     ' Insert into the TEMP_FIT tables the applicable SSN's based on MCT
4340    graduation date
4341    304     db1.Execute "INSERT INTO TEMP_FIT ( SSN_FK )SELECT MARINE.SSN_PK FROM
4342    qrySSN WHERE GradDate = " & lngGradDate & ""
4343    305     db1.Execute "INSERT INTO TEMP_FIT1 ( SSN_FK )SELECT MARINE.SSN_PK
4344    FROM qrySSN WHERE GradDate = " & lngGradDate & ""
4345    306
4346    307     Application.SetOption "Confirm Action Queries", True
4347    308
4348    309     ' Create a dynaset of the temp_fit table
4349    310     strSQL = "SELECT * FROM TEMP_FIT"
4350    311     Set recMarClsFit = db1.OpenRecordset(strSQL, dbOpenDynaset)
4351    312     recMarClsFit.MoveLast
4352    313
4353    314     ' Create the MarineIndex for the Temp_fit table
4354    315     If recMarClsFit.RecordCount > 0 Then
4355    316         recMarClsFit.MoveFirst
4356    317         For i = 1 To recMarClsFit.RecordCount
4357    318             recMarClsFit.Edit
4358    319             recMarClsFit!MarineIndex = i
4359    320             recMarClsFit.Update
4360    321             recMarClsFit.MoveNext
4361    322         Next i
4362    323     End If
4363    324
4364    325     ' Create a dynaset of the Class table
4365    326     strSQL = "SELECT SCourseNumber_FK, AMOS_FK FROM CLASS"
4366    327     Set recClass = db1.OpenRecordset(strSQL, dbOpenDynaset)
4367    328         If recClass.EOF = False Then
4368    329             recClass.MoveLast
4369    330             recClass.MoveFirst
4370    331         End If
4371    332
4372    333     ' Create a dynaset of the Marine table based on TEMP_FIT1 table
4373    334     strSQL = "SELECT * FROM MARINE , TEMP_FIT1 WHERE
4374    (((MARINE.SSN_PK)=[TEMP_FIT1].[SSN_FK]));"
4375    335     Set recMarine = db1.OpenRecordset(strSQL, dbOpenDynaset)
4376    336
4377    337     If recMarine.EOF = False Then
4378    338         recMarine.MoveLast
4379    339         recMarine.MoveFirst
4380    340     End If
4381    341
4382    342     ' Initialize variables
4383    343     booMandProp = True
4384    344     intFit = 0
4385    345     booPEF = False
4386    346     booMandatory = False
4387    347     intNormalFactor = 100
4388    348     intLevel1 = 32
4389    349     intLevel2 = 16
4390    350     intLevel3 = 8
4391    351     intLevel4 = 4
4392    352     intLevel5 = 2
4393    353     intLevel6 = 1
4394    354     sngDesiredWeight =
4395    355     sngMandFitValue = (1 - sngDesiredWeight) * intNormalFactor
4396    356     varRetVal = SysCmd(acSysCmdInitMeter, "Status...",
4397    357     sngEpsilon = 0.000001
4398    358
4399    359     ' Verify recClass has values
4400    360     If recClass.RecordCount > 0 Then
4401    361         strClassNumber = "X"
4402    362         strAMOS = "X"
4403    363
4404    364         ' Cycle through all the classes
4405    365         For i = 1 To recClass.RecordCount
4406    366             ' Yeild control temporarily to Windows
4407    367             DoEvents
4408    368             ' Update status bar
4409    369             varRetVal = SysCmd(acSysCmdUpdateMeter, i)
4410    370
4411    371             ' Update the classindex_FK for the MarClsFit record
4412    372             If recMarClsFit.RecordCount > 0 Then
4413    373                 recMarClsFit.MoveFirst
4414    374                 For j = 1 To recMarClsFit.RecordCount
4415    375                     recMarClsFit.Edit
4416    376                     recMarClsFit!ClassIndex_FK = i
4417    377                     recMarClsFit.Update
4418    378                     recMarClsFit.MoveNext
4419    379                 Next j
```

```
380             recMarClsFit.MoveFirst
381         End If
382         ' Check to see if next record is the same school as the last
383         ' If it is, append it to the MAR_CLS_FIT table
384         If (recClass!SCourseNumber_FK = strClassNumber) And
(recClass!AMOS_FK = strAMOS) Then
385             Application.SetOption "Confirm Action Queries", False
386             db1.Execute "INSERT INTO MAR_CLS_FIT ( SSN_FK,
MarineIndex, ClassIndex_FK, Fitness ) SELECT TEMP_FIT.SSN_FK,
TEMP_FIT.MarineIndex, TEMP_FIT.ClassIndex_FK, TEMP_FIT.Fitness FROM
TEMP_FIT WHERE TEMP_FIT.Fitness > 0;"
387             Application.SetOption "Confirm Action Queries", True
388             recClass.MoveNext
389         Else
390             sngSumOfFit = 0
391             sngCount = 0
392             booPEF = False
393             'Verify recMarine has values
394             If recMarine.RecordCount > 0 Then
395
396                 ' Initialize variables
397                 strInput1 = "" & recClass!SCourseNumber_FK & ""
398                 strInput2 = "" & recClass!AMOS_FK & ""
399                 intInput = 0
400
401                 ' Find all PEF's associated with the current school
402                 strSQL = "SELECT PEF_FK FROM SCH_PEF WHERE
SCourseNumber_FK = " & strInput1 & " and AMOS_FK = '& strInput2 & ""
403                 Set recPEF = db1.OpenRecordset(strSQL,
404                 If recPEF.EOF = False Then
405                     recPEF.MoveLast
406                     recPEF.MoveFirst
407                 End If
408
409                 ' Find mandatory fundamental properties associated
with the current school
410                 strSQL = "SELECT FPropertyName_FK FROM FUND_SCH_PROP
WHERE SCourseNumber_FK = " & strInput1 & " AND AMOS_FK = " & strInput2 &
 AND Level = & intInput & ""
411                 Set recFMand = db1.OpenRecordset(strSQL,
412                 If recFMand.EOF = False Then
413                     recFMand.MoveLast
414                     recFMand.MoveFirst
415                 End If
416
417                 ' Find mandatory logical properties associated with
the current school
418                 strSQL = "SELECT LPropertyName_FK FROM LOG_SCH_PROP
WHERE SCourseNumber_FK = " & strInput1 & " AND AMOS_FK = " & strInput2 &
 AND Level = & intInput & ""
419                 Set recLMand = db1.OpenRecordset(strSQL,
420                 If recLMand.EOF = False Then
421                     recLMand.MoveLast
422                     recLMand.MoveFirst
423                 End If
424
425                 ' Find desired fundamental properties associated with
 the curren school
426                 strSQL = "SELECT FPropertyName_FK, [Level] FROM
FUND_SCH_PROP WHERE SCourseNumber_FK = " & strInput1 & " AND AMOS_FK = "
& strInput2 & " AND Level <> " & intInput & ""
427                 Set recFDesired = db1.OpenRecordset(strSQL,
428                 If recFDesired.EOF = False Then
429                     recFDesired.MoveLast
430                     recFDesired.MoveFirst
431                 End If
432
433                 ' Find desired logical properties associated with the
 curren school
434                 strSQL = "SELECT LPropertyName_FK, [Level] FROM
LOG_SCH_PROP WHERE SCourseNumber_FK = " & strInput1 & " AND AMOS_FK = " &
 strInput2 & " AND Level <> " & intInput & ""
435                 Set recLDesired = db1.OpenRecordset(strSQL,
436                 If recLDesired.EOF = False Then
437                     recLDesired.MoveLast
438                     recLDesired.MoveFirst
439                 End If
440
441                 ' Cycle through all the marines
442                 For j = 1 To recMarine.RecordCount
443                     ' Yield control temporarily to Windows
444                     DoEvents
445                     ' Check to see if marine's PEF is an open
446                     If recMarine!PEF <> "00" Then
447                         booPEF = False
448                         If recPEF.EOF = False Then
449                             ' Check to see if marine's PEF is
associated with the school
450                             For k = 1 To recPEF.RecordCount
451                                 If recPEF!PEF_FK = recMarine!PEF Then
452                                     booPEF = True
453                                 End If
454                                 recPEF.MoveNext
455                             Next k
456                             recPEF.MoveFirst
457                         End If
458                     Else
459                         booPEF = True
460                     End If
461                     ' Compare marine attributes with the mandatory
properties
462                     If booPEF = True Then
463                         ' Check for mandatory fundamental properties
464                         If recFMand.EOF = False Then
465                             For k = 1 To recFMand.RecordCount
466                                 ' All tests must prove true for the
mandatory properties
467                                 If booMandProp = True Then
468                                     If
FundPropTest(recFMand!FPropertyName_FK, recMarine!SSN_PK) = False Then
469                                         booMandProp = False
470                                     End If
471                                     recFMand.MoveNext
472                                 End If
473                             Next k
474                             recFMand.MoveFirst
475                         End If
476                         ' Check for mandatory logical properties
477                         If booMandProp = True Then
478                             If recLMand.EOF = False Then
479                                 For k = 1 To recLMand.RecordCount
480                                     ' All tests must prove true for
the mandatory properties
481                                     If booMandProp = True Then
482                                         If
LogPropTest(recLMand!LPropertyName_FK, recMarine!SSN_PK) = False Then
483                                             booMandProp = False
484                                         End If
485                                         recLMand.MoveNext
486                                     End If
487                                 Next k
488                                 recLMand.MoveFirst
489                             End If
490                             If booMandProp = True Then
491                                 intFit = 1
492                             End If
493                         End If
494                         ' Reset booMandProp for next check
495                         booMandProp = True
496                         ' Compare marine attributes with the desired
properties
497                         If intFit > 0 Then
498                             ' Compare with desired fundamental
499                             If recFDesired.EOF = False Then
500                                 For l = 1 To recFDesired.RecordCount
501                                     If
FundPropTest(recFDesired!FPropertyName_FK, recMarine!SSN_PK) = True Then
502                                         Select Case recFDesired!Level
503                                             Case 1
504                                                 intFit = intFit +
505                                             Case 2
506                                                 intFit = intFit +
507                                             Case 3
508                                                 intFit = intFit +
509                                             Case 4
510                                                 intFit = intFit +
511                                             Case 5
512                                                 intFit = intFit +
513                                             Case 6
514                                                 intFit = intFit +
515                                         End Select
516                                     End If
517                                     recFDesired.MoveNext
518                                 Next l
519                                 recFDesired.MoveFirst
520                             End If
521                             ' Compare with desired logical properties
522                             If recLDesired.EOF = False Then
523                                 For l = 1 To recLDesired.RecordCount
524                                     If
FundPropTest(recLDesired!FPropertyName_FK, recMarine!SSN_PK) = True Then
525                                         Select Case recLDesired!Level
526                                             Case 1
527                                                 intFit = intFit +
528                                             Case 2
529                                                 intFit = intFit +
530                                             Case 3
531                                                 intFit = intFit +
532                                             Case 4
533                                                 intFit = intFit +
534                                             Case 5
535                                                 intFit = intFit +
536                                             Case 6
537                                                 intFit = intFit +
538                                         End Select
539                                     End If
540                                     recLDesired.MoveNext
541                                 Next l
542                                 recLDesired.MoveFirst
543                             End If
544                         End If ' If intFit > 0 Then
545                     End If ' If booPEF = True Then
546                     recMarClsFit.Edit
547                     ' If the marine's PEF is not satisfied by the
school or the marine does not meet
548                         ' all the mandatory properties for the school, a
score of zero is assigned.
549                         ' Otherwise, a value >= 1 is assigned. A score of
 1 indicates the school had no
550                         ' desired properties or the marine did not
satisfy any of the desired properties.
551                         ' Later, 1 is subtracted from these scores and
replaced with a small value called
552                         ' epsilon. The epsilon is used to prevent a
division by zero problem.
553                     recMarClsFit!Fitness = intFit
554                     recMarClsFit.Update
555                     recMarClsFit.MoveNext
556                     ' Prevents counting the entries with zeros since
they are removed later
557                         ' Sums up all the fits and gets a count of how
many fits there are
558                     If intFit > 0 Then
559                         ' The 1 is subtracted from the intFit score
and replaced with a small epsilon
560                         sngSumOfFit = sngSumOfFit + CSng(intFit - 1)
#NAME?
561                         sngCount = sngCount + 1
562                     End If
563                     intFit = 0
564                     recMarine.MoveNext
565                 Next j
566                 ' Prevent dividing by zero
567                 If sngCount > 0 Then
568                     sngAvgFit = sngSumOfFit / sngCount
569                     recMarClsFit.MoveFirst
570                     ' Update the TEMP_FIT table with the normalized
fitness values
571                     For k = 1 To recMarClsFit.RecordCount
572                         If recMarClsFit!Fitness > 0 Then
573                             recMarClsFit.Edit
574                             ' The 1 is subtracted from the stored
fitness score and replaced with a small epsilon
575                             sngFitness = CSng(recMarClsFit!Fitness -
1) + sngEpsilon
576                             recMarClsFit!Fitness =
CInt(sngMandFitValue + sngFitness * intNormalFactor * sngDesiredWeight /
577                             recMarClsFit.Update
578                         End If
579                         recMarClsFit.MoveNext
580                     Next k
581                     ' Insert the TEMP_FIT records into the
582                     Application.SetOption "Confirm Action Queries",
583                     db1.Execute "INSERT INTO MAR_CLS_FIT ( SSN_FK,
MarineIndex, ClassIndex_FK, Fitness ) SELECT TEMP_FIT.SSN_FK,
TEMP_FIT.MarineIndex, TEMP_FIT.ClassIndex_FK, TEMP_FIT.Fitness FROM
TEMP_FIT WHERE TEMP_FIT.Fitness > 0;"
584                     Application.SetOption "Confirm Action Queries",
585                     strClassNumber = recClass!SCourseNumber_FK
586                     strAMOS = recClass!AMOS_FK
587                 End If
588                 recMarine.MoveFirst
589                 If recClass.EOF = False Then
590                     recClass.MoveNext
591                 End If
592             End If ' If recMarine.RecordCount > 0 Then
593         End If ' If (recClass.SCourseNumber_FK = strClassNumber) And
(recClass.AMOS_FK = strAMOS) Then
594         Next i
595     End If '
```

```
4680    596
4681    597  recMarClsFit.Close
4682    598
4683    599  ' Prepare the AMPL rdm1.dat data file
4684    600  Application.SetOption "Confirm Action Queries", False
4685    601  db1.Execute "DELETE * FROM TEMP_FIT;"
4686    602  ' Specify the number of marines
4687    603  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK, MarineIndex) VALUES
4688    ('param totalMarines :=', " & recMarine.RecordCount & ");"
4689    604  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES (';');"
4690    605  ' Specify the number of school classes
4691    606  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK, MarineIndex) VALUES
4692    ('param totalClasses :=', " & recClass.RecordCount & ");"
4693    607  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES (';');"
4694    608  ' Specify the class demand
4695    609  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES ('param demand
4696    610  db1.Execute "INSERT INTO TEMP_FIT ( ClassIndex_FK, Fitness ) SELECT
4697    CLASS.ClassIndex, CLASS.Quota FROM CLASS;"
4698    611  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES (';');"
4699    612  ' Specify the class penalty
4700    613  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES ('param penalty
4701    614  db1.Execute "INSERT INTO TEMP_FIT ( ClassIndex_FK, Fitness ) SELECT
4702    PENALTY.ClassIndex_FK, PENALTY.Penalty FROM PENALTY;"
4703    615  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES (';');"
4704    616  DoCmd.TransferText acExportFixed, "Rdm1_Data Export Specification",
4705    TEMP_FIT, "C:\RDM\AmplData\rdm1.dat"
4706    617
4707    618  ' Prepare the AMPL rdm2.dat data file
4708    619  db1.Execute "DELETE * FROM TEMP_FIT;"
4709    620  db1.Execute "INSERT INTO TEMP_FIT SELECT * FROM MAR_CLS_FIT;"
4710    621  db1.Execute "ALTER TABLE TEMP_FIT DROP COLUMN SSN_FK;"
4711    622  db1.Execute "ALTER TABLE TEMP_FIT ADD COLUMN SSN_FK text;"
4712    623  db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES ('param fitness
4713    default 0:=');"
4714    624  db1.Execute "SELECT * INTO AMPL_TEMP FROM TEMP_FIT ORDER BY SSN_FK
4715    625  db1.Execute "INSERT INTO AMPL_TEMP (SSN_FK) VALUES (';');"
4716    626  DoCmd.TransferText acExportFixed, "Rdm2_Data Export Specification",
4717    AMPL_TEMP, "C:\RDM\AmplData\rdm2.dat"
4718    627  Application.SetOption "Confirm Action Queries", True
4719    628
4720    629  recMarine.Close
4721    630  recClass.Close
4722    631  ' Remove the created tables
4723    632  db1.Execute "DROP TABLE TEMP_FIT;"
4724    633  db1.Execute "DROP TABLE AMPL_TEMP;"
4725    634  db1.Execute "Drop Table TEMP_FIT1;"
4726    635  db1.Close
4727    636  ' Remove the meter from the status bar
4728    637  varRetVal = SysCmd(acSysCmdRemoveMeter)
4729    638
4730    639 End Sub
4731    640
4732    641
4733    642 Public Sub Quota_Penalty(varUDateBound As Variant)
4734    643    Dim varDate
4735    644    Dim rec As Recordset, rec1 As Recordset
4736    645    Dim strSQL As String, strConvert As String
4737    646    Dim i As Integer
4738    647    Dim db1 As Database
4739    648
4740    649
4741    650    ' Create new class quota file
4742    651    Application.SetOption "Confirm Action Queries", False
4743    652
4744    653    ' Delete the old penalty and class files
4745    654    DoCmd.RunSQL "DELETE * FROM PENALTY"
4746    655    DoCmd.RunSQL "DELETE * FROM CLASS"
4747    656
4748    657    ' Query generates new class quota
4749    658    DoCmd.SetWarnings (False)
4750    659    DoCmd.OpenQuery "qryNewQuota"
4751    660    DoCmd.SetWarnings (True)
4752    661
4753    662    ' Delete any values < 0
4754    663    DoCmd.RunSQL "DELETE * FROM CLASS WHERE Quota <= 0;"
4755    664
4756    665    Application.SetOption "Confirm Action Queries", True
4757    666
4758    667    ' Create index for the class table and format the FY for RD3 file
4759    668    Set db1 = CurrentDb()
4760    669    strSQL = "SELECT CLASS.*, SCHOOL.PenaltyFactor FROM SCHOOL INNER JOIN
4761    CLASS ON (SCHOOL.AMOS_PK = CLASS.AMOS_FK) AND (SCHOOL.SCourseNumber_PK =
4762    CLASS.SCourseNumber_FK);"
4763    670    Set rec = db1.OpenRecordset(strSQL, dbOpenDynaset)
4764    671    rec.MoveLast
4765    672
4766    673    If rec.RecordCount > 0 Then
4767    674       rec.MoveFirst
4768    675       For i = 1 To rec.RecordCount
4769    676          rec.Edit
4770    677          rec!ClassIndex = i
4771    678          ' Converts the fiscal year to a two digit number for RD3 file
4772    679          strConvert = CStr(rec!FiscalYear_PK)
4773    680          strConvert = Right(strConvert, 2)
4774    681          rec!FiscalYear_PK = CInt(strConvert)
4775    682          rec.Update
4776    683          rec.MoveNext
4777    684       Next i
4778    685    End If
4779    686
4780    687    ' Create penalty file
4781    688    ' First, update the index from the CLASS table
4782    689    Application.SetOption "Confirm Action Queries", False
4783    690    DoCmd.OpenQuery "qryPenaltyIndex"
4784    691    Application.SetOption "Confirm Action Queries", True
4785    692    ' Create a record set of the PENALTY table
4786    693    strSQL = "SELECT * FROM PENALTY"
4787    694    Set rec1 = db1.OpenRecordset(strSQL, dbOpenDynaset)
4788    695    If IsNull(varUDateBound) = False Then
4789    696       ' Format this date for use in the DateDiff function
4790    697       varUDateBound = Format(varUDateBound, "####/##/##")
4791 Module: modFitnessDetermination    Page: 16
4792    698
4793    699       If rec.RecordCount > 0 Then
4794    700          rec.MoveFirst
4795    701          For i = 1 To rec.RecordCount
4796    702             rec1.Edit
4797    703             ' Get report date from class record
4798    704             varDate = rec!ReportDate
4799    705             ' Format the report date for use in the DateDiff function
4800    706             varDate = Format(varDate, "####/##/##")
4801    707             ' Make the penalty the difference between the two values
4802    708             ' This makes the penalty linear, based on date
4803    709             rec1!Penalty = DateDiff("d", varDate, varUDateBound) *
4804    rec1!PenaltyFactor \ 24
4805    710             rec1.Update
4806    711             rec.MoveNext
4807    712             rec1.MoveNext
4808    713          Next i
4809    714       End If
4810    715    End If
4811    716
4812    717 End Sub
4813    718
4814    719 Public Sub DeleteTable()
4815    720 On Error GoTo DeleteTable_Err
4816    721
4817    722    Dim db1 As Database
4818    723
4819    724    Set db1 = CurrentDb()
4820    725    ' Delete the table if it already exists
4821    726    db1.TableDefs.Delete "PERCENTAGE"
4822    727 DeleteTable_Exit:
4823    728    Exit Sub
4824    729
4825    730 DeleteTable_Err:
4826    731    Select Case Err.Number
4827    732       ' This number occurs if the PERCENTAGE table does not exist.
4828    733       Case 3265
4829    734          Resume DeleteTable_Exit
4830    735       Case Else
4831    736          MsgBox Err.Description
4832    737          Resume DeleteTable_Exit
4833    738          Resume
4834    739       End Select
4835    740
4836    741 End Sub
4837
4838 Module: modImportExportData
4839    Code
4840    1  Attribute VB_Name = "modImportExportData"
4841    2  Option Compare Database
4842    3  Option Explicit
4843    4
4844    5  Public Sub ImportBNA()
4845    6
4846    7     DoCmd.TransferText acImportFixed, "BNAExtract", "qryBNAExtract",
4847    C:\ToBeRdm\RdmData\BNA.txt
4848    8
4849    9  End Sub
4850    10
4851    11 Public Sub ImportRD1()
4852    12
4853    13    DoCmd.TransferText acImportFixed, "RD1", "qryMarine",
4854    C:\ToBeRdm\RdmData\RD1.txt
4855    14
4856    15 End Sub
4857    16
4858    17 Public Sub ImportToTarget_MOS() '(ActionQuery As String)
4859    18
4860    19    CurrentDb().Execute "INSERT INTO TARGET_MOS ( CourseNumber_PK,
4861    TargetMOS_PK, MCC ) SELECT DISTINCT BNA_EXTRACT.CourseNumber_PK,
4862    BNA_EXTRACT.TargetMOS_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT"
4863    20
4864    21 End Sub
4865    22
4866    23 Public Sub Ampl_Result()
4867    24
4868    25    DoCmd.TransferText acImportDelim, "Ampl_Result", "AMPL_RESULT",
4869    C:\ToBeRdm\AmplData\rdm.out, True
4870    26
4871    27 End Sub
4872    28
4873    29 Public Sub AmplData()
4874    30    Dim db1 As Database
4875    31
4876    32    Set db1 = CurrentDb()
4877    33
4878    34    ' Prepare the AMPL rdm.dat data file
4879    35    Application.SetOption "Confirm Action Queries", False
4880    36    db1.Execute "CREATE TABLE CONSTANTS (ConstantName TEXT, IntValue
4881    37    ' Specify the fit constant
4882    38    db1.Execute "INSERT INTO CONSTANTS (ConstantName, IntValue) VALUES
4883    ('param fitConstant :=', " & Forms!frmPrepareAndExecuteSolver!cmbFit &
4884    39    db1.Execute "INSERT INTO CONSTANTS (ConstantName) VALUES (';');"
4885    40    ' Specify the fill constant
4886    41    db1.Execute "INSERT INTO CONSTANTS (ConstantName, IntValue) VALUES
4887    ('param fillConstant :=', " & Forms!frmPrepareAndExecuteSolver!cmbFill &
4888    42    db1.Execute "INSERT INTO CONSTANTS (ConstantName) VALUES (';');"
4889    43    DoCmd.TransferText acExportFixed, "Rdm_Data Export Specification",
4890    CONSTANTS, "C:\ToBeRdm\AmplData\rdm.dat"
4891    44    Application.SetOption "Confirm Action Queries", True
4892    45
4893    46    ' Remove the created table
4894    47    db1.Execute "DROP TABLE CONSTANTS;"
4895    48    db1.Close
4896    49 End Sub
4897    50
4898    51 Public Sub ExportRD3()
4899    52
4900    53    DoCmd.TransferText acExportFixed, "ExportRD3File", "qryRD3File",
4901    C:\ToBeRDM\RdmData\RD3.txt
4902    54    Application.SetOption "Confirm Action Queries", False
4903    55    DoCmd.OpenQuery "qryUpdateAssignmentType"
4904    56    DoCmd.OpenQuery "qryAppendResultToAssignment"
4905    57    Application.SetOption "Confirm Action Queries", True
4906    58
4907    59
4908    60 End Sub
4909    61
4910    62 Public Sub ArchiveData()
4911    63
4912    64    Dim varTodaysDate
4913    65    Dim strSQL As String
4914    66
4915    67    varTodaysDate = Date
4916    68
4917    69    ' Format the date
4918    70    varTodaysDate = Format(varTodaysDate, "yyyymmdd")
4919    71    ' Convert the date into a number
4920    72    varTodaysDate = CLng(varTodaysDate)
4921    73
4922    74    ' Archive the applicable data
4923    75    DoCmd.RunSQL "INSERT INTO MARINE_ASSIGN_ARCHIVE SELECT DISTINCTROW
4924    MARINE.SSN_PK, MARINE.LName, MARINE.Initials, MARINE.Gender,
4925    MARINE.DriverLicense, MARINE.SDI, MARINE.TrainingCompany,
4926    MARINE.MCRD_Code, MARINE.PltNumber, MARINE.Citizenship, MARINE.GradDate,
4927    MARINE.SwimQual, MARINE.Obligation, MARINE.Age, MARINE.Education,
4928    MARINE.HighestMath, MARINE.MentalAbility, MARINE.ClearanceEst,
4929    MARINE.Height, MARINE.Vision, MARINE.ColorPerception, MARINE.EthnicGroup,
4930    MARINE.DLAB, MARINE.EDFT, MARINE.DistrictCode, MARINE.Typing, MARINE.EL,
4931    MARINE.MM, MARINE.CL, MARINE.GT, MARINE.Component, MARINE.PEF,
4932    MARINE.CivilConviction, MARINE.RSN, MARINE.AFADBD,
4933    ASSIGNMENT.CourseNumber, ASSIGNMENT.AMOS, ASSIGNMENT.MCC,
4934    ASSIGNMENT.ClassNumber, ASSIGNMENT.ReportDate, ASSIGNMENT.FiscalYear,
4935    ASSIGNMENT.AssignmentType FROM MARINE INNER JOIN ASSIGNMENT ON
4936    MARINE.SSN_PK = ASSIGNMENT.SSN_FK WHERE ASSIGNMENT.ReportDate < " &
4937    76
4938    77    ' Delete the applicable data from the Marine and Assignment tables
4939    78    DoCmd.RunSQL "DELETE DISTINCTROW MARINE.*, ASSIGNMENT.* FROM MARINE
```

```
4939     78      DoCmd.RunSQL "DELETE DISTINCTROW MARINE.*, ASSIGNMENT.* FROM MARINE
4940     INNER JOIN ASSIGNMENT ON MARINE.SSN_PK = ASSIGNMENT.SSN_FK WHERE
4941     ASSIGNMENT.ReportDate < " & varTodaysDate & ""
4942     79
4943     80      Select Case MsgBox("Delete unassigned Marines from the database?",
4944     vbYesNo + vbQuestion, "Delete Unassigned Marines")
4945     81          Case 6 ' If yes is clicked
4946     82              ' Deletes entries in the Marine table without a corresponding
4947      entry in the Assignment table
4948     83              DoCmd.RunSQL "DELETE DISTINCTROW MARINE.*, ASSIGNMENT.SSN_FK
4949     FROM MARINE LEFT JOIN ASSIGNMENT ON MARINE.SSN_PK = ASSIGNMENT.SSN_FK
4950     WHERE (((ASSIGNMENT.SSN_FK) Is Null));"
4951     84          Case 7 ' If no is clicked
4952     85              ' No action taken
4953     86      End Select
4954     87
4955     88  End Sub
```

LIST OF REFERENCES

[1] USMC, Manpower Management Information Systems Branch, Manpower and Reserve Affairs Department. "Statement of Work for the USMC Manpower Model Modernization: Reengineering of the Enlisted Assignment Model and Recruit Distribution Model," September, 1997.

[2] Decision Support Associates Inc. *Users Manual-Recruit Distribution Model.* Developed In-house, Decision Support Associates Inc., 1993.

[3] C.J. Date. *An Introduction to Database Systems.* Reading, MA: Addison-Wesley, 1992.

[4] J. Dumas. *Designing User Interfaces for Software.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

[5] CPlex, A Division of ILOG. CPlex Home Page, 1998. [http://www.cplex.com].

[6] R.H. Sprague. "A Framework for the Development of Decision Support Systems." *MIS Quarterly* 4, Pages 1-26, 1980.

[7] Logic Works, Inc. "Logic Works BPWin Tutorial Guide," 1994-1995.

[8] E.F. Codd. "A Relational Model of Data for Large Shared Databanks." *Communications of the ACM* 13, June 1970.

[9] R. Smith, D. Sussman. *Beginning Access 97 VBA Programming.* Wrox Press, 1997.

[10] R. Fourer, D. Gay, B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Boyd & Fraser Publishing Company, 1993.

[11] B. Render, R. Stair. *Quantitative Analysis for Management.* Prentice-Hall Inc., 1997.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., Ste 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Professor Hemant Bhargava (Code SM/BH) 7
   Systems Management Department
   Naval Postgraduate School
   Monterey, CA 93940-5000

4. Professor Suresh Sridhar (Code SM/SR) 1
   Systems Management Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Professor Daniel Dolk (Code SM/Dk) 1
   Systems Management Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. LT Kevin J. Snoap 2
   541 Westway Drive N.W.
   Grand Rapids, MI 49544